ENIGE OPMERKINGEN OVER DE WERKWIJZE VAN EEN COMPUTER

door Drs. A. van Ballegooyen

De laatste jaren wordt voor administratieve doeleinden steeds meer gebruik gemaakt van computers.

In de toekomst zullen de arbeidskosten van kantoorwerk en de daarmede verbandhoudende kosten, zoals huisvestingskosten, waarschijnlijk stijgen. Naarmate de technische ontwikkeling voortschrijdt en het gebruik van computers toeneemt zullen de kosten van computers vermoedelijk relatief gaan dalen. Daarom mag verwacht worden dat in de toekomst nog meer gebruik zal worden gemaakt van computers.

Dit verklaart misschien de stijgende behoefte aan kennis van de grondslagen volgens welke een computer werkt.

Veelal is wel bekend dat onder een computer een elektronische rekenmachine wordt verstaan en dat door een computer berekeningen worden uitgevoerd. Op welke wijze deze berekeningen worden uitgevoerd behoeft nadere toelichting. Belangrijker echter dan het geven van een definitie van een computer achten wij het uiteenzetten van de werking van een computer. In het navolgende zullen we pogen uiteen te zetten hoe een computer, wat de hoofdlijnen betreft, werkt.

Daarbij zullen we ons alleen bezighouden met computers welke voor administratieve doeleinden worden gebruikt.

We beperken ons tot de principes. Dat er tussen de diverse soorten en merken computers zekere nuanceringen bestaan, welke afhankelijk zijn van de soort computer (large scale, medium scale of small scale) en het merk, spreekt vanzelf. De principes worden echter daardoor geen geweld aangedaan.

Indien de manier waarop een computer werkt bekend is zal men zich een eenvoudiger voorstelling kunnen maken van de toepassingen welke mogelijk zijn. Daarmede wordt voorkomen dat de mogelijkheden van de moderne apparatuur worden onderschat. Binnen een goede organisatie zijn logische problemen veelal, technisch, met een computer oplosbaar. Anderzijds wordt voorkomen dat de mogelijkheden van deze apparatuur worden overschat. De computer is geen wondermachine welke de meest onlogische dingen op een logische wijze kan oplossen.

Om de wijze waarop een computer werkt eenvoudig voor te kunnen stellen, gaan we uit van de werkwijze van een schrijvende telmachine, welke bekend mag worden verondersteld.

Bij een schrijvende telmachine leest de mens de gegevens af van een document en slaat deze gegevens aan op de telmachine door de overeenkomstige cijfertoetsen in te drukken. Door hierna op de plus-toets te drukken wordt de instructie gegeven op te tellen, door daarentegen op de min-toets te drukken wordt de instructie gegeven de in de cijfertoetsen voorkomende cijfers af te trekken van de cijfers welke in de telwielen van het telwerk voorkomen. Na het drukken op de totaaltoets wordt de verkregen uitkomst (som, verschil) afgedrukt op de telstrook.

De op deze telstrook verkregen uitkomst zal men willen bewaren en daartoe noteert de mens via het menselijk geheugen deze uitkomst op een document zodat deze uitkomst vast ligt. Eventueel wordt de telstrook bijgevoegd.

Indien we de verrichte werkzaamheden analyseren dan worden de volgende handelingen door de mens verricht:

1. lezen van de cijfers welke op het document voorkomen
2. aanslaan van deze cijfers op de cijfer-toetsen van de telmachine d.w.z. het omzetten van de cijfers (het gegeven) in machinetaal
3. drukken op de instructietoets (plus-, min-toets), waardoor de in de cijfer-toetsen voorkomende cijfers overgebracht worden naar het telwerk en daar geteld worden
4. drukken op de totaaltoets d.w.z. het omzetten van het in machinetaal voorkomende totaal in leesbaar schrift; het gelijktijdig schoonmaken van het telwerk
5. noteren van de uitkomst vanaf de telstrook op het document d.w.z. het vastleggen, bewaren van de uitkomst.

Bij bovenstaande met de telmachine uitgevoerde handelingen is geen sprake van automatisering. Voor iedere uit te voeren handeling is menselijk ingrijpen noodzakelijk.

In het navolgende zal uiteengezet worden hoe bij het gebruik maken van een computer volledige automatisering van de uit te voeren handelingen wordt verkregen.

Deze automatisering ontstaat door:

I. *Automatisering van de in te lezen gegevens,*
door deze gegevens vooraf om te zetten in machinetaal en de computer uit te rusten met een leesmechanisme.

II. *Automatisering van de uit te voeren berekeningen:*
A. door de computer uit te rusten met
1. een geheugen
2. een rekenmechanisme
B. door gebruik te maken van vooraf opgestelde instructies (het programma), waardoor de besturing geautomatiseerd wordt.

III. *De zelfbesturing* d.w.z. het zelfstandig door de computer nemen van logische beslissingen.

ad I. Automatisering van de in te lezen gegevens door deze te vertalen in machinetaal.

Indien we de door de telmachine uitgevoerde handelingen bezien dan blijkt dat de handelingen sub 1, 2 en 4 betrekking hebben op het vertalen. Men moet nl. twee keer vertalen.

Ten eerste moet het leesbare schrift omgezet worden in machinetaal (code).

Ten tweede moet, nadat de bewerkingen in machinetaal zijn uitgevoerd, de uitkomst worden vertaald van machinetaal in leesbaar schrift.

Het vertalen van leesbaar schrift in machinetaal komt tot stand door manuaal werk, in casu door het indrukken van toetsen.

Het lezen van de op te tellen gegevens en het, door menselijke handeling, omzetten ervan in machinetaal zou achterwege kunnen blijven indien de telmachine beschikken zou over een leesmechanisme, dat in staat zou zijn de op het document voorkomende gegevens meteen vanaf het document te lezen. Uiteraard gaat dit

alleen dan op indien de taal, welke op dit document voorkomt, door de telmachine verstaan zou worden en indien de gegevens die op dit document voorkomen aan zekere eisen voldoen (vaste afmetingen etc.).

Een voorbeeld van een taal welke door een machine wordt verstaan is de ponstaal welke is aangebracht in een ponskaart. Door middel van de ponskaartenvorm wordt de vaste plaats, de vaste afmeting etc. gewaarborgd. De op een document voorkomende gegevens worden met een ponsmachine in een ponskaart geponst. Bij dit ponsen zijn de handelingen sub 1 en 2 noodzakelijk. Indien deze gegevens echter eenmaal in ponstaal in een ponskaart voorkomen kunnen zij zonder meer door een machine, welke over een leesmechanisme beschikt, gelezen worden. De handelingen sub 1 en 2 zijn dan overbodig geworden omdat de ponskaart een vorm van een geheugen is, waarin de gegevens bewaard blijven in een voor machines verstaanbare taal. Hierdoor worden de in te lezen gegevens geautomatiseerd. Zij kunnen telkens opnieuw worden gebruikt.

Een ander voorbeeld van een taal welke door een machine wordt verstaan is de telex-taal.

Een voorbeeld van een moderne machinetaal is de character-reading. In dit laatste geval komen de gegevens reeds op het basis document voor in machinetaal, zodat nu de handelingen sub 1 en 2 volledig overbodig zijn geworden. Deze machinetaal heeft nu de vorm van leesbaar schrift, dat zowel door de mens als ook door de machine verstaanbaar is.

Zowel t.a.v. de machinetaal als t.a.v. het leesmechanisme is een grote ontwikkeling aan de gang.

Het zal duidelijk zijn dat een telmachine als regel niet beschikt over een leesmechanisme omdat dit te kostbaar zou zijn.

Een computer beschikt wel over een leesmechanisme in tegenstelling tot de schrijvende telmachine zodat alle gegevens, welke in deze machinetaal zijn gecodeerd, door de computer gelezen en verstaan kunnen worden.

Welke machinetaal hierbij wordt gebruikt hangt af van de soort en merk computer. Er komen computers voor welke werken met ponsband, magnetic tape of een andere wijze van invoer-uitvoer.

Van welk medium ook gebruik gemaakt wordt doet aan het principe niets af. Het principe is dat de gegevens eerst worden omgezet in het medium (ponskaart, -band, tape).

Daarna kunnen de gegevens via het medium worden ingevoerd in de computer. Men noemt dit de inputgegevens.

De gegevens kunnen via het leesmechanisme automatisch door de computer worden gelezen, zonder menselijke interventie.

Hetzelfde wat zich voordoet bij de in te voeren gegevens kan zich voordoen bij de uit te voeren gegevens (de met de inputgegevens verkregen uitkomsten).

Indien de uitkomsten eenmaal door de computer zijn berekend komen zij voor in machinetaal. Deze uitkomsten kunnen zonder meer in deze machinetaal worden verstrekt d.w.z. door de computer worden geponst in ponskaarten, ponsband etc. Met een aparte vertaalmachine zal dit medium dan moeten worden omgezet in leesbaar schrift (de printer).

Het is echter ook mogelijk dat er aan de computer een vertaalmachine gekoppeld is, een zgn. gekoppelde printer, welke de in machinetaal beschikbare uitkom-

sten direct uit het geheugen op papier afdrukt in leesbaar schrift. In dat geval wordt niet van het ponskaarten medium of een ander medium gebruik gemaakt, nu wordt vertaald van machinetaal in leesbaar schrift. Indien de verkregen uitkomsten later opnieuw ingevoerd moeten worden in de computer dienen deze tevens in machinetaal te worden verstrekt.

Afhankelijk van het gebruikte medium zal dus het invoermechanisme (leesmechanisme) en uitvoermechanisme (ponsmechanisme) verschillen. Hoewel het invoer-uitvoer-mechanisme van de verschillende soorten computers verschillend kan zijn behoeft de werkwijze van de computer hierdoor niet te veranderen.

In feite worden de in het inleesmechanisme van de computer ingelezen gegevens nog eens vertaald, nl. van tientallig stelsel in tweetallig stelsel.

De ingelezen gegevens zijn uitgedrukt in het tientallig stelsel. Bij het tientallig stelsel zijn tien tafels van vermenigvuldiging noodzakelijk, bij een tweetallig stelsel slechts twee (0 en 1).

decimaal cijfer	*binair*
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001

De gemakkelijkste taal voor een machine is dan ook het tweetallig stelsel, wat vertaling van decimale code in binaire code noodzakelijk maakt. Berekeningen voert de computer uit in binaire code (tweetallig stelsel), of een vorm daarvan.

ad II. Automatisering van de uit te voeren berekeningen:

ad II A 1. Het Geheugen.

We gaan verder met de door de schrijvende telmachine uitgevoerde bewerking.

Wil men dezelfde gegevens, of de uitkomst, later nog eens gebruiken, doch nu in een andere volgorde of voor een andere bewerking, dan dienen deze gegevens via het menselijk geheugen *opnieuw* op de telmachine te worden aangeslagen, d.w.z. omgezet in machinetaal. Daarbij dienen de corresponderende instructies (optellen, aftrekken) gegeven te worden.

Een geheugen waarin deze gegevens, op een bepaalde plaats, liggen opgeslagen ontbreekt. Hierdoor moet de mens steeds opnieuw de gegevens van het document (ook een vorm van een geheugen) in de telmachine inbrengen, terwijl de uitkomsten ook weer door de mens vanaf de telstrook moeten worden genoteerd op hetzelfde of een ander document. Het ontbreken van een geheugen maakt automatisering van een aantal verschillende bewerkingen onmogelijk.

Een computer beschikt over een geheugen. Men zou dit geheugen kunnen vergelijken met de band van een bandrecorder of met een grammofoonplaat. De te

verwerken gegevens worden in een door de computer verstaanbare taal omgezet, b.v. in ponstaal bij gebruik van ponskaarten.

Daarna worden deze ponskaarten, of een ander medium, ingevoerd in het leesmechanisme van de computer, waar de ponskaarten gelezen worden. De gegevens welke in de ingelezen ponskaart voorkomen worden van het leesstation overgebracht naar het geheugen; dus opgeslagen in het geheugen. De ingelezen ponskaart wordt als het ware gefotografeerd in het geheugen.

Gedurende de tijd dat het gegeven uit een input-ponskaart opgeslagen ligt in het geheugen kunnen achtereenvolgens alle mogelijke soorten bewerkingen hiermede worden uitgevoerd zonder menselijke tussenkomst. Het gegeven ligt nu eenmaal vast op een bepaalde plaats in het geheugen en daarover kan steeds weer opnieuw worden beschikt, zonder dat verder iets in de computer behoeft te worden ingevoerd en zonder dat het gegeven dat in het geheugen vastligt te niet gaat. Dit gegeven wordt gebruikt doch niet verbruikt. Het wordt eerst verbruikt (vernietigd) indien geprogrammeerd wordt dat het gegeven op dat moment vernietigd moet worden. Dit in tegenstelling tot de werking van een schrijvende telmachine waar het gegeven meteen vernietigd (verbruikt) wordt.

Ook de uitkomst kan opgeslagen (bewaard) worden in het geheugen zodat ook daarover kan worden beschikt zonder menselijke tussenkomst. Ook hierbij is er een verschil met de schrijvende telmachine.

ad II A 2. Het Rekenmechanisme.

De gegevens (de data) waarmede de bewerking wordt uitgevoerd liggen opgeslagen in het geheugen en blijven daar bewaard, zodat zij later te allen tijde beschikbaar zijn. Hierdoor kunnen later met deze gegevens alle mogelijke bewerkingen worden uitgevoerd.

Het geheugen kan geen bewerkingen uitvoeren, dus niet rekenen, doch alleen maar cijfers, letters en symbolen bewaren. Het rekenen wordt in de *rekeneenheid* uitgevoerd. Eenvoudig voorgesteld is de rekeneenheid niets anders dan een grote telmachine welke razend snel optelt en aftrekt. De gegevens (data) waarmede een bewerking moet worden uitgevoerd dienen eerst uit het geheugen naar de rekeneenheid te worden overgebracht, waarna de bewerking kan worden uitgevoerd. Welke bewerking zal worden uitgevoerd hangt af van de instructie welke de computer ontvangt. Door op de plus-toets te drukken kreeg de schrijvende telmachine de instructie op te tellen, door op de min-toets te drukken kreeg de telmachine de instructie af te trekken. Door op *verschillende* toetsen te drukken worden verschillende bewerkingen uitgevoerd in casu optellen, aftrekken.

Een derde bewerking b.v. vermenigvuldigen wordt mogelijk door aan de schrijvende telmachine een vermenigvuldig-toets toe te voegen en hierop te drukken. Er zouden dus zoveel verschillende soorten berekeningen door de (tel) machine kunnen worden uitgevoerd als er bewerkingstoetsen voorkomen.

ad II B De Instructie en het Programma.

De bewerkingen welke de computer uit moet voeren worden niet uitgevoerd door op toetsen te drukken, doch worden *automatisch*, zonder menselijke interventie, uitgevoerd. Evenals de telmachine zijn instructie kreeg om op te tellen of af te trekken, moet ook de computer een soortgelijke instructie ontvangen.

De instructie welke de telmachine ontvangt is in feite tweeledig.

Door op de instructietoets (plus-toets) te drukken wordt, ten eerste, het gegeven dat in de cijfertoetsen voorkomt overgebracht naar het telwerk, het rekenmechanisme. Ten tweede wordt de bewerking in dit rekenmechanisme uitgevoerd, overeenkomstig de ingedrukte instructie-toets (optellen, aftrekken).

De instructie welke de computer ontvangt is eveneens tweeledig en in feite gelijk aan de instructie welke de telmachine ontvangt; alleen de vorm waarin de instructie wordt gegeven verschilt.

Het eerste deel van de instructie aan de computer dient weer om het benodigde gegeven uit het geheugen van de computer naar de rekeneenheid van de computer te brengen. Het tweede deel van de instructie aan de computer geeft aan welke bewerkingen moeten worden uitgevoerd, vermenigvuldigen, optellen etc. Op beide delen waaruit deze instructie is opgebouwd gaan wij nader in, terwijl we daarna de vorm waarin de computer de instructie ontvangt zullen uiteenzetten.

De gegevens (data) waarmede gerekend moet worden liggen opgeslagen in het geheugen. Of het geheugen gevormd wordt door een roterende magnetische trommel, of een magneetkernengeheugen, of op een andere wijze, doet aan het principe niets af. Het systeem is dat cijfers, letters en symbolen kunnen worden opgeslagen in alle mogelijke geheugen-vormen. Ook een ponskaart is een vorm van geheugen, zij het een extern geheugen. De grootte van het geheugen van een computer kan variëren. De grootte van het geheugen drukt men uit door aan te geven hoeveel posities (cijfers) in het geheugen kunnen worden bewaard (opgeslagen). Een positie is een cijfer, of een letter of een symbool. Er zijn geheugens van 100 posities doch ook computers met een geheugen van 100.000 posities.

Steeds wordt het geheugen in onderdelen verdeeld, zoals een straat verdeeld wordt in adressen. Evenals aan ieder huisadres een nummer is gegeven waardoor bij adressering volledige localisering van dat huis in die straat mogelijk is, is aan iedere kleinste geheugeneenheid een vast nummer, het adres, gegeven. Met dit adres-nummer is de plaats waar een gegeven zich in het geheugen bevindt volledig bepaald.

De grootte van zo'n kleinste geheugeneenheid kan bij de verschillende soorten computers variëren b.v. van 1 positie (cijfer, letter of symbool) tot 10 posities per geheugeneenheid, evenals men op één huisadres huizen heeft met één kamer en huizen met tien kamers. Essentieel is dat iedere kleinste geheugeneenheid steeds een apart adresnummer heeft zodat deze localiseerbaar, of beter, adresseerbaar is.

Indien een geheugeneenheid 10 posities (cijfers) groot is kan men b.v. bij de loonberekening in adresnr. 0001 (geheugeneenheid nr. 1) het uurloon brengen en in adresnr. 0003 de gewerkte uren. Met dit adresnummer van de geheugeneenheid kan men de gegevens welke in het geheugen voorkomen oproepen en naar de rekeneenheid brengen.

Zo roept men met adres 0001 het uurloon op.

Indien in adres 0001 zowel het uurloon als de gewerkte uren zijn opgeborgen dan is deze geheugeneenheid niet homogeen. Men zal nu eerst deze geheugeneenheid in twee delen moeten splitsen zodat hiervoor in feite twee geheugeneenheden worden vastgesteld.

Eerst daarna kunnen zinvolle bewerkingen worden uitgevoerd.

Indien de geheugeneenheid uit een vast aantal posities bestaat b.v. steeds uit 10 posities bestaat, dan zijn de adressen 0001 en 0003 slechts voor 40 % bezet, aannemende dat uurloon en gewerkte uren beide 4 posities groot zijn.

Om een zo hoog mogelijk rendement van de geheugencapaciteit te behalen heeft men bij sommige computers de vaste woordlengte van de geheugeneenheid losgelaten en vervangen door de variabele woordlengte van de geheugeneenheid.

Een gegeven (data, woord) van 4 cijfers vergt dan slechts een geheugencapaciteit van 4 posities in plaats van 10, terwijl een gegeven van 3 cijfers slechts 3 posities geheugenruimte in beslag neemt in plaats van 10.

Indien men nu de gewerkte uren met het uurloon wil vermenigvuldigen dient de *inhoud* van adres 0003 naar een of andere vorm van een rekeneenheid te worden gebracht en daarna de inhoud van adres 0001.

Vervolgens wordt na een daartoe ontvangen instructie de inhoud van adres 0003 met de inhoud van adres 0001 vermenigvuldigd in de rekeneenheid.

De uitkomst, het basisloon, moet nu bewaard blijven en dient daartoe uit de rekeneenheid naar b.v. adres 0004 van het geheugen gebracht te worden. Indien later het basisloon aan het brutoloon moet worden toegevoegd zal de inhoud van adres 0004 weer uit het geheugen naar de rekeneenheid moeten worden gebracht en met een instructie opgeteld moeten worden bij het brutoloon dat b.v. in adres 0005 wordt bewaard. M.a.w. de inhoud van adres 0004 wordt naar de rekeneenheid gebracht, de inhoud van adres 0005 wordt naar de rekeneenheid gebracht, de inhoud van beide adressen worden in de rekeneenheid (of een vorm daarvan) bij elkaar geteld en de som wordt daarna uit de rekeneenheid overgebracht naar adres 0005 van het geheugen.

Ieder adresnummer is d.m.v. een codecijfer te identificeren. Hoe dit codenummer er uit ziet doet aan het beginsel niets af.

Op welke plaats van het geheugen men een bepaald gegeven opbergt is een kwestie van systeem en programmeringstechniek, afhankelijk van de soort computer. In beginsel kan men hiervoor echter elk adres nemen.

Voor ieder programma kan men voor eenzelfde gegeven een ander adres vaststellen (programmeren).

Enerzijds bestaat een instructie dus uit een *adresaanduiding*, d.i. met welke geheugeneenheid de bewerking moet worden uitgevoerd.

Anderzijds moet de instructie ook een aanduiding bevatten *welke* bewerking moet worden uitgevoerd, de zgn. operation.

Ook hiervoor worden codes gebruikt:

b.v. operation-code
- 15 = optellen, overbrengen van het geheugen naar de rekeneenheid en toevoegen aan de inhoud ervan.
- 11 = aftrekken
- 19 = vermenigvuldigen
- 20 = overbrengen van gegevens uit de rekeneenheid naar een bepaalde plaats van het geheugen.
- 70 = inlezen van een ponskaart
- 71 = ponsen van een ponskaart

Indien we nu de twee delen waaruit een instructie bestaat combineren dan stelt instructie 15.0001 voor dat de inhoud van geheugeneenheid 0001, het uurloon, overgebracht wordt naar de rekeneenheid en opgeteld wordt bij de inhoud van de rekeneenheid. Stel dat de inhoud van de rekeneenheid nul is dan komt het uurloon in de rekeneenheid te staan. De volgende instructie luidt nu 19.0003 d.w.z. dat de inhoud van adres 0003 (de gewerkte uren) naar de rekeneenheid

moet worden gebracht en daar vermenigvuldigd moet worden met de inhoud van de rekeneenheid. Nu worden uurloon en gewerkte uren met elkaar vermenigvuldigd en de uitkomst wordt gevormd in de rekeneenheid. Met de volgende instructie 20.0005 wordt de inhoud van de rekeneenheid (het basisloon) overgebracht naar adres-nr. 0005 van het geheugen.

D.m.v. programmering, d.i. het opstellen van instructies, wordt bepaald welke instructies achtereenvolgens moeten worden uitgevoerd. Daartoe gaat aan de programmering vooraf de systeem-analyse. Door deze systeem-analyse worden de uit te voeren werkzaamheden geanalyseerd in onderdelen (elementen), daarna geordend en omgezet in een blokdiagram, waarna geprogrammeerd kan worden.

Onder een programma verstaat men het afgesloten geheel van instructies dat op een bepaald karwei betrekking heeft.

Indien men het starten en wegrijden van een auto zou moeten programmeren zou dit programma uit de volgende instructies (programmastappen) bestaan:

1. koppelingspedaal indrukken
2. contactsleutel omdraaien waardoor de motor gestart wordt
3. schakelingshandle in eerste versnelling zetten
4. handrem vrij zetten
5. kijken naar achteroprijdend en tegemoetkomend verkeer of weggereden kan worden zonder het verkeer te hinderen

6a. zo niet, niet wegrijden en blijven kijken tot dit wel mogelijk is

6b. zo ja, gaspedaal indrukken

7. koppelingspedaal op laten komen
8. sturen, wegrijden

In dit programma komt onder 6, een vraagstelling voor waarop het antwoord ja of neen kan zijn. Hierop wordt in het navolgende teruggekomen.

Het zal duidelijk zijn dat de instructies in een bepaalde *volgorde* moeten worden uitgevoerd.

Indien in het bovenstaande voorbeeld van een helling weggereden moet worden zal de handrem eerst in een later stadium vrij gezet moeten worden, anders rijdt de auto achteruit. Instructie 4. moet in dat geval tussen instructie 7. en 8. geprogrammeerd worden.

Zo zal men, in het genoemde voorbeeld van vermenigvuldiging van gewerkte uren met het uurloon, eerst het uurloon in adres 0001 van het geheugen moeten brengen. Pas daarna mag de instructie om de gewerkte uren met het uurloon te vermenigvuldigen worden uitgevoerd. Zou men de volgorde omkeren dan zou de uitkomst van de vermenigvuldiging altijd nul zijn of foutief, omdat de inhoud van adres 0001 dan nul is of een foutief gegeven bevat.

De volgorde van de instructies binnen een programma moet altijd hetzelfde zijn. Zo vormen de opeenvolgende instructies als het ware de schalmen van een ketting. Een programma bestaat dan ook uit een ketting van instructies. De eerste instructie van een ketting van instructies zal moeten luiden dat een nieuwe input-ponskaart ingelezen (ingevoerd) moet worden in de computer. De volgende instructies zullen o.a. moeten inhouden de ingelezen gegevens naar hun bestemde plaats in het geheugen te brengen. Daarna worden alle mogelijke berekeningen met deze in het geheugen opgeslagen gegevens uitgevoerd d.m.v. instructies. Vervolgens zal aan het slot van de ketting van instructies een instructie voor moeten

komen „pons de berekende gegevens (de uitkomsten) uit in output-ponskaarten" of „schrijf de verkregen uitkomsten op papier".

Eén van de daarop volgende instructies zal moeten luiden „stel die geheugeneenheden waarin de data (zoals 0001-uurloon en 0003-gewerkte uren) zijn opgeborgen op nul" waardoor de inhoud van deze geheugeneenheden wordt uitgewist. Zou men deze instructie achterwege laten dan zou een werknemer welke geen ploegentoeslag in zijn loon mag ontvangen toch ploegentoeslag ontvangen indien de vorige werknemer ploegentoeslag ontvangt.

De laatste instructie van de ketting van instructies welke tevens de eerste instructie van de ketting is luidt: „Lees een nieuwe input-ponskaart". De gegevens welke in deze nieuw ingelezen ponskaart voorkomen doorlopen weer dezelfde ketting van instructies. Eén van die instructies zal zijn het uurloon uit deze ponskaart op te bergen in geheugeneenheid 0001. Daardoor verandert de *inhoud* van adres 0001.

Eerst was het uurloon van werknemer A (b.v. 1,30) in geheugeneenheid 0001 opgeborgen, doch na het inlezen van de volgende ponskaart en na het uitvoeren van de betrokken instructie is in diezelfde geheugeneenheid 0001 het uurloon (b.v. 1,36) van werknemer B opgeborgen.

De inhoud van deze geheugeneenheden wisselt steeds. Het is alsof op een bepaald huisnummer mijnheer A verhuisd is terwijl het huis nu bewoond is door mijnheer B. Het huisnummer verandert hier niet door. Het *systeem blijft gelijk* ook al wisselt de inhoud van iedere geheugeneenheid of een deel van de geheugeneenheden. Als men a × b vermenigvuldigt blijft de formule bestaan, doch de waarde van a en b (d.i. in computertaal de inhoud van een geheugenadres) kan telkens veranderen.

Programmeren is te vergelijken met het maken van een zetsel bij het drukken. De loden letters worden samengevoegd tot woorden. De woorden worden aaneengeregen tot zinnen.

Een aantal zinnen tesamen vormen een zetsel (cliché).

Een programma wordt op dezelfde wijze gemaakt.

De codes worden tot instructies gevormd.

Een aantal instructies tesamen vormen een bewerking.

Een aantal bewerkingen tesamen geven het programma van een job.

Er zijn zetsels van 100 woorden, doch er zijn ook zetsels van 2000 woorden.

Zo zijn er programma's van 100 instructies doch ook programma's van 2000 instructies.

Dit hangt af van het probleem en de soort en merk computer.

In de ene computer kan met één instructie meer berekend worden dan in een andere computer met drie instructies.

Een programma bestaat al gauw uit honderden instructies.

De te gebruiken letters in een zetsel en de volgorde van de letters kunnen, afhankelijk van het zetwerk, bij ieder zetsel verschillend zijn.

Zo kunnen in een programma de te gebruiken instructies, de volgorde van de instructies en het aantal instructies verschillend zijn.

Dit is afhankelijk van de programmering door de programmeur en de systeem analyse door de systeem analyst.

Het schrijven van het programma en de er aan voorafgaande systeem analyse is een tijdrovende aangelegenheid.

Voor ieder karwei zal een afzonderlijk programma gemaakt moeten worden. Men heeft dus in beginsel evenveel programma's in voorraad als er jobs uitgevoerd moeten worden. Dus b.v. een factureringsprogramma, een programma om de lonen te berekenen, enz.

Deze programma's staan voor het gebruik gereed evenals de zetsels voor het gebruik gereed staan, indien daar herhaaldelijk gebruik van zou worden gemaakt.

Zodra men wil gaan drukken wordt het betrokken zetsel uit de kast genomen en in de drukpers bevestigd. De blanco vellen papier worden ingevoerd in de drukpers en door het zetsel van de overeenkomstige opdruk voorzien.

Zodra men een bepaald karwei met de computer gaat uitvoeren wordt het betrokken programma (zetsel) uit de kast genomen en in de computer bevestigd. Daarna worden de ponskaarten, waarin de gegevens voorkomen welke berekend moeten worden, ingevoerd in de computer evenals de blanco vellen papier bij het drukken.

De uitkomsten welke nu worden verkregen zullen voor ieder programma verschillend zijn evenals bij het drukken.

Na het gebruik wordt het programma (zetsel) weer uit de computer gehaald en opgeborgen in de kast tot hetzelfde karwei de volgende keer moet worden uitgevoerd. Steeds wordt van hetzelfde programma gebruik gemaakt zoals ook bij het drukken steeds van hetzelfde zetsel gebruik gemaakt wordt.

Een zetfout in het zetsel veroorzaakt een drukfout op ieder vel papier. Een zetfout in het programma, dus een foutieve instructie, heeft tot gevolg dat iedere uitkomst dezelfde fout bevat.

De vorm waarin dit programma voorkomt zal in het navolgende separaat worden behandeld.

In de drukpers worden blanco vellen papier ingevoerd, tegen het zetsel gedrukt en afgevoerd.

Zo worden input-ponskaarten (of band, tape), waarin de gegevens voorkomen welke berekend moeten worden, ingevoerd in de computer. Bij de loonberekening zal voor iedere werknemer waarvan het loon berekend moet worden, een input-ponskaart worden ingevoerd in de computer. In deze kaart komen o.a. voor: het nummer van de werknemer, zijn uurloon, zijn aantal gewerkte uren, de klasse loonbelasting, etc. Deze input-gegevens doorlopen nu het programma dat in de computer voorkomt. Het loon van iedere ingevoerde werknemer wordt volgens dit programma berekend en de uitkomst (het bruto loon, de inhoudingen, het netto loon e.d.) afgedrukt op papier en/of in output-ponskaarten geponst. Werknemer na werknemer wordt ingevoerd, berekend en afgedrukt door de computer.

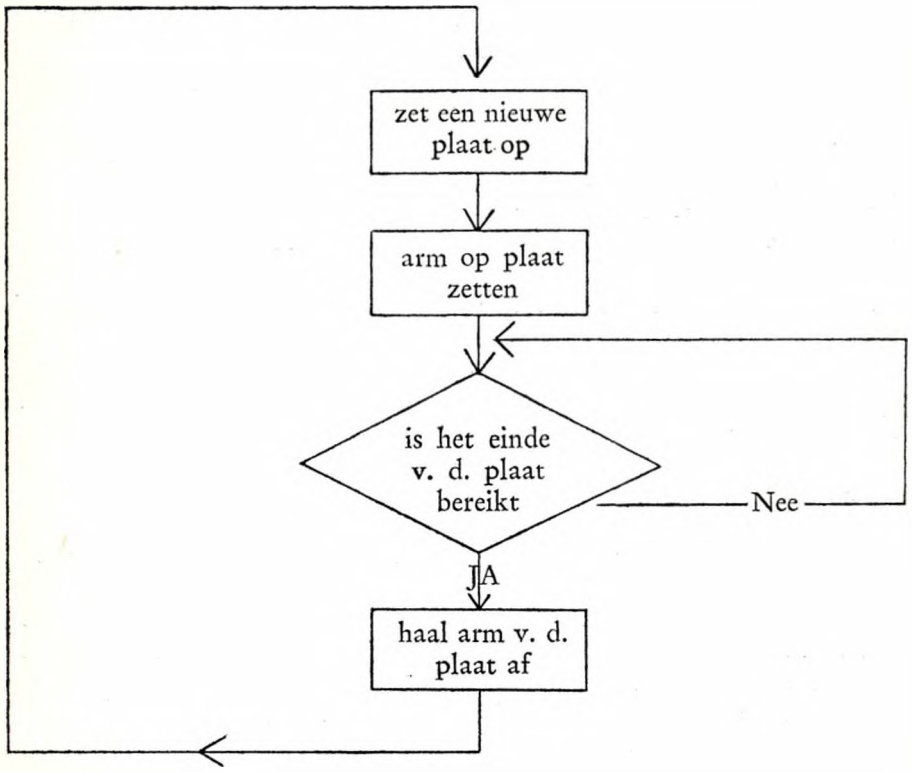
Er kunnen nu een onbeperkt aantal input-ponskaarten (of andere input-media) worden ingevoerd in de computer, evenals er een onbeperkt aantal vellen blanco papier tegen het zetsel gedrukt kunnen worden en van dezelfde opdruk voorzien.

Iedere input-ponskaart doorloopt daarbij in principe hetzelfde stuk van het programma, of beter, dezelfde ketting van instructies in het programma.

De ketting van instructies vormt in feite een gesloten keten d.i. een vicieuse cirkel. Het einde van de ketting (ponsen en nulstellen) wordt aangesloten aan het begin van de ketting (lezen), waardoor *automatisering* ontstaat.

Men zou dit kunnen vergelijken met een grammofoon voorzien van een *automatische platenwisselaar.* Aan de grammofoon bevindt zich een beweegbare arm waaraan de naald bevestigd is. De arm met naald wordt telkens automatisch ingesteld op de eerste groef van de grammofoonplaat. Daarna doorloopt de naald de instructies d.m.v. de groeven in de grammofoonplaat. Indien de naald het eind van de grammofoonplaat bereikt, wordt de arm met de naald van de eerste grammofoonplaat verwijderd en automatisch op de eerste groef van de tweede grammofoonplaat (het begin) ingesteld.

Indien men bovenstaande handelingen weergeeft in de vorm van een blokdiagram dan ziet dit er als volgt uit:



De basis voor iedere programmering vormt het blokdiagram, waarin de te verrichten handelingen in blokvorm zijn weergegeven.

Een berekening (optellen enz.) wordt daarbij aangegeven in de vorm van een rechthoek. Het lezen respectievelijk het ponsen wordt aangegeven in de vorm van een ponskaart.

Het branchen wordt aangegeven in de vorm van een ruit.

ad III. De zelfbesturing d.w.z., het door de computer nemen van logische beslissingen.

Wat verstaat men onder het nemen van logische beslissingen, het zgn. branchen? Dit kan het eenvoudigste aan de hand van een voorbeeld worden duidelijk gemaakt. Stel dat er een trein met goederenwagons vertrekt uit Breda in noorde-

lijke richting en stel dat iedere wagon gecodeerd is met een nummer dat ligt tussen 0 en 9. In Dordrecht aangekomen rijdt de trein over een wissel dat er zorg voor draagt dat alle wagons met code 0 naar de richting Geldermalsen rijden; de rest van de trein rijdt door. In Rotterdam gekomen rijdt de trein over een andere wissel. Alle wagons met code 1 gaan richting Gouda; de rest van de trein gaat door naar Schiedam. De wagons met code 2 gaan naar Hoek van Holland; de rest van de trein gaat door naar Den Haag enz. Hoewel er in Breda een complete trein vertrok komen de wagons, door de geplaatste wissels, op verschillende bestemmingen aan. Deze bestemming is afhankelijk van de codering van de wagons en de wissels. In feite komt het er op neer dat door deze wisselzetting stukken traject worden overgeslagen van het vooraf opgezette geheel van spoorlijnen.

In beginsel gebeurt bij een computer hetzelfde.

De ponskaarten of een ander medium worden alle ingevoerd in de computer.

Afhankelijk van de codering in deze ponskaarten is de uit te voeren berekening verschillend, omdat een ander traject van instructies wordt gevolgd, d.m.v. deze branche instructies (wisselzetting).

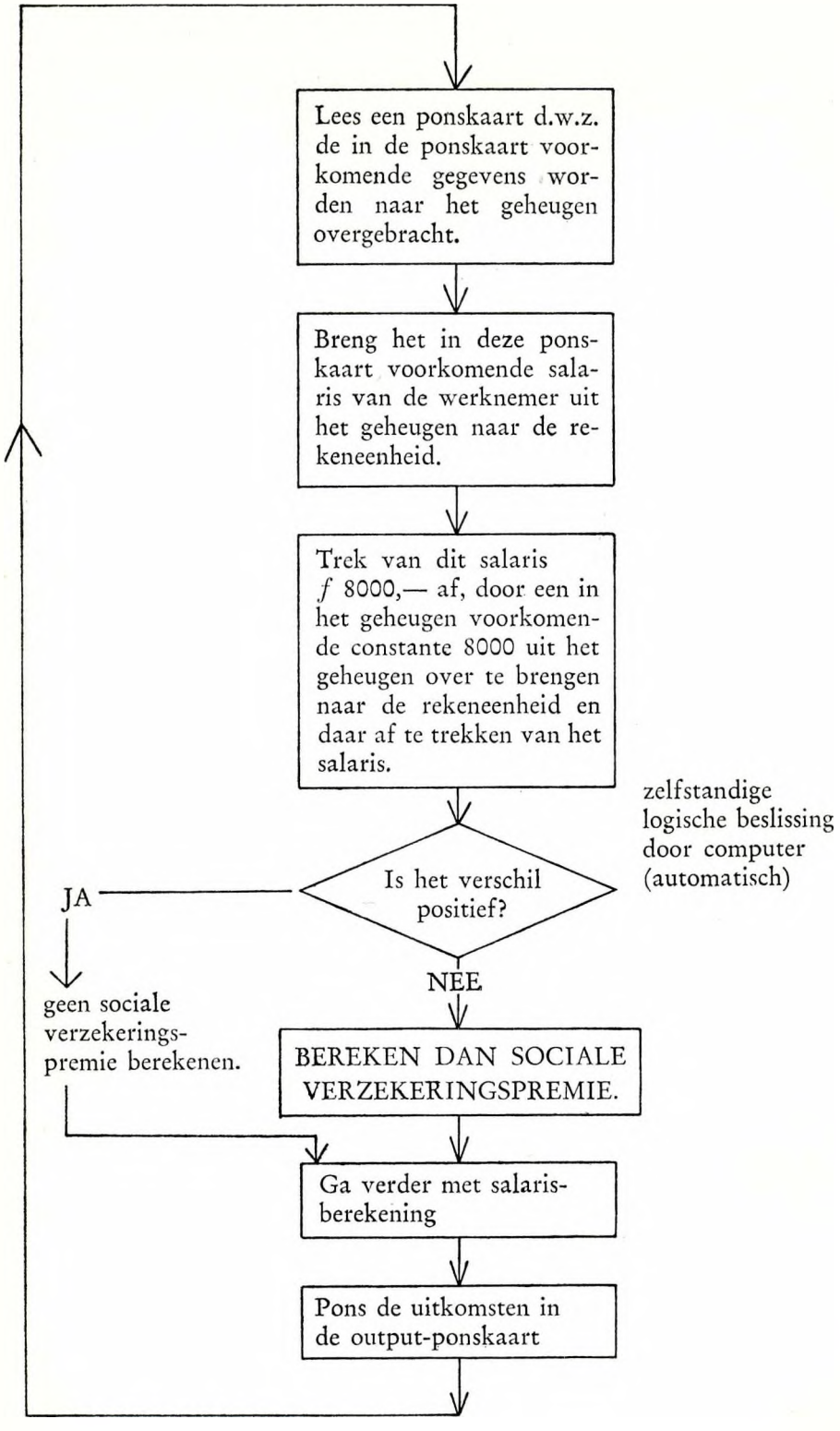
Daarmede zijn uiteraard de uitkomsten eveneens verschillend.

Voorwaarde hiervoor is, dat van de juiste codering gebruik gemaakt wordt en dat op het juiste moment, de juiste branche instructie geprogrammeerd wordt.

Onder branchen verstaat men het zetten van een wissel of beter een vraagstelling. De vraagstelling of aan een bepaalde voorwaarde voldaan is. Welke voorwaarde dit is, is een kwestie van coderingstechniek. De codering is dus een zeer belangrijk hulpmiddel waar alle zorg aan moet worden besteed. Door middel van deze mogelijkheid van branchen heeft de computer de mogelijkheid zelfstandig een keuze te doen uit een aantal mogelijkheden.

Als voorbeeld noemen wij een salaris-berekening. Indien een werknemer meer dan *f* 8000,— per jaar verdient behoeft geen sociale verzekeringspremie te worden ingehouden; bij *f* 8000,— salaris of minder dient wel sociale verzekeringspremie te worden ingehouden.

Het blokdiagram ziet er in dat geval als volgt uit:


Lees een ponskaart d.w.z. de in de ponskaart voorkomende gegevens worden naar het geheugen overgebracht.
Breng het in deze ponskaart voorkomende salaris van de werknemer uit het geheugen naar de rekeneenheid.
Trek van dit salaris f 8000,— af, door een in het geheugen voorkomende constante 8000 uit het geheugen over te brengen naar de rekeneenheid en daar af te trekken van het salaris.
zelfstandige logische beslissing door computer (automatisch)
Is het verschil positief?
JA
NEE
geen sociale verzekeringspremie berekenen.
BEREKEN DAN SOCIALE VERZEKERINGSPREMIE.
Ga verder met salarisberekening
Pons de uitkomsten in de output-ponskaart

Door middel van branching is het dus mogelijk bepaalde delen van het programma over te slaan en in het ene geval, afhankelijk van de code, wel s.v. premie te berekenen en in het andere geval niet.

Deze branching geeft aan de computer de mogelijkheid van zelfbesturing. Deze zelfwerkzaamheid geeft "the man in the street" wel eens de indruk dat de computer hersens heeft en zelfstandig denken kan. Niets is minder waar. De computer kan alleen datgene uitvoeren wat hem d.m.v. programmering geleerd is; de computer voert alleen instructies uit. De programmeur bepaalt wat de computer alzo kan uitvoeren en ontwerpt daarmede als het ware het spoorboekje. Naarmate in een programma meer branchings voorkomen wordt de zelfwerkzaamheid van de computer groter en krijgt het programma meer het karakter van een spoorboekje. Het blokdiagram waaiert uit. Doordat de computer zelfstandig logische beslissingen kan nemen, afhankelijk van de programmering, wordt volledige automatisering mogelijk in de uit te voeren berekeningen. In het ene geval wordt wel een berekening uitgevoerd nl. als het salaris *f* 8000,— of minder per jaar is. In het andere geval wordt geen berekening uitgevoerd nl. als het salaris groter is dan *f* 8000,—.

In een programma kunnen honderden „vraagstellingen (wissels)" voorkomen en de computer kan zelfstandig na iedere vraagstelling een logische beslissing nemen.

Het is niet verwonderlijk dat het feilloos nemen van honderden logische beslissingen wel eens de indruk wekt dat de computer „denkt" ook al is dit dan volgens „programma".

Afgezien van de zelfbesturing beschikt de computer over *zelfcontrole* waardoor de hoogste graad van betrouwbaarheid van de computer wordt verkregen. Door deze zelfcontrole wordt bereikt dat iedere berekening welke de computer uitvoert goed is. De mate van zelfcontrole varieert per soort en merk computer. Er zijn computers waarbij gecheckt wordt dat iedere letter of cijfer welke de computer d.m.v. de gekoppelde printer op papier afdrukt goed is. Dit is mogelijk ondanks de hoge afdruksnelheid van b.v. 50.000 letters per minuut. Ter vergelijking kan daartegenover de afdruksnelheid van een schrijfmachine worden gesteld van 400 letters per minuut.

Hoe ontvangt de computer nu zijn instructies? (m.a.w. hoe ziet het programma, zetsel eruit?)

Hiervoor zijn eigenlijk twee mogelijkheden, afhankelijk van de soort computer.

1. De eerste mogelijkheid is dat de instructies (in codeschrift) geponst worden in ponskaarten, de zgn. programmakaarten. In dat geval zal men, voor men met de computer gaat processen eerst de programmakaarten moeten invoeren in de computer. Met de eerste programmaponskaarten wordt het geheugen waarop op dat moment nog de instructies van het voorafgaande programma voorkomen, uitgewist. De instructies welke in de volgende programmaponskaarten voorkomen worden nu overgebracht naar het geheugen, waar ze verankerd blijven tot aan het moment dat een nieuw programma wordt opgeladen. Een programma bestaat dus uit een stapeltje ponskaarten. Nadat de inhoud van deze ponskaarten opgeladen is in het geheugen verloopt de besturing volgens deze ketting van instructies uit het geheugen.

2. De tweede mogelijkheid van het toedienen van instructies aan de computer is d.m.v. een schakelbord. Indien men de eerste mogelijkheid als *interne* besturing aanduidt zou men deze tweede mogelijkheid als *externe* besturing kunnen aanduiden. De uit te voeren instructies worden d.m.v. snoertjes geschakeld op het schakelbord waardoor contact gemaakt wordt en de uit te voeren instructie effectief wordt. Voor ieder programma heeft men dan een apart schakelbord nodig. Veelal komt de tweede vorm van instructietoediening (besturing) van de computer voor bij kleine computers met een klein geheugen.

Op welke plaatsen van het geheugen zijn de instructies achtereenvolgens opgeborgen?

Bij de behandeling van het geheugen hebben we erop gewezen dat er geheugens voorkomen welke bestaan uit een magneetkernen geheugen en geheugens welke bestaan uit een draaiende magnetische cylinder.

Bij beide soorten geheugens worden de instructies op verschillende plaatsen in het geheugen opgeborgen. De *volgorde* van de plaatsen waar de opeenvolgende instructies in het geheugen voorkomen verschilt.

Bij gebruik van het kernen geheugen liggen de achtereenvolgens uit te voeren instructies in diezelfde volgorde achter elkaar opgeslagen in het geheugen.

Indien het geheugen de vorm heeft van een draaiende magnetische cylinder wordt deze volgorde niet gebruikt. De reden daarvan is dat de gegevens met leesborstels afgelezen moeten worden van de draaiende cylinder. Daar deze leesborstels maar op één vaste plaats of enkele vaste plaatsen zitten en het draaien van de cylinder tijd kost, kan het zijn dat het benodigde gegeven (adres) op het moment dat dit opgeroepen wordt net de leesborstel al heeft gepasseerd. In dat geval moet er een cylinderomgang worden gewacht voor het betrokken gegeven ingelezen kan worden. Hierdoor wordt de snelheid zeer gedrukt en wordt een *lage optimaliteit* bereikt. Om dit te voorkomen wordt de plaats van de instructies in het geheugen zo gekozen (zo geprogrammeerd) dat iedere instructie juist onder de leesborstels gedraaid is op het moment dat deze nodig is. Door de plaats van de instructie zo te kiezen komen de opeenvolgende instructies zigzagsgewijs over de cylinder verspreid te liggen.

De volgorde van de instructies in het geheugen is op de eenvoudigste manier achter elkaar (sequentieel).

Op de volgende adresnummers van het geheugen zijn b.v. de volgende instructies opgeborgen:

Adresnr.	*instructie*	
	1001 1006	
1001-1006	15.0001	(optellen met de inhoud van adres 1 van het geheugen)
	1007 1012	
1007-1012	19.0003	(vermenigvuldigen met de inhoud van adres 3)
	1013 1018	
1013-1018	20.0005	(opbergen van uitkomsten in adres 5)

De instructies liggen achter elkaar of beter naast elkaar, opgeslagen in geheu-

gen adres 1001, 1007 e.v. Eerst wordt de instructie van adres 1001 uitgevoerd, daarna de instructie van adres 1007, daarna 1013 enz.

Indien het geheugen de vorm heeft van een *roterende* magnetische trommel (cylinder) zullen deze instructies i.v.m. de optimaliteit zigzagsgewijs over de trommel zijn verspreid. Ook al worden de instructies dan achter elkaar uitgevoerd, de plaats van de instructies in het geheugen is *niet* achter elkaar, doch zigzagsgewijs.

Bijvoorbeeld:

adresnr.	*instructie*
1000	15.0001
1500	19.0003
1300	20.0035

Het zal uit bovenstaand voorbeeld duidelijk zijn dat nu nog een *derde* aanduiding aan de instructie moet worden toegevoegd, nl. de aanduiding van het adres (de plaats in het geheugen) waar de volgende instructie is opgeborgen. De bovenstaande instructies luiden in dat geval:

adresnr.	*instructie*
1000	15.0001 . 1500
1500	19.0003 . 1300
1300	20.0035 . 1030
1030	enz.

De instructie is in dit geval uit drie delen opgebouwd:

1e. de operation code (wat moet ik doen)
2e. waarmee
3e. met welke plaats van instructie moet ik verder.

Nu blijkt nog duidelijker dat de instructies schalmen van een „ketting van instructies" zijn. Het einde van de voorgaande instructie is bevestigd aan de volgende instructie.

Indien men het bovenstaande voorbeeld zou willen completeren met de instructies voor het lezen van een kaart en die voor het ponsen dan ziet dit voorbeeld er als volgt uit:

adresnr.	*instructie*	
0750	70.0701.0020	lees een nieuwe ponskaart in
0020	 0800	) breng de gegevens uit die ponskaart naar
0800	 1000	) adres 1 en adres 3
1000	15.0001.1500	
1500	19.0003.1300	
1300	20.0035.1030	
1030	71.0035.0750	pons een ponskaart uit waarin de inhoud van adres 35 o.a. voorkomt.

De laatste instructie moet worden opgevolgd door de instructie 0750 d.i. de eerste instructie voor het lezen van een nieuwe ponskaart. Hiermede is dus de kringloop gesloten.