Eindhoven University of Technology

MASTER

Toepassingen voor en een model van een associatief geheugen

van Loon, J.M.; van Nunen, G.J.W.

Award date:
1975

Link to publication

Disclaimer
This document contains a student thesis (bachelor's or master's), as authored by a student at Eindhoven University of Technology. Student theses are made available in the TU/e repository upon obtaining the required degree. The grade received is not published on the document as presented in the repository. The required complexity or quality of research of student theses may vary by program, and the required minimum study period may vary in duration.

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain

TOEPASSINGEN VOOR EN EEN MODEL VAN EEN ASSOCIATIEF GEHEUGEN

J M van Loon
G J W van Nunen
Th P C Stoffele

Verslag van een afstudeer-onderzoek

uitgevoerd in de vakgroep EB

afstudeerhoogleraar: Prof.ir. A Heetman

datum: 26 juni 1975

A F D E L I N G D E R E L E K T R O T E C H N I E K
T E C H N I S C H E H O G E S C H O O L E I N D H O V E N

Inhoudsopgave

		pag.
	Samenvatting	4
	Summary	5
	Trefwoorden	6
	Voorwoord	7
	Opdracht	8
	Evaluatie van de opdracht	8
	DEEL I	
	Inleiding	10
1.	Geschiedenis en principe	13
2.	Opbouw van het ontworpen associatief geheugen	24
	DEEL II	
3.	Toepassingen voor een associatief geheugen	31
4.	Sorteerprocessen	34
5.	Zoekprocessen	50
6.	Bewerkingen op karakterstrings	55
7.	Matrixbewerkingen	59
8.	Simulatie- en netwerkproblemen	69
9.	Dynamische geheugentoewijzing	81

10. Het opslaan van labels 90

11. Dynamische stack-adressering 95

12. Een associatief werkgeheugen 102

13. Patroonherkenning met een associatief geheugen 108

14. Data management met een associatief geheugen 114

15. Konklusie, nabeschouwing 120

DEEL III

16. Systeemopzet 123

17. Het model 126

18. Basisroutines 151

19. Flowdiagrammen voor het testen van het model 157

Literatuur 170

Appendix I : Programma van de basisroutines

Appendix II: Testprogramma voor het model

SAMENVATTING

Een associatief geheugen (associative memory, content-addressable memory) is een digitaal geheugen, waarin de toegang tot de opgeslagen informatie plaats vindt op grond van de informatie zelf.
Een associatieve geheugenchip met 8 woorden van 8 bits is ontworpen in IIL-techniek, evenals een besturingschip, die acht woorden kan besturen.
Met deze elementen is een associatief geheugen ontworpen met een kapaciteit van 1024 woorden van 80 bits.
Aan het associatief geheugen is een konventioneel half-geleider geheugen (Random Access Memory) gekoppeld.
Het geheel wordt via een besturings-eenheid, uitgevoerd met TTL-circuits, aan een komputer gekoppeld.
Uitgebreid wordt ingegaan op een aantal mogelijke toepassingen voor het associatief geheugen.
Afhankelijk van de toepassingen kan men het geheugen beschouwen als een randapparaat van een komputer, als een geintegreerd onderdeel van een komputersysteem, of als een essentieel onderdeel van een speciaal informatieverwerkend apparaat.
De toepassingen zijn dan ook in drie klassen verdeeld en van elke klasse is een aantal representanten uitgewerkt.
Om tot een goede proces-analyse van de verschillende toepassingen te komen, is een model gebouwd van 8 woorden van 24 bit.
Dit model is gekoppeld aan het MCS-8 microcomputersysteem.
Voor het model zijn basisroutines en een testprogramma geschreven.

SUMMARY

An associative memory is a memory whose registers are not identified by their name or position but by their contents.
An associative memory-chip, which contains 8 words of 8 bits each has been designed.
Also an associative control-chip, which is able to control 8 memory words of any length has been developed.
With these elements an associative memory has been designed.
Its capacity is 1024 words of 80 bits each.
The associative memory together with a random access memory will be connected to a general purpose computer by means of a data-control unit.
The main part of this report deals with applications of associative memories.
Depending on the application, the memory can be considered as an I/O-device of a computer, as an integrated part of a computer system or as a part of a special purpose information-handling system.
The applications are divided into three classes and within each class a number of representatives are described.
In order to get a good analysis of several processes, a model of 8 words of 24 bits each has been built.
The model is connected to the MCS-8 microcomputer system.
A number of basic routines have been written and a test program has been developed.

TREFWOORDEN

associatief geheugen
associative memory
content addressable memory (CAM)
IIL = integrated injection logic
kenmerkregister
maskerregister
besturing van het CAM
data besturing
sorteren
matrixbewerkingen
dynamische geheugentoewijzing
labels
stacks
patroonherkenning
data management
basisroutines
testprogramma
MCS-8 microcomputersysteem

VOORWOORD

Dit verslag is ontstaan naar aanleiding van een onderzoek naar de toepassingsmogelijkheden voor een associatief geheugen. Het onderzoek is door ondergetekenden uitgevoerd als afstudeeropdracht aan de Technische Hogeschool te Eindhoven, Afdeling der Elektrotechniek, vakgroep EB en stond onder leiding van Prof ir A Heetman.

Dit verslag sluit aan op het afstudeerverslag, dat onder de titel 'Het realiseren van een associatief geheugen m.b.v. Injektie-Logica' is geschreven door R G J Dijkstra.

Het verslag beoogt enige klaarheid te brengen in de toepassingsmogelijkheden voor associatieve geheugens, en beschrijft een gebouwd model van een associatief geheugen.
Wij spreken in dit verband liever van een Content Addressable Memory (afgekort tot CAM) als uitsluitend het geheugen wordt bedoeld.
De naam Associatief Geheugen duidt dan aan: het geheel van CAM en bijbehorende besturing, waardoor het geheugen op de juiste wijze een aantal taken kan verrichten, die in dit verslag worden beschreven.

Tijdens onze afstudeerperiode hebben wij hulp en steun ondervonden van velen uit de groep EB.
Vooral van degenen die deel uitmaakten van de associatieve werkgroep, waarin problemen met betrekking tot het associatief geheugen werden besproken.
In het bijzonder willen wij noemen onze afstudeerhoogleraar Prof ir A Heetman, die steeds met suggesties voor nieuwe toepassingen klaar stond, en die ons enthousiasme voor het onderwerp wist bij te brengen.

Wij hopen, dat dit verslag een goede handleiding zal zijn voor de definitieve versie van het associatief geheugen en dat het een belangrijke bijdrage zal leveren tot een beter begrip van de mogelijkheden en de beperkingen van associatieve geheugens.

Eindhoven, juni 1975

J M van Loon
G J W van Nunen
Th P C Stoffele

OPDRACHT

Het bestuderen van de mogelijkheden van een besturing voor associatieve geheugens en het realiseren van deze besturing.

EVALUATIE VAN DE OPDRACHT

Er werd begonnen met het zoeken naar toepassingen, waarvoor het gebruik van een associatief geheugen zinvol leek. De literatuur verschaft een redelijke hoeveelheid informatie over sorteerprocessen met behulp van een associatief geheugen. Over andere toepassingen voor associatieve geheugens is nog weinig gepubliceerd.

Uit de bijeenkomsten van de zogenaamde associatieve werkgroep, die in september 1974 werd gevormd, kwamen een aantal suggesties en ideeen naar voren.
Een aantal daarvan zijn in dit verslag verder uitgewerkt. Daarnaast is teruggegrepen op enkele algemene overzichtsartikelen (zie literatuurlijst) voor een aantal toepassingen. De meeste van de in de literatuur voorgestelde toepassingen zijn nooit werkelijk uitgevoerd met een associatief geheugen, voornamelijk omdat een geheugen niet beschikbaar was.
Voor allerlei processen is nagegaan wat voor voordelen het gebruik van een associatief geheugen biedt.
Tegelijkertijd werd een literatuurlijst aangelegd en in de loop van de tijd uitgebreid.

Door C Stork werd een nieuwe associatieve chip ontwikkeld, die 8 woorden van 8 bits elk bevat; de vorige chip bevatte 4 woorden van 10 bits.
In onderling overleg werd besloten om de besturingchip zo te wijzigen, dat het geheugen ook beschrijfbaar is op adres.
Dit biedt voor bepaalde processen duidelijke voordelen zo is uit het onderzoek gebleken.
Het expliciet uitlezen van het CAM is niet mogelijk bij de ontworpen chip; het zou echter een belangrijk winstpunt zijn voor een aantal processen als dit wel mogelijk was.
Door het via de besturing gekoppelde RAM kan dit probleem echter worden ondervangen.

Nadat enkele processen waren onderzocht, werd een model van 8 woorden van 24 bits elk gebouwd om enige ervaring op te doen met associatieve geheugens en de werking ervan. Het model is via de Data Besturing gekoppeld aan het aanwezige MCS-8 microcomputersysteem.
Er zijn een aantal basisroutines geschreven en een testprogramma, waarmee de juiste werking van het associatief geheugen kan worden gekontroleerd.

DEEL I

INLEIDING

Een associatief geheugen (associative memory, content-addressable memory) is een digitaal geheugen, waarin de toegang tot de geheugeninformatie plaats vindt op grond van de informatie zelf.
Dit in tegenstelling tot een konventioneel geheugen (random access memory), waarin de informatie toegankelijk is op grond van een adres, dat de plaats aangeeft waar de informatie is opgeslagen.
Door het toepassen van een associatief geheugen kan bij een aantal processen een belangrijke snelheidswinst worden verkregen.
Daarentegen is een associatief geheugen ingewikkelder van opbouw dan een konventioneel geheugen en daardoor ook kostbaarder. Dat is dan ook een van de redenen waarom associatieve geheugens tot nu toe slechts op zeer beperkte schaal en met een betrekkelijk kleine geheugenkapaciteit zijn toegepast, meestal in experimentele systemen.
Door de opkomst van de integratietechnieken is het echter mogelijk geworden om geheugen-chips te maken met een tamelijk groot aantal bits per chip.

Een andere reden voor de beperkte toepassing is de onbekendheid met de mogelijkheden van een associatief geheugen.
Aan dit laatste komt dit afstudeerverslag tegemoet.
Er wordt ingegaan op de algemene opbouw van een associatief geheugen. De opbouw van een dergelijk geheugen, zoals dat gerealiseerd gaat worden zal worden besproken, evenals de besturing voor het geheugen.
Voor de geheugen- en de besturings-chips is gebruik gemaakt van de IIL-techniek.
Het grootste deel van dit verslag geeft een overzicht van de mogelijke toepassingen van een associatief geheugen; vooral die toepassingen, waarbij het gebruik van een associatief geheugen een belangrijke tijdsbesparing oplevert t.o.v het gebruik van een konventioneel geheugen.
Het hier beschouwde geheugen(AG) bestaat uit een associatief gedeelte (CAM = Content Addressable Memory) met besturing (BC = Besturing CAM) en een konventioneel gedeelte (RAM = Random Access Memory).
Deze kombinatie vormt het totale associatieve geheugen(AG).
De manipulaties met gegevens en adressen ten behoeve van de processen, die het associatief geheugen moet uitvoeren worden verzorgd door de zogenaamde Data Besturing (DB), uitgevoerd in TTL, die tevens als interface dienst doet tussen het associatief geheugen en de komputer waaraan het geheugen is gekoppeld.

De verwerking van data in een komputer kan bestaan uit een aantal operaties, zoals ordenen, sorteren, zoeken, enz.
Bij een konventioneel geheugen moet woord voor woord worden uitgelezen en buiten het geheugen met de gegeven zoek- of sorteerkriteria worden vergeleken.

Bij een aantal N opgeslagen woorden (ongeordend) zijn voor het zoeken N geheugentoegangen en vergelijkingen noodzakelijk; voor het ordenen ongeveer N toegangen en vergelijkingen, en voor het sorteren nog meer.
Dit alles kost veel tijd. Bovendien zijn er omvangrijke programma's nodig voor de komputer en vaak heeft men nog extra ruimte in het geheugen nodig om tussentijdse gegevens op te slaan.

Bij het gebruik van een associatief geheugen vindt de toegang tot de afzonderlijke woorden niet meer plaats op grond van hun adres, maar op grond van de opgeslagen informatie zelf.

Om een bepaald woord in het geheugen te vinden wordt een kenmerk (sleutel, sleutelwoord, keyword) in een kenmerkregister (vraagregister, vergelijkingsregister) opgeslagen.
De inhoud van dit register wordt gelijktijdig met alle woorden die in het geheugen aanwezig zijn vergeleken, en alleen het woord, waarvan de inhoud op de vastgestelde plaatsen met de registerinhoud overeenstemt geeft een positieve reaktie.
We behoeven echter geen volledig kenmerk aan te bieden; het is ook mogelijk een deelkenmerk aan te bieden. Daaronder wordt verstaan een kenmerk, waarvan een aantal bits don't cares zijn.
Het vormen van don't cares wordt maskeren genoemd en gebeurt met behulp van een maskerregister, ook wel maskeerregister genoemd.
Het kan voorkomen, dat meerdere woorden aan het kenmerk voldoen en tegelijkertijd reageren (multiple response).
De besturing van het associatieve geheugen dient hierin te voorzien.

Het ordenen en sorteren is met een associatief geheugen ook eenvoudiger, omdat een zoekaktie 'parallel' wordt uitgevoerd over alle woorden,zodat het doorzoeken van het gehele geheugen in een zoekcyklus kan worden voltooid, onafhankelijk van het aantal woorden dat in het geheugen is opgeslagen.

Vanaf het ontstaan van de eerste informatieverwerkende apparaten is het steeds zo geweest dat de snelste geheugens langzamer werkten dan de snelste logische schakelingen.
Bij de keuze van een geheugen moet in de meeste gevallen een kompromis worden gesloten tussen de snelheid en de kosten, afhankelijk van de gewenste geheugenruimte.
Anderzijds zijn er een groot aantal problemen, waarvan de verwerking steeds de hoogst mogelijke snelheid van de informatieverwerkende apparatuur vereist.

De ontwikkeling van de geheugentechniek in de associatieve richting is een noodzakelijke stap om de snelheid en het prestatievermogen van de komputer nog verder te verbeteren.

In principe bestaat het onderscheid tussen een konventioneel geheugen en een associatief geheugen daarin, dat in een associatief geheugen ieder bitelement (iedere cel) naast de mogelijkheid tot beschrijven, opslaan en uitlezen, ook nog de mogelijkheid tot vergelijken in zich heeft.

Dit brengt dus een uitbreiding van de logika met zich mee ten opzichte van een konventioneeel geheugen.

Men kan een associatief geheugen vooral goed gebruiken in die gevallen, waarin vergelijken een grote rol speelt.

Bij veel administratieve informatieverwerking worden met grote hoeveelheden informatie slechts eenvoudige reken-operaties uitgevoerd. Het opzoeken van data is daarbij een belangrijke en tijdrovende zaak, waarbij de voordelen van het gebruik van een associatief geheugen duidelijk naar voren komen. Dergelijke administatieve taken vragen een zeer grote geheugenkapaciteit; iets wat nu nog niet in de vorm van een associatief geheugen te realiseren is, alleen al niet vanwege de kosten.

Het associatieprincipe van het geheugen maakt het mogelijk dat men informatie in het geheugen opslaat of uitleest, zonder dat men de plaats van de informatie weet.
Dit opent nieuwe mogelijkheden, zowel voor het programmeren als voor de organisatie van informatieverwerkende apparaten.

Ook een associatief geheugen heeft zijn beperkingen; het is zeker niet voor iedere toepassing het meest geschikte geheugen.

Het associatief geheugen kan in een systeem die taken uitvoeren, waarbij een duidelijke tijdswinst wordt verkregen ten opzichte van het gebruik van een konventioneel geheugen.

1 GESCHIEDENIS EN PRINCIPE

1.1 Geschiedenis

Associatieve geheugens zijn bekend onder verschillende benamingen zoals: catalog memories, (parallel) search memories, content-addressable memories, data-addressed memories, tag memories etc.

De International Federation of Information Processing Society Glossary definieert een associatief geheugen als een geheugen, waarvan de woorden niet geidentificeerd worden d.m.v.naam of plaats maar door middel van hun inhoud.

De eerste publicatie betreffende een associatief geheugen dateert van 1956, toen Slade en McMahon een 'catalog memory' voorstelden, opgebouwd met cryogene elementen. De belangstelling voor dit soort elementen nam snel toe. Ontwikkelingen op het gebied van magnetische films en de technologie van de ferrieten leidden tot experimenten met niet-cryogene elementen, waarmee belangrijke ervaringen op het gebied van associatieve geheugens werden opgedaan. Recente ontwikkelingen op het gebied van de large scale integration (LSI) hebben hernieuwde belangstelling gewekt voor associatieve geheugens.

Verschillende ontwerpen voor bipolaire - en MOS-circuits zijn de laatste jaren voorgesteld.
Zie daarvoor referenties: 19, 28, 38, 42, 68, 81.
De laatste ontwikkelingen op het gebied van LSI wat betreft random access geheugens geven een indicatie dat CAM-geheugens ook gemaakt kunnen worden met een groot aantal bits per chip. Het aantal bits (cellen) per chip is bij associatieve chips niet zozeer de beperkende faktor, maar eerder het aantal aansluitpunten per chip.

De geringe interesse die tot nu toe voor associatieve geheugens bestaat is onder andere het gevolg van:
- het niet of nauwelijks verkrijgbaar zijn van associatieve chips
- de onbekendheid met de toepassingsmogelijkheden van een associatief geheugen
- de onbekendheid met de implicaties op het gebied van de programmatuur, die een associatief geheugen met zich meebrengt.

In de afgelopen jaren is veel aandacht besteedt een de hardware aspekten van grote en kleine associatieve geheugens en associatieve processors.
Kleine associatieve geheugens hebben toepassing gevonden bijvoorbeeld in 'paging boxes' voor virtuele geheugensystemen.

De enige grote associatieve processor, die kommercieel verkrijgbaar is, is de STARAN S, geintroduceerd door de Goodyear Aerospace Corporation in 1972.
STARAN is economisch in te zetten voor toepassingen, die het real-time processen van zeer veel gegevens vereisen, zoals leiding van luchtverkeer en het verwerken van radargegevens.

1.2 Basisopbouw van een associatief geheugen

Allereerst zal nu een associatief geheugen worden gedefinieerd. De basiselementen, waaruit het is opgebouwd zijn:

1. De geheugenmatrix - deze verzorgt de opslag van data.

2. Het kenmerkregister (vergelijkingregister, comparand register) - dit bevat de data, die met de geheugeninhoud vergeleken moeten worden voor zoek-operaties.
   Het register kan dienst doen bij de data-overdracht tussen de geheugenmatrix en de general purpose komputer.

3. Het maskerregister - dit bevat data, die specificeert, welke delen van het woord in het kenmerkregister meedoen bij een operatie.

4. Het bezetregister - dit is een register met evenveel plaatsen als er woorden in het geheugen zijn.
   Een 1 op een plaats in het bezetregister geeft aan dat het bijbehorende woord in het geheugen beschreven is.

5. Het associatieregister - dit is een register met evenveel plaatsen als er woorden in het geheugen zijn.
   Op de plaats waar een associatie optreedt, krijgt het bijbehorende bit in dit register de waarde 1.

6. De beslissings-logica - deze logica bepaalt de plaats van de bits van het associatie-register die 1 zijn geworden.

7. De besturings-logica - deze logica zorgt ervoor dat de commando's die het geheugen krijgt aangeboden, op de juiste wijze uitgevoerd worden.

Fig 1.1 geeft de basisstruktuur van een associatief geheugen weer. We onderscheiden de geheugenmatrix; verder het bezetregister, het associatie-register en de beslissingslogika. Deze laatste drie samen worden ook wel de besturing van het CAM genoemd. De besturingslogika dient voor het wijzigen van de bits van masker- en kenmerk-register en als interface tussen het geheugen en de general purpose komputer.

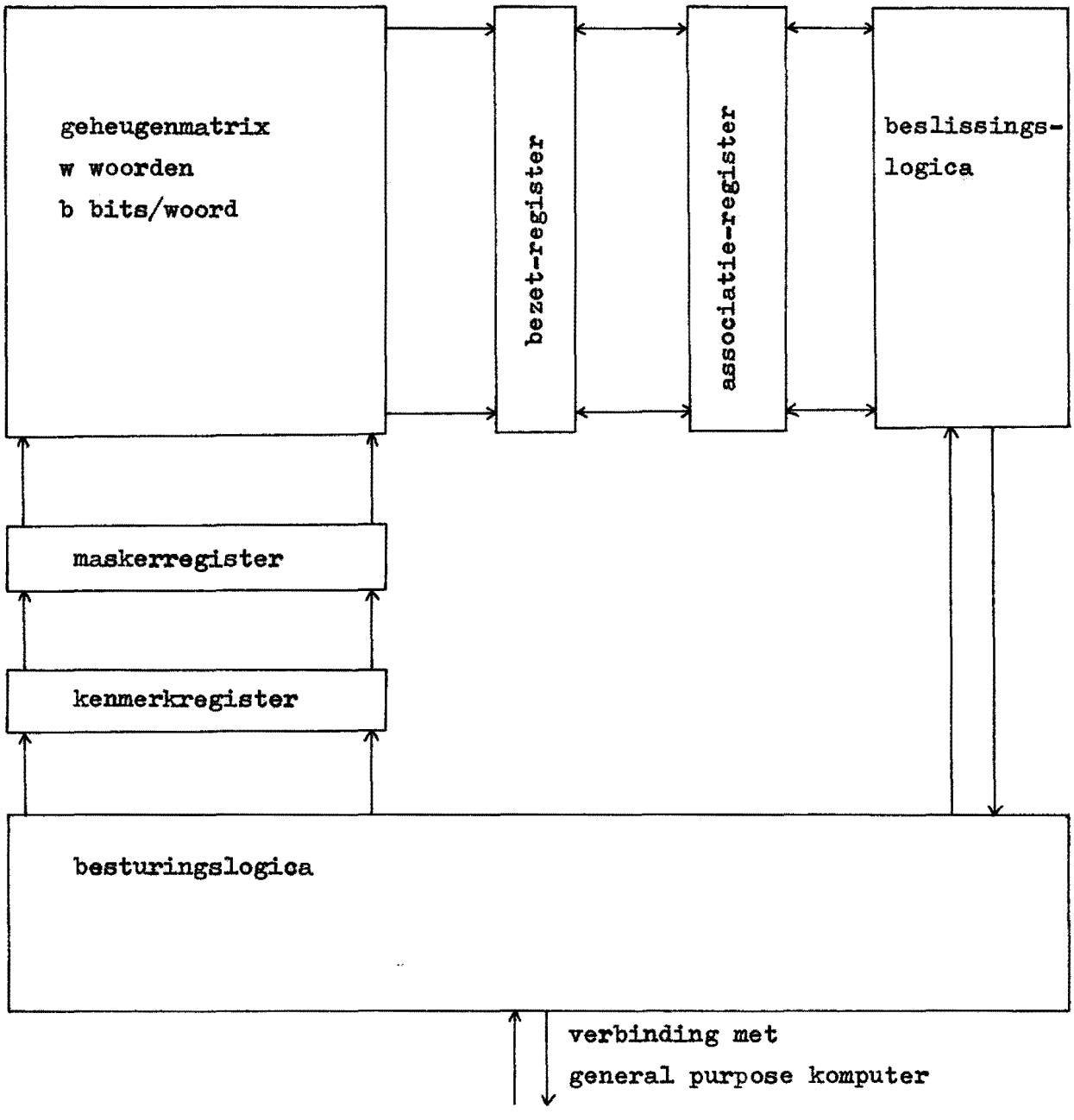


Fig 1.1 Basisstruktuur van een associatief geheugen

1.3 Onderscheid van associatieve geheugens in verschillende typen, afhankelijk van de werkingswijze.

Men kan 5 verschillende typen associatieve geheugens onderscheiden, gelet op de werkingswijzen van het ondervragingsmechanisme.
De 5 typen zijn in fig 1.2 aangegeven.

1. Een herkenningsgeheugen.
   Als alle bitplaatsen van het geheugen vergeleken dienen te worden om een associatie te verkrijgen, dan noemt men het betreffende geheugen een herkenningsgeheugen (recognition, catalog, identifying memory).
   Dit is de eenvoudigste opzet voor een associatief geheugen.

2. Een vast-label geheugen.
   Als het geheugen een associatief deel en een data-deel heeft en alle bitplaatsen van het associatieve deel vergeleken dienen te worden met het kenmerk om een associatie te verkrijgen, dan wordt het geheugen een vast-label geheugen genoemd (fixed tag memory).

3. Een partieel-label geheugen.
   Als het geheugen een associatief deel heeft en een data-deel en het vergelijkingskriterium mag betrekking hebben op een willekeurige kombinatie van bits in het associatieve bereik dan noemt men een dergelijk geheugen een partieel-label geheugen (partial tag memory).

4. Een volledig associatief geheugen.
   Als alle bitplaatsen van het geheugen met het vergelijkingskriterium vergeleken kunnen worden, en elke kombinatie van bits kan als vergelijkingskriterium dienen, dan wordt het geheugen een volledig associatief geheugen genoemd (fully associative memory).

5. Een hybride associatief geheugen.
   Dit bestaat uit een volledig associatief geheugen met daaraan gekoppeld een datagedeelte in de vorm van een RAM.
   Deze opbouw is in fig 1.2 onder 5 weergegeven.
   Het data-deel kan rechtstreeks gekoppeld zijn aan het associatieve deel.
   Een betere oplossing is om de plaats van een associatief woord te koderen en het adres van het betreffende woord door te geven aan het data-deel.
   De voordelen daarvan zijn:
   - we hebben het adres van het woord dat associeert ter beschikking, ook voor andere doeleinden dan het adresseren van een woord in het konventioneel geheugen.
   - we kunnen met een gegeven adres een woord uit het data-deel uitlezen, zonder dat we gebruik behoeven te maken van het associatieve deel.

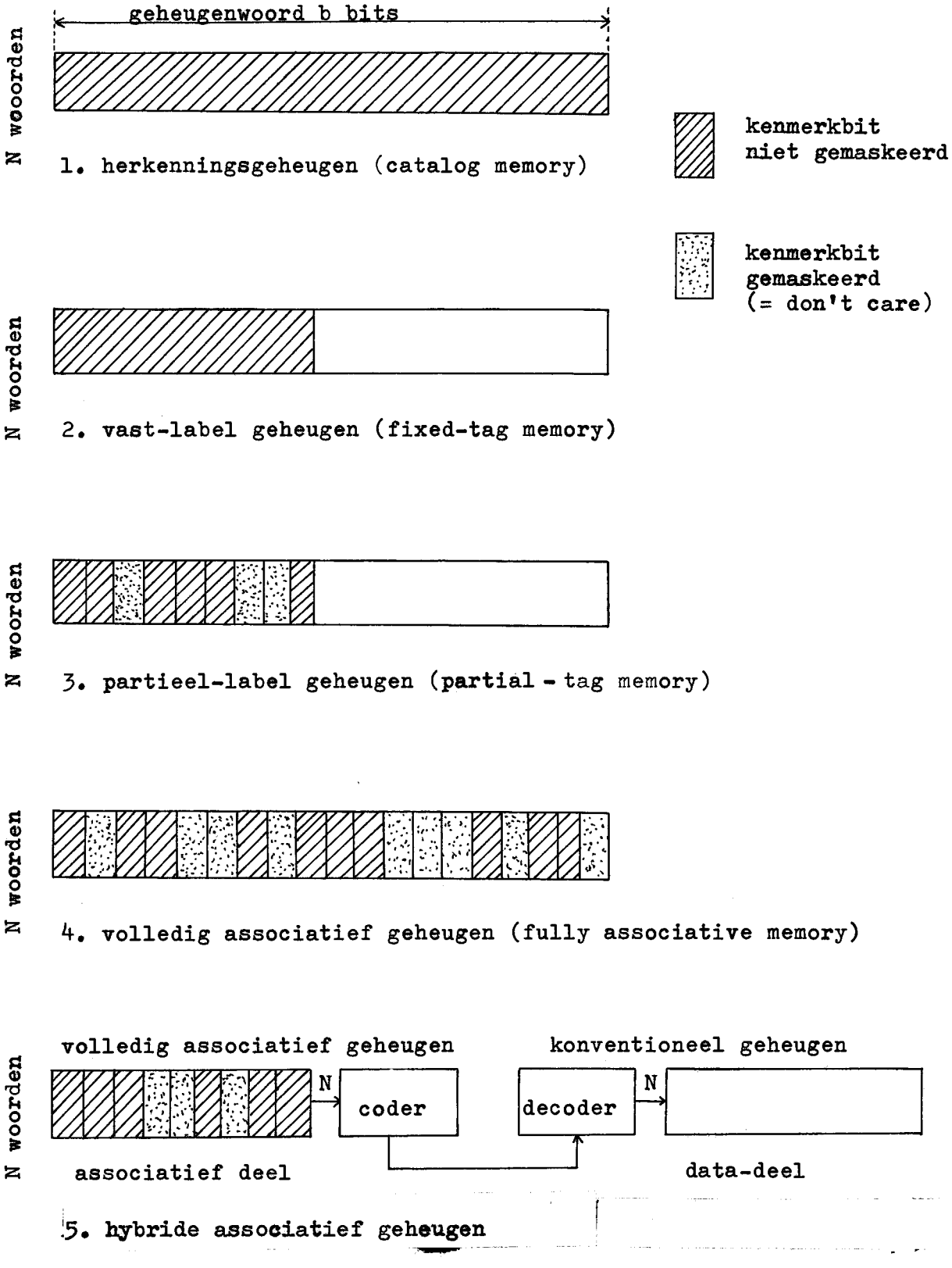


fig 1.2 Onderscheid van associatieve geheugens naar type en werkingswijze.

1.4 Eisen voor de organisatie

De vraag of een associatief geheugen voordelen biedt voor een specifiek probleem, geeft aanleiding tot de volgende beschouwingen.

- configuratie: er is een grote varieteit aan manieren waarop een associatief geheugen aan een general purpose komputer gekoppeld kan worden.
Bijvoorbeeld: het associatief geheugen kan zich aan de komputer voordoen als een I/O-device, of als een deel van het konventioneel geheugen.
Beschouwingen omtrent de wijze waarop het associatief geheugen en de general purpose komputer met elkaar worden verbonden zijn van uitermate groot belang voor de effectiviteit van het systeem.

- de besturing van het CAM met het associatie-register.
Het associatie-register heeft de grootte van het aantal woorden van het geheugen. Het bevat voor de aanvang van de ondervraging enkel nullen.
Na de ondervraging komen er enen op de plaats waar associaties optreden.
De mogelijkheid van het associatief geheugen om resultaten van zoekprocessen op te slaan en om operaties op verzamelingen van die resultaten uit te voeren, hangt af van de opbouw van het associatie-register en de bijbehorende logica.
Deze besturing is een belangrijke niet te verwaarlozen kostenfaktor bij de opbouw van een associatief geheugen.

- eisen voor het laden (beschrijven) en het uitlezen van het associatief geheugen.
Een bepaald probleem zal vereisen dat het associatief geheugen regelmatig wordt geladen en/of uitgelezen.
De voordelen van een associatief geheugen kunnen volledig teniet worden gedaan, als het laden en uitlezen zeer vaak moet gebeuren. Dus de eisen voor snel laden en snel uitlezen zijn basisgegevens bij de beslissing of een associatief geheugen zinvol is in een bepaald geval of niet.

Deze drie punten hangen nauw met elkaar samen.
Een probleem bij het ontwikkelen van associatieve geheugens is: hoe kunnen we voldoende associatieve geheugens operationeel krijgen, zodat programmeurs er een optimaal gebruik van kunnen maken, waardoor de vraag naaar dergelijke geheugens zal toenemen, en de ontwikkeling een impuls krijgt.

1.5 Varianten van associatieve geheugens

Er zijn talrijke varianten van associatieve geheugens. Men kan daarbij verschillende aspekten onderscheiden:

1. Logische organisatie:
   de variaties reiken van associatieve geheugens met minimale besturing tot eenheden die onafhankelijk zijn van de centrale processor.
2. Interface met de centrale processor:
   de variaties reiken van een opzet met geen wijzigingen in de centrale processor tot volledige integratie van het associatief geheugen met de general purpose komputer.
3. Instructie-set:
   de variaties reiken van een minimale verzameling tot een verzameling instructies met gekompliceerde zoek- en manipulatiemogelijkheden.

De volgende typen associatieve geheugens kunnen worden onderscheiden:

1. Een associatief geheugen met een minimum aan besturingshardware, dat via een datakanaal verbonden is met een general purpose komputer.
2. Een associatief geheugen met volledige parallele hardware, verbonden met de general purpose komputer via een DMA-kanaal.
3. Een associatief geheugen met volledige parallele hardware, geintegreerd met de general purpose komputer, dat wil zeggen, dat de laatste w woorden van het geheugen van de general purpose komputer zijn vervangen door het associatief geheugen.

1.6 Realisaties van associatieve geheugens

Er zijn verschillende associatieve geheugens gebouwd en getest. De meeste hadden een zeer beperkte kapaciteit en waren opgebouwd met ferrietkernen of cryogene elementen.
Verschillende firma's hebben geexperimenteerd met associatieve geheugens voor het verwezenlijken van bepaalde besturingsfunkties in konventionele komputers.

Een klein associatief geheugen van 64 woorden van 16 bit werd experimenteel opgenomen in een IBM 360/40 komputer om de geheugenruimte te regelen, die beschikbaar was voor time-sharing gebruikers.

De Burroughs komputer, die de Illiac IV parallel processor bestuurt, bevat een associatief geheugen van 64 woorden.
Het geheugen vervult hier een 'instruktie-look-ahead' funktie.

Control Data Corp heeft een klein associatief geheugen aangekondigd ten behoeve van de interne boekhouding van een konventionele komputer.

Het grootste associatieve geheugen, tot nu toe bekend, is ontwikkeld in 1968 door de Goodyear Aerospace Corp ten behoeve van het USAF Rome Air Development Centre.
Het is een 2k geheugen met woorden van 50 bits, waarvan 48 databits, 1 busybit en 1 paritybit.
Het geheugen is opgebouwd met ferrietkernen en kan 11 verschillende zoekprocessen uitvoeren.
Het associatief geheugen werd in eerste instantie als randapparaat gekoppeld aan een CDC-1604B komputer.
Later zijn verbeteringen aangebracht in deze opzet, want de I/O-subroutines hadden meer tijd nodig voor data-overdracht tussen komputer en associatief geheugen dan voor de verwerking in het geheugen zelf nodig was.
Het geheugen is nu via een DMA-kanaal verbonden met het kerngeheugen van de 1604B komputer, waardoor snelle informatieuitwisseling mogelijk is.
Het associatief geheugen bestuurt dit kanaal.

Met een enkele instruktie kan een van de zoekprocessen worden uitgevoerd.
Het associatief geheugen bevat hardware, die een tamelijk onafhankelijk opereren van het geheugen toelaat.
Er is een programmateller, instruktieregister, 3 indexregisters, en verschillende andere registers.
Instrukties, kenmerken en maskers worden verkregen uit het kerngeheugen van de komputer via het DMA-kanaal.
Het associatief geheugen is afhankelijk van het kerngeheugen voor de opslag van alle instrukties, kenmerken en maskers.
De operaties, verricht door het associatief geheugen worden geinitialiseerd door de komputer en het geheugen verricht verder autonoom zijn werk, totdat de komputer weer moet bijspringen. De 1604B kan de status van het associatief geheugen nagaan, dat wil zeggen, of het geheugen aan het werk is of niet.

Er is een interrupt-mogelijkheid die resulteert in een interrupt van de komputer zodra het associatief geheugen een opgedragen taak heeft uitgevoerd.
Het geheugen is direkt adresseerbaar voor enkele 'load'- en 'unload' instrukties.
Het is inhouds-adresseerbaar met betrekking tot zoekinstrukties en voor andere instrukties die gebruik maken van de resultaten van eerder uitgevoerde zoekprocessen.

Het associatief geheugen bevat:
- een adresregister dat het startadres van het geheugen bevat
- een kenmerkregister, dat wordt gebruikt in zoekprocessen en bij 'load'-operaties
- een maskerregister, dat het masker bevat
- een programmateller
- een instruktieregister
- drie indexregisters
- een woord-tel-register dat wordt gebruikt bij 'load'- en 'unload' instrukties
- een schuif-tel-register, dat het aantal plaatsen aangeeft waarover de inhoud van het kenmerkregister geschoven moet worden

De tijd benodigd voor zoekprocessen varieert met de komplexiteit van de zoekprocedures, het aantal bits dat bekeken moet worden en of wel of niet gemaskeerd moet worden.

In par 4.7 wordt een sorteerproces besproken dat met de genoemde konfiguratie uitgevoerd kan worden.
Fig 1.3 geeft de opbouw van het RADC Associatieve Geheugensysteem.
Voor meer details over het Goodyear Associatieve Geheugen wordt verwezen naar de referenties: 77,104.

De Goodyear Aerospace Corporation heeft de eerste en tot nu toe enige kommercieel verkrijgbare associatieve processor ontworpen.
Het is de STARAN S Associatieve Processor (de S duidt op solid-state), die sinds 1972 op de markt is.
STARAN S kan zoekoperaties, rekenkundige operaties, logische operaties en I/O-operaties op vele datasets simultaan uitvoeren.
Fig 1.4 geeft de architectuur van de STARAN S associatieve processor.
Men kan zelf het aantal arrays kiezen (tussen 2 en 32) dat men wil gebruiken.
Als de associatieve processor gekoppeld wordt aan een konventionele processor dan zal een interface vereist zijn.

STARAN S is opgebouwd uit de volgende eenheden:
1. Associative arrays die bestaan uit modulen van 256 woorden bij 256 bit.
   Maximaal kunnen 32 associatieve arrays worden gebruikt.
2. Associative Processor (AP) Control, die manipulaties met data uitvoert in de associatieve arrays.
3. AP Control Memory, dat de AP besturingsinstrukties bevat en dienst doet als buffer tussen de AP besturing en de andere elementen van het systeem.
4. Program Pager, die programmasegmenten transporteert naar snelle geheugens.
5. Sequential Controller en geheugen, dat randapparaten bestuurt en voorziet in een communicatie met de operator.
6. External Function Logic, die besturings-informatie naar de STARAN elementen transporteert.

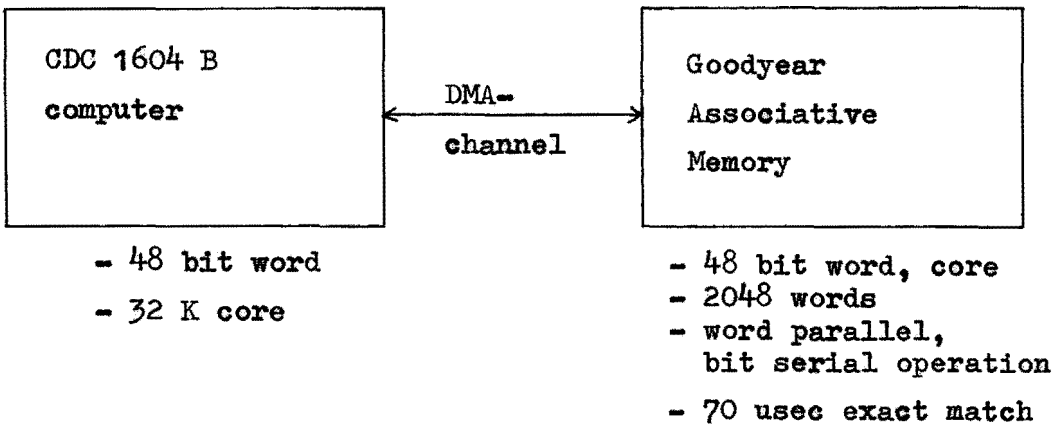


Fig 1.3 Opbouw van het RADC associatief geheugen systeem

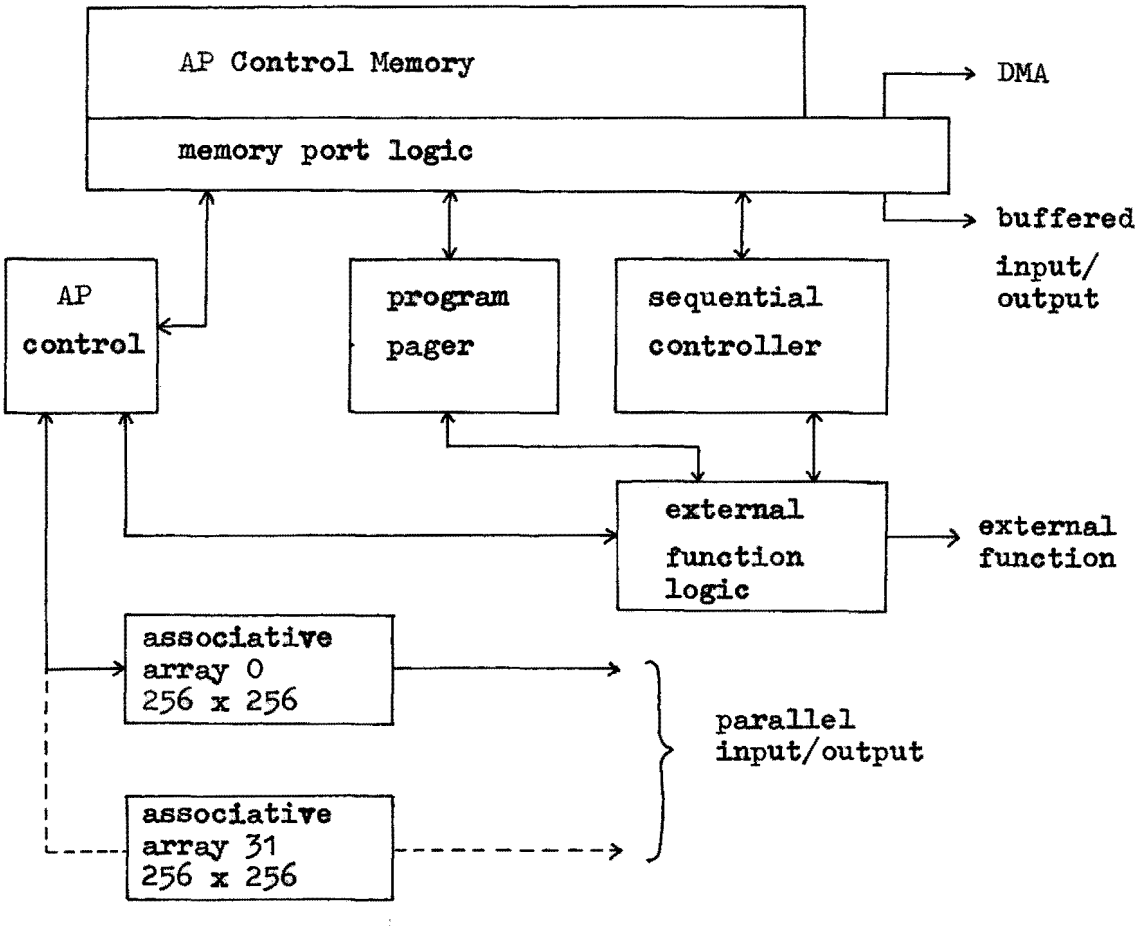


Fig 1.4 Architectuur van de STARAN S associatieve processor

De instruktieset bestaat uit zo'n 100 instrukties, die kunnen worden onderverdeeld in de volgende vier typen:
1. Word addressing
2. Immediate addressing
3. Data pointer addressing
4. Array addressing

Om een bit van het geheugen te vergelijken met het overeenkomstige bit in het kenmerkregister voor een kleiner dan, groter dan of een exact match kost 0.15 usec.
Een exact match (associatie) kost 0.15x32=4.8 usec voor een breedte van 32 bit.
Het uitvoeren van een optelling tussen bits van verschillende velden van een woord kost 0.85 usec per bit.

De referenties 69, 75, 86, 87, 88, 93, 94 geven uitgebreide informatie over de opbouw, de hardware - en de software-aspekten van de STARAN-S.

2 OPBOUW VAN HET ONTWORPEN ASSOCIATIEF GEHEUGEN

Het ontworpen associatief geheugen zal in het kort worden besproken. Voor een gedetailleerde beschrijving van de opbouw van het geheugen wordt verwezen naar het verslag hierover door R G J Dijkstra; zie referentie 1.
Als integratietechniek werd gekozen voor Integrated Injection Logic (IIL) van Philips.
Nadat in eerste instantie een IC werd ontworpen met 40 geheugencellen (4 woorden van 10 bit elk) per chip, werd later een nieuwe geheugenchip ontworpen met 64 geheugencellen, namelijk 8 woorden van 8 bit elk. Deze laatste chip is gunstiger wat pin per cell ratio betreft.

Er zijn per chip nodig 8x2+8x2+4=36 aansluitingen en wel:
- 16 associatielijnen
- 8 leeslijnen
- 8 commandolijnen
- 4 voedingsaansluitingen

De funktie van deze lijnen wordt verderop verklaard.
Het principe van de opbouw van de chip is in fig 2.1 weergegeven.

Een cel heeft de volgende funkties:
a. Een aangeboden logische 0 of 1 moet na een schrijfcommando onthouden worden
b. Zolang niet weer een schrijfcommando gegeven wordt, mag de geheugeninhoud van deze cel niet meer veranderen
c. De cel moet aangeven of de aangeboden logische 0 of 1 overeenstemt met de logische inhoud van de cel, we zeggen dan dat deze cel associeert
d. De cel moet kunnen associeren onafhankelijk van de inhoud van de cel. Aan de cel wordt dan een don't care aangeboden.

Doordat we, zoals onder d is aangegeven ook een don't care moeten kunnen aan bieden, naast een logische 0 of 1, hebben we 2 draden per cel nodig om deze drie toestanden te kunnen aangeven.
Dit zijn de twee associatielijnen die in fig 2.1 zijn aangegeven.
Door middel van deze associatielijnen wordt het kenmerk aan het geheugen aangeboden.

Kenmerkbit	associatielijn A	B
0	0	1
1	1	0
don't care	1	1

Naast de 'vertikale' associatielijnen zijn er per woord nog twee 'horizontale' lijnen, namelijk een commandolijn en een leeslijn.

De commandolijn moet hoog gemaakt worden als een woord op die plaats geschreven moet worden.
De leeslijn wordt laag als een of meerdere bits van het betreffende woord niet associeren; als alle bits associeren en dus het woord associeert, dan is de leeslijn hoog.
Met de genoemde chips van 8 woorden en 8 bits per woord is een associatief geheugen ontworpen van 1024 woorden van 80 bits per woord.
Voor deze geheugengrootte zijn 1280 associatieve chips nodig.

De overwegingen die geleid hebben tot de keuze van deze geheugengrootte zijn de volgende:

1. Een woordbreedte van 80 bits laat toe, dat er 10 alfanumerieke tekens per woord kunnen worden opgeslagen. Dit is een redelijk aantal bij bewerkingen op karakterstrings. Verder kan een woord van 10 bytes voor allerlei doeleinden worden opgedeeld in een aantal velden, bijvoorbeeld bij het opslaan van matrices in het associatief geheugen.
2. De keuze van het aantal woorden in het geheugen is een kompromis tussen een redelijke geheugengrootte aan de ene kant en de kosten en de fysische grootte van het geheugen aan de andere kant.

Het geheugen van 1024 woorden van 80 bits opgebouwd met de 1280 associatieve chips zal verder met CAM (Content Adressable Memory) worden aangeduid.
We zullen een nieuw teken invoeren voor de aanduiding van het CAM, namelijk het teken ⋕, dat is afgeleid uit het teken #, dat wordt gebruikt voor het aanduiden van een RAM (Random Access Memory), en de letter A (van associatief).

Naast het geheugen zelf is er voor de juiste werking nog een hoeveelheid besturingslogica nodig.
Deze besturingslogica is tamelijk omvangrijk, en dat is de reden dat ook deze besturing in IIL-techniek geintegreerd zal worden.

Deze besturing, die we met BC (besturing CAM) zullen aanduiden, heeft de volgende taken:

1. Bijhouden welke woorden geschreven zijn.
2. Bij een schrijfopdracht het woord in het kenmerkregister schrijven.
3. Bij een wisopdracht, alle woorden die associeren wissen.
4. Bij een resetopdracht het woord met het opgegeven adres wissen.
5. Het adres van het woord dat associeert produceren.
6. Indien meerdere woorden associeren, het adres produceren van het woord met de hoogste adreswaarde van de woorden die associeren.
7. Na iedere zogenaamde 'tijdelijk vrij-puls' die aan de besturing wordt gegeven, het adres produceren van het opvolgende woord dat associeert.

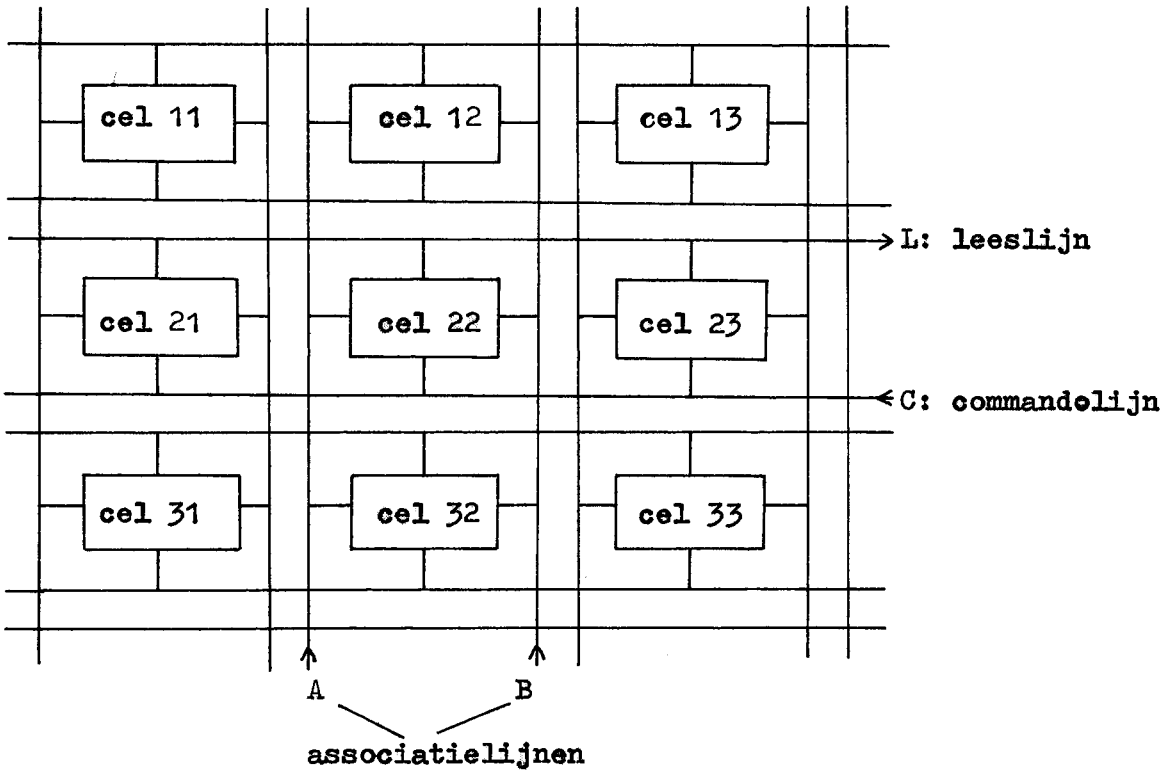


Fig 2.1 Opbouw van de geheugenmatrix

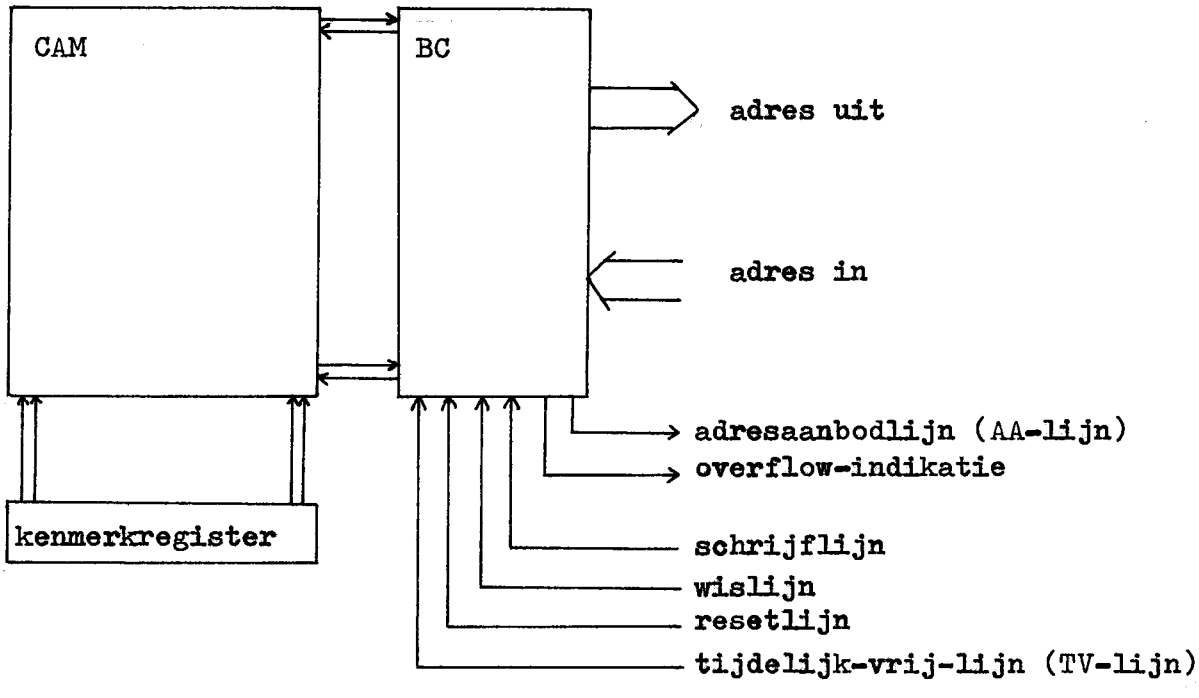


Fig 2.2 Samenbouw van het CAM en de besturing BC (Besturing CAM)

Voor de kommunikatie met de buitenwereld heeft de besturing BC een aantal ingaande en uitgaande lijnen.
Fig 2.2 geeft de samenbouw van het CAM en de besturing BC.
Het CAM en de besturing BC zijn met elkaar verbonden door 1024 leeslijnen en 1024 commandolijnen.

De uitgaande lijnen van de besturing BC zijn:

1. De adresaanbodlijn (AA-lijn).
   Deze lijn wordt aktief als een of meer woorden in het CAM associeren bij het aanbieden van een kenmerk via het kenmerkregister.
2. De overflowlijn.
   Deze lijn meldt het optreden van overflow, dat wil zeggen alle woorden van het geheugen zijn beschreven.
3. De uitgaande adreslijnen (10).
   Deze lijnen geven het adres van het eerste woord in het geheugen dat associeert.

De ingaande lijnen van de besturing BC zijn:

1. De schrijflijn.
   Als er data in het kenmerkregister wordt aangeboden en de schrijflijn wordt geaktiveerd, dan wordt deze data in het geheugen opgeslagen op een plaats, die door de besturing wordt aangegeven (namelijk de eerste lege plaats in het geheugen).
2. De wislijn.
   Als bij het aanbieden van een kenmerk een of meerdere associaties optreden, dan kunnen de betreffende woorden in het CAM worden gewist door de wislijn aktief te maken.
3. De resetlijn.
   Als op de ingaande adreslijnen een adres wordt aangeboden en de resetlijn wordt geaktiveerd, dan wordt het woord op de geadresseerde plaats gewist.
4. De tijdelijk-vrij-lijn (TV-lijn).
   Als een kenmerk aan het geheugen wordt aangeboden en er treden een of meer associaties op, dan verschijnt in eerste instantie het adres van het eerste woord dat associeert op de adresuitgang van BC.
   Wil men ook de adressen verkrijgen van de opvolgende woorden die associeren, dan moet men achtereenvolgens TV-pulsen geven.
   Na iedere TV-puls verschijnt het adres van het opvolgende woord dat associeert op de adresuitgang van de besturing.
5. De ingaande adreslijnen (10).
   Indien men een woord met een bekend adres in het CAM wil wissen, dan dient dit adres op deze adreslijnen te worden aangeboden. Tevens moet de resetlijn geaktiveerd worden.

Met een geheugen, dat volgens fig 2.2 is opgebouwd kunnen vele processen efficienter worden uitgevoerd, dan met een konventioneel geheugen.
Een konfiguratie met meer mogelijkheden wordt verkregen als we met het CAM een random access geheugen (RAM) koppelen.

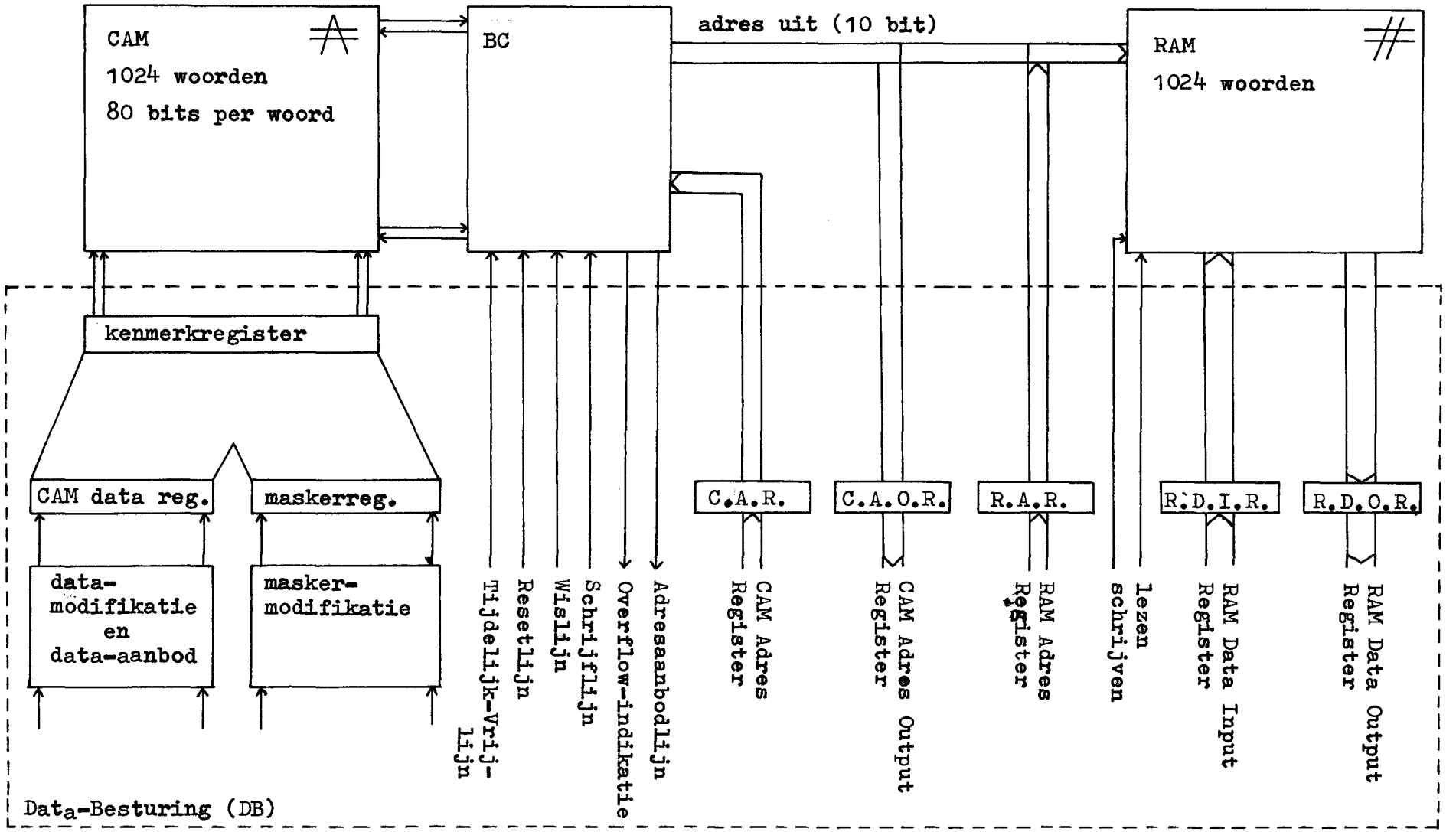


Fig 2.3 Algehele opbouw van het ontworpen associatieve geheugen

Het adres, dat de besturing BC afgeeft kan dienen voor het adresseren van de woorden in het RAM.
Op deze wijze is met ieder woord van het CAM, door middel van het genoemde adres, een woord in het RAM gekoppeld.
Voor vele toepassingen is een dergelijke samenstelling gunstig, vooral ook omdat het CAM niet expliciet uitgelezen kan worden en het RAM wel.

Als we een deelkenmerk (dat wil zeggen een kenmerk waarvan een aantal bits don't cares zijn) aanbieden aan het CAM en we willen de woorden die associeren expliciet uitlezen, dan kunnen we in het RAM een kopie schrijven van de informatie in het CAM.
Het volledige woord kan dan uit het RAM worden uitgelezen, wat bij sorteerprocessen erg zinvol is (zie hoofdstuk 4).
In andere gevallen staat op een plaats in het RAM bepaalde informatie die behoort bij het betreffende woord in het CAM (maar die niet hetzelfde is).

In de volgende hoofdstukken, die handelen over toepassingen van het associatief geheugen, zal blijken, dat de gekozen opzet zinvol is.

De besproken eenheden dienen op de een of andere wijze te worden gekoppeld met een general purpose komputer of komputersysteem. Daarvoor is nodig, afhankelijk van de eisen die men stelt wat snelheid en autonomie van het associatief geheugen betreft, dat een eenheid wordt ontworpen, die regelend optreedt bij de kommunikatie tussen komputer en associatief geheugen.
De verschillende mogelijkheden daarvoor zijn in hoofdstuk 1 al besproken.

In fig 2.3 is aangegeven hoe het geheel er uit komt te zien. De eenheid, die de kommunikatie tussen komputer en het associatief geheugen verzorgt en tevens zorgt voor het modificeren van de bits van het CAM data register en het maskerregister, wordt de Data Besturing (DB) van het associatief geheugen genoemd.

DEEL II

3 TOEPASSINGEN VOOR EEN ASSOCIATIEF GEHEUGEN

Er zijn legio toepassingen gesuggereerd, waarvoor associatieve geheugens bijzonder geschikt zouden zijn. Zowel administatieve, militaire als wetenschappelijke toepassingen worden door verschillende auteurs genoemd.
Hanlon (1966), zie ref 5, geeft een uitgebreid overzicht van de mogelijke toepassingen.
Hij onderscheidt acht klassen van toepassingen:

a. Operaties op lijsten en tabellen
b. Patroonherkenning
c. Oplossen van wiskundige problemen
d. Opslag en terugzoeken van informatie
e. Vertaling
e. Militaire toepassingen
g. Toepassingen in de telecommunicatietechniek
h. Diversen

Er dient echter wel te worden opgemerkt, dat slechts weinig van de voorgestelde toepassingen werkelijk zijn uitgevoerd met een associatief geheugen of een associatieve processor.
Bovendien vereisen sommige toepassingen, zoals beschreven, een speciale opbouw van het geheugen of een uitgebreide logika van de afzonderlijke geheugencellen, zodat niet meer van een 'general purpose' associatief geheugen gesproken kan worden.
De eerste publikaties handelen hoofdzakelijk over sorteerprocessen en elementaire zoekprocessen.
Onderstaande opsomming geeft een overzicht van een aantal toepassingen, zoals deze door een aantal auteurs zijn beschreven. Het nummer van de referenties heeft betrekking op de aan dit verslag toegevoegde literatuurlijst.

Toepassing	auteur(s)	jaar	ref nr
- sorteerprocessen	Seeber	1960	97
	Frei, Goldberg	1961	98
	Seeber, Lindquist	1962	99
	DeFiore	1970	104
- zoekprocessen	Chu	1965	30
- symboolmanipulaties	Green	1961	109
	Mullery	1963	110
	Beaven, Lewin	1972	113

- matrixoperaties	Katz	1970	157
	Gilmore	1974	160
- berekenen van eigenwaarden en eigenvektoren	Estrin	1962	117
- oplossen van partiele differentiaalvergelijkingen	Gilmore	1971	116
- data management	DeFiore, Stillman, Bruce Berra	1971	62
	Linde	1973	78
	DeFiore, Bruce Berra	1973	77
		1974	83
- terugzoeken van informatie	Lee, Paul	1963	16
- 'look-aside' geheugen	Lee	196	48
- dynamische geheugentoewijzing	Chu	1965	131
- verwerken van Sonar-signalen	Bird	1970	157
- verwerken van Radar-signalen	Bergland	1972	70
	Cornell	1972	
	Moulder	1973	
	Couranz	1974	160
- luchtverkeersleiding	Thurber	1971	
	Downs	1973	76
- patroonherkenning	Fuller, Bird	1965	152
	Yau, Yang	1966	153

Enige in bedrijf zijnde systemen zijn reeds in hoofdstuk 1 genoemd.

Het Goodyear associatief geheugen wordt onder andere gebruikt voor sorteerprocessen (DeFiore, ref 104) en voor data-management problemen (DeFiore,Bruce Berra, ref 77).

De STARAN S wordt onder andere gebruikt voor data management problemen (DeFiore, Bruce Berra, ref 86).

De PEPE associatieve processor wordt gebruikt voor het verwerken van Radar-signalen (Bergland,Hunnicutt, ref 70).

In de volgende hoofdstukken zullen een aantal toepassingen meer of minder gedetailleerd worden uitgewerkt.
Er is een onderverdeling gemaakt in drie groepen, afhankelijk van de funktie die het associatief geheugen in het apparaat vervult.

1. Het associatief geheugen is, wat we noemen een randapparaat en heeft alleen een funktie bij die processen, waarbij het gebruik ervan zinvol is.
Het geheugen is hier dus een apparaat, dat het mogelijk maakt om een aantal door de programmeur gebruikte 'subroutines' sneller en efficienter uit te voeren.
Dit impliceert ook dat de software voor de programmeur vereenvoudigd wordt (zie 4.7 voor een sorteerproces).
In hoofdstuk 1 is al opgemerkt, dat in dit geval het associatief geheugen niet werkelijk een randapparaat behoeft te zijn; het kan ook een onderdeel zijn van het konventioneel geheugen.

2. Het associatief geheugen is een geintegreerd onderdeel van het komputersysteem, waarbij het een zinvol hulpmiddel is voor de besturingsfunkties van het systeem.
Zo kan het associatief geheugen worden gebruikt voor dynamische geheugentoewijzing bij een virtueel geheugensysteem (paging), voor het opslaan van aanvragen van I/O-devices en voor besturingsfunkties in time-sharing en multi-programmeringssystemen.
Een associatief geheugen, toegepast voor deze doeleinden zal over het algemeen van kleiner formaat kunnen zijn dan voor de onder punt 1 bedoelde toepassingen.

3. Een associatief geheugen kan deel uitmaken van een special purpose apparaat.
Als voorbeelden zijn te noemen:
- machines voor patroonherkenning
- lerende-geheugen-systemen
- message-switching-systemen
- radar-detektie-systemen

In de volgende hoofdstukken wordt beschreven hoe een associatief geheugen voor de drie typen van toepassingen kan worden gebruikt.

4 SORTEERPROCESSEN

4.1 Inleiding

Sorteren komt neer op het in volgorde plaatsen van een aantal items. Sorteer-algorithmes zijn in het algemeen niet moeilijk te begrijpen, maar een vergelijking tussen de verschillende algorithmes, wat hun efficiency betreft is niet zo eenvoudig.
Bij de meeste sorteer-algorithmes worden tijdens het sorteren de elementen van een plaats naar een andere plaats overgebracht.
Er bestaat geen enkele sorteertechniek die optimaal is voor iedere situatie.
De snelste methoden zijn moeilijker te programmeren; dit kost te veel tijd en moeite als slechts enkele relatief korte lijsten gesorteerd moeten worden.

De keuze van het algorithme hangt ook af van:
- de lengte van de te sorteren lijst
- de verhouding tussen de lengte van de lijst en het aantal woorden in het centrale geheugen van de machine
- de mate waarin de elementen reeds gesorteerd zijn
- de verdeling van de waarden van de elementen

In het volgende worden in het kort enige sorteertechnieken aangegeven bij gebruik van een konventioneel geheugen.
De efficiency van deze technieken wordt onderling vergeleken.
Daarna wordt nagegaan hoe een sorteerproces verloopt als we gebruik maken van een associatief geheugen, en wat de winst is t.o.v het gebruik van een normaal geheugen.

4.2 Konventionele sorteermethoden

We kunnen deze methoden splitsen in twee klassen:

- de interne sorteermethoden, waarbij alle te sorteren elementen tegelijkertijd in het werkgeheugen van de rekenmachine aanwezig kunnen zijn.
- de externe sorteermethoden, waarbij niet alle elementen tegelijkertijd in het werkgeheugen aanwezig kunnen zijn.

Bij de interne sorteermethoden onderscheiden we:
- invoegmethoden (insertion)
- selektiemethoden (selection)
- verwisselingsmethoden (exchange)
- samenvoegmethoden (merge)
- radixmethoden (radix)
- splitsingsmethoden (distribution)

sorteermethode	geheugen-ruimte	aantal vergelijkingen	aantal verwisselingen/ opschuivingen
invoegmethoden	n	$\frac{1}{4}n^2$	$\frac{1}{4}n^2$
selektiemethoden	n	$\frac{1}{2}n^2$	n
verwisselingsmethoden			
- bubble sort	n	$\frac{1}{4}n(n+1)$	$\frac{1}{4}n(n-1)$
- brick sort	n	$\frac{1}{4}n(n+1)$	$\frac{1}{4}n(n-1)$
- Shell sort	n	n.log n	n.log n
samenvoegingsmethoden 1)			
2-way	2n	$n.^2\log n$	$n.^2\log n$
p-way	2n	$2(p-1)^p\log n$	$n.^p\log n$
radixmethoden 2)			
10-tallige sleutel	10n	n.k	
splitsingsmethoden			
radix-echange 3)	n	n.k	$\frac{1}{4}n.k$
quick-sort	n+log n	2n.log n	n.log n
sample-sort	n	$n.^2\log n$	

1) Samenvoegmethoden worden gebruikt bij externe sorteerprocessen met behulp van magneetbanden.

2) Bij radixmethoden wordt direkt gebruik gemaakt van de representatie van de sleutel (k bit) van een record in een willekeurig talstelsel.

3) Radix-echange voor binaire sleutels van k bits.

Tabel 4.1 Enkele karakteristieke gegevens van sorteermethoden voor random access geheugens.

In tabel 4.1 zijn de verschillende methoden opgenomen, met de benodigde geheugenruimte en het gemiddeld aantal vergelijkingen en verwisselingen, dat uitgevoerd moet worden om n elementen te sorteren.
Men moet voorzichtig zijn met het zonder meer trekken van konklusies uit deze tabel.
De eigenschappen van de te sorteren rij zelf zijn van groot belang voor de keuze van een sorteermethode.

Over de verschillende externe sorteermethoden valt weinig konkreets te zeggen, wat betreft snelheid en efficiency. Deze hangen sterk af van de beschikbare ruimte in het centrale geheugen van de komputer en van de beschikbare magneetbandeenheden en/of schijvengeheugens en van de toegangstijd van deze geheugens.
Over het algemeen worden bij externe sorteermethoden merge-processen toegepast.
Bij merge-processen worden twee of meer gesorteerde of gedeeltelijk gesorteerde deelrijen samengevoegd tot een (meer) gesorteerde rij.
Een merge-proces kan in een aantal fasen verlopen.

Enkele externe sorteermethoden zijn:
- balanced sort
- polyphase sort
- replacement sort
- oscillating sort

Voor een uitvoeriger behandeling van de verschillende sorteermethoden wordt verwezen naar de literatuur op dit gebied, met name de referenties: 95,96,105.

4.3 Sorteren met behulp van een associatief geheugen

Met een associatief geheugen kunnen sorteerprocessen snel worden uitgevoerd en wel op grootheden, die in binaire representatie in het associatief geheugen zijn opgeslagen.

We moeten echter tevoren wel het volgende bedenken:
- als het associatief geheugen niet voor andere doeleinden wordt gebruikt gedurende de tijd dat de items aktief dienen te zijn, dan zal sorteren waarschijnlijk niet nodig zijn. Men heeft namelijk via een kenmerk direkt toegang tot elk woord in het geheugen.
- als het associatief geheugen zo snel mogelijk weer beschikbaar moet zijn voor andere doeleinden, of de inhoud van het geheugen is een deel van een groter bestand, dat gesorteerd moet worden, dan zal wel zo snel mogelijk een sorteerproces uitgevoerd moeten worden.

Allereerst bespreken we het sorteren van getallen in binaire vorm.

We kunnen sorteren naar oplopende binaire waarde of naar afnemende binaire waarde. Sorteren van getallen in floating-point notatie wordt voorlopig buiten beschouwing gelaten.
De huidige opzet van het geheugen maakt het mogelijk om maximaal 1k woorden van 80 bits op te slaan.
Hoewel het voldoende is als de te sorteren elementen alleen in het CAM aanwezig zijn, is het voor de snelheid van het proces gunstig als de elementen ook in het RAM aanwezig zijn. Het inschrijven van de elementen in CAM en RAM kan tegelijkertijd gedaan worden, en wel op overeenkomstige lokaties.

Achtereenvolgens zullen nu drie sorteermethoden voor associatieve geheugens worden besproken, en daarna zal voor een methode worden aangegeven hoe deze gerealiseerd kan worden met het ontworpen associatieve geheugen.
De verhouding tussen het aantal keren dat een kenmerk moet worden aangeboden en het aantal keren dat de ondervraging succes heeft (d.w.z.dat een woord wordt uitgelezen), is een funktie van het aantal woorden dat gesorteerd moet worden en het aantal bits waarop gesorteerd moet worden.
Deze funktie wordt gegeven door R(b,w), waarbij w het aantal woorden is en b het aantal bits.

We lichten elk van de sorteerprocessen toe aan de hand van een voorbeeld van het sorteren van zes woorden van zes bit.

Woordnummer	binair woord					
1	1	1	0	1	1	0
2	1	1	0	1	1	1
3	0	1	1	0	0	1
4	1	0	0	1	1	0
5	0	0	0	1	0	1
6	0	1	0	0	0	1

tabel 4.2 zes binaire woorden die gesorteerd moeten worden

4.3.1 Het sorteerproces van Frei en Goldberg

In eerste instantie bevat het woord dat als kenmerk aan het associatief geheugen wordt aangeboden alleen don't cares. Daarna worden de bits achtereenvolgens gewijzigd, te beginnen bij het meest significante bit.

Een bit van het kenmerk, dat oorspronkelijk een don't care is, wordt achtereenvolgens 0 en 1 gemaakt.
Er is een associatie-indikator(AI) met twee standen, namelijk 0(geen associatie) en 1(een of meer associaties).
Tabel 4.3 geeft het verloop van het sorteerproces van Frei en Goldberg toegepast op de woorden van tabel 4.2.

Cyclus nr	kenmerk						AI	woorden die associeren
1	x	x	x	x	x	x	1	1,2,3,4,5,6
2	0	x	x	x	x	x	1	3,5,6
3	0	0	x	x	x	x	1	5
4	0	0	0	x	x	x	1	5
5	0	0	0	0	x	x	0	
6	0	0	0	1	0	x	1	5
7	0	0	0	1	0	0	0	
8	0	0	0	1	0	1	1	5 =
9	0	0	0	1	1	x	0	
10	0	0	1	x	x	x	0	
11	0	1	x	x	x	x	1	3,6
12	0	1	0	x	x	x	1	6
13	0	1	0	0	x	x	1	6
14	0	1	0	0	0	x	1	6
15	0	1	0	0	0	0	0	
16	0	1	0	0	0	1	1	6 =
17	0	1	0	0	1	x	0	
18	0	1	0	1	x	x	0	
19	0	1	1	x	x	x	1	3
20	0	1	1	0	x	x	1	3
21	0	1	1	0	0	x	1	3
22	0	1	1	0	0	0	0	
23	0	1	1	0	0	1	1	3 =
24	0	1	1	0	1	x	0	
25	0	1	1	1	x	x	0	
26	1	x	x	x	x	x	1	1,2,4
27	1	0	x	x	x	x	1	4
28	1	0	0	x	x	x	1	4
29	1	0	0	0	x	x	0	
30	1	0	0	1	0	x	0	
31	1	0	0	1	1	0	1	4 =
32	1	0	0	1	1	1	0	
33	1	0	1	x	x	x	0	
34	1	1	x	x	x	x	1	1,2
35	1	1	0	x	x	x	1	1,2
36	1	1	0	0	x	x	0	
37	1	1	0	1	0	x	0	
38	1	1	0	1	1	0	1	1 =
39	1	1	0	1	1	1	1	2 =
40	1	1	1	x	x	x	0	

tabel 4.3 verloop van het sorteerproces volgens Frei en Goldberg

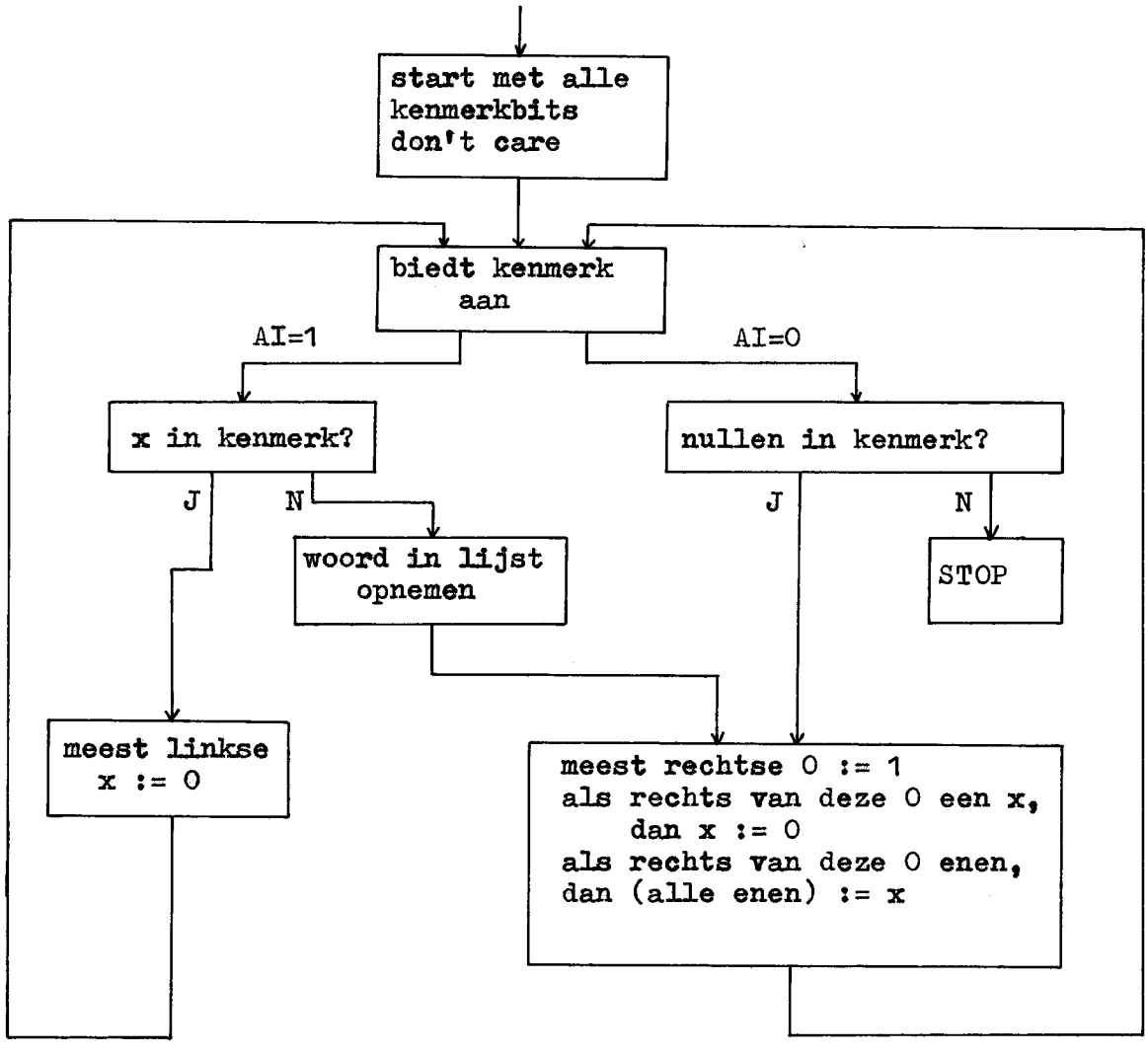


Fig 4.4 Flow-diagram van het sorteerproces volgens Frei en Goldberg

In de voorlaatste kolom is de stand van de associatie-indikator weergegeven, en in de laatste kolom welke woorden op dat kenmerk associeren.
Als een woord gevonden is, wordt dat door een gelijkteken in de laatste kolom aangegeven, na het nummer van het gevonden woord.
Bij dit voorbeeld van de methode van Frei en Goldberg is R(b,w) = 40/6 = 6,67.

4.3.2 Het sorteerproces van Seeber en Lindquist

Dit proces lijkt veel op het proces van Frei en Goldberg. Men start ook met het aanbieden van don't cares. Daarna worden de bits van links naar rechts in een aantal stappen gewijzigd.

Er zijn echter twee verschillen met het vorige proces:
- de associatie-indikator (AI) heeft drie standen:
  stand 0 = geen associaties
  stand 1 = een associatie
  stand m = meer dan een associatie
- als een woord op een deelkenmerk associeert kan het expliciet worden uitgelezen.

Als we bij deze methode dus een associatie vinden, (AI)=1, dan hebben we een woord gevonden, wat aan de gesorteerde lijst toegevoegd moet worden.

Om de woorden expliciet te kunnen uitlezen moet:
- of het CAM uit te lezen zijn op associatie
- of het bijbehorend RAM uit te lezen zijn, op dezelfde lokatie, waarop bij het CAM een associatie optreedt.

Uiteraard dient in het laatste geval in het CAM en in het RAM dezelfde informatie aanwezig te zijn.
Dit laatste is bij het ontworpen geheugen het geval.
De onderstaande tabel voor het sorteerproces van Seeber en Lindquist is op dezelfde wijze opgezet als de tabel bij de vorige methode.
Ook nu weer geeft een gelijkteken (=) aan dat een woord gevonden is.

Cyclus nr	kenmerk						AI	woorden die associeren
1	x	x	x	x	x	x	m	1,2,3,4,5,6
2	0	x	x	x	x	x	m	3,5,6
3	0	0	x	x	x	x	1	5 =
4	0	1	x	x	x	x	m	3,6
5	0	1	0	x	x	x	1	6 =
6	0	1	1	x	x	x	1	3 =

7	1	x	x	x	x	x	m	1,2,4
8	1	0	x	x	x	x	1	4 =
9	1	1	x	x	x	x	m	1,2
10	1	1	0	x	x	x	m	1,2
11	1	1	0	0	x	x	0	
12	1	1	0	1	0	x	0	
13	1	1	0	1	1	0	1	1 =
14	1	1	0	1	1	1	1	2 =
15	1	1	1	x	x	x	0	

tabel 4.5 verloop van het sorteerproces volgens Seeber en Lindquist

Bij dit voorbeeld van de methode van Seeber en Lindquist is R(b,w) = 15/6 = 2,50.

4.3.3 Het sorteerproces volgens Lewin

Dit is het snelste van de drie processen, die hier behandeld worden. Het stelt echter speciale eisen aan de uitvoeringsvorm van het associatief geheugen.
Er wordt nu geen gebruik gemaakt van een associatie-indikator, maar voor iedere bitpositie van het geheugen zijn er twee lijnen aanwezig, die aangeven of voor alle geselekteerde woorden deze bit gelijk is of niet.
Deze lijnen worden sense-draden genoemd.
De tabel geeft de betekenis aan van de signalen op de sense-lijnen.

Naam	signalen op sense-draden	betekenis
0	1 0	alle geselekteerde woorden hebben een 0 op deze plaats
1	0 1	alle geselekteerde woorden hebben een 1 op deze plaats
m	1 1	sommige geselekteerde woorden hebben een 0 op deze plaats; anderen een 1
y	0 0	er worden geen woorden geselekteerd (alleen gebruikt bij eerste stap)

tabel 4.6 de signalen op de sense-draden bij het proces volgens Lewin

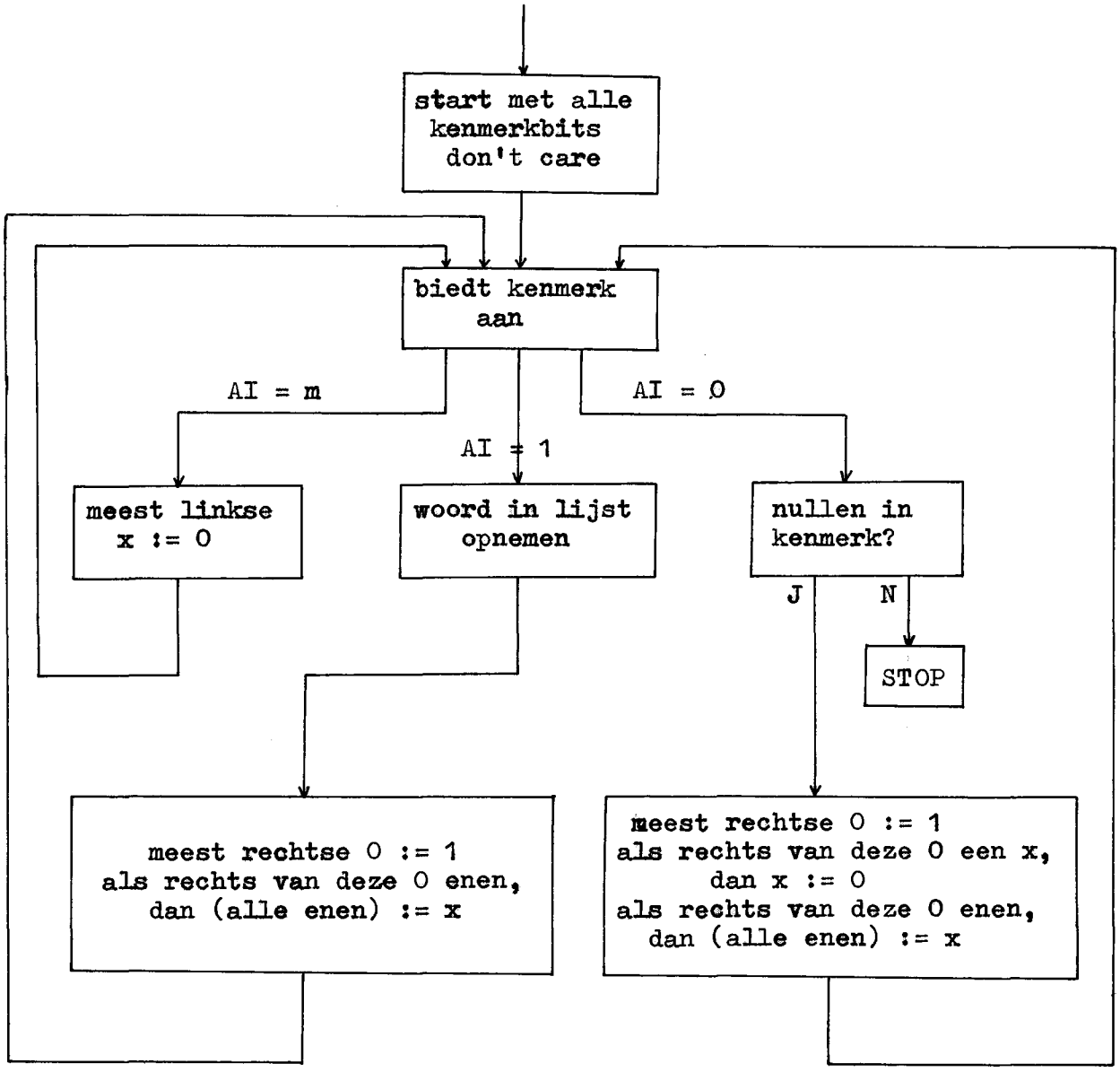


Fig 4.7 Flowdiagram van het sorteerproces volgens Seeber en Lindquist.

Aan de hand van de informatie ,die de sense-draden leveren worden de bits van het kenmerk gewijzigd.

Om het sorteerproces van Lewin toe te lichten nemen we weer de zes woorden van zes bits uit de tabel.
Begonnen wordt weer met het aanbieden van don't cares als kenmerk.
Het proces verloopt verder zoals in tabel 4.8 is weergegeven.
Fig 4.9 geeft het flowdiagram van het sorteerproces van Lewin.

Cyclus	kenmerk						sense-signalen						woorden die associeren
1	x	x	x	x	x	x	m	m	m	m	m	m	1,2,3,4,5,6
2	0	x	x	x	x	x	0	m	m	m	m	m	3,5,6
3	0	0	x	x	x	x	0	0	0	1	0	1	5 =
4	0	1	x	x	x	x	0	1	m	0	0	1	3,6
5	0	1	0	x	x	x	0	1	0	0	0	1	6 =
6	0	1	1	x	x	x	0	1	1	0	0	1	3 =
7	1	x	x	x	x	x	1	m	0	1	1	m	1,2,4
8	1	0	x	x	x	x	1	0	0	1	1	0	4 =
9	1	1	x	x	x	x	1	1	0	1	0	m	1,2
10	1	1	x	x	x	0	1	1	0	1	0	0	1 =
11	1	1	x	x	x	1	1	1	0	1	0	1	2 =

tabel 4.8 verloop van het sorteerproces volgens Lewin

Bekijken we nog even cyclus 4; hier wordt als kenmerk 0,1,x,x,x,x aangeboden.
Er zijn twee woorden, namelijk 3 en 6, die met 0,1 beginnen.
Deze waarden komen op de sense-draden van bit 1 en 2.
Bit 4,5 en 6 van de woorden 3 en 6 zijn hetzelfde, dus deze waarden (0,0,1) komen op de sense-draden.
De woorden 3 en 6 verschillen alleen op positie 3, dus er komt een m op de sense-draden van bit 3.
Het resultaat is dat we op de sense-draden 0,1,m,0,0,1 krijgen.

Bij dit voorbeeld van de methode van Lewin is $R(b,w) = 11/6 = 1,83$.

4.4 Vergelijking van sorteerprocessen voor een associatief geheugen

We nemen aan, dat het associatief geheugen gevuld is met n elementen, die gesorteerd moeten worden.
Tabel 4.10 geeft de benodigde geheugenruimte en het aantal vergelijkingen (gemiddeld, minimaal en maximaal) voor elk van de drie methoden die eerder zijn besproken.

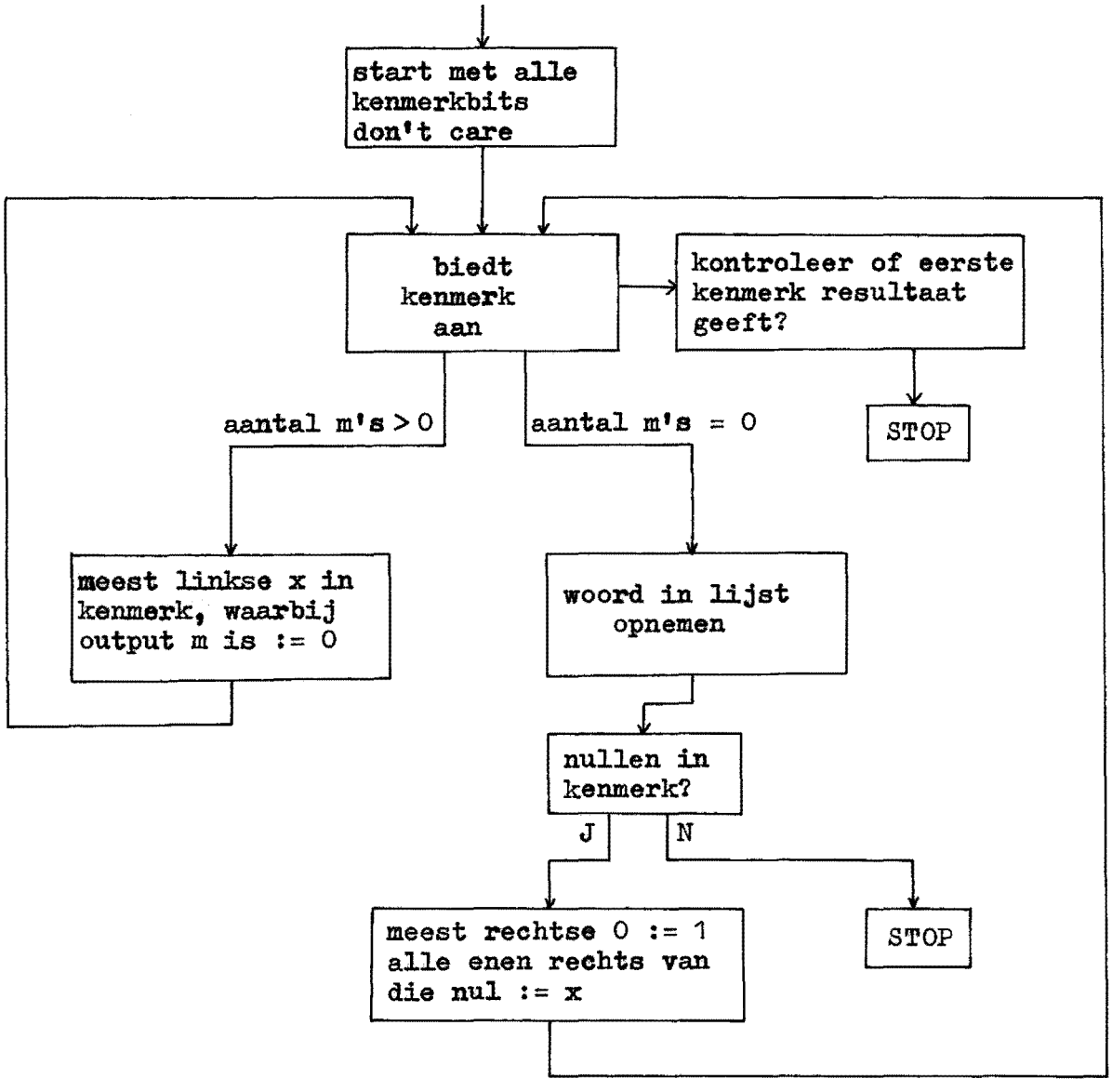


Fig 4.9 Flowdiagram van het sorteerproces volgens Lewin

Voor elk van de drie genoemde methoden hebben we niet meer dan w plaatsen aan geheugenruimte nodig, als w het aantal woorden is dat gesorteerd moet worden. Het aantal bits waarop gesorteerd moet worden wordt aangegeven door b.
Het gemiddeld aantal vergelijkingen is het kleinst bij de methode van Lewin. Dit proces kan echter niet door het ontworpen associatief geheugen worden uitgevoerd.
Het vereist namelijk twee extra draden per bit van het geheugen; dit zijn de reeds genoemde sense-draden.
Het geheugen voor de methode van Lewin is dan ook speciaal ontworpen om een sorteerproces zo snel mogelijk te kunnen uitvoeren.

Sorteermethode	geheugen-ruimte	aantal vergelijkingen		
		gemidd	min	max
Frei/Goldberg	w	bw	2w	3/2.bw
Seeber/Lindquist	w	2,5w	w	bw
Lewin	w	2w-1	2w-1	2w-1

w: aantal woorden in het geheugen
b: aantal bits per woord, waarop gesorteerd moet worden

tabel 4.10 vergelijking van de drie sorteerprocessen met behulp van een associatief geheugen

het sorteerproces volgens Seeber en Lindquist is wel geschikt om te worden uitgevoerd met het ontworpen geheugen; daarom is gekozen voor deze methode.

Als voordelen van het gebruik van een associatief geheugen boven een random access geheugen voor een sorteerproces kunnen worden genoemd:
- er is geen extra geheugenruimte nodig tijdens het uitvoeren van het sorteerproces.
- het aantal vergelijkingen om w elementen te sorteren is gemiddeld 2,5w bij de methode van Seeber en Lindquist.
- een vergelijking met behulp van een associatief geheugen duurt aanmerkelijk korter dan een vergelijking bij gebruik van een random access geheugen.

Om een element van een RAM te kunnen vergelijken met een ander element, moet het betreffende element uit het RAM worden uitgelezen en extern met het andere element worden vergeleken.

Bij een associatief geheugen wordt het kenmerk tegelijkertijd met alle elementen in het geheugen vergeleken.
Bovendien hoeven bij het sorteerproces na iedere vergelijking slechts een of een paar bits van het CAM Data Register of van het masker te worden veranderd; dan kan het volgende kenmerk al weer worden aangeboden.

4.5 Realisatie van het sorteerproces van Seeber en Lindquist

Om het sorteerproces van Seeber en Lindquist te kunnen uitvoeren met het associatief geheugen, zal een besturing gerealiseerd moeten worden, die er voor zorgt dat het proces verloopt volgens het flow-diagram van fig 4.7.
De gehele opbouw van CAM, de besturing van CAM (BC), RAM en de Data Besturing komen er dan uit te zien als in fig 4.11.
Als het proces begint worden alle bits van het data-register 0 gemaakt en alle maskerbits worden 1 gemaakt, dat wil zeggen, er worden enkel don't cares aangeboden.
De associatie-indikator wordt gereset: (AI):=0.
Via het kenmerkregister wordt het kenmerk aan het associatief geheugen aangeboden.
De besturing BC reageert als volgt:

1. De adresaanbodlijn (AA-lijn) wordt aktief als er een associatie is. Dan wordt (AI):=1 en de TV-pulsgever krijgt opdracht om een TV-puls te geven.
Er doen zich dan twee mogelijkheden voor:
-a. De AA-lijn blijft aktief, dan wordt (AI):=m, dat wil zeggen dat er meerdere associaties zijn.
De stand van de associatie-indikator (AI) mag nu aan het data-masker-modifikatiemechanisme worden doorgegeven.
Het gevolg is dat een of meerdere bits van het data-register en/of het maskerregister woden gewijzigd, waarna het volgende kenmerk aan het CAM kan worden aangeboden.
-b. In geval dat na de TV-puls de AA-lijn niet aktief blijft was er slechts een associatie.
Het gevolg is dat het RAM een leesopdracht krijgt en het woord waarvan het adres op de adres-uitgang van BC verschijnt, wordt uitgelezen.
Door het data-masker-modifikatiemechanisme worden nu weer de data-bits en de masker-bits veranderd.

2. De AA-lijn wordt niet aktief; er is dan geen associatie.
Dit wordt doorgegeven aan het data-masker-modifikatiemechanisme, dat zorgt voor wijziging van data- en masker-bits.
Volgens deze methode worden de woorden in volgorde van oplopende binaire waarde uit het RAM uitgelezen.

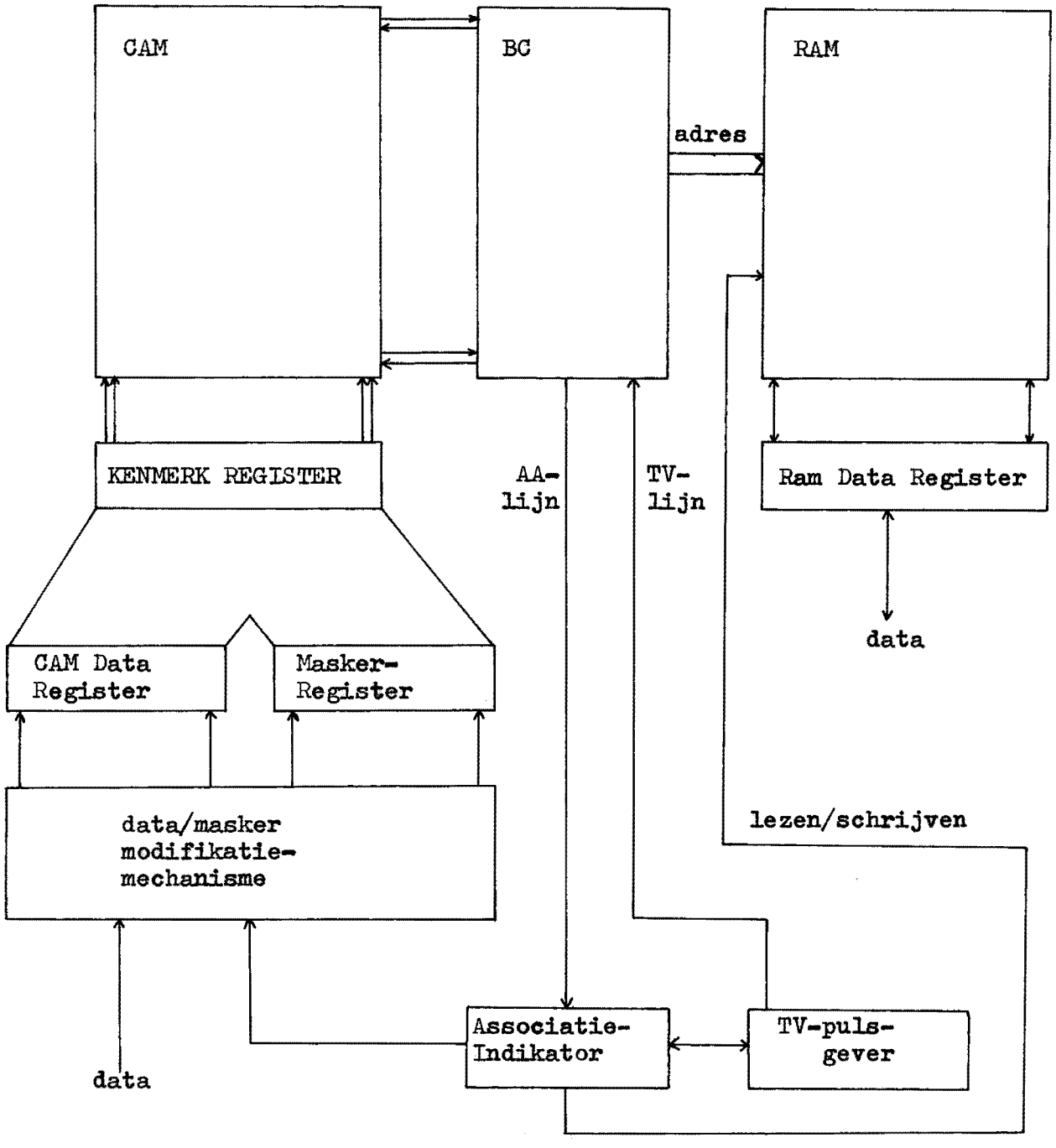


Fig 4.11 Organisatie van het associatief geheugen voor het uitvoeren van het sorteerproces van Seeber en Lindquist.

Het sorteerproces wordt beeindigd als alle woorden zijn gesorteerd. Dit wordt gekonstateerd als volgt:
- als het aantal te sorteren woorden bekend is kan men met een teller bijhouden hoeveel er op ieder moment zijn uitgelezen en dus weet men wanneer het proces beeindigd kan worden.
- als het aantal te sorteren woorden onbekend is, dan kan het proces worden beeindigd als het eerste bit van het masker een 1 is geworden, dat wil zeggen het eerste bit van het kenmerk wordt gemaskeerd.

Behalve sorteren in oplopende volgorde kan met minieme wijzigingen ook gesorteerd worden in afnemende volgorde. Als we namelijk in het flow-diagram van fig 4.7 voor x,0,1 hun komplementaire waarden, resp x,1,0 nemen, dan vindt sorteren plaats naar afnemende binaire waarde.

In veel gevallen zal het nodig zijn om de inhoud van het CAM te sorteren niet op het gehele kenmerk, maar op een deelkenmerk. We moeten kunnen sorteren op de bits tussen twee willekeurig gekozen grenzen van het kenmerk. De data besturing dient dan ook hierin te voorzien.

Bij sommige processen zal het nuttig zijn dat met de bits van het CAM data register of van het kenmerkregister geschoven kan worden om snel een kenmerk te kunnen veranderen. Dat is ook een van de eisen die gesteld moet worden aan de data besturing van het associatief geheugen (DB).

4.6 Sorteren van grote bestanden met behulp van een associatief geheugen

Magnetische banden bieden de mogelijkheid tot het opslaan van grote hoeveelheden informatie. Om met deze informatie te werken zal het in de praktijk nodig zijn, dat de informatie geordend op de band staat. Is dit niet het geval, dan dient men de informatie eerst te ordenen (sorteren) alvorens tot verwerking over te gaan. Dit sorteren verloopt in verschillende etappes. Er kan hierbij ook gebruik worden gemaakt van het associatief geheugen. Allereerst wordt van de ongesorteerde band een zo groot mogelijk stuk informatie in het associatief geheugen ingelezen. Deze informatie wordt intern gesorteerd. De gesorteerde stukken informatie worden afwisselend op andere banden geschreven (verdeelgang). Tenslotte worden de gesorteerde stukken gemengd tot langere gesorteerde stukken, totdat een gesorteerde rij informatie ontstaat (menggangen). Over deze externe sorteermethoden zijn meer gegevens te vinden in de referenties 95 en 96. De snelheidswinst die hier door het gebruik van het associatief geheugen wordt verkregen is relatief klein ten opzichte van de tijd die het kost om de gegevens vanaf de banden binnen te halen en weer weg te schrijven op een of meer magneetbanden.

4.7 Sorteerproces met het Goodyear Associatief Geheugen en de CDC-1604B komputer

Het Goodyear Associatief Geheugen, zoals dat in hoofstuk 1 is beschreven kan worden afgezocht op een exact match (gelijkheid) in 70 usec, inclusief de tijd voor het verkrijgen van een kenmerk, de instruktie en een masker. Het zoeken naar de maximum of de minimum waarde in het geheugen vereist minder dan 160 usec als niet gemaskeerd behoeft te worden.

Een associatief sorteerproces voor het sorteren van woorden in oplopende of afnemende volgorde kan met het geheugen worden uitgevoerd. Een algorithme daarvoor is geschreven in JOVIAL voor de kombinatie CDC 1604B-associatief geheugen en vergeleken met de bubble-sorteermethode die normaal beschikbaar is in de 1604B.
Bij het sorteren van 2k woorden van 48 bit was het associatief sorteeralgorithme 375 maal zo snel als het bubble sorteerproces.

De JOVIAL-procedure ziet er als volgt uit:

```
ASORT(A,B,C,D,E,F,)

A: naam inputgebied
B: naam outputgebied
C: integer I, 0<I<2049
D: V(INC) voor oplopend sorteren
D: V(DEC) voor afnemed sorteren
E: integer I, 0<I<48
     geeft startbit aan in ieder woord
F: integer I, 0<I<49
     aantal bits van ieder woord waarop gesorteerd moet
     worden
```

Dit systeem met de CDC-1604B komputer en het Goodyear Associative Memory is beschreven in de ref's 77, 104.

5 ZOEKPROCESSEN

Onder zoekprocessen verstaan we het opzoeken van woorden in het associatief geheugen, die een zekere relatie bezitten tot een aangeboden woord.

We kunnen daarbij de volgende zoekprocessen onderscheiden:
1. Zoeken of een gegeven woord in het geheugen aanwezig is
2. Het bepalen van het kleinste woord in het geheugen
3. Het bepalen van het grootste woord in het geheugen
4. Het bepalen van het woord in het geheugen, dat in waarde voorafgaat aan een gegeven woord
5. Het bepalen van het woord in het geheugen, dat in waarde volgt op een gegeven woord
6. Het bepalen van alle woorden in het geheugen die kleiner zijn dan een gegeven woord
7. Het bepalen van alle woorden in het geheugen die groter zijn dan een gegeven woord
8. Het bepalen van de woorden in het geheugen die liggen tussen twee opgegeven grenzen
9. Het bepalen van de woorden in het geheugen die de kleinste Hamming-afstand hebben tot een gegeven woord

In het RAM kan dezelfde informatie worden opgeslagen als in het CAM, als men namelijk naar aanleiding van een associatie waarbij het kenmerk nog een aantal don't cares bevat, het woord expliciet wil kunnen uitlezen.
Anderzijds kan het RAM informatie bevatten die volledig verschillend is van de informatie die in het CAM is opgeslagen. Deze informatie kan dan bij een associatie worden uitgelezen. In weer andere gevallen is het voldoende het adres van het woord dat associeert ter beschikking te krijgen.

We zullen nu de verschillende zoekprocessen aan een nader onderzoek onderwerpen en aangeven hoe ze met behulp van het voorgestelde associatieve geheugen uitgevoerd kunnen worden.

1. Zoeken of een gegeven woord in het geheugen aanwezig is.
Dit is het meest elementaire proces bij een associatief geheugen. Het woord waarvan een aantal bitplaatsen don't cares kan bevatten (deze bitplaatsen worden gemaskeerd), wordt aan het CAM aangeboden en de AA-lijn (adresaanbod lijn) geeft aan of er al dan niet een associatie optreedt.
Meerdere associaties kunnen worden gedetekteerd door achtereenvolgens TV-pulsen (tijdelijk-vrij pulsen) te geven.

2. Bepalen van het kleinste woord in het geheugen.
Daartoe wordt het sorteerproces (Seeber en Lindquist) gestart (zie hoofdstuk 4).

Bij de eerste ondervraging waarbij de associatie-indikator (AI) in de stand 1 (dat wil zeggen 1 associatie) blijft staan is het gezochte woord gevonden.

3. Bepalen van het grootste woord in het geheugen.
Daartoe wordt het 'omgekeerde' sorteerproces gestart, dat wil zeggen, het proces waarbij de woorden in afnemende volgorde worden gesorteerd.
Bij de eerste ondervraging, waarbij de associatie-indikator in de stand 1 blijft staan is het gezochte woord gevonden.

4. Bepalen van het woord dat voorafgaat aan een gegeven woord.
In het kenmerkregister wordt het gegeven woord opgeslagen.
De associatie-indikator wordt in de stand 0 gezet en het sorteerproces voor afnemende volgorde wordt gestart als eerste stap het modificeren van het kenmerk en/of het masker zoals dat bij het sorteerproces gebeurt als de associatie-indikator na een ondervraging in de stand 0 is gekomen. De eerste keer dat de stand van de associatie-indikator ongelijk aan nul is, is het gezochte woord gevonden.

5. Bepalen van het woord in het geheugen dat volgt op een gegeven woord.
In het kenmerkregister wordt het gegeven woord opgeslagen.
De associatie-indikator wordt in de stand 0 gezet en het sorteerproces voor sorteren naar oplopende volgorde wordt gestart met als eerste stap het modificeren van de bits van het CAM Data Register en/of van het maskerregister, zoals dat bij het sorteerproces gebeurt als de associatie-indikator na een ondervraging in de stand 0 is terecht gekomen.
De eerste keer dat de associatie-indikator een waarde ongelijk aan nul krijgt is het gezochte woord gevonden en kan het proces worden gestopt.

6. Bepalen van woorden in het geheugen die kleiner zijn dan een gegeven woord.
Het proces wordt gestart volgens de onder punt 4 beschreven methode. Nadat het eerste woord is gevonden, wordt het proces voortgezet, totdat alle maskerbits 1 zijn geworden, dat wil zeggen, totdat alleen nog maar don't cares worden aangeboden.

7. Bepalen van alle woorden in het geheugen die groter zijn dan een gegeven woord.
Het proces wordt gestart volgens de onder punt 5 beschreven methode. Nadat het eerste woord is gevonden, wordt het proces voortgezet, totdat alle maskerbits 1 zijn geworden, dat wil zeggen, totdat alleen nog maar don't cares worden aangeboden.

8. Het bepalen van de woorden in het geheugen die liggen tussen twee opgegeven woorden.
We beginnen met het aanbieden van de ondergrens als kenmerk. De associatie-indikator wordt op 0 gezet en het sorteerproces voor oplopende volgorde zoals dat in hoofdstuk 4 is beschreven wordt gestart.
We moeten nu wel bij iedere verandering van het kenmerk kontroleren of de bovengrens misschien al bereikt is.
Dit kost nogal wat extra tijd.
Daarom kunnen we ook als volgt te werk gaan als er in het geheugen nog ruimte over is.
Daartoe wordt de bovengrens aan het geheugen aangeboden en er wordt een schrijfopdracht gegeven.
Het adres dat daarop door de besturing BC wordt afgegeven wordt opgeslagen buiten het geheugen.
Treedt er bij het sorteerproces nu een enkele associatie op, dan is een woord tussen de opgegeven grenzen gevonden.
Er moet dan tevens worden gekontroleerd of het bijbehorende adres misschien het adres van de bovengrens is.
Zo niet, dan wordt het proces voortgezet.
Is het adres van het gevonden woord wel gelijk aan het adres van de bovengrens, dan associeerde de bovengrens en moet het proces worden beeindigd.

9. Bepalen van de woorden in het geheugen, die de kleinste Hamming-afstand (d) bezitten tot een gegeven woord.
Het gegeven woord wordt in het CAM data register opgeslagen en aan het geheugen aangeboden via het kenmerkregister.
Er wordt niet gemaskeerd.
Is er een associatie, dan is het gezochte woord gevonden en d=0.
Als de associatie-indikator nul blijft, dan wordt van het CAM data register telkens een bit gekomplementeerd te beginnen met het meest linkse bit. Na iedere wijziging wordt gekontroleerd of er misschien een associatie optreedt.
Is dat zo dan is het woord met de kleinste Hamming-afstand gevonden en kan verdere aktie worden ondernomen.
Het achtereenvolgens komplementeren van telkens 1 bit van het CAM data register moet echter wel worden voortgezet tot alle mogelijke kenmerken met Hamming-afstand 1 aan het geheugen zijn aangeboden. Levert het zoeken naar woorden met een Hamming-afstand 1 geen resultaat op, dan kan worden doorgegaan met het zoeken naar woorden met een Hamming-afstand 2 als dat gewenst is. In dat geval wordt het eerste bit van het data register gekomplementeerd en daarna vanaf bit 2 tot en met bit n wordt telkens 1 bit gekomplementeerd. Het kenmerk wordt aangeboden en er wordt nagegaan of er een associatie optreedt. Dit kan men ook tot grotere Hamming-afstanden voortzetten.
Het nagaan of er in het geheugen woorden zijn opgeslagen met een kleine Hamming-afstand tot een gegeven woord noemt men het zoeken naar een approximate match.

Detektie van woorden met een approximate match die in het geheugen zijn opgeslagen is onder andere van belang bij het herkennen van patronen, zoals cijfers, letters foto's, chromosomen enz.
Een beeld van een letter of cijfer kan worden omgezet in een patroon van bezette en onbezette punten die een binaire matrix vormen.
Vaak worden matrices gebruikt van duizend tot enkele duizenden punten. Patroonherkennen komt neer op het reduceren van deze informatie. In eerste instantie verkrijgt men hieruit bijvoorbeeld een patroon van 100 bits, waardoor de vorm van het teken gekarakteriseerd wordt.
Voor vele mogelijke patronen wordt deze reeks van 100 bits bepaald en in het CAM opgeslagen.
Indien een aangeboden patroon van 100 bits niet exact overeenstemt met een opgeslagen patroon, dan kan door te zoeken naar een approximate match tot een herkenning van de aangeboden vorm (cijfer, letter, chromosoom) worden gekomen.
Hierover echter meer in een later hoofdstuk.

We zullen het proces van het bepalen van de kleinste Hamming-afstand toelichten aan een voorbeeld

Stel in het geheugen zijn de volgende zes woorden van zes bits elk opgeslagen.

Woordnummer	bitpatroon					
1	0	1	0	0	1	0
2	1	0	0	0	1	1
3	0	1	0	1	0	1
4	1	1	0	1	1	0
5	1	1	1	0	0	1
6	1	0	0	0	0	0

tabel 5.1 de in het geheugen opgeslagen woorden

We willen de woorden selekteren met de kleinste Hamming-afstand tot het gegeven woord: 1 0 1 1 0 1 .
Het proces verloopt zoals in onderstaande tabel is weergegeven.

Als de bits 2 en 4 gewijzigd worden vinden we het woord met de kleinste Hamming-afstand.
Nadat alle wijzigingen van twee bits ten opzichte van het gegeven woord zijn uitgevoerd blijkt dat er geen andere woorden in het geheugen voorkomen met een Hamming-afstand gelijk aan twee, en daarmee is het proces beeindigd.

Gewijzigde bits	aangeboden kenmerk						associatie indikator	
	1	0	1	1	0	1	0	
1	0	0	1	1	0	0	0	
2	1	1	1	1	0	1	0	
3	1	0	0	1	0	1	0	
4	1	0	1	0	0	1	0	
5	1	0	1	1	1	1	0	
6	1	0	1	1	0	0	0	
1,2	0	1	1	1	0	1	0	
1,3	0	0	0	1	0	1	0	
1,4	0	0	1	0	0	1	0	
1,5	0	0	1	1	1	1	0	
1,6	0	0	1	1	0	0	0	
2,3	1	1	0	1	0	1	0	
2,4	1	1	1	0	0	1	1	=
2.5	1	1	1	1	1	1	0	
2,6	1	1	1	1	0	0	0	
3,4	1	0	0	0	0	1	0	
3,5	1	0	0	1	1	1	0	
3,6	1	0	0	1	0	0	0	
4,5	1	0	1	0	1	0	0	
4,6	1	0	1	0	0	0	0	
5,6	1	0	1	1	1	0	0	

tabel 5.2 Het bepalen van het woord met de kleinste Hamming-afstand tot een gegeven woord

6. BEWERKINGEN OP KARAKTERSTRINGS

6.1 Eisen voor een machine voor symboolverwerking

Naast de wijd verbreidde toepassing op het gebied van numerieke dataverwerking, wordt de digitale komputer veelvuldig gebruikt voor niet-numerieke dataverwerking, dat wil zeggen bewerkingen op symbolen.
Problemen van dit soort, zoals simulatie, opzoeken en sorteren van alfanumerieke informatie, compileerprocessen en vele anderen vereisen ingewikkelde programmeertechnieken en zijn kostbaar wat software-ontwikkeling betreft, vanwege de eraan besteedde tijd.
Het uiteindelijk resultaat is vaak nog niet erg efficient.
De oorzaak is, dat een komputer, oorspronkelijk bedoeld voor numerieke rekenkundige en logische bewerkingen, toegepast wordt voor niet-numerieke bewerkingen.
Alle huidige symbool-verwerkingstechnieken voor een general purpose komputer komen er op neer dat een komplexe vertaling wordt uitgevoerd van de symbool-verwerkingsfunkties naar een verzameling equivalente rekenkundige operaties.
Er zijn een aantal gespecialiseerde talen voor symbool-manipulatie ontstaan, zoals LIST en STRING.
Deze talen stellen de gebruiker in staat om kunstmatig het geheugen van de komputer te herstruktureren, zodat dit de voorstelling van een meer algemene datastruktuur toelaat.
Een andere methode om de komputer efficienter deze problemen te laten verwerken, is het gebruik van een micro-programmeerbare komputer, met een aangepast microprogramma voor symboolmanipulaties.
Genoemde methoden veranderen echter niet de fundamentele inconsistentie van het gebruik van een 'numerieke' machine voor niet-numerieke operaties.

De eisen te stellen aan een symbool-verwerkings machine zijn:
1. Efficiente opslag van de symbolische data.
2. Veelzijdige manipulatiemogelijkheden van de opgeslagen data.
3. Efficiente communicatie in de machine zelf en door middel van de interface met de gebruiker.

6.2 Eigenschappen van data en manipulatiemogelijkheden.

De symbolische data heeft de volgende eigenschappen:
1. De kleinste ondeelbare eenheid is het symbool.
2. Het alfabet van symbolen moet groot genoeg zijn om alle alfanumerieke - en leestekens te kunnen omvatten.
3. De symbool-informatie wordt het beste opgeslagen in de vorm van strings.
4. Een symboolstring moet een dynamisch te veranderen lengte hebben.

De gebruiker dient een ruime manipulatie-mogelijkheid te hebben met de data. Dit houdt in:
met de data. Dit houdt in:

1. Het introduceren van nieuwe strings.
2. Het verwijderen van strings.
3. Het scheiden van strings.
4. Het samenvoegen van strings.
5. Het vergelijken van twee strings.
6. Invoegen van nieuwe data in bestaande strings.
7. Delen van strings weglaten.
8. Schuiven van strings.

Een machine die gekonstrueerd is om symboolmanipulaties zo optimaal mogelijk uit te voeren zal een ingewikkelde celstruktuur bezitten en ook de mogelijkheid dienen te hebben voor kommunikatie tussen de buurcellen onderling.

6.3 Symboolmanipulaties met een associatief geheugen.

Een associatief geheugen, zoals dat ontworpen is, heeft een struktuur, die het toelaat, dat enkele typen symboolmanipulaties sneller uitgevoerd kunnen worden dan met een konventioneel geheugen.
Zo kan onder andere snel worden nagegaan of een bepaalde substring in het geheugen aanwezig is.

Inhoud CAM				adres
a	l	f	a	0
b	e	t	,	1
a	u	t	o	2
m	o	b	i	3
e	l	,	a	4
u	t	o	f	5
a	b	r	i	6
e	k	,	,	7

fig 6.1 Inhoud van het associatief geheugen bij tekenverwerking. Een komma(,) stelt een blank voor.

Een nieuwe string kan worden tussengevoegd in plaats van een oude string, mits deze laatste niet korter is dan de nieuwe string.
Alfabetische woorden in het geheugen kunnen in alfabetische volgorde gesorteerd worden.
In dit geval is het nuttig in verband met het achtereenvolgens aanbieden van symbolen in het kenmerkregister om een ROM ter beschikking te hebben voor het uitlezen van de bitpatronen van de alfanumerieke tekens.

Het ontworpen associatief geheugen heeft een grootte van 1k80, dat wil zeggen, er kunnen 10 alfanumerieke tekens van 1 byte in een geheugenwoord worden opgeslagen.
Totaal kunnen dus 1024*10=10240 alfanumerieke symbolen in het geheugen worden opgeslagen.
Bij 60 regels per pagina en 60 tekens per regel betekent dat, dat er 10240/(60*60) = 2,8 pagina's met alfabetische tekens in het geheugen passen.

Fig 6.1 geeft een voorbeeld van een geheugen met een aantal alfabetische woorden, waarin een bepaalde substring moet worden bepaald.
Dit komt overeen met de procedure INDEX in PL/1.
Deze procedure bepaalt waar in een string een opgegeven substring voorkomt.
Een string wordt in natuurlijke volgorde op de opeenvolgende plaatsen van het associatief geheugen opgeslagen.
Bij het ontworpen geheugen gebeurt dit automatisch.

Er zal een summiere beschrijving worden gegeven van een methode om de procedure INDEX te realiseren met behulp van een associatief geheugen.
We willen bepalen of de substring 'autofabriek' in de string voorkomt.
We bieden daartoe achtereenvolgens aan:

kenmerk	associatie-adres	aktie
a u t o	2	adres 2 opslaan
f a b r	geen	adres 2 voldoet niet
, a u t	geen	
x , a u	geen	
x x , a	0,4	deze adressen opslaan
u t o f	5	dus adres 0 voldoet niet
a b r i	6	opslaan, volgt op 5
e k , x	7	opslaan, volgt op 6

Alle karakters zijn daarmee gelokaliseerd.
De totale substring beslaat meerdere geheugenwoorden; nagegaan zal dus moeten worden of de woorden die associeren inderdaad op opeenvolgende plaatsen staan, zoals dat hier gedaan is.

We vinden P = (adres van de eerste associatie van de substring) * aantal tekens per woord + de beginpositie van het eerste teken van het woord.
Dus P = 4 * 4 + 4 = 20.

Het vervangen van een substring door een andere.
Daartoe moet eerst de betreffende substring worden gezocht.
Dan worden de delen van de in te voegen substring op de plaats van de oude substring geschreven.
Dit schrijven gebeurt op adres, met don't cares op de plaatsen die ongewijzigd dienen te blijven.
Op de niet gebruikte symboolposities worden blanks (,) ingevuld.

7 MATRIX-BEWERKINGEN

7.1 Soorten matrixbewerkingen

De meest voorkomende matrixbewerkingen zijn:
- matrix-inversie
- bepalen van de eigenwaarden en eigenvektoren
- oplossen van vergelijkingen door middel van bewerkingen op de coefficienten van een matrix
- optellen van twee matrices
- vermenigvuldigen van twee matrices
- bewerkingen op schaars gevulde matrices

In het algemeen worden de elementen van een matrix in het geheugen van een komputer opgeslagen op een aantal opeenvolgende plaatsen (adressen), bijvoorbeeld rij na rij.
Het beginadres (referentie-adres) van de matrix wordt vastgesteld.
De toegang tot een bepaald element van de matrix vindt plaats door het adres van de plaats waar het element is opgeslagen te berekenen.
Uitgaande van een matrix M(1:m,1:n) met m rijen en n kolommen, moet om de waarde van het element M(i,j) uit het geheugen te kunnen uitlezen het adres worden bepaald als volgt: adres M(i,j):= ref-adres +(i-1)*n + j.
Iedere toegang tot een element vereist dus een dergelijke rekenkundige operatie.

7.2 Schaars gevulde matrices

Schaars gevulde matrices, dat wil zeggen matrices met weinig elementen ongelijk aan nul komen vaak voor bij problemen als:
- komputer aided design
- numeriek oplossen van differentiaalvergelijkingen
- grafentheorie
- netwerktheorie
- lineaire programmering

De dichtheid van een matrix is het aantal niet-nul elementen gedeeld door het totale aantal elementen.
De vulfaktor is de dichtheid in procenten.
De orde van een matrix wordt met I aangeduid.
Als grenzen van schaarse matrices stelt men ongeveer:
- I is van de orde 30 a 50
- de vulfaktor f is kleiner dan 15 a 25 %

De vulfaktor f neemt snel af naarmate I hoger is.
Als I < 50 maakt het weinig uit of de totale matrix in het geheugen wordt opgeslagen of alleen de elementen die niet nul zijn.

Bij I > 50 is het efficienter wat executie-tijd en geheugenruimte betreft om alleen de niet-nul elementen op te slaan.

Bij bewerkingen op schaarse matrices willen we optimaliseren met betrekking tot:
- de benodigde geheugenruimte
- het aantal benodigde rekenkundige operaties
- het behoud van de schaarsheid van de matrix
- de ontstane afrondfouten

Alle genoemde punten hangen nauw met elkaar samen.
Het zoveel mogelijk behouden van de schaarsheid is van belang in verband met de benodigde geheugenruimte en het aantal rekenkundige operaties.
Om de schaarsheid te behouden dient men de volgorde waarin de onbekenden worden beschouwd bij decompositie zodanig te kiezen, dat er zo weinig mogelijk vullers (fill-ins) bijkomen.
Behoud van schaarsheid is dan ook het voornaamste doel bij het manipuleren met schaarse matrices.
Zie daarvoor onder andere de referenties:119 t/m 124.

7.3 Gebruik van een associatief geheugen voor matrixbewerkingen

Zoals onder 7.1 beschreven wordt het adres van een matrixelement in een konventioneel geheugen bepaald door een rekenkundige operatie uit te voeren.
Als dit vaak moet gebeuren, zoals bij decompositie van grote matrices, dan zal veel komputertijd worden besteed aan genoemde adresberekeningen.
Hier kan een associatief geheugen met voordeel worden toegepast. In dat geval bevat het CAM de matrixnaam met de bijbehorende indices voor ieder element dat ongelijk aan nul is. Dit laatste is vooral van nut bij schaars gevulde matrices.
Bij een konventioneel geheugen moeten we de adressen van de elementen ongelijk aan nul op de een of andere wijzen linken als we alleen de elementen ongelijk aan nul willen opslaan.
Dit kost extra geheugenruimte.
Bovendien ontstaan bij decompositie extra elementen (vullers), die op een of andere wijze gelinked moeten worden met de reeds aanwezige elementen van de matrix.
Hier is het associatief geheugen dus in het voordeel.
Het aan het CAM gekoppelde RAM zal de waarden van de elementen moeten bevatten.
Fig 7.1 geeft aan hoe een matrix in het associatief geheugen kan worden opgeslagen.

Inhoud CAM			inhoud RAM
matrix naam	i	j	waarde
M	1	1	1
M	1	2	5
M	2	1	6
M	2	2	3
M	3	1	4
M	3	2	2

fig 7.1 Opslag van een matrix in het associatief geheugen

In het CAM hebben we een woordbreedte van 10 byte ter beschikking. Daarvan gebruiken we bijvoorbeeld 4 byte voor de matrixnaam en twee byte voor elk van de indices.
Als een matrix in het associatief geheugen is ingeschreven, kunnen we de waarden van de elementen snel uitlezen door als kenmerk aam het CAM M(i,j) aan te bieden en de bijbehorende waarde uit het RAM uit te lezen.
Als meerdere elementen na elkaar moeten worden uitgelezen zijn twee tellers nodig voor het ophogen van de i- en j-waarden.
Treedt na het aanbieden van een kenmerk een associatie op, dan kan het bijbehorende element worden uitgelezen.
Treedt geen associatie op, dan is het betreffende element blijkbaar nul. Het voordeel is dat deze methode geen adresberekeningen vergt, die nogal wat tijd kosten.

7.4 Optellen van twee matrices

Optellen van twee matrices geeft als resultaat een matrix waarvan elk element de som is van de overeenkomstige elementen van de oorspronkelijke matrices.
Het aantal elementen ongelijk aan nul van de sommatrix S is ten hoogste gelijk aan de som van het aantal elementen ongelijk aan nul van de oorspronkelijke matrices A en B.

We gaan uit van twee matrices A en B, waarvan alleen de elementen ongelijk aan nul in het geheugen worden opgeslagen.
We bieden X(i,j) als kenmerk aan (X stelt voor: don't cares op de plaats van de matrixnaam).
Als er een associatie optreedt, dan is of A(i,j) ongelijk aan nul, of B(i,j) ongelijk aan nul, of beiden ongelijk aan nul.
De gevonden waarde (van A(i,j) of B(i,j)) wordt in een register R1 opgeslagen: R1 := A(i,j) of R1 := B(i,j).
We weten op dit moment nog niet of er een of meer associaties zijn. Vervolgens wordt een TV-puls gegeven.

Er zijn nu drie mogelijkheden:

A. Er treedt nog een associatie op.
Dan is er nog een adresaanbod, en wel van B(i,j).
We tellen B(i,j) op bij R1 en slaan het resultaat op in het associatief geheugen. We doen dit op de plaats van A(i,j) en wel als volgt: In het CAM staat nog A(i,j), waarvan we het adres in een register Ra hebben opgeslagen.
Op dit adres in het RAM schrijven we A(i,j) + B(i,j).

B. Er treedt na de TV-puls geen associatie meer op.
Dan is of A(i,j) of B(i,j) ongelijk aan nul.
We gaan dan gewoon verder met de volgende stap en A(i,j) of B(i,j) met de bijbehorende waarde blijft op de oorspronkelijke plaats staan.

C. Een bijzonder geval dat onder a valt , is het geval dat A(i,j) - B(i,j) = 0.
Dit levert dus een nulelement op de plaats S(i,j) op.
We dienen dan A(i,j) en B(i,j), ofwel X(i,j) te wissen.
X geeft hier aan dat voor de naam van de matrix don't cares worden ingevuld.
Fig 7.2 geeft een voorbeeld van het optellen van twee matrices, waarbij de inhoud van CAM en RAM zijn weergegeven voor en na de optelling.

Voor optelling		na optelling		
CAM	RAM	CAM	RAM	somelement
A11	8	A11	11	S11
A12	6	A12	7	S12
A13	-2	A13	-2	S13
A14	1			wissen
A21	3	A21	5	S21
A23	8	A23	9	S23
A32	-7	A32	-7	S32
A33	4	A33	1	S33
A41	1	A41	5	S41
A44	-7	A44	-1	S44
B11	3	B11	3	
B12	1	B12	1	
B14	-1			wissen
B21	2	B21	2	
B22	4	B22	4	S22
B23	1	B23	1	
B31	1	B31	1	S31
B33	-3	B33	-3	
B41	4	B41	4	
B43	2	B43	2	S43
B44	6	B44	6	

fig 7.2 Inhoud van het associatief geheugen voor en na het optellen van twee matrices

Na afloop van het optelproces bevatten de elementen van de A- en de B-matrix de somelementen, zoals dat in fig 7.2 is aangegeven in de laatste kolom.
Transport vanuit het associatief geheugen naar het centrale geheugen van een komputer kan plaats vinden via een DMA-kanaal, waarbij de waarden van de elementen worden gevonden in het associatief geheugen door achtereenvolgens X11, X12, X13, X14, X21, ...aan te bieden en bij een associatie alleen het eerste woord dat associeert uit te lezen, dat wil zeggen de bijbehorende waarde van het element in het RAM. Indien bij het aanbieden van een kenmerk geen associatie optreedt, dan is het betreffende element van de sommatrix nul. Fig 7.3 geeft het flow-diagram voor het optellen van twee matrices.

7.5 Vermenigvuldigen van twee matrices

Om het element P(i,j) van de produktmatrix van twee matrices A en B te bepalen, worden de overeenkomstige elementen van rij i en kolom j met elkaar vermenigvuldigd, waarna de aldus verkregen produkten worden opgeteld.

De snelheidswinst die bij het vermenigvuldigen van twee matrices kan worden behaald bij gebruik van een associatief geheugen ligt weer in het overbodig zijn van de adresberekeningen. De velden van het data-register, die dienen voor de naam, de i-index en de j-index, dienen snel gewijzigd te kunnen worden. Bijvoorbeeld onder programma-besturing vanuit de general purpose komputer, waaraan het associatief geheugen is gekoppeld.
Bij deze eerste methode mogen de elementen van de matrices A en B in een willekeurige volgorde in het geheugen zijn opgeslagen. De elementen die nul zijn worden niet in het geheugen opgeslagen.

In ALGOL luidt het pseudo-algorithme voor het uitvoeren van een matrixvermenigvuldiging van twee matrices A(1:m,1:n) en B(1:n,1:m):

```
for i:=1 step 1 until m do
for j := 1 step 1 until m do
begin P(i,j) := 0;
      for t := 1 step 1 until n do
      P(i,j) := P(i,j) + A(i,t) * B(t,j)
end
```

Het CAM data register is verdeeld in drie velden, zoals in fig 7.5 is aangegeven.
Het algorithme geeft aan hoe bij matrixvermenigvuldiging de inhoud van deze velden gewijzigd moet worden:

1. De naam: deze wordt afwisselend A en B.
2. De eerste index wordt conform de algorithme gewijzigd.
3. De tweede index eveneens.

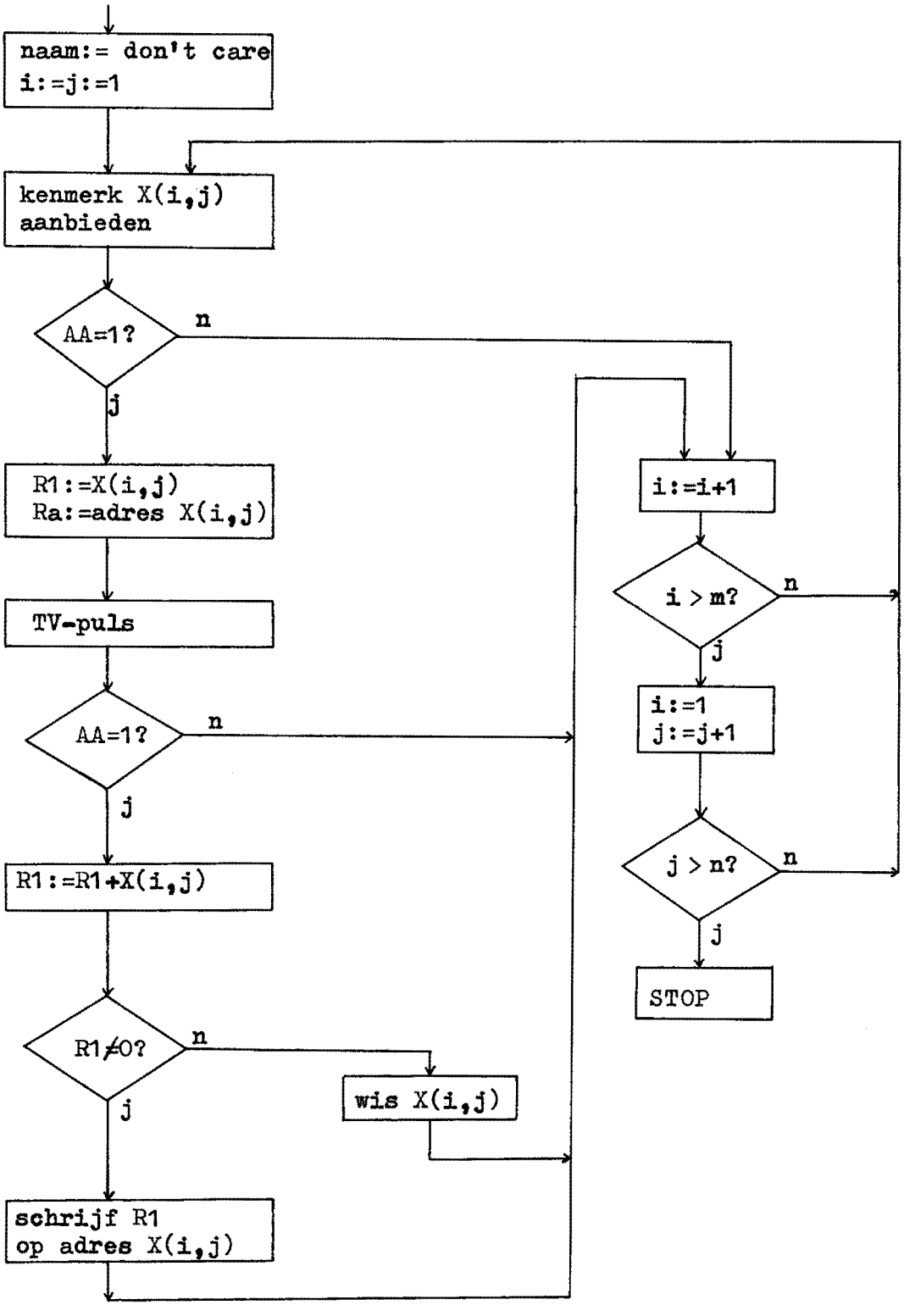


fig 7.3 Flow-diagram voor het optellen van twee matrices A(1:m,1:n) en B(1:m,1:n)

Als er bij het aanbieden van een kenmerk geen associatie optreedt, dan is het betreffende element blijkbaar nul, want elementen gelijk aan nul worden niet in het geheugen opgeslagen.

Een andere methode voor het uitlezen van de matrixwaarden voor het vermenigvuldigen van twee matrices verloopt op de volgende manier.
We gaan er daarbij van uit dat in het geheugen eerst de matrix A, kolom voor kolom of rij voor rij is opgeslagen, en vervolgens de matrix B, eveneens kolom voor kolom of rij voor rij.
Ook de elementen die nul zijn worden opgeslagen; deze methode is dus niet bedoeld voor het opslaan en bewerken van schaars gevulde matrices.
Het uitlezen verloopt dan sneller dan bij de vorige methode.
Het flowdiagram van fig 7.4 geeft aan hoe het proces verloopt.
We beginnen met het bepalen van het element P(1,1).
P(1,1)=A(1,1)*B(1,1)+A(1,2)*B(2,1)+ .... + A(1,n)*B(n,1).
Zoals het flowdiagram aangeeft beginnen we met i=1, j=1, p=0, M = A (matrixnaam). Als kenmerk wordt aangeboden in de drie velden: matrixnaam, i-index, j-index respektievelijk A, 1, x (x = don't care).
Er wordt geen TV-puls gegeven (p = 0).
We lezen op het adres, dat de besturing BC afgeeft in het RAM de waarde van het element A(1,1) uit.
Vervolgens wordt de matrixnaam B en we bieden aan: B, x, 1 .
Er wordt geen TV-puls gegeven en we kunnen nu B(1,1) uitlezen.
Dit gaat zo verder totdat A(1,n) en B(n,1) zijn gevonden.
De waarden van A(i,j) en B(i,j) die zo steeds worden gevonden worden met elkaar vermenigvuldigd en opgeteld bij de reeds gevonden subsom van het betreffende produktelement.
Deze laatste akties vinden plaats buiten het associatief geheugen en de snelheid van de totale matrixvermenigvuldiging zal dan ook sterk afhangen van de snelheid waarmee deze rekenkundige operaties uitgevoerd worden.

7.7 Bewerkingen op schaars gevulde matrices

Bij vele problemen komen schaars gevulde matrices voor (zie 7.2). In het algemeen zijn dit matrices met grote afmetingen.
Bij het uitvoeren van een decompositie van zo'n matrix ontstaan er extra elementen (vullers) op plaatsen, waar eerst een nul stond.
Zo kan het voorkomen, dat een matrix, die aanvankelijk nog in het associatief geheugen pastte, in de loop van het decompositie-proces te groot wordt om in het associatief geheugen op te slaan. Om toch het associatief geheugen te kunnen gebruiken kunnen we bepaalde rijen uit het associatief geheugen halen en wegschrijven in een konventioneel extern geheugen.
In het associatief geheugen dient dan een verwijzing voor te komen naar de betreffende rij, die in het konventioneel extern geheugen is weggeschreven.

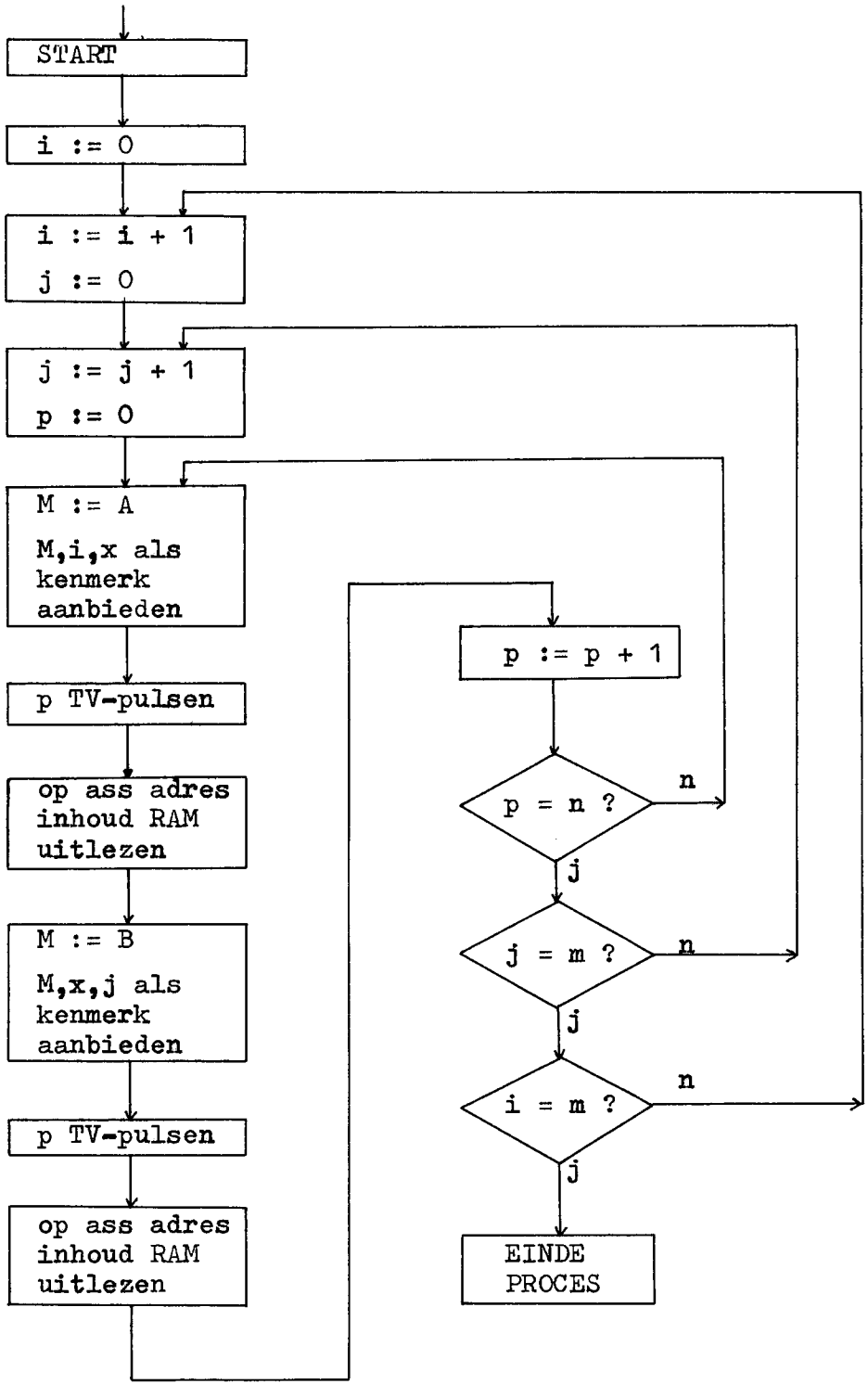


fig 7.4 Algorithme voor het uitlezen van de matrix-elementen van twee matrices A en B bij vemenigvuldiging.

De rij wordt in het extern geheugen opgeslagen volgens een gebruikelijke opslagmethode voor schaarse matrices in een konventioneel geheugen.

Stel, dat het in de loop van de decompositie aanbieden van een element aanleiding geeft tot het optreden van overfow bij het associatief geheugen.
Dan moeten de volgende akties worden ondernomen:

1. Alle elementen van de betreffende rij moeten worden uitgelezen en weggeschreven in een extern geheugen.
2. Op de eerste lege plaats in het associatief geheugen wordt een verwijzing gegeven naar het beginadres van de rij in het extern geheugen en wel als volgt: het kenmerk wordt A,i,0, waarbij de nul er op duidt, dat de betrefffende rij is weggeschreven (in plaats van een nul kan ook een andere indikatie in het betreffende veld worden gebruikt).

Op het aanbieden van A,i,0 geeft de besturing BC een adres voor het RAM af. Op deze plaats in het RAM wordt het referentie-adres opgeslagen van de rij die in het extern geheugen is weggeschreven.

Het uitlezen van een element van de matrix gaat als volgt:
We bieden A,i,j aan. Het resultaat kan zijn:

1. Er treedt een associatie op; dan is het gezochte element gevonden.
2. Er treedt geen associatie op.
   Dan A,1,0 aanbieden. Het resultaat kan zijn:
   A. Geen associatie, dan is element A(i,j) = 0.
   B. Wel associatie, dan kan via het adres uit het RAM de rij in het extern geheugen worden gevonden. Het element kan dan ook nog nul zijn.

Als de matrix van het begin af aan al niet in het associatief geheugen past, dan zal de beschreven methode van opslag van rijen in een extern geheugen al meteen moeten worden toegepast.
Fig 7.6 geeft aan hoe het associatief geheugen gebruikt kan worden bij bewerkingen op matrices.

Zoals in 7.2 reeds is opgemerkt is behoud van schaarsheid een zeer belangrijke zaak bij bewerkingen op schaarse matrices. Het gebruikt van een algorithme dat daarop is gericht is dan ook van het grootste belang.

Het is moeilijk te voorspellen, in hoeverre het gebruik van een associatief geheugen van nut is voor gebruik bij matrixbewerkingen.
Dit hangt sterk af van de snelheid van het datatransport tussen de komputer en het associatief geheugen, dus van de wijze van koppeling van het geheugen aan de komputer.

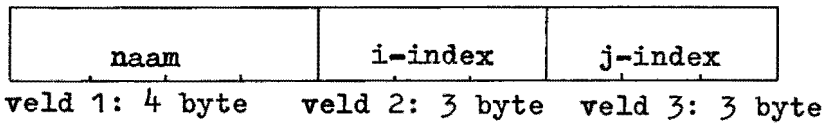


fig 7.5 Verdeling van het geheugen in drie velden ten behoeve van matrixbewerkingen

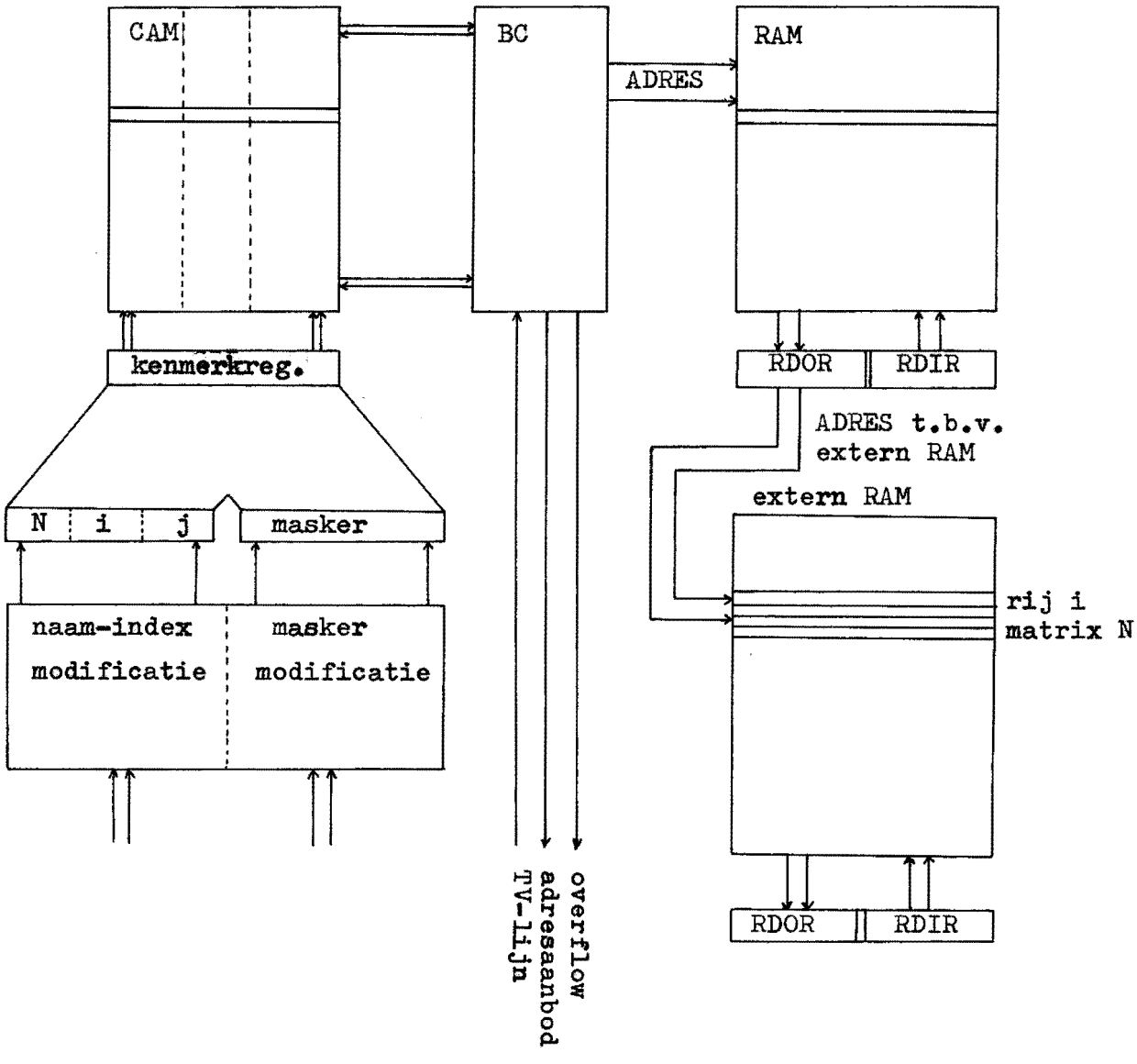


fig 7.6 Methode voor het opslaan van schaars gevulde matrices in het associatief geheugen, waarbij tevens een extern RAM wordt gebruikt

8 SIMULATIE- EN NETWERKPROBLEMEN

8.1 Simulatieproblemen

Bij vele simulatieproblemen moet na iedere tijdsstap worden nagegaan of een aktie moet worden ondernomen en welke. De tijdstippen en de bijbehorende akties zijn in een geheugen opgeslagen. Omdat op een willekeurig tijdstip tot een aktie op enig later tijdstip kan worden besloten, zullen de tijdstippen niet in volgorde in het geheugen zijn opgeslagen.
Om toch zo snel mogelijk aktie te kunnen ondernemen als dat nodig is, kan een associatief geheugen worden toegepast.

Een voorbeeld van zo'n simulatieprobleem is het simuleren van een telefooncentrale.
De vraag bij deze simulatie is na te gaan welke bezetting in een model van een telefooncentrale gemiddeld optreedt bij een stochastische verdeling van:

1. De oproeptijden.
2. De houdtijden (gespreksduren).

Bijgehouden moet worden:

1. Welke abonnees in gesprek zijn
2. Over welke lijn
3. Hoe lang

Op het moment dat de verbinding tot stand komt wordt de gespreksduur geloot (dus het afvalmoment staat dan vast).
De lijnnummering is van 1 tot en met N, met N < 1024.

De lijnbezettingsgegevens zijn:

1. Of er via de lijn gesproken wordt
2. Zo ja, door welke abonnee
3. Tot welk afvalmoment

Fig 8.1 geeft het flowdiagram weer voor het verloop van de simulatie.

In fig 8.2 is een voorbeeld gegeven hoe een en ander met behulp van een associatief geheugen gerealiseerd kan worden.

We gaan uit van een centrale met N lijnen.
Op t<0 zijn alle tot dan toe gelootte abonnees, oproepmomenten en afvalmomenten in het geheugen opgeslagen.
De abonneenamen zijn ook in het bijbehorende RAM opgeslagen.
De indikatie-bits (I-bits) in het RAM zijn nul.
Op t=0 wordt de klok gestart in de stand 0.
Dan gebeurt het volgende:

1. Deze stand wordt als deelkenmerk aan veld 3 van het associatief geheugen aangeboden.

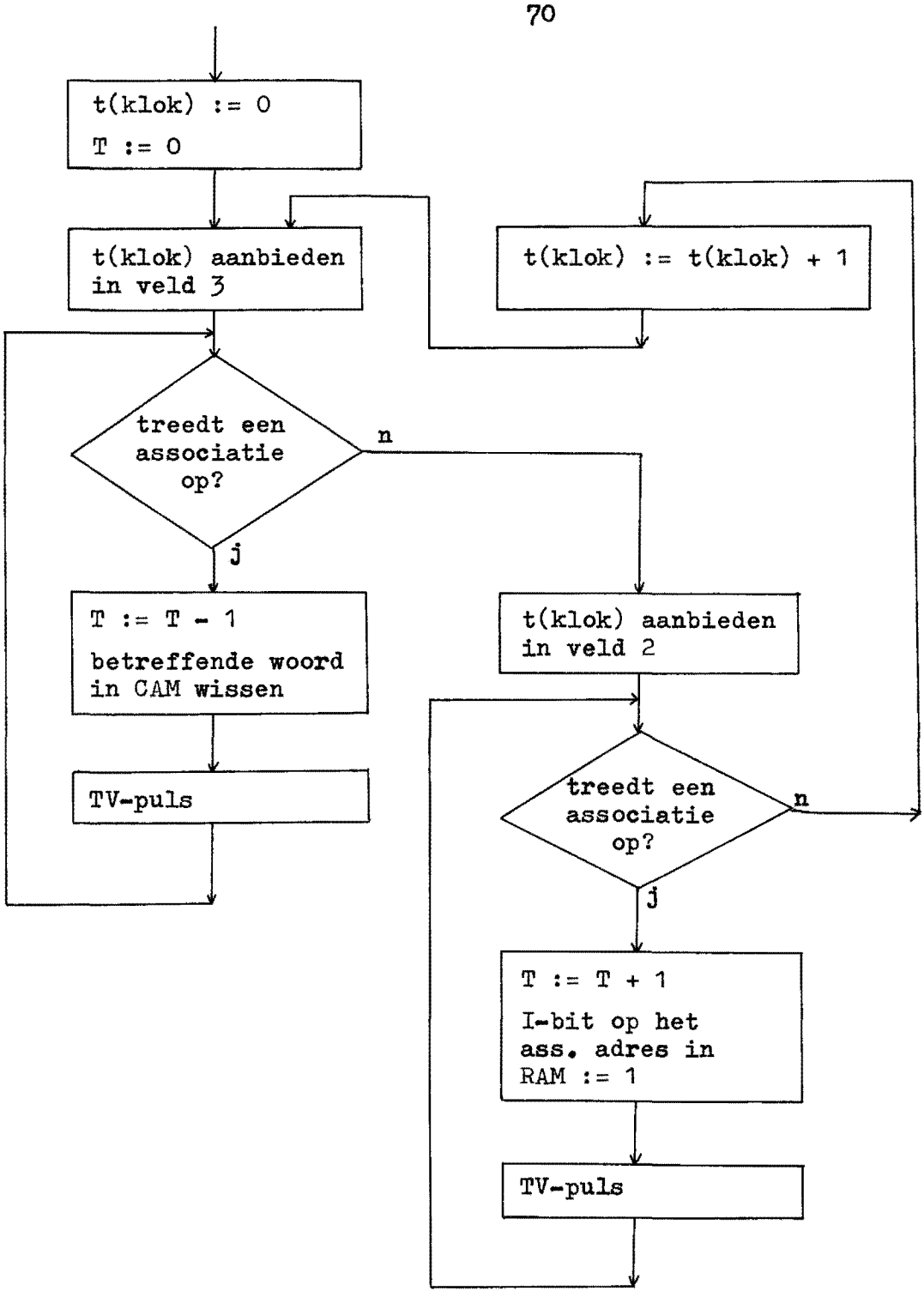


fig 8.1 Flow-diagram voor de simulatie van een telefooncentrale.
T is de stand van de up-down teller.
t(klok) is de stand van de klok.

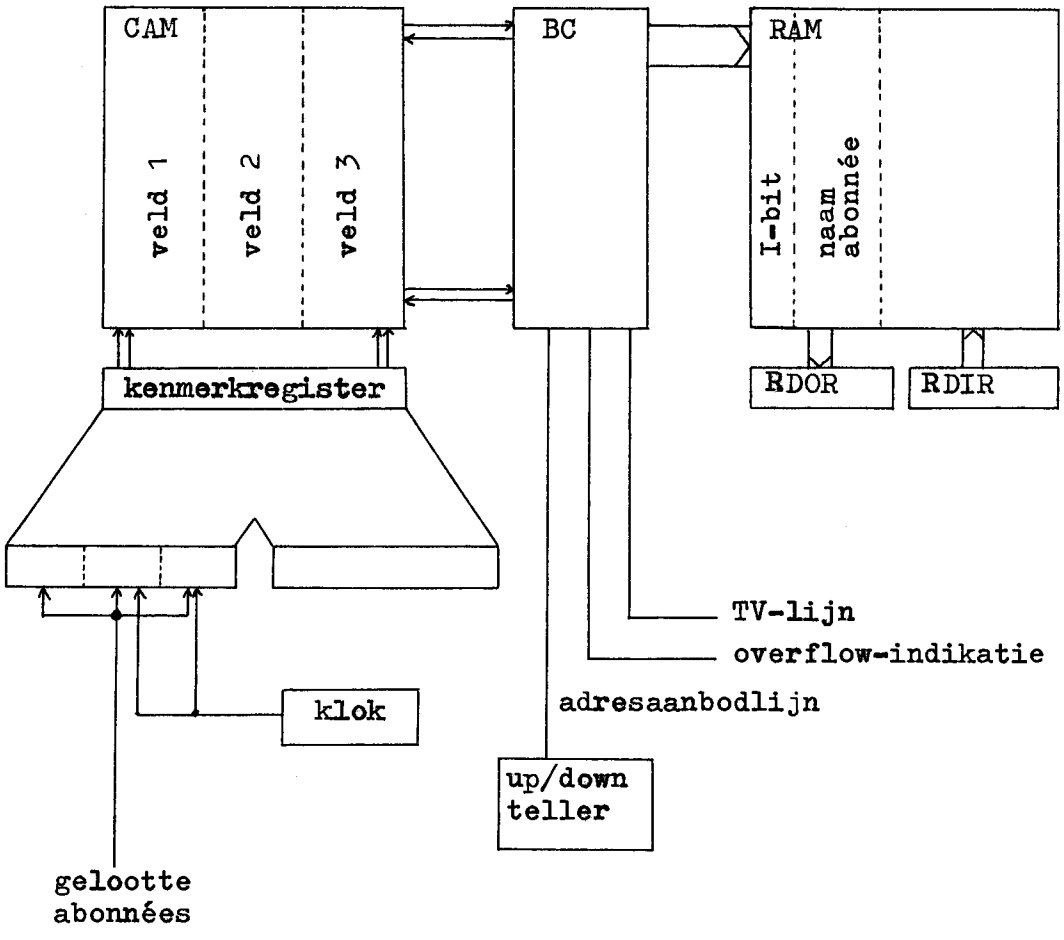


fig 8.2 Associatief geheugen voor de simulatie van een telefooncentrale.

veld 1: naam abonnée
veld 2: oproepmoment
veld 3: afvalmoment

In het RAM: Indikatiebit en naam abonée.

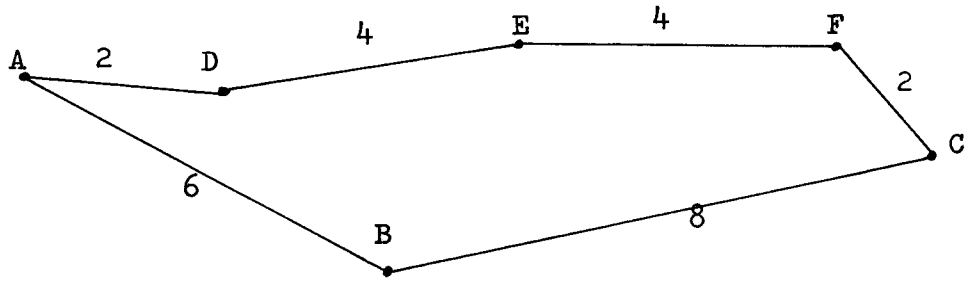


fig 8.3 Netwerk voor het bepalen van de kortste route bij gegeven knooppunten en onderlinge afstanden.

Alle woorden met een afvalmoment op het betreffende tijdstip zullen associeren en worden met een wisopdracht aan de besturing van het CAM gewist. In het geval dat t=0 is zullen nog geen woorden gewist behoeven te worden.
De up-down teller wordt verlaagd met de waarde die het aantal te verbreken verbindingen aangeeft.

2. Vervolgens wordt de stand van de klok aan veld 2 aangeboden, om na te gaan of er nieuwe abonnees bijkomen op het betreffende tijdstip.
Zo ja, dan worden de bijbehorende I-bits in het RAM gelijk aan 1 gemaakt. De up-down teller wordt opgehoogd met de waarde, die het aantal nieuwe abonnees aangeeft.
Daarna wordt de stand van de klok met 1 opgehoogd en wordt weer met 1 begonnen.

Het voordeel van het gebruik van een associatief geheugen is in dit geval, dat we in een geheugencyclus alle abonnees met een bepaald oproepmoment verkrijgen en in een andere cyclus alle abonnees met een bepaald afvalmoment.
Er zijn dus geen langdurige zoekprocedures nodig.
Verschillende gegevens die men wil bijhouden kan men in een extern geheugen opslaan.
Het zou te ver voeren om daar op in te gaan.

8.2 Kortste route in een netwerk

Gegeven is een aantal punten en een netwerk dat deze punten onderling verbindt.
De onderlinge samenhang van het netwerk is gegeven als volgt: in een tabel is van elk punt gegeven, welke de buurpunten zijn van het betreffende punt en op welke afstand deze liggen.
Onder buurpunten wordt verstaan: twee punten die door een lijn met elkaar zijn verbonden zonder dat deze lijn andere punten passeert.

Om van een beginpunt A een eindpunt B te bereiken zijn er een groot aantal mogelijkheden.
Als de kortste route van A naar B loopt via punt C, dan bestaat deze route uit de kortste route van A naar C, gevolgd door de kortste route van C naar B.
Als van het beginpunt A bekend is langs welk buurpunt de kortste route naar B loopt, dan is dit gegeven voldoende om de kortste route van A naar B te bepalen, mits dit gegeven bekend is voor alle punten, die dichter bij B liggen dan bij A.

Bij het uitvoeren van het proces wordt er van uitgegaan, dat aan het associatief geheugen een RAM is gekoppeld.
Ieder woord van CAM en RAM bevatten:
- een indikatiebit
- de naam van een punt
- de naam van een buurpunt
- de afstand tussen de twee genoteerde punten

Dit wordt in het geheugen opgeslagen, zoals hieronder is aangegeven:

CAM				RAM			
0	P1	P2	d12	0	P1	P2	d12
0	P1	P4	d14	0	P1	P4	d14
0	P1	P5	d15	0	P1	P5	d15

fig 8.4 Wijze van opslag in het geheugen van de tabel met de onderlinge afstanden van de punten.

Voor het zoeken van de kortste route is het niet nodig, dat de punten in een bepaalde volgorde in het geheugen zijn opgeslagen. Als het indikatiebit nul is duidt dat op de corspronkelijke punten met hun afstanden.
Voor de gegevens die tijdens en na de bewerkingen in het geheugen worden opgeslagen wordt als indikatiebit een 1 ingeschreven.

Een voordeel van deze methode is verder, dat er weinig geheugenruimte nodig is in vergelijking tot andere methoden. Stel we hebben n punten met gemiddeld p verbindingen per punt met andere punten. Dan zijn daarvoor n*p geheugenplaatsen nodig. Gedurende de bewerkingen komen er hoogstens nog n geheugenplaatsen bij; totaal dus n(p+1) geheugenplaatsen nodig.

Het proces verloopt ook op de juiste wijze als de kortste route van P1 naar Pi via verschillende punten loopt, terwijl een langere route via een kleiner aantal punten loopt.

Het verloop van het proces, zoals dat met behulp van het associatief geheugen uitgevoerd kan worden is in het flowdiagram van fig 8.5 weergegeven.

Tabel 8.6 geeft een voorbeeld van het bepalen van de kortste route tussen de punten A en C van het netwerk van fig 8.3.
De corspronkelijke tabel heeft de indikatiebit 0; de nieuwe tabel krijgt de indikatiebit 1.
Gedurende het proces worden ook de gegevens bepaald, die zijn aangegeven met een x er achter.
Deze gegevens worden echter niet in het geheugen opgenomen, omdat het geen minimale afstanden zijn.

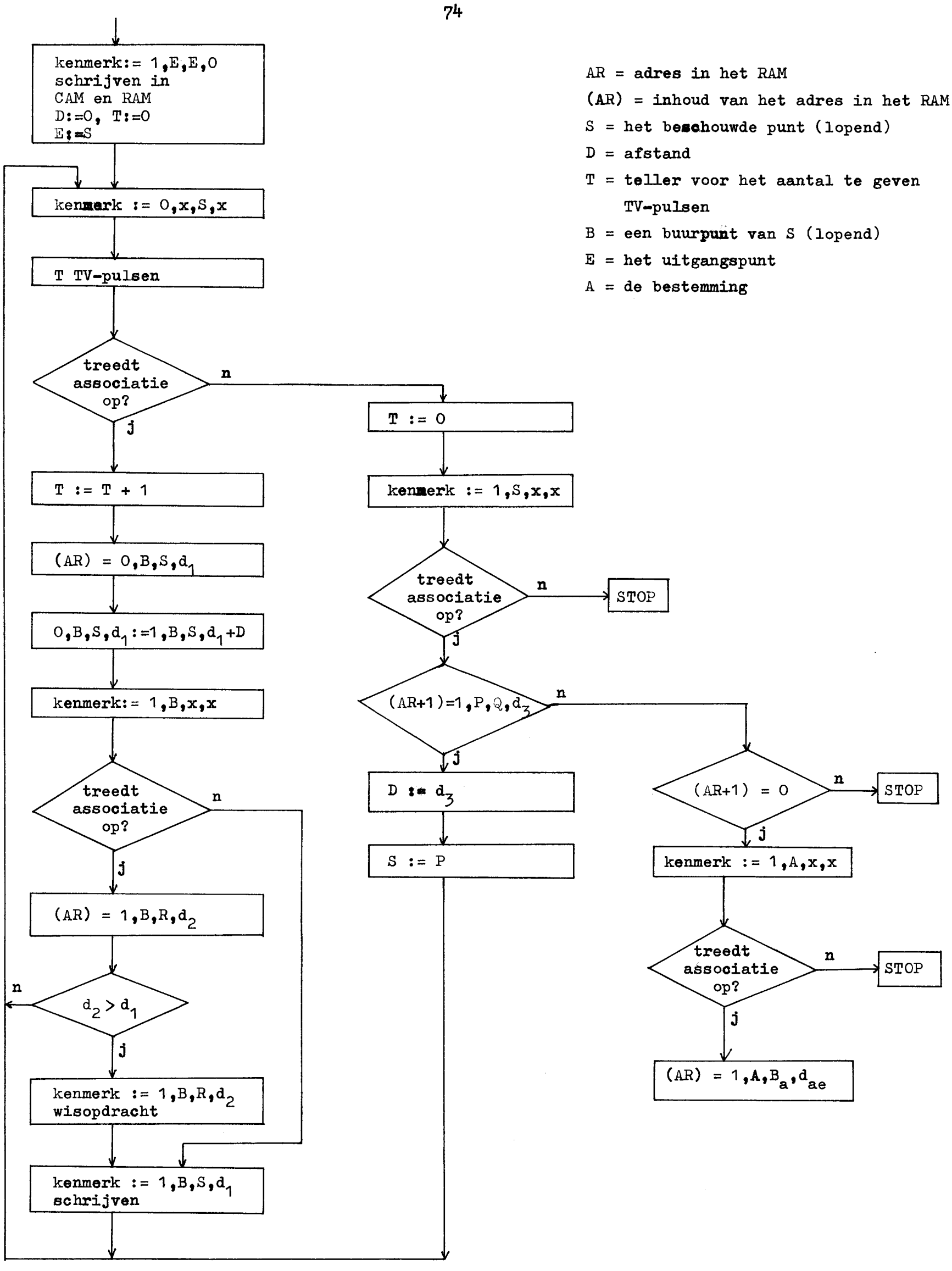


Fig 8.5 Flowdiagram voor het zoeken van de kortste weg tussen twee gegeven punten in een netwerk.

Inhoud RAM en CAM

indikatiebit	punt	buurpunt	afstand	
0	D	E	4	
0	B	C	8	
0	E	F	4	
0	C	F	2	
0	A	D	2	
0	E	D	4	
0	B	A	6	
0	C	B	8	
0	F	C	2	
0	D	A	2	
0	A	B	6	
0	F	E	4	
1	C	C	0	
1	B	C	8	
1	F	C	2	
1	C	B	16	x
1	A	B	14	x
1	E	F	6	
1	C	F	4	x
1	B	A	20	x
1	D	A	16	x
1	D	E	10	
1	F	E	10	x
1	A	D	12	
1	E	D	14	x

tabel 8.6 Inhoud van de verschillende velden in RAM en CAM bij het bepalen van de kortste route in een netwerk. De gegevens met indikatiebit gelijk aan nul staan oorspronkelijk in het ge heugen; de gegevens met indikatiebit 1 komen er bij gedurende het proces.

Als het proces beeindigd is, kan uit het associatief geheugen en uit het RAM de kortste weg worden uitgelezen. We lezen daartoe alle woorden uit met een indikatiebit gelijk aan 1 (uitlezen uit RAM) en vinden dan alle punten, via welke de kortste route van A naar C loopt.

8.3 Bepalen van de minimale boom, die n punten onderling verbindt

Gegeven zijn n punten die onderling verbonden moeten worden door verbindingen van punt tot punt, die we takken zullen noemen, zodanig dat de som van de lengtes van alle takken minimaal is.
De punten zijn gegeven door hun naam: P1, P2,....,Pn, en de bijbehorende koordinaten (x1,y1), (x2,y2),.....,(xn,yn). Het is ondoenlijk om alle mogelijke bomen te beschouwen, die de n punten verbinden. Dit aantal is namelijk n tot de macht (n-2). Er is een methode om dit probleem op te lossen, waarbij we een nuttig gebruik kunnen maken van een associatief geheugen.
Het proces van het bepalen van de minimale boom verloopt als volgt:

We verdelen de n punten in twee groepen G1 en G2. Voor de aanvang van het proces bevat groep G1 alle punten. De groep G2, aanvankelijk leeg, bevat in een bepaalde fase van het proces m punten, die door m-1 takken onderling zijn verbonden. De genoemde m-1 takken zijn de kortste verbindingen tussen deze m punten (ze vormen een minimale subboom).
De andere groep G1 bevat de overige m-n punten.
In fig 8.7 zijn de groepen G1 en G2 weergegeven; de punten van G2 vormen de minimale subboom.
In een aantal stappen worden de punten van G1 aan G2 toegevoegd. Als m=n, dan is het proces voltooid en is de minimale boom gevonden.
Als Pl een punt is van G1, dan is de afstand van Pl tot G2 gedefinieerd als de afstand van Pl tot het dichtstbij gelegen punt van G2. In de uiteindelijke boom is minstens 1 punt uit G2 verbonden met 1 punt uit G1.
Als we de afstanden van alle P's uit G1 tot G2 berekenen, dan is van elke P uit G1 bekend welke zijn kortste afstand is tot de minimale subboom. Voor de n-m punten uit G1 krijgen we dus n-m afstanden tot G2.
Van deze n-m afstanden moet de minimale afstand bepaald worden. De bij de kortste afstand behorende punten Pk in G2 en Pl in G1 zijn de punten die dan door een tak moeten worden verbonden. Het gevonden punt Pl wordt daardoor aan de minimale boom toegevoegd, die dan uit m+1 punten bestaat. Voor de overige n-(m+1) punten die nog tot G2 behoren, moet nu worden nagegaan of hun afstand tot G2 is gewijzigd door het toevoegen van punt Pl aan de minimale boom.
Als er afstanden gewijzigd zijn, dan worden de oorspronkelijke afstand en de bijbehorende punten Pi in G2 en Pj in G1 vervangen door de nieuwe afstand en de nieuwe punten.
Het proces begint dan weer van voor af aan, zoals beschreven is.

Het blijkt, dat het aantal bewerkingen wel afhangt van de plaats van het beginpunt. Hoe dat moet worden gekozen voor een optimaal resultaat is niet bekend; het beste lijkt een punt in het meetkundige midden van de puntenverzameling te voldoen.

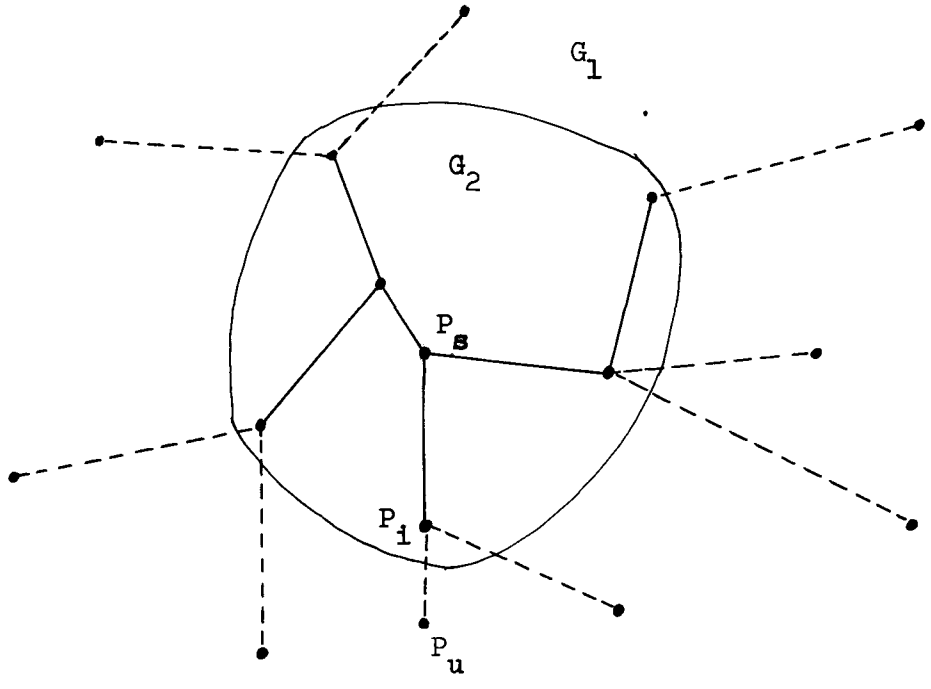


Fig 8.7 Verdeling van de punten waartussen de minimale boom gevonden moet worden in twee groepen G_1 en G_2 in een bepaalde fase van het proces. De punten van G_2 vormen de minimale subboom.

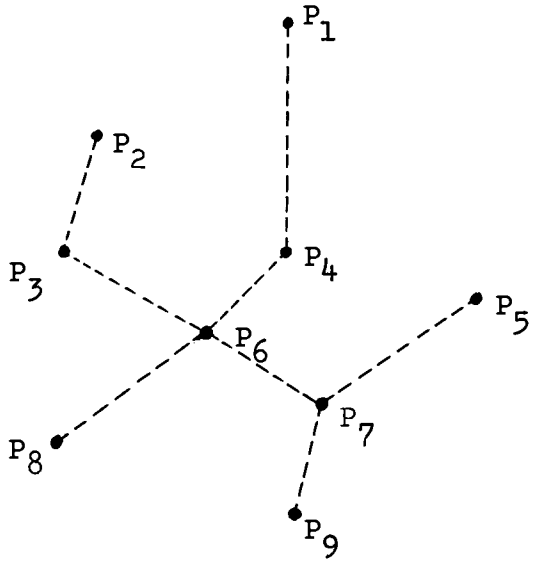


Fig 8.8 Negen punten die moeten worden verbonden door een minimale boom. Als startpunt wordt P_6 genomen.

In fig 8.8 is een verzameling van 9 punten gegeven. Gevraagd wordt de minimale boom te bepalen, die deze punten onderling verbindt.

In fig 8.9 is in een aantal kolommen telkens de geheugeninhoud van het associatief geheugen weergegeven.
Als uitgangspunt werd P6 gekozen. De ondelinge afstanden van de punten worden berekend uit de koordinaten van de punten.
De eerste kolom geeft de oorspronkelijke geheugeninhoud, waarbij alle indikatiebits nul zijn.
De ondelinge afstanden tussen de punten zijn zowel in het CAM als in het RAM opgeslagen.
We kunnen van de afstanden in het CAM snel het minimum van alle afstanden bepalen.
Zo zien we dat de minimale afstand in de eerste kolom (veld 4) gelijk aan 3 is; dit is de afstand P4,P6.
In plaats van 0,P4,P6,3 wordt nu op de vierde plaats in het CAM 1,P4,P6,3 opgeslagen.
De 1 als indikatiebit duidt aan dat P4,P6 een tak is van de minimale boom. Vervolgens wordt gekontroleerd of de afstand van enig ander punt tot deze minimale subboom (1 tak in dit geval) is afgenomen. Dit is het geval voor punt P1, want afstand P1,P6 is 8, echter afstand P1,P4 is 6.
Eveneens is afstand P5,P6 gelijk aan 7 en afstand P5,P4 is 5. De twee nieuwe afstanden worden in het geheugen opgenomen (zie fig 8.9, kolom 2, veld 4).
Nu wordt weer het minimum bepaald van de afstanden, echter alleen voor die gevallen, waarbij het indikatiebit nul is.
De wijzigingen, die zo optreden, zijn achtereenvolgens in de verschillende kolommen van fig 8.9 aangegeven.
Kolom 10 geeft de uiteindelijke inhoud van het CAM na afloop van het proces.

Het voordeel van het toepassen van een associatief geheugen voor dit proces is:
- van een (groot) aantal afstanden kan snel de minimale afstand worden bepaald.
- de benodigde geheugenruimte blijft gelijk in de loop van het proces.

Tenslotte geeft fig 8.10 het flow-diagram, dat aangeeft hoe het proces verloopt.

0,P1,P6,8	0,P1,P4,6			
0,P2,P6,6		0,P2,P3,3	1,P2,P3,3	
0,P3,P6,4		1,P3,P6,4		
0,P4,P6,3	1,P4,P6,3			
0,P5,P6,7	0,P5,P4,5			0,P5,P7,5
0,P7,P6,4				1,P7,P6,4
0,P8,P6,5				
0,P9,P6,5				0,P9,P7,3

			1,P1,P4,6	1,P1,P4,6
				1,P2,P3,3
				1,P3,P4,6
				1,P4,P6,3
	1,P5,P7,5			1,P5,P7,5
				1,P7,P6,4
		1,P8,P6,5		1,P8,P6,5
1,P9,P7,3				1,P9,P7,3

Fig 8.9 Geheugen-inhoud bij het bepalen van de minimale boom die de punten van fig 8.8 verbindt. In de opeenvolgende kolommen zijn telkens de wijzigingen aangegeven. De laatste kolom geeft het uiteindelijke resultaat.

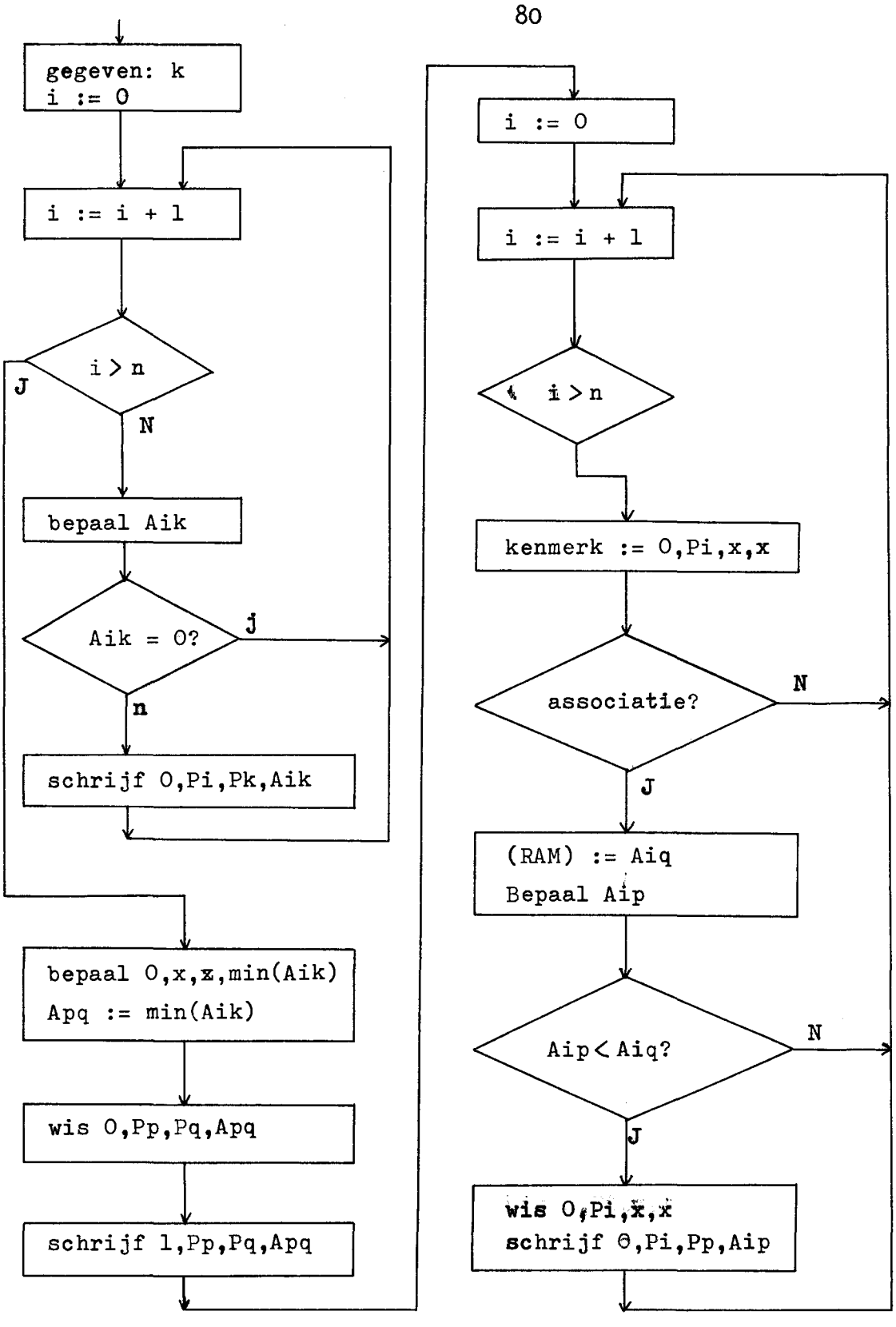


Fig 8.10 Flowdiagram voor het bepalen van de minimale boom die n gegeven punten verbindt.

9 DYNAMISCHE GEHEUGENTOEWIJZING

9.1 Het virtuele geheugen principe

Bij het opstellen van een programma heeft de programmeur een abstrakt fiktief werkgeheugen voor ogen (de adresruimte) waarin hij met de adressen van zijn programma opereert.
Daarna pas moet het gehele programma aan de beperkte mogelijkheden van de beschikbare komputer worden aangepast.
Het principe van het virtuele geheugen maakt dit laatste overbodig. De variabelen en velden van het programma bestaan voor de gebruiker voortaan alleen abstrakt; de realisatie ergens in de komputer is bij een virtueel geheugensysteem niet meer zijn zorg, maar de zorg van de komputer zelf.

We dienen een duidelijk onderscheid te maken tussen adresruimte en geheugenruimte.

- de adresruimte A
  A is de hoeveelheid adressen van een programma, dus juist die fiktieve ruimte, die de programmeur voor ogen heeft, als hij zijn programma opzet, d.i.het virtuele geheugen.

- de geheugenruimte G
  G is de hoeveelheid reeel beschikbare geheugenplaatsen, dus het fysisch beschikbare werkgeheugen.

9.2 Adresmodifikatie

Opdat een programma verwerkt kan worden moet het door het programma bezette deel van de adresruimte A in de geheugenruimte G worden afgebeeld (zie fig 9.1).

Er moet dus een afbeelding f:A in G gerealiseerd worden, die aan ieder adres a in A precies een geheugenplaats a' in G toewijst.
Natuurlijk moet het operating systeem om f te kunnen realiseren, de adresruimte A kunnen vinden: A is in een snel en voldoende groot extern geheugen opgeborgen.

De grootte N (dit is het aantal woorden) van de adresruimte A is in de regel een veelvoud van de grootte M van de geheugenruimte G.
De woordstruktuur is hetzelfde voor A en G.

Het 'working set' principe:
Dit is het belangrijkste principe van het virtuele geheugengebruik: van ieder programma dienen tijdens de verwerking op ieder moment alleen de op dat moment aktieve instrukties en data in het werkgeheugen aanwezig te zijn.

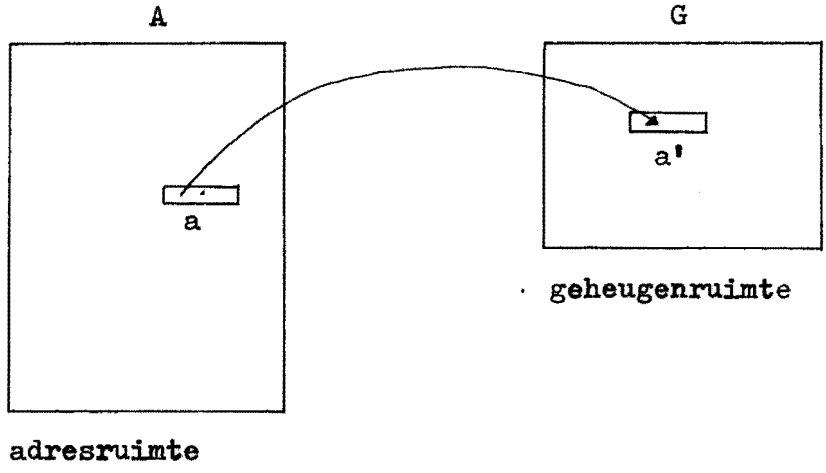


fig 9.1 Afbeelding f van een adres a van de adresruimte A in de geheugenruimte G.

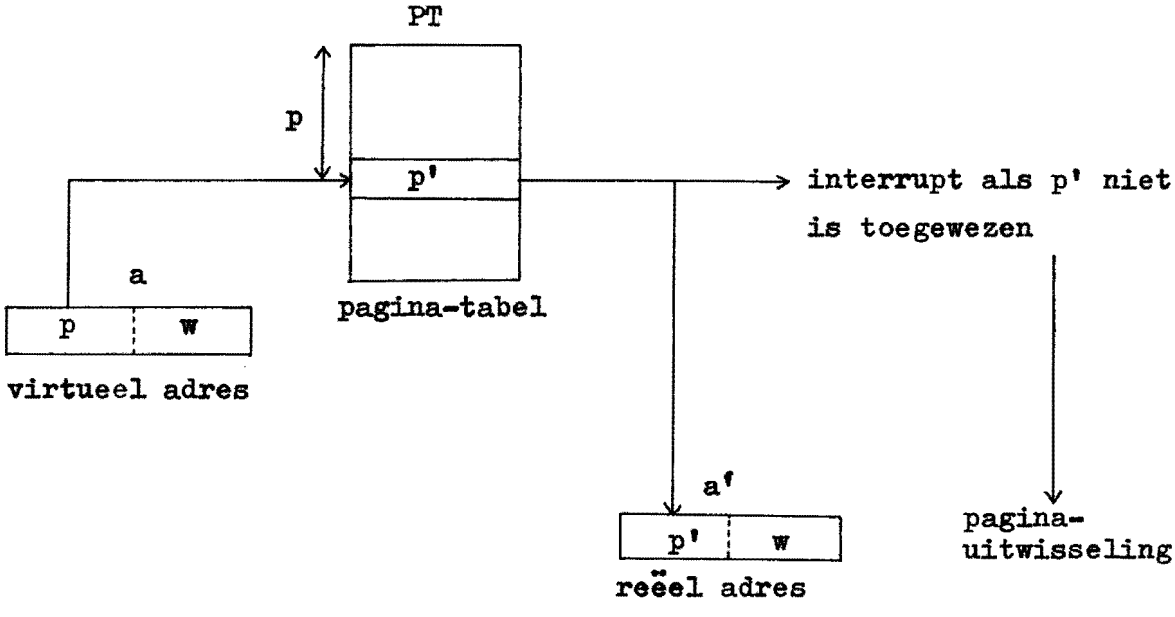


fig 9.2 Adresmodifikatie bij een virtueel geheugensysteem met behulp van pagina-tabellen

Dat dit doel in de praktijk ook bereikt wordt, is niet altijd zeker. Het vaststellen van de 'working set' is het belangrijkste probleem bij het gebruik van een virtueel geheugen systeem. Men tracht het aantal toegangen tot het externe geheugen zo klein mogelijk te houden, omdat de toegangstijd van het externe geheugen de kritische grootheid is.

Men deelt de adresruimte A op in n pagina's met een vaste lengte van d woorden en de geheugenruimte G wordt verdeeld in m frames van dezelfde lengte d.
Een virtueel adres a in A bestaat dan uit een paginanummer p en een woordnummer w.
Er geldt: $a = (p,w)$ $0<p<n$, $0<w<d$.
De reele geheugenplaatsen a' in G zijn overeenkomstig opgebouwd: $a' = (p',w)$ $0<p'<m$, waarbij p' het framenummer is.
De adresmodifikatie f:A naar G zorgt nu alleen voor een toewijzing van de pagina's en de frames: $a=(p,w)$; $f(a)=a'=(p',w)$.
Het woordnummer w blijft ongewijzigd; f is dus een pagina-gewijze identieke afbeelding van A in G.
Het uitwisselen van pagina's tussen extern geheugen en werkgeheugen staat bekend onder de naam paging.

Voor de realisatie heeft het systeem de volgende lijsten nodig:

1. Een frame-tabel FT
   Deze bevat m plaatsen voor de m ter beschikking staande frames.
2. Paginatabellen PTi
   Voor ieder programma i is er een tabel met n plaatsen. Deze tabellen zijn op het paginanummer p in oplopende volgorde genummerd en bevatten op de p-de plaats een framenummer p', indien de pagina p van het betreffende programma op dat moment in het frame p' aanwezig is.

Het adresmodifikatie-algorithme verloopt dan als volgt:
Als programma i aktief is, dan stelt het Operating System voor het uitvoeren van iedere instruktie van i uit de paginatabel PTi door een eenvoudige adresberekening de geheugenplaats a' vast, die uitgelezen moet worden.
In veel gevallen zal nog geen frame p' toegewezen zijn en zal in PTi dus de gezochte frame-aanduiding ontbreken.
Dan moet het programma i onderbroken worden en moet een paginauitwisseling plaats vinden.
De algorithmes, die deze pagina-uitwisseling verrichten beinvloeden de efficiency van het virtuele geheugenkoncept zeer sterk.
Omdat iedere adresmodifikatie een geheugencyclus gebruikt, wordt daardoor alleen al de verwerkingstijd van het programma verdubbeld, afgezien nog van de lange tijden die nodig zijn voor pagina-uitwisseling.

9.3 Reduktie van tabellen

Het gebruik van een virtueel geheugen volgens het beschreven koncept vereist teveel ruimte voor tabellen zonder dat deze tabellen ook werkelijk helemaal gebruikt worden.
Daarom wordt een aantal pagina's samengevat tot een segment, dat wil zeggen van de tot nu toe gebruikte paginanummers wordt een segmentnummer afgesplitst en een virtueel adres heeft dan de vorm : a = (s,p,w), waarbij s het segmentnummer aangeeft, en p het paginanummer binnen het segment s aangeeft.
Voor de adresmodifikatie heeft men voor ieder programma i een segmenttabel STi nodig, waarin de beginadressen van de paginatabellen PTis (behorend bij segment s) zijn opgeslagen.
Deze organisatie geeft een aanmerkelijke besparing van plaatsruimte.
Deze methode van segmentvorming is iets anders dan de zogenaamde segmentering die men ook bij paging aantreft.
Daarbij worden programma's verdeeld in logische eenheden, die eveneens segmenten worden genoemd.

9.4 Adresmodifikatie met een associatief geheugen

Om bij adresmodifikatie tijd te sparen, kan men gebruik maken van een associatief geheugen, waarvan alle geheugenwoorden snel en tegelijkertijd met een bepaald kenmerk vergeleken kunnen worden.
Met een associatief geheugen kan de adresmodifikatie tegelijk met de dekodering van de instrukties plaats vinden binnen een cyclus, dus zonder tijdverlies, omdat het allebei hardwareprocessen zijn.
Iedere adresmodifikatie loopt nu uitsluitend via het CAM en voor ieder frame van de geheugenruimte G is er een woord in het CAM beschikbaar.
Het virtueel adres is nu verdeeld in een paginagedeelte (p) en een woordgedeelte (w).
Het paginagedeelte wordt via het kenmerkregister aan het CAM aangeboden. Als de betreffende pagina in een frame van de geheugenruimte G is opgeslagen, dan is p in het CAM aanwezig.
Er treedt dan bij het aanbieden van p in het kenmerkregister een associatie op, wat aangegeven wordt door een aktieve adresaanbodlijn en tevens verschijnt het bij p behorende reele adres p' op de adresuitgang van de besturing BC.
Dit adres p' gekoncateneerd met het woordgedeelte w vormt het totale reele adres, via welk de betreffende plaats in het werkgeheugen kan worden geadresseerd.
Fig 9.3 geeft de gang van zaken weer bij associatieve adresmodifikatie.
Is p niet in het associatieve geheugen te vinden, dan treedt een interrupt op en een 'page-handler' verzorgt dan een paginauitwisseling.
Er zijn verschillende methoden bekend, waarop deze paginauitwisseling kan plaatsvinden.

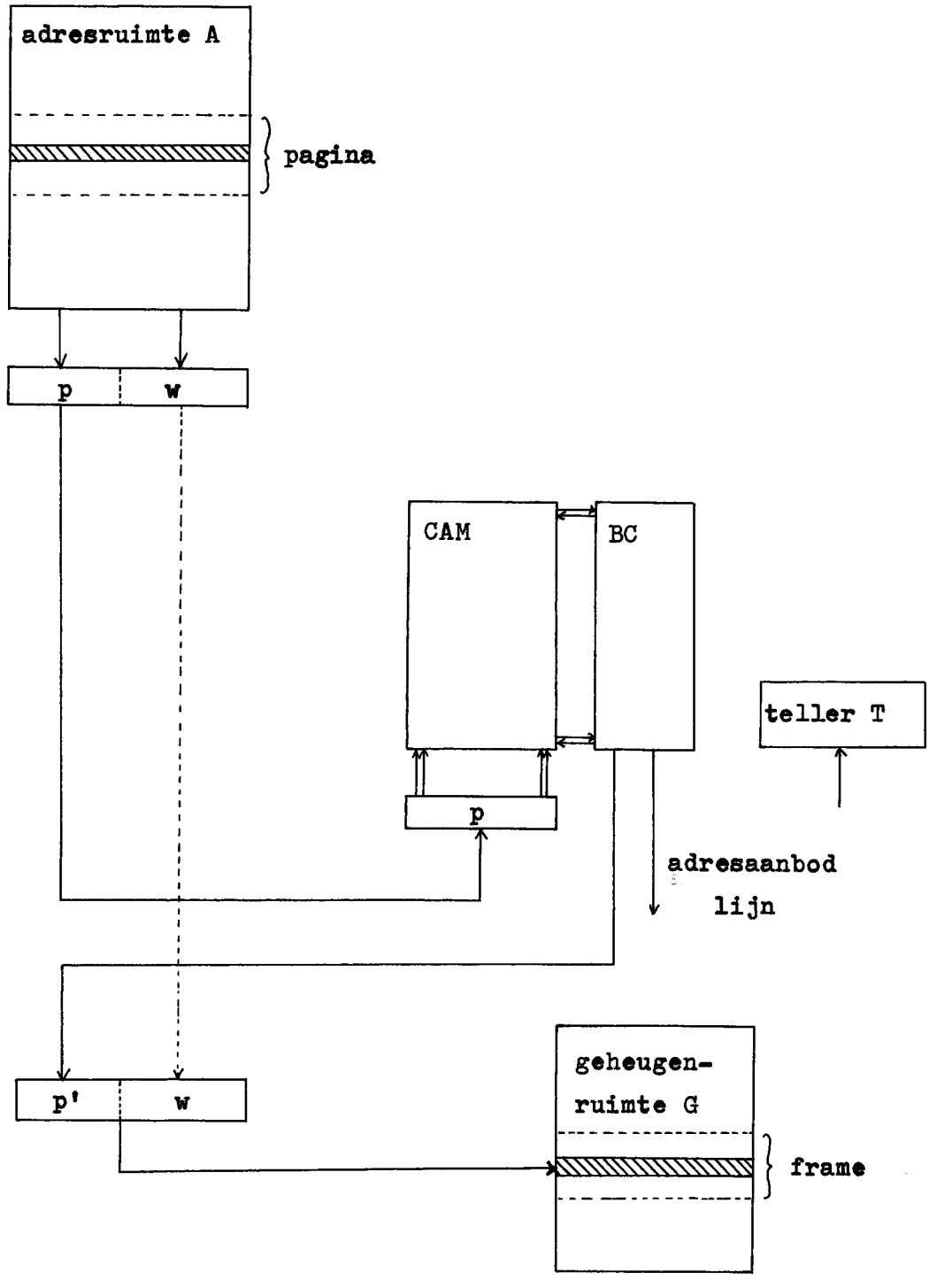


fig 9.3 Paging met behulp van een associatief geheugen.
De teller T heeft op ieder moment een stand, die
één hoger is dan het adres van het laatst geschreven woord in het CAM (paging volgens FIFO-principe)

Zo kan het FIFO-principe worden toegepast, dat wil zeggen, de pagina die het langste in de geheugenruimte G staat wordt uitgewisseld tegen de benodigde pagina.
Het FIFO-principe is hier eenvoudig te realiseren met een teller die bij het CAM aanwezig is.
Deze teller heeft op ieder moment een stand, die 1 hoger is dan het adres van het laatst in het CAM geschreven woord.
Bijvoorbeeld bij een aantal van 256 frames in de geheugenruimte is een teller van 8 bits voldoende.
Als de m frames van G gevuld zijn en dus ook de m woorden van het CAM, dan moet de teller weer in de stand 0 komen.
De stand van de teller geeft aan welke pagina van de geheugenruimte vervangen moet worden door de aangevraagde pagina op de eerstvolgende keer, dat het aanbieden van een pagina-adres p geen associatie oplevert.
Omdat op adres geschreven kan worden, kan als adres voor het CAM de momentane tellerstand worden genomen en op dat adres kan het nieuwe pagina-adres worden ingeschreven.

Een andere vaak gebruikte methode voor pagina-uitwisseling is het Least Recently Used principe (LRU-principe).
Dit principe is moeilijker te realiseren en hierop zal verder dan ook niet worden ingegaan.

9.5 Dynamische geheugentoewijzing

In de vorige paragraaf is al beschreven hoe een pagina-uitwisseling kan plaatsvinden door gebruik te maken van een associatief geheugen als adressseringsmechanisme.
In het algemeen zal een programma uit meerdere pagina's bestaan.
We beschrijven hier een methode van dynamische geheugentoewijzing, waarbij een zo groot mogelijk deel van een programma, en zo mogelijk het gehele programma, dat moet worden uitgevoerd tegelijkertijd in het werkgeheugen aanwezig is.
Bij het toewijzen van een reeel adres aan ieder woord, zal ook vaak een overeenkomstige wijziging van het operand-adres van de instruktie moeten plaatsvinden.

Bij de keuze van een geschikte frame-grootte spelen de volgende overwegingen een rol:

1. De pagina's (frames) mogen niet te klein zijn. Het werkgeheugen wordt dan onderverdeeld in zeer vele frames, waardoor het bijhouden van gegevens omtrent de uitwisseling extra tijd en ruimte gaat kosten, en de pagina-uitwisseling zelf extra tijd gaat kosten.

2. De pagina's mogen niet te groot zijn, omdat dan te veel ruimte wordt verspild. Alle pagina's zullen namelijk niet geheel gevuld zijn; de laatste pagina van een programma zal maar voor een gedeelte gevuld zijn, zodat bij groter gekozen pagina's meer ruimte verspild wordt.
   Dit speelt vooral een grote rol bij veel programma's van een kleine omvang.

Bij deze methode kunnen we bijvoorbeeld pagina's kiezen van 256 woorden.
Gaan we er van uit dat het werkgeheugen een grootte van 64k heeft, dus 256 frames van 256 woorden, dan bestaat een adres uit 16 bits, namelijk 8 bits voor het frame-adres en 8 bits voor het lokaal adres (het woordgedeelte).

Enkele problemen, die bij de toewijzing van ruimte in het werkgeheugen optreden zijn:

- bekend moet zijn welke frames in het werkgeheugen voor toewijzing beschikbaar zijn.
- is het aantal beschikbare frames groot genoeg.
- hoe kunnen de frames beschermd worden tegen ongeoorloofde toegang door een ander programma; ze liggen namelijk door het gehele werkgeheugen verspreid.
- nadat de reele adresssen zijn toegewezen aan een programma, moeten de operand-adressen gewijzigd worden.
- het aantal toegewezen frames moet misschien gedurende de uitvoering van het programma worden gewijzigd.
- niet meer benodigde frames moeten vrijgegeven kunnen worden.
- een aantal frames moet eventueel gebruikt kunnen worden door meerdere programma's.
- opnieuw toewijzen van geheugenruimte aan programma's, die gedumpt moesten worden om ruimte te maken voor het opslaan van een groot programma.

Een associatief geheugen kan hier zinvol worden toegepast. Het kan worden onderverdeeld in een aantal velden, waarin de verdeling van de pagina's over de frames kan worden opgeslagen. We veronderstellen, dat het Operating System de beslissingen neemt omtrent toewijzingen en een volorde van programma's kiest, zoals deze door de processor moeten worden uitgevoerd.
Het associatief geheugen wordt verdeeld in velden, zoals in fig 9.4 aangegeven.
Er zijn drie velden van 8 bits voor de frame-adressen A1, A2, A3. Verder een veld van 8 bits voor het volgnummer en eveneens een veld van 8 bits voor de status-indikatie.
Onder een frame-reeks verstaan we een aantal frames, waarvan de inhoud samen een programma of een programma-segment vormt.

Frame-adres A1 geeft de plaats aan van een frame in het werkgeheugen en frame-adres A2 geeft de plaats aan van het opvolgende frame van het programma.
Met andere woorden, door middel van 256 paren frame-adressen A1 en A2 kunnen de 256 frames van het werkgeheugen gelinked worden op elke gewenste manier.
Door middel van het volgnummer kunnen de 256 frames worden samengesteld tot 256 verschillende frame-reeksen.
Frame-adres A3 wordt gebruikt om de volgorde van het frame in een frame-reeks aan te geven, zoals verderop beschreven zal worden.

0	toegewezen
1	vrijgegeven
2	gereserveerd
3	permanent toegewezen
4	beschikbaar voor toewijzing
5	gedeeld door meerdere programma's
6	volgnummer register

Tabel 9.5 Inhoud van het statusveld en betekenis

A1	A2	A3	volgnr	status
0	*	2	3	3
1	5	1	1	0
2	7	1	4	4
3	0	1	3	3
4	8	2	2	1
5	9	2	1	0
6	4	1	2	1
7	*	2	4	4
8	*	3	2	1
9	*	3	1	0
1	a		1	6
6	b		2	6
3	c		3	6
2	d		4	6

fig 9.6 Voorbeeld van vier frame-reeksen, waarvan de karakteristieke gegevens zijn opgeslagen in het associatief geheugen.

De frames van een bepaalde frame-reeks krijgen eenzelfde volgnummer toegewezen om de frame-reeksen van elkaar te onderscheiden.
In het status-veld wordt de status van het bijbehorende frame aangegeven. Tabel 9.5 geeft de mogelijke inhoud van het statusveld met de bijbehorende betekenis.

Fig 9.6 geeft een voorbeeld van de inhoud van het associatief geheugen, waarbij verondersteld is, dat er 10 frames in het werkgeheugen aanwezig zijn.
We kunnen in fig 9.6 vier frame-reeksen onderscheiden, waarvan de frame-adressen in het associatief geheugen zijn opgenomen. Ze zijn hieronder weergegeven:

volgnummer	frames	status
1	1 - 5 - 9	0
2	6 - 4 - 8	1
3	3 - 0	3
4	2 - 7	4

Een * in het veld A2 geeft het einde van een frame-reeks aan.
Zoals te zien is aan de status, zijn de frames van frame-reeks 4 beschikbaar voor toewijzing.
Deze reeks wordt beschikbare frame-reeks genoemd.
De frames van reeks 1 zijn toegewezen; die van reeks 2 zijn vrijgegeven en die van reeks 3 zijn permanent toegewezen, bijvoorbeeld voor het Operating System.
Verder worden voor deze vier reeksen nog vier woorden in het geheugen gebruikt voor het opslaan van het adres van het eerste frame van elk van de vier frame-reeksen.
Dit gebeurt in het veld van frame-adres A1. De frame-adressen A2 en A3 worden samen gebruikt om de adressen op te slaan waarnaar moet worden teruggekeerd na een interrupt.

De frames van het programma dat uitgevoerd moet worden, worden gelinked. Als het Operating System een programma kiest, vergelijkt het de grootte van het gekozen programma, dat wil zeggen het aantal benodigde frames, met de beschikbare ruimte, dat wil zeggen het aantal beschikbare frames.
Als er genoeg vrije frames zijn, dan geeft het Operating System een volgnummer uit aan de gekozen frame-reeks en geeft opdracht voor geheugentoewijzing.
Als er niet genoeg vrije frames zijn, moet het Operating System een andere frame-reeks kiezen of een dumping van een of meer frame-reeksen (programma's) naar een extern geheugen initialiseren, om meer ruimte te verkrijgen, alvorens de toewijzing kan plaatsvinden.

10 OPSLAAN VAN LABELS IN EEN ASSOCIATIEF GEHEUGEN

10.1 Opslaan van labels

Als we alfanumerieke tekens opslaan in een kode, die 8 bits per karakter gebruikt, dan kunnen in een woord van het associatief geheugen, dat 10 byte groot is, 10 tekens worden opgeslagen.

Labels, zoals die in een programma voorkomen, duiden plaatsen aan, waar naar toe gesprongen kan worden met behulp van een sprong-instruktie.
Het label zal dan in een tabel moeten worden opgezocht om het bijbehorende adres ter beschikking te krijgen.
Bij gebruik van een associatief geheugen voor de opslag van labels, kan de label-naam in het CAM worden opgeslagen en het bijbehorend adres in het RAM.
Een label mag in dit geval uit maximaal 10 toegelaten tekens bestaan.

10.2 Scrambling van lange labels

Bij lange labels kan men een scrambling-proces gebruiken om de label in te korten.
Het is niet zeker, dat dit altijd unieke labels oplevert.
Controle daarop is eenvoudig, door het gescrambelde label aan het CAM aan te bieden en na te gaan of er een associatie optreedt.

Een mogelijke wijze van scrambling is bijvoorbeeld afwisselend het eerste en het laatste teken van het lange label te nemen, tot de zo gekonstrueerde teken-reeks 10 tekens bevat. Het blijkt in de praktijk zo te zijn, dat veelal labels worden gebruikt, die op dezelfde manier beginnen en alleen in de 'staart' treedt een verschil op.

Lange label	gescrambelde label
s i g m a a	s a i a
s i g m a b	s b i a
s i g m a c	s c i a
l a m b d a 1	l 1 a a
l a m b d a 1 3	l 3 a 1
t r i d i a g	t g r a

fig 10.1 Scrambling van lange labels
en opslag in het CAM

fig 10.1 geeft een voorbeeld van scrambling van enkele labels, waarbij er van uit is gegaan, dat het CAM 4 alfanumerieke tekens kan bevatten.

Als het scrambling-proces leidt tot een reeds aanwezige label, dan kan een ander proces worden gebruikt of een volgnummer worden uitgegeven.
Het voordeel van een associatief geheugen is hier, dat telkens als een scrambling-proces heeft plaatsgevonden, snel kan worden nagegaan of het gevormde label al aanwezig is.

10.3 Opslaan van lange labels in het CAM

Men kan ook de oorspronkelijke onverkorte labels in het CAM opslaan, waarbij gelijke delen van labels (deellabels) slechts eenmaal worden opgeslagen.
Dit kan en zal een aanzienlijke besparing van geheugenruimte opleveren, aangezien meestal vele gelijksoortige labels in een programma worden gebruikt.
De intentie is om gelijke deellabels slechts eenmaal op te slaan. Dit zal niet altijd mogelijk zijn, maar met een geschikte opzet van de methode van inschrijven, wissen en uitlezen, kan een groot aantal labels in een kleine ruimte worden opgeslagen, door deellabels gemeenschappelijk te gebruiken.

Uitgaande van de labels van fig 10.1 kunnen we deze labels opslaan, zoals in fig 10.2 is aangegeven.
We nemen weer aan dat een geheugenwoord vier alfanumerieke tekens kan bevatten.

Inhoud CAM	inhoud RAM
s i g m	*
a a	adres(sigmaa)
a b	adres(sigmab)
a c	adres(sigmac)
l a m b	*
d a 1	adres(lambda1)
d a 1 3	adres(lambda13)
t r i d	*
i a g	adres(tridiag)

fig 10.2 Opslaan van lange labels in het CAM

Het is echter duidelijk dat dit niet zonder meer kan.
De deellabels kunnen namelijk op willekeurige onbekende plaatsen in het CAM zijn opgelsagen.

We zullen bij de deellabels indikaties moeten opslaan, waardoor een verbinding van de deellabels onderling tot stand komt.
Een unieke grootheid voor ieder woord van het CAM is het adres dat de besturing van het CAM bij een associatie afgeeft.
We kunnen dit dus als indikatie gebruiken.
Fig 10.3 geeft aan hoe dat kan gebeuren.

Inhoud CAM			inhoud RAM
*	deellabel 1	adres(i)	*
adres(i)	deellabel 2	adres(i+1)	*
adres(i+1)	deellabel 3	adres(i+2)	adres(label)

fig 10.3 Linken van deellabels in het CAM

We moeten ook kunnen aangeven of een deellabel het begin van een label is en dus als zodanig ook gebruikt kan worden voor andere labels.
In het vervolg zullen we bij de voorbeelden aannemen, dat een enkel alfanumeriek teken in een woord van het CAM kan worden opgeslagen.
We nemen als voorbeeld de labels: m b r, q b r, q b w, q c r.
In fig 10.4 is aangegeven hoe deze labels in het geheugen worden opgeslagen.

Inhoud CAM				inhoud RAM
begin label	link adres	label	CAM adres	
1	*	m	0	*
0	0	b	1	*
0	1	r	2	adres(mbr)
1	*	q	3	*
0	3	b	4	*
0	4	r	5	adres(qbr)
0	4	w	6	adres(qbw)
0	3	c	7	*
0	7	r	8	adres(qcr)

fig 10.4 Wijze van opslag van lange labels in het CAM en de adressen in het RAM

Er zijn bij gebruik van deze methode dus 9 geheugenwoorden nodig in plaats van 12.

Alle begindeellabels kunnen worden samengenomen. Bij ieder laatste deellabel moet een adres in het RAM worden opgeslagen. Dat wil zeggen, dat voor ieder laatste deel van een label een woord in het geheugen nodig is.

Er onstaan echter nog problemen als een label een begindeel is van een langer label.
Stel we hebben de lange label q b r opgeslagen, zie fig 10.5.

Inhoud CAM				inhoud RAM
begin label	link adres	label	CAM adres	
1	*	q	0	*
0	0	b	1	*
0	1	r	2	adres(qbr)

fig 10.5 Label qbr opgeslagen in het geheugen

Als q b r al is ingeschreven en we willen nagaan of label q al is ingeschreven, dan blijkt q inderdaad aanwezig te zijn. Het RAM bevat op adres 0 echter een onbepaalde inhoud en niet het adres behorende bij de label q.
We reserveren nu nog een extra bit in het CAM, dat de volgende betekenis heeft.

Als het bit 0 is, dan is het adres in het RAM niet geldig.
Als het bit 1 is, dan is het adres in het RAM geldig.

We willen nu de labels q b r, q, b opslaan, zie fig 10.6.

Inhoud CAM				inhoud RAM	
begin label	adres geldig	link adres	label	adres CAM	adres van het label
1	1	*	q	0	adres(q)
0	0	0	b	1	*
1	0	1	r	2	adres(qbr)
1	1	*	b	3	adres(b)

fig 10.6 Het opslaan van de labels qbr, q, b in het associatief geheugen.

Label b moet nu apart worden opgeslagen, het is namelijk het einde van een label. Zouden we het deellabel b van de label qbr gebruiken, dan moet het tweede bit van woord 1 in een 1 worden gewijzigd. Dit zou er echter op kunnen duiden, dat er een label qb bestaat, waarbij het adres hoort, dat op adres 2 van het RAM staat.

De voordelen van het gebruik van een associatief geheugen het opslaan van labels ligt daarin, dat gelijke delen van labels in vele gevallen maar een keer behoeven te worden opgeslagen. Of een deellabel al in het geheugen aanwezig is kan snel worden gekonstateerd, door het deellabel aan het CAM aan te bieden.

11 DYNAMISCHE STACK-ADRESSERING

11.1 Stacks

Een stack funktioneert als een last in - first out (LIFO) geheugengebied, dat aan een programma wordt toegewezen voor de opslag van programma-variabelen, data-referenties en tijdelijke data van het in bewerking zijnde programma.
Een stack wordt gekarakteriseerd door zijn grenzen, die het gebied afbakenen dat in het geheugen voor de stack beschikbaar is.
Een associatief geheugen kan op verschillende manieren worden gebruikt voor de opslag van stack-gegevens en de stack-inhoud zelf.
Een stack die in een geheugen kan worden opgebouwd heeft in het geval van een enkelvoudige stack vier kenmerken:

1. Het stacknummer (of de stacknaam) aangeduid met SN
2. De upper limit van de stack, aangeduid met UL
3. De lower limit van de stack, aangeduid met LL
4. De top-stack (of stack-pointer), aangeduid met TS

Twee manieren voor het opslaan van stacks met behulp van een associatief geheugen zullen worden besproken, namelijk:

1. Alleen de karakteristieke stack-adressen worden in het associatief geheugen opgeslagen
2. De stack-adressen en een gedeelte van de stack-inhoud worden in het associatief geheugen opgeslagen

De karakteristieke stack-adressen kunnen in drie woorden van het geheugen worden opgeslagen.
Het associatief geheugen wordt daartoe in drie velden verdeeld en ook het bijbehorend RAM wordt gebruikt.
In veld 1 van elk van de drie voor de stack benodigde woorden, wordt het stack-nummer opgeslagen.
In veld 2 worden respektievelijk de upper limit (UL), de lower limit (LL) en de top stack (TS) opgeslagen, zoals in fig 11.1 is aangegeven.
In veld 3 worden de bijbehorende adressen opgeslagen.
Deze adressen komen tevens in het RAM te staan; ze verwijzen naar het RAM (werkgeheugen) waar de stack zelf is opgeslagen.

11.2 Opslaan van stack-gegevens in het associatief geheugen

Het aantal elementen van een stack kan voortdurend aan wijzigingen onderhevig zijn.
De stack-inhoud neemt daardoor voortdurend toe of af.

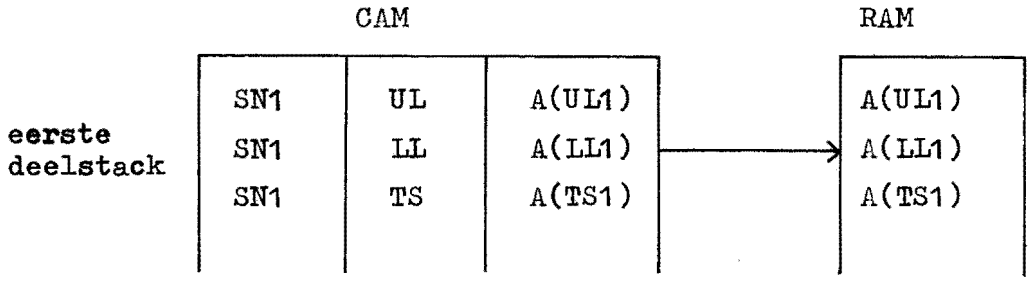


Fig 11.1 Wijze van opslaan van de karakteristieke stack-gegevens in CAM en RAM.

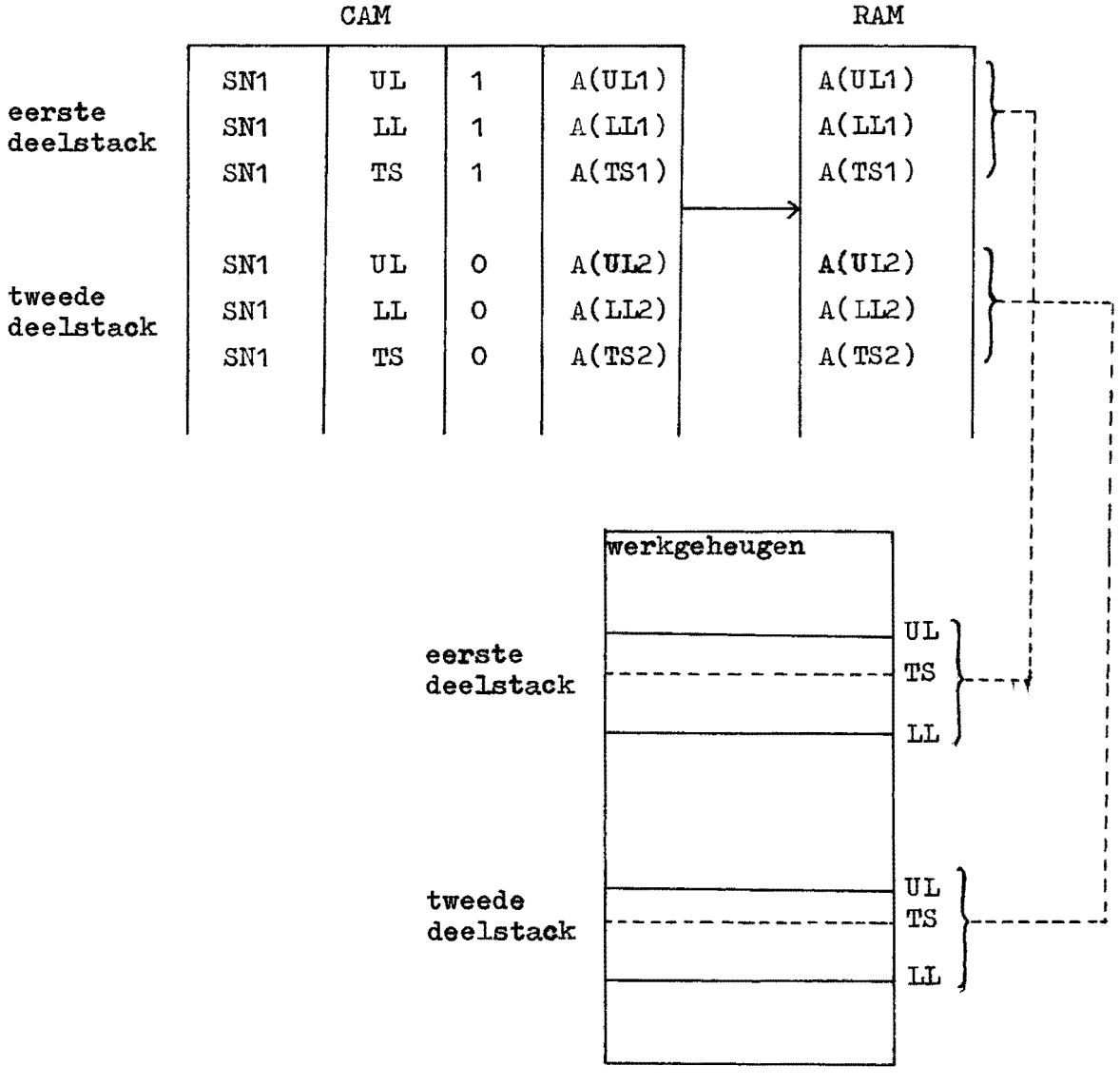


fig 11.2 Uitbreiding van de eerste deelstack (de oorspronkelijke stack) met een tweede deelstack.

1. Het verlagen van de stack kan als volgt gebeuren: eerste wordt SN,TS,x,x aangeboden, daarna wordt SN,LL,x,x aangeboden. Als de adressen, behorend bij TS en LL nog niet aan elkaar gelijk zijn dan is de lower limit van de stack nog niet bereikt. Als de genoemde adressen wel aan elkaar gelijk zijn, dan is de lower limit bereikt en is de stack uitgeput.

2. Het verhogen van de stack kan als volgt gebeuren: eerst wordt SN,TS,x,x aangeboden, en daarna SN,UL,x,x. Zolang de upper limit van de stack nog niet is bereikt kunnen nog gegevens worden toegevoegd. Als de upper limit is bereikt, dan is de stack vol en zullen we een ander gebied moeten zoeken om de stack uit te breiden.

In het algemeen zal er in het geheugen waarin de stack SN1 is opgeslagen, boven de upper limit (UL) geen plaats zijn om de stack naar boven uit te breiden. We zullen dus een ander gebied moeten zoeken. Dit kan een taak zijn van het Operating System. Bij de upper limit in het CAM en in het RAM moet dan een link-adres worden opgeslagen, dat het oude deel van de stack met het nieuwe verbindt.
We zullen de oorspronkelijke stack dan de eerste deelstack noemen en de uitbreiding noemen we de tweede deelstack.
In het associatief geheugen slaan we de tweede deelstack op onder hetzelfde stacknummer als de eerste deelstack.
Om de tweede deelstack en de eerste van elkaar te kunnen onderscheiden, maken we gebruik van een indikatie-bit (I-bit) in het associatief geheugen. Dit bit wordt 1 voor de drie karakteristieke gegevens van de eerste deelstack en 0 voor de drie karakteristieke gegevens van de tweede deelstack.

Fig 11.2 geeft weer hoe de opslag van de karakteristieke stack-gegevens kan plaatsvinden.

Om nu de gegevens van de stack uit te lezen na uitbreiding met een tweede deelstack wordt SN1,TS,0,x aangeboden.
We verkrijgen het adres van de momentane top stack.
Als alleen de eerste deelstack aanwezig is, dan dient dus het indikatie-bit daarbij ook nul te zijn.
Op deze wijze kunnen we steeds nieuwe deelstacks aan de oorspronkelijke stack toevoegen en de voor deze nieuwe deelstacks gereserveerde ruimte ook weer laten vervallen, als de stack-inhoud zo klein wordt, dat de oorspronkelijke stack voldoende is. Deze methode met het associatief geheugen beoogt dus een optimaal gebruik van de geheugenruimte voor het opslaan van stacks.

Voor een dynamische wijziging van de stack-grootte dient het CAM te worden onderverdeeld in 5 velden, namelijk voor:

1. Het stacknummer
2. De stack-grootheden UL, LL, TS
3. Het laagnummer
4. Het linknummer
5. Het adres van het extern geheugen

stack nummer	stack gegevens	laag nummer	link nummer	stack adressen		stack adressen
SN1	UL	1	*	A(UL1)		A(UL1)
SN1	LL	1	*	A(LL1)		A(LL1)
SN1	TS	1	*	A(TS1)	→	A(TS1)
SN1	UL	0	1	A(UL2)		A(UL2)
SN1	LL	0	1	A(LL2)		A(LL2)
SN1	TS	0	1	A(TS2)		A(TS2)

fig 11.3a Oorspronkeleijke stack (eerste deelstack) uitgebreid met tweede deelstack.

stack nummer	stack gegevens	laag nummer	link nummer	stack adressen		stack adressen
SN1	UL	1	*	A(UL1)		A(UL1)
SN1	LL	1	*	A(LL1)		A(LL1)
SN1	TS	1	*	A(TS1)		A(TS1)
SN1	UL	2	1	A(UL2)	→	A(UL2)
SN1	LL	2	1	A(LL2)		A(LL2)
SN1	TS	2	1	A(TS2)		A(TS2)
SN1	UL	0	2	A(UL3)		A(UL3)
SN1	LL	0	2	A(LL3)		A(LL3)
SN1	TS	0	2	A(TS3)		A(TS3)

fig 11.3b Uitbreiding van de twee deelstacks van fig 11.3a met een derde deelstack.

Het adres van het extern geheugen staat tevens in het RAM.
Als de eerste deelstack is uitgebreid met een tweede deelstack, dan is de geheugen-inhoud zoals in fig 11.3a is aangegeven. De aktuele deelstack is deelstack 2.
De toegang tot de aktuele deelstack vindt altijd plaats door als laagnummer een 0 in veld 3 aan te bieden.

Als stack-uitbreiding moet plaatsvinden, dan dient ruimte voor een derde deelstack te worden gezocht.
Is deze ruimte gevonden dan wordt de inhoud van het associatief geheugen zoals in fig 11.3b aangegeven.
De gegevens van de derde deelstack worden eveneens in drie woorden van het CAM opgeslagen, met als laagnummer een 0, omdat deelstack 3 de aktuele stack is.
In het veld voor het linknummer komt een 2 als verwijzing naar de tweede deelstack.
De adressen van de gevonden ruimte worden in het CAM en in het RAM opgeslagen.
De gegevens van de tweede deelstack,zoals die waren opgeslagen blijven ongewijzigd, behalve dat het laagnummer 2 wordt in plaats van 0.

11.3 Opslaan van stack-gegevens en -data in het associatief geheugen

Een andere methode om een uitbreiding van ruimte ten behoeve van een stack te realiseren is het reserveren van ruimte in het associatief geheugen om ook stack-data op te slaan. Dit naast de opslag van de karakteristieke stack-gegevens, zoals in de vorige paragraaf aangegeven.
Als de top stack van de oorspronkelijke stack de upper limit stack (UL) heeft bereikt, dan wordt voorlopig geen nieuwe deelstack gekonstrueerd in het werkgeheugen. Er wordt voorlopig een deelstack opgebouwd in het associatief geheugen.
Als het aantal gebruikte data-woorden in het associatief geheugen een zekere limiet overschrijdt, dan worden alle data-woorden gedumped op een geschikte plaats in het extern geheugen en wordt een verwijzing in het CAM opgenomen, zoals in par 11.2 is beschreven.
In fig 11.4 is aangegeven hoe eveneens data-opslag in het associatief geheugen gerealiseerd kan worden.
We hebben in het RAM een veld gereserveerd voor adres/data-indikatie.
De betekenis van de inhoud van dit veld is de volgende:

0: hetgeen volgt is een adres en er zijn geen data van deze stack in het associatief geheugen opgeslagen
1: hetgeen volgt is een adres (het TS-adres = het UL-adres). Er zijn data in het associatief geheugen opgeslagen
2. Wat volg zijn data

stack nummer	stack gegevens	laag nummer	link nummer	stack adressen
SN1	UL	0	*	A(UL1)
SN1	LL	0	*	A(LL1)
SN1	TS	0	*	A(TS1)
SN1	TS	1	*	
SN1	TS	2	1	
SN1	TS	3	2	
SN1	TS	4	3	
SN1	TS	0	4	

D/A indikatie	stack adressen
0	A(UL1)
0	A(LL1)
1	A(TS1)
2	data 1
2	data 2
2	data 3
2	data 4
2	data 5

fig 11.4 Opslaan van stack-gegevens en -data in het associatief geheugen.

Als we de inhoud van de plaats van de top stack (TS) willen hebben, dan bieden we SN1,TS,0,x,x aan.
Er treden, als ook data in het CAM zijn ingeschreven twee associaties op.

1. Bij de gegevens van de oorspronkelijke stack associeert SN1,TS,0,x,A(TS0).
2. Bij de data van de voorlopige deelstack associeert SN1,TS,0,5,?.

De komma's geven de scheiding van de velden aan.
Welke van deze twee de eerste is die associeert hangt af van de onderlinge positie in het CAM.
Als 1 de eerste associatie oplevert en in het adres/data-veld staat een 1 dan moeten we het volgend woord hebben dat associeert en we kunnen dan meteen de data uitlezen.
Als 2 de eerste associatie oplevert, dan vinden we een 2 in het adres/data-veld en kunnen we meteen de data uitlezen.

12 EEN ASSOCIATIEF WERKGEHEUGEN

Een onderzoek naar de mogelijkheden om een associatief geheugen te gebruiken als werkgeheugen voor een rekenmachine is gedaan aan de Universiteit van Braunschweig (zie ref 148).

Uitgangspunt is een processor, die instruktie voor instruktie van het in het werkgeheugen opgeslagen programma ophaalt, ze dekodeert, de operanden uit hetzelfde geheugen haalt en de operaties met behulp van de CPU, de data-kanalen etc uitvoert.
Het werkgeheugen heeft associatieve eigenschappen, echter alleen het oproepmechanisme van de instrukties en de data is associatief, terwijl de inhoud van de woorden niet associatief toegankelijk is, maar alleen geschreven of gelezen kan worden.

Het onderzochte rekenmachinekoncept onderscheidt zich van de uit de literatuur bekende 'associatieve komputers' of 'associatieve processors', waarin het essentiele punt is, dat de associatief toegankelijke datavelden parallel bewerkt worden, terwijl de opeenvolging van instrukties door een konventioneel komputersysteem met een adresseerbaar geheugen gegenereerd wordt.

Voor het oproepen van instrukties geldt, dan een ononderbroken keten van instrukties, een lineaire tak genoemd, onder een enkele naam in de juiste volgorde opgeslagen kan worden. Bij een oproep (en associatie) worden al deze woorden tegelijkertijd gemarkeerd en kunnen ze achtereenvolgens worden uitgelezen.
Een instruktie-teller is hierbij niet nodig.
Er is alleen een instruktie-zoek-register met naamgedeelte en maskergedeelte. Dit register kan na afloop van de lineaire tak op de volgende naam (volgnaam genoemd) worden ingesteld.
Deze volgnaam kan door een serie voorwaarden uit de aktuele toestand van de data tijdens het doorlopen van de lineaire tak worden samengesteld.
Het bleek nuttig te zijn om zogenaamde 'enkelvoudige wijzigings-instrukties' voor speciaal uitgekozen bits van de volgnaam in te voeren.
Daarnaast komen 'totale wijzigings-instrukties' overeen met de gebruikelijke voorwaardelijke sprong.
Bij data-aanvraag kan men ook de mogelijkheden benutten, die een associatief geheugen biedt.
Zo is bijvoorbeeld het oproepen van een geheel data-blok in een oproep mogelijk.
Voor het gehele systeem geldt, dat er speciale instrukties, hardware-eenheden en programmeertechnieken zijn, waarmee willekeurige programmastrukturen aan de struktuur van het associatief geheugen kunnen worden aangepast.
Een adresseerbaar geheugen staat alleen een gunstige aanpassing van lineaire programma-segmenten toe aan het geheugen door de tel-operatie van de instruktie-teller.

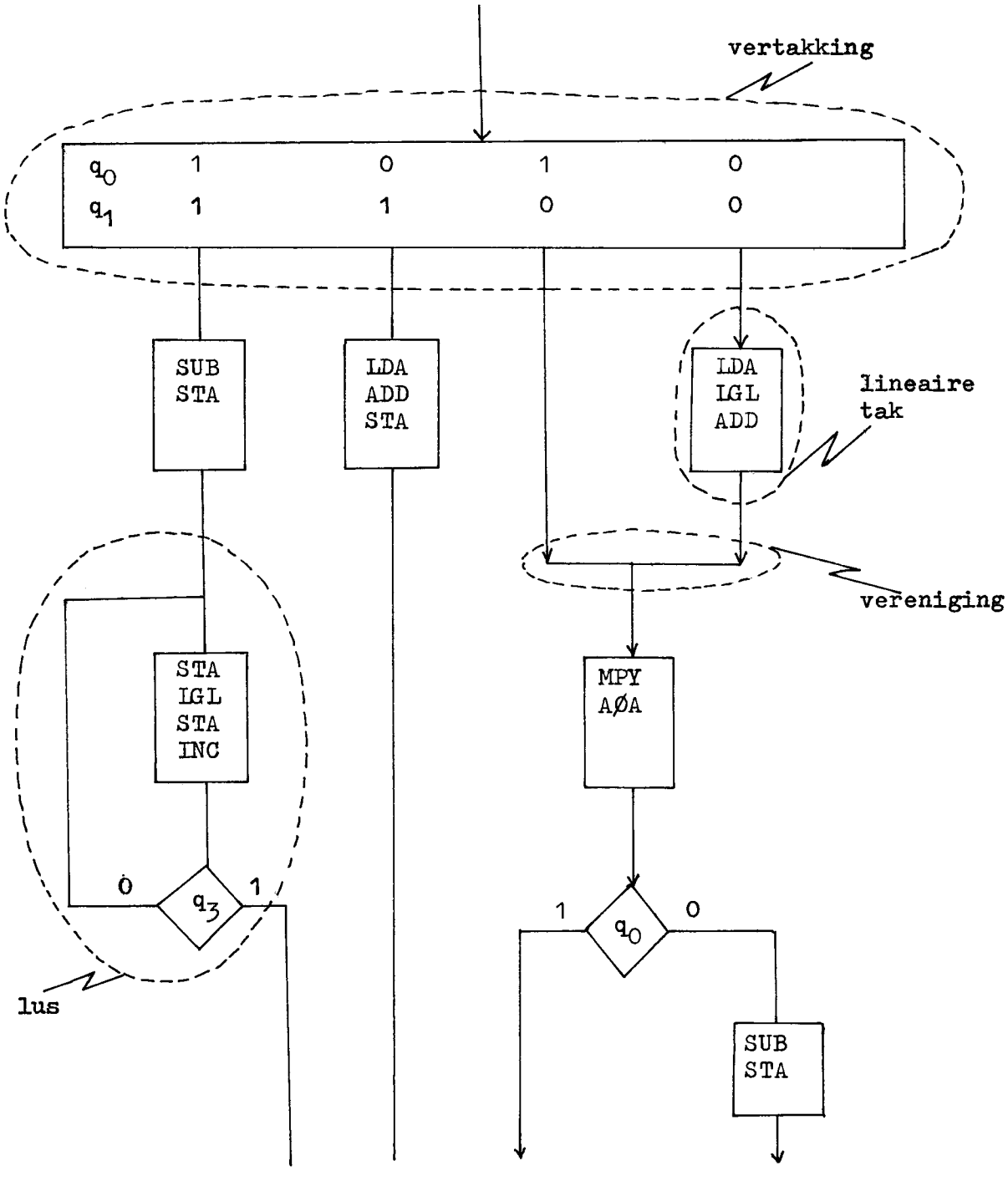


fig 12.1 De drie programma-struktuur-elementen:

a. lineaire tak (aktie)
b. vertakking
c. vereniging

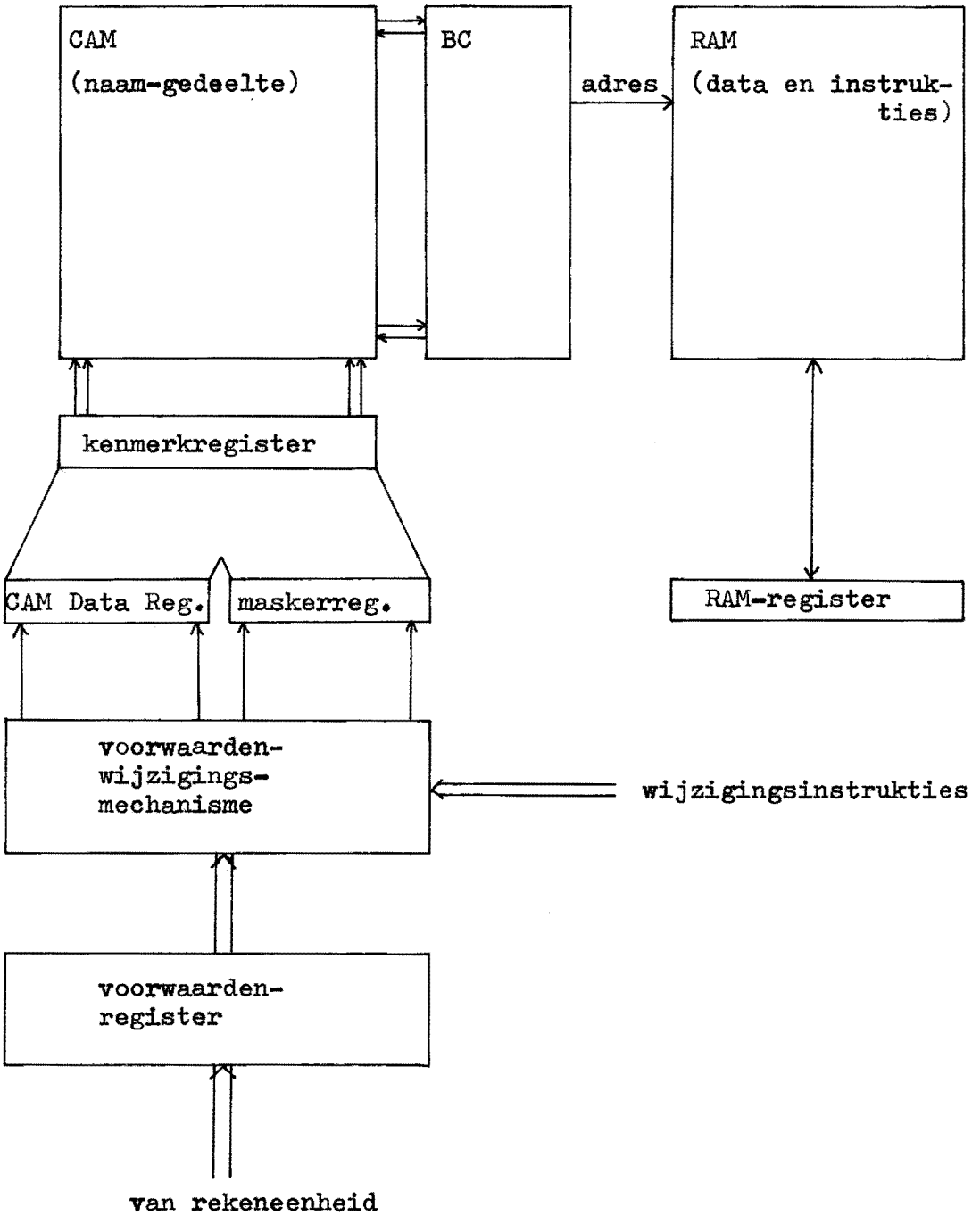


fig 12.2 Een associatief werkgeheugen met mogelijkheden voor opslag van instrukties en data.

Het associatief geheugen biedt met zijn vrij te kiezen namen en de linearisering door de besturings-logica (sequentieel aftasten van de woorden die associeren) meer flexibiliteit voor een aanpassing van de kant van het geheugen.

Om de programmeerregels systematisch te kunnen ontwikkelen, moeten allereerst de strukturele eigenschappen van programma's en het associatieve geheugen aan elkaar worden aangepast.
Iedere willekeurige programma-struktuur kan men met de volgende drie programma-struktuur-elementen opbouwen; zie fig 12.1.

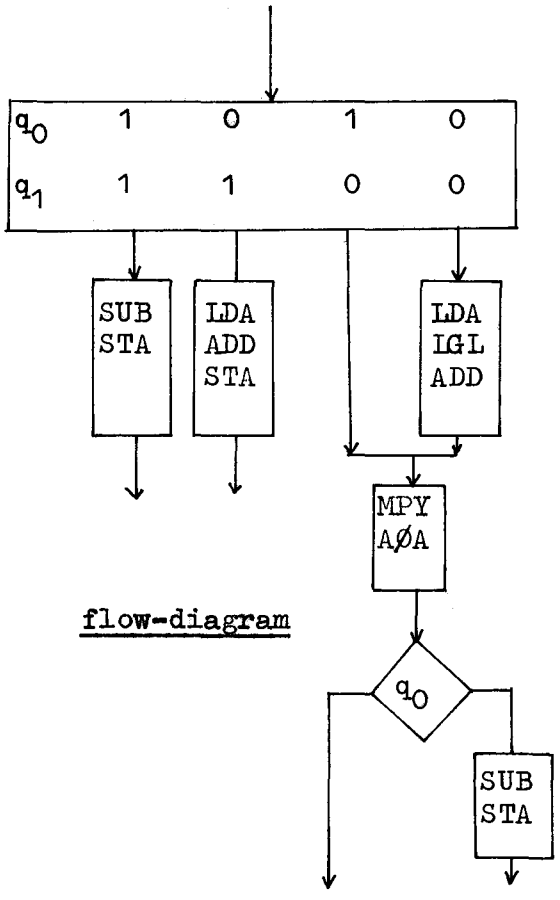
1. Lineaire tak (aktie); dit is een lineaire volgorde van instrukties zonder inwendige vertakkingen of verenigingen.
2. Vertakking; een vertakking vindt plaats in het programma afhankelijk van bepaalde voorwaarden.
3. Vereniging; dit is een samenloop van meerdere lineaire takken in een lineaire tak.

Fig 12.2 geeft het CAM en het RAM met de bijbehorende registers en logica voor het gebruik van het associatief geheugen voor de opslag van programma's.
Bovendien bevat de komputer een zogenaamd voorwaarden-wijzigingsmechanisme, dat de waarden van het voorwaarden-register (VR), dat de vertakkingen bestuurt, toewijst aan bepaalde plaatsen in het naam- en maskerregister.
De gegevens voor het voorwaarden-register komen uit de rekeneenheid. Het voorwaarden-wijzigings-mechanisme wordt gestuurd door wijzigings-instrukties.

1. De lineaire tak (fig 12.3)
- de instrukties worden in hun natuurlijke volgorde, waarin ze uitgevoerd moeten worden, in het associatief geheugen opgeslagen
- iedere instruktie krijgt dezelfde naam in het associatief geheugen
- om aktie op te roepen moet het instruktie-zoek-woord (hier 00) naar het naamregister worden gebracht. Alle namen, die associeren kunnen dan achtereenvolgens worden afgetast

2. De vertakking (zie fig 12.3)
- afhankelijk van de voor de vertakking relevante inhoud van het voorwaarden-register, vertakt het programma zich in een van meerdere opvolgende takken.
- de opslagvolgorde van de verschillende akties is willekeurig
- iedere aktie krijgt een naam toegewezen, waarvan de afzonderlijke bits worden bepaald door de betreffende bits van het voorwaarden-register, die de vertakkingen besturen.

3. De vereniging (zie fig 12.3)
- bij een vereniging monden meerdere voorafgaande takken uit in een volgtak
- de verenigings-aktie moet onder de voorafgaande takken worden opgeslagen



flow-diagram

VW-reg.	q_0	1	0	1	0
	q_1	1	1	0	0
A^1	SUB	*			
	STA	*			
A^2	LDA				*
	IGL				*
	ADD				*
A^3	LDA		*		
	ADD		*		
	STA		*		
A^4	MPY			*	
	AØA			*	
A^5	SUB				*
	STA				*

beslissingstabel

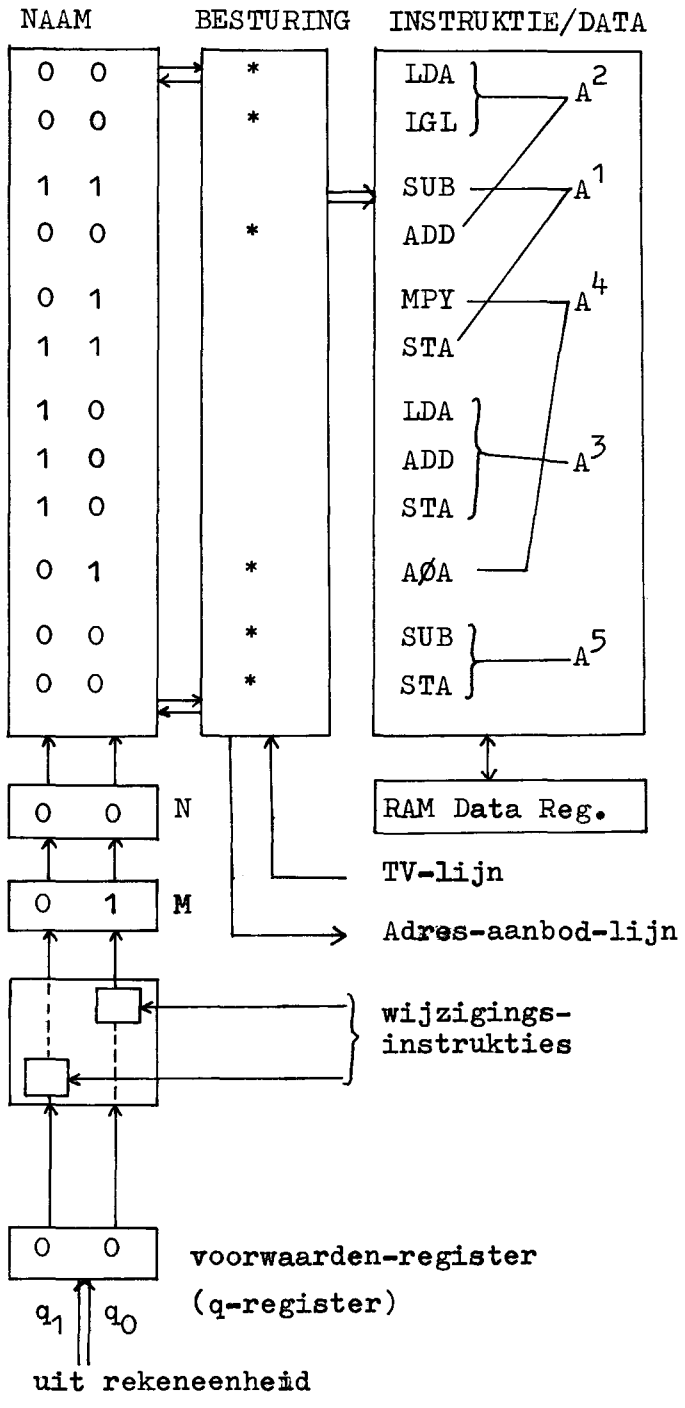


fig 12.3 Flow-diagram, beslissingstabel en methode van opslag in het associatief geheugen van de drie programma-struktuur-elementen.

- de naam van de volgtak wordt zo gekozen, dat hij tegelijkertijd met het aanroepen van de voorafgaande tak associeert.
- het zoekwoord kan op geschikte plaatsen gemaskeerd worden, afhankelijk van bepaalde voorwaarden. Men kan daardoor bewerkstelligen, dat zowel een aan de vereniging voorafgaande tak als de volgtak associeert

Op grond van de logische mogelijkheden van het associatief geheugen mag men verwachten, dat 'associatieve programma's' minder besturing en besturingssignalen vereisen als konventionele programma's.
Bij een lineaire volgorde van de instrukties zal het uitlezen en dekoderen voor een adresseerbaar of een associatief geheugen, wat de benodigde tijd betreft, niet veel verschillen.
Als echter gekompliceerde verenigingen en vertakkingen optreden, is bij een associatief programma slechts eenzelfde hoeveelheid besturingssignalen nodig als bij een lineaire tak.
De besparingen aan geheugenruimte en executietijd hangen sterk af van de struktuur van de programma's.

13 PATROONHERKENNEN MET EEN ASSOCIATIEF GEHEUGEN.

Patroonherkennen komt neer op het verdelen van een aantal vormen in een aantal klassen.
Enerzijds kunnen de vormen die kunnen voorkomen bekend zijn, zoals bij het herkennen van standaardtekens zoals ISO-A of ISO-B.
Anderzijds kan een veelheid aan vormen optreden, zoals bijvoorbeeld bij handgeschreven cijfers.
Er zal worden nagegaan hoe een associatief geheugen kan worden gebruikt voor het herkennen van handgeschreven cijfers.
Bij handgeschreven cijfers heeft men te maken met een groot aantal vormen. Het herkenningsproces komt neer op het verdelen van al deze vormen over de tien klassen 0 t/m 9.
In vrijwel alle bestaande systemen wordt een beeld gevormd van het cijfer met behulp van een TV-kamera, een flying-spot scanner of een fotodiode-array.
Dit beeld wordt gedigitaliseerd en eventueel genormeerd, zodat een binaire matrix van zwarte (bezette) en witte (onbezette) punten wordt verkregen; in de komputer voorgesteld door een reeks van nullen en enen.
Uit deze matrix worden een aantal kenmerken bepaald. Het al of niet aanwezig zijn van een kenmerk kan door een bit (0 of 1) worden aangegeven. Eventueel kan de mate waarin het kenmerk aanwezig is door meerdere bits worden aangegeven.
Op deze wijze krijgen we een string van bits, die een bepaalde vorm karakteriseren.
Voor redelijke handgeschreven cijfers zullen we volgens onderzoekingen ongeveer 100 bits nodig hebben.

Nemen we voorlopig aan, dat we met net geschreven cijfers te maken hebben, dan zal een string van 80 bits voldoende zijn; dat komt neer op een geheugenwoord van het associatief geheugen per cijfervorm.
Daar ieder cijfer zich in een aantal 'vormen' kan manifesteren, zullen er voor ieder cijfer een aantal geheugenwoorden nodig zijn.

Het herkennen is een proces van data-reduktie. Uitgaande van 1024 bits (cijfervorm en achtergrond) moeten we deze reduceren tot minder dan 4 bits.

Als we uitgaan van een cijfervorm in een matrix bestaande uit 32 x 32 punten, dan kan daarop een proces worden toegepast dat afdunnen of skeleteren heet. De vorm die dan ontstaat, wordt het skelet genoemd. Het skelet wordt gemaakt door achtereenvolgens randpunten van de cijfervorm weg te nemen, totdat we een vorm krijgen, die overal nog slechts een matrix-positie breed is. Fig 13.1 geeft een voorbeeld van een cijfervorm, die een afdunproces heeft ondergaan.
Voor een gedetailleerde beschrijving van het opnemen en het afdunnen wordt verwezen naar de referenties 155 en 156.

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 . . . . . 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 . . . 2 2 2 2 2 2 2 . . . . 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 . . . 2 . . 0 0 . . . 2 2 2 . . 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 . . 2 . 0 0 0 0 0 0 0 . . . 2 . . 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 . . 1 . 0 0 0 0 0 0 0 0 0 0 . . 2 . . 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 . . . 0 0 0 0 0 0 0 0 0 0 0 . . 2 . . 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 . . 2 . . 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 . . 2 . . 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 . . 2 . . 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 . . 2 . 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 . . . 2 . . 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 . . . . 2 . 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 . . . . 2 . 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 . . . 2 . 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 . . 2 . . 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 . . . 2 . . 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 . . . 2 . . 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 . . 2 . . 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 . . 2 . . 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 . . 2 . 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 . . 0 0 0 0 0 0 0 0 0 0 0 0 0 0 . . 2 . 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 . . 1 . 0 0 0 0 0 0 0 0 0 0 0 0 0 0 . . 2 . . 0 0 0 0 0 0
0 0 0 0 0 0 0 0 . . . 2 . . 0 0 0 0 0 0 0 0 0 0 0 0 . . 2 . 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 . . . 2 2 . . . . 0 0 0 0 0 0 . . 2 . . 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 . . 2 2 2 . . . 0 0 0 . . 2 . . 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 . . 2 . . . . . 2 . . . 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 . 2 2 2 2 2 2 . . 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 . . . . . 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Fig 13.1 Matrix bestaande uit 32 x 32 punten.
Het oorspronkelijke cijfer wordt gevormd door de punten (.) en de cijfers groter dan nul. Deze laatsten (hier alleen de cijfers 1 en 2 omdat er geen driesprongen zijn) vormen het afgedunde cijfer, ook wel skelet genoemd.

Het skelet wordt gevormd door de punten, die met een 3 zijn aangeduid (de driesprongen, hier niet aanwezig), de punten, die met een 2 zijn aangeduid (de verbindingspunten) en de punten, die met een 1 zijn aangeduid (de eindpunten).
Als het afdunproces ten einde is worden de zogenaamde bijzondere punten bepaald; dat zijn eindpunten en driesprongen, met hun koordinaten.
Tot de bijzondere punten behoren: eindpunten, driesprongen en viersprongen.
Een punt met een bezette buur is een eindpunt.
Een punt met drie bezette buren is een driesprong.
Een punt met vier bezette buren is een viersprong.

Aan de hand van het aantal eindpunten (e), het aantal driesprongen (d) en het aantal viersprongen (v) worden groepen gevormd.
Omdat viersprongen in de praktijk erg weinig voorkomen, en om niet onnodig veel groepen te krijgen, wordt een viersprong beschouwd als zijnde twee driesprongen met dezelfde coordinaten.
We kunnen een groep dan aanduiden met een tweecijferig getal, waarvan het eerste cijfer het aantal eindpunten (e) aangeeft en het tweede cijfer het aantal driesprongen (d).

Fig 13.2 geeft een aantal groepen met 'net geschreven' cijfers uit elke groep.

Het associatief geheugen kan gebruikt worden:
- in de experimenteer- en testfase
- bij de definitieve versie van het apparaat.

Bij de testfase wordt het geheugen telkens gebruikt voor de cijfervormen binnen een bepaalde groep.
In de definitieve versie worden de bitstrings, die een cijfervorm voorstellen in het geheugen opgeslagen en het herkennen kan snel gebeuren door een aangeboden bitstring te vergelijken met de gehele geheugen-inhoud.
De strings in het associatief geheugen zijn als volgt opgebouwd.

We reserven in het geheugen bepaalde bits of groepen van bits voor een bepaald kenmerk.
Hieronder is aangegeven hoe dat kan gebeuren.

Bits	afk	kenmerk
1,2,3	e:	aantal eindpunten
4,5,6	d:	aantal driesprongen
7	slb:	is er een sliert naar links boven
8	srb:	is er een sliert naar rechts boven
9	slo:	is er een sliert naar links onder
10	sro:	is er een sliert naar rechts onder
11,12	ei1:	ligging van het eerste eindpunt
13,14	ei2:	ligging van het tweede eindpunt
15,16,17	mdh:	max aantal doorsnijdingen horizontaal
18,19,20	mdv:	max aantal doorsnijdingen vertikaal

groep	cijfervormen
0 0	0
0 2	8 8
1 1	9 6 9
2 0	1 2 3 5 6 7
3 1	2 3 4 5
4 0	4 5
4 2	4 7

Fig 13.2 De meest voorkomende cijfervormen en de groepen waartoe ze behoren.

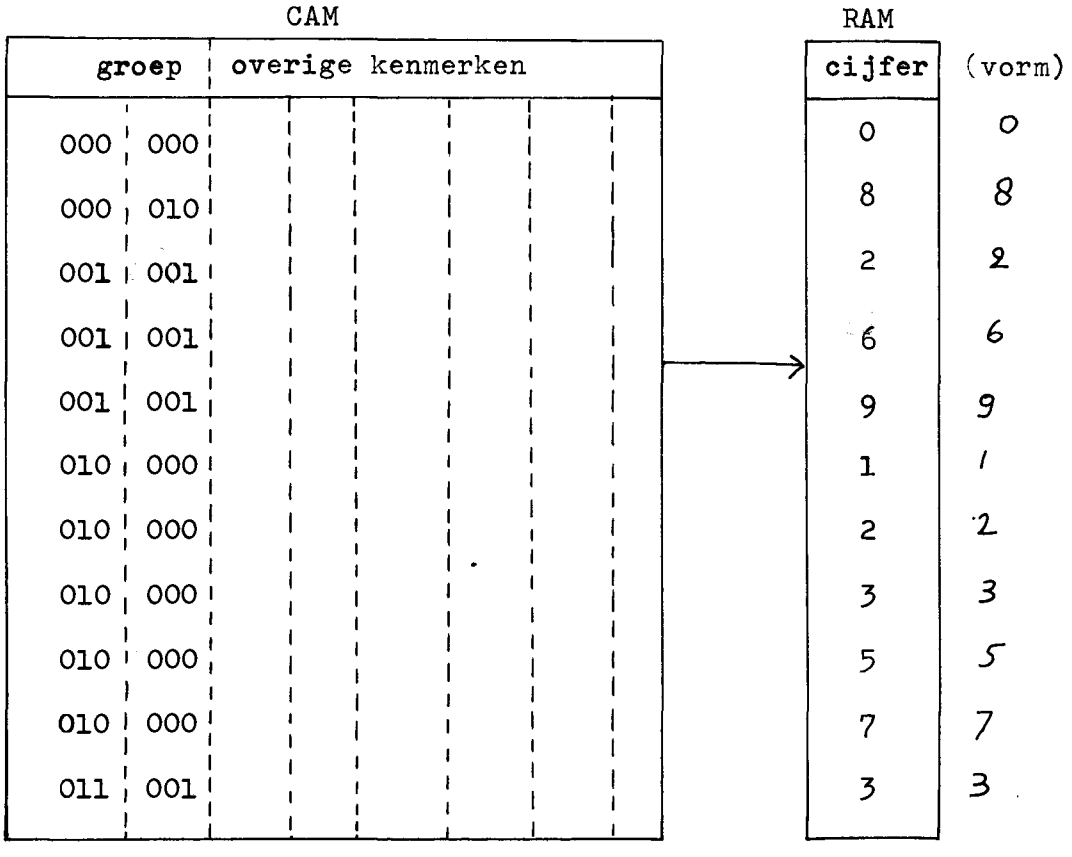


Fig 13.3 Wijze van opslag van de bitstrings, die de verschillende cijfervormen voorstellen in het CAM.

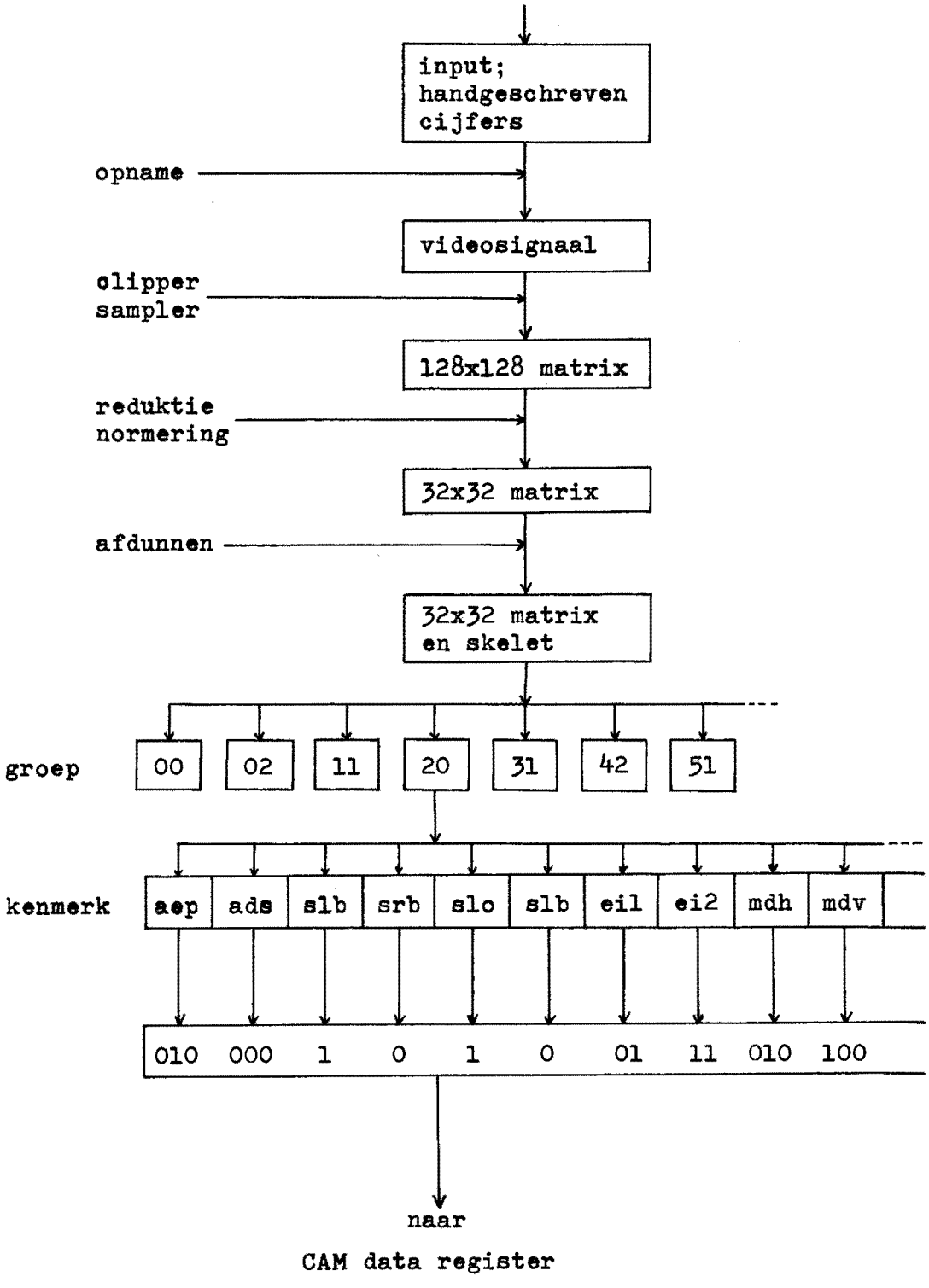


Fig 13.4 Verloop van het herkenningsproces. Er zijn drie fasen, namelijk: opname, voorbewerking, herkenning. Als voorbeeld is hier de 3 van fig 13.1 genomen, waarvoor de karakteristieke bitstring is gegeven.

Fig 13.3 geeft de gegevens van de cijfervormen van enkele groepen, zoals die in het associatief geheugen worden opgeslagen. De overige kenmerken worden bepaald door zeer vele cijfervormen te testen en aan de hand daarvan cijfervormen te selekteren. Tijdens de testfase moet ook worden nagegaan of bepaalde krteria juist zijn gekozen, of dat beter andere kunnen worden gekozen. Als alle bitstrings van alle groepen in het associatief geheugen zijn opgeslagen dan is de leerfase, althans voorlopig, voltooid.
Van een cijfer dat vanaf papier wordt gelezen wordt de betreffende bitstring op de beschreven wijze bepaald en deze wordt aan het geheugen aangeboden.
Treedt een associatie op, dan kan uit het RAM worden uitgelezen welk het cijfer is en nog eventuele bijzonderheden omtrent de vorm van het cijfer, die in het RAM zijn opgeslagen.

Fig 13.4 geeft weer hoe het proces van opname, voorbewerking en herkenning verloopt.

Het kan echter ook voorkomen, dat geen enkel woord in het associatief geheugen associeert. Dan kunnen we nagaan of er een woord in het geheugen bestaat met een approximate match.
Dit kan snel gebeuren door een of meer bits van het kenmerk te maskeren.
Daarbij mogen de bits, die de groep aanduiden nooit gemaskeerd worden.

14 DATA MANAGEMENT MET EEN ASSOCIATIEF GEHEUGEN

14.1 Data management systemen

Data management systemen staan bekend onder verschillende benamingen, zoals
- Data base management systems
- Information retrieval systems
- Management information systems

We zouden dit kunnen vertalen met: systemen voor het beheer van gegevensverzamelingen.

Bij een data base systeem geschiedt de bestands- en opslagbeschrijving los van de programma's, die van de gegevens gebruik maken.
Het goed onderhouden van een bestand houdt in, dat de volgende drie bewerkingen tijdig en juist moeten worden uitgevoerd.

1. Invoegen (inserting) van een aantal records in een bestand.
2. Veranderen (updating) van delen van een of meer records.
3. Verwijderen (deleting) van een of meer records.

Er zijn vele soorten data management systemen in gebruik. Ze varieren van systemen, die tamelijk algemeen zijn tot systemen die op zeer specifieke taken zijn gericht.

14.2 Strukturen bij data management systemen

Voor een dergelijk informatiesysteem moeten grote aantallen gegevens gedefinieerd, opgeslagen en verwerkt worden. Voor de gegevens en hun samenhang kunnen we drie niveau's van strukturering onderscheiden:

- logische struktuur
- opslag struktuur
- fysieke opslag struktuur

1. De logische struktuur wordt ook wel gegevensstruktuur genoemd. De kleinste bouwstenen zijn karakters, waarmee items kunnen worden opgebouwd. Een item heeft een naam en een waarde. Een entity is een verzameling items, die samen voor de gegevensverwerking als een logische eenheid fungeren.
Een entity heeft een naam en een waarde (entry).
Als de waarde van een bepaald item elke entry eenduidig bepaald, dan kan dit item fungeren als sleutel-item.
Entities, die in eenzelfde informatiesysteem voorkomen kunnen onderlinge relaties hebben.

2. De opslagstruktuur.
De logische strukturen moeten op geheugentrajekten worden opgeslagen, dat wil zeggen ze moeten worden afgebeeld op opslagstrukturen.
We bekijken twee soorten opslagstrukturen:

1. Per itemwaarde en per entry.
   Voor elke waarde van een item zal een geheugentrajekt gereserveerd moeten worden; zo'n geheugentrajekt wordt een veld genoemd. Een veld kan een vaste lengte of een variabele lengte hebben. Een vaste lengte is voordelig voor de programmatuur en nadelig wat de benodigde geheugenruimte betreft; voor een variabele lengte geldt het omgekeerde.
   Een record is een aaneengesloten trajekt, waarop een entity-waarde wordt afgebeeld.
   Voor de afbeelding van entries op records zijn vier grondstrukturen mogelijk:
   - fixed structure
   - indexed structure
   - separator structure
   - label structure
   We gaan hier niet verder op in en verwijzen naar referentie 96.

2. Per entity, dus voor een verzameling van entries
   a. In een tabelstruktuur, dat wil zeggen in opeenvolgende geheugentrajekten
   b. In een lijststruktuur; deze komt tot stand door verwijzing van een of meer geheugentrajekten naar een of meer andere geheugentrajekten.
      We onderscheiden hierbij drie soorten strukturen:
      - sequentiele strukturen
      - boomstrukturen
      - netwerkstrukturen

3. Fysieke strukturen
We moeten de opslagstrukturen afbeelden op de opslagstruktuur van het gegeven komputersysteem, dat wil zeggen op de fysieke opslagstruktuur.
Daarbij moeten de beperkingen van het systeem zo goed mogelijk in rekening worden gebracht.
Vaak worden een aantal records verenigd tot een blok, dat met een enkele opdracht tussen kerngeheugen en extern geheugen verplaatst kan worden.

Veel van de gebruikte data management systemen hebben als uitgangspunt hierarchische datastrukturen en geinverteerde lijst-indexering.
Een hierarchische datastruktuur heeft de vorm van een boomstruktuur.
Bij de methode van geinverteerde lijst-indexering wordt de informatie geindexeerd en opgeslagen in verwijzingslijsten.
Om informatie uit te kunnen lezen wordt de verwijzingslijst doorzocht, die sleutels tot de data bevat.
Als de gewenste sleutels gevonden zijn kan aan de hand daarvan de data worden uitgelezen.

14.3 Een associatief geheugen voor data management

De huidige data management systemen zijn in het algemeen geimplementeerd op komputers met een random access geheugen, dat wil zeggen, dat de data is opgeslagen op specifieke plaatsen en dat het verwerken van de data adres-georienteerd is. Daarentegen wordt bij gebruik van een associatief geheugen de informatie opgeslagen op onbekende plaatsen en verwerkt op basis van de inhoud. Omdat veel problemen op het gebied van data management manipulatie van data met zich meebrengen, waarbij het gaat om de inhoud in plaats van het adres, lijkt het gebruik van associatieve geheugens zinvol voor het oplossen van deze problemen.

Voor het werken met grote verzamelingen van gegevens wordt meestal gebruik gemaakt van magneetbanden en adresseerbare werkgeheugens.
Nu de kosten van de hardware voortdurend dalen, en de kosten van software en personeel stijgen, is het belangrijk om na te gaan, wat voor voordelen het toepassen van een associatief geheugen kan hebben bij data management problemen.
Zoekprocessen zijn basisprocessen bij een associatief geheugen en het zijn eveneens basisprocessen bij de bewerkingen die nodig zijn voor het bijhouden van grote bestanden.
Er bestaan zeer vele data management problemen.
Dit zou moeten leiden tot het ontwikkelen van special purpose komputers, die speciaal gekonstrueerd zijn om de processen op het gebied van data management zo snel mogelijk te latan verlopen.

We geven hier in het kort een indikatie hoe een associatief geheugen voor data management gebruikt kan worden.
Een bestand bestaat uit een aantal entities, alle af te beelden op een record. Als een geheugenwoord van het associatief geheugen groter is of gelijk aan de recordlengte, dan kan een entity in een geheugenwoord worden opgelsagen.
Dit is aangegeven in fig 14.1.

Is daarentegen het geheugenwoord te klein, wat vaak het geval zal zijn, dan zullen meerdere geheugenwoorden voor een entity moeten worden gebruikt.
Door middel van linknummers kunnen de bij elkaar behorende deelrecords met elkaar worden verbonden, zoals in fig 14.2 is aangegeven.

Omdat associatieve geheugens kostbaar zijn en een betrekkelijk kleine kapaciteit bezitten, dient voor een zo optimaal mogelijk geheugengebruik te worden gekozen.
De methode die dan gebruikt dient te worden zal sterk afhangen van het data management systeem in kwestie dat men wil implementeren op het associatief geheugen.

naam	adres	woonpl.	geb. datum	opleiding	funktie
NA	AC	WA	DA	OA	FD
NA	AP	WD	DM	OP	FH
NB	AQ	WA	DR	OP	FM
NC	AM	WF	DT	OM	FH
ND	AT	WA	DL	OM	FR

Fig 14.1 Opslag van data in het associatief geheugen als een geheugenwoord lang genoeg is voor een record.

deel-record	link-nummer	naam	adres	woonpl.	
1	1	NA	AC	WA	
1	2	NA	AP	WD	
1	3	NB	AQ	WA	
1	4	NC	AM	WF	
1	5	ND	AT	WA	
		geb. datum	opleiding	funktie	
2	1	DA	OA	FD	
2	2	DM	OP	FH	
2	3	DR	OP	FM	
2	4	DT	OM	FH	
2	5	DL	OM	FR	

Fig 14.2 Opslag van data in het geheugen als een geheugenwoord niet lang genoeg is om een record op te slaan.

deel-record	naam	adres	geb. datum	wpl.	opl.	fkt.
1	NA	AC	DA	1	1	1
1	NA	AP	DM	2	3	2
1	NB	AQ	DR	1	3	3
1	NC	AM	DT	3	2	2
1	ND	AT	DL	1	2	4

	link-nummer	woonplaats
2	1	WA
2	2	WD
2	3	WF

		opleiding
3	1	OA
3	2	OM
3	3	OP

		funktie
4	1	FD
4	2	FH
4	3	FM
4	4	FR

Fig 14.3 Gebruik van linknummers bij de opslag van records in een associatief geheugen om de deelrecords te linken. Dit levert besparing van geheugenruimte, omdat items, die hetzelfde zijn voor een groot aantal records maar eenmaal worden opgeslagen (bijv. woonplaats, opleiding, funktie).

Zo kan men bijvoorbeeld, zoals in fig 14.3 is aangegeven naam en adres in een geheugenwoord opslaan en een aantal (smalle) velden reserveren voor het opslaan van linknummers, die verwijzen naar andere items, zoals woonplaats, opleiding, funktie. In het bestand kunnen namelijk veel personen voorkomen met dezelfde woonplaats of dezelfde opleiding.
De woonplaats en opleiding hoeven ieder dan maar een keer te worden opgeslagen, ieder met het eigen deelrecord-nummer.

Voordelen en nadelen van het gebruik van associatieve geheugens voor data management problemen.

- bij het gebruik van een associatief geheugen zijn weinig extra strukturen nodig voor de representatie van de gegevens in het geheugen.
Er is geen noodzaak voor indexering of voor het werken met pointers. Vanuit informatie-standpunt bezien is een index een redundante komponent bij representatie van data.
Bij gebruik van een associatief geheugen zijn alle voordelen van indexering aanwezig met weinig redundancy.

- als nadelen van het gebruik van een associatief geheugen kunnen de kosten van het geheugen worden genoemd.

In referentie 83 (DeFiore, Bruce Berra) wordt een kwantitatieve analyse gegeven van het gebruik van associatieve geheugens voor data management.
De auteurs hebben mathematische modellen ontwikkeld die kwantitatief retrieval, benodigde geheugenruimte, updating en flexibiliteit beschrijven voor een klasse van huidige data management problemen en voor een associatief systeem.
Er wordt aangetoond, dat de huidige systemen minstens twee maal zo veel geheugenruimte in beslag nemen als een associatief systeem.
De flexibiliteit van een data management systeem geeft aan hoe soepel een aanpassing van het systeem aan gewijzigde eisen, zoals nieuwe ondervragingsklassen en gewijzigde datastrukturen verloopt.

15 KONKLUSIE, NABESCHOUWING

15.1 Konklusie

Of een associatief geheugen gekoppeld aan een general purpose komputer een zinvolle konfiguratie is, hangt af van:

1. De opbouw van het geheugen
2. De grootte van het geheugen
3. De wijze van koppeling met de komputer
4. Het uit te voeren proces
5. Hoe snel het geheugen beschreven kan worden

Ad 1 Het geheugen dient in verband met de gewenste snelheid een bit parallel/woord parallel type te zijn.
De optredende associaties dienen snel gedetekteerd te kunnen worden.

Ad 2 Een associatief geheugen zal in het algemeen een aantal klassen data bevatten. Afhankelijk van de toepassingen stelt dit eisen aan de grootte van het geheugen.
Een groot geheugen is in vele gevallen gunstig, maar ook erg kostbaar.

Ad 3 Een associatief geheugen gekoppeld aan een general purpose komputer via de I/O-kanalen is geen geschikte methode. De I/O-subroutines zullen namelijk te veel tijd nodig hebben voor overdracht van data en instrukties in vergelijking tot de tijd dat het associatief geheugen aktief is.
Een geschikte oplossing is daarom een DMA-kanaal als verbinding tussen het geheugen en de komputer, waarbij het associatief geheugen tamelijk autonoom processen kan uitvoeren.
Een nog betere methode zou zijn het integreren van het associatief geheugen in het random access geheugen van de komputer. Dit vraagt echter een aantal wijzigingen in de general purpose komputer zelf, wat vaak met grote problemen gepaard zal gaan.

Ad 4 Voor vele van de in de voorgaande hoofdstukken beschreven processen moet het geheugen in een aantal velden kunnen worden onderverdeeld. Vaak wordt alleen een deelkenmerk (in een bepaald veld) aangeboden en de rest van het kenmerk wordt gemaskeerd. De bits van het CAM data register en van het maskerregister moeten dus snel veranderd kunnen worden. Bovendien moet de inhoud van deze registers over een op te geven afstand geschoven kunnen worden.

Ad 5 Dit is een zeer belangrijk punt indien grote hoeveelheden data beschikbaar zijn die niet in een keer in het associatief geheugen kunnen worden opgeslagen.
Als slechts weinig operaties op de data in het associatief geheugen te hoeven worden uitgevoerd dan is de procestijd een fraktie van de tijd, die het kost om het geheugen te vullen. Zelfs als het associatief geheugen meer tijd nodig heeft voor het uitvoeren van een proces, dan het kost men een general purpose komputer, kunnen er toch nog voordelen zijn die het gebruik van een associatief geheugen rechtvaardigen. Bijvoorbeeld het gemak dat een associatief geheugen oplevert voor data management problemen of eenvoudiger programma's voor verschillende processen.

15.2 Nabeschouwing

Voor een aantal processen biedt een associatief geheugen duidelijk voordelen; het is echter niet voor alle toepassingen het meest geschikte geheugen.
Het heeft dan ook bestaansrecht naast het random access geheugen.

In een tijd, waarin de kosten van de hardware een dalende tendens vertonen en de software-kosten een steeds groter deel gaan uitmaken van de totale kosten van een komputersysteem, kan een associatief geheugen er toe bijdragen terrein terug te winnen ten gunste van de hardware, wat tevens de verwerkingssnelheid ten goede zal komen.

In de voorgaande hoofdstukken zijn vele processen geanalyseerd en nagegaan is hoe deze met een associatief geheugen uitgevoerd kunnen worden.

De literatuur op dit gebied is daarbij een belangrijke leidraad geweest, zij het dat geen pasklare oplossingen werden geboden. Echter wel indikaties, die enige richting gaven aan het verloop van het onderzoek en de mogelijke toepassingen.

Om enige ervaring op te doen met een associatief geheugen, zowel op het gebied van hardware als het gebied van software is een model gebouwd, waarvan de opbouw, de eigenschappen en de tot nu toe geschreven software in de volgende hoofdstukken worden beschreven.

DEEL III

16 SYSTEEMOPZET.

Na de algemene voorstudie omtrent de toepasbaarheid van een associatief geheugen voelde men spoedig de behoefte om een model te bouwen dat met zo'n geheugen werkt.
De uit het literatuuronderzoek naar voren gekomen processen kunnen m.b.v. dit model gesimuleerd worden, hetgeen van belang is voor een goede procesanalyse.
Bij het bepalen van de systeemopzet voor zo'n model is in eerste instantie gezocht naar een opzet die een snelle uitvoering van de processen garandeert. De systeemopzet is weergegeven in fig.16.1.

Bij deze opzet is voor elk proces een stuk hardware ontworpen. Dit impliceert dat er veel hardware nodig is, waarvan steeds slechts een klein deel aktief is. Alle hardware-blokken zijn, evenals het associatieve deel van het model, verbonden met de data en commando-bus, terwijl de sturing van het geheel verzorgd wordt d.m.v. een microprogramma.
Het microprogramma selekteert de voor een bepaald proces ontworpen hardware en stuurt de communicatie tussen deze hardware en het associatieve deel.
Met deze systeemopzet is het steeds noodzakelijk om nieuwe hardware te ontwerpen zodra een ander proces gesimuleerd moet worden. De hardware zal zelfs aangepast dienen te worden indien een bepaald proces op een andere wijze wordt uitgevoerd.
Dit alles maakt het systeem star waardoor het beslist niet universeel is.
Een ander bezwaar tegen deze systeemopzet is dat de hardware eigenlijk te snel is. Het associatieve deel van het model heeft een snelheid in de orde van 1 microsec.,terwijl het geen probleem is om hardware te bouwen die een factor 5 sneller is.
Omdat uit het literatuuronderzoek bleek dat de processen waarbij het gebruik van een associatief geheugen zinvol is, resulteerde in een aantal basiseisen voor het associatieve deel (zie hoofdstuk 17), is besloten om voor een andere opzet te kiezen. Bij deze nieuwe systeemopzet (zie fig.16.2) is ervan uitgegaan dat de basishandelingen voor het associatieve deel snel uitvoerbaar moeten zijn, terwijl het geheel zo universeel mogelijk moet blijven.

De nieuwe opzet bestaat naast het associatieve deel uit slechts een stuk hardware dat de basishandelingen verzorgt. De besturing van het model gebeurt m.b.v. een microprocessor, waarmee elk gewenst proces gesimuleerd kan worden. Voor elk te simuleren proces dient nu software geschreven te worden, waardoor het systeem aanzienlijk trager is geworden.
De microprocessor genereert de data voor het geheugen en geeft commando's af aan het systeem. Het model interpreteert deze commando's en zorgt ervoor dat de opdrachten uitgevoerd worden.
Afhankelijk van bepaalde testlijnen kan de microprocessor beslissingen nemen die resulteren in nieuwe data en commando signalen voor het model.

Omdat binnen de vakgroep EB een microcomputersysteem (MCS-8) voorhanden is, werd besloten om het te bouwen model te koppelen aan deze microcomputer.

In het volgende hoofdstuk wordt aan de hand van een blokschema het gerealiseerde model beschreven, waarna dieper wordt ingegaan op de werking van de verschillende blokken. Het eerst wordt besproken hoe de timing de verschillende stuursignalen genereert en op welke tijdstippen deze stuursignalen geaktiveerd moeten worden. Daar de telcyclus van de timing van het grootste belang is wordt deze apart besproken evenals het genereren van de pulsen voor het schuiven van de inhoud van het kenmerkregister.
De besturing van het associatieve geheugen is het volgende onderdeel dat uiteengezet wordt. Hier worden de eisen gegeven waaraan deze besturing moet voldoen en hoe deze eisen zijn uitgevoerd.
Vervolgens worden het CAM en het RAM met de benodigde data-besturing besproken.

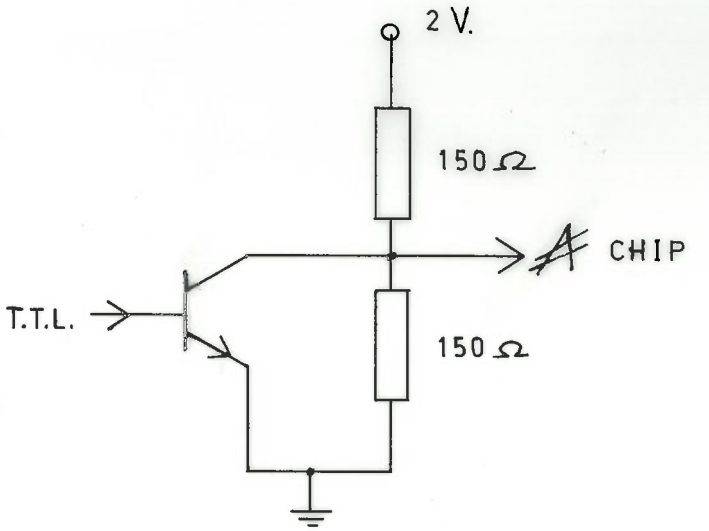
In de laatste hoofdstukken zal worden ingegaan op de ontwikkelde software voor het model. Deze software bestaat uit een aantal basisroutines die ontstaan zijn uit het analyseren van verschillende processen. M.b.v. deze basisroutines is het mogelijk om verschillende processen op het model te simuleren. Tevens wordt het programma voor deze basisroutines gegeven. Voor het testen van het model zijn flow-diagrammen opgesteld, die beschrijven hoe dit testen verloopt. Deze flow-diagrammen en de programma's, die geschreven zijn aan de hand van deze flow-diagrammen, worden tot besluit van dit afstudeerverslag gegeven.

17. HET MODEL.

17.1 Blokschema van het model.

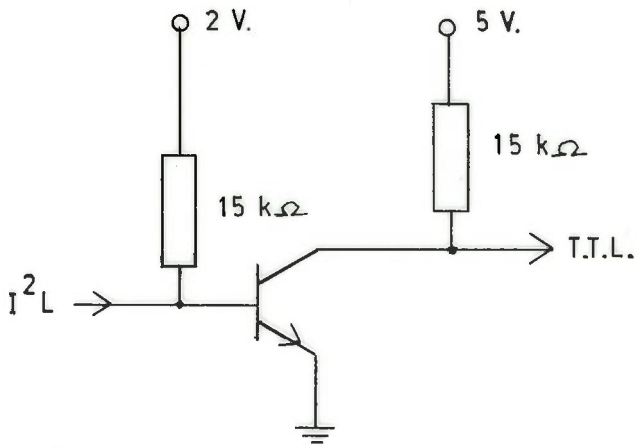
Uit het gedane vooronderzoek omtrent de bruikbaarheid en toepasbaarheid van een associatief geheugen, zijn een aantal eisen gedefinieerd waaraan een te bouwen associatief geheugen moet voldoen.
De manier waarop binnen het gebouwde model aan deze eisen is voldaan, wordt in de volgende paragrafen besproken.
Het model bestaat uit een Content Addressable Memory (CAM) van 8 woorden van 24 bit, een besturing van het CAM, een Random Access Memory (RAM) dat even groot is en de noodzakelijke data besturing voor het geheel.
De opbouw is zodanig dat zowel het aantal woorden als het aantal bits per woord eenvoudig uitbreidbaar is.
Het model is gekoppeld aan de MCS-8 die de voor het model noodzakelijke input levert (data en commando signalen) en de output verwerkt (data en test signalen).
Het is de bedoeling dat met behulp van het model algorithmes worden uitgevoerd om tot een goede procesanalyse te komen.
De software voor het testen van het model en het uitvoeren van de basisoperaties is geschreven.

Omdat het CAM met behulp van de IIL-techniek geintegreerd is en de besturing gebruik maakt van TTL circuits is een aanpassing tussen beide technieken noodzakelijk. Deze aanpassing is reeds op de chip geintegreerd maar bleek niet optimaal te functioneren. Voorlopig is daarom een extra aanpassing toegevoegd in afwachting van een onderzoek hoe de aanpassing op de chip verbeterd kan worden.
Fig 17.1.a geeft deze extra aanpassing weer.
Deze extra aanpassing is vereist voor de aansluitingen van de associatie- en commandelijnen van de geheugenchip.
Fig 17.1.b geeft de aanpassing voor de overgang IIL→TTL.
Deze aanpassing is alleen voor de aansluitingen van de leeslijnen van de geheugenchip noodzakelijk.
De voedingsspanning voor de associatieve geheugenchip is 2 V.



AANPASSING T.T.L. → CHIP

FIG. 17.1a



AANPASSING I^2L → T.T.L.

FIG. 17.1b

Uit het schema van fig.17.2 blijkt dat de communicatie tussen de MCS-8 en het model gebeurt via de in- en outputpoorten van selectiekaarten die verbonden zijn met de multiplex-bus van de MCS-8.
De in- en outputpoorten zijn 8 bit breed, waardoor het datatransport byte georganiseerd is.
De data en masker-bus zijn 8 bit breed, terwijl de commando-bus uit 19 lijnen bestaat. Uit deze getallen blijkt dat er 5 outputpoorten noodzakelijk zijn. Het aantal inputpoorten bedraagt 2, een voor data-input en een om een drietal testlijnen aan de MCS-8 te koppelen.

De timing zorgt ervoor dat het RAM, de data-besturing van het CAM en het RAM en de besturing van het CAM de noodzakelijke stuursignalen krijgen. Zodra de commandolijn START geaktiveerd wordt zal de timing afhankelijk van het toegevoerde commando de benodigde stuursignalen genereren. Wanneer de opdracht is uitgevoerd zal de timing een READY-signaal geven. Via de data en masker-bus kan data aangeboden worden aan het CAM. Het bitpatroon op de masker-bus zorgt ervoor dat bepaalde bits van de data gemaskeerd worden (zie par.17.6).
Via dezelfde data-bus wordt informatie aan het RAM aangeboden.
De besturing voor het CAM zorgt ervoor dat informatie in het CAM geschreven of gewist wordt. Bovendien detecteert deze besturing of informatie die geschreven is associeert met de informatie in het kenmerkregister en of het CAM vol is.
Hij bepaald bovendien het adres van het eerste niet beschreven woord en indien een woord associeert met de informatie in het kenmerkregister, wordt het adres van dit woord gegenereerd en opgeslagen in het CAO-register (Cam Address Output-register).
Indien meerdere woorden associeren dan kunnen de adressen van deze woorden worden uitgelezen door T.V.
(Tijdelijk Vrij) pulsen aan de besturing aan te bieden.
Aan de besturing kan via de data-bus ook een adres worden aangeboden waarop in het CAM geschreven of gewist moet worden.
Het RAM kan op adres geschreven of uitgelezen worden.
Het adres kan via de data-bus door de MCS-8 of via het CAO register worden toegevoerd.
Indien het RAM is uitgelezen wordt de informatie opgeslagen in het outputregister alvorens de MCS-8 de informatie ter verwerking via een inputpoort inklokt.

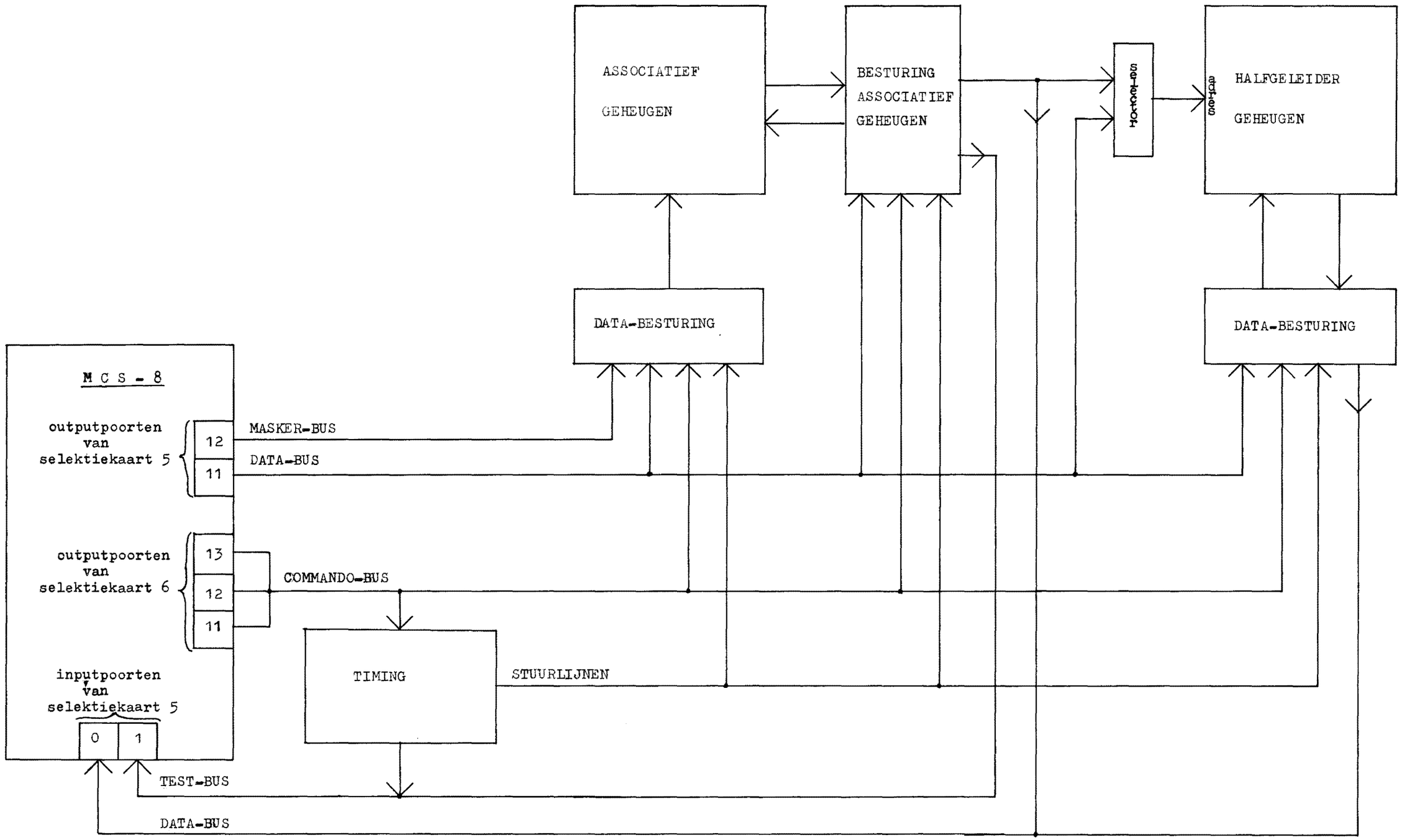


FIG. 17.2

Blokschema van het model.

17.2 De timing.

De input van de timing bestaat uit een aantal commandolijnen die door de MCS-8 geaktiveerd kunnen worden. De output omvat de stuurlijnen die signalen leveren voor een goede werking van het model.
Om in het vervolg een onderscheid te maken tussen de verschillende lijnen zal een commandolijn aangeduid worden met C-naam, een stuurlijn met S-naam en een testlijn met T-naam.
Voor een goede werking van het model is het van het grootste belang dat de stuursignalen op het juiste moment gegenereerd worden.
Zoals uit het schema van fig.17.3 blijkt, bestaat de timing uit een centrale klok, een handklok, een tweetal counters en een aantal flip-flops en poorten.
De timing maakt gebruik van een centrale klok met een (instelbare) frekwentie van 4 MHz.
De cyclus waarin de stuurlijnen aktief kunnen zijn wordt bepaald door counter 1. Deze begint te tellen even nadat de commandolijn START hoog is geworden en wekt na de 16e telpuls een READY-signaal op, ten teken dat de cyclus beeindigd is.
De tijdcyclus bedraagt ongeveer 4 microsec.
De mogelijkheid bestaat om de cyclus single-step door te lopen.
Uit het tijddiagram van fig.17.4 is te zien op welk tijdstip binnen de telcyclus een bepaalde stuurlijn aktief is wanneer hij door een commandolijn wordt aangestuurd.
In tabel 17.5 is aangegeven welke stuurlijnen binnen de telcyclus geaktiveerd moeten worden indien een bepaalde commandolijn aktief is.

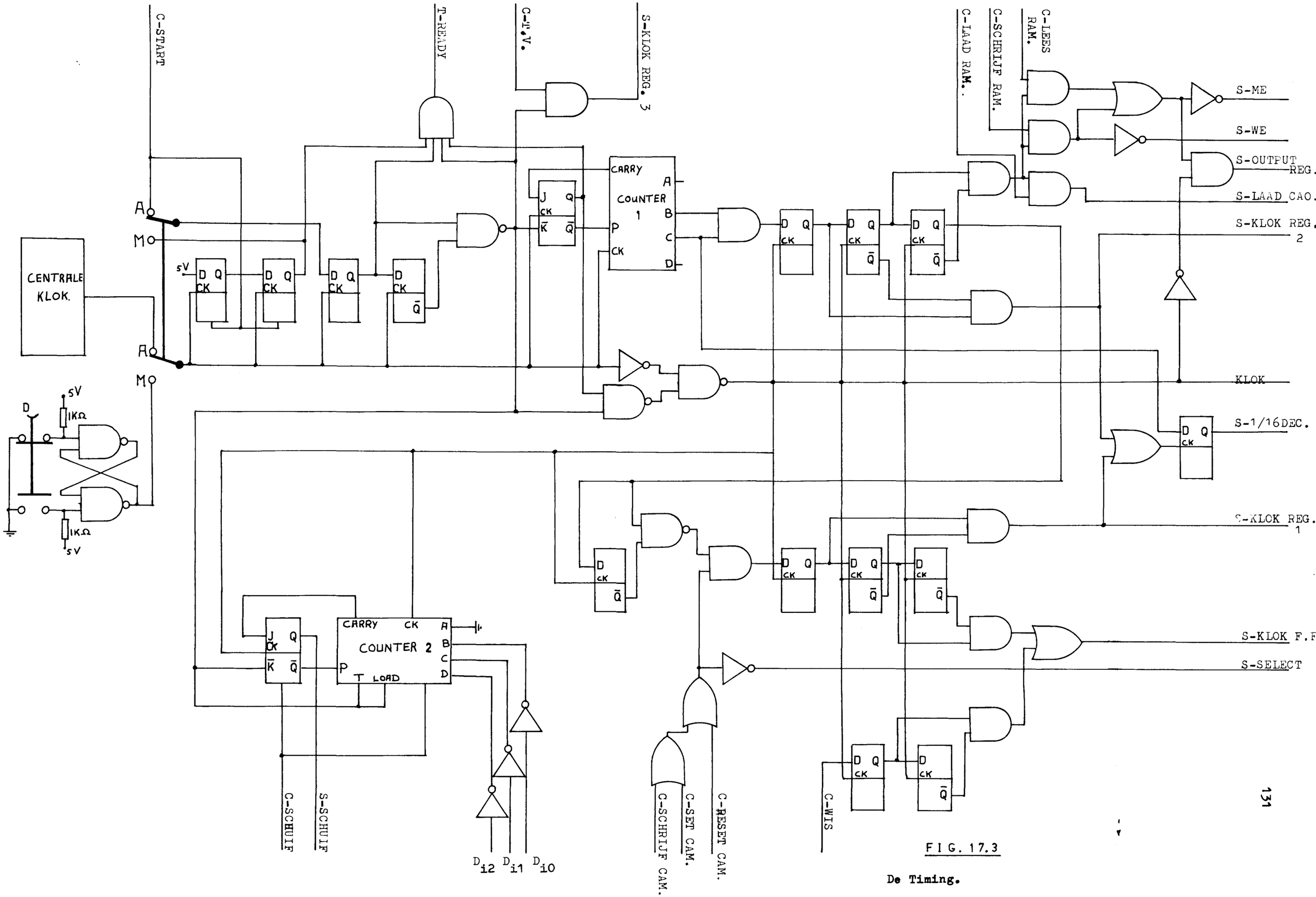


FIG. 17.3

De Timing.

De timing genereert de benodigde stuursignalen voor het model.

1. De stuursignalen voor het RAM zijn :
   S-ME
   S-WE
   S-OUTPUT REG
   Deze drie lijnen worden geaktiveerd door de commandolijnen C-LEES RAM en C-SCHRIJF RAM.
2. De stuurlijnen voor het CAM en haar besturing zijn :
   -S-KLOK REG 2 ; deze lijn wordt in elke telcyclus geaktiveerd en is dus onafhankelijk van commandolijnen.
   -S-KLOK REG 1,
   S-KLOK F.F.,
   S-SELECT,
   S-1/16 DEC ;deze lijnen worden geaktiveerd indien een schrijf of wis opdracht aan het CAM gegeven wordt.
   De commandolijnen die een schrijf of wis opdracht aan het CAM geven zijn:
   C-SCHRIJF CAM (1e lege pl.)
   C-SET CAM (schrijf op adres)
   C-RESET CAM (wis op adres)
   C-WIS CAM (wis op associatie)
   Voor de juiste aansturing zie tabel17.5.
   -S-SCHUIF ; deze lijn wordt aktief wanneer de data in het kenmerk register geschoven moet worden. De commandolijn C-SCHUIF zal hoog zijn en bovendien moet de MCS-8 op de ingangen BCD van counter 2 het aantal te schuiven posities aangeven (hiervoor worden de drie minst significante bits van de data-bus gebruikt).
   -S-LAAD CAO ; deze lijn wordt aktief wanneer een associatie adres uitgelezen moet worden. De C-LAAD CAO zal deze stuurlijn aktiveren. Het adres wordt uitgelezen via de drie minst significante bits van de data-bus.
   -S-KLOK REG 3 ;deze lijn wordt aktief wanneer de C-T.V. hoog wordt. Het associatie adres wordt dan ter verwerking in een register geklokt.

Indien een stuurlijn aan het begin van de cyclus aktief moet zijn, dan wordt een puls gemaakt met behulp van een aantal D-flip-flops. Wanneer een stuurlijn aan het eind van de cyclus aktief moet zijn (b.v S-ME en S-WE), dan wordt de uitgang van counter 1 gebruikt om met behulp van D-flip-flops en poorten op het juiste moment een puls te krijgen.

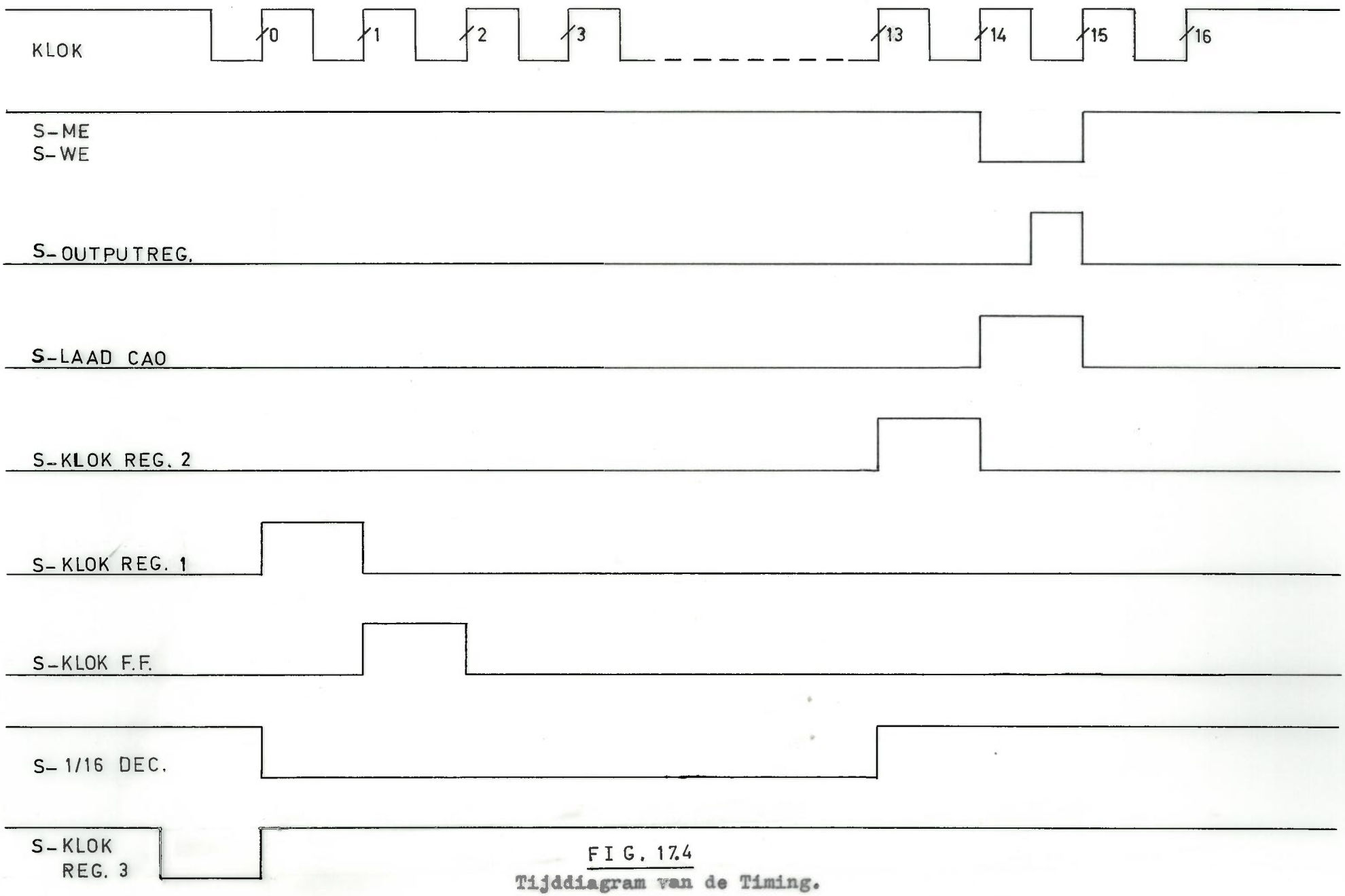


FIG. 17.4

Tijddiagram van de Timing.

Door de MCS-8 te aktiveren commandolijn :	Door de timing te aktiveren stuurlijn(en) :
1. C-LEES-RAM	S-ME S-OUTPUT REG
2. C-SCHRIJF-RAM	S-ME S-WE S-OUTPUT REG
3. C-LAAD-CAO	S-LAAD CAO
4. C-SCHRIJF-CAM	S-KLOK REG 1 S-KLOK F.F. S-SELECT S-1/16 DEC
5. C-SET-CAM	S-KLOK REG 1 S-KLOK F.F. S-SELECT S-1/16 DEC
6. C-RESET-CAM	S-KLOK REG 1 S-KLOK F.F. S-SELECT S-1/16 DEC
7. C-WIS	S-KLOK F.F.
8. C-SCHUIF	S-SCHUIF
9. C-T.V.	S-KLOK REG 3

De stuurlijn die in elke cyclus geaktiveerd wordt is :
KLOK REG 2-lijn

C-START : start de Timing
T-READY : testsignaal voor de MCS-8

tabel 17.5

17.3 De telcyclus.

Zoals uit het voorgaande duidelijk is geworden zal de timing gaan werken nadat de commandolijn START hoog is geworden.
In fig.17.6 is het schema gegeven van de schakeling die er voor zorgt dat de cyclus op de juiste wijze wordt gestart en gestopt. Indien schakelaar S in stand A (automatic) staat, dan zijn de flip-flops I en II van geen belang. De commandolijn START is dan verbonden met de J-ingang van flip-flop III. Omdat de commandolijn voordat hij aktief wordt laag is, geldt voor de uitgangen Q van flip-flop III (Q_{III}) en $\bar{Q}$ van flip-flop IV ($\bar{Q}_{IV}$) dat deze respectievelijk laag (0) en hoog (1) zijn. De $\bar{K}$-ingang van flip-flop V zal hoog zijn, waardoor de Q-uitgang van flip-flop V (hoog) er voor zorgt dat NAND-poort A geen klokpulsen doorlaat. Counter 1 zal niet tellen omdat zijn P-ingang laag is.
De commandolijn START wordt nu hoog. Het gevolg hiervan is dat de stand van Q_{III} $\bar{Q}_{IV}$ na een klokpuls 1 1 is, waardoor flip-flop V op de flank van de tweede klokpuls gereset wordt. Deze tweede klokpuls zorgt er ook voor dat Q_{III} $\bar{Q}_{IV}$ in de stand 1 0 komt, waardoor de $\bar{K}$-ingang van flip-flop V gedurende het verdere verloop van de cyclus hoog zal zijn (flip-flop V wordt niet meer gereset). De $\bar{Q}$-uitgang van flip-flop V is hoog geworden, waardoor counter 1 gaat tellen. De carry-puls die counter 1 tijdens de 15e telpuls genereert, zorgt ervoor dat flip-flop V op de flank van de 16e puls gereset wordt, waardoor counter 1 in de nulstand stopt en poort A, die de 16 telpulsen heeft doorgelaten, weer blokkeert. Uit het tijddiagram van fig.17.7 blijkt dat er 17 positieve flanken tijdens de telcyclus gegenereerd worden. De 0e flank wordt verkregen doordat de $\bar{K}$-ingang van flip-flop V een klokpuls eerder laag is dan de Q-uitgang van flip-flop V, met het gevolg dat de puls die flip-flop V reset reeds door poort A wordt doorgegeven (dit is dus niet de eerste telpuls voor counter 1).

Om de cyclus single-step door te lopen, moet de schakelaar in stand M (manual) staan. Het hierboven beschreven proces gaat er van uit dat alvorens de C-START aktief wordt Q_{III} $\bar{Q}_{IV}$ in de stand 0 1 staat. Deze stand wordt gegenereert in de tijd dat de commandolijn C-START laag is. Met S in de stand A genereert de Centrale Klok namelijk voldoende klokpulsen om in deze beginstand te komen. Indien S in stand M staat is het echter onmogelijk om m.b.v de handklok de pulsen te geven die Q_{III} $\bar{Q}_{IV}$ in de beginstand zetten. Om toch vanuit deze beginstand het proces te kunnen doorlopen zijn de flip-flops I en II in de schakeling opgenomen. De ingang van flip-flop I is met de voedingsspanning verbonden. Wanneer S in stand M staat is Q_{II} met de ingang van flip-flop III verbonden, terwijl de commandolijn C-START alleen de clear-ingangen van de flip-flops I en II stuurt.

Wanneer de commandolijn C-START hoog wordt
dan geldt voor de uitgangen achtereenvolgens:

	Q_{I}	Q_{II}	Q_{III}	$\bar{Q}_{IV}$
na START commando:	0	0	1	0
na 1e klokpuls:	1	0	0	0
na 2e klokpuls:	1	1	0	1
na 3e klokpuls:	1	1	1	1
na 4e klokpuls:	1	1	1	0

De begintoestand voor het beschreven proces is bereikt na de 2e klokpuls. De $\bar{K}$-ingang van flip-flop V is dan laag. De 4e klokpuls reset flip-flop V, waardoor counter 1 gaat tellen en het proces start.

Wanneer de telcyclus voorbij is (na ± 4 micro-seconden) genereert de timing een READY-signaal waaruit het aan het model gekoppelde systeem (MCS-8) mag concluderen dat een volgende instruktie uitgevoerd kan gaan worden. Omdat het model veel sneller is dan de MCS-8 (instruktietijd ± 12,5 micro-seconden) is dit READY-signaal voor deze koppeling niet strikt noodzakelijk.

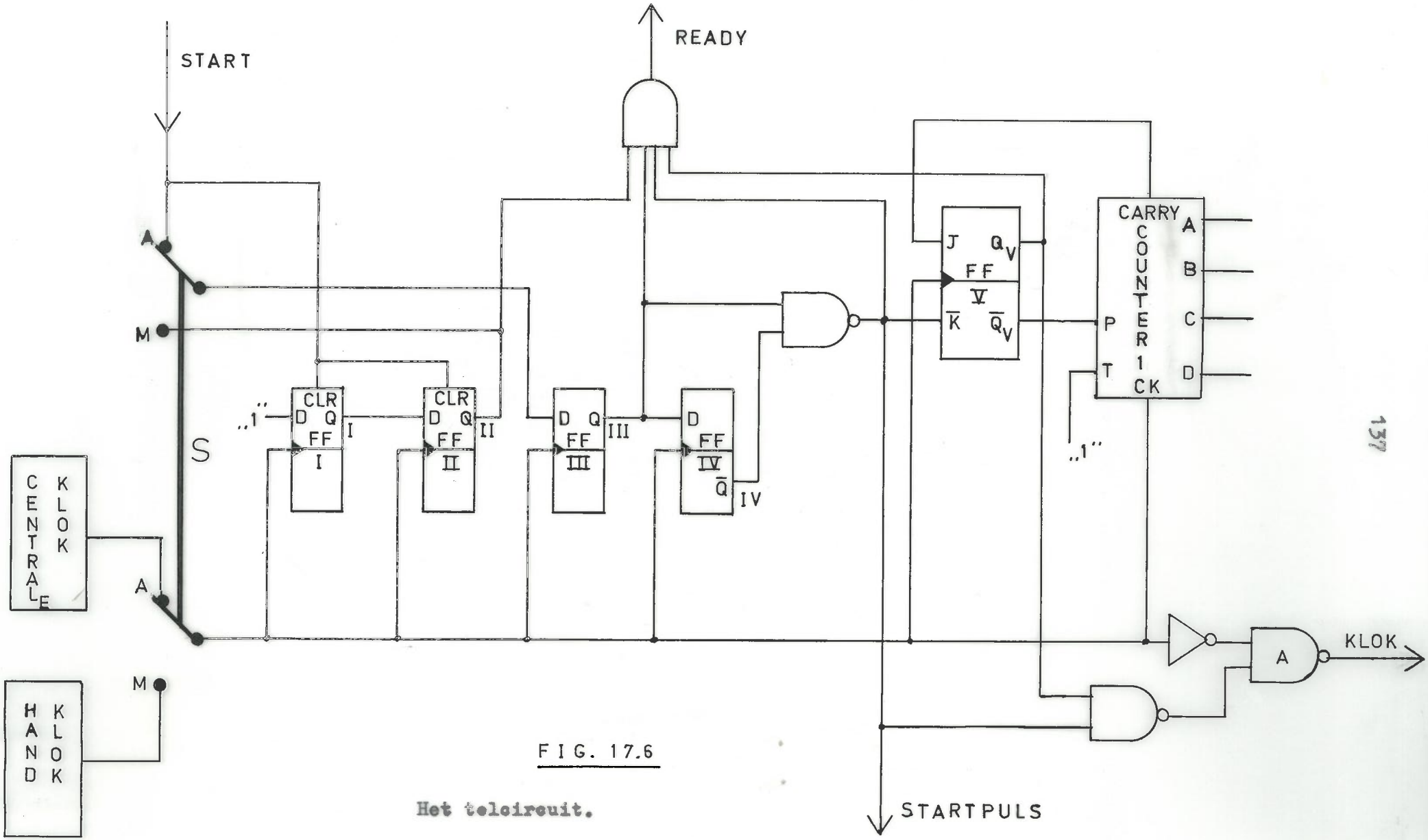


FIG. 17.6

Het teleircuit.

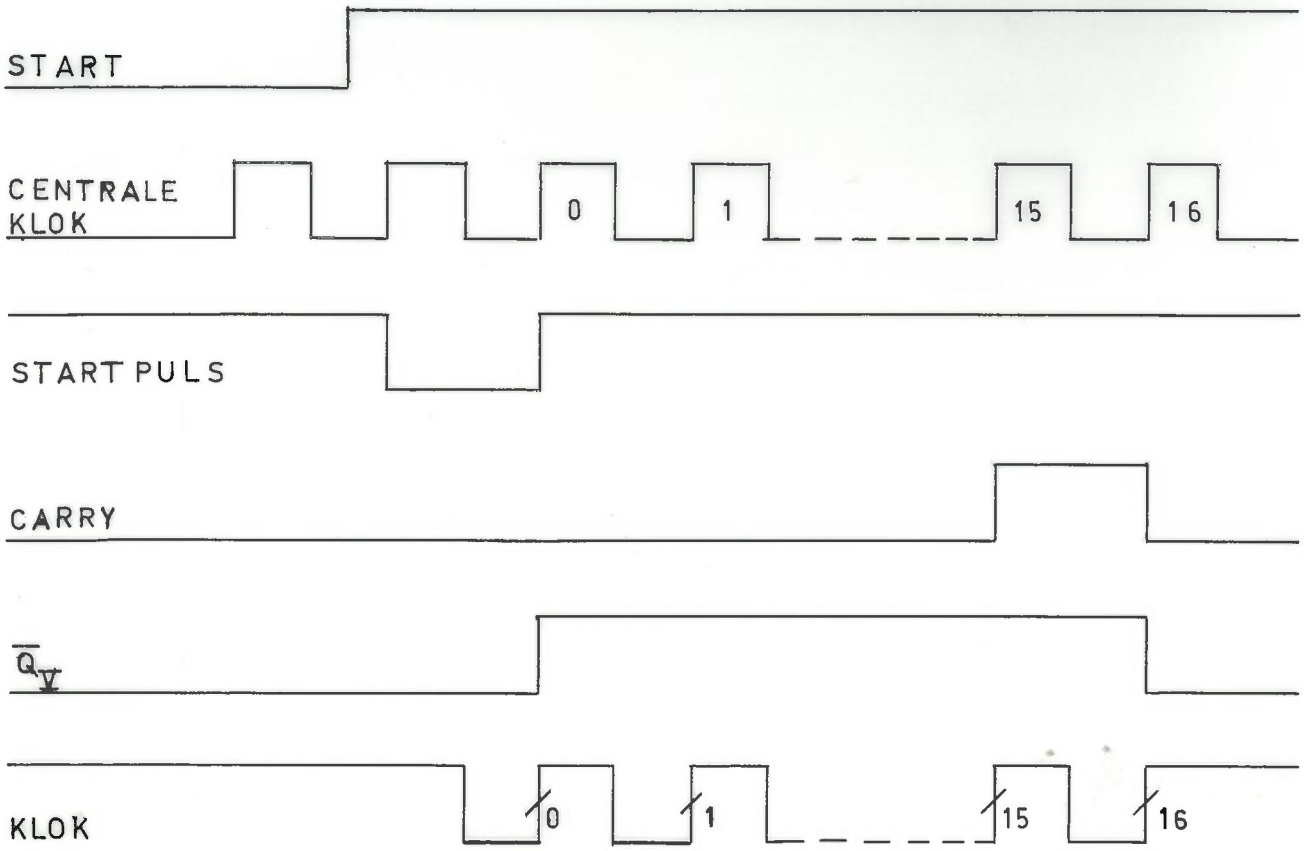
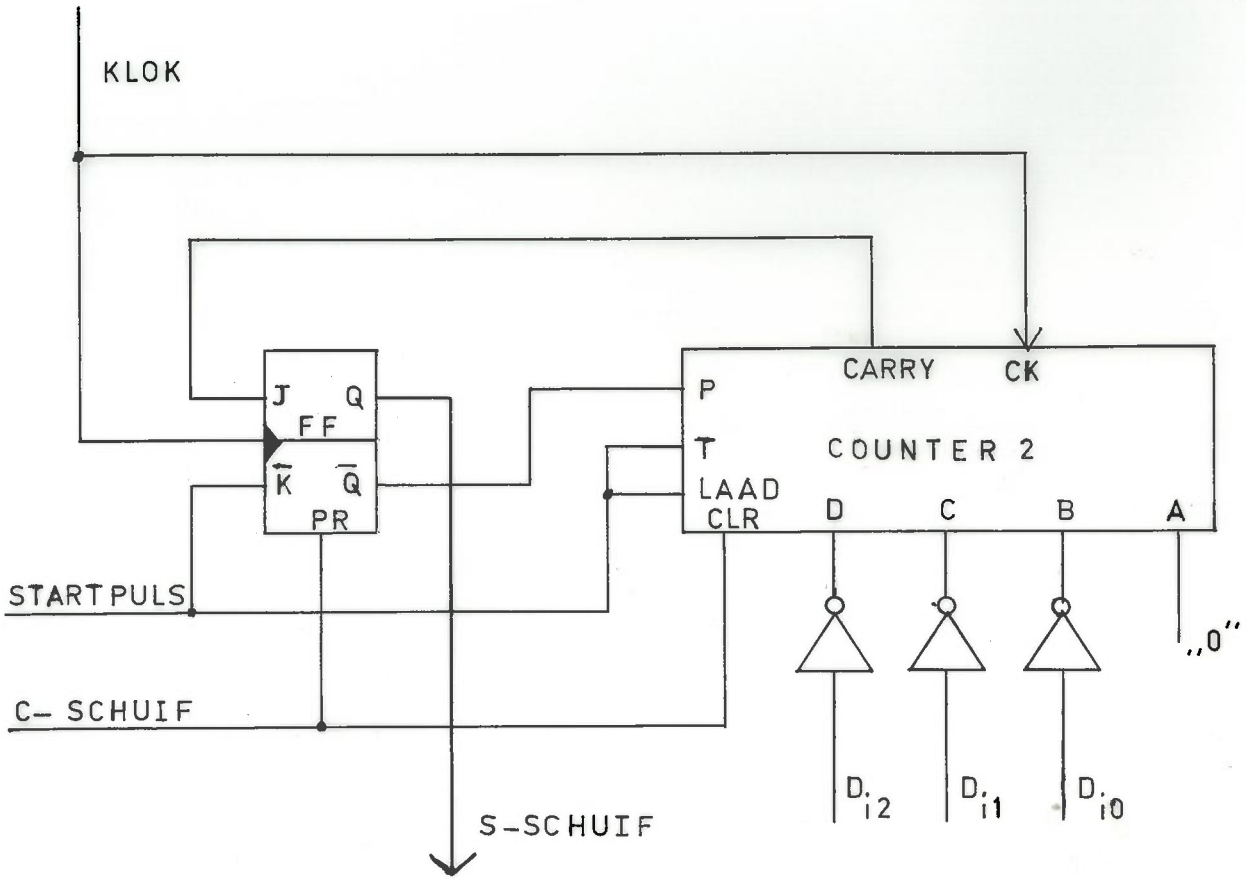


FIG. 17.7

Tijddiagram van het telcircuit.

17.4 Schuiven.

Zoals uit het voorgaande blijkt is het mogelijk om data in het kenmerkregister van het CAM over een gewenst (maximaal 16) aantal posities te schuiven. In fig.17.8 is het schema en het tijddiagram voor de schakeling weergegeven. Indien de commandelijn C-SCHUIF geaktiveerd is dan zal de $\overline{R}$-ingang van de flip-flop enige tijd na het startsignaal een kortstondig laag niveau opgedrukt krijgen. Dit niveau wordt doorgegeven aan de T en LAAD inputs van de counter, waardoor de informatie die dan aan de ingangen A,B,C en D van de counter staat wordt ingeklokt. De T en LAAD ingangen worden hierna hoog en bovendien zal de P-ingang, door het resetten van de flip-flop, hoog worden, waardoor de counter gaat tellen vanaf de ingeklokte waarde tot 16. Tijdens de 15e telpuls wordt een carry-puls gegenereerd waardoor de flip-flop op de flank van de 16e puls geset wordt en de counter stopt (P-ingang laag!). Gedurende de telcyclus van de counter is de Q-uitgang van de flip-flop laag waardoor klokpulsen doorgegeven kunnen worden aan het kenmerkregister van het CAM (zie par.17.6).
Om er voor te zorgen dat de Q-uitgang van de flip-flop hoog is als er niet geschoven dient te worden, is de commandolijn C-SCHUIF verbonden met de preset ingang van de flip-flop.
De teller staat voor het inklokken van data in de nulstand. Omdat ingang A van de teller met aarde (0) verbonden is, zal de Q-uitgang van de flip-flop (S-SCHUIF) steeds gedurende $2(n+1)$ klokpulsen laag blijven indien de ingangen B,C en D het getal n weergeven ($0 \leqslant n \leqslant 7$).
Deze transformatie is noodzakelijk omdat elke bit in het CAM vertegenwoordigd wordt door twee bits in het kenmerkregister. Wanneer dus over K bits in het CAM geschoven wordt, moeten er aan het kenmerkregister 2K klokpulsen voor het schuiven gegeven worden.
De ingangen B,C en D van de counter worden d.m.v. de drie minst significante bits van de data-bus gestuurd.



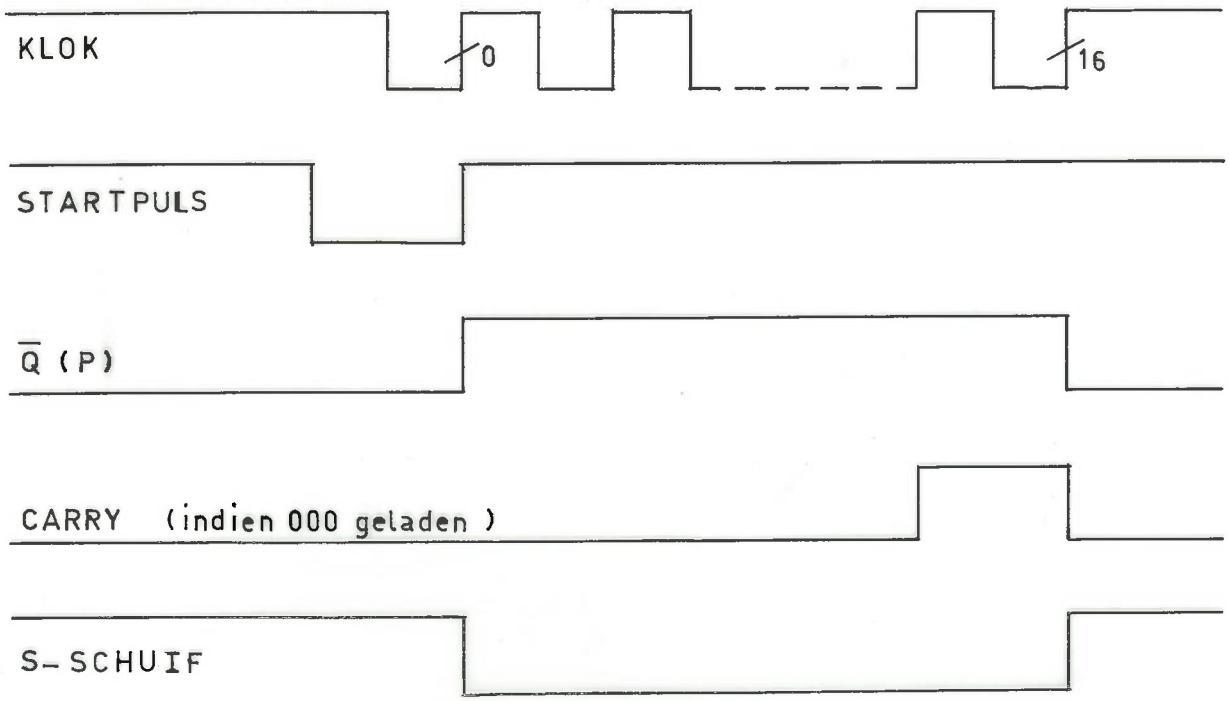


FIG. 17.8

Het schuifcircuit en het tijddiagram.

17.5 Besturing van het CAM.

De eisen waaraan deze besturing moet voldoen zijn door R. DIJKSTRA opgesteld. De besturing is door hem in IIL-techniek ontworpen, maar de chip is nog niet gereed.
De besturing in het model is daarom gebouwd m.b.v. TTL circuits. Van belang is te weten dat er nog een eis is bijgevoegd waaraan de besturing moet voldoen.
In tabel 17.10 worden de eisen daarom nogmaals opgesomd.
Fig 17.9 geeft het schema van de besturing van het CAM zoals deze voor het model is gebouwd.

Elk woord van het CAM is in de besturing vertegenwoordigd door een JK-flip-flop. De stand van de flip-flop zegt of het overeenkomstige woord in het CAM al dan niet beschreven is. De uitgangen van de flip-flops zijn verbonden met de ingangen van priority-encoder 1 (P.E.-1), die aan zijn uitgang het adres geeft van het eerste woord dat niet beschreven is (de overeenkomstige ingang is dan laag).

Met de data-selector kan nu of het adres van dit eerste lege woord, of een door de MCS-8 opgedrukt adres gekozen worden, dat in register 1 wordt geklokt.
Met behulp van de D-ingang van de 1/16 decoder wordt een der outputs 1 t/m 8 of 9 t/m 16 gekozen. Een der outputs 1 t/m 8 wordt gekozen indien een schrijfopdracht aan het CAM wordt gegeven. De gekozen output wordt laag waardoor de flip-flop geset en de commandolijn, die met het betreffende woord in het CAM verbonden is, geaktiveerd (hoog) wordt.
De inputs ABC van de 1/16 decoder zijn verbonden met register 1 en geven het adres waarop geschreven moet worden.

Indien ingang D een der outputs 9 t/m 16 selecteert, dan moet er op adres gewist worden. Het adres wordt dan door de MCS-8 opgedrukt en staat weer op de ingangen ABC van de 1/16 decoder. Bij het wissen wordt de overeenkomstige flip-flop gereset.
Het is ook mogelijk om de woorden te wissen waarvan de inhoud associeert met de informatie in het kenmerkregister van het CAM. Daartoe is van elk woord uit het CAM de leeslijn tesamen met de commandolijn C-WIS verbonden met een NAND-poort. Indien nu een of meerdere woorden van het CAM associeren, zullen de overeenkomstige leeslijnen hoog worden. Omdat de commandolijn ook hoog is, zullen de betreffende flip-flops gereset worden en is de wisopdracht uitgevoerd.

Als een woord associeert moet de overeenkomstige JK-flip-flop geset en de leeslijn hoog zijn.

Om het adres van het eerste woord dat associeert te genereren, zijn daarom de leeslijnen tesamen met de uitgangen van de JK-flip-flops via poorten verbonden met de ingangen van priority-encoder 2. Ook zijn op de ingangen van deze poorten de uitgangen van een poortschakeling aangesloten.
De ingangen van de poortschakeling worden aangestuurd door een 1/8 decoder. Deze decoder wordt alleen enabled als er een C-T.V. gegeven is. De uitgangen van de poortschakeling zijn daarom normaliter hoog. Indien een woord geschreven is (JK-flip-flop geset) en de leeslijn is hoog dan zal de desbetreffende ingang van priority-encoder 2 laag worden. Priority-encoder 2 genereert nu het adres van het eerste woord dat associeert. Dit adres wordt aan het eind van elke telcyclus in register 2 en in het C.A.O. register geklokt.

Als de commandolijn C-T.V. geaktiveerd wordt dan zal de 1/8 decoder enabled worden. Het adres in register 2 wordt in register 3 geklokt, waardoor de overeenkomstige uitgang van de decoder laag wordt. De poortschakeling zorgt ervoor dat alle woorden die associeren en een adres lager en gelijk aan het adres uit register 3 hebben, onderdrukt worden.
Op het eind van de telcyclus zal het adres van het volgende woord dat associeert in register 2 geklokt worden.
Zodoende kunnen de associatie-adressen sequentieel uitgelezen worden.
Na een T.V. commando moet altijd een T.V.-RESET commando gegeven worden. Dit in tegenstelling tot de besturing die geintegreerd wordt, waar een verandering in het kenmerkregister, waarop de woorden die een T.V.-puls ontvangen hebben niet associeren, automatisch impliceert dat wanneer de oude data weer aangeboden wordt de gegeven T.V.-pulsen geen invloed meer hebben.

Zoals uit tabel 17.10 blijkt, moet de besturing signaleren of er al dan niet een woord associeert en of het CAM vol is. De lijnen die deze signalering geven zijn respectievelijk verbonden met de EO-output van priority-encoder 2 en de GS-output van priority-encoder 1. Indien een of meerdere woorden associeren, dan zullen een of meerdere inputs van priority-encoder 2 laag worden en zal de EO-output hoog zijn.
Indien geen enkel woord associeert dan is de EO-output laag.
Wanneer het CAM vol is, dan zullen alle flip-flops geset zijn, d.w.z. dat alle inputs van priority-encoder 1 hoog zijn.
De GS-output van deze encoder is dan ook hoog, terwijl deze output in alle andere gevallen laag is.

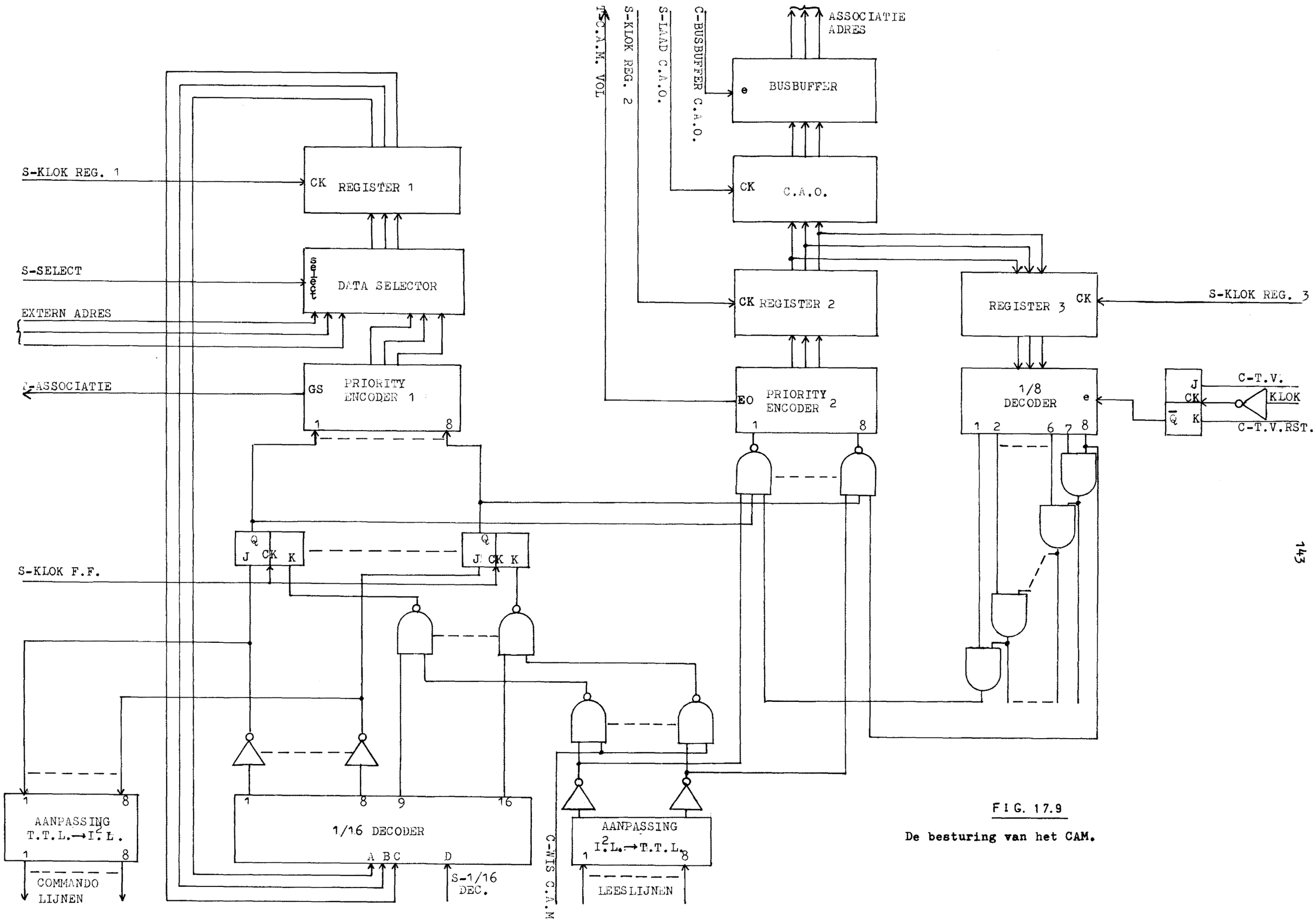


FIG. 17.9

De besturing van het CAM.

Eisen waaraan de besturing voor het CAM moet voldoen.

1) Bijhouden welke woorden geschreven zijn.
2) Aanwijzen van de eerste lege plaats in het CAM.
3) Schrijven van informatie op de eerste lege plaats in het CAM.
4) Schrijven van informatie op een adres in het CAM.
5) Wissen op een adres in het CAM.
6) Wissen op een associatie-adres in het CAM.
7) Genereren van het adres van het eerste woord dat associeert.
8) Het kunnen genereren van de adressen van alle woorden die associeren (in volgorde).
9) Signaleren indien er een woord associeert.
10) Signaleren indien het CAM vol is.

Tabel 17.10.

17.6 Het CAM en zijn data-besturing.

De data-besturing moet de volgende funkties kunnen realiseren:

a) het per byte inlezen van het kenmerkregister.
b) het vanaf een bepaalde byte laden van het kenmerkregister met dezelfde informatie.
c) het schuiven van de data in het kenmerkregister over een even (maximaal 16) aantal posities.

In fig.17.11 is het CAM en zijn data-besturing weergegeven. Het in het model opgenomen CAM bestaat uit 8 woorden van 24 bit. Dit betekent dat er 48 associatielijnen aangestuurd moeten worden. Daartoe is het kenmerkregister opgebouwd uit 6 schuifregisters. M.b.v. de mode control ingangen S_0 en S_1 kunnen de schuifregisters parallel geladen of hun inhoud parallel geschoven worden. Elke byte (8 bit) in het CAM heeft twee schuifregisters nodig om de 16 associatielijnen aan te kunnen sturen. Om de beschrijving van de data-besturing te vereenvoudigen wordt voortaan gesproken over schuifregister 0,1 en 2 waarmee respectievelijk de schuifregisters 1 en 2, 3 en 4, 5 en 6 bedoeld worden.

De data-besturing voor het CAM bestaat naast de schuifregisters uit een poortschakeling die de data en het masker combineert, een decoder en een poortschakeling voor de sturing van de mode control ingangen S_0 en S_1 van de schuifregisters. De poortschakeling die de data en het masker combineert bestaat uit NAND poorten. Als een maskerbit "1" is dan zullen de uitgangen van de desbetreffende twee NAND poorten afhankelijk zijn van de overeenkomstige data-bit. Is namelijk de data-bit "0" dan zijn de uitgangen "1 0", is de data-bit "1" dan zijn de uitgangen "0 1". Indien het masker-bit "0" is dan zullen de uitgangen van de twee NAND poorten onafhankelijk zijn van de data-bit. Deze uitgangen zijn dan altijd "1 1". We zeggen nu dat de data-bit gemaskeerd is.

De decoder heeft 2 adresingangen die uitgevoerd zijn als commandolijnen, terwijl ook de enable ingang door een commandolijn (C-LAAD CAM) gestuurd wordt. De poortschakeling voor de sturing van de S_0 en S_1 ingangen van de schuifregisters krijgt de commandolijn C-LAAD VANAF en de stuurlijn S-SCHUIF toegevoerd.
Wanneer de commandolijnen niet geaktiveerd zijn hebben deze een laag niveau. Als de stuurlijn niet geaktiveerd is dan is deze hoog. Wordt nu C-LAAD CAM geaktiveerd dan zullen, afhankelijk van C-ADRES 1 en C-ADRES 2, de ingangen S_0 en S_1 van schuifregister 0, 1 of 2 geaktiveerd worden waardoor dit schuifregister parallel geladen wordt. Worden C-LAAD CAM en C-LAAD VANAF geaktiveerd dan zal, afhankelijk van C-ADRES 1 en

C-ADRES 2, schuifregister 0, 1 en 2 of schuifregister 1 en 2 of schuifregister 2 parallel geladen worden met dezelfde informatie. De stuurlijn S-SCHUIF kan gedurende een even aantal klokpulsen (maximaal 16) laag gemaakt worden.(zie par.17.4). Hierdoor zal van alle schuifregisters de ingang S_1 laag blijven en de ingang S_0 hoog worden. Dit heeft tot gevolg dat bij iedere klokpuls de inhoud van de schuifregisters een bit naar rechts geschoven wordt. De meest linkse bit wordt daarbij "1". Door het invoeren van enen voor de meest linkse bit worden deze bits voor het CAM don't cares.

Met de hierboven beschreven schakeling is het nu mogelijk om het kenmerkregister vanuit de MCS-8 te laden en de inhoud over een aantal bits te schuiven.
Bovendien wordt de informatie in het kenmerkregister aangeboden aan het CAM en zullen de leeslijnen van de woorden die associeren geaktiveerd worden. Indien een "commando"-ingang van het CAM geaktiveerd wordt zal de inhoud van het kenmerkregister in het CAM geschreven worden.

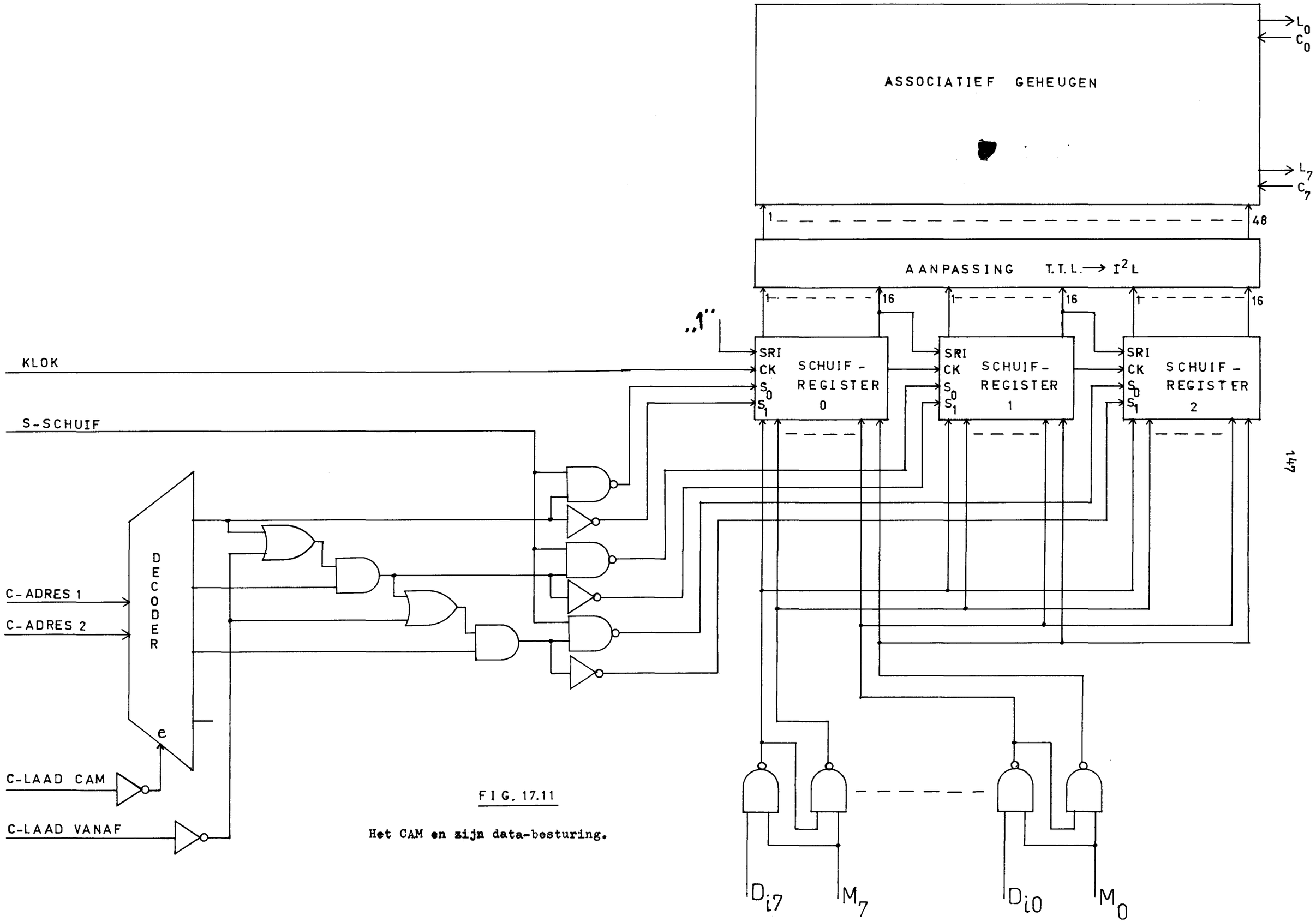


FIG. 17.11

Het CAM en zijn data-besturing.

17.7 Het RAM en zijn data-besturing.

Het RAM en zijn data-besturing moet de volgende funkties uit kunnen voeren:

a) het schrijven van het RAM op een adres dat wordt gegenereerd door de besturing van het CAM (het associatie-adres).
b) het lezen van het RAM op het associatie-adres.
c) het schrijven van het RAM op een adres dat wordt aangeboden via de drie minst significante bits van de data-bus (het extern adres).
d) het lezen van het RAM op het extern adres.
e) het per byte inlezen van het inputregister.
f) het per byte op de data-bus zetten van de data in het outputregister.

In fig.17.12 is het RAM en zijn data-besturing weergegeven.

Het RAM bestaat uit 16 woorden van 24 bit waarvan slechts de even woorden gebruikt worden. Er bestaat dus de mogelijkheid om aan ieder woord in het CAM 2 woorden in het RAM toe te voegen. Om dit te realiseren zou een adresingang van het RAM als commandolijn uitgevoerd kunnen worden.
Het RAM heeft open-collector uitgangen waardoor het noodzakelijk is om de uitgangen via een weerstand van 1 kΩ met de voedingsspanning te verbinden.
Bij lezen en schrijven van het RAM komt de data geinverteerd aan de uitgang. Hierdoor is het nodig dat de data op de data-bus geinverteerd wordt alvorens deze in de inputregisters wordt geladen.
Het RAM krijgt de stuursignalen S-ME en S-WE toegevoerd.
Indien de commandolijn C-LEES RAM respectievelijk C-SCHRIJF RAM wordt geaktiveerd dan zal de timing de stuurlijn S-ME resp. de stuurlijnen S-ME en S-WE aktiveren (zie par.17.2).
Hierdoor zal het RAM gelezen of geschreven worden.

De data-besturing voor het RAM bestaat uit een data-selector, een inputregister met bijbehorend selektiemechanisme, een outputregister en busbuffers met bijbehorend selektiemechanisme.
De data-selector verbindt het associatie-adres of het extern adres door met de adresingangen van het RAM. Dit gebeurt door C-SELECT niet respectievelijk wel te aktiveren.
Het inputregister bestaat uit inputregister 0, 1 en 2 die elk 2 registers van 4 bit bevatten.
Het selektiemechanisme, bestaande uit een 2 naar 4 decoder en een drietal OR-poorten, zorgt ervoor dat inputregister 0,1 of 2 geladen wordt. Door namelijk C-LAAD RAM te aktiveren zal, afhankelijk van C-ADRES 1 en C-ADRES 2, het kloksignaal doorgegeven worden aan de klokingang van een inputregister waardoor dit inputregister geladen zal worden.

Het outputregister heeft dezelfde configuratie als het inputregister. Het outputregister wordt geladen gedurende het lezen en schrijven van het RAM. Wanneer namelijk C-LEES RAM en C-SCHRIJF RAM geaktiveerd worden dan zal ook S-OUTPUT REG aktief worden (zie par.17.2).
Daar de data in het outputregister via een 8 bit brede databus naar de MCS-8 getransporteerd moet worden zijn er busbuffers en een selektiemechanisme nodig. De busbuffers hebben tri-state uitgangen. De data op de ingang verschijnt op de uitgang zodra de enable-ingang hoog wordt. Is de enable-ingang laag dan is de uitgang hoogohmig.
We onderscheiden de busbuffers 0, 1 en 2.
Het selektiemechanisme, bestaande uit een 2 naar 4 decoder en een drietal inverters, zorgt ervoor dat de busbuffers 0,1 of 2 enabled worden. De overeenkomstige byte in het outputregister wordt dan op de data-bus gezet. Door namelijk C-BUS-BUFFER RAM te aktiveren zal, afhankelijk van C-ADRES 1 en C-ADRES 2, de enable-ingang van de busbuffers 0,1 of 2 hoog worden.

Met de hierboven beschreven schakeling is het nu mogelijk om het RAM te lezen en te schrijven hetzij op een associatieadres hetzij op een extern adres, het inputregister per byte te laden en de data in het outputregister per byte op de databus te zetten.

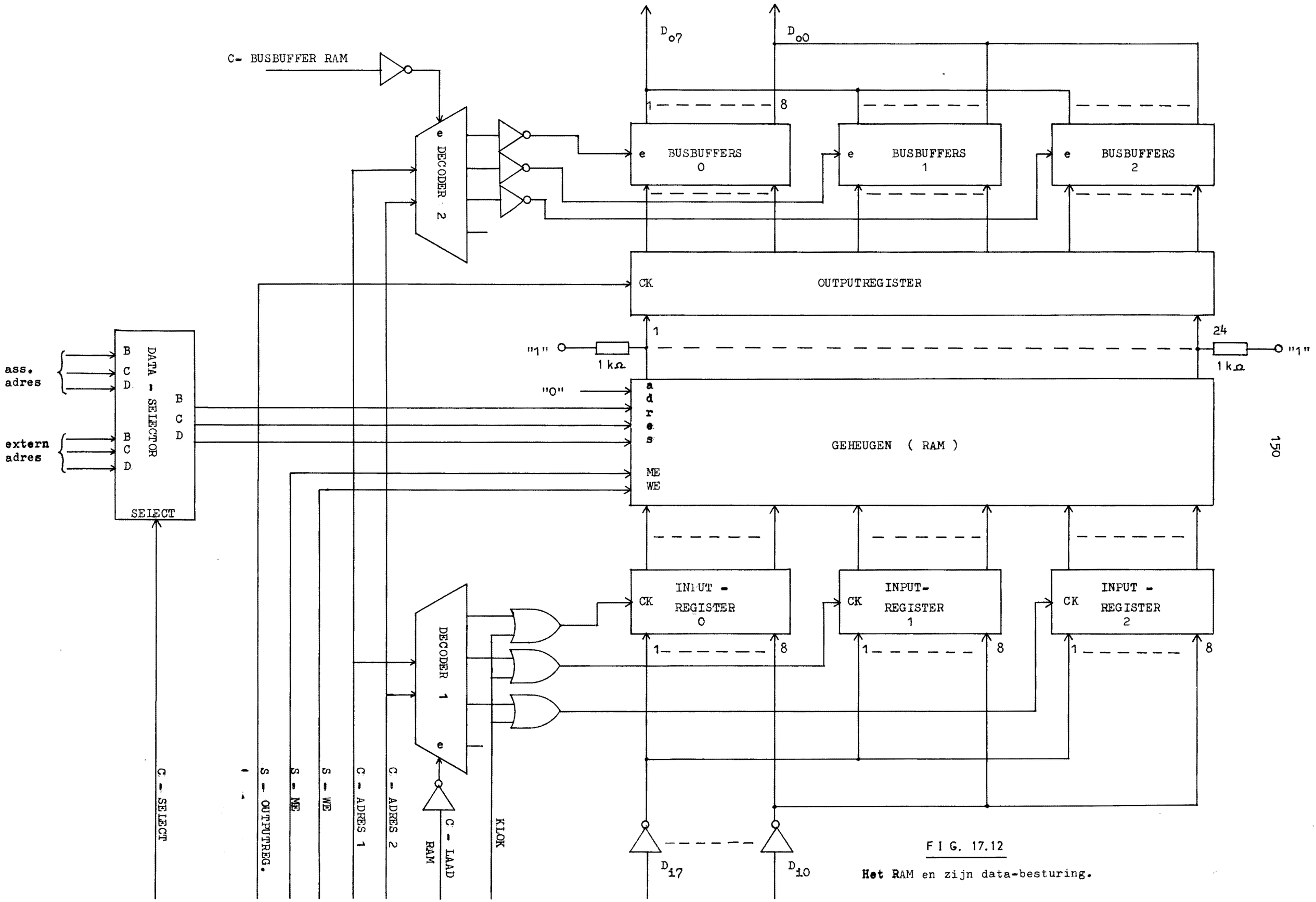


FIG. 17.12

Het RAM en zijn data-besturing.

18 BASISROUTINES.

Van de instrukties die het model uit kan voeren zijn basisroutines voor de MCS-8 geschreven. Deze routines zorgen er voor dat, wanneer men een routine aanvraagt, de outputpoorten van een selektiekaart geset worden en de timing gestart wordt. Bovendien wordt na iedere instruktie getest of deze uitgevoerd is. Dit gebeurt m.b.v. het READY-signaal.
Verder zijn er nog basisroutines geschreven om data transport te kunnen plegen tussen de MCS-8 en het model.
De eerste categorie routines zijn onder te verdelen in:
a) RAM routines. Deze routines zijn alleen bestemd voor het RAM en zijn data-besturing.
b) CAM routines. Deze routines zijn alleen bestemd voor de data-besturing en de besturing van het CAM.
c) CAM en RAM routines (CRAM routines).
Deze routines zijn bestemd voor het gehele model.
De tweede categorie routines zijn onder te verdelen in:
a) Input-Output routines.
b) Test-routines.
De eerste categorie routines maakt gebruik van de 3 outputpoorten van selektiekaart 6. De tweede categorie routines maakt gebruik van 2 outputpoorten en 2 inputpoorten van selektiekaart 5.
In tabelvorm worden nu de verschillende routines weergegeven.
Tabel 18.1 geeft de eerste categorie routines weer.
De eerste kolom geeft aan onder welke naam de routine kan worden aangeroepen. De tweede kolom de informatie die dan in de outputpoorten van selektiekaart 6 staat. De derde kolom geeft een beschrijving van de routine.
Tabel 18.2 geeft de routines van de tweede categorie weer.
De eerste kolom geeft aan onder welke naam de routine kan worden aangeroepen. De tweede kolom geeft een beschrijving van de routine. Deze kolom geeft ook aan welke registers van de MCS-8 geladen moeten zijn alvorens deze routine aan te roepen, of waar het resultaat staat als men deze routine heeft aangeroepen.
Verder is er een lijst bijgevoegd die aangeeft welke lijnen er op de output- en inputpoorten van de selektiekaarten zijn aangesloten. M.b.v. deze lijst en tabellen is het mogelijk om eenvoudig na te gaan welke commandolijnen geaktiveerd worden bij het aanroepen van een routine.
In tabel 18.1 worden de tekens X en 0/1 gebruikt.
X wil zeggen dat de bit "0" of "1" mag zijn (don't care).
0/1 wil zeggen dat eerst deze bit "0" wordt gemaakt waarna op deze plaats een "1" gezet wordt.
Dit is noodzakelijk om het startsignaal te genereren.
Zodra deze bit "1" wordt zal de timing gestart worden.

Naast de in tabel 18.1 en 18.2 beschreven routines bestaat er nog een routine. Deze routine, START genaamd, maakt gebruik van enkele routines in de tabellen. De routine wordt aangeroepen wanneer het model ingeschakeld wordt.
Het zet flip-flops van de besturing van het CAM en de teller van de timing in de juiste stand zodat het model gereed is om te gaan werken.
In Appendix I is het programma van de basisroutines gegeven.

Tabel 18.1

naam	selectiekaart 6			instruktiebeschrijving
	outputpoort 11	outputpoort 12	outputpoort 13	
a) RAM routines				
LRAM0	00000001	01000000	xxxxx0/100	laad RAM inputregister 0
LRAM1	00001001	01000000	xxxxx0/100	laad RAM inputregister 1
LRAM2	00010001	01000000	xxxxx000	laad RAM inputregister 2
ORAM0	01000000	01000000	xxxxx000	enable RAM busbuffers 0
ORAM1	01001000	01000000	xxxxx000	enable RAM busbuffers 1
ORAM2	01010000	01000000	xxxxx000	enable RAM busbuffers 2
WRAM	000xx000	01100000	xxxxx0/100	schrijf RAM op associatie-adres
RRAM	000xx000	01010000	xxxxx0/100	lees RAM op associatie-adres
SRAM	000xx000	11100000	xxxxx0/100	schrijf RAM op extern adres
RARAM	000xx000	11010000	xxxxx0/100	lees RAM op extern adres
b) CAM routines				
LCAM0	00000010	01000000	xxxxx0/100	laad schuifregister 0 van het kenmerkregister
LCAM1	00001010	01000000	xxxxx0/100	laad schuifregister 1 van het kenmerkregister
LCAM2	00010010	01000000	xxxxx0/100	laad schuifregister 2 van het kenmerkregister
LCAMV0	00001110	01000000	xxxxx0/100	laad schuifregister 0,1, 2 v.h. kenmerkregister met dezelfde informatie
LCAMV1	00001110	01000000	xxxxx0/100	laad schuifregister 1 en 2 v.h. kenmerkregister met dezelfde informatie

naam	selectiekaart 6			instruktiebeschrijving
	outputpoort 11	outputpoort 12	outputpoort 13	
LCAMV2	00010110	01000000	xxxxx0/100	laad schuifregister 2 van het kenmerkregister
SHIFT	001xx000	01000000	xxxxx0/100	schuif inhoud v.h. kenmerkregister over een aantal bits (max.16) naar rechts en voer don't cares in
OCAM	100xx000	01000000	xxxxx000	zet inhoud CAO register op de data-bus
WCAM	000xx000	01000001	xxxxx0/100	schrijf CAM op de 1e lege plaats
SCAM	000xx000	01000010	xxxxx0/100	schrijf CAM op extern adres
CLCAM	000xx000	01001000	xxxxx0/100	wis CAM op alle adressen die associeren op de inhoud van het kenmerkregister
RSTCAM	000xx000	010001 00	xxxxx0/100	wis CAM op extern adres
TVPULS	000xx000	010000 00	xxxxx0/101	geef tijdelijk vrij puls d.w.z. na iedere t.v. puls komt het volgende woordt dat associeert tevoorschijn
TVRST	000xx000	01000000	xxxxx0/110	hef tijdelijk vrij pulsen op
c) CRAM routines				
LCRAM0	01000000	01000000	xxxxx0/100	laad schuifregister 0 en RAM inputregister 0
LCRAM1	00001011	01000000	xxxxx0/100	laad schuifregister 1 en RAM inputregister 1
LCRAM2	00010011	01000000	xxxxx0/100	laad schuifregister 2 en RAM inputregister 2

naam	selectiekaart 6			instruktiebeschrijving
	outputpoort 11	outputpoort 12	outputpoort 13	
WCRAM	000xx000	01100000	xxxxx0/100	schrijf CAM op 1e lege plaats en schrijf RAM op bijbehorend adres
SCRAM	000xx000	11100010	xxxxx0/100	schrijf CAM en RAM op extern adres

Tabel 18.2

a) Input-Output routines

naam	funktiebeschrijving
DMOUT	zet data op de data-bus en de masker-bus. reg A moet eerst met de data die op de data-bus moet komen geladen worden reg B moet eerst met de data die op de masker-bus moet komen geladen worden
DATAIN	breng data van de data-bus in de MCS 8 reg B bevat daarna de data

b) Test-routines

naam	funktiebeschrijving
CAMVOL	test of het CAM volgeschreven is. Zo ja, dan wordt carry flip-flop geset.
ASS.	test of het CAM associeert op de inhoud van de kenmerkregisters. Zo ja, dan wordt carry flip-flop geset.

Lijst met aansluitingen van de selectiekaarten

selectiekaart 6

bit	outputpoort 11B	outputpoort 12B	outputpoort 13B
7	enable CAO-buffer	selekt adres RAM	n.c.
6	enable output-buffer RAM	laad CAO	n.c.
5	schuif	schrijf RAM	n.c.
4	adres 2	lees RAM	n.c.
3	adres 1	wis CAM	n.c.
2	Ø of 0	reset CAM	start
1	laad CAM register	set CAM	t.v.-reset
0	laad RAM register	schrijf CAM	t.v.

inputpoorten zijn niet in gebruik

selectiekaart 5

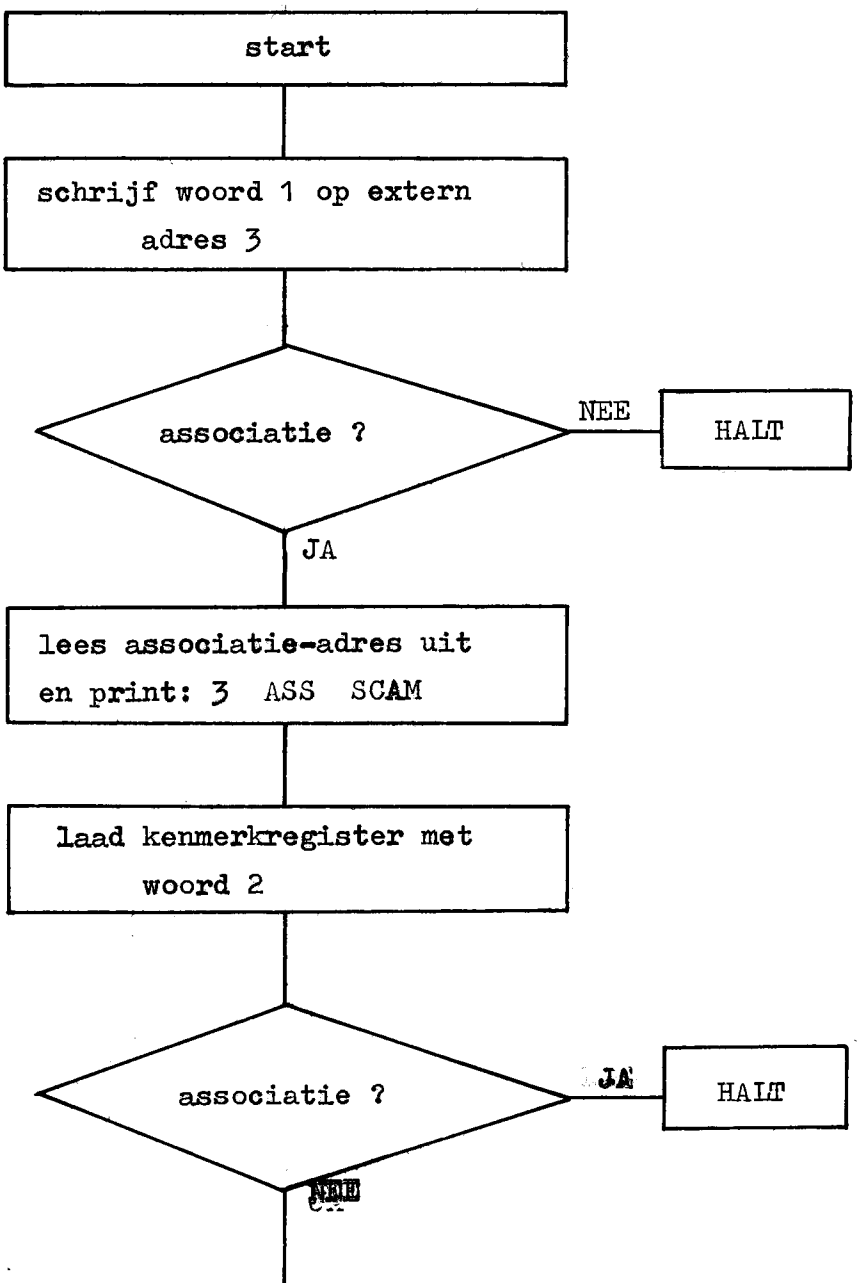
bit	outputpoort 11B	outputpoort 12B	inputpoort 0B	inputpoort 1B
7	data MSB	masker MSB	data MSB	n.c.
6	"	"	"	n.c.
5	"	"	"	n.c.
4	"	"	"	n.c.
3	"	"	"	n.c.
2	"	"	"	associatielijn
1	"	"	"	vol
0	data LSB	masker LSB	data LSB	ready

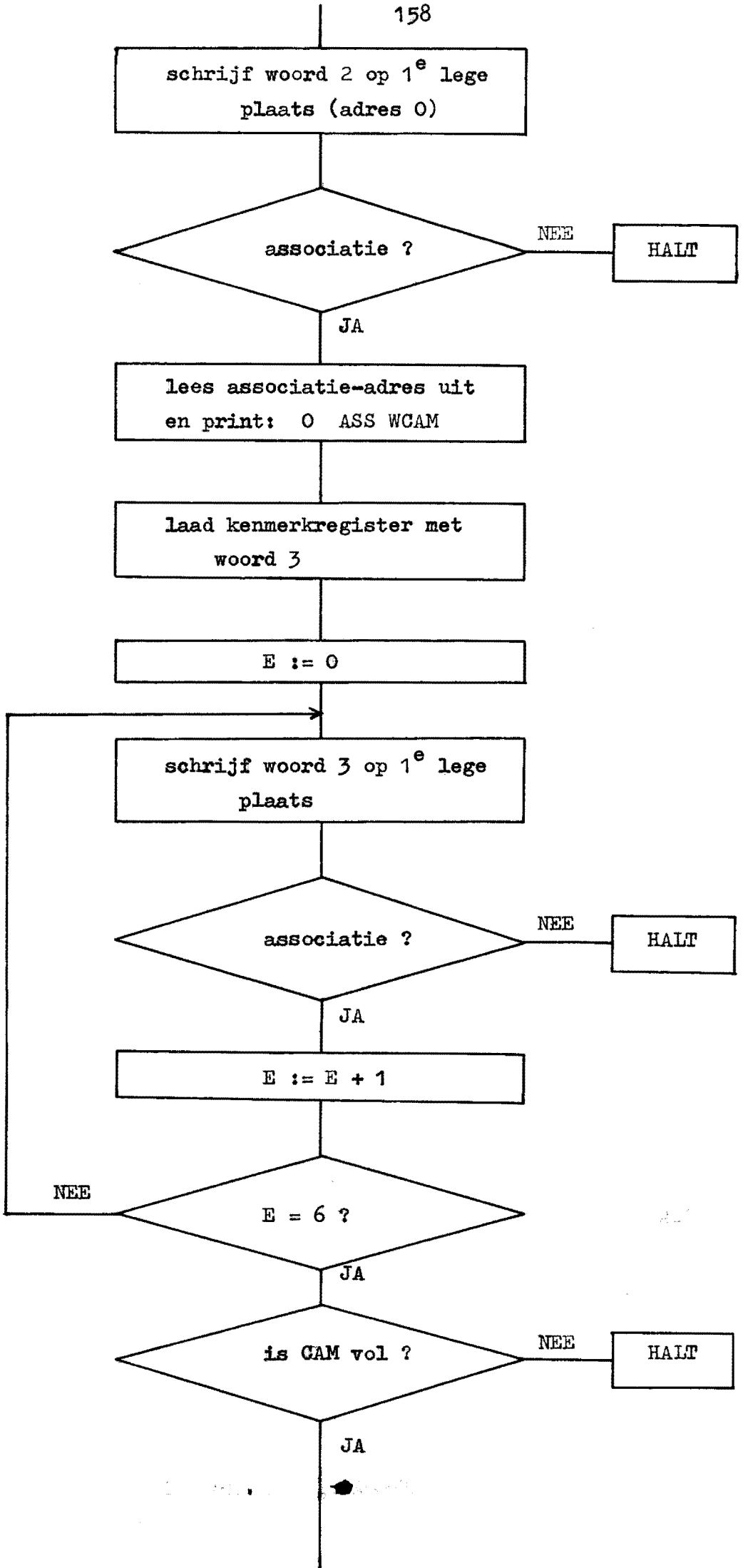
outputpoort 13B is niet in gebruik

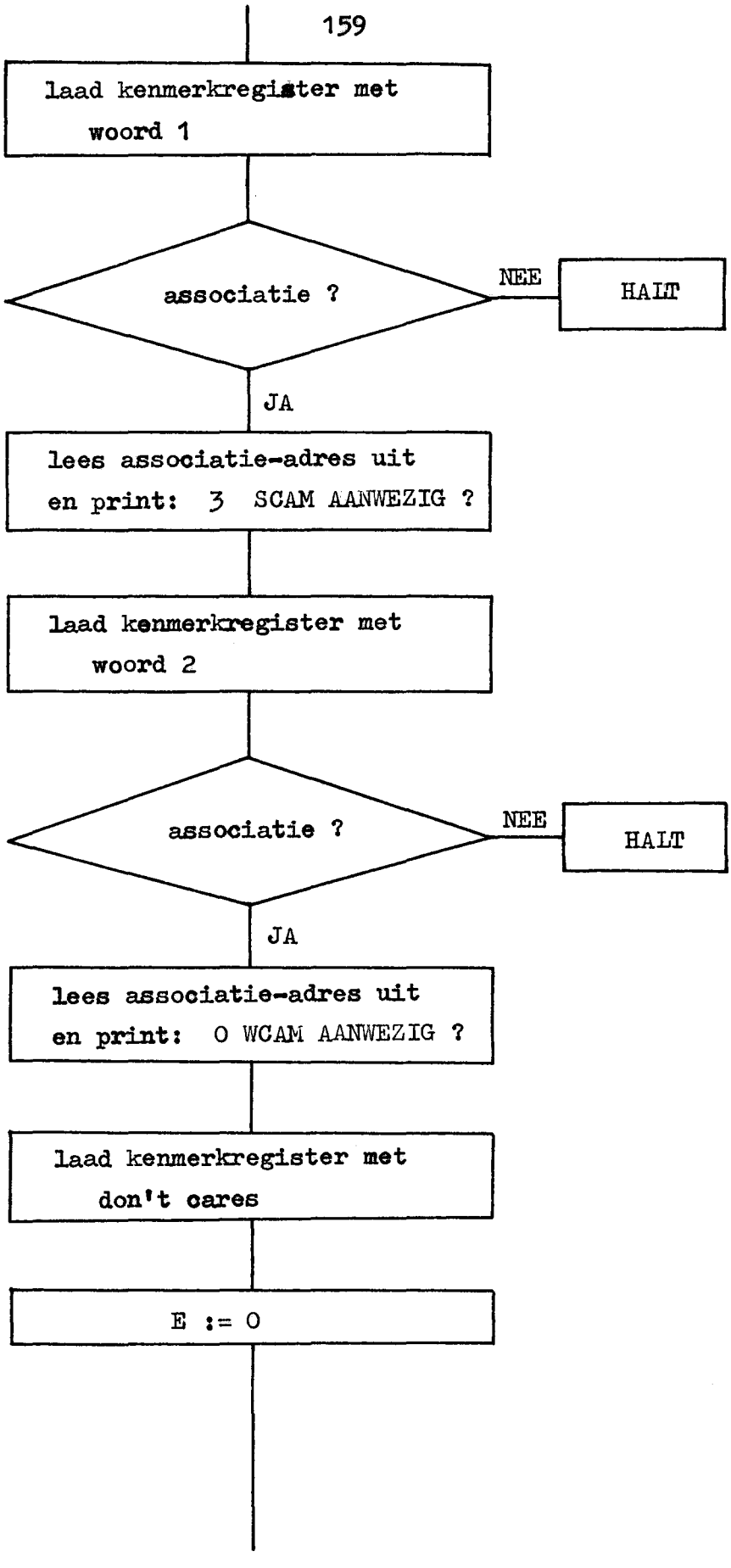
19 FLOW-DIAGRAMMEN VOOR HET TESTEN VAN HET MODEL.

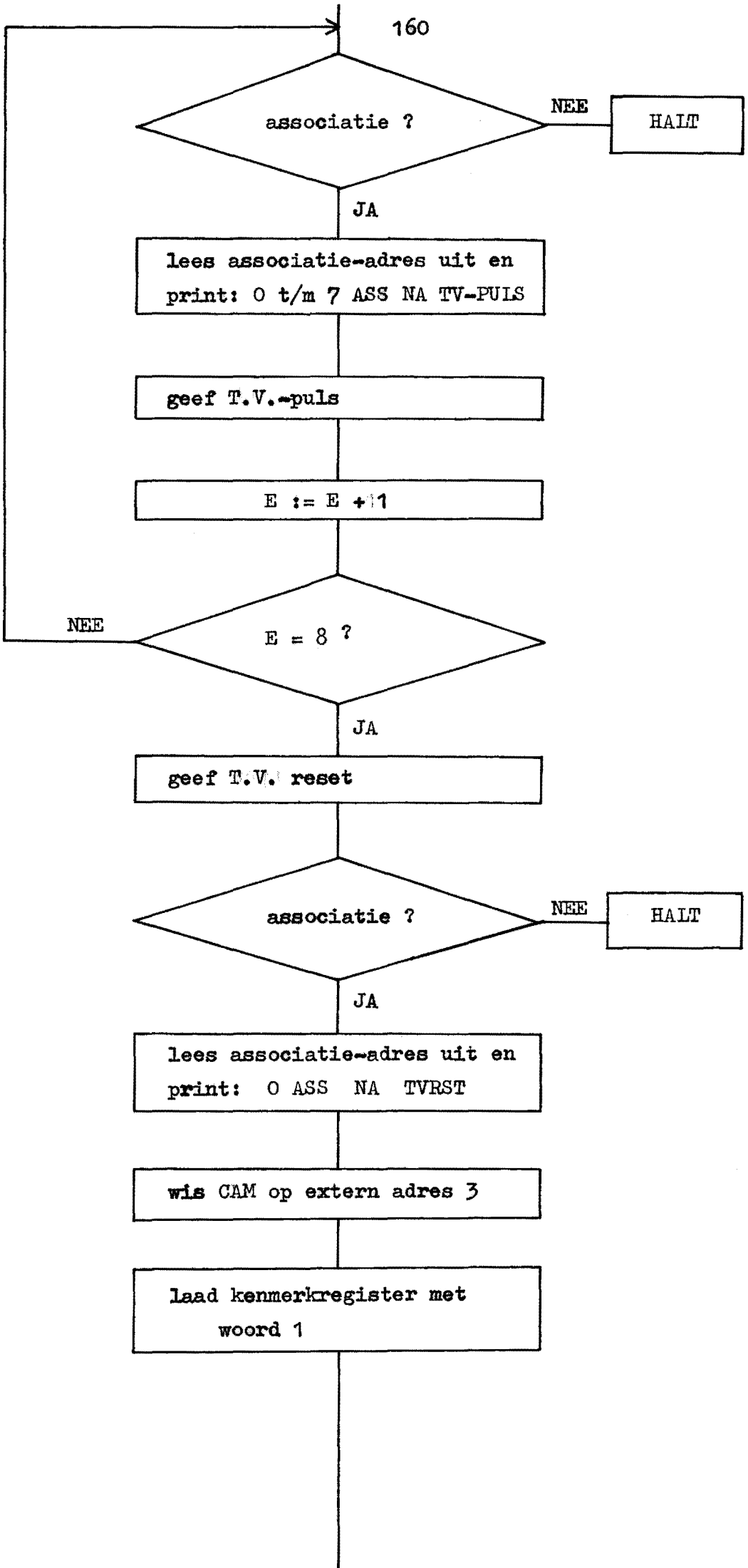
In dit hoofdstuk zijn de flow-diagrammen weergegeven volgens welke het model getest is.
Het eerste flow-diagram geeft aan hoe de data-besturing en de besturing van het CAM en het CAM zelf getest wordt met inbegrip van de timing.
Het tweede flow-diagram doet hetzelfde voor het RAM en de data-besturing van het RAM. Tevens geeft dit flow-diagram weer hoe het testen verloopt wanneer het CAM en het RAM samenwerken.
Aan de hand van deze flow-diagrammen is m.b.v. de basisroutines een testprogramma geschreven.
In Appendix II is het testprogramma gegeven.

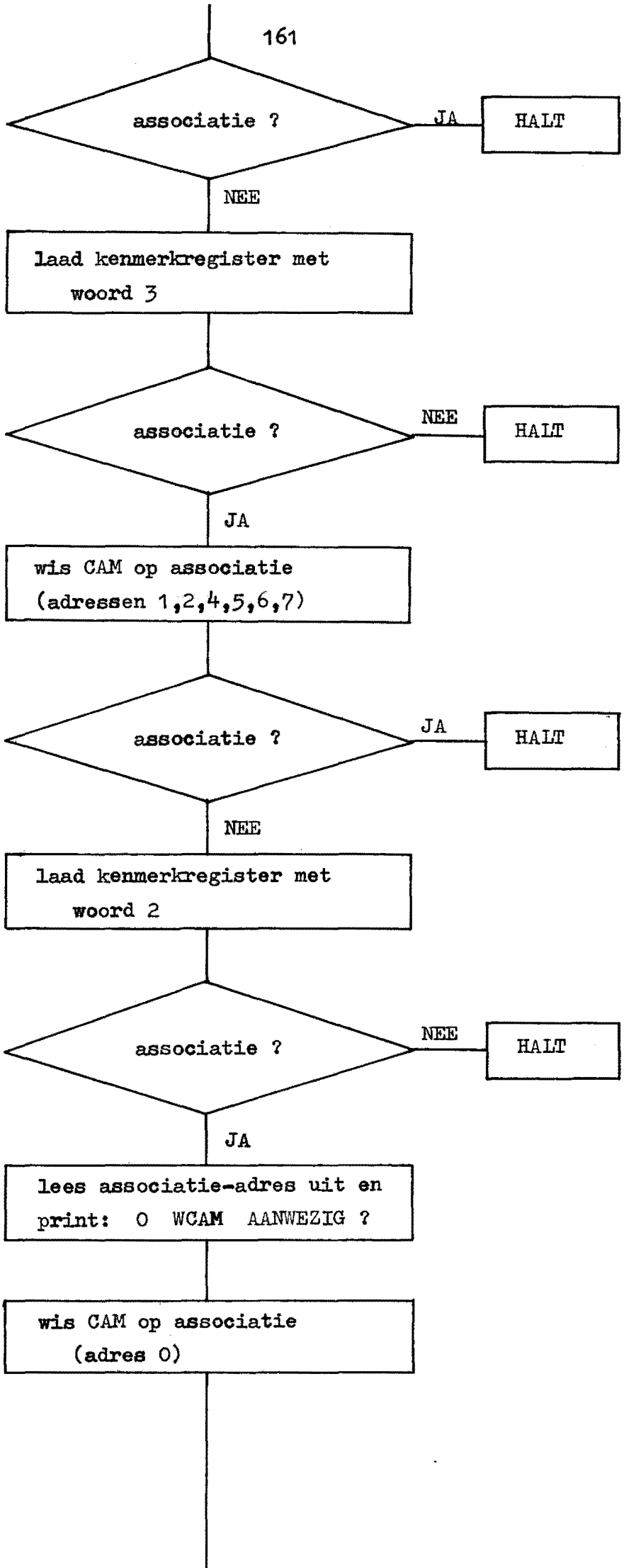
Flow-diagram I

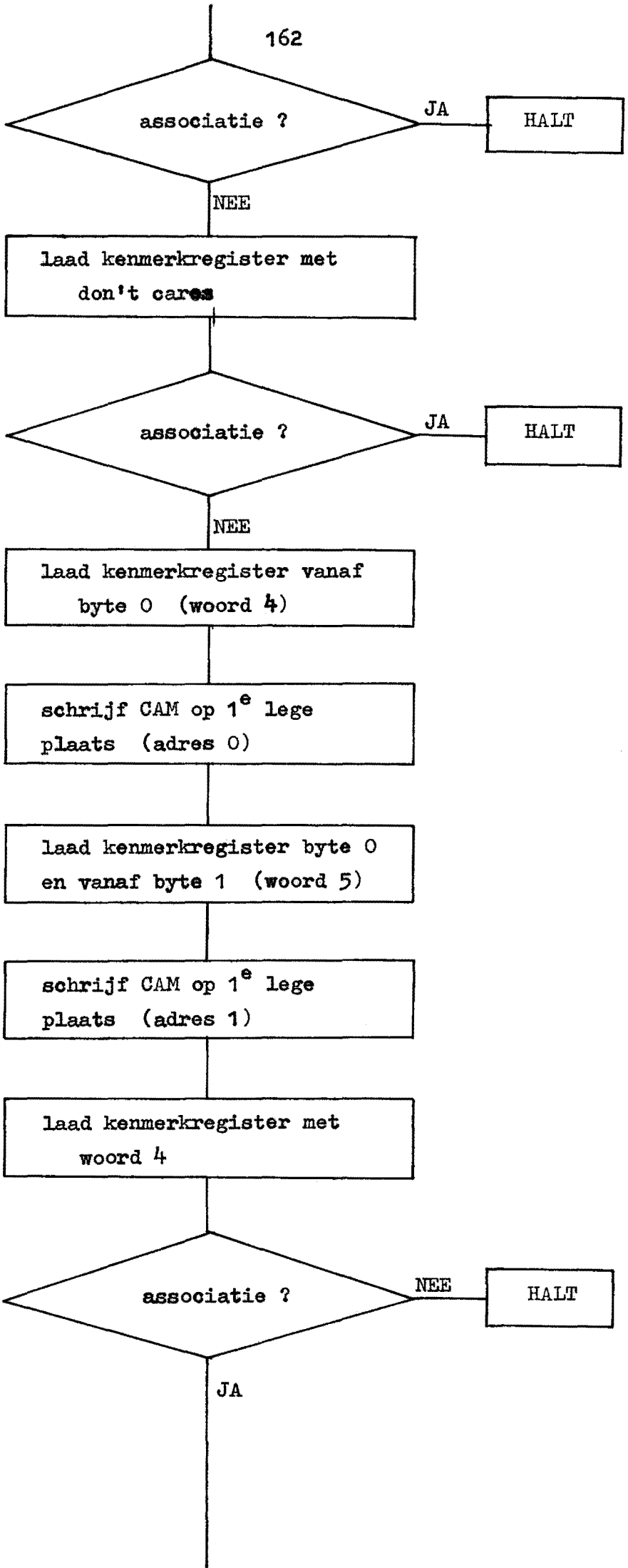


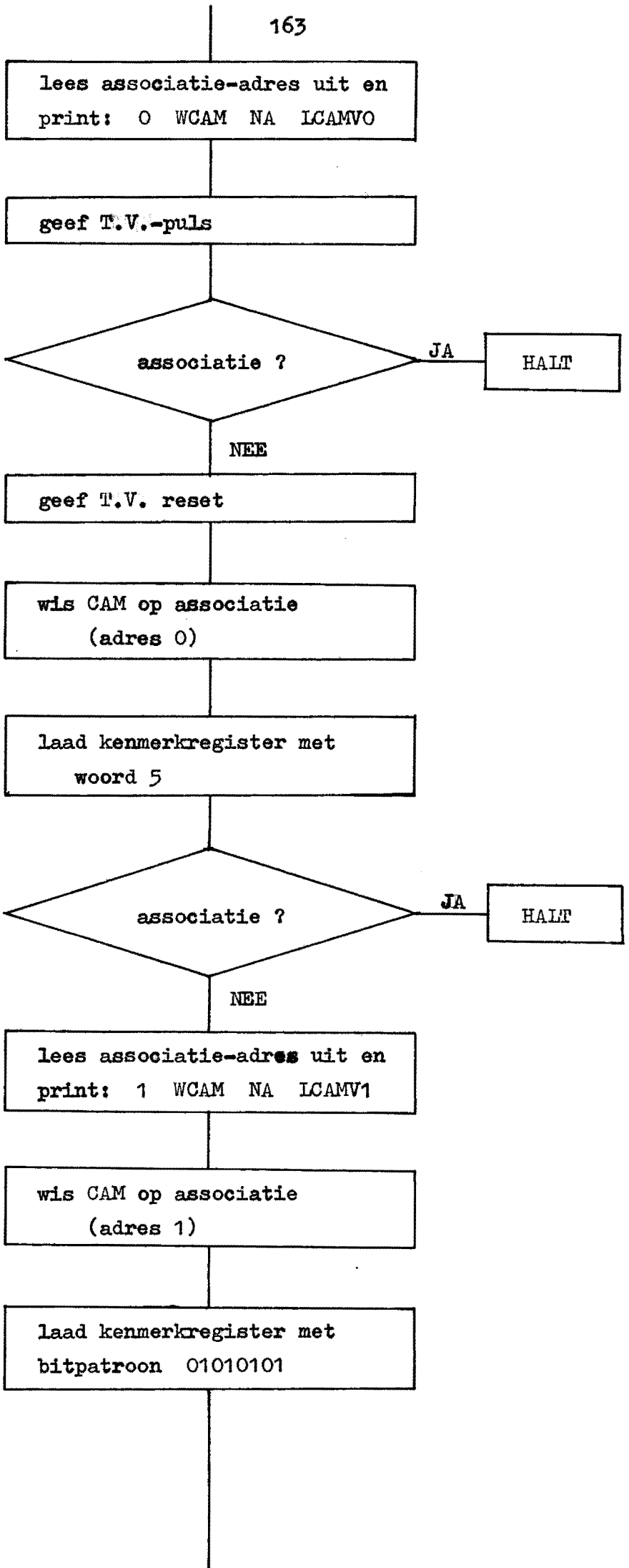

schrijf woord 2 op 1e lege plaats (adres 0)
associatie ?
NEE
HALT
JA
lees associatie-adres uit en print: 0 ASS WCAM
laad kenmerkregister met woord 3
E := 0
schrijf woord 3 op 1e lege plaats
associatie ?
NEE
HALT
JA
E := E + 1
NEE
E = 6 ?
JA
is CAM vol ?
NEE
HALT
JA

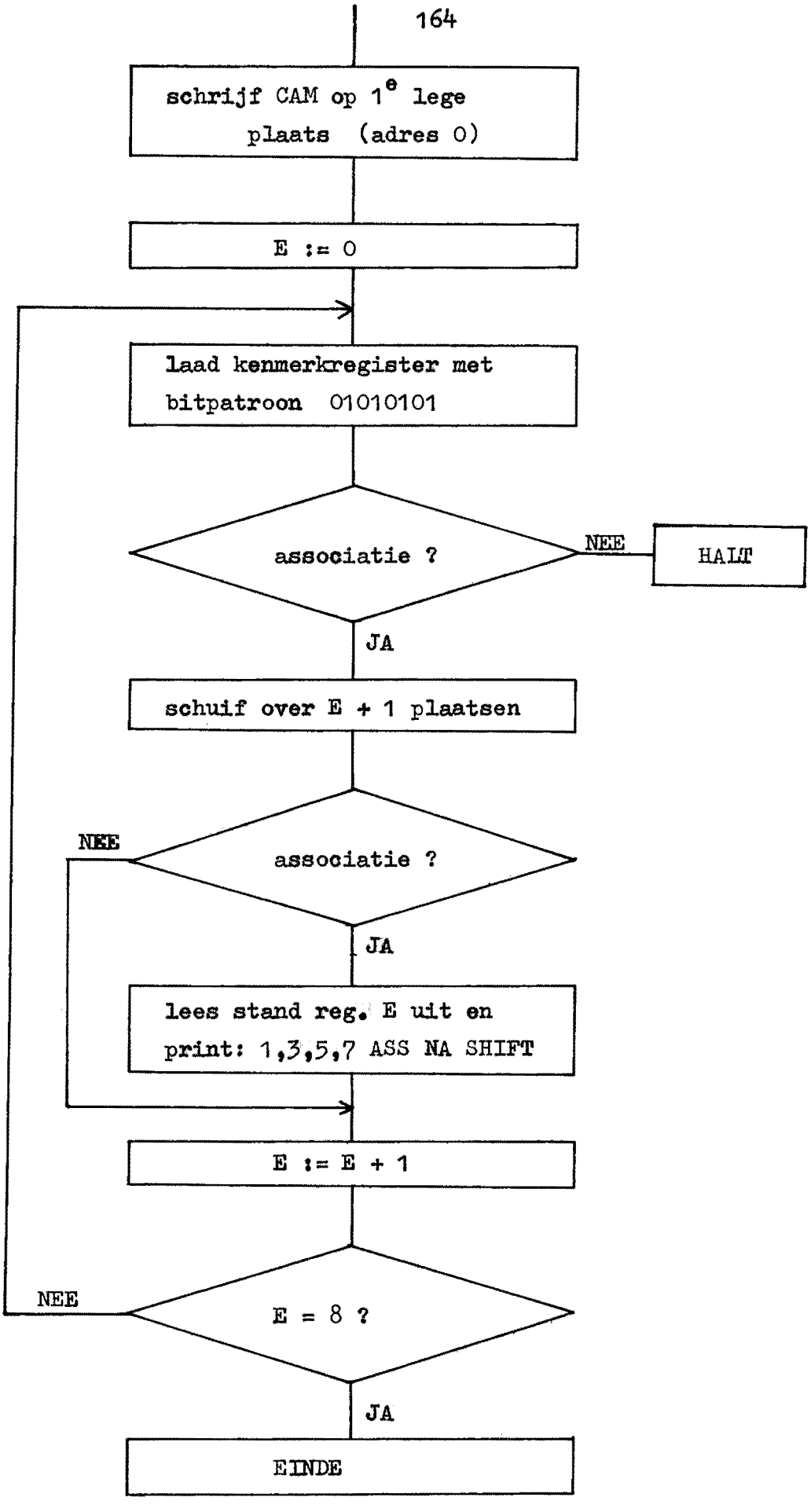

laad kenmerkregister met
woord 1
associatie ?
NEE
HALT
JA
lees associatie-adres uit
en print: 3 SCAM AANWEZIG ?
laad kenmerkregister met
woord 2
associatie ?
NEE
HALT
JA
lees associatie-adres uit
en print: 0 WCAM AANWEZIG ?
laad kenmerkregister met
don't cares
E := 0


160
associatie ?
NEE
HALT
JA
lees associatie-adres uit en
print: 0 t/m 7 ASS NA TV-PULS
geef T.V.-puls
E := E + 1
NEE
E = 8 ?
JA
geef T.V. reset
associatie ?
NEE
HALT
JA
lees associatie-adres uit en
print: 0 ASS NA TVRST
wis CAM op extern adres 3
laad kenmerkregister met
woord 1

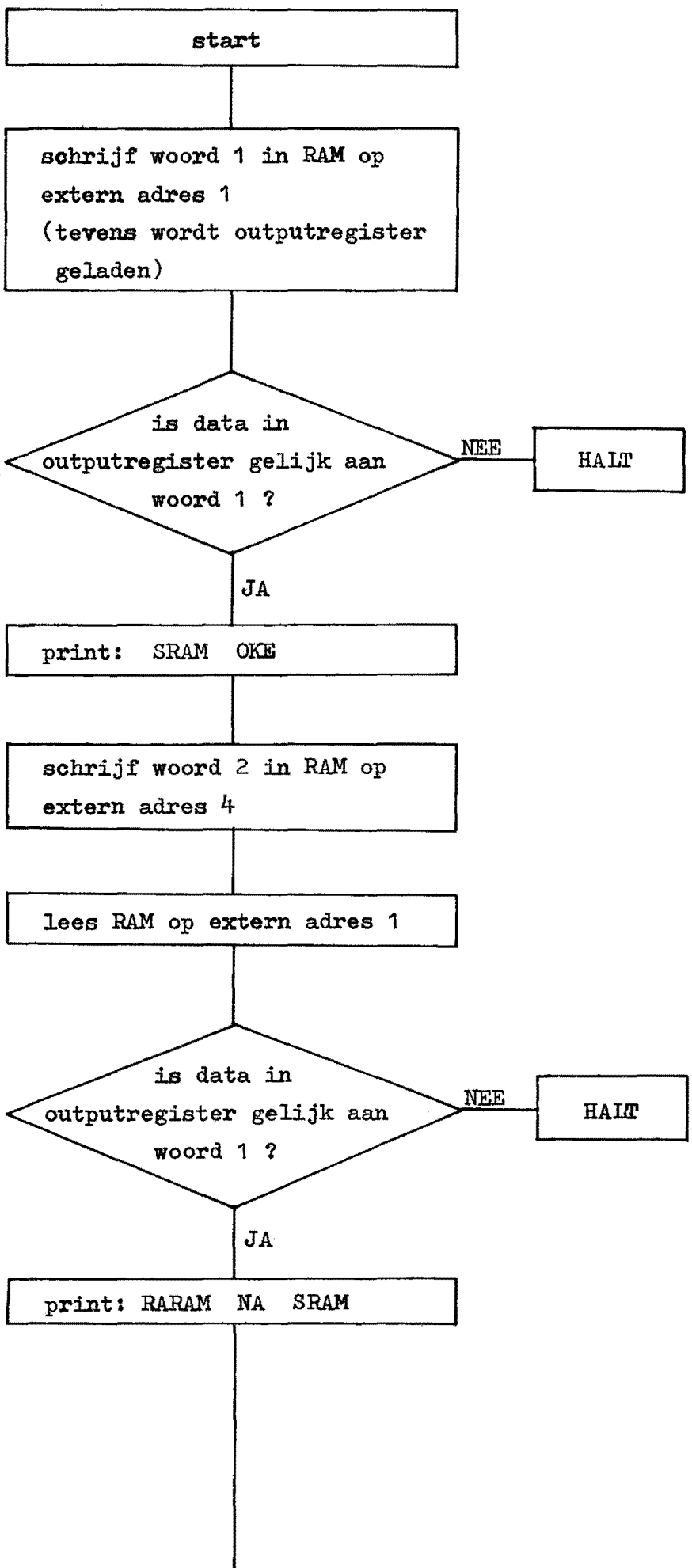

associatie ?
JA
HALT
NEE
laad kenmerkregister met
woord 3
associatie ?
NEE
HALT
JA
wis CAM op associatie
(adressen 1,2,4,5,6,7)
associatie ?
JA
HALT
NEE
laad kenmerkregister met
woord 2
associatie ?
NEE
HALT
JA
lees associatie-adres uit en
print: 0 WCAM AANWEZIG ?
wis CAM op associatie
(adres 0)

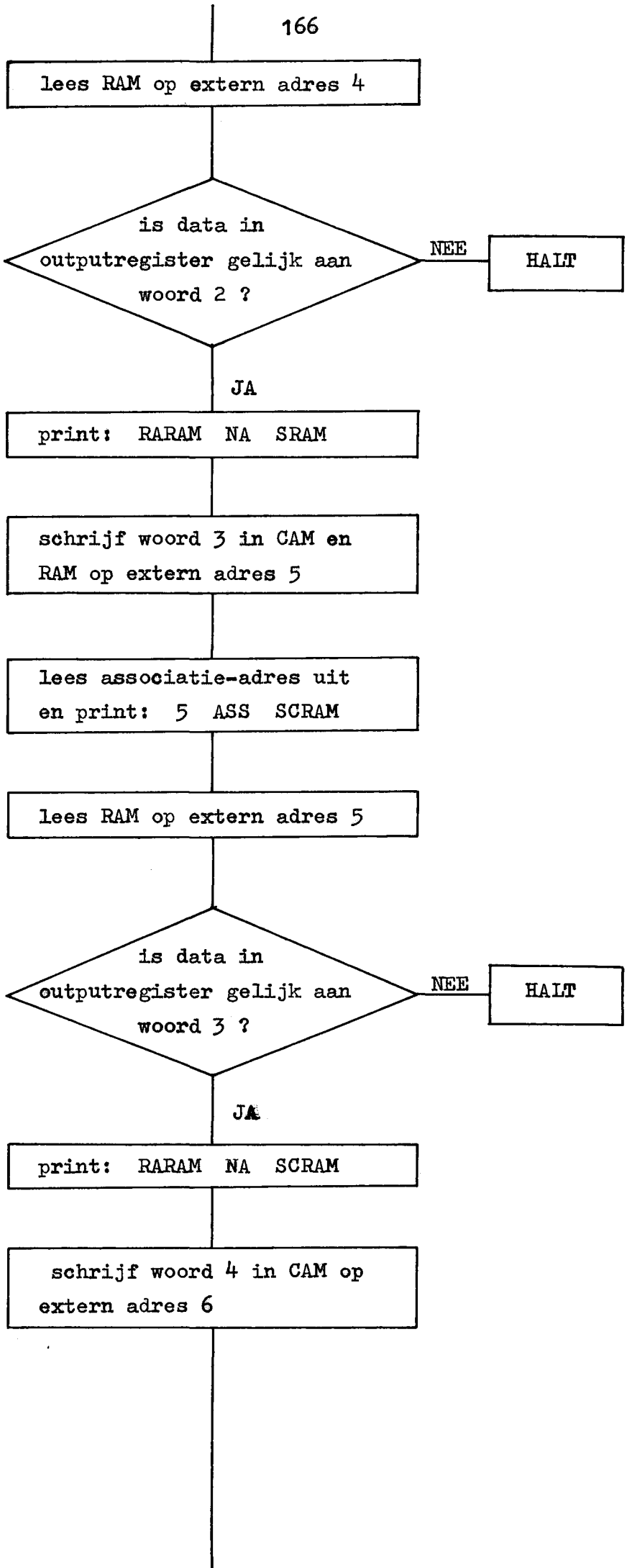

associatie ?
JA
HALT
NEE
laad kenmerkregister met don't cares
associatie ?
JA
HALT
NEE
laad kenmerkregister vanaf byte 0 (woord 4)
schrijf CAM op 1e lege plaats (adres 0)
laad kenmerkregister byte 0 en vanaf byte 1 (woord 5)
schrijf CAM op 1e lege plaats (adres 1)
laad kenmerkregister met woord 4
associatie ?
NEE
HALT
JA

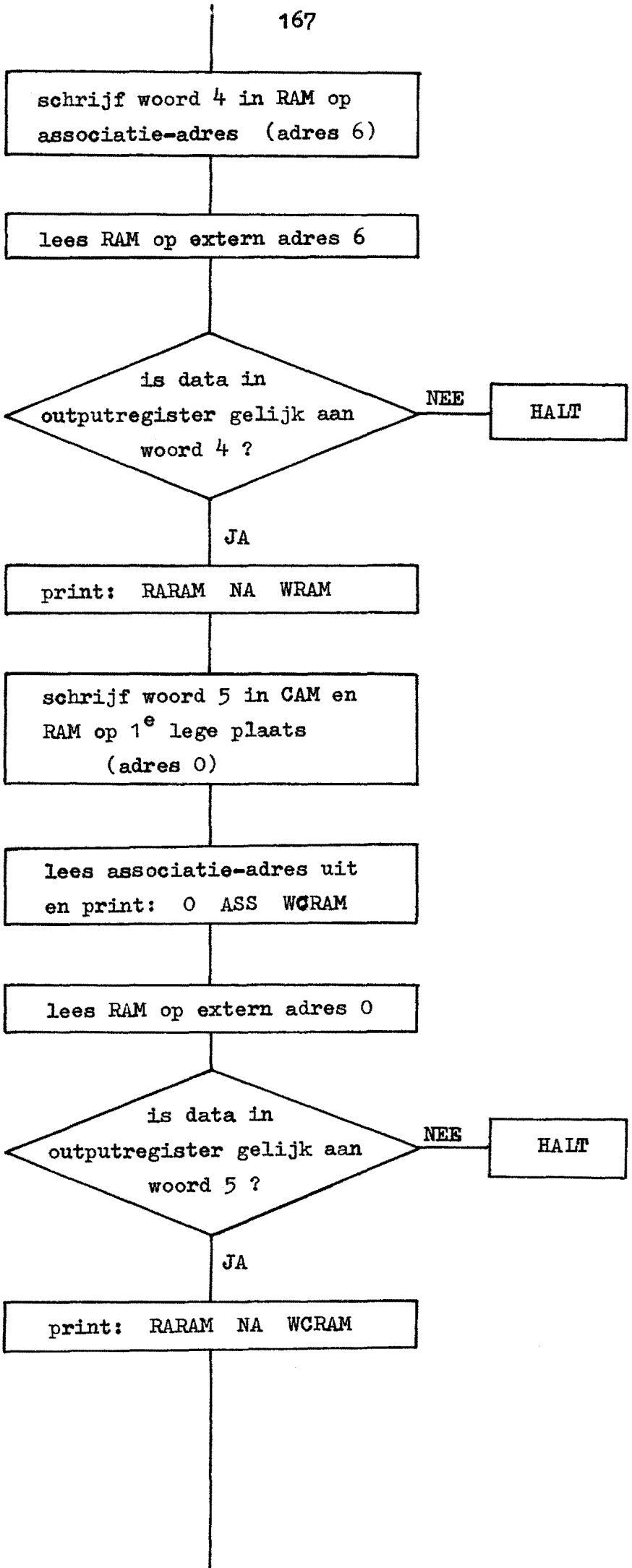

lees associatie-adres uit en
print: 0 WCAM NA LCAMVO
geef T.V.-puls
associatie ?
JA
HALT
NEE
geef T.V. reset
wis CAM op associatie
(adres 0)
laad kenmerkregister met
woord 5
associatie ?
JA
HALT
NEE
lees associatie-adres uit en
print: 1 WCAM NA LCAMV1
wis CAM op associatie
(adres 1)
laad kenmerkregister met
bitpatroon 01010101

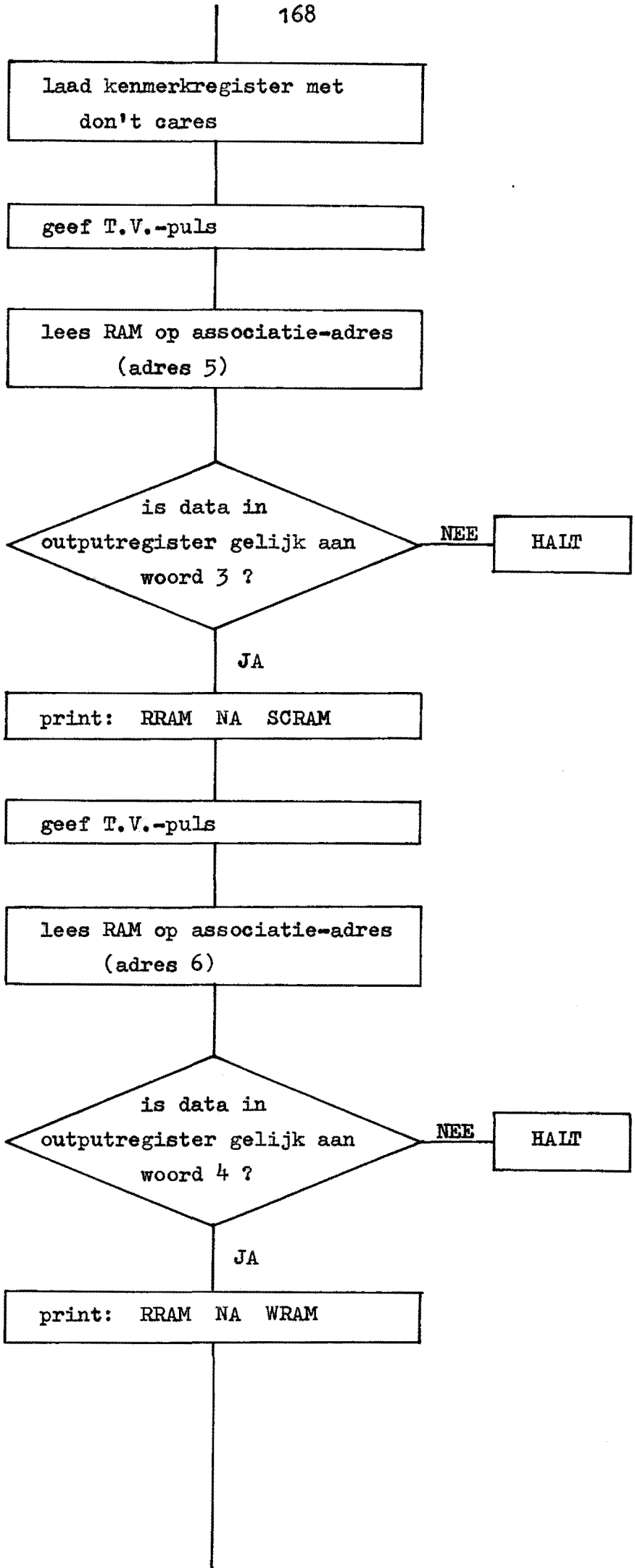

schrijf CAM op 1e lege plaats (adres 0)
E := 0
laad kenmerkregister met bitpatroon 01010101
associatie ?
NEE
HALT
JA
schuif over E + 1 plaatsen
NEE
associatie ?
JA
lees stand reg. E uit en print: 1,3,5,7 ASS NA SHIFT
E := E + 1
NEE
E = 8 ?
JA
EINDE

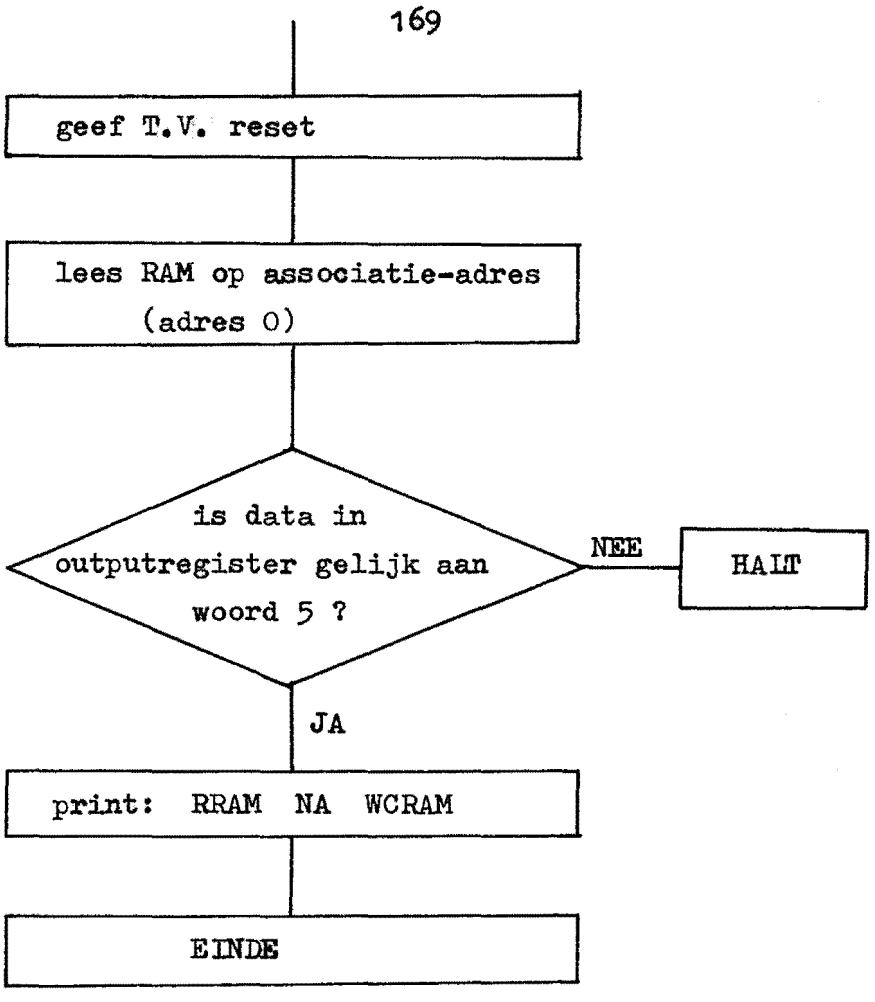
Flow-diagram II




lees RAM op extern adres 4
is data in outputregister gelijk aan woord 2 ?
NEE
HALT
JA
print: RARAM NA SRAM
schrijf woord 3 in CAM en RAM op extern adres 5
lees associatie-adres uit en print: 5 ASS SCRAM
lees RAM op extern adres 5
is data in outputregister gelijk aan woord 3 ?
NEE
HALT
JA
print: RARAM NA SCRAM
schrijf woord 4 in CAM op extern adres 6


schrijf woord 4 in RAM op associatie-adres (adres 6)
lees RAM op extern adres 6
is data in outputregister gelijk aan woord 4 ?
NEE
HALT
JA
print: RARAM NA WRAM
schrijf woord 5 in CAM en RAM op 1e lege plaats (adres 0)
lees associatie-adres uit en print: 0 ASS WCRAM
lees RAM op extern adres 0
is data in outputregister gelijk aan woord 5 ?
NEE
HALT
JA
print: RARAM NA WCRAM


laad kenmerkregister met don't cares
geef T.V.-puls
lees RAM op associatie-adres (adres 5)
is data in outputregister gelijk aan woord 3 ?
NEE
HALT
JA
print: RRAM NA SCRAM
geef T.V.-puls
lees RAM op associatie-adres (adres 6)
is data in outputregister gelijk aan woord 4 ?
NEE
HALT
JA
print: RRAM NA WRAM


geef T.V. reset
lees RAM op associatie-adres
(adres 0)
is data in
outputregister gelijk aan
woord 5 ?
NEE
HALT
JA
print: RRAM NA WCRAM
EINDE

LITERATUUR

LITERATUUR

De hierna volgende artikelen zijn onderverdeeld in een aantal groepen, die met hoofdletters zijn aangeduid.
De artikelen zijn per groep geordend naar het jaar van publikatie.
Ze hebben niet allen betrekking op associatieve geheugens of toepassingen daarvan, echter wel op processen, waarbij het toepassen van een associatief geheugen zinvol is.

		Van	t/m
A	Het ontworpen associatieve geheugen	1	2
B	Artikelen, die een algemeen overzicht geven	3	9
C	Diverse facetten van associatieve geheugens	10	89
D	Associatieve talen, software	90	94
E	Sorteren	95	108
F	Symboolmanipulaties	109	114
G	Diff vergelijkingen, fouriertransformatie	115	116
H	Matrixbewerkingen	117	127
I	Grafentheorie	128	130
J	Dynamische geheugentoewijzing	131	144
K	Stack-adressering	145	147
L	Een associatief werkgeheugen	148	
M	Time-sharing	149	151
N	Patroonherkenning	152	156
O	Boeken	157	160

LITERATUUR

1 Het realiseren van een associatief geheugen m.b.v. Injectie-logica
R G J Dijkstra
Afstudeerverslag, groep ECB,
Technische Hogeschool Eindhoven, okt 1974

2 Model van een associatief geheugen
J M v Loon, Th P C Stoffele
Afstudeerverslag, groep EB,
Technische Hogeschool Eindhoven, juni 1975
(deel III van dit verslag)

3 Assoziatieve Speicherung
E Jessen
Elektronische Datenverarbeitung
Beiheft 5, 1965, pag 1-43

4 Der assoziatieve Speicher - Anwendung und Realisierungsmoeglichkeiten
F F Tsui
Frequenz, Maerz 1966, Bd 20, nr 3, pag 69

5 Content-adressable and associative memory systems
A G Hanlon
IEEE Trans on El Comp, vol EC-15
no 4, aug 1966, pag 509

6 Assoziative Speicher - eine Uebersicht
H O Leilich
Institut fuer Datenverarbeitungsanlagen
T.U. Braunschweig, 18-03-1971

7 An overview of associative or content addressable memory systems and a KWIC Index to the literature
J Minker
University of Maryland Computer Science Center
Technical Report TR-157, june 1971, 140 pp

8 Associative memories and processors: A description and appraisal
J Minker
University of Maryland and Auerbach Corp.
Technical Report TR-195 (july 1972), 70 pp

9 Associative memories and processors: An overview and selected bibliography
B Parhami
Proc of the IEEE, vol 61, no 6, june 1973, pag 722

10 The cryotron catalog memory system
A E Slade, H O McMahon
Proc Eastern Joint Comp Conference
dec 10-12, 1956, New York, vol 18, pag 115

11 A superconductive associative memory
Paul M Davies
Proc Spring Joint Comp Conf (AFIPS)
vol 21, 1962, pag 79

12 A cryogenic data-addressed memory
V L Newhouse, R E Fruin
Proc Spring Joint Comp Conf (AFIPS)
vol 21, 1962, pag 89

13 The SOLOMON Computer
D L Slotnick, W C Borck, R C Reynolds
Proc Fall Joint Comp Conf (AFIPS)
vol 22, 1962, pag 97

14 Intercommunicating cells, basis for a distibuted logic computer
C Y Lee
Proc Fall Joint Comp Conf (AFIPS)
vol 22, 1962, pag 130

15 On the use of the SOLOMON parallel-processing computer
R C Bollinger, T A Jeeves, R C McReynolds, D H Shaffer
Proc Fall Joint Comp Conf (AFIPS)
vol 22, 1962, pag 137

16 A content adressable distributed logic memory with applications in information retrieval
C Y Lee, M C Paul
Proc of the IEEE, june 1963, pag 924

17 Proposal for ordered sequential detection of simultaneous multiple responses
H Weinstein
IEEE Trans on El Comp
vol EC-12, oct 1963, pag 564

18 Some applications for content-addressable memories
G Estrin, R H Fuller
Proc Fall Joint Comp Conf (AFIPS)
vol 24, nov 1963, pag 495

19 Associative techniques with complementary flip-flops
E S Lee
Proc Spring Joint Comp Conf (AFIPS)
vol 23, 1963, pag 381

20 A 300 nanosecond search memory
C A Rowland, W O Berge
Proc Fall Joint Comp Conf (AFIPS),
vol 24, nov 1963, pag 59

21 A search memory subsystem for a general purpose computer
A Kaplan
Proc Fall Joint Comp Conf (AFIPS),
vol 24, nov 1963, pag 193

22 Associative logic for highly parallel systems
R R Seeber, A B Lindquist
Proc Fall Joint Comp Conf (AFIPS),
vol 24, nov 1963, pag 489

23 Application of an associatively addressed
distributed memory
G J Simmons
Proc Spring Joint Comp Conf (AFIPS)
vol 25, 1964, pag 493

24 Design of an experimental multiple instantaneous
response file
E L Younker, C H Heckler, D P Masher,
J M Yarborough
Proc Spring Joint Comp Conf (AFIPS)
vol 25, 1964, pag 515

25 Associative memory system, implementation
and characteristics
J E McAteer, J A Capobianco
Proc Fall Joint Comp Conf (AFIPS)
vol 26, 1964, pag 81

26 An associative processor
R G Ewing, P M Davis
Proc Fall Joint Comp Conf (AFIPS)
vol 26, 1964, pag 147

27 A hardware integrated general purpose computer/search memory
F G Gall
Proc Spring Joint Comp Conf (AFIPS)
vol 26, 1964, pag 159

28 An improved cell memory
R S Gaines, C Y Lee
IEEE Trans on El Comp
vol EC-14, no 2, feb 1965, pag 72

29 Content addressable memories
R R Seeber
Proc of IFIP Congress
vol 2, may 1965, pag 479

30 A destructive-readout associative memory
Y Chu
IEEE Trans on El Comp
vol EC-14, no 8, aug 1965, pag 600

31 Content addressable memory system concepts
A V Campi, R M Dunn, B H Gray
IEEE Trans on Aerospace Electron Systems
vol AES-1, oct 1965, pag 168

32 The associative memory structure
B T McKeever
Proc Fall Joint Comp Conf (AFIPS)
vol 27, part 1, 1965, pag 371

33 A cutpoint cellular associative memory
C C Yang, S S Yau
IEEE Trans on El Comp
vol EC-15, no 8, aug 1966, pag 522

34 A nonbulk addition technique for associative processors
S S Yau
IEEE Trans on El Comp
vol EC-15, no 12, dec 1966, pag 938

35 Highly parallel information processing systems
J C Murtha
In: Advances in Computers. Editors: Alt, Rubinoff
vol 7, 1966, Academic Press, New York

36 A study of the utility of associative memory processors
J A Dugan, R S Green, J Minker, W S Shindle
Proc of the 21st National Conference
of the ACM, 1966, pag 347

37 Associative parallel processing
R H Fuller
Proc Spring Joint Comp Conf (AFIPS)
vol 30, 1967, pag 471

38 An integrated MOS transistor associative memory
system with 100 ns cycle time
R Igarashi, T Yaita
Proc Spring Joint Comp Conf (AFIPS)
vol 30, 1967, pag 499

39 Determination of priority in associative memories
C C Foster
IEEE Trans on Comp
vol C-17, no 7, july 1968, pag 788

40 ASP - a ring implemented associative structure package
C A Lang, J C Gray
Comm of the ACM
vol 11, no 8, aug 1968, pag 550

41 Associative memories in large computer systems
D Aspinall, D J Kinniment, D B G Edwards
Information Processing 1968, pag 796

42 An integrated associative memory matrix
D Aspinall, E J Kinniment, D B G Edwards
Information Processing 1968, pag 800a

43 A magnetic associative memory
Tse-Yun Feng
Proc Spring Joint Comp Conf (AFIPS)
vol 32, 1968, pag 275

44 Associative processing for general purpose computers through the use of modified memories
H S Stone
Proc Fall Joint Comp Conf (AFIPS)
vol 33, part 2, 1968, pag 949

45 A multiaccess associative memory
N K Natarajan, Paul A V Thomas
IEEE Trans on Comp
vol C-18, no 5, may 1969, pag 424

46 Cellular logic-in-memory arrays
William H Kautz
IEEE Trans on Comp
vol C-18, no 8, aug 1969, pag 719

47 Unified interval classification and unified 3-classification for associative memories
A Wolinsky
IEEE Trans on Computers
vol C-18, oct 1969, pag 899

48 Study of a "look-aside" memory
Francis F Lee
IEEE Trans on Comp
vol C-18, no 11, nov 1969, pag 1062

49 Silicon-on-sapphire complementary MOS circuits for high speed associative memory
J R Burns, J H Scott
Proc Fall Joint Comp Conf (AFIPS)
vol 35, 1969, pag 469

50 System design of a cellular APL computer
K J Thurber, J W Myrna
IEEE Trans on Comp
vol C-19, no 4, april 1970, pag 291

51 With associative memory, speed limit is no barrier
Jack A Rudolph, Louis C Fulmer, Williard C Weilander
Electronics, june 22, 1970, pag 96

52 MOS associative memories
L D Wald
The Electronic Engineer, august 1970, pag 54

53 Associative memory chips: fast, versatile and here
James Bartlett, John Mudge, John Springer
Electronics, aug 17, 1970, pag 217

54 The architecture of a large associative processor
G J Lipovski
Proc Spring Joint Comp Conf (AFIPS)
vol 36, 1970, pag 385

55 Associative capabilities for mass storage through array organization
S T Sacks, N A Palley, H Shubin, A A Afifi
Proc Fall Joint Comp Conf (AFIPS)
vol 37, 1970, pag 609

56 Iterrupt processing with queued content-addressable memories
J D Erwin, E D Jensen
Proc Fall Joint Comp Conf (AFIPS)
vol 37, 1970, pag 621

57 An augmented content-adressed memory array for implementation with large scale integration
William H Kautz
Jrnl of the Ass for Comp Machinery
vol 18, no 1, jan 1971, pag 19

58 The hybrid.associative memory concept
A Weinberger
Computer Design, jan 1971, pag 77

59 Design approaches for cache memory control
R M Meade
Computer Design, jan 1971, pag 87

60 The coming age of the associative processor
J A Rudolph, L C Fulmer, W C Meilander
Electronics, vol 44, febr 15, 1971, pag 91

61 Design of an associative memory
Willis K King
IEEE Trans on Comp
vol C-20, no 6, june 1971, pag 671

62 Associative techniques in the solution of data management problems
C R DeFiore, N J Stillman, P Bruce Berra
Proc of the annual conference of the ACM
aug 3-5, 1971

63 Applications of associative processors
K J Thurber, R D Berg
Computer Design, nov 1971, pag 103

64 Counting responders in an associative memory
C C Foster, F D Stockton
IEEE Trans on Comp
vol C-20, no 12, dec 1971, pag 1580

65 An associative processor for air traffic control
K J Thurber
Proc Spring Joint Comp Conf (AFIPS)
vol 38, 1971, pag 49

66 Associative processing of line-drawings
N J Stillman, C R DeFiore, P Bruce Berra
Proc Spring Joint Comp Conf (AFIPS)
vol 38, 1971, pag 557

67 The associative memory and its applications
A B E Ellis
The Marconi Review, first quarter, 1972, pag 42

68 Low-cost associative memory
J H Mundy, J F Burgess, R E Joynson, C Neugebauer
IEEE Jrnl of Solid State Circuits
vol SC-7, no 5, oct 1972, pag 364

69 A production implementation of an associative array processor - STARAN
Jack A Rudolph
Proc Fall Joint Comp Conf (AFIPS)
vol 41, 1972, pag 229

70 -PEPE Computer architecture
B A Crane ea, pag 57
-The PEPE support software system
D E Wilson, pag 61
-Application of PEPE to radar dating processing
G D Bergland, C F Hunnicutt, pag 65
-Parrallel processing of ballistic missile defense radar data with PEPE
J A Cornell, pag 69
Compcon 1972
Sixt annual IEEE Comp Society Internat Conf

71 Hardware floating point arithmetic on an associative processor
K J Thurber, P C Patton
Sixth annual IEEE Comp Society Internat Conf
Compcon 1972, pag 275

72 The OMEN computers: associative array processors
L G Higbie
Sixth annual IEEE Comp Society Internat Conf
Compcon 1972, pag 287

73 Parallel digital differential analyzer with
arbitrary stored interconnections
D S Dawoud, N Z El-Araby
IEEE Trans on Comp
vol C-22, no 1, jan 1973, pag 41

74 A search memory for many-to-many comparisons
D W Digby
IEEE Trans on Comp
vol C-22, no 1, jan 1973, pag 768

75 An implementation of a data management system on an
associative processor
Richard Moulder
Proc Nat Comp Conf (AFIPS)
vol 42, 1973, pag 171

76 Aircraft conflict detection in an associative processor
H R Downs
Proc Nat Comp Conf (AFIPS)
vol 42, 1973, pag 177

77 A data management system utilizing an associative
memory
Caspar R DeFiore, P Bruce Berra
Proc Nat Comp Conf (AFIPS)
vol 42, 1973, pag 181

78 Associative processer applications to real-time
data management
R R Linde, R Gates, Te-Fu Peng
Proc Nat Comp Conf (AFIPS)
vol 42, 1973, pag 187

79 HAPPE Honeywell associative parallel processing ensemble
D E Marvel
Proc of the 1st annual symposium on computer
architecture, dec 1973, vol 2, no 4, pag 261

80 Assoziativspeicher: Konzeption-Anwendung-Aufbau
G Weil
Nachrichten Technische Zeitschrift
Kurier 1973, pag K175

81 Toward a low-cost cell design for high-speed
MOS associative memories
R M Lea
Sientific and research paper, vol 2,
Datafair 1973, conference papers
The British Computer Society

82 An efficient associative processor using bulk storage
H H Love
Proc of the Sagamore Comp Conf on Parallel
Processing, 1973, pag 103

83 A quantitative analysis of the utilisation of associative memories in data management
Caspar R DeFiore, P Bruce Berra
IEEE Trans on Comp
vol C-23, no 2, feb 1974, pag 121

84 Multiple match resolvers: A new design method
G A Anderson
IEEE Trans on Comp
vol C-23, no 12, dec 1974, pag 1317

85 Assoziatieve Datenstrukturen
W Kestner, J Saniter, W Strasser, U Trambacz
Analyse und Synthese dynamischer Systeme
Einfuehrung in Computer Graphics
no 55, 1974, par 4.7, pag 4.69
Technische Universitaet Berlin

86 Some problems in associative processor applications to data base management
P Bruce Berra
Proc Nat Comp Conf (AFIPS)
vol 43, 1974, pag 1

87 RADCAP - An operational parallel processing facility
James D Feldman, Louis C Fulmer
Proc Nat Comp Conf (AFIPS)
vol 43, 1974, pag 7

88 STARAN parallel processor system hardware
Kenneth E Batcher
Proc Nat Comp Conf (AFIPS)
vol 43, 1974, pag 405

89 A study of fault tolerance techniques for associative processors
B Parhami, A Avizienis
Proc Nat Comp Conf (AFIPS)
vol 43, 1974, pag 643

90 APL - a language for associative data handling in PL/1
G G Dodd
Proc Fall Joint Comp Conf (AFIPS)
vol 29, 1966, pag 677

91 An Algol-based associative language
Jerome A Feldman, Paul D Rovner
Comm of the ACM, vol 12, no 8, aug 1969, pag 439

92 A parallel assembler for ILLIAC IV
J M Randal
Proc of the Sagamore Comp Conf on Parallel Processing, 1973, pag 78

93 STARAN parallel processor system software
Edward W Davis
Proc Nat Comp Conf (AFIPS)
vol 43, 1974, pag 17

94 Some thoughts on associative processing languages
William W Patterson
Proc Nat Comp Conf (AFIPS)
vol 43, 1974, pag 23

95 Sorting
William A Martin
Computing Surveys, vol 3, no 3, dec 1971, pag 147

96 Collegedictaat Informatica II
hoofdstuk 3, Sorteren
Prof.dr.R J Lunbeck
Onderafdeling der Wiskunde
Technische Hogeschool Eindhoven

97 Associative self-sorting memory
R R Seeber
Proc Eastern Joint Comp Conference
dec 1960, New York, vol 18, pag 179

98 A method for resolving multiple responses in a
parallel search file
E H Frei, J Goldberg
IRE Trans on El Computers
vol EC-10, nc 12, dec 1961, pag 718

99 Associative memory with ordered retrieval
R R Seeber, A B Lindquist
IBM Journal, jan 1962, pag 126

100 Retrieval of ordered lists from a content-adressed memory
Morton H Lewin
RCA-Review, june 1962, pag 215

101 Resolving multiple responses in an associative memory
H S Miller
IEEE Trans on El Comp
vol EC-13, okt 1964, pag 614

102 On ordered retrieval from an associative memory
L R Johnson, M H McAndrew
IBM-Jounal, april 1965, pag 189

103 A simple proof of Lewin's ordered retrieval
theorem for associative memories
A Wolinsky
Communications of the ACM
vol 11, july 1968, pag 488

104 Fast sorting
Caspar R DeFiore
Datamation, aug 1, 1970, pag 47

105 Sorting in a paging environment
B S Brawn, F G Gustavson, E S Mankin
Communications of the ACM
Vol 13, no 8, aug 1970, pag 483

106 On binary search trees
J Nievergelt, C K Wong
Proc of IFIP Congress, 1971, vol 1, pag 91

107 Nearly optimal binary search trees
J Bruno, E G Coffman
Proc of IFIP Congress, 1971, vol 1, pag 99

108 Analysis of associative retrieval algorithms
R L Rivest
Rapport de Recherche no. 54, Fevrier 1974
IRIA-Laboria, Le Chesnay - France

109 Computer languages for symbol manipulations
B F Green
IRE Trans on El Comp
vol EC-10, dec 1961, pag 729

110 ADAM - A problem-oriented symbol processor
A P Mullery, R F Schauer, R Rice
Proc Spring Joint Comp Conf (AFIPS)
vol 23, 1963, pag 367

111 Symbol manipulation languages
P W Abrahams
In: Advances in Computers. Editors: Alt, Rubinoff
vol 9, 1968, Academic Press, New York

112 Symbol manipulation with an associative memory
R R Seeber
Proc of the 16th National ACM Meeting, sept 1961

113 An associative parallel processing system for
non-numerical computation
P A Beaven, D W Lewin
The Computer Journal, vol 15, no 4, 1972, pag 343

114 Het ontwerpen en het onderzoeken van de bruikbaarheid
van een processor voor tekenverwerking
J G J van den Berg
Afstudeerverslag, Lab.voor schakeltechniek en techniek
van de informatieverwerkende machines, 28 okt, 1974
Technische Hogeschool Delft

115 Associative parallel processing for the fast Fourier transform
M A Wesley
IEEE Trans on Audio and Electroacoustics
vol AU-17, no 2, june 1969, pag 162

116 Numerical solution of partial differential equations by associative processing
P A Gilmore
Proc Fall Joint Comp Conf (AFIPS)
vol 39, nov 1971, pag 411

117 Organization of a 'fixed-plus-variable' structure computer for computation of eigenvalues and eigenvectors of real symmetric matrices
G Estrin, C R Viswanathan
Journal of the ACM, vol 9, no 1, jan 1962, pag 41

118 Organizing matrices and matrix operations for paged memory systems
A C McKellar, E G Coffman
Comm of the ACM, vol 12, no 3, march 1969, pag 153

119 Sparsity-directed decomposition for Gaussian elimination on matrices
E G Ogbuobiri, William F Tinney, John W Walker
IEEE Trans on power apparatus and systems
vol PAS-89, no 1, jan 1970, pag 141

120 Dynamic storage and retrieval in sparsity programming
E G Ogbuobiri
IEEE Trans on power apparatus and systems,
vol PAS-89, no 1, jan 1970, pag 150

121 An optimal ordering of electronic circuit equations for a sparse matrix solution
Robert D Berry
IEEE Trans on circuit theory, vol CT-18,
no 1, jan 1971, pag 40

122 On optimal-pivoting algorithms in sparse matrices
H Y Hsieh, M S Ghausi
IEEE Trans on circuit theory, jan 1972, pag 93

123 Sparse matrix techniques in computer-aided design
I DeLotto, G Pierini, P Tiberio
International Computer Symposium 1973

124 A survey of indexing techniques for sparse matrices
Udo W Pooch, Al Nieder
Computing Surveys, vol 5, no 2, june 1973, pag 109

125 Matrix inversion using parallel processing
Marshall C Pease
Journal of the Ass for Comp Machinery,
Vol 14, No 4, oct 1967, pag 757

126 Direct solutions of sparse network equations by optimally ordered triangular factorization
William F Tinney, John W Walker
Proc of the IEEE, vol 55, no 11, nov 1967, pag 1801

127 Some programming techniques for processing multi-dimensional matrices in a paging environment
J L Elshoff
Proc Nat Comp Conf (AFIPS)
vol 43, 1974, pag 185

128 An algorithm for path connections and its applications
C Y Lee
IRE Trans on El Comp
no 9, sept 1961, pag 346

129 Path finding with associative memory
B A Crane
IEEE Trans on Comp
vol C-17, july 1968, pag 691

130 An efficient search algorithm to find the elementary circuits in a graph
J C Tiernan
Communications of the ACM
vol 13, no 12, dec 1970, pag 722

131 Application of content-addressed memory for dynamic storage allocation
Y Chu
RCA Review, vol 26, march 1965, pag 140

132 Dynamic program behaviour under paging
G H Fine, C W Jackson, P V McIsaac
Proc of the 21st National Conference of the ACM, 1966, pag 223

133 Dynamic storage allocation systems
B Randell, C J Kuehner
Comm of the ACM, vol 11, no 5, may 1968, pag 297

134 Virtual memory, processes, and sharing in MULTICS
Robert C Daley, Jack B Dennis
Comm of the ACM, vol 11, no 5, may 1968, pag 306

135 The working set model for program behaviour
Peter J Denning
Comm of the ACM, vol 11, no 5, may 1968, pag 323

136 Demand paging in perspective
C J Kuehner, B Randell
Proc Fall Joint Comp Conf (AFIPS)
vol 33, part 2, 1968, pag 1011

137 Program behaviour in a paging environment
B S Braun, F G Gustavson
Proc Fall Joint Comp Conf (AFIPS)
vol 33, part 2, 1968, pag 1019

138 Paging studies made on the ICT Atlas computer
M H J Baylis, D G Fletcher, D J Howarth
Proc of IFIP Congress, 1968, vol 1, pag 831

139 Storage hierarchy systems
H Katzan
Proc Spring Joint Comp Conf (AFIPS)
vol 38, 1971, pag 325

140 Storage organization and management in TENEX
D L Murphy
Proc Fall Joint Comp Conf (AFIPS)
vol 41, 1972, pag 23

141 The evaluation of virtual machine architecture
J P Buzen, U O Gagliardi
Proc Nat Comp Conf (AFIPS)
vol 42, 1973, pag 291

142 Architecture of virtual machines
R P Goldberg
Proc Nat Comp Conf (AFIPS)
vol 42, 1973, pag 309

143 An efficient virtual machine implementation
R J Srodawa, L A Bates
Proc Nat Comp Conf (AFIPS)
vol 42, 1973, pag 301

144 Experimental data on page replacement algorithm
N A Oliver
Proc Nat Comp Conf (AFIPS)
vol 43, 1974, pag 179

145 The mechanization of a push-down stack
C B Carlson
Proc Fall Joint Comp Conf (AFIPS),
vol 24, nov 1963, pag 243

146 Burroughs B6500/B7500 stack mechanism
E A Hauck, E A Dent
Spring Joint Comp Conference, 1968, pag 245

147 Computer with stack memory
A N Myamlin, V K Smirnov
Proc of IFIP Congress 1968, vol 2, pag 818

148 Ein Rechnerkonzept mit assoziativem Arbeitsspeicher
- Prinzip und Organisation, pag 79
H O Leilich, I Karlowsky, W Lawrenz,H Ch Ziedler
- Speicher- und Namensverwalrung, pag 91
I Karlowsky
- Programmorganisation, pag 101
W Lawrenz
Lecture notes in Computer Science
Fachtagung Struktur und Betrieb von Rechnersystemen
Braunschweig, 20-03-1974, Springer Verlag

149 The SDS Sigma 7: a real-time time-sharing computer
M J Mendelson, A W England
Proc Fall Joint Comp Conf (AFIPS)
vol 29, 1966, pag 51

150 An optimization model for time-sharing
D W Fife
Proc Spring Joint Comp Conf (AFIPS)
vol 28, 1966, pag 97

151 A time-sharing system using an associative memory
A B Lindquist, R R Seeber, L W Comeau
Proc of the IEEE, vol 54, dec 1966, pag 1774

152 An associative parallel processor with application
to picture processing
R H Fuller, R M Bird
Proc Fall Joint Comp Conf (AFIPS)
vol 27, part 1, 1965, pag 105

153 Pattern recognition using an associative memory
S S Yau, C C Yang
IEEE Trans on El Comp
vol EC-15, dec 1966, pag 944

154 Associative memory for spatio-temporal patterns
K Fukushima
First USA-JAPAN Comp Conf
1972, session 2-4-2, pag 36

155 Een flexibele methode voor het machinaal lezen van geschreven cijfers
M Beun
Philips Technische Tijdschrift
Jaargang 33, 1973, no 4, pag 93 en no 5, pag 135

156 Automatisch herkennen van handgeschreven cijfers
G J W van Nunen
Verslag van een onderzoek aan het Philips Natuurkundig Laboratorium te Eindhoven
juli 1973, 84 pag en bijlagen

157 Parallel processor systems, technologies and applications
L C Hobbs
Spartan Books, New York

158 Simulation using GPSS
T J Schriber
John Wiley and Sons

159 Multiprocessors and parallel processing
Comte Corporation
P H Enslow
John Wiley and Sons

160 Parallel processing
Lecture Notes in Computer Science
edited by G Goos and J Hartmanis
Springer Verlag, 1974

aanvulling met recente artikelen

161 Simultaneous complex searches in associative memories
D P Agrawal
Conf on Computer Systems and Technology 1974
Conference Publication nr 121, pag 180

162 Storing extensible tables in associative memories with fixed word-lengths
B Parhami
Conf Record of the Seventh Asilomar Conference on Circuits, Systems and Computers
1974, pag 439

A P P E N D I X I

PROGRAMMA VAN DE BASISROUTINES

```
 1   0 0900                      ORG   2304
 2   0 0900            *
 3   0 0900            *         BIJ DE ROUTINES WORDEN DE REGISTERS
 4   0 0900            *         A,B,C EN D GEWIJZIGD.
 5   0 0900            *
 6   0 0900            *
 7   0 0900            *         LADEN VAN DE OUTPUTPOORTEN 11,12 EN 13.
 8   0 0900            *         SET SELECTIEKAART 6 EN GENEREER STARTPULS.
 9   0 0900            *
10   0 0900  11000001  SEL6      LAB                 DATA NAAR ACC.
11   1 0901  01010011            OUT   11B           LAAD OUTPUTPOORT 11
12   2 0902  11000010            LAC                 DATA NAAR ACC.
13   3 0903  01010101            OUT   12B           LAAD OUTPUTPOORT 12
14   4 0904  11000011            LAD                 DATA NAAR ACC.
15   5 0905  01010111            OUT   13B           LAAD OUTPUTPOORT 13
16   6 0906  00000110            LAI   006B          SELECTEER SELECTIEKAART 6
     7 0907  00000110
17   8 0908  01010001            OUT   10B
18   9 0909  00000110            LAI   306B          ENABLE SELECTIEKAART 6
    10 090A  11000110
19  11 090B  01010001            OUT   10B
20  12 090C  00000110            LAI   006B          DISABLE SELECTIEKAART 6
    13 090D  00000110
21  14 090E  01010001            OUT   10B
22  15 090F  00000110            LAI   4
    16 0910  00000100
23  17 0911  10000011            ADD                 GEEF STARTSIGNAAL
24  18 0912  01010111            OUT   13B
25  19 0913  00000110            LAI   006B          SELECTEER SELECTIEKAART 6
    20 0914  00000110
26  21 0915  01010001            OUT   10B
27  22 0916  00000110            LAI   306B          ENABLE SELECTIEKAART 6
    23 0917  11000110
28  24 0918  01010001            OUT   10B
29  25 0919  00000110            LAI   006B          DISABLE SELECTIEKAART 6
    26 091A  00000110
30  27 091B  01010001            OUT   10B
31  28 091C  01000100            JMP   READY
    29 091D  11110110
    30 091E  00001001
32  31 091F            *
33  31 091F            *         LADEN VAN DE OUTPUTPOORTEN 11,12 EN 13
34  31 091F            *         SET SELECTIEKAART 6 EN GEEF GEEN STARTPULS
35  31 091F            *
36  31 091F  11000001  OUT6      LAB                     DATA NAAR ACC
37  32 0920  01010011            OUT   11B               LAAD OUTPUTPOORT 11
38  33 0921  00000110            LAI   100B
    34 0922  01000000
39  35 0923  01010101            OUT   12B               LAAD OUTPUTPOORT 12
40  36 0924  00000110            LAI   000B
    37 0925  00000000
41  38 0926  01010111            OUT   13B               LAAD OUPUTPOORT 13
42  39 0927  00000110            LAI   006B              SELECTEER SELECTIEKAART 6
    40 0928  00000110
43  41 0929  01010001            OUT   10B
44  42 092A  00000110            LAI   306B              ENABLE SELECTIEKAART 6
    43 092B  11000110
45  44 092C  01010001            OUT   10B
46  45 092D  00000110            LAI   006B              DISABLE SELECTIEKAART 6
```

```
    46 092E  00000110
47  47 092F  01010001          OUT   10B
48  48 0930  00000111          RET
49  49 0931            *
50  49 0931            *       RESET REGISTERS C EN D.
51  49 0931            *
52  49 0931  00010110 RT1213   LCI   100B
    50 0932  01000000
53  51 0933  00011110          LDI   000B
    52 0934  00000000
54  53 0935  01000100          JMP   SEL6
    54 0936  00000000
    55 0937  00001001
55  56 0938            *
56  56 0938            *       RESET REGISTERS B EN D.
57  56 0938            *
58  56 0938  00001110 RT1113   LBI   000B
    57 0939  00000000
59  58 093A  00011110          LDI   000B
    59 093B  00000000
60  60 093C  01000100          JMP   SEL6
    61 093D  00000000
    62 093E  00001001
61  63 093F            *
62  63 093F            *       LAAD RAM INPUTREGISTER 0.
63  63 093F            *
64  63 093F  00001110 LRAM0    LBI   001B
    64 0940  00000001
65  65 0941  01000100          JMP   RT1213
    66 0942  00110001
    67 0943  00001001
66  68 0944            *
67  68 0944            *       LAAD RAM INPUTREGISTER 1.
68  68 0944            *
69  68 0944  00001110 LRAM1    LBI   011B
    69 0945  00001001
70  70 0946  01000100          JMP   RT1213
    71 0947  00110001
    72 0948  00001001
71  73 0949            *
72  73 0949            *       LAAD RAM INPUTREGISTER 2.
73  73 0949            *
74  73 0949  00001110 LRAM2    LBI   021B
    74 094A  00010001
75  75 094B  01000100          JMP   RT1213
    76 094C  00110001
    77 094D  00001001
76  78 094E            *
77  78 094E            *       LAAD SCHUIFREG. 0 VAN KENMERKREG.
78  78 094E            *
79  78 094E  00001110 LCAM0    LBI   002B
    79 094F  00000010
80  80 0950  01000100          JMP   RT1213
    81 0951  00110001
    82 0952  00001001
81  83 0953            *
82  83 0953            *       LAAD SCHUIFREG. 1 VAN KENMERKREG.
83  83 0953            *
84  83 0953  00001110 LCAM1    LBI   012B
    84 0954  00001010
```

```
 85   85 0955  01000100          JMP   RT1213
      86 0956  00110001
      87 0957  00001001
 86   88 0958            *
 87   88 0958            *       LAAD SCHUIFREG. 2 VAN KENMERKREG.
 88   88 0958            *
 89   88 0958  00001110 LCAM2    LBI   022B
      89 0959  C0010010
 90   90 095A  01000100          JMP   RT1213
      91 095B  00110001
      92 095C  00001001
 91   93 095D            *
 92   93 095D            *       LAAD SCHUIFREG. 0 EN INPUTREG. 0
 93   93 095D            *
 94   93 095D  C0001110 LCRAM0   LBI   003B
      94 095E  00000011
 95   95 095F  01000100          JMP   RT1213
      96 0960  00110001
      97 0961  C0001001
 96   98 0962            *
 97   98 0962            *       LAAD SCHUIFREG. 1 EN INPUTREG. 1
 98   98 0962            *
 99   98 0962  C0001110 LCRAM1   LBI   013B
      99 0963  00001011
100  100 0964  01000100          JMP   RT1213
     101 0965  00110001
     102 0966  C0001001
101  103 0967            *
102  103 0967            *       LAAD SCHUIFREG. 2 EN INPUTREG. 2
103  103 0967            *
104  103 0967  C0001110 LCRAM2   LBI   023B
     104 0968  00010011
105  105 0969  01000100          JMP   RT1213
     106 096A  00110001
     107 096B  00001001
106  108 096C            *
107  108 096C            *       LAAD SCHUIFREG. 0,1 EN 2
108  108 096C            *
109  108 096C  C0001110 LCAMV0   LBI   006B
     109 096D  00000110
110  110 096E  01000100          JMP   RT1213
     111 096F  00110001
     112 0970  C0001001
111  113 0971            *
112  113 0971            *       LAAD SCHUIFREG. 1 EN 2
113  113 0971            *
114  113 0971  C0001110 LCAMV1   LBI   016B
     114 0972  00001110
115  115 0973  01000100          JMP   RT1213
     116 0974  00110001
     117 0975  C0001001
116  118 0976            *
117  118 0976            *       LAAD SCHUIFREGISTER 2
118  118 0976            *       LAAD VANAF 2-DE BYTE DATA IN CAM REGISTER.
119  118 0976            *
120  118 0976  C0001110 LCAMV2   LBI   026B
     119 0977  00010110
121  120 0978  01000100          JMP   RT1213
     121 0979  00110001
     122 097A  00001001
```

```
122 123 097B            *
123 123 097B            *        SCHUIF INHOUD CAM REGISTER OVER AANTAL BITS
124 123 097B            *        AANTAL BITS STAAT AANGEGEVEN OP DATA-BUS.
125 123 097B            *        LAAD DAARNA ASSOCIATIE-ADRES IN CAO.
126 123 097B            *
127 123 097B  C0001110 SHIFT     LBI   040B
    124 097C  00100000
128 125 097D  01000110           CAL   RT1213
    126 097E  00110001
    127 097F  C0001001
129 128 0980  00010110           LCI   100B
    129 0981  01000000
130 130 0982  01000100           JMP   RT1113
    131 0983  00111000
    132 0984  C0001001
131 133 0985            *
132 133 0985            *        ENABLE BUSBUFFER 0. BYTE 0 VAN RAM
133 133 0985            *        OUTPUTREG. WORDT OP DATA-BUS GEZET.
134 133 0985            *
135 133 0985  00001110 ORAM0     LBI   100B
    134 0986  01000000
136 135 0987  01000100           JMP   OUT6
    136 0988  C0011111
    137 0989  00001001
137 138 098A            *
138 138 098A            *        ENABLE BUSBUFFER 1. BYTE 1 VAN RAM
139 138 098A            *        OUTPUTREG. WORDT OP DATA-BUS GEZET.
140 138 098A            *
141 138 098A  C0001110 ORAM1     LBI   110B
    139 098B  01001000
142 140 098C  01000100           JMP   OUT6
    141 098D  00011111
    142 098E  C0001001
143 143 098F            *
144 143 098F            *        ENABLE BUSBUFFER 2. BYTE 2 VAN RAM
145 143 098F            *        OUTPUTREG. WORDT OP DATA-BUS GEZET.
146 143 098F            *
147 143 098F  00001110 ORAM2     LBI   120B
    144 0990  01010000
148 145 0991  01000100           JMP   OUT6
    146 0992  00011111
    147 0993  00001001
149 148 0994            *
150 148 0994            *        ZET ASSOCIATIE-ADRES OP DATA-BUS.
151 148 0994            *
152 148 0994  00001110 OCAM      LBI   200B
    149 0995  10000000
153 150 0996  01000100           JMP   OUT6
    151 0997  00011111
    152 0998  00001001
154 153 0999            *
155 153 0999            *        SCHRIJF CAM OP EERSTE LEGE PLAATS.
156 153 0999            *
157 153 0999  00010110 WCAM      LCI   101B
    154 099A  01000001
158 155 099B  01000100           JMP   RT1113
    156 099C  00111000
    157 099D  00001001
159 158 099E            *
160 158 099E            *        SCHRIJF RAM OP ASSOCIATIE-ADRES.
```

```
161 158 099E            *
162 158 099E  C0010110 WRAM     LCI   140B
    159 099F  01100000
163 160 09A0  01000100          JMP   RT1113
    161 09A1  00111000
    162 09A2  00001001
164 163 09A3            *
165 163 09A3            *       SCHRIJF CAM OP EERSTE LEGE PLAATS.
166 163 09A3            *       SCHRIJF RAM OP ASSOCIATIE-ADRES.
167 163 09A3            *
168 163 09A3  00010110 WCRAM    LCI   141B
    164 09A4  01100001
169 165 09A5  01000100          JMP   RT1113
    166 09A6  00111000
    167 09A7  00001001
170 168 09A8            *
171 168 09A8            *       SCHRIJF CAM OP EXTERN ADRES.
172 168 09A8            *
173 168 09A8  00010110 SCAM     LCI   102B
    169 09A9  01000010
174 170 09AA  01000100          JMP   RT1113
    171 09AB  00111000
    172 09AC  00001001
175 173 09AD            *
176 173 09AD            *       SCHRIJF RAM OP EXTERN ADRES.
177 173 09AD            *
178 173 09AD  00010110 SRAM     LCI   340B
    174 09AE  11100000
179 175 09AF  01000100          JMP   RT1113
    176 09B0  00111000
    177 09B1  00001001
180 178 09B2            *
181 178 09B2            *       SCHRIJF CAM OP EXTERN ADRES.
182 178 09B2            *       SCHRIJF RAM OP EXTERN ADRES.
183 178 09B2            *
184 178 09B2  00010110 SCRAM    LCI   342B
    179 09B3  11100010
185 180 09B4  01000100          JMP   RT1113
    181 09B5  00111000
    182 09B6  00001001
186 183 09B7            *
187 183 09B7            *       WIS CAM OP EXTERN ADRES.
188 183 09B7            *
189 183 09B7  00010110 RSTCAM   LCI   104B
    184 09B8  01000100
190 185 09B9  01000100          JMP   RT1113
    186 09BA  00111000
    187 09BB  00001001
191 188 09BC            *
192 188 09BC            *       WIS CAM OP ASSOCIATIE.
193 188 09BC            *
194 188 09BC  00010110 CLCAM    LCI   110B
    189 09BD  01001000
195 190 09BE  01000100          JMP   RT1113
    191 09BF  00111000
    192 09C0  00001001
196 193 09C1            *
197 193 09C1            *       LEES RAM OP ASSOCIATIE-ADRES.
198 193 09C1            *
199 193 09C1  00010110 RRAM     LCI   120B
```

```
    194 09C2  01010000
200 195 09C3  01000100          JMP   RT1113
    196 09C4  00111000
    197 09C5  00001001
201 198 09C6            *
202 198 09C6            *       LEES RAM OP EXTERN ADRES.
203 198 09C6            *
204 198 09C6  00010110 RARAM    LCI   320B
    199 09C7  11010000
205 200 09C8  01000100          JMP   RT1113
    201 09C9  00111000
    202 09CA  00001001
206 203 09CB            *
207 203 09CB            *       GEEF TIJDELIJK VRIJ PULS.
208 203 09CB            *
209 203 09CB  00001110 TVPULS   LBI   000B
    204 09CC  00000000
210 205 09CD  00010110          LCI   100B
    206 09CE  01000000
211 207 09CF  00011110          LDI   001B
    208 09D0  00000001
212 209 09D1  01000100          JMP   SEL6
    210 09D2  00000000
    211 09D3  00001001
213 212 09D4            *
214 212 09D4            *       OPHEFFEN VAN DE TIJDELIJK VRIJ PULSEN.
215 212 09D4            *
216 212 09D4  00001110 TVRST    LBI   000B
    213 09D5  00000000
217 214 09D6  00010110          LCI   100B
    215 09D7  01000000
218 216 09D8  00011110          LDI   002B
    217 09D9  00000010
219 218 09DA  01000100          JMP   SEL6
    219 09DB  00000000
    220 09DC  00001001
220 221 09DD            *
221 221 09DD            *       ZET DATA EN MASKER OP DE BUS.
222 221 09DD            *       REGISTER A MOET DE DATA BEVATTEN.
223 221 09DD            *       REGISTER B MOET HET MASKER BEVATTEN.
224 221 09DD            *
225 221 09DD  01010011 DMOUT    OUT   11B       ZET DATA IN OUTPUTPOORT 11
226 222 09DE  11000001          LAB
227 223 09DF  01010101          OUT   12B       ZET MASKER IN OUTPUTPOORT 12
228 224 09E0  00000110          LAI   005B      SELECTEER SELECTIEKAART 5
    225 09E1  00000101
229 226 09E2  01010001          OUT   10B
230 227 09E3  00000110          LAI   305B      ENABLE SELECTIEKAART 5 OUT
    228 09E4  11000101
231 229 09E5  01010001          OUT   10B
232 230 09E6  00000110          LAI   005B      DISABLE SELECTIEKAART 5
    231 09E7  00000101
233 232 09E8  01010001          OUT   10B
234 233 09E9  00000111          RET
235 234 09EA            *
236 234 09EA            *       LEES DATA VAN DE DATA-BUS.
237 234 09EA            *       REGISTER B BEVAT INGELEZEN DATA.
238 234 09EA            *
239 234 09EA  00000110 DATAIN   LAI   005B      SELECTEER SELECTIEKAART 5
    235 09EB  00000101
```

```
240 236 09EC  01010001          OUT   10B
241 237 09ED  00000110          LAI   205B      ENABLE SELECTIEKAART 5 IN
    238 09EE  10000101
242 239 09EF  01010001          OUT   10B
243 240 09F0  01000001          INP   0B
244 241 09F1  11001000          LBA             REG B:=DATA
245 242 09F2  00000110          LAI   005B      DISABLE SELECTIEKAART 5
    243 09F3  00000101
246 244 09F4  01010001          OUT   10B
247 245 09F5  00000111          RET
248 246 09F6            *
249 246 09F6            *       TEST OF ASSOCIATIEVE GEHEUGEN
250 246 09F6            *       OPDRACHT HEEFT UITGEVOERD.
251 246 09F6            *
252 246 09F6  00000110 READY    LAI   005B      SELECTEER SELECTIEKAART 5
    247 09F7  00000101
253 248 09F8  01010001          OUT   10B
254 249 09F9  00000110          LAI   205B      ENABLE SELECTIEKAART 5 IN
    250 09FA  10000101
255 251 09FB  01010001          OUT   10B
256 252 09FC  01000011          INP   1B
257 253 09FD  00011010          RAR             ZET READY-BIT IN CARRY FF
258 254 09FE  01000000          JFC   READY     TEST READY=HOOG
    255 09FF  11110110
      0 0A00  00001001
259   1 0A01  00000110          LAI   005B      DISABLE SELECTIEKAART 5
      2 0A02  00000101
260   3 0A03  01010001          OUT   10B
261   4 0A04  00000111          RET
262   5 0A05            *
263   5 0A05            *       TEST OF ASSOCIATIEVE GEHEUGEN VOL IS.
264   5 0A05            *       ZO JA DAN WORDT CARRY-FF GESET.
265   5 0A05            *
266   5 0A05  00000110 CAMVOL   LAI   005B      SELECTEER SELECTIEKAART 5
      6 0A06  00000101
267   7 0A07  01010001          OUT   10B
268   8 0A08  00000110          LAI   205B      ENABLE SELECTIEKAART 5 IN
      9 0A09  10000101
269  10 0A0A  01010001          OUT   10B
270  11 0A0B  01000011          INP   1B
271  12 0A0C  00011010          RAR
272  13 0A0D  00011010          RAR             ZET AVOL-BIT IN CARRY-FF
273  14 0A0E  00000110          LAI   005B      DISABLE SELECTIEKAART 5
     15 0A0F  00000101
274  16 0A10  01010001          OUT   10B
275  17 0A11  00000111          RET
276  18 0A12            *
277  18 0A12            *       TEST OF ASSOCIATIEVE GEHEUGEN ASSOCIEERT.
278  18 0A12            *       ZO JA DAN WORDT CARRY-FF GESET.
279  18 0A12            *
280  18 0A12  00000110 ASS     LAI   005B      SELECTEER SELECTIEKAART 5
     19 0A13  00000101
281  20 0A14  01010001          OUT   10B
282  21 0A15  00000110          LAI   205B      ENABLE SELECTIEKAART 5 IN
     22 0A16  10000101
283  23 0A17  01010001          OUT   10B
284  24 0A18  01000011          INP   1B
285  25 0A19  00011010          RAR
286  26 0A1A  00011010          RAR
287  27 0A1B  00011010          RAR             ZET ASSOCIATIE-BIT IN CARRY-FF
```

```
288  28 0A1C  00000110          LAI   005B      DISABLE SELECTIEKAART 5
     29 0A1D  00000101
289  30 0A1E  01010001          OUT   10B
290  31 0A1F  00000111          RET
291  32 0A20            *
292  32 0A20            *       OPSTARTEN ASSOCIATIEVE GEHEUGEN.
293  32 0A20            *
294  32 0A20  00001110 START    LBI   000B
     33 0A21  00000000
295  34 0A22  01000110          CAL   DMOUT     ZET DON'T CARES OP DE BUS
     35 0A23  11011101
     36 0A24  00001001
296  37 0A25  01000110          CAL   LCAMVO    LAAD DON'T CARES IN CAM REGISTE
     38 0A26  01101100
     39 0A27  00001001
297  40 0A28  01000110          CAL   CLCAM     WIS CAM VOLLEDIG
     41 0A29  10111100
     42 0A2A  00001001
298  43 0A2B  01000110          CAL   TVRST
     44 0A2C  11010100
     45 0A2D  00001001
299  46 0A2E  00000111          RET
300  47 0A2F                    END

NO ERRORS
```

A P P E N D I X II

TESTPROGRAMMA VOOR HET MODEL

```
 1    0 0C00                        ORG    3072
 2    0 0C00             *
 3    0 0C00             *                SAVE REGISTERS H,L EN E
 4    0 0C00             *
 5    0 0C00   11000101  STORE      LAH
 6    1 0C01   11001110            LBL
 7    2 0C02   00101110            LHI    11
      3 0C03   00001011
 8    4 0C04   00110110            LLI    15
      5 0C05   00001111
 9    6 0C06   11111000            LMA
10    7 0C07   00110000            INL
11    8 0C08   11111001            LMB
12    9 0C09   11000100            LAE
13   10 0C0A   00110000            INL
14   11 0C0B   11111000            LMA
15   12 0C0C   00000111            RET
16   13 0C0D             *
17   13 0C0D             *                PRINT TEKST
18   13 0C0D             *
19   13 0C0D   11000111  TEKST      LAM
20   14 0C0E   00111100            CPI    'f'
     15 0C0F   00100100
21   16 0C10   01101000            JTZ     CRLF
     17 0C11   11000101
     18 0C12   00000001
22   19 0C13   11001000            LBA
23   20 0C14   01000110            CAL    TTYOUT
     21 0C15   10100001
     22 0C16   00000001
24   23 0C17   00110000            INL
25   24 0C18   01001000            JFZ    LNIETO
     25 0C19   00011100
     26 0C1A   00001100
26   27 0C1B   00101000            INH
27   28 0C1C   01000100  LNIETO    JMP    TEKST
     29 0C1D   00001101
     30 0C1E   00001100
28   31 0C1F             *
29   31 0C1F             *                RELOAD REGISTERS H,L EN E
30   31 0C1F             *
31   31 0C1F   00101110  RELOAD     LHI    11
     32 0C20   00001011
32   33 0C21   00110110            LLI    15
     34 0C22   00001111
33   35 0C23   11000111            LAM
34   36 0C24   00110000            INL
35   37 0C25   11001111            LBM
36   38 0C26   00110000            INL
37   39 0C27   11100111            LEM
38   40 0C28   11101000            LHA
39   41 0C29   11110001            LLB
40   42 0C2A   00000111            RET
41   43 0C2B             *
42   43 0C2B             *                LADEN CAM-REGISTERS
43   43 0C2B             *
44   43 0C2B   11000111  LOAD       LAM
45   44 0C2C   00001110            LBI    377B
     45 0C2D   11111111
```

```
46   46 0C2E  01000110            CAL   DMOUT
     47 0C2F  11011101
     48 0C30  00001001
47   49 0C31  01000110            CAL   LCAM0     LAAD SCHUIFREG. 0
     50 0C32  01001110
     51 0C33  00001001
48   52 0C34  11000000            LAA
49   53 0C35  00110000            INL
50   54 0C36  11000111            LAM
51   55 0C37  00001110            LBI   377B
     56 0C38  11111111
52   57 0C39  01000110            CAL   DMOUT
     58 0C3A  11011101
     59 0C3B  00001001
53   60 0C3C  01000110            CAL   LCAM1     LAAD SCHUIFREG. 1
     61 0C3D  01010011
     62 0C3E  00001001
54   63 0C3F  11000000            LAA
55   64 0C40  00110000            INL
56   65 0C41  11000111            LAM
57   66 0C42  00001110            LBI   377B
     67 0C43  11111111
58   68 0C44  01000110            CAL   DMOUT
     69 0C45  11011101
     70 0C46  00001001
59   71 0C47  01000110            CAL   LCAM2     LAAD SCHUIFREG. 2
     72 0C48  01011000
     73 0C49  00001001
60   74 0C4A  11000000            LAA
61   75 0C4B  00110000            INL
62   76 0C4C  00000111            RET
63   77 0C4D            *
64   77 0C4D            *                  ADRES UIT CAM INLEZEN EN PONSEN
65   77 0C4D            *
66   77 0C4D  01000110 ADROUT     CAL   OCAM      CAO-REG. OP DATABUS
     78 0C4E  10010100
     79 0C4F  00001001
67   80 0C50  01000110            CAL   DATAIN
     81 0C51  11101010
     82 0C52  00001001
68   83 0C53  00000110            LAI   007B
     84 0C54  00000111
69   85 0C55  10100001            NDB
70   86 0C56  00000100            ADI   260B      CONVERSIE NAAR ASCII
     87 0C57  10110000
71   88 0C58  11001000            LBA
72   89 0C59  01000110            CAL   TTYOUT
     90 0C5A  10100001
     91 0C5B  00000001
73   92 0C5C  11000000            LAA
74   93 0C5D  00000111            RET
75   94 0C5E  11111000 STOREA     LMA               ZET ACC IN GEHEUGEN
76   95 0C5F  01000110            CAL   DMOUT     ZET REG A EN B OP DATA-BUS
     96 0C60  11011101
     97 0C61  00001001
77   98 0C62  00110000            INL
78   99 0C63  11000000            LAA
79  100 0C64  00000111            RET
80  101 0C65  01000110 COMP       CAL   DATAIN    LEES DATA IN
    102 0C66  11101010
```

```
    103 0C67  00001001
 81 104 0C68  11000111          LAM
 82 105 0C69  00110000          INL
 83 106 0C6A  10111001          CPB            VERGELIJK DATA MET GEHEUGEN
 84 107 0C6B  11000000          LAA
 85 108 0C6C  C0000111          RET
 86 109 0C6D            *
 87 109 0C6D            *              INFORMATIE INLEZEN IN BANK 15
 88 109 0C6D            *
 89 109 0C6D  00101110          LHI    11
    110 0C6E  00001011
 90 111 0C6F  00110110          LLI    000B
    112 0C70  00000000
 91 113 0C71  00111110          LMI    252B
    114 0C72  10101010
 92 115 0C73  00110000          INL
 93 116 0C74  00111110          LMI    125B
    117 0C75  01010101
 94 118 0C76  00110000          INL
 95 119 0C77  00111110          LMI    017B
    120 0C78  00001111
 96 121 0C79  00110000          INL
 97 122 0C7A  00111110          LMI    125B
    123 0C7B  01010101
 98 124 0C7C  00110000          INL
 99 125 0C7D  00111110          LMI    252B
    126 0C7E  10101010
100 127 0C7F  00110000          INL
101 128 0C80  00111110          LMI    360B
    129 0C81  11110000
102 130 0C82  00110000          INL
103 131 0C83  00111110          LMI    377B
    132 0C84  11111111
104 133 0C85  00110000          INL
105 134 0C86  00111110          LMI    125B
    135 0C87  01010101
106 136 0C88  00110000          INL
107 137 0C89  00111110          LMI    000B
    138 0C8A  00000000
108 139 0C8B            *
109 139 0C8B            *              BEGIN TESTPROGRAMMA CAM
110 139 0C8B            *
111 139 0C8B  01000110          CAL    START      LADEN BEGINCONDITIES
    140 0C8C  00100000
    141 0C8D  00001010
112 142 0C8E  00110110          LLI    000B
    143 0C8F  00000000
113 144 0C90  01000110          CAL    LOAD       1E WOORD IN KENMERKREG.
    145 0C91  00101011
    146 0C92  00001100
114 147 0C93  00000110          LAI    003B
    148 0C94  00000011
115 149 0C95  01000110          CAL    DMOUT
    150 0C96  11011101
    151 0C97  00001001
116 152 0C98  01000110          CAL    SCAM       SCHRIJF CAM EXTERN ADRES
    153 0C99  10101000
    154 0C9A  00001001
117 155 0C9B  11000000          LAA
118 156 0C9C  01000110          CAL    ASS
```

```
    157 0C9D  00010010
    158 0C9E  00001010
119 159 0C9F  01100000          JTC   G1
    160 0CA0  10100100
    161 0CA1  00001100
120 162 0CA2  11111111          HLT
121 163 0CA3  11000000          LAA
122 164 0CA4  11000000 G1       LAA
123 165 0CA5  01000110          CAL   ADROUT
    166 0CA6  01001101
    167 0CA7  00001100
124 168 0CA8  01000110          CAL   STORE
    169 0CA9  00000000
    170 0CAA  00001100
125 171 0CAB  00101110          LHI   B(TEKST1)
    172 0CAC  00010001
126 173 0CAD  00110110          LLI   A(TEKST1)
    174 0CAE  11100111
127 175 0CAF  01000110          CAL   TEKST     PRINT: 3 ASS SCAM
    176 0CB0  00001101
    177 0CB1  00001100
128 178 0CB2  01000110          CAL   RELOAD
    179 0CB3  00011111
    180 0CB4  00001100
129 181 0CB5  11000000          LAA
130 182 0CB6  01000110          CAL   LOAD      2E WOORD IN KENMERKREG.
    183 0CB7  00101011
    184 0CB8  00001100
131 185 0CB9  11000000          LAA
132 186 0CBA  01000110          CAL   ASS
    187 0CBB  00010010
    188 0CBC  00001010
133 189 0CBD  01000000          JFC   G2
    190 0CBE  11000010
    191 0CBF  00001100
134 192 0CC0  11111111          HLT
135 193 0CC1  11000000          LAA
136 194 0CC2  11000000 G2       LAA
137 195 0CC3  01000110          CAL   WCAM      SCHRIJF CAM 1E LEGE PL.
    196 0CC4  10011001
    197 0CC5  00001001
138 198 0CC6  11000000          LAA
139 199 0CC7  01000110          CAL   ASS
    200 0CC8  00010010
    201 0CC9  00001010
140 202 0CCA  01100000          JTC   G3
    203 0CCB  11001111
    204 0CCC  00001100
141 205 0CCD  11111111          HLT
142 206 0CCE  11000000          LAA
143 207 0CCF  11000000 G3       LAA
144 208 0CD0  01000110          CAL   ADROUT
    209 0CD1  01001101
    210 0CD2  00001100
145 211 0CD3  01000110          CAL   STORE
    212 0CD4  00000000
    213 0CD5  00001100
146 214 0CD6  00101110          LHI   B(TEKST2)
    215 0CD7  00010001
147 216 0CD8  00110110          LLI   A(TEKST2)
```

```
    217 0CD9  11110100
148 218 0CDA  01000110          CAL   TEKST     PRINT: 0 ASS WCAM
    219 0CDB  00001101
    220 0CDC  C0001100
149 221 0CDD  01000110          CAL   RELOAD
    222 0CDE  C0011111
    223 0CDF  00001100
150 224 0CE0  11000000          LAA
151 225 0CE1  01000110          CAL   LOAD      3E WOORD IN KENMERKREG.
    226 0CE2  00101011
    227 0CE3  00001100
152 228 0CE4  00100110          LEI   000B
    229 0CE5  00000000
153 230 0CE6  11000000          LAA
154 231 0CE7  11000000 BACK     LAA
155 232 0CE8  01000110          CAL   WCAM      SCHRIJF CAM OP 1E LEGE PLAATS
    233 0CE9  10011001
    234 0CEA  00001001
156 235 0CEB  01000110          CAL   ASS
    236 0CEC  C0010010
    237 0CED  00001010
157 238 0CEE  01100000          JTC   G4
    239 0CEF  11110011
    240 0CF0  C0001100
158 241 0CF1  11111111          HLT
159 242 0CF2  11000000          LAA
160 243 0CF3  11000000 G4       LAA
161 244 0CF4  00100000          INE
162 245 0CF5  00000110          LAI   006B
    246 0CF6  00000110
163 247 0CF7  10111100          CPE
164 248 0CF8  01001000          JFZ   BACK
    249 0CF9  11100111
    250 0CFA  C0001100
165 251 0CFB  11000000          LAA
166 252 0CFC  01000110          CAL   CAMVOL    CAM VOL ?
    253 0CFD  00000101
    254 0CFE  C0001010
167 255 0CFF  01100000          JTC   G5
      0 0D00  00000100
      1 0D01  00001101
168   2 0D02  11111111          HLT
169   3 0D03  11000000          LAA
170   4 0D04  11000000 G5       LAA
171   5 0D05  00110110          LLI   000B
      6 0D06  00000000
172   7 0D07  01000110          CAL   LOAD      1E WOORD IN KENMERKREG.
      8 0D08  00101011
      9 0D09  00001100
173  10 0D0A  11000000          LAA
174  11 0D0B  01000110          CAL   ASS
     12 0D0C  00010010
     13 0D0D  00001010
175  14 0D0E  01100000          JTC   G6
     15 0D0F  00010011
     16 0D10  C0001101
176  17 0D11  11111111          HLT
177  18 0D12  11000000          LAA
178  19 0D13  11000000 G6       LAA
179  20 0D14  01000110          CAL   ADROUT
```

```
     21 0D15  01001101
     22 0D16  00001100
180  23 0D17  01000110          CAL   STORE
     24 0D18  00000000
     25 0D19  00001100
181  26 0D1A  00101110          LHI   B(TEKST3)
     27 0D1B  00010010
182  28 0D1C  00110110          LLI   A(TEKST3)
     29 0D1D  00000001
183  30 0D1E  01000110          CAL   TEKST      PRINT: 3 SCAM AANWEZIG?
     31 0D1F  00001101
     32 0D20  00001100
184  33 0D21  01000110          CAL   RELOAD
     34 0D22  00011111
     35 0D23  00001100
185  36 0D24  11000000          LAA
186  37 0D25  01000110          CAL   LOAD       2E WOORD IN KENMERKREG.
     38 0D26  00101011
     39 0D27  00001100
187  40 0D28  11000000          LAA
188  41 0D29  01000110          CAL   ASS
     42 0D2A  00010010
     43 0D2B  00001010
189  44 0D2C  01100000          JTC   G7
     45 0D2D  00110001
     46 0D2E  00001101
190  47 0D2F  11111111          HLT
191  48 0D30  11000000          LAA
192  49 0D31  11000000 G7       LAA
193  50 0D32  01000110          CAL   ADROUT
     51 0D33  01001101
     52 0D34  00001100
194  53 0D35  01000110          CAL   STORE
     54 0D36  00000000
     55 0D37  00001100
195  56 0D38  00101110          LHI   B(TEKST4)
     57 0D39  00010010
196  58 0D3A  00110110          LLI   A(TEKST4)
     59 0D3B  00010101
197  60 0D3C  01000110          CAL   TEKST      PRINT: 0 WCAM AANWEZIG?
     61 0D3D  00001101
     62 0D3E  00001100
198  63 0D3F  01000110          CAL   RELOAD
     64 0D40  00011111
     65 0D41  00001100
199  66 0D42  11000000          LAA
200  67 0D43  00001110          LBI   000B
     68 0D44  00000000
201  69 0D45  01000110          CAL   DMOUT      DON'T CARES OP MASKER-BUS
     70 0D46  11011101
     71 0D47  00001001
202  72 0D48  01000110          CAL   LCAM0      LAAD SCHUIFREG. 0
     73 0D49  01001110
     74 0D4A  00001001
203  75 0D4B  00001110          LBI   000B
     76 0D4C  00000000
204  77 0D4D  01000110          CAL   DMOUT      DON'T CARES OP MASKERBUS
     78 0D4E  11011101
     79 0D4F  00001001
205  80 0D50  01000110          CAL   LCAM1      LAAD SCHUIFREG. 1
```

```
     81 0D51  01010011
     82 0D52  00001001
206  83 0D53  01000110          CAL   DMOUT     DON'T CARES OP MASKERBUS
     84 0D54  11011101
     85 0D55  00001001
207  86 0D56  00001110          LBI   000B
     87 0D57  00000000
208  88 0D58  01000110          CAL   LCAM2     LAAD SCHUIFREG. 2
     89 0D59  01011000
     90 0D5A  00001001
209  91 0D5B  11000000          LAA
210  92 0D5C  00100110          LEI   000B
     93 0D5D  00000000
211  94 0D5E  11000000          LAA
212  95 0D5F  11000000 IN       LAA
213  96 0D60  01000110          CAL   ASS       ASS. OP DON'T CARES ?
     97 0D61  00010010
     98 0D62  00001010
214  99 0D63  01100000          JTC   G8
    100 0D64  01101000
    101 0D65  00001101
215 102 0D66  11111111          HLT
216 103 0D67  11000000          LAA
217 104 0D68  11000000 G8       LAA
218 105 0D69  01000110          CAL   ADROUT
    106 0D6A  01001101
    107 0D6B  00001100
219 108 0D6C  01000110          CAL   STORE
    109 0D6D  00000000
    110 0D6E  00001100
220 111 0D6F  00101110          LHI   B(TEKST5)
    112 0D70  00010010
221 113 0D71  00110110          LLI   A(TEKST5)
    114 0D72  00101001
222 115 0D73  01000110          CAL   TEKST     PRINT: 0 T/M 7 ASS NA TV-PULS
    116 0D74  00001101
    117 0D75  00001100
223 118 0D76  01000110          CAL   RELOAD
    119 0D77  00011111
    120 0D78  00001100
224 121 0D79  01000110          CAL   TVPULS
    122 0D7A  11001011
    123 0D7B  00001001
225 124 0D7C  11000000          LAA
226 125 0D7D  00100000          INE
227 126 0D7E  00000110          LAI   010B
    127 0D7F  00001000
228 128 0D80  10111100          CPE
229 129 0D81  01101000          JTZ   OUT
    130 0D82  10001001
    131 0D83  00001101
230 132 0D84  11000000          LAA
231 133 0D85  11000000          LAA
232 134 0D86  01000100          JMP   IN
    135 0D87  01011111
    136 0D88  00001101
233 137 0D89  01000110 OUT      CAL   ASS       ASS. NA T.V.-PULSEN
    138 0D8A  00010010
    139 0D8B  00001010
234 140 0D8C  01000000          JFC   OUTA
```

```
    141 0D8D  10010000
    142 0D8E  C0001101
235 143 0D8F  11111111          HLT
236 144 0D90  01000110 OUTA     CAL   TVRST
    145 0D91  11010100
    146 0D92  CC001001
237 147 0D93  11000000          LAA
238 148 0D94  11000C00          LAA
239 149 0D95  01000110          CAL   ASS        ASS. NA TV-RESET ?
    150 0D96  00010010
    151 0D97  00001010
240 152 0D98  01100000          JTC   OUTB
    153 0D99  10011101
    154 0D9A  C0001101
241 155 0D9B  11111111          HLT
242 156 0D9C  11000000          LAA
243 157 0D9D  01000110 OUTB     CAL   ADROUT
    158 0D9E  01001101
    159 0D9F  00001100
244 160 0DA0  01000110          CAL   STORE
    161 0DA1  00000000
    162 0DA2  C0001100
245 163 0DA3  00101110          LHI   B(TEKST6)
    164 0DA4  00010010
246 165 0DA5  00110110          LLI    A(TEKST6)
    166 0DA6  00111100
247 167 0DA7  01000110          CAL   TEKST      PRINT: 0 ASS NA TV-RST
    168 0DA8  C0001101
    169 0DA9  00001100
248 170 0DAA  01000110          CAL   RELOAD
    171 0DAB  00011111
    172 0DAC  C0001100
249 173 0DAD  11000000          LAA
250 174 0DAE  11000000          LAA
251 175 0DAF            *
252 175 0DAF            *               WISSEN VAN CAM
253 175 0DAF            *
254 175 0DAF  00000110          LAI   003B
    176 0DB0  00000011
255 177 0DB1  01000110          CAL   DMOUT
    178 0DB2  11011101
    179 0DB3  C0001001
256 180 0DB4  01000110          CAL   RSTCAM     WIS CAM EXTERN ADRES
    181 0DB5  10110111
    182 0DB6  00001001
257 183 0DB7  11000000          LAA
258 184 0DB8  00110110          LLI   000B
    185 0DB9  00000000
259 186 0DBA  01000110          CAL   LOAD       1E WOORD IN KENMERKREG.
    187 0DBB  00101011
    188 0DBC  00001100
260 189 0DBD  11000000          LAA
261 190 0DBE  01000110          CAL   ASS
    191 0DBF  00010010
    192 0DC0  00001010
262 193 0DC1  01000000          JFC   G9
    194 0DC2  11000110
    195 0DC3  00001101
263 196 0DC4  11111111          HLT
264 197 0DC5  11000000          LAA
```

```
265 198 0DC6  11000000 G9        LAA
266 199 0DC7  00110110           LLI   006B
    200 0DC8  00000110
267 201 0DC9  01000110           CAL   LOAD       3E WOORD IN KENMERKREG.
    202 0DCA  00101011
    203 0DCB  C0001100
268 204 0DCC  11000000           LAA
269 205 0DCD  01000110           CAL   ASS
    206 0DCE  00010010
    207 0DCF  00001010
270 208 0DD0  01100000           JTC   G10
    209 0DD1  11010101
    210 0DD2  00001101
271 211 0DD3  11111111           HLT
272 212 0DD4  11000000           LAA
273 213 0DD5  11000000 G10       LAA
274 214 0DD6  01000110           CAL   CLCAM      WIS CAM OP ASSOCIATIE
    215 0DD7  10111100
    216 0DD8  00001001
275 217 0DD9  11000000           LAA
276 218 0DDA  01000110           CAL   ASS
    219 0DDB  00010010
    220 0DDC  00001010
277 221 0DDD  01000000           JFC   G11
    222 0DDE  11100010
    223 0DDF  C0001101
278 224 0DE0  11111111           HLT
279 225 0DE1  11000000           LAA
280 226 0DE2  11000000 G11       LAA
281 227 0DE3  00110110           LLI   003B
    228 0DE4  00000011
282 229 0DE5  01000110           CAL   LOAD       2E WOORD IN KENMERKREG.
    230 0DE6  00101011
    231 0DE7  C0001100
283 232 0DE8  11000000           LAA
284 233 0DE9  01000110           CAL   ASS
    234 0DEA  00010010
    235 0DEB  C0001010
285 236 0DEC  01100000           JTC   G12
    237 0DED  11110001
    238 0DEE  00001101
286 239 0DEF  11111111           HLT
287 240 0DF0  11000000           LAA
288 241 0DF1  11000000 G12       LAA
289 242 0DF2  01000110           CAL   ADROUT
    243 0DF3  01001101
    244 0DF4  00001100
290 245 0DF5  01000110           CAL   STORE
    246 0DF6  00000000
    247 0DF7  C0001100
291 248 0DF8  00101110           LHI   B(TEKST4)
    249 0DF9  C0010010
292 250 0DFA  00110110           LLI   A(TEKST4)
    251 0DFB  C0010101
293 252 0DFC  01000110           CAL   TEKST      PRINT: 0 WCAM AANWEZIG?
    253 0DFD  C0001101
    254 0DFE  00001100
294 255 0DFF  01000110           CAL   RELOAD
      0 0E00  00011111
      1 0E01  C0001100
```

```
295   2 0E02  11000000          LAA
296   3 0E03  01000110          CAL   CLCAM     WIS CAM OP ASSOCIATIE
      4 0E04  10111100
      5 0E05  00001001
297   6 0E06  01000110          CAL   ASS
      7 0E07  00010010
      8 0E08  00001010
298   9 0E09  01000000          JFC   G13
     10 0E0A  00001110
     11 0E0B  00001110
299  12 0E0C  11111111          HLT
300  13 0E0D  11000000          LAA
301  14 0E0E  11000000 G13      LAA
302  15 0E0F  00001110          LBI   000B
     16 0E10  00000000
303  17 0E11  01000110          CAL   DMOUT     DON'T CARES OP MASKER-BUS
     18 0E12  11011101
     19 0E13  00001001
304  20 0E14  01000110          CAL   LCAMV0    LAAD SCHUIFREG.0,1 EN 2
     21 0E15  01101100
     22 0E16  00001001
305  23 0E17  01000110          CAL   ASS
     24 0E18  00010010
     25 0E19  00001010
306  26 0E1A  01000000          JFC   G14
     27 0E1B  11111111
     28 0E1C  11111111
307  29 0E1D  11111111          HLT
308  30 0E1E  11000000          LAA
309  31 0E1F            *
310  31 0E1F            *               LADEN CAMREG. VANAF BEP. BYTE
311  31 0E1F            *
312  31 0E1F  00000110          LAI   160B
     32 0E20  01110000
313  33 0E21  00001110          LBI   377B
     34 0E22  11111111
314  35 0E23  01000110          CAL   DMOUT
     36 0E24  11011101
     37 0E25  00001001
315  38 0E26  01000110          CAL   LCAMV0    LAAD SCHUIFREG.0,1 EN 2
     39 0E27  01101100
     40 0E28  00001001
316  41 0E29  11000000          LAA
317  42 0E2A  01000110          CAL   WCAM      SCHRIJF CAM 1E LEGE PL.
     43 0E2B  10011001
     44 0E2C  00001001
318  45 0E2D  00000110          LAI   000B
     46 0E2E  00000000
319  47 0E2F  00001110          LBI   377B
     48 0E30  11111111
320  49 0E31  01000110          CAL   DMOUT
     50 0E32  11011101
     51 0E33  00001001
321  52 0E34  01000110          CAL   LCAM0     LAAD SCHUIFREG. 0
     53 0E35  01001110
     54 0E36  00001001
322  55 0E37  11000000          LAA
323  56 0E38  00000110          LAI   217B
     57 0E39  10001111
324  58 0E3A  00001110          LBI   377B
```

```
     59 0E3B  11111111
325  60 0E3C  01000110          CAL   DMOUT
     61 0E3D  11011101
     62 0E3E  C0001001
326  63 0E3F  01000110          CAL   LCAMV1    LAAD SCHUIFREG. 1 EN 2
     64 0E40  01110001
     65 0E41  00001001
327  66 0E42  11000000          LAA
328  67 0E43  01000110          CAL   WCAM      SCHRIJF CAM 1E LEGE PL.
     68 0E44  10011001
     69 0E45  00001001
329  70 0E46  11000000          LAA
330  71 0E47  00000110          LAI   160B
     72 0E48  01110000
331  73 0E49  00001110          LBI   377B
     74 0E4A  11111111
332  75 0E4B  01000110          CAL   DMOUT
     76 0E4C  11011101
     77 0E4D  00001001
333  78 0E4E  01000110          CAL   LCAM0     LAAD SCHUIFREG. 0
     79 0E4F  01001110
     80 0E50  C0001001
334  81 0E51  00000110          LAI   160B
     82 0E52  01110000
335  83 0E53  00001110          LBI   377B
     84 0E54  11111111
336  85 0E55  01000110          CAL   DMOUT
     86 0E56  11011101
     87 0E57  00001001
337  88 0E58  01000110          CAL   LCAM1     LAAD SCHUIFREG. 1
     89 0E59  01010011
     90 0E5A  00001001
338  91 0E5B  00000110          LAI   160B
     92 0E5C  01110000
339  93 0E5D  00001110          LBI   377B
     94 0E5E  11111111
340  95 0E5F  01000110          CAL   DMOUT
     96 0E60  11011101
     97 0E61  00001001
341  98 0E62  01000110          CAL   LCAM2     LAAD SCHUIFREG. 2
     99 0E63  01011000
    100 0E64  C0001001
342 101 0E65  11000000          LAA
343 102 0E66  01000110          CAL   ASS       ASS. DATA I ?
    103 0E67  00010010
    104 0E68  C0001010
344 105 0E69  01100000          JTC   G15
    106 0E6A  01101101
    107 0E6B  00001110
345 108 0E6C  11111111          HLT
346 109 0E6D  11000000 G15      LAA
347 110 0E6E  01000110          CAL   ADROUT
    111 0E6F  01001101
    112 0E70  C0001100
348 113 0E71  01000110          CAL   STORE
    114 0E72  00000000
    115 0E73  00001100
349 116 0E74  00101110          LHI   B(TEKST7)
    117 0E75  00010010
350 118 0E76  00110110          LLI   A(TEKST7)
```

```
    119 0E77  01001101
351 120 0E78  01000110          CAL   TEKST     PRINT: 0 WCAM NA LCAMV0
    121 0E79  00001101
    122 0E7A  00001100
352 123 0E7B  01000110          CAL   RELOAD
    124 0E7C  00011111
    125 0E7D  00001100
353 126 0E7E  11000000          LAA
354 127 0E7F  01000110          CAL   TVPULS
    128 0E80  11001011
    129 0E81  00001001
355 130 0E82  01000110          CAL   ASS       ASS. NA TVPULS ?
    131 0E83  00010010
    132 0E84  00001010
356 133 0E85  01000000          JFC   G16
    134 0E86  10001010
    135 0E87  00001110
357 136 0E88  11111111          HLT
358 137 0E89  11000000          LAA
359 138 0E8A  01000110 G16      CAL   TVRST
    139 0E8B  11010100
    140 0E8C  00001001
360 141 0E8D  01000110          CAL   CLCAM     WIS CAM OP ASSOCIATIE
    142 0E8E  10111100
    143 0E8F  00001001
361 144 0E90  11000000          LAA
362 145 0E91  11000000          LAA
363 146 0E92  00000110          LAI   000B
    147 0E93  00000000
364 148 0E94  00001110          LBI   377B
    149 0E95  11111111
365 150 0E96  01000110          CAL   DMOUT
    151 0E97  11011101
    152 0E98  00001001
366 153 0E99  01000110          CAL   LCAM0     LAAD SCHUIFREG. 0
    154 0E9A  01001110
    155 0E9B  00001001
367 156 0E9C  11000000          LAA
368 157 0E9D  00000110          LAI   217B
    158 0E9E  10001111
369 159 0E9F  00001110          LBI   377B
    160 0EA0  11111111
370 161 0EA1  01000110          CAL   DMOUT
    162 0EA2  11011101
    163 0EA3  00001001
371 164 0EA4  01000110          CAL   LCAM1     LAAD SCHUIFREG. 1
    165 0EA5  01010011
    166 0EA6  00001001
372 167 0EA7  00000110          LAI   217B
    168 0EA8  10001111
373 169 0EA9  00001110          LBI   377B
    170 0EAA  11111111
374 171 0EAB  01000110          CAL   DMOUT
    172 0EAC  11011101
    173 0EAD  00001001
375 174 0EAE  01000110          CAL   LCAM2     LAAD SCHUIFREG. 2
    175 0EAF  01011000
    176 0EB0  00001001
376 177 0EB1  11000000          LAA
377 178 0EB2  01000110          CAL   ASS
```

```
    179 0EB3  00010010
    180 0EB4  00001010
378 181 0EB5  01100000          JTC   G17
    182 0EB6  10111010
    183 0EB7  00001110
379 184 0EB8  11111111          HLT
380 185 0EB9  11000000          LAA
381 186 0EBA  11000000 G17      LAA
382 187 0EBB  01000110          CAL   ADROUT
    188 0EBC  01001101
    189 0EBD  00001100
383 190 0EBE  01000110          CAL   STORE
    191 0EBF  00000000
    192 0EC0  00001100
384 193 0EC1  00101110          LHI   B(TEKST8)
    194 0EC2  00010010
385 195 0EC3  00110110          LLI   A(TEKST8)
    196 0EC4  01100000
386 197 0EC5  01000110          CAL   TEKST     PRINT: 1 WCAM NA LCAMV1
    198 0EC6  00001101
    199 0EC7  00001100
387 200 0EC8  01000110          CAL   RELOAD
    201 0EC9  00011111
    202 0ECA  00001100
388 203 0ECB  11000000          LAA
389 204 0ECC  01000110          CAL   CLCAM     WIS CAM OP ASSOCIATIE
    205 0ECD  10111100
    206 0ECE  00001001
390 207 0ECF  11000000          LAA
391 208 0ED0  00000110          LAI   125B
    209 0ED1  01010101
392 210 0ED2  00001110          LBI   377B
    211 0ED3  11111111
393 212 0ED4  01000110          CAL   DMOUT
    213 0ED5  11011101
    214 0ED6  00001001
394 215 0ED7  01000110          CAL   LCAMV0    LAAD SCHUIFREG.0,1 EN 2
    216 0ED8  01101100
    217 0ED9  00001001
395 218 0EDA  11000000          LAA
396 219 0EDB  11000000          LAA
397 220 0EDC  01000110          CAL   ASS
    221 0EDD  00010010
    222 0EDE  00001010
398 223 0EDF  01000000          JFC   G18
    224 0EE0  11100100
    225 0EE1  00001110
399 226 0EE2  11111111          HLT
400 227 0EE3  11000000          LAA
401 228 0EE4  11000000 G18      LAA
402 229 0EE5  01000110          CAL   WCAM      SCHRIJF CAM 1E LEGE PL.
    230 0EE6  10011001
    231 0EE7  00001001
403 232 0EE8  00100110          LEI   000B
    233 0EE9  00000000
404 234 0EEA  00000110 BEGIN    LAI   125B
    235 0EEB  01010101
405 236 0EEC  00001110          LBI   377B
    237 0EED  11111111
406 238 0EEE  01000110          CAL   DMOUT
```

```
    239 0EEF  11011101
    240 0EF0  C0001001
407 241 0EF1  01000110          CAL   LCAMV0    LAAD SCHUIFREG.0,1 EN 2
    242 0EF2  01101100
    243 0EF3  00001001
408 244 0EF4  11000000          LAA
409 245 0EF5  01000110          CAL   ASS
    246 0EF6  C0010010
    247 0EF7  00001010
410 248 0EF8  01100000          JTC   G19
    249 0EF9  11111101
    250 0EFA  C0001110
411 251 0EFB  11111111          HLT
412 252 0EFC  11000000          LAA
413 253 0EFD  11000000 G19      LAA
414 254 0EFE  11000100          LAE
415 255 0EFF  01000110          CAL   DMOUT
      0 0F00  11011101
      1 0F01  00001001
416   2 0F02  01000110          CAL   SHIFT     SCHUIF KENMERK E+1 PL.
      3 0F03  01111011
      4 0F04  C0001001
417   5 0F05  11000000          LAA
418   6 0F06  11000000          LAA
419   7 0F07  01000110          CAL   ASS
      8 0F08  C0010010
      9 0F09  00001010
420  10 0F0A  01000000          JFC   FALSE
     11 0F0B  00100001
     12 0F0C  C0001111
421  13 0F0D  11000100          LAE
422  14 0F0E  00000100          ADI   260B
     15 0F0F  10110000
423  16 0F10  11001000          LBA
424  17 0F11  01000110          CAL   TTYOUT
     18 0F12  10100001
     19 0F13  00000001
425  20 0F14  01000110          CAL   STORE
     21 0F15  00000000
     22 0F16  C0001100
426  23 0F17  00101110          LHI   B(TEKST9)
     24 0F18  00010010
427  25 0F19  00110110          LLI   A(TEKST9)
     26 0F1A  01110011
428  27 0F1B  01000110          CAL   TEKST     PRINT: 1/3/5/7 ASS NA SHIFT
     28 0F1C  C0001101
     29 0F1D  00001100
429  30 0F1E  01000110          CAL   RELOAD
     31 0F1F  00011111
     32 0F20  C0001100
430  33 0F21  00100000 FALSE    INE
431  34 0F22  00000110          LAI   010B
     35 0F23  00001000
432  36 0F24  10111100          CPE
433  37 0F25  01001000          JFZ   BEGIN
     38 0F26  11101010
     39 0F27  00001110
434  40 0F28  11111111          HLT
435  41 0F29            *
436  41 0F29            *        TESTPROGRAMMA RAM.
```

```
437  41 0F29            *
438  41 0F29  01000110          CAL   START
     42 0F2A  00100000
     43 0F2B  C0001010
439  44 0F2C  00101110          LHI   11
     45 0F2D  00001011
440  46 0F2E  00110110          LLI   0
     47 0F2F  00000000
441  48 0F30  00000110          LAI   000B
     49 0F31  00000000
442  50 0F32  01000110          CAL   STOREA    BYTE 0 IN GEH. EN OP DATA-BUS
     51 0F33  01011110
     52 0F34  00001100
443  53 0F35  01000110          CAL   LRAM0     LAAD RAM INPUTREG. 0
     54 0F36  00111111
     55 0F37  C0001001
444  56 0F38  11000000          LAA
445  57 0F39  C0000110          LAI   377B
     58 0F3A  11111111
446  59 0F3B  01000110          CAL   STOREA    BYTE 1 IN GEH. EN OP DATA-BUS
     60 0F3C  01011110
     61 0F3D  C0001100
447  62 0F3E  01000110          CAL   LRAM1     LAAD RAM INPUTREG. 1
     63 0F3F  01000100
     64 0F40  00001001
448  65 0F41  11000000          LAA
449  66 0F42  00000110          LAI   125B
     67 0F43  01010101
450  68 0F44  01000110          CAL   STOREA    BYTE 2 IN GEH. EN OP DATA-BUS
     69 0F45  01011110
     70 0F46  00001100
451  71 0F47  01000110          CAL   LRAM2     LAAD RAM INPUTREG. 2
     72 0F48  01001001
     73 0F49  C0001001
452  74 0F4A  11000000          LAA
453  75 0F4B  00000110          LAI   001B
     76 0F4C  00000001
454  77 0F4D  01000110          CAL   DMOUT     ADRES OP DATA-BUS
     78 0F4E  11011101
     79 0F4F  C0001001
455  80 0F50  01000110          CAL   SRAM      SCHRIJF RAM OP ADRES 1
     81 0F51  10101101
     82 0F52  00001001
456  83 0F53  11000000          LAA
457  84 0F54  00110110          LLI   0
     85 0F55  00000000
458  86 0F56  01000110          CAL   ORAM0     BYTE 0 RAM UIT
     87 0F57  10000101
     88 0F58  00001001
459  89 0F59  01000110          CAL   COMP      VERGELIJK DATA
     90 0F5A  01100101
     91 0F5B  C0001100
460  92 0F5C  01101000          JTZ   SB11
     93 0F5D  01100000
     94 0F5E  00001111
461  95 0F5F  11111111          HLT
462  96 0F60  01000110 SB11     CAL   ORAM1     BYTE 1 RAM UIT
     97 0F61  10001010
     98 0F62  00001001
463  99 0F63  01000110          CAL   COMP      VERGELIJK DATA
```

```
    100 0F64  01100101
    101 0F65  C0001100
464 102 0F66  01101000          JTZ   SB12
    103 0F67  01101010
    104 0F68  00001111
465 105 0F69  11111111          HLT
466 106 0F6A  01000110 SB12     CAL   ORAM2     BYTE 2 RAM UIT
    107 0F6B  10001111
    108 0F6C  00001001
467 109 0F6D  01000110          CAL   COMP      VERGELIJK DATA
    110 0F6E  01100101
    111 0F6F  C0001100
468 112 0F70  01101000          JTZ   SW4
    113 0F71  01110100
    114 0F72  00001111
469 115 0F73  11111111          HLT
470 116 0F74  01000110 SW4      CAL   STORE
    117 0F75  00000000
    118 0F76  00001100
471 119 0F77  00101110          LHI   B(TKST10)
    120 0F78  00010010
472 121 0F79  00110110          LLI   A(TKST10)
    122 0F7A  10000100
473 123 0F7B  01000110          CAL   TEKST     PRINT: SRAM OKE
    124 0F7C  00001101
    125 0F7D  C0001100
474 126 0F7E  01000110          CAL   RELOAD
    127 0F7F  C0011111
    128 0F80  00001100
475 129 0F81  00000110          LAI   125B
    130 0F82  01010101
476 131 0F83  01000110          CAL   STOREA    BYTE 0 IN GEH. EN OP DATA-BUS
    132 0F84  01011110
    133 0F85  C0001100
477 134 0F86  01000110          CAL   LRAM0     LAAD RAM INPUTREG. 0
    135 0F87  00111111
    136 0F88  00001001
478 137 0F89  11000000          LAA
479 138 0F8A  00000110          LAI   252B
    139 0F8B  10101010
480 140 0F8C  01000110          CAL   STOREA    BYTE 1 IN GEH. EN OP DATA-BUS
    141 0F8D  01011110
    142 0F8E  00001100
481 143 0F8F  01000110          CAL   LRAM1     LAAD RAM INPUTREG. 1
    144 0F90  01000100
    145 0F91  C0001001
482 146 0F92  11000000          LAA
483 147 0F93  00000110          LAI   063B
    148 0F94  00110011
484 149 0F95  01000110          CAL   STOREA    BYTE 2 IN GEH. EN OP DATA-BUS
    150 0F96  01011110
    151 0F97  C0001100
485 152 0F98  01000110          CAL   LRAM2     LAAD RAM INPUTREG. 2
    153 0F99  01001001
    154 0F9A  00001001
486 155 0F9B  11000000          LAA
487 156 0F9C  00000110          LAI   004B
    157 0F9D  00000100
488 158 0F9E  01000110          CAL   DMOUT     ADRES OP DATA-BUS
    159 0F9F  11011101
```

```
    160 0FA0  00001001
489 161 0FA1  01000110          CAL   SRAM      SCHRIJF RAM OP ADRES 4
    162 0FA2  10101101
    163 0FA3  00001001
490 164 0FA4  11000000          LAA
491 165 0FA5  00000110          LAI   001B
    166 0FA6  00000001
492 167 0FA7  01000110          CAL   DMOUT     ADRES OP DATA-BUS
    168 0FA8  11011101
    169 0FA9  00001001
493 170 0FAA  01000110          CAL   RARAM     LEES RAM OP ADRES 1
    171 0FAB  11000110
    172 0FAC  00001001
494 173 0FAD  11000000          LAA
495 174 0FAE  00110110          LLI   0
    175 0FAF  00000000
496 176 0FB0  01000110          CAL   ORAM0     BYTE 0 RAM UIT
    177 0FB1  10000101
    178 0FB2  00001001
497 179 0FB3  01000110          CAL   COMP      VERGELIJK DATA
    180 0FB4  01100101
    181 0FB5  00001100
498 182 0FB6  01101000          JTZ   RAB11
    183 0FB7  10111010
    184 0FB8  00001111
499 185 0FB9  11111111          HLT
500 186 0FBA  01000110 RAB11    CAL   ORAM1     BYTE 1 RAM UIT
    187 0FBB  10001010
    188 0FBC  00001001
501 189 0FBD  01000110          CAL   COMP      VERGELIJK DATA
    190 0FBE  01100101
    191 0FBF  00001100
502 192 0FC0  01101000          JTZ   RAB12
    193 0FC1  11000100
    194 0FC2  00001111
503 195 0FC3  11111111          HLT
504 196 0FC4  01000110 RAB12    CAL   ORAM2     BYTE 2 RAM UIT
    197 0FC5  10001111
    198 0FC6  00001001
505 199 0FC7  01000110          CAL   COMP      VERGELIJK DATA
    200 0FC8  01100101
    201 0FC9  00001100
506 202 0FCA  01101000          JTZ   RAW4
    203 0FCB  11001110
    204 0FCC  00001111
507 205 0FCD  11111111          HLT
508 206 0FCE  01000110 RAW4     CAL   STORE
    207 0FCF  00000000
    208 0FD0  00001100
509 209 0FD1  00101110          LHI   B(TKST11)
    210 0FD2  00010010
510 211 0FD3  00110110          LLI   A(TKST11)
    212 0FD4  10010011
511 213 0FD5  01000110          CAL   TEKST     PRINT: RARAM NA SRAM
    214 0FD6  00001101
    215 0FD7  00001100
512 216 0FD8  01000110          CAL   RELOAD
    217 0FD9  00011111
    218 0FDA  00001100
513 219 0FDB  00000110          LAI   004B
```

```
    220 0FDC  00000100
514 221 0FDD  01000110          CAL   DMOUT     ADRES OP DATA-BUS
    222 0FDE  11011101
    223 0FDF  C0001001
515 224 0FE0  01000110          CAL   RARAM     LEES RAM OP ADRES 4
    225 0FE1  11000110
    226 0FE2  00001001
516 227 0FE3  11000000          LAA
517 228 0FE4  01000110          CAL   ORAM0     BYTE 0 RAM UIT
    229 0FE5  1C000101
    230 0FE6  00001001
518 231 0FE7  01000110          CAL   COMP      VERGELIJK DATA
    232 0FE8  01100101
    233 0FE9  C0001100
519 234 0FEA  01101000          JTZ   RAB41
    235 0FEB  11101110
    236 0FEC  00001111
520 237 0FED  11111111          HLT
521 238 0FEE  01000110 RAB41    CAL   ORAM1     BYTE 1 RAM UIT
    239 0FEF  1C001010
    240 0FF0  00001001
522 241 0FF1  01000110          CAL   COMP      VERGELIJK DATA
    242 0FF2  01100101
    243 0FF3  C0001100
523 244 0FF4  01101000          JTZ   RAB42
    245 0FF5  11111000
    246 0FF6  00001111
524 247 0FF7  11111111          HLT
525 248 0FF8  01000110 RAB42    CAL   ORAM2     BYTE 2 RAM UIT
    249 0FF9  10001111
    250 0FFA  00001001
526 251 0FFB  01000110          CAL   COMP      VERGELIJK DATA
    252 0FFC  01100101
    253 0FFD  C0001100
527 254 0FFE  01101000          JTZ   CRAM
    255 0FFF  00000010
      0 1000  00010000
528   1 1001  11111111          HLT
529   2 1002  01000110 CRAM     CAL   STORE
      3 1003  00000000
      4 1004  00001100
530   5 1005  00101110          LHI   B(TKST11)
      6 1006  00010010
531   7 1007  00110110          LLI   A(TKST11)
      8 1008  10010011
532   9 1009  01000110          CAL   TEKST     PRINT: RARAM NA SRAM
     10 100A  00001101
     11 100B  C0001100
533  12 100C  01000110          CAL   RELOAD
     13 100D  00011111
     14 100E  00001100
534  15 100F  C0000110          LAI   377B
     16 1010  11111111
535  17 1011  C0001110          LBI   377B
     18 1012  11111111
536  19 1013  01000110          CAL   STOREA    BYTE 0 IN GEH. EN OP DATA-BUS
     20 1014  01011110
     21 1015  C0001100
537  22 1016  01000110          CAL   LCRAM0    LAAD SCHUIFREG.0 EN INPUTREG.0
     23 1017  01011101
```

```
     24 1018  00001001
538  25 1019  11000000          LAA
539  26 101A  00000110          LAI   000B
     27 101B  00000000
540  28 101C  00001110          LBI   377B
     29 101D  11111111
541  30 101E  01000110          CAL   STOREA    BYTE 1 IN GEH. EN OP DATA-BUS
     31 101F  01011110
     32 1020  00001100
542  33 1021  01000110          CAL   LCRAM1    LAAD SCHUIFREG.1 EN INPUTREG.1
     34 1022  01100010
     35 1023  00001001
543  36 1024  11000000          LAA
544  37 1025  00000110          LAI   252B
     38 1026  10101010
545  39 1027  00001110          LBI   377B
     40 1028  11111111
546  41 1029  01000110          CAL   STOREA    BYTE 2 IN GEH. EN OP DATA-BUS
     42 102A  01011110
     43 102B  00001100
547  44 102C  01000110          CAL   LCRAM2    LAAD SCHUIFREG.2 EN INPUTREG.2
     45 102D  01100111
     46 102E  00001001
548  47 102F  11000000          LAA
549  48 1030  00000110          LAI   005B
     49 1031  00000101
550  50 1032  01000110          CAL   DMOUT     ADRES OP DATA-BUS
     51 1033  11011101
     52 1034  00001001
551  53 1035  01000110          CAL   SCRAM     SCHRIJF CAM EN RAM OP ADRES 5
     54 1036  10110010
     55 1037  00001001
552  56 1038  01000110          CAL   ADROUT
     57 1039  01001101
     58 103A  00001100
553  59 103B  01000110          CAL   STORE
     60 103C  00000000
     61 103D  00001100
554  62 103E  00101110          LHI   B(TKST12)
     63 103F  00010010
555  64 1040  00110110          LLI   A(TKST12)
     65 1041  10101000
556  66 1042  01000110          CAL   TEKST     PRINT: 5  ASS SCRAM
     67 1043  00001101
     68 1044  00001100
557  69 1045  01000110          CAL   RELOAD
     70 1046  00011111
     71 1047  00001100
558  72 1048  00000110          LAI   005B
     73 1049  00000101
559  74 104A  01000110          CAL   DMOUT     ADRES OP DATA-BUS
     75 104B  11011101
     76 104C  00001001
560  77 104D  01000110          CAL   RARAM     LEES RAM OP ADRES 5
     78 104E  11000110
     79 104F  00001001
561  80 1050  00110110          LLI   6
     81 1051  00000110
562  82 1052  01000110          CAL   ORAM0     BYTE 0 RAM UIT
     83 1053  10000101
```

```
     84 1054  00001001
563  85 1055  01000110          CAL   COMP      VERGELIJK DATA
     86 1056  01100101
     87 1057  C0001100
564  88 1058  01101000          JTZ   RAB51
     89 1059  01011100
     90 105A  00010000
565  91 105B  11111111          HLT
566  92 105C  01000110 RAB51    CAL   ORAM1     BYTE 1 RAM UIT
     93 105D  10001010
     94 105E  00001001
567  95 105F  01000110          CAL   COMP      VERGELIJK DATA
     96 1060  01100101
     97 1061  C0001100
568  98 1062  01101000          JTZ   RAB52
     99 1063  01100110
    100 1064  00010000
569 101 1065  11111111          HLT
570 102 1066  01000110 RAB52    CAL   ORAM2     BYTE 2 RAM UIT
    103 1067  10001111
    104 1068  00001001
571 105 1069  01000110          CAL   COMP      VERGELIJK DATA
    106 106A  01100101
    107 106B  C0001100
572 108 106C  01101000          JTZ   WCAM6
    109 106D  01110000
    110 106E  00010000
573 111 106F  11111111          HLT
574 112 1070  01000110 WCAM6    CAL   STORE
    113 1071  C0000000
    114 1072  00001100
575 115 1073  00101110          LHI   B(TKST13)
    116 1074  00010010
576 117 1075  00110110          LLI   A(TKST13)
    118 1076  10110111
577 119 1077  01000110          CAL   TEKST     PRINT: RARAM NA SCRAM
    120 1078  00001101
    121 1079  C0001100
578 122 107A  01000110          CAL   RELOAD
    123 107B  C0011111
    124 107C  00001100
579 125 107D  C0000110          LAI   063B
    126 107E  00110011
580 127 107F  C0001110          LBI   377B
    128 1080  11111111
581 129 1081  01000110          CAL   STOREA    BYTE 0 IN GEH. EN OP DATA-BUS
    130 1082  01011110
    131 1083  C0001100
582 132 1084  01000110          CAL   LCRAM0    LAAD SCHUIFREG.0 EN INPUTREG.0
    133 1085  01011101
    134 1086  00001001
583 135 1087  11000000          LAA
584 136 1088  00000110          LAI   314B
    137 1089  11001100
585 138 108A  00001110          LBI   377B
    139 108B  11111111
586 140 108C  01000110          CAL   STOREA    BYTE 1 IN GEH. EN OP DATA-BUS
    141 108D  01011110
    142 108E  00001100
587 143 108F  01000110          CAL   LCRAM1    LAAD SCHUIFREG.1 EN INPUTREG.1
```

```
    144 1090  01100010
    145 1091  C0001001
588 146 1092  11000000          LAA
589 147 1093  C0000110          LAI   017B
    148 1094  00001111
590 149 1095  C0001110          LBI   377B
    150 1096  11111111
591 151 1097  01000110          CAL   STOREA    BYTE 2 IN GEH. EN OP DATA-BUS
    152 1098  01011110
    153 1099  00001100
592 154 109A  01000110          CAL   LCRAM2    LAAD SCHUIFREG.2 EN INPUTREG.2
    155 109B  01100111
    156 109C  00001001
593 157 109D  11000000          LAA
594 158 109E  00000110          LAI   006B
    159 109F  00000110
595 160 10A0  01000110          CAL   DMOUT     ZET ADRES OP DATA-BUS
    161 10A1  11011101
    162 10A2  00001001
596 163 10A3  01000110          CAL   SCAM      SCHRIJF CAM OP ADRES 6
    164 10A4  10101000
    165 10A5  C0001001
597 166 10A6  01000110          CAL   WRAM      SCHRIJF RAM OP ASS.-ADRES
    167 10A7  10011110
    168 10A8  00001001
598 169 10A9  00000110          LAI   006B
    170 10AA  00000110
599 171 10AB  01000110          CAL   DMOUT     ADRES OP DATA-BUS
    172 10AC  11011101
    173 10AD  C0001001
600 174 10AE  01000110          CAL   RARAM     LEES RAM OP ADRES 6
    175 10AF  11000110
    176 10B0  00001001
601 177 10B1  00110110          LLI   9
    178 10B2  00001001
602 179 10B3  01000110          CAL   ORAM0     BYTE 0 RAM UIT
    180 10B4  10000101
    181 10B5  C0001001
603 182 10B6  01000110          CAL   COMP      VERGELIJK DATA
    183 10B7  01100101
    184 10B8  00001100
604 185 10B9  01101000          JTZ   RAB61
    186 10BA  10111101
    187 10BB  00010000
605 188 10BC  11111111          HLT
606 189 10BD  01000110 RAB61    CAL   ORAM1     BYTE 1 RAM UIT
    190 10BE  10001010
    191 10BF  C0001001
607 192 10C0  01000110          CAL   COMP      VERGELIJK DATA
    193 10C1  01100101
    194 10C2  00001100
608 195 10C3  01101000          JTZ   RAB62
    196 10C4  11000111
    197 10C5  00010000
609 198 10C6  11111111          HLT
610 199 10C7  01000110 RAB62    CAL   ORAM2     BYTE 2 RAM UIT
    200 10C8  10001111
    201 10C9  C0001001
611 202 10CA  01000110          CAL   COMP      VERGELIJK DATA
    203 10CB  01100101
```

```
    204 10CC  00001100
612 205 10CD  01101000          JTZ   WCRAM0
    206 10CE  11010001
    207 10CF  00010000
613 208 10D0  11111111          HLT
614 209 10D1  01000110 WCRAM0   CAL   STORE
    210 10D2  00000000
    211 10D3  C0001100
615 212 10D4  00101110          LHI   B(TKST14)
    213 10D5  C0010010
616 214 10D6  00110110          LLI   A(TKST14)
    215 10D7  11001101
617 216 10D8  01000110          CAL   TEKST     PRINT: RARAM NA WRAM
    217 10D9  C0001101
    218 10DA  00001100
618 219 10DB  01000110          CAL   RELOAD
    220 10DC  00011111
    221 10DD  00001100
619 222 10DE  00000110          LAI   041B
    223 10DF  00100001
620 224 10E0  00001110          LBI   377B
    225 10E1  11111111
621 226 10E2  01000110          CAL   STOREA    BYTE 0 IN GEH. EN OP DATA-BUS
    227 10E3  01011110
    228 10E4  00001100
622 229 10E5  01000110          CAL   LCRAM0    LAAD SCHUIFREG.0 EN INPUTREG.0
    230 10E6  01011101
    231 10E7  C0001001
623 232 10E8  11000000          LAA
624 233 10E9  00000110          LAI   343B
    234 10EA  11100011
625 235 10EB  C0001110          LBI   377B
    236 10EC  11111111
626 237 10ED  01000110          CAL   STOREA    BYTE 1 IN GEH. EN OP DATA-BUS
    238 10EE  01011110
    239 10EF  C0001100
627 240 10F0  01000110          CAL   LCRAM1    LAAD SCHUIFREG.1 EN INPUTREG.1
    241 10F1  01100010
    242 10F2  00001001
628 243 10F3  11000000          LAA
629 244 10F4  00000110          LAI   245B
    245 10F5  10100101
630 246 10F6  00001110          LBI   377B
    247 10F7  11111111
631 248 10F8  01000110          CAL   STOREA    BYTE 2 IN GEH. EN OP DATA-BUS
    249 10F9  01011110
    250 10FA  00001100
632 251 10FB  01000110          CAL   LCRAM2    LAAD SCHUIFREG.2 EN INPUTREG.2
    252 10FC  01100111
    253 10FD  C0001001
633 254 10FE  01000110          CAL   WCRAM     SCHRIJF CRAM OP 1-STE LEGE PLAA
    255 10FF  10100011
      0 1100  00001001
634   1 1101  01000110          CAL   ADROUT
      2 1102  01001101
      3 1103  00001100
635   4 1104  01000110          CAL   STORE
      5 1105  00000000
      6 1106  00001100
636   7 1107  00101110          LHI   B(TKST15)
```

```
      8 1108  00010010
637   9 1109  00110110          LLI   A(TKST15)
     10 110A  11100010
638  11 110B  01000110          CAL   TEKST      PRINT: 0  ASS WCRAM
     12 110C  00001101
     13 110D  00001100
639  14 110E  01000110          CAL   RELOAD
     15 110F  00011111
     16 1110  00001100
640  17 1111  00000110          LAI   000B
     18 1112  00000000
641  19 1113  01000110          CAL   DMOUT      ADRES OP DATA-BUS
     20 1114  11011101
     21 1115  00001001
642  22 1116  01000110          CAL   RARAM      LEES RAM OP ADRES 0
     23 1117  11000110
     24 1118  00001001
643  25 1119  00110110          LLI   12
     26 111A  00001100
644  27 111B  01000110          CAL   ORAM0      BYTE 0 RAM UIT
     28 111C  10000101
     29 111D  00001001
645  30 111E  01000110          CAL   COMP       VERGELIJK DATA
     31 111F  01100101
     32 1120  00001100
646  33 1121  01101000          JTZ   RAB01
     34 1122  00100101
     35 1123  00010001
647  36 1124  11111111          HLT
648  37 1125  01000110 RAB01    CAL   ORAM1      BYTE 1 RAM UIT
     38 1126  10001010
     39 1127  00001001
649  40 1128  01000110          CAL   COMP       VERGELIJK DATA
     41 1129  01100101
     42 112A  00001100
650  43 112B  01101000          JTZ   RAB02
     44 112C  00101111
     45 112D  00010001
651  46 112E  11111111          HLT
652  47 112F  01000110 RAEC2    CAL   ORAM2      BYTE 2 RAM UIT
     48 1130  10001111
     49 1131  00001001
653  50 1132  01000110          CAL   COMP       VERGELIJK DATA
     51 1133  01100101
     52 1134  00001100
654  53 1135  01101000          JTZ   TSTRAM
     54 1136  00111001
     55 1137  00010001
655  56 1138  11111111          HLT
656  57 1139  01000110 TSTRAM   CAL   STORE
     58 113A  00000000
     59 113B  00001100
657  60 113C  00101110          LHI   B(TKST16)
     61 113D  00010010
658  62 113E  00110110          LLI   A(TKST16)
     63 113F  11110001
659  64 1140  01000110          CAL   TEKST      PRINT: RARAM NA WCRAM
     65 1141  00001101
     66 1142  00001100
660  67 1143  01000110          CAL   RELOAD
```

```
     68 1144  00011111
     69 1145  00001100
661  70 1146  00001110          LBI   000B
     71 1147  00000000
662  72 1148  01000110          CAL   DMOUT     DON'T CARES OP MASKER-BUS
     73 1149  11011101
     74 114A  00001001
663  75 114B  01000110          CAL   LCAMV0    DON'T CARES IN KENMERKREG.
     76 114C  01101100
     77 114D  00001001
664  78 114E  01000110          CAL   TVPULS    GEEF T.V.-PULS
     79 114F  11001011
     80 1150  00001001
665  81 1151  01000110          CAL   RRAM      LEES RAM OP ASS.-ADRES
     82 1152  11000001
     83 1153  00001001
666  84 1154  11000000          LAA
667  85 1155  00110110          LLI   6
     86 1156  00000110
668  87 1157  01000110          CAL   ORAM0     BYTE 0 RAM UIT
     88 1158  10000101
     89 1159  00001001
669  90 115A  01000110          CAL   COMP      VERGELIJK DATA
     91 115B  01100101
     92 115C  00001100
670  93 115D  01101000          JTZ   SCRB51
     94 115E  01100001
     95 115F  00010001
671  96 1160  11111111          HLT
672  97 1161  01000110 SCRB51   CAL   ORAM1     BYTE 1 RAM UIT
     98 1162  10001010
     99 1163  00001001
673 100 1164  01000110          CAL   COMP      VERGELIJK DATA
    101 1165  01100101
    102 1166  00001100
674 103 1167  01101000          JTZ   SCRB52
    104 1168  01101011
    105 1169  00010001
675 106 116A  11111111          HLT
676 107 116B  01000110 SCRB52   CAL   ORAM2     BYTE 2 RAM UIT
    108 116C  10001111
    109 116D  00001001
677 110 116E  01000110          CAL   COMP      VERGELIJK DATA
    111 116F  01100101
    112 1170  00001100
678 113 1171  01101000          JTZ   WB60
    114 1172  01110101
    115 1173  00010001
679 116 1174  11111111          HLT
680 117 1175  01000110 WB60     CAL   STORE
    118 1176  00000000
    119 1177  00001100
681 120 1178  00101110          LHI   B(TKST17)
    121 1179  00010011
682 122 117A  00110110          LLI   A(TKST17)
    123 117B  00000111
683 124 117C  01000110          CAL   TEKST     PRINT: RRAM NA SCRAM
    125 117D  00001101
    126 117E  00001100
684 127 117F  01000110          CAL   RELOAD
```

```
    128 1180  00011111
    129 1181  C0001100
685 130 1182  01000110          CAL   TVPULS    GEEF T.V.-PULS
    131 1183  11001011
    132 1184  00001001
686 133 1185  01000110          CAL   RRAM      LEES RAM OP ASS.-ADRES
    134 1186  11000001
    135 1187  C0001001
687 136 1188  11000000          LAA
688 137 1189  01000110          CAL   ORAM0     BYTE 0 RAM UIT
    138 118A  10000101
    139 118B  C0001001
689 140 118C  01000110          CAL   COMP      VERGELIJK DATA
    141 118D  01100101
    142 118E  00001100
690 143 118F  01101000          JTZ   WB61
    144 1190  10010011
    145 1191  C0010001
691 146 1192  11111111          HLT
692 147 1193  01000110 WB61     CAL   ORAM1     BYTE 1 RAM UIT
    148 1194  10001010
    149 1195  C0001001
693 150 1196  01000110          CAL   COMP      VERGELIJK DATA
    151 1197  01100101
    152 1198  00001100
694 153 1199  01101000          JTZ   WB62
    154 119A  10011101
    155 119B  00010001
695 156 119C  11111111          HLT
696 157 119D  01000110 WB62     CAL   ORAM2     BYTE 2 RAM UIT
    158 119E  10001111
    159 119F  C0001001
697 160 11A0  01000110          CAL   COMP       VERGELIJK DATA
    161 11A1  01100101
    162 11A2  00001100
698 163 11A3  01101000         JTZ   WCRB00
    164 11A4  10100111
    165 11A5  C0010001
699 166 11A6  11111111          HLT
700 167 11A7  01000110 WCRB00   CAL   STORE
    168 11A8  00000000
    169 11A9  C0001100
701 170 11AA  00101110          LHI   B(TKST18)
    171 11AB  00010011
702 172 11AC  00110110          LLI   A(TKST18)
    173 11AD  C0011100
703 174 11AE  01000110          CAL   TEKST     PRINT: RRAM NA WRAM
    175 11AF  C0001101
    176 11B0  00001100
704 177 11B1  01000110          CAL   RELOAD
    178 11B2  00011111
    179 11B3  C0001100
705 180 11B4  01000110          CAL   TVRST     GEEF T.V. RESET
    181 11B5  11010100
    182 11B6  00001001
706 183 11B7  01000110         CAL   RRAM      LEES RAM OP ASS.-ADRES
    184 11B8  11000001
    185 11B9  00001001
707 186 11BA  11000000          LAA
708 187 11BB  01000110          CAL   ORAM0     BYTE 0 RAM UIT
```

```
    188 11BC  10000101
    189 11BD  C0001001
709 190 11BE  01000110          CAL   COMP      VERGELIJK DATA
    191 11BF  01100101
    192 11C0  00001100
710 193 11C1  01101000          JTZ   WCRB01
    194 11C2  11000101
    195 11C3  00010001
711 196 11C4  11111111          HLT
712 197 11C5  01000110 WCRB01   CAL   ORAM1     BYTE 1 RAM UIT
    198 11C6  10001010
    199 11C7  00001001
713 200 11C8  01000110          CAL   COMP      VERGELIJK DATA
    201 11C9  01100101
    202 11CA  00001100
714 203 11CB  01101000          JTZ   WCRB02
    204 11CC  11001111
    205 11CD  00010001
715 206 11CE  11111111          HLT
716 207 11CF  01000110 WCRB02   CAL   ORAM2     BYTE 2 RAM UIT
    208 11D0  10001111
    209 11D1  C0001001
717 210 11D2  01000110          CAL   COMP      VERGELIJK DATA
    211 11D3  01100101
    212 11D4  00001100
718 213 11D5  01101000          JTZ   EINDE
    214 11D6  11011001
    215 11D7  00010001
719 216 11D8  11111111          HLT
720 217 11D9  01000110 EINDE    CAL   STORE
    218 11DA  00000000
    219 11DB  C0001100
721 220 11DC  00101110          LHI   B(TKST19)
    221 11DD  C0010011
722 222 11DE  00110110          LLI   A(TKST19)
    223 11DF  0011C0C0
723 224 11E0  01000110          CAL   TEKST     PRINT: RRAM NA WCRAM
    225 11E1  C0001101
    226 11E2  00001100
724 227 11E3  01000110          CAL   RELOAD
    228 11E4  00011111
    229 11E5  C0001100
725 230 11E6  11111111          HLT
726 231 11E7           TEKST1   DEF   '  ASS SCAM  f'
727 244 11F4           TEKST2   DEF   '  ASS WCAM  f'
728   1 1201           TEKST3   DEF   '  SCAM AANWEZIG ?  f'
729  21 1215           TEKST4   DEF   '  WCAM AANWEZIG ?  f'
730  41 1229           TEKST5   DEF   '  ASS NA TV-PULS  f'
731  60 123C           TEKST6   DEF   '  ASS NA TVRST  f'
732  77 124D           TEKST7   DEF   '  WCAM NA LCAMV0  f'
733  96 1260           TEKST8   DEF   '  WCAM NA LCAMV1  f'
734 115 1273           TEKST9   DEF   '  ASS NA SHIFT  f'
735 132 1284           TKST10   DEF   '   SRAM  OKE  f'
736 147 1293           TKST11   DEF   '   RARAM  NA  SRAM  f'
737 168 12A8           TKST12   DEF   '  ASS  SCRAM  f'
738 183 12B7           TKST13   DEF   '   RARAM  NA  SCRAM  f'
739 205 12CD           TKST14   DEF   '   RARAM  NA  WRAM  f'
740 226 12E2           TKST15   DEF   '  ASS  WCRAM  f'
741 241 12F1           TKST16   DEF   '   RARAM  NA  WCRAM  f'
742   7 1307           TKST17   DEF   '   RRAM  NA  SCRAM  f'
```

```
743  28 131C          TKST18   DEF   '   RRAM  NA  WRAM  f'
744  48 1330          TKST19   DEF   '   RRAM  NA  WCRAM  f'
745 197 01C5          CRLF     ORG   453
746 161 01A1          TTYOUT   ORG   417
747  63 093F          LRAM0    ORG   2367
748  68 0944          LRAM1    ORG   2372
749  73 0949          LRAM2    ORG   2377
750  78 094E          LCAM0    ORG   2382
751  83 0953          LCAM1    ORG   2387
752  88 0958          LCAM2    ORG   2392
753  93 095D          LCRAM0   ORG   2397
754  98 0962          LCRAM1   ORG   2402
755 103 0967          LCRAM2   ORG   2407
756 108 096C          LCAMV0   ORG   2412
757 113 0971          LCAMV1   ORG   2417
758 118 0976          LCAMV2   ORG   2422
759 123 097B          SHIFT    ORG   2427
760 133 0985          ORAM0    ORG   2437
761 138 098A          ORAM1    ORG   2442
762 143 098F          ORAM2    ORG   2447
763 148 0994          OCAM     ORG   2452
764 153 0999          WCAM     ORG   2457
765 158 099E          WRAM     ORG   2462
766 163 09A3          WCRAM    ORG   2467
767 168 09A8          SCAM     ORG   2472
768 173 09AD          SRAM     ORG   2477
769 178 09B2          SCRAM    ORG   2482
770 183 09B7          RSTCAM   ORG   2487
771 188 09BC          CLCAM    ORG   2492
772 193 09C1          RRAM     ORG   2497
773 198 09C6          RARAM    ORG   2502
774 203 09CB          TVPULS   ORG   2507
775 212 09D4          TVRST    ORG   2516
776 221 09DD          DMOUT    ORG   2525
777 234 09EA          DATAIN   ORG   2538
778 246 09F6          READY    ORG   2550
779   5 0A05          CAMVOL   ORG   2565
780  18 0A12          ASS      ORG   2578
781  32 0A20          START    ORG   2592
782  32 0A20                   END
```

EXPLANATION OF ERROR-CODES

```
 1 : LABEL TOO LONG
 2 : NO OPERATION SPECIFIED
 3 : INVALID NUMBER SPECIFICATION
 4 : INVALID OCTAL NUMBER SPECIFICATION
 5 : DUPLICATE LABEL
 6 : OPERAND MISSING
 7 : INVALID OPERATION CODE
 8 : END CARD MISSING
 9 : INVALID OPERAND
10 : UNDEFINED LABEL
```