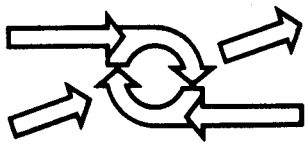
Multiprocessorsystemen

Dings, P.J.F.; Janssen, M.M.H.M.; Köllner, E.

Award date:
1976



GROEP METEN EN REGELEN
AFDELING DER TECHNISCHE NATUURKUNDE
T.H.Eindhoven, Postbus 513
Telefoon 040 - 472885

Multiprocessorsystemen 1

P. Dings

M. Janssen

E. Köllner

Rapport van het afstudeerwerk, verricht onder leiding van Ir. G. Dekker.

Met dank aan Ineke Köllner voor haar bijdrage in het typewerk.

November 1975

SAMENVATTING

IN de groep Meten en Regelen is een inleidende studie gemaakt van de mogelijkheden die multiprocessor-systemen bieden bij de regeling van processen.
Het onderzoek heeft zich gericht op mogelijke opzetten van multiprocessorsystemen.
Er wordt een beschrijving van de hierin voorkomende hardware elementen met de problemen, die ontstaan bij de koppeling van deze elementen.
Verder is onderzocht, wat de funkties zijn van het besturingssysteem.

INHOUDSOPGAVE

VERKLARENDE WOORDENLIJST

Absolute adressering	Adresseringsmethode waarbij het werkelijke adres* in de instruktie* staat aangegeven.
Accesstijd	De tijd die nodig is om data* in of uit een geheugen* of randapparaat* te krijgen.
Accumulator	Een algemeen register* dat gebruikt wordt bij rekenkundige en logische bewerkingen.
Achtergrondgeheugen	Een geheugen* dat een grotere kapaciteit heeft dan het werkgeheugen* en een grotere accesstijd*.
Adres	Een getal dat registers*, geheugenplaatsen, randapparatuur etc. identificeert.
ALU	Deel van een processor* waarin rekenkundige en logische bewerkingen worden uitgevoerd.
Anoniem systeem	Centraal systeem*, waarbij zowel het besturingsprogramma alsook de taken dynamisch worden toegewezen.
Array processing	Vorm van multiprocessing, waarbij alle processoren tegelijkertijd dezelfde bewerking uitvoeren op een verzameling van dataelementen, die met elkaar verband houden. Iedere processor* neemt hierbij één element van deze verzameling voor zijn rekening.
Asynchroon	Een manier van datatransmissie waarbij de ene gebeurtenis wordt gestart door beëindiging van de ander
Besturingssysteem	Een verzameling van programma's en regels die gemaakt zijn om alle mogelijkheden van een computer* volledig te benutten.
Bit	Binairy digit. Binair getal dat de waarden 0 en 1 kan aannemen.
Buffer	(1) Geheugen* dat gebruikt wordt om verschillen in snelheid van informatiestroom te kompenseren. (2) Electronische schakeling die voor de aanpassing tussen twee circuits zorgt.
Bus	Een gemeenschappelijke weg waarlangs informatie kan stromen.
Byte	Een aantal bit* dat als eenheid wordt beschouwd; meestal acht bit.

Centraal systeem	Zie bladzijde ?.3.
Computer(systeem)	Een apparaat dat in staat is om informatie op te nemen, hierop van te voren gedefinieerde bewerkingen uit te voeren en de resultaten hiervan af te geven.
Computer netwerk	Een aantal onafhankelijke computers* die met elkaar verbonden zijn via een communicatiekanaal. Elke computer* kan zelfstandig werken onder zijn eigen besturingssysteem*, of deelnemen aan netwerkaktivitei
CPU	Central processing unit; eenheid die rekenkundige en logische bewerkingen uitvoert en de interpretatie en uitvoering van instrukties* controleert.
Daisy chain	Een cascade verbinding waarin de plaats bepalend is voor de prioriteit*.
Data	Elke vorm van informatie, symbolen of getallen, die door een computer* kan worden gebruikt.
Deadlock	Toestand waarbij twee of meer processoren*(programma's) eeuwig op elkaar blijven wachten. De ene blijft wachten tot de andere een kritische sektie* verlaat.
Decentraal systeem	Zie bladzijde ?.3.
Decoder	Een schakeling die één uitgang bekrachtigt, als reaktie op een kombinatie van ingangssignalen.
Device	(Rand)apparaat*.
Directe adressering	Zie absolute adressering*.
Disk	Een magnetisch geheugen*, bestaande uit een aantal schijven, die bedekt zijn met een laag magnetiseerbaar materiaal.
Dispatching	Het toewijzen van taken* aan processoren*.
DMA	Direct memory access; rechtstreekse toegang tot het werkgeheugen* van een computer* door een randapparaat*.
Eén-adres-machine	Computer* waarbij per instruktie* maximaal één operand kan worden aangegeven.
Encoder	Een schakeling die signalen in gecodeerde digitale vorm brengt.

EROS	Experimenteel real time* operating systeem*, ontwikkeld in groep VMR.
Executive	Een onderdeel van EROS; de term wordt vaak gebruikt als synoniem voor besturingssysteem*.
FIFO	first in, first out.
Flip-flop	Een schakeling, die twee stabiele toestanden kent.
Floppy disk	Een disk* met verwisselbare, flexibele schijven.
Geheugen	Een opslagplaats voor informatie.
Geheugencyclus	Een opeenvolging van gebeurtenissen, nodig om één geheugenwoord te lezen.
Gekoppeld systeem	Een systeem van onafhankelijke computers*, die met elkaar verbonden zijn via een gemeenschappelijk achtergrondgeheugen*.
Hardware	De totale apparatuur, waaruit een computer* is opgebouwd. Ook: door middel van apparatuur.
Hardware konflikt	Konfliktsituatie, die ontstaat wanneer twee of meer gebruikers tegelijkertijd van hetzelfde apparaat (geheugen*) gebruik willen maken.
HOLD	Een toestand van de INTEL 8080, waarbij data*- en adres*lijnen zweven.
I.C.	Integrated circuit.
Indexeren	Het aanpassen van een instruktie*-adres* aan de momentane plaats van het programma in het werkgeheugen*.
Indirekte adressering	Een adresseringsmethode, waarbij het adresgedeelte van de instruktie* verwijst naar een andere geheugenplaats, die het gewenste adres* bevat.
Instruktie	Een gedeelte van een computerprogramma, dat de computer* vertelt, welke bewerking hij op dat moment moet uitvoeren.
Interface	De plaats, waar twee systemen of subsystemen aan elkaar gekoppeld zijn.
Interruptie	Een onderbreking van de lopende aktiviteiten van de processor* door een extern apparaat.
Interrupt handler	Een subroutine* om interruptes te verwerken.

I/O	Input/output.
Jobhead	Een lijst met gegevens ten behoeve van een taak* in EROS.
Joblist	Een gedeelte van de jobhead.*
Kanaal	Een weg, waarlangs signalen kunnen worden verzonden.
Kerngeheugen	Een geheugen*, bestaande uit magnetiseerbare ringetjes, vaak gebruikt als werkgeheugen.*
Kritische sektie	Een geheugensegment, waarvan te allen tijde slechts één processor* of programma gebruik mag maken.
Latch	Een register*, dat de ingangsdata overneemt en vasthoudt.
LIFO	Last in, first out.
Lock/unlock	Een mechanisme om de toegang tot gemeenschappelijke faciliteiten te regelen.
Loop	Een reeks instrukties*, die herhaaldelijk wordt uitgevoerd, tot voldaan is aan een vooraf gestelde voorwaarde.
L.S.I.	Large scale integration.
Machinecyclus	Het tijdsinterval, waarin een computer* een bepaald aantal operaties kan uitvoeren.
Massageheugen	Zie achtergrondgeheugen.*
Microcomputer(systeem)	Computer(systeem)* rond één of meerdere microprocessoren.*
Microprocessor	CPU*, geïntegreerd in één of meerdere L.S.I.* I.C.'s.*
MOS	Metal oxide semiconductor (FET met geïsoleerde gate)
Multiplexen	Ruimtelijk: mogelijkheid om één kanaal* naar willekeur te verbinden met één uit meerdere kanalen. In de tijd: zie time sharing.*
Multiprocessorsysteem	Een computersysteem*, bestaande uit een aantal processoren*, die samenwerken onder één besturingssysteem.* De mate van integratie tussen de systeemelementen is erg groot. Alle processoren* hebben toegang tot een gemeenschappelijk voorgrondgeheugen* en I/O-apparaten. (zie par.1.1)
Multiprogrammeren	Een manier om verschillende programma's quasi-simultaan te laten verwerken door één processor.*

Netwerk	Zie computernetwerk*.
Operating systeem	Zie besturingssysteem.*
Parallel processing coëfficiënt	Een getal, dat de mate van parallelle verwerking aangeeft.
Peripheral	Zie randapparaat*.
Pipe lining	Een methode, waarbij iedere processor* herhaaldelijk dezelfde bewerking uitvoert op verschillende elementen uit één enkele datastroom. Het resultaat van de bewerking van de ene processor* is ingangsgegeven voor de volgende processor*.
Pointer	Een wijzer om een bepaalde geheugenplaats aan te geven.
Polling	Een procedure om sequentieel de status van een aantal apparaten te testen teneinde te bepalen, welke apparaten om aandacht vragen.
Prioriteit	Een getal om de relatieve belangrijkheid aan te geven.
Procescomputer	Een computer*, die gebruikt wordt bij de regeling van processen.
Processor	Algemeen: ieder apparaat dat in staat is om bewerkingen uit te voeren op data*. De term wordt meestal gebruikt als synoniem voor CPU
Processorcyclus	Deze term wordt gebruikt als synoniem voor machinecyclus* om het verschil met geheugencyclus* te accentueren. Zie verder machinecyclus*.
Processorstatus	De inhoud van de processorregisters.
Program counter	Het register*, dat het adres* bevat van de volgende te halen instruktie*.
Programmastatus	De resultaten, die in de loop van het programma zijn verkregen.
PROM	Programable read only memory
RAM	Random access memory
Randapparaat	Een apparaat, dat kan werken onder besturing van een computer*.
Random access	De toegang tot een volgende adres* is onafhankelijk van het adres*, dat momenteel wordt gerefereerd.

Ready/wait faciliteit	Faciliteit van de INTEL 8080 om de snelheid van de processor* aan te passen aan de snelheid van het geheugen*.
Real time	De eis, dat een programma zo snel wordt uitgevoerd, dat de resultaten nog bruikbaar zijn.
Re-entrant subroutine	Een subroutine*, die zichzelf niet verandert en geen red/werkgebied in zichzelf bevat. De routine* kan daardoor door verschillende gebruikers gelijktijdig worden benut.
Register	Een speciale geheugenplaats met een capaciteit, die meestal gelijk is aan de woordlengte van de computer*.
Register-indirekte adressering	Een adresseringsmethode, waarbij het te refereren geheugenadres in een processorregister staat.
Relocatable programma	Een programma, dat op elke plaats in het geheugen* kan 'draaien'.
Resource	Hulpbron
Responsietijd	De tijd, die verloopt tussen het moment van aanvraag en het moment, waarop met de uitvoering wordt begonnen.
RESTART	Speciale instrukties* van de INTEL8080, die gebruikt worden om een vectored* interrupt* te creëren.
ROM	Read only memory
Round robin	Een strategie, waarbij potentiële gebruikers cyclisch worden afgetast.
Routine	Zie subroutine.*
Scheduling	Het opstellen van een lijst van uit te voeren werkzaamheden (werklijst*).
Software	De programmatuur in een computersysteem*. Ook: door middel van programmatuur.
Software konflikt	Een konfliktsituatie, die ontstaat, wanneer een processor*(programma) gebruik wil maken van een kritische sektie*.
Stack	stapelgeheugen, waarbij opeenvolgende geheugenwoorden op LIFO*-basis gerefereerd kunnen worden.
Stackpointer	Een wijzer, die de geheugenplaats, volgende op de laatst gevulde geheugenplaats, aangeeft.

Status-informatie	De signalen, die de externe schakellogica informeren over het soort machinecyclus*, dat zal worden doorlopen.
Subroutine	Een gedeelte van een programma, dat een gesloten eenheid vormt en in het algemeen vaker wordt gebruikt.
Symmetrisch systeem	Zie anoniem systeem.*
SYNC	Een signaal, dat bij de INTEL 8080 het begin van een machinecyclus* markeert.
Systeemfunktie	Een bepaalde funktie van het besturingssysteem.*
Systeemtaak	Een taak*, die een bepaalde systeemfaciliteit uitvoert ten behoeve van één of meerdere gebruikers.
Synchroon	Een manier van datatransmissie, waarbij de transmissiesnelheid tussen de apparaten wordt bepaald door gebeurtenissen in de rest van het systeem.
Taak	Een groep instrukties*, die samen een afgeronde hoeveelheid werk vormen.
Three state (tri-state) uitgangen	Uitgangen, die een zwevende toestand kennen, dat wil zeggen dat de uitgang noch logisch 1 noch logisch 0 is.
Time-sharing	Een manier, waarbij meerdere gebruikers concurrerend een gemeenschappelijke faciliteit kunnen gebruiken.
Toestand	Een term, die bij de INTEL 8080 wordt gebruikt om het tijdsinterval tussen twee opeenvolgende klokpulsen aan te geven.
TTL	transistor - transistor - logic.
Uniprocessorsysteem	Een computersysteem*, waarin één enkele processor* werkzaam is.
Vectored interrupt	Een mechanisme, waarbij interrupties* automatisch geïdentificeerd worden.
Vector processing	Zie array-processing.*
Vlag	Een éénbits register*, dat gebruikt wordt om de toestand van een apparaat aan te geven.
Voorgrondgeheugen	Het geheugen* van de computer*, dat direkt toegankelijk is.

Werkgeheugen	Zie voorgrondgeheugen.*
Werklijst	Lijst van uit te voeren werkzaamheden.
Woord	Het aantal bits,* dat kan worden opgeslagen in één geheugenplaats en door de computer* wordt behandeld als een eenheid.

INLEIDING

Computersystemen, waarin meerdere processoren werkzaam zijn, kunnen we verdelen in:

- computernetwerken
- gekoppelde systemen
- multiprocessorsystemen

Netwerken bestaan uit een aantal onafhankelijke computers, die met elkaar verbonden zijn via een communicatiekanaal. Elke computer kan zelfstandig werken onder zijn eigen besturingssysteem of kan deelnemen in een netwerkaktiviteit. De mate van koppeling is gering.

In gekoppelde systemen zijn de deelnemende computers ook onafhankelijk. De mate van koppeling is sterker dan bij de netwerken, omdat de computers hier met elkaar verbonden zijn via een gemeenschappelijk achtergrondgeheugen. Omdat meerdere processoren toegang hebben tot dit achtergrondgeheugen, moet de toegang worden gecoördineerd met behulp van een lock/unlock mechanisme.

Een multiprocessorsysteem bestaat uit een aantal processoren, die samenwerken onder één besturingssysteem. De mate van integratie tussen de systeemelementen is in tegenstelling tot de vorige systemen erg groot. Alle processoren hebben toegang tot een gemeenschappelijk voorgrondgeheugen en I/O-apparatuur.

In dit verslag beperken we ons tot een beschrijving van multiprocessorsystemen.

Bij een indeling van multiprocessorsystemen kunnen we uitgaan van het niveau, waarop parallel wordt gewerkt. Parallel werken kan gebeuren op:

- elementair niveau
- funktioneel niveau

Parallel werken op elementair niveau houdt in, dat meerdere processoren gelijktijdig bewerkingen uitvoeren op dezelfde datastroom. We krijgen hierbij de volgende mogelijkheden:

1. Alle processoren voeren gelijktijdig dezelfde bewerking uit op een verzameling van data-elementen, die met elkaar verband houden. Iedere processor neemt hierbij één element uit de datastroom voor zijn rekening (vektor- of array-processing: zie fig.1).

. Iedere processor voert herhaaldelijk dezelfde bewerking uit op verschillende elementen van een enkele datastroom. Het resultaat van de bewerking van de ene processor is ingangsgegeven voor de volgende processor (pipe lining: zie fig.2).

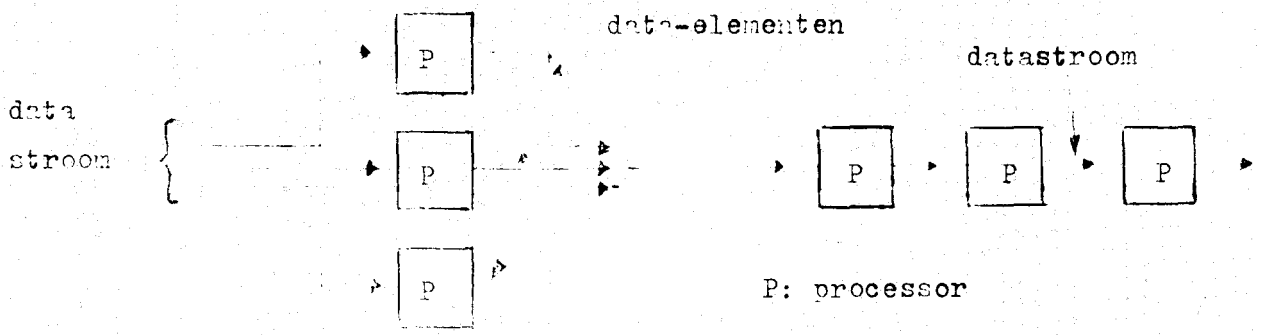


fig.1 vektor- of array-processing

fig.2 pipe lining

Parallel werken op elementair niveau wordt verder niet in beschouwing genomen.

Parallel werken op funktioneel niveau betekent, dat de programmatuur wordt verdeeld in funktionele delen, die elk een min of meer zelfstandige eenheid vormen. De aanwezigheid van meerdere processoren biedt de mogelijkheid om de delen parallel uit te voeren.

In het verslag wordt verder steeds uitgegaan van multiprocessorsystemen, waarbij parallel wordt gewerkt op funktioneel niveau.

Allereerst worden, uitgaande van de definitie van een multiprocessorsysteem, de hierin voorkomende hardware elementen beschreven met de problemen, die ontstaan bij de koppeling daartussen (hoofdstuk 1).
Vervolgens wordt een beschrijving gegeven van een aantal systeemopzetten, waarvan een enkele nader wordt uitgewerkt (hoofdstukken 2 en 5).
Een ander aspekt is het besturingssysteem. Er zal worden onderzocht, wat de funkties zijn van het besturingssysteem. Hierbij wordt uitgegaan van het besturingssysteem van een uniprocessorsysteem (hoofdstukken 3 en 6).
De verschillende manieren van het gemeenschappelijk gebruik van geheugens worden beschreven in hoofdstuk 4.
Ten slotte wordt een aantal konklusies uit het werk gepresenteerd (hoofdstuk 7).

HOOFDSTUK 1 HET MULTIPROCESSORSYSTEEM

1.1 Definitie van een multiprocessorsysteem

Onder een multiprocessorsysteem verstaan we een computersysteem dat voldoet aan de volgende eisen:

- Er zijn twee of meer processoren. Een processor of CPU (Central Processing Unit) is een eenheid, die rekenkundige bewerkingen uitvoert en de interpretatie en uitvoering van instructies controleert. Wij eisen bovendien dat alle processoren gelijkwaardig zijn.
- Alle processoren hebben toegang tot een gemeenschappelijk hoofdgeheugen. Daarnaast kan iedere processor de beschikking hebben over een kleine hoeveelheid privégeheugen.
- Alle processoren hebben tenminste een gedeelte van de input/output gemeenschappelijk (kanalen, besturingseenheden, randapparatuur).
- Het systeem staat onder controle van één enkel operating systeem (besturingssysteem), dat de supervisie heeft over alle hardware en software.
- Er is zowel hardware als software interaktie mogelijk.

Uit deze definitie volgt dat er hardware elementen aan elkaar gekoppeld moeten worden. In het resterend gedeelte van dit hoofdstuk zullen we deze elementen allereerst in het kort behandelen en vervolgens de problemen bespreken, die optreden wanneer deze elementen aan elkaar gekoppeld worden. Hierbij zal de grootste nadruk worden gelegd op de koppeling tussen processor en geheugen: het gemeenschappelijk gebruik van geheugen door een aantal processoren.

1.2 Hardware elementen binnen een multiprocessorsysteem

1.2.1 Inleiding

In een multiprocessorsysteem onderscheiden we een drietal zogenaamde hardware elementen, te weten:

- Processoren
- Geheugens
- Input/output (I/O) apparatuur.

Deze elementen zullen we in de nu volgende paragrafen in het kort behandelen.

1.2.2 De processor

Als processor is gekozen voor de 8080 van INTEL: een 8-bit CPU (Central Processing Unit), geïntegreerd in een enkel NMOS I.C.
De 8080 bevat:

- zes 8-bit dataregisters (B,C,D,E,H en L) die ook als registerpaar gebruikt kunnen worden. Het registerpaar H&L wordt gebruikt bij register-indirecte instrukties.
- Een 8-bit rekenregister (de Accumulator A) .
- Vijf conditieflipflops (Carry, Zero, Sign, Parity en Auxilliary Carry).
- Een tijdelijk registerpaar (W & Z) dat niet adresseerbaar is.
- Een 16-bit program counter (PC), waarmee 65536 geheugenadressen kunnen worden gerefereerd.
- Een 16-bit stack pointer (SP). Via deze SP kan ieder willekeurig gedeelte van het geheugen gebruikt worden als last-in-first-out (lifo) stack. Er kunnen ook meerdere stacks geopend worden. Op de stack kunnen behalve terugspringadressen van subroutines , ook alle dataregisters, de accu en conditieflipflops gered worden.
- Een 8-bit parallel rekenelement (Arithmetic Logic Unit, ALU).

De communicatie met de buitenwereld geschiedt via een 8-bit databus, een 16-bit adresbus en een aantal besturingslijnen, die in tegenstelling tot de 8008 grotendeels gedecodeerd zijn. Data en adres kunnen dus parallel door de processor worden uitgezonden. Hierdoor kan de 8080 aanzienlijk sneller werken dan de 8008, die voor adres en data gebruik maakt van een gemultiplexte databus. Dit komt tot uitdrukking in de executietijd, die voor een zogenaamde non-memory-referencing instruktie

2 microsec. bedraagt bij de 8080 en 20 microsec. bij de standaard 8008.
De belangrijkste eigenschappen van de 8080 zijn verder:
- Het is een één-adres-machine. Rekenen op de stack is niet mogelijk.
- De instruktieset bestaat uit 74 basisinstrukties (waaronder alle 48 instrukties van zijn voorganger :de INTEL 8008).
- De 8080 kent acht interruptieniveaus (vektorinterrupt).
- Direct memory access (DMA) is mogelijk.
- De CPU kan werken met geheugens met een willekeurige snelheid.
- De adresseringsmethode kan zijn:
  - direkt, het adres zit opgesloten in byte 2 en 3 van de instruktie.
  - register-indirekt, het geheugenadres staat in een (intern) registerpaar.
  - "immediate", de data staat in het volgende geheugenadres.

1.2.3 Geheugens

Geheugens kunnen onderverdeeld worden in:
- Voorgrondgeheugens of werkgeheugens
- Achtergrondgeheugens of massageheugens.

De achtergrondgeheugens, zoals magnetische schijfgeheugens ((floppy) disks), kunnen beschouwd worden als periferals, en zullen in deze paragraaf dan ook buiten beschouwing worden gelaten.
Als werkgeheugen komen in aanmerking:
- Magnetische kerngeheugens
- Electronische geheugens, ook halfgeleidergeheugens genoemd.

Halfgeleidergeheugens zijn flexibel en passen het best bij een microprocessor. Wat de werkgeheugens betreft zullen we ons beperken tot de halfgeleidergeheugens, welke geheugens op hun beurt weer onderverdeeld kunnen worden in;
- Read only memories (ROM)
- Read/Write memories.

Read only memories zijn onder te verdelen in:
- custom-programmed ROM's. Deze ROM's worden door de fabrikant geprogrammeerd volgens een door de klant opgegeven bitpatroon. Deze geheugens zijn alleen economisch verantwoord bij gebruik van grote aantallen.
- programmable ROM's (PROM) . Deze geheugens kunnen eenmalig door de gebruiker geprogrammeerd worden.

- erasable and (re)programmable ROM's (EPROM of RPROM). Deze geheugens kunnen door middel van ultra-violet licht gewist worden en vervolgens opnieuw worden geprogrammeerd. De gebruiker kan hierin dus vaste programmatuur opslaan en in een later stadium eventueel veranderen. Deze vaste programma's kunnen bijvoorbeeld opstartroutines zijn, of delen van het operating systeem. Van de PROM's zijn voor ons alleen deze EPROM's van belang. Wanneer we in het vervolg spreken over PROM's, zullen we steeds EPROM's bedoelen.

Read/Write memories, meestal RAM genoemd (Random Access Memory), kunnen we onderverdelen in:

- statische RAM's
- dynamische RAM's.

Statische RAM's behouden in tegenstelling tot dynamische RAM's hun informatie zolang echter de voedingsspanning is aangesloten. Bij de dynamische RAM's vindt de informatieopslag plaats in de vorm van electrische lading. Om te voorkomen dat opgeslagen informatie verloren gaat moet een dynamische RAM periodiek "ververst" worden.Dit verversen vindt plaats bij een leesoperatie.

We beperken ons wat de RAM betreft tot de statische RAM.Een achtergrondgeheugen zoals een floppy disk om belangrijke informatie voor langere tijd te bewaren (ook bij netspanningsstoringen belangrijk), is naast een of meerdere ROM's (voor opstarten e.d.) onmisbaar bij gebruik van electronische geheugens.

Statische RAM's zijn verder nog onder te verdelen in:

- Bipolaire RAM's
- MOS RAM's (MOS = Metal Oxide Semiconductor).

Bij bipolaire RAM's worden "gewone" (bipolaire) transistoren als aktief element gebruikt. MOS RAM's bestaan uit geïntegreerde MOSFET's. De voordelen van MOS RAM's boven bipolaire RAM's zijn:

- De bitdichtheid is groter.
- De vermogensdissipatie per bit is geringer.
- De prijs per bit is lager.

Hoewel de accesstijd van MOS RAM's groter is dan die van de bipolaire RAM's , beperken we ons voorlopig tot de MOS RAM.

Op de statische MOS RAM wordt nader ingegaan in appendix B

1.2.4 Input/output apparatuur

Voor de communicatie tussen de computer en de buitenwereld (de operateur, het proces) zijn een aantal input/output (I/O) apparaten onmisbaar. Voorbeelden hiervan zijn: teletypes, keyboards, displays, bandponsers, bandlezers, disks, A/D omzetters etc. In een multiprocessorsysteem kan bovendien de ene processor als I/O apparaat voor de andere beschouwd worden.

Gezien vanuit de processor bestaat ieder I/O device uit een aantal registers, die met een inputinstruktie kunnen worden ingelezen en met een outputinstruktie worden volgeschreven. De door een processor uitgezonden informatie kan door een I/O device geïnterpreteerd worden als data of als kommando.

Sommige registers hebben een bijzondere funktie: de zogenaamde "vlaggen". Een vlag is een één-bit-register dat de status van een bepaald randapparaat aangeeft. Al naar gelang de funktie van het apparaat zal de vlag gezet worden als het apparaat gereed is om informatie te ontvangen, of als het apparaat data heeft geproduceerd.

Vlaggen zijn noodzakelijk in verband met de synchronisatie van het randapparaat met de processor. Verder zij nog opgemerkt dat sommige apparaten funktioneel gezien uit meerdere apparaten bestaat. Zo bestaat een teletype bijvoorbeeld uit een keyboard en een printer.

1.3 Koppeling van de hardware elementen

1.3.1 Inleiding

In deze paragraaf zullen we ons bezig houden met de problemen die ontstaan wanneer de in par. 1.2 genoemde hardware elementen (processoren, geheugens en randapparatuur) aan elkaar gekoppeld worden. Hierbij zal het probleem tengevolge van het gemeenschappelijk gebruik van geheugens centraal staan.

Achtereenvolgens zullen de volgende koppelingen beschouwd worden:

- processor —— geheugen
- processor —— randapparaat
- geheugen —— randapparaat.

1.3.2 Koppeling van processoren aan geheugens

Een algemeen probleem bij de uitwisseling van informatie tussen twee apparaten is de "timing" van het datatransport.

Meestal zullen de snelheden waarmee de apparaten informatie kunnen uitwisselen verschillen.
Het timing-probleem tussen processor en geheugen wordt bij de 8080 processor opgelost door gebruik te maken van de ready/wait faciliteit van de CPU. Zie appendix A en lit. 7

Behalve het timing-probleem ontstaan in een multiprocessorsysteem coördinatie-problemen, bijvoorbeeld bij de koppeling van processoren aan geheugens.
In de eerste plaats ontstaan er konflikten wanneer twee of meer processoren tegelijkertijd van dezelfde geheugeneenheid gebruik willen maken. Ze zijn het gevolg van de hardware organisatie en het adresseringsmechanisme van het geheugen. Zoals in fig. 1 van appendix B te zien is, verschijnt het minst significante deel van een processoradres (in het voorbeeld A_7 t/m A_0) op alle bouwstenen waaruit het geheugen is opgebouwd. Tevens zijn de overeenkomstige datalijnen van alle bouwstenen doorverbonden. Wanneer twee of meer processoren tegelijkertijd een adres aanbieden aan dezelfde geheugeneenheid ontstaan er dus konflikten, door ons hardware geheugenkonflikten genoemd.

Wanneer de hardware geheugenkonflikten zijn opgelost kunnen er nog konflikten ontstaan tengevolge van het feit dat meerdere processoren toegang hebben tot zogenaamde kritische geheugensekties. Stel bijvoorbeeld dat twee processoren A en B beide toegang hebben tot een tabel waarin de uit te voeren taken staan aangegeven (werklijst).
Er kan nu het volgende gebeuren.
Processor A, op zoek naar een nieuwe taak, vindt op een bepaald moment dat taak t aan de beurt is om uitgevoerd te worden. Onmiddellijk hierna krijgt processor B toegang tot de werklijst en vindt ook taak t, nog voordat processor A taak t heeft kunnen afvoeren van de werklijst. Het gevolg is dat zowel A als B aan taak t beginnen.
Zo'n werklijst is dus een kritische geheugensektie. De konflikten die ontstaan wanneer twee of meer processoren toegang proberen te krijgen tot een kritische geheugensektie, zullen we software geheugenkonflikten noemen, omdat de oorzaak van de konflikten is gelegen in de software van het systeem.

1.3.3 Hardware geheugenkonflikten

Hardware geheugenkonflikten ontstaan wanneer twee of meer processoren tegelijkertijd toegang proberen te krijgen tot hetzelfde geheugen. Om deze konflikten te voorkomen moet de toegang tot ieder geheugenblok gecoördineerd worden.De rol die de processoren spelen in de oplossing van het coördinatie-probleem kan zijn:

- aktief
- passief.

In het eerste geval coördineren de processoren zélf de toegang tot het gemeenschappelijk geheugen door vóór een referentie aan dit geheugen te testen of het vrij is. Hiervoor moet iedere processor een zekere hoeveelheid privégeheugen bezitten. Bij de bespreking van de software geheugenkonflikten zal blijken dat het moeilijk is om processoren zelfstandig de toegang tot kritische sekties te laten coördineren. Omdat bovendien bij iedere referentie aan het gemeenschappelijk geheugen hardware geheugenkonflikten kunnen optreden is het ondoenlijk om deze konflikten door de processoren zélf te laten oplossen.

De rol die de processoren spelen bij de oplossing van het coördinatieprobleem zal dan ook passief zijn, dit wil zeggen dat de hardware geheugenkonflikten worden opgelost door een aparte module, die we GEHCON (GEHeugen CONtroller) zullen noemen.

Om eenduidig vast te kunnen stellen welke processor op een bepaald moment toegang krijgt tot het gemeenschappelijk geheugen zijn spelregels (prioriteitsregels) noodzakelijk. Mogelijke prioriteitsregels zijn:

- first-in-first-out (fifo). Aanvragen worden in volgorde van binnenkomst behandeld.
- last-in-first-out (lifo). De volgorde van afhandeling is tegengesteld aan de volgorde van aanvrage.
- round robin. Alle potentiële aanvragers worden cyclisch afgetast.
- vaste prioriteit. De volgorde van afhandeling wordt bepaald door de prioriteit van de aanvrage. Deze prioriteit kan statisch of dynamisch zijn. We spreken van statische prioriteit als de prioriteitsvolgorde alleen door de operateur veranderd kan worden (veranderen van aansluiting, wijzigen van de prioriteitentabel etc.). De prioriteit is dynamisch als het systeem zelfstandig de prioriteitsvolgorde kan veranderen.

De keuze van de prioriteitsregel hangt nauw samen met de keuze tussen hardware/software GEHCON. De laatstgenoemde keuze is afhankelijk van

het niveau waarop processoren toegang kunnen krijgen tot het gemeenschappelijk geheugen. We spreken hier van
- microniveau en
- macroniveau.

Met microniveau bedoelen we dat de toegang tot het gemeenschappelijk geheugen steeds wordt geregeld voor het lezen of schrijven van één processorwoord (byte).
Onder macroniveau verstaan we duurzamere koppelingen tussen processor en geheugen (bijvoorbeeld gedurende de tijd die nodig is om een bepaald programma te doorlopen).

Wanneer er op microniveau wordt geschakeld, is gezien de schakelfrequentie alleen een hardware GEHCON bruikbaar.
De prioriteitsregels fifo en lifo zijn in hardware moeilijker te implementeren dan round robin of vaste prioriteit. We beperken ons hier tot de twee laatstgenoemde prioriteitsregels. Vaste prioriteit kan in hardware worden gerealiseerd door gebruik te maken van een speciaal TTL I.C. (priority encoder, b.v. SN 74148) of door "daisy chaining" toe te passen. **Zie appendix** C . In deze appendix is tevens een hardware round robin gegeven.
De responsietijd op een aanvrage is bij de hier genoemde hardware prioriteitsstrategiën zeer klein (vaste prioriteit: ca. 50 nsec. ; round robin: ca. n x 20 nsec., n = aantal potentiële aanvragers).
Wanneer vaste prioriteit wordt aangehouden, kan het voorkomen dat gebruikers met een zeer lage prioriteit nooit toegang krijgen tot het gemeenschappelijk geheugen vanwege de grote vraag van gebruikers met een hogere prioriteit. Dit kan deadlock veroorzaken.
Totale blokkering van één of meer processoren kan worden voorkomen door telkens groepjes van aanvragen te verwerken. Pas als alle aanvragen van een groep zijn verwerkt (in een vaste prioriteitsvolgorde), wordt een nieuwe groep gevormd, bestaande uit alle aanvragen die er op dat moment zijn. Zie appendix C . Deze methode paart een korte responsietijd aan gelijkberechtiging.

Wanneer er op macroniveau wordt geschakeld, is zowel een hardware als software GEHCON bruikbaar. Een software GEHCON verdient o.i. dan de voorkeur, o.a. vanwege de flexibiliteit en de mogelijkheid om een optimale toewijzingsstrategie toe te passen.

1.3.4 Software geheugenkonflikten

Software geheugenkonflikten ontstaan wanneer een processor toegang probeert te krijgen tot een zogenaamde kritische geheugensektie (zie 1.3.2). Deze konflikten kunnen ook ontstaan in een multiprogrammeringssysteem, wanneer bijvoorbeeld meerdere programma's toegang hebben tot een kritische tabel. In dit geval is zo'n tabel een kritische sektie.
De oplossing voor deze konflikten lijkt eenvoudig. Reserveer voor iedere kritische sektie een binaire variabele (lock), om aan te geven dat de kritische sektie vrij dan wel bezet is. Laat ieder programma "lock" testen alvorens de kritische sektie binnen te gaan.
Deze oplossing is gegeven in appendix D . Ze is echter niet acceptabel omdat er situaties mogelijk zijn waarin een programma toegang krijgt tot een kritische sektie, terwijl deze bezet is.

Uitgaande van het gegeven dat de programma's alleen gemeenschappelijke variabelen kunnen testen en veranderen zijn in appendix D nog enkele "oplossingen" gepresenteerd. Deze oplossingen moeten echter eveneens worden verworpen, omdat er gevaar bestaat voor deadlock of after-you-blocking.(In beide gevallen krijgt geen programma de kans om de kritische sektie binnen te lopen terwijl deze vrij is)

Een acceptabele oplossing voor dit konflikt werd voor een uniprocessor multiprogrammeringssysteem geïntroduceerd door Dijkstra in 1968. Zie lit.36 en appendix C . Deze oplossing maakt gebruik van speciale variabelen (semaphores of seinpalen), die door middel van één ondeelbare instruktie getest en eventueel veranderd kunnen worden. Een programma dat via een test op de semaphore toegang krijgt tot een kritische sektie, kan deze semaphore tijdig veranderen ,zodat een volgend programma geen toegang meer kan krijgen tot dezelfde kritische sektie. Een nadeel van deze oplossing is echter dat programma's die geen toegang krijgen tot de kritische sektie, toch voortdurend gebruik blijven maken van het gemeenschappelijk geheugen, waardoor aan niet-wachtende programma's onnodig geheugencycli onthouden worden.

In een multiprocessorsysteem is deze oplossing van Dijkstra moeilijk realiseerbaar, omdat een ondeelbare "test en set" instruktie zal moeten worden gekreëerd en bovendien voorkomen zal moeten worden dat twee processoren gedurende twee opeenvolgende geheugencycli dezelfde "test en set" instruktie opnalen.

In appendix D is een oplossing van het software geheugenkonflikt voor een multiprocessorsysteem gepresenteerd. Het testen en veranderen van gemeenschappelijke variabelen wordt gecoördineerd door een hardware schakeling. Deze schakeling (bijvoorbeeld een round robin schakeling) zorgt ervoor dat er nooit twee of meer processoren tegelijkertijd [1] toegang hebben tot de kritische variabelen (N.B. niet de kritische secties zelf).

Een andere manier om de software geheugenkonflikten op te lossen is de toegang te laten regelen door een aparte processor, bijvoorbeeld de processor die ook de andere organisatorische werkzaamheden voor zijn rekening neemt.(Zie hoofdstuk 3).
In hoofdstuk 4 worden een aantal manieren van gemeenschappelijk geheugengebruik nader aan de orde gesteld.

1.3.5 Koppeling van processoren aan randapparatuur

Het belangrijkste probleem bij de koppeling van input/output (I/O) apparatuur aan één processor is dat van de juiste timing. Een datatransport kan pas plaatsvinden als zowel de processor als het randapparaat hiervoor klaar zijn. Dit probleem kan in principe op drie manieren worden opgelost:

- Onvoorwaardelijk transport. Wanneer de processor er zeker van is dat het randapparaat klaar staat om informatie te ontvangen of uit te zenden, kan hij met één I/O instruktie het datatransport voltooien. Dit is echter meestal niet het geval.
- Voorwaardelijk transport. Voordat het datatransport plaats kan vinden, test de processor de status (vlag) van het apparaat. Pas nadat het randapparaat door het zetten van zijn vlag te kennen heeft gegeven dat hij klaar is, voert de processor de eigenlijke I/O instruktie uit.
- Transport na interruptieaanvraag [2]. De vlaggen van een aantal randapparaten kunnen worden doorverbonden met de interruptlijn van de processor. Wanneer een randapparaat klaar is om informatie uit te wisselen met de processor, geeft hij door het zetten van zijn vlag een interruptie . De processor onderbreekt daardoor het lopende programma en iden-

---

1) Tegelijkertijd heeft hier een ruimere betekenis: gedurende hetzelfde tijdsinterval.

2) Interrupties van randapparaten die DMA willen plegen worden hier buiten beschouwing gelaten.

tificeert in de interruptserviceroutine de herkomst van de interruptie, waarna de juiste I/O instruktie kan worden uitgevoerd.

Het hier geschetste probleem bestaat ook in een uniprocessorsysteem. In een multiprocessorsysteem zullen er bovendien problemen ontstaan wanneer twee of meer processoren toegang hebben tot hetzelfde randapparaat. In analogie met de koppeling tussen processor en geheugen zou men ook hier kunnen spreken van hardware I/O konflikten en software I/O konflikten. Hardware I/O konflikten ontstaan wanneer twee of meer processoren tegelijkertijd hetzelfde randapparaat proberen te bedienen. We kunnen onder een software I/O konflikt verstaan, de situatie die optreedt wanneer een processor een randapparaat wil gebruiken dat bezig is met een uit meerdere woorden bestaand datatransport ten behoeve van een andere processor. Wanneer bijvoorbeeld een bepaalde processor bezig is met het uittypen van een regel tekst op een teletype, mag een andere processor niet ondertussen ook een regel tekst uit laten typen door dezelfde teletype. Het resultaat van deze handeling zou immers een ongedefinieerde zin opleveren. Deze software I/O konflikten zullen bij voorkeur in software worden opgelost (bijvoorbeeld via het operating systeem). Wanneer de software I/O konflikten zijn opgelost, zijn ook automatisch de hardware I/O konflikten uit de weg geruimd.

Bij de koppeling van processoren aan I/O apparatuur is het mogelijk om iedere processor via aparte lijnen te verbinden met ieder randapparaat. Aangezien de tijd die nodig is om een datatransport te plegen tussen twee registers veel kleiner is dan de gemiddelde tijdsduur tussen twee datatransporten, kan voor alle input/output gebruik worden gemaakt van een time-shared bus (Overnemen van data in een register duurt ca. 100nsec. terwijl de minimale tijdsduur tussen twee I/O transporten - per processor - gelijk is aan 5 usec.
Een BUSCONTROLLER (analoog aan de schakelingen van appendix C) zorgt er voor dat telkens slechts één van alle processoren die een aanvrage hebben gepleegd voor I/O, wordt toegelaten tot de bus. Door middel van het adres dat deze processor op de adreslijnen van de bus plaatst, wordt het juiste randapparaat met de datalijnen van de bus verbonden.
Deze BUSCONTROLLER lost dus de extra hardware konflikten op, die gekreëerd worden doordat alle processoren ten aanzien van de I/O gebruik maken van één bus.

Het grote voordeel van de time-shared bus voor de I/O is dat iedere processor slechts één adresregister (8 bit) en twee dataregisters (2 x 8 bit, input- en outputdata zijn gescheiden) nodig heeft om informatie te kunnen uitwisselen met alle randapparaten. Bovendien is het aantal verbindingsdraden tussen de diverse modules van het systeem minimaal.

1.3.6 Koppeling van geheugens aan randapparatuur

Bij de bespreking van de geheugens (zie par. 1.2.3) is naar voren gekomen dat bij gebruik van electronische lees/schrijf geheugens (RAM's), achtergrondgeheugens zoals floppy disks onmisbaar zijn, omdat met het wegvallen van de voedingsspanning ook de informatie in de RAM verdwijnt. Om processortijd te sparen brengt men vaak een koppeling aan tussen floppy disk en werkgeheugen en laat men de floppy disk rechtstreeks lezen of schrijven in het werkgeheugen. Dit noemt men Direct Memory Access (DMA). Een DMA transport wordt geïnitialiseerd door een processor en verder geregeld door een DMA-device. Een apparaat dat op DMA basis communiceert met het geheugen kan op dezelfde manier worden gekoppeld aan het geheugen als een processor. De problemen die ontstaan wanneer een randapparaat rechtstreeks toegang heeft tot het geheugen verschillen dus in wezen weinig van de problemen die ontstaan bij koppeling van processoren aan geheugens. Zie hiervoor par. 1.3.2.

1.3.7 Samenvatting

Resumerend kunnen we de problemen bij de koppeling van de diverse hardware elementen onderscheiden in:

- timing problemen. Een datatransport kan pas plaatsvinden als beide communicatiepartners hiervoor gereed zijn.
- coördinatieproblemen. Wanneer bijvoorbeeld twee processoren tegelijkertijd van hetzelfde geheugen gebruik willen maken, ontstaan konflikten. Toegangskonflikten kunnen verder worden onderverdeeld in:
  - hardware toegangskonflikten. Deze ontstaan wanneer meerdere gebruikers op hetzelfde moment besluiten om van hetzelfde hulpmiddel gebruik te maken.
  - software toegangskonflikten. Hiervan spreken we als een gebruiker toegang wil hebben tot een kritisch geheugengedeelte dat bezet is, of tot een randapparaat dat bezig is met de verwerking van een aaneengesloten datastroom ten behoeve van een andere gebruiker.

HOOFDSTUK 2 SYSTEEMOPZETTEN

2.1 Indeling van multiprocessorsystemen

Bij de indeling van multiprocessorsystemen zijn we uitgegaan van het niveau, waarop parallel wordt gewerkt. Parallel werken kan gebeuren op

- elementair niveau
- funktioneel niveau

Parallel werken op elementair niveau houdt in, dat meerdere processoren gelijktijdig bewerkingen uitvoeren op dezelfde datastroom. We krijgen hierbij de volgende mogelijkheden:

1. Alle processoren voeren gelijktijdig dezelfde bewerking uit op een verzameling van data-elementen, die met elkaar verband houden. Iedere processor neemt hierbij één element uit de datastroom voor zijn rekening (vektor- of array-processing: zie fig.2.1).
2. Iedere processor voert herhaaldelijk dezelfde bewerking uit op verschillende elementen van een enkele datastroom. Het resultaat van de bewerking van de ene processor is ingangsgegeven voor de volgende processor (pipe lining: zie fig.2.2).

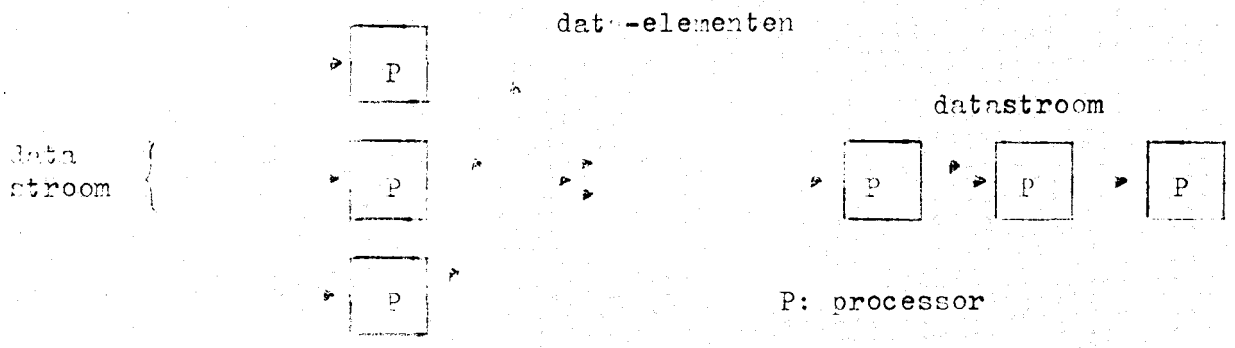


fig.2.1 vektor- of array-processing

fig.2.2 pipe lining

Parallel werken op elementair niveau wordt verder niet in beschouwing genomen.

Parallel werken op funktioneel niveau betekent, dat de programmatuur wordt verdeeld in funktionele delen, die elk een min of meer zelfstandige eenheid vormen en parallel uitgevoerd kunnen worden.

Allereerst kan in het totale programmapakket een scheiding worden gemaakt tussen besturingsprogrammatuur en applicatieprogrammatuur. Terwijl één processor zich bezighoudt met de uitvoering van het besturings

programma, kan parallel daaraan een andere processor bezig zijn met applicatieprogrammatuur.

In plaats van het besturingsprogramma als funktionele eenheid kunnen ook routines hierbinnen worden genomen, die onafhankelijk van elkaar (en dus ook parallel aan elkaar) uitgevoerd kunnen worden. Op dezelfde manier kan binnen de applicatieprogrammatuur als funktionele eenheid de applicatietaak worden gedefinieerd. Een applicatietaak bestaat uit een groep instruktier, die samen een afgeronde hoeveelheid werk vormen. Bij het creëren van funktionele eenheden kan dus zowel het besturingsprogramma als de applicatieprogrammatuur worden verdeeld. Dit is schematisch weergegeven in fig. 2.3.

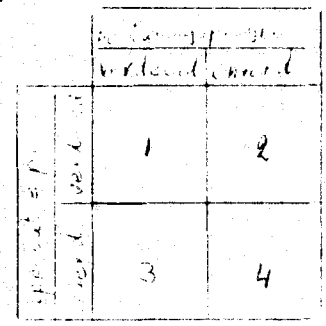


fig. 2.3 Onderverdeling binnen het funktioneel niveau

Het doel van het computersysteem is het uitvoeren van applicatietaken. Het kiezen van de totale applicatieprogrammatuur als funktionele eenheid is niet zinvol, omdat we in een multiprocessorsysteem juist meerdere applicatietaken parallel uit willen voeren. Als eenheid wordt hier dus steeds de taak gekozen.

In het schema van fig. 2.3 blijven nu nog twee mogelijkheden over, nl.

1. het besturingsprogramma verdeeld
2. het besturingsprogramma niet verdeeld

Ad 1. Bij de verdeling van het besturingsprogramma maken we onderscheid tussen

- systeemfunkties
- systeemtaken

Systeemfunkties zijn die werkzaamheden, die direkt te maken hebben met de verdeling van de tijd, die de processoren besteden aan de uitvoering van taken, zowel systeemtaken als applicatietaken. Hieronder vallen de verwerking van vlaggen, en het opbouwen en afbreken van de werklijst, waarin alle taakaanvragen staan.

Systeemtaken zijn taken, die **een bepaalde systeemfaciliteit uitvoeren ten behoeve van één of meerdere gebruikers.**

De elementen van het besturingssysteem, die hier zijn gepresenteerd, komen uitgebreid aan de orde in hoofdstuk 3.

Uitgaande van de systeemfunkties definiëren we nu binnen de systemen, waarin het besturingsprogramma is verdeeld, een

- centraal systeem
- decentraal systeem

In een centraal systeem zijn de systeemfunkties niet verdeeld. Ze worden uitgevoerd door één processor. Dit hoeft niet steeds dezelfde te zijn. De systeemtaken worden in eerste instantie op dezelfde manier behandeld als de applicatietaken.

In een decentraal systeem zijn de systeemfunkties verdeeld. Verschillende processoren kunnen er onafhankelijk van elkaar mee bezig zijn.

Ad 1. Als het besturingsprogramma helemaal niet verdeeld is, wordt het steeds doorlopen door slechts één processor. Dit hoeft niet steeds dezelfde te zijn. Omdat hier ook de systeemfunkties niet verdeeld zijn, spreken we in dit geval van een onverdeeld centraal systeem.

Samengevat komen we, uitgaande van het besturingsprogramma, tot de volgende indeling (zie schema fig. 2.4):

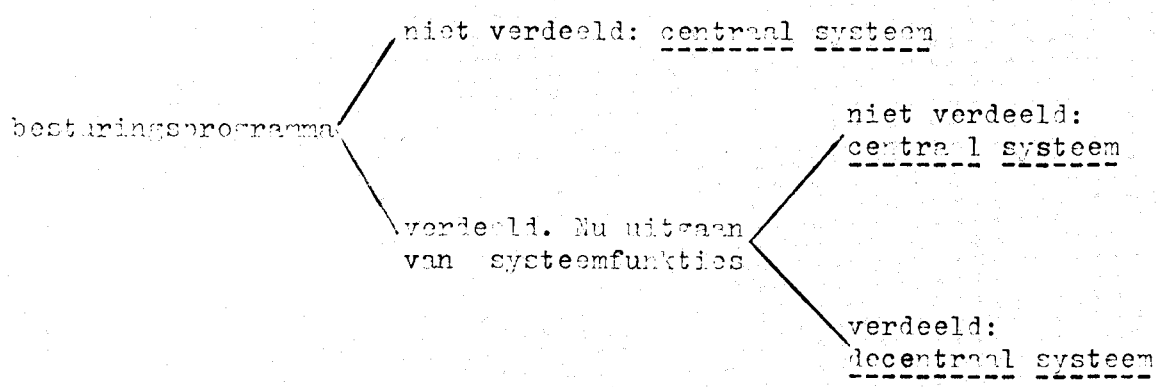


fig. 2.4 Indeling op funktioneel niveau, uitgaande van het besturingsprogramma

De centrale systemen, zowel onverdeeld centraal als verdeeld centraal, worden behandeld in par. 2.4. In par. 2.3 wordt nog iets vermeld over de decentrale systemen.

2.2 Centrale systemen

2.2.1 Inleiding

Zoals in de vorige paragraaf is beschreven, kunnen de centrale systemen op twee verschillende manieren worden gevormd:

- het besturingssysteem is niet verdeeld. Zowel de systeemfunkties als de systeemtaken horen bij elkaar en worden beschouwd als één eenheid (onverdeeld centraal systeem).
- het besturingssysteem is wel verdeeld. De systeemfunkties zijn niet verdeeld. Ze worden uitgevoerd door één processor. Dit hoeft niet steeds dezelfde te zijn. De systeemtaken worden op dezelfde manier behandeld als de applicatietaken (verdeeld centraal systeem).

In het nu volgende gedeelte wordt uitgegaan van het verdeeld centraal systeem om zodoende de mogelijkheid te creëren dat taken zoveel mogelijk parallel uitgevoerd kunnen worden. Het zal blijken (par.3.2.4), dat dit in sommige systeemopzetten leidt tot moeilijkheden in de koppeling van processoren met randapparaten. Door in die gevallen de systeemtaken te laten uitvoeren door dezelfde processor, die ook de systeemfunkties uitvoert (onverdeeld centraal), wordt de systeemopzet aanzienlijk verbeterd.

In het verdeeld centraal systeem is nog niet vermeld, welke werkzaamheden door welke processoren worden uitgevoerd. Dit wordt nu gebruikt om onderverdelingen aan te brengen binnen dit systeem.
De programma's, die toegewezen moeten worden, zijn de systeemfunkties en de taken. Hierbij wordt geen onderscheid gemaakt tussen systeemtaken en applicatietaken.

De systeemfunkties zijn steeds gekoppeld aan slechts één processor. Als het steeds dezelfde processor is, zijn de systeemfunkties statisch toegewezen. Indien het steeds een andere processor kan zijn, is er sprake van een dynamische processorallocatie. In beide gevallen kan voor de taakafhandeling nog gekozen worden tussen een dynamische of een statische toewijzing van taken. De toewijzing is statisch, als een taak steeds aan dezelfde processor wordt toegewezen, en dynamisch, als een taak steeds aan een andere processor kan worden toegewezen. We komen zo tot de volgende indeling binnen het verdeeld centraal systeem (zie schema fig.):

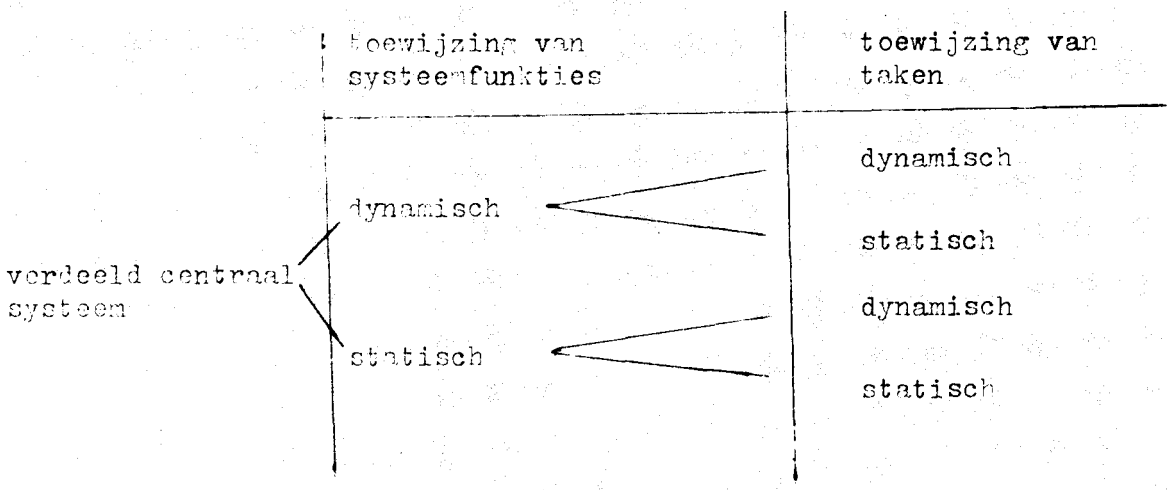


fig. 2.5 Schematische weergave van de onderverdeling binnen het verdeeld centraal systeem.

Bij de dynamische toewijzing van programma's aan processoren kunnen programma's worden toegewezen aan die processoren, die tot op dat moment vrij waren of minder belangrijk werk deden. Het systeem is daardoor erg flexibel. De mogelijkheid is gecreëerd om de verdeling van programma's over de processoren te optimaliseren.

De konsekwentie van de dynamische processorallocatie is, dat de programma's moeten staan in een gemeenschappelijk geheugen, dat bereikbaar is voor alle processoren. Dit kan ertoe leiden, dat bij de toegang tot dat gemeenschappelijk geheugen wachttijden kunnen ontstaan, doordat processoren op elkaar moeten wachten (zie **hoofdstuk 4**).

Bij een statische toewijzing worden de programma's steeds door dezelfde processoren uitgevoerd. Een aangevraagd programma moet nu op uitvoering wachten, tot de processor, die verantwoordelijk is voor dat programma, vrij is om het programma uit te voeren. Sommige processoren kunnen nu overbelast zijn, terwijl andere nauwelijks iets te doen hebben. Omdat de programma's nu steeds door dezelfde processoren worden uitgevoerd, mogen ze in privégeheugens staan. Bij de toegang tot een privégeheugen kunnen geen wachttijden ontstaan, omdat een processor als enige toegang heeft tot dat geheugen.

Het systeem, waarbij zowel de toewijzing van de systeemfunkties als ook de toewijzing van de taken dynamisch is, is het meest **universele**. De overige centrale systemen kunnen hieruit worden afgeleid door systeemfunkties en/of uitvoering van taken statisch toe te wijzen aan processoren.

Achtereenvolgens worden nu in het kort de centrale systemen besproken.

2.2.2 Centraal systeem met systeemfunkties en taken dynamisch

In dit systeem kunnen alle processoren zowel de systeemfunkties als ook alle taken uitvoeren. Het wordt in de literatuur* genoemd het symmetrisch of anoniem systeem. Symmetrisch vanwege de gelijkwaardige behandeling van de processoren, en anoniem vanwege de anonimiteit van de processoren bij het uitvoeren van programma's.

Alle programma's kunnen worden uitgevoerd door alle processoren. Dit betekent, dat elk programma bereikbaar moet zijn voor elke processor, hetgeen als volgt gerealiseerd kan worden:
Elk programma kan in elk geheugenblok worden gezet (dynamische geheugenallocatie van programma's) en elk geheugenblok is bereikbaar voor elke processor (zie fig.2.6).

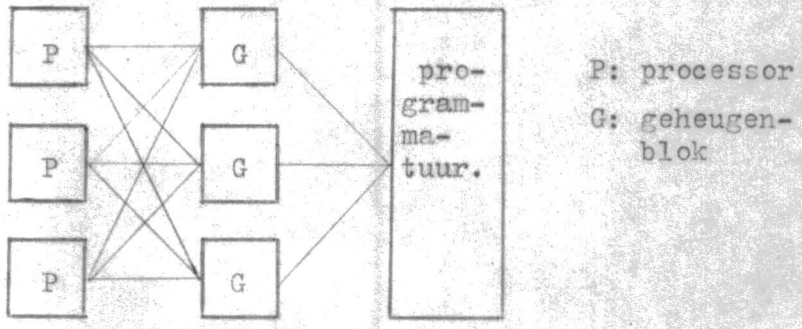


fig.2.6 Elk programma kan in elk geheugenblok worden gezet en elk geheugenblok is bereikbaar voor elke processor

Dit is het meest algemene geval, waaruit twee andere manieren kunnen worden afgeleid:

- elk programma staat steeds op een vaste plaats in één van de geheugenblokken(statische geheugenallocatie van de programma's). De processoren hebben direkte toegang tot alle geheugenblokken (zie fig.2.7).
- Elk programma kan in elk geheugenblok worden gezet (dynamische geheugenallocatie van programma's). De geheugenblokken hoeven slechts met één processor verbonden te zijn (zie fig.2.8). We hebben nu te maken met een overzetsysteem, waarbij elke processor een privégeheugen heeft (zie par.4.5).

---

* Zie lit. 1

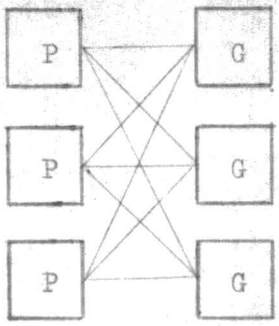


fig.2.7 Elk programma staat op een vaste plaats in één van de geheugenblokken. Elk geheugenblok is direkt bereikbaar voor elke processor.

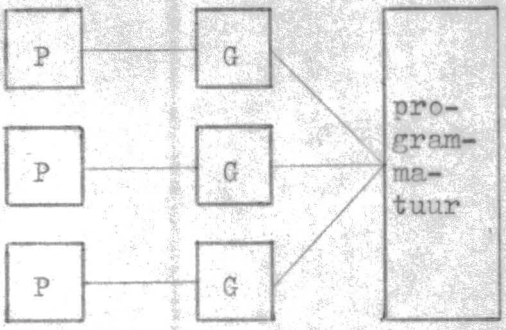


fig.2.8 Elk programma kan in elk geheugenblok worden gezet. De geheugenblokken zijn verbonden met slechts één processor.

Het gebruik van de privégeheugens is interessant, als de processoren niet onderbroken worden bij de uitvoering van taken. Een eenmaal begonnen taak wordt na transport naar het privégeheugen zonder onderbreking afgewerkt. De aangevraagde taken worden na plaatsing in de werklijst op volgorde van prioriteit in behandeling genomen. In deze opzet, waarbij een eenmaal begonnen taak niet meer wordt onderbroken, wordt de prioriteit binnen de taakafhandeling slechts in zoverre gehandhaafd, dat bij aanvang van een nieuwe taak steeds wordt begonnen met de taak met de hoogste prioriteit. Dit heeft tot gevolg, dat een aanvraag voor een belangrijke taak pas wordt gehonoreerd als een processor een taak heeft beëindigd en een nieuwe taak vraagt. In een gunstig geval wordt de aanvraag meteen gehonoreerd (als een processor al vrij was), in andere gevallen wordt de aanvraag pas later gezien. Er is een transportorganisatie nodig om het transport van de programma's naar de privégeheugens op de juiste tijd en de juiste manier te regelen.

Als de uitvoering van taken onderbroken kan worden, is het gebruik van privégeheugens niet zinvol. Buiten een processorstatus, die gered moet worden, is er nu ook een grote programmastatus, nl. alle resultaten in het privégeheugen. Om het mogelijk te maken, dat de taak later weer hervat kan worden door een willekeurige (andere) processor, moet de taak eerst weer naar het centraal geheugen getransporteerd worden, waarna hij, voordat hij hervat kan worden, weer in een privégeheugen moet worden gezet.

Deze omslachtige behandeling kanvermeden worden door, indien onderbreking van taken wordt toegestaan, de processoren rechtstreeks te laten werken met het centraal geheugen. De processoren zijn nu verbonden met alle geheugenblokken.

Als de uitvoering van een taak nu onderbroken wordt, wordt de processorstatus gered in het centraal geheugen. Elke taak heeft voor dit doel een vaste redruimte. Bij voortzetting van de taak wordt de status hersteld en meteen begonnen met de verdere uitvoering.

Er kan nu gezorgd worden voor een optimale prioriteitsresponsie. Dit betekent dat van de n taken in de werklijst het systeem met m processoren steeds de m meest urgente taken behandelt. Als elke taak een vaste prioriteit heeft, worden de taken zodanig uitgevoerd, dat steeds de som van de prioriteiten maximaal is. Deze som heet de systeemprioriteit. Om de systeemprioriteit maximaal te houden moet bij elke nieuwe taakaanvraag worden onderzocht, of de systeemprioriteit vergroot kan worden door een processor met een nieuwe taak te laten beginnen.

Randapparatuur is gekoppeld aan alle processoren. Dit is nodig omdat alleen op die manier kan worden gewaarborgd, dat alle I/O-taken kunnen worden uitgevoerd door alle processoren (zie fig. 2.9).

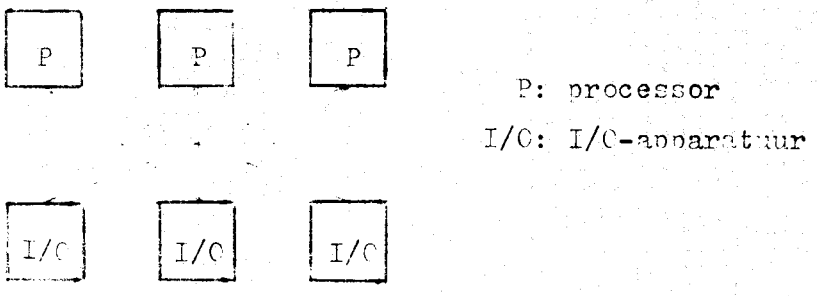


fig. 2.9. Koppeling van processoren met I/O-apparaten in het symmetrisch of anoniem systeem.

Voordelen van het anoniem systeem:

- Het systeem is flexibel. Elke processor kan elk programma uitvoeren. Elke processor heeft dus ook de mogelijkheid om systeemfunkties uit te voeren en als zodanig op te treden als 'meester'. Een 'meester' hoeft slechts gecreëerd te worden als dat nodig is. De rest van de tijd kunnen de processoren besteden aan de uitvoering van taken.
- De mogelijkheid is aanwezig om steeds de belasting van de processoren

afhankelijk van de aangevraagde taken te sturen.
- Uitvoering van belangrijke taken wordt niet onderbroken door aanvragen voor minder belangrijke taken. Als een processor onderbroken kan worden om organisatorische werkzaamheden op te knappen, is dat steeds diegene, die bezig was met de taak met laagste prioriteit.
- Het systeem kan gemakkelijk en snel worden aangepast aan het uitvallen van een processor.

Nadelen:
- Alle I/O-apparatuur moet gekoppeld zijn aan alle processoren.
- Als meerdere processoren toegang hebben tot hetzelfde werkgeheugen, wordt de doorzet nadelig beïnvloed, omdat bij de geheugenreferenties wachttijden kunnen ontstaan (zie hoofdstuk 4).
  Bij gebruik van privégeheugens wordt de doorzet nadelig beïnvloed door de transporten van de programma's naar de privégeheugens. Na het transport kan een processor echter op volle snelheid werken met zijn privégeheugen, omdat hij bij de toegang op niemand hoeft te wachten.

Het anoniem systeem wordt nader uitgewerkt in hoofdstuk 5.

2.2. Centraal systeem met systeemfunkties dynamisch en taken statisch

Elke processor voert een vaste taak of een vaste groep taken uit. Bovendien kan elke processor de systeemfunkties uitvoeren, die nodig zijn voor een korrekte afhandeling van de taken. Er is steeds maar één processor bezig met de uitvoering van systeemfunkties. Dit mag echter steeds een andere zijn.

De taken staan in privégeheugens van de processoren. Uitvoering van taken kan op volle snelheid gebeuren, omdat bij de geheugenreferenties geen wachttijden kunnen ontstaan. Elke processor werkt immers met zijn privégeheugen.

Er is een gemeenschappelijk geheugen nodig, waarin de systeemfunkties van het besturingsprogramma staan. Deze moeten bereikbaar zijn voor elke processor, omdat elke processor de mogelijkheid moet hebben om op te treden als 'meester'.

Vlaggen, die aanleiding kunnen geven tot organisatorische werkzaamheden, zoals interrupties van randapparaten, moeten verbonden zijn met alle processoren. Ze worden verwerkt door die processor, die bezig was met

de minst belangrijke taak binnen het systeem. Dit hoeft niet steeds dezelfde processor te zijn.

Systeemtaken worden, evenals applicatietaken, steeds gedaan door dezelfde processor. Hierbij hoeven niet alle systeemtaken te worden gedaan door één en dezelfde processor. Het gevolg hiervan is, dat de dataregisters van de randapparaten verbonden zijn met één processor, en niet met alle zoals in het anoniem systeem.

Vlaggen van de randapparaten worden nu verwerkt door processoren, die zelf niet in staat zijn om de dataregisters van die apparaten te bereiken. Vlaggen worden dus niet altijd verwerkt door de processor, waaraan de dataregisters zijn gekoppeld.

Het grote nadeel van dit systeem is, dat de flexibiliteit in de organisatie (elke processor kan de systeemfunkties uitvoeren) is gekoppeld aan de starheid in de uitvoering van taken (elke processor heeft zijn eigen taken). Daardoor lijkt het systeem ongeschikt om nader te worden uitgewerkt.

2.2.4 Centraal systeem met systeemfunkties statisch en taken dynamisch

In dit systeem worden de systeemfunkties steeds uitgevoerd door één en dezelfde processor. De vlaggen van de randapparaten zijn gekoppeld aan deze processor. Hij verwerkt de vlaggen en beslist, welke akties genomen moeten worden. Hij stelt de werklijst op, waarin de systeemtaken en de applicatietaken komen te staan, die door de andere processoren moeten worden uitgevoerd.

Dataregisters van de randapparaten zijn gekoppeld aan alle processoren, die taken uit moeten voeren.

Aan de hand van een sterk vereenvoudigd voorbeeld wordt nu de koppeling van randapparaten aan het systeem toegelicht.

Van een typemachine, die bestaat uit twee onafhankelijke delen, nl. toetsenbord en printer, zijn de dataregisters verbonden met de taakprocessoren. De vlaggen, zijn aangesloten op de interrupt-bus van de processor, die de systeemfunkties uitvoert (zie fig.2.10).

Wanneer de operateur op het toetsenbord een karakter aanslaat, komt dit te staan in een inputregister van de taakprocessoren. Tegelijkertijd komt er een interruptie bij de organisatieprocessor. Deze geeft aan één van de taakprocessoren door, dat op het toetsenbord een karakter is ingetikt.

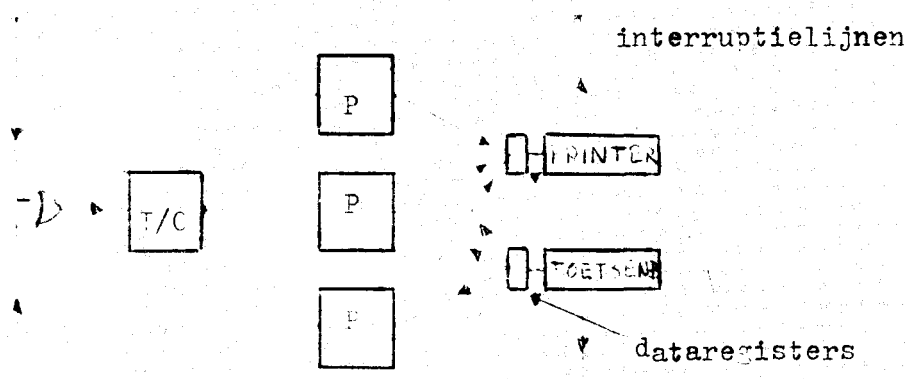


fig. 2.10 Koppeling van typemachine aan het systeem

De taakprocessor reageert hierop door het karakter met behulp van een inputinstruktie naar binnen te halen. Met behulp van een outputinstruktie wordt nu gezorgd voor een reflectie van het karakter. De vlag, die door de printer wordt gezet, nadat het karakter is afgedrukt, moet weer worden verwerkt door de organisatieprocessor.

De moeilijke koppeling van de randapparaten aan de processoren, die er bovendien de oorzaak van is, dat er vaak inefficiënt wordt gewerkt, kan worden vermeden door het creëren van een aparte processor, die alle I/O verzorgt. Alle I/O-apparatuur is gekoppeld aan deze processor. Alle vlaggen van de buitenwereld komen bij hem binnen. Hij stelt de werklijst op voor de andere processoren. We hebben op deze manier een systeem gemaakt, waarbij alle systeemfunkties en systeemtaken worden uitgevoerd door één processor, terwijl de applicatietaken dynamisch worden toegewezen aan de taakprocessoren. Dit systeem, waarbij het besturingssysteem niet is verdeeld en statisch is gekoppeld aan één processor, wordt beschreven in de volgende paragraaf (2.2.5).

2.2.5 Onverdeeld centraal systeem met besturingssysteem statisch en applicatietaken dynamisch

In dit systeem worden alle systeemfunkties en systeemtaken steeds gedaan door één en dezelfde processor. Alle randapparaten zijn gekoppeld aan deze processor. Hij verwerkt de vlaggen en beslist, welke akties genomen moeten worden. Hij stelt de werklijst op voor de taakprocessoren.

De nu gecreëerde aparte organisatieprocessor mag verbonden zijn met een privégeheugen, waarin het besturingsprogramma staat. Hij is immers vanwege de statische toewijzing van het besturingsprogramma de enige, die dit programma mag doorlopen.
De organisatieprocessor enerzijds en de taakprocessoren anderzijds hoeven ten aanzien van de programmatuur dus geen gemeenschappelijk geheugen te hebben. Ze hebben alleen een gemeenschappelijk geheugen nodig voor de communicatie. Als bijvoorbeeld één van de taakprocessoren een bericht heeft voor de buitenwereld, plaatst hij dit in het communicatiegeheugen en doet bij de organisatieprocessor een aanvraag. Deze moet nu voor een verdere behandeling van het bericht zorgen, omdat hij de enige is, die de randapparaten kan bedienen.
De taakprocessoren voeren uitsluitend applicatietaken uit. Deze taken worden dynamisch toegewezen aan de processoren. Dit betekent, dat elke taak bereikbaar moet zijn voor elke processor, hetgeen evenals bij het anoniem systeem als volgt gerealiseerd kan worden:
Elke taak kan in elk geheugenblok worden gezet (behalve in het privégeheugen van de organisatieprocessor) en elk geheugenblok is bereikbaar voor elke processor. Het gemeenschappelijk 'taakgeheugen' wordt hierbij tevens gebruikt als communicatiegeheugen. Alle randapparaten zijn verbonden met de organisatieprocessor (zie fig.2.11).

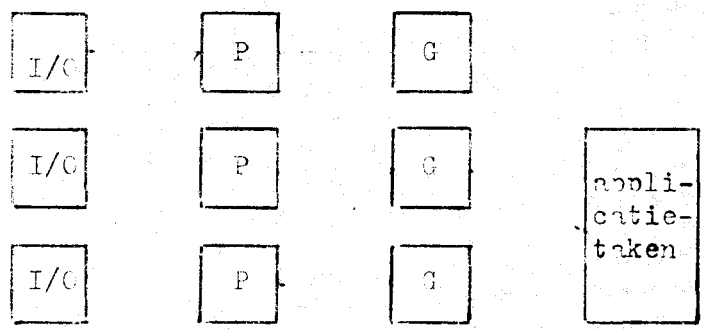


fig.2.11 Onverdeeld centraal systeem met aparte organisatieprocessor
De organisatieprocessor heeft een privégeheugen. Voor de rest kan elke applicatietaak in elk geheugenblok worden geplaatst. Elk van deze geheugenblokken is bereikbaar voor alle taakprocessoren en voor de organisatieprocessor.

Uit dit meest algemene geval kunnen weer twee andere manieren worden afgeleid:

- elke taak staat steeds op een vaste plaats in één van de geheugenblokken (statische geheugenallocatie van de applicatietaken). De taakprocessoren hebben directe toegang tot deze geheugenblokken. Het gemeenschappelijk 'taakgeheugen' wordt weer gebruikt als communicatiegeheugen (zie fig.2.1

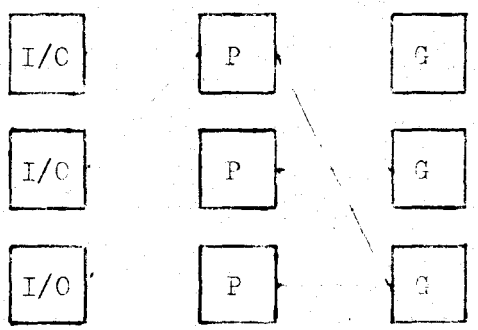


fig.2.12 Elke taak staat op een vaste plaats in één van de geheugenblokken. Elk van deze geheugenblokken is bereikbaar voor alle taakprocessoren en voor de organisatieprocessor.

- elke taakprocessor heeft een privégeheugen, waarin elke uit te voeren applicatietaak kan worden gezet (dynamische geheugenallocatie van applicatietaken). Bovendien is er nu nog een communicatiegeheugen nodig, dat bereikbaar moet zijn voor alle processoren (zie fig.2.13)

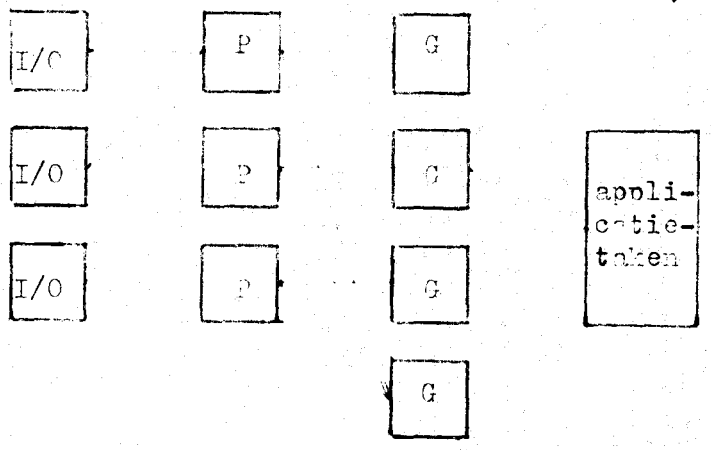


fig.2.13 Elke applicatietaak kan in het privégeheugen van elke taakprocessor worden gezet. Alle in het systeem aanwezige processoren hebben toegang tot een gemeenschappelijk communicatiegeheugen.

Het gebruik van privégeheugens voor taakprocessoren is interessant, als de processoren niet onderbroken worden bij de uitvoering van applicatietaken. Een eenmaal begonnen taak wordt na transport naar het privégeheugen zonder onderbreking uitgevoerd. De transportorganisatie, die nodig is om de taken op de juiste tijd en de juiste manier over te zetten naar de privégeheugens, wordt verzorgd door de organisatieprocessor. Data, die naar de buitenwereld moeten worden gevoerd, worden door de taakprocessoren in het communicatiegeheugen gezet. Dit kan op directe manier gebeuren, als de taakprocessoren rechtstreeks verbonden zijn met het communicatiegeheugen, of op indirekte manier, als de data langs een andere weg (DMA) in het communicatiegeheugen moeten worden gezet.

Als de uitvoering van taken onderbroken kan worden, is het gebruik van privégeheugens niet zinvol vanwege de gecreëerde programmastatus. Om het nu mogelijk te maken, dat de taak later weer door een willekeurige (andere) processor hervat kan worden, zijn immers weer twee transporten nodig. De processoren kunnen nu beter rechtstreeks verbonden zijn met het centraal geheugen.

Er kan weer gezorgd worden voor een optimale prioriteitsresponsie. Van de n applicatietaken in de werklijst voert het systeem met m taakprocessoren steeds de m meest urgente taken uit. Bij elke nieuwe taakaanvraag onderzoekt de organisatieprocessor, of deze nieuwe taak direkt in behandeling moet worden genomen.

Voordelen van het systeem met een aparte organisatieprocessor:

- alle I/O-apparaten zijn gekoppeld aan één processor. Alle signalen van en naar de buitenwereld worden door deze processor verwerkt.
- het systeem biedt de mogelijkheid om de verdeling van de applicatietaken over de verschillende taakprocessoren dynamisch te optimaliseren. Als een taakprocessor onderbroken wordt om een belangrijker applicatietaak in behandeling te nemen, is dat steeds de taakprocessor, die bezig was met de minst belangrijke taak. De andere processoren kunnen blijven doorgaan met de uitvoering van taken.
- het besturingsprogramma staat in het privégeheugen van de organisatieprocessor. Bij de toegang tot dit geheugen kunnen dientengevolge geen wachttijden ontstaan.

Nadelen:

- bij een groot aantal taakprocessoren kan de organisatieprocessor de bottle neck vormen van het systeem. Dit kan op zijn beurt resulteren

in een inefficiënt gebruik van de taakprocessoren.

- Als meerdere taakprocessoren toegang hebben tot hetzelfde werkgeheugen, wordt de doorzet nadelig beïnvloed, omdat bij de geheugenreferenties wachttijden kunnen ontstaan (hoofdstuk 4).
  Bij gebruik van privégeheugens van de taakprocessoren wordt de doorzet nadelig beïnvloed door de transporten van de applicatietaken naar de privégeheugens. Na een transport kan de taakprocessor echter op volle snelheid werken, omdat bij de geheugenreferenties geen wachttijden kunnen ontstaan.

Als het gemeenschappelijk geheugen de bottle neck vormt van het systeem, moet gebruik ervan zoveel mogelijk worden beperkt. Alle programma's worden in privégeheugens gezet en het gemeenschappelijk geheugen wordt alleen gebruikt voor de communicatie.

2.2.6 Onverdeeld systeem met besturingssysteem en applicatietaken statisch

Het systeem is onverdeeld om de moeilijke koppeling tussen randapparaten en processoren te vermijden, die ontstaat, wanneer systeemfunkties door andere processoren worden uitgevoerd dan systeemtaken(zie par.2.2.4).
Het besturingsprogramma wordt steeds uitgevoerd door één en dezelfde processor. De applicatietaken zijn statisch toegewezen, dus elke processor heeft een vaste taak of een vaste groep taken.
De applicatietaken staan in privégeheugens van de processoren. Hiernaast is er nog een gemeenschappelijk geheugen nodig, dat gebruikt wordt als communicatiegeheugen. Hierin worden bijvoorbeeld data gezet, die door een taak worden geproduceerd en waarmee een andere taak (en ook een andere processor) verder moet werken. We krijgen nu de volgende configuratie (zie fig.2.14)

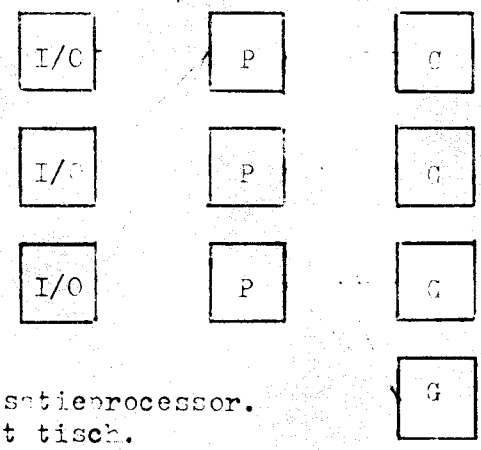


fig. 2.14 Systeem met aparte organisatieprocessor.
Toewijzing van taken is statisch.

De organisatieprocessor stelt de werklijst op van alle taakprocessoren. Voor elke taakprocessor is er een aparte werklijst. Dit kan in werkelijkheid één grote lijst zijn met bij elke taak het nummer van de processor, die de applicatietaak moet uitvoeren.

Alle verkeer met de buitenwereld geschiedt via de organisatieprocessor. Deze haalt bijvoorbeeld data op uit het communicatiegeheugen en zorgt voor een korrekte verwerking ervan.

Als in dit systeem de taakprocessoren niet onderbroken mogen worden tijdens de uitvoering van taken, wordt de prioriteit bij de uitvoering van taken doorbroken, behalve in het geval, dat elke processor slechts één taak heeft. Bij meerdere taken per processor kan de verdeling van de taken over de processoren zodanig worden gekozen, dat de prioriteit bij uitvoering voldoende blijft gewaarborgd (bijvoorbeeld korte taken bij elkaar). Het blijft hier echter steeds mogelijk, dat sommige processoren overbelast zijn, terwijl andere nauwelijks iets te doen hebben.

Voordelen:

- De doorzet kan groot zijn, omdat bij de geheugenreferenties geen wachttijden ontstaan. Iedere processor heeft immers een privégeheugen. Een uitzondering hierop vormt het communicatiegeheugen, dat door alle processoren in het systeem gerefereerd kan worden.
- Als taakprocessoren onderbroken kunnen worden bij de uitvoering van taken om te beginnen met een taak met hogere prioriteit, kan de responsietijd goed zijn. Bovendien is de responsietijd niet afhankelijk van de belasting bij andere taakprocessoren.

Nadelen:

- Piekbelastingen, die ontstaan bij een taakprocessor, kunnen niet door andere taakprocessoren worden opgevangen. Dit is nadelig voor de doorzet.
- Subroutines, die vanuit meerdere taken (en meerdere processoren) aangeroepen kunnen worden, moeten aanwezig zijn in meerdere privégeheugens.
- Er is een gemeenschappelijk geheugen nodig voor communicatie tussen de modulen onderling, enerzijds voor communicatie tussen de organisatieprocessor en de taakprocessoren en anderzijds voor communicatie tussen de taakprocessoren onderling. Bij de toegang tot dat gemeenschappelijk geheugen kunnen wachttijden ontstaan, doordat processoren op elkaar moeten wachten.

2.3 Decentrale systemen

In een decentraal systeem zijn de systeemfunkties verdeeld. Meerdere processoren kunnen tegelijkertijd onafhankelijk van elkaar bezig zijn met de uitvoering van systeemfunkties. De toewijzing ervan kan nog dynamisch of statisch zijn. Ook mengvormen zijn nu mogelijk: sommige funkties worden dynamisch toegewezen en andere statisch. Het aantal funkties kan nog worden bepaald, maar dit is sterk afhankelijk van de struktuur van het besturingssysteem.(zie hoofdstuk 3).

In het algemeen is in een decentraal systeem de toewijzing van de funkties statisch, omdat bij een dynamische toewijzing steeds moet worden bijgehouden, welke processor met welk gedeelte van het besturingssysteem bezig is. Dit leidt weer tot een zekere mate van centralisatie. Bij enkele funkties is het echter niet belangrijk, door welke processor ze worden uitgevoerd; als er maar steeds voor gezorgd wordt, dat er niet meerdere processoren gelijktijdig mee bezig zijn.(zie par.3.3).

Bij statische toewijzing van de systeemfunkties voert een processor steeds dezelfde funkties uit. De systeemtaken worden nu meestal ook statisch toegewezen, afhankelijk van de verdeling van de funkties over de verschillende processoren.
Een mogelijke manier om te decentraliseren is het decentraal aansluiten van randapparaten.(zie fig.2.15)

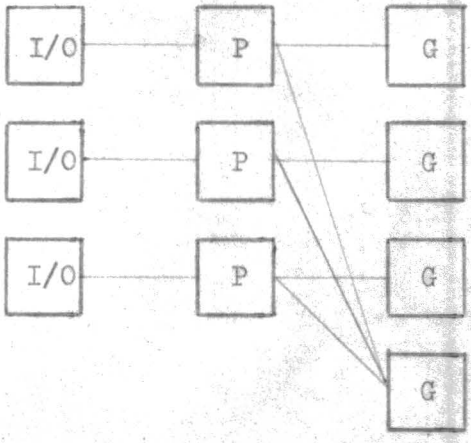


fig.2.15 Mogelijke opzet voor een decentraalsysteem.

De vlaggen van de randapparatuur, die aanleiding kunnen geven tot organisatorische werkzaamheden komen decentraal binnen. De programma's voor de juiste verwerking ervan staan in de privégeheugens van de processoren en kunnen parallel aan elkaar worden uitgevoerd. Er is een gemeenschappelijk geheugen nodig voor communicatie tussen de modulen onderling.

Decentrale systemen kunnen worden verkregen door het opdelen van het besturingssysteem in een aantal elementaire funkties. Deze funkties worden uitgevoerd door 'plankjes', bestaande uit een processor met zijn privégeheugen.

2.4 Konklusies

Bij de centrale systemen is het symmetrisch of anoniem systeem het meest flexibel. In dit systeem is elk programma bereikbaar voor elke processor. Alle randapparaten zijn gekoppeld aan alle processoren. Dit systeem wordt verder beschreven in hoofdstuk 5.
Alle overige centrale systemen kunnen hieruit worden afgeleid door programma's en randapparaten statisch toe te wijzen aan processoren.

Koppeling van alle randapparaten aan één processor heft een belangrijk nadeel van het anoniem systeem op, nl. de omvangrijke coördinatie tussen randapparaten en processoren. De flexibiliteit blijft gedeeltelijk nog behouden, doordat applicatietaken nog uitgevoerd kunnen worden door alle taakprocessoren. De hier gecreëerde aparte organisatieprocessor verzorgt alle verkeer met de buitenwereld en maakt de werklijst klaar voor de taakprocessoren.

Een andere manier, om de moeilijke verbinding in het anoniem systeem tussen randapparaten en processoren te vermijden is de koppeling van verschillende randapparaten aan verschillende processoren, doch ieder randapparaat is verbonden met slechts één processor. We hebben nu een decentraal systeem. In het uiterste geval heeft elk randapparaat een eigen processor, die de volledige besturing ervan verzorgt. Vlaggen komen decentraal binnen; de hele besturing is gedecentraliseerd.

Bij de in de vorige paragrafen besproken systeemopzetten is steeds de noodzaak van een gemeenschappelijk geheugen naar voren gekomen. De funkti van dat geheugen kan zijn:

werkgeheugen (anoniem systeem) of communicatiegeheugen (systeem met besturingsprogramma en applicatietaken statisch). Over de realisatie van een gemeenschappelijk geheugen is niets vermeld. De mogelijke manieren, waarop gemeenschappelijk gebruik van geheugens gerealiseerd kunnen worden, vormen het onderwerp van hoofdstuk 4.

Tot slot van dit hoofdstuk wordt hier nog een overzicht gegeven van systemen, waarin meerdere processoren werkzaam zijn. Alle in dit hoofdstuk genoemde systemen zijn hierin opgenomen. Voor de volledigheid zijn ook de systemen opgenomen, die alleen in de inleiding zijn vermeld (zie fig. 2.1).

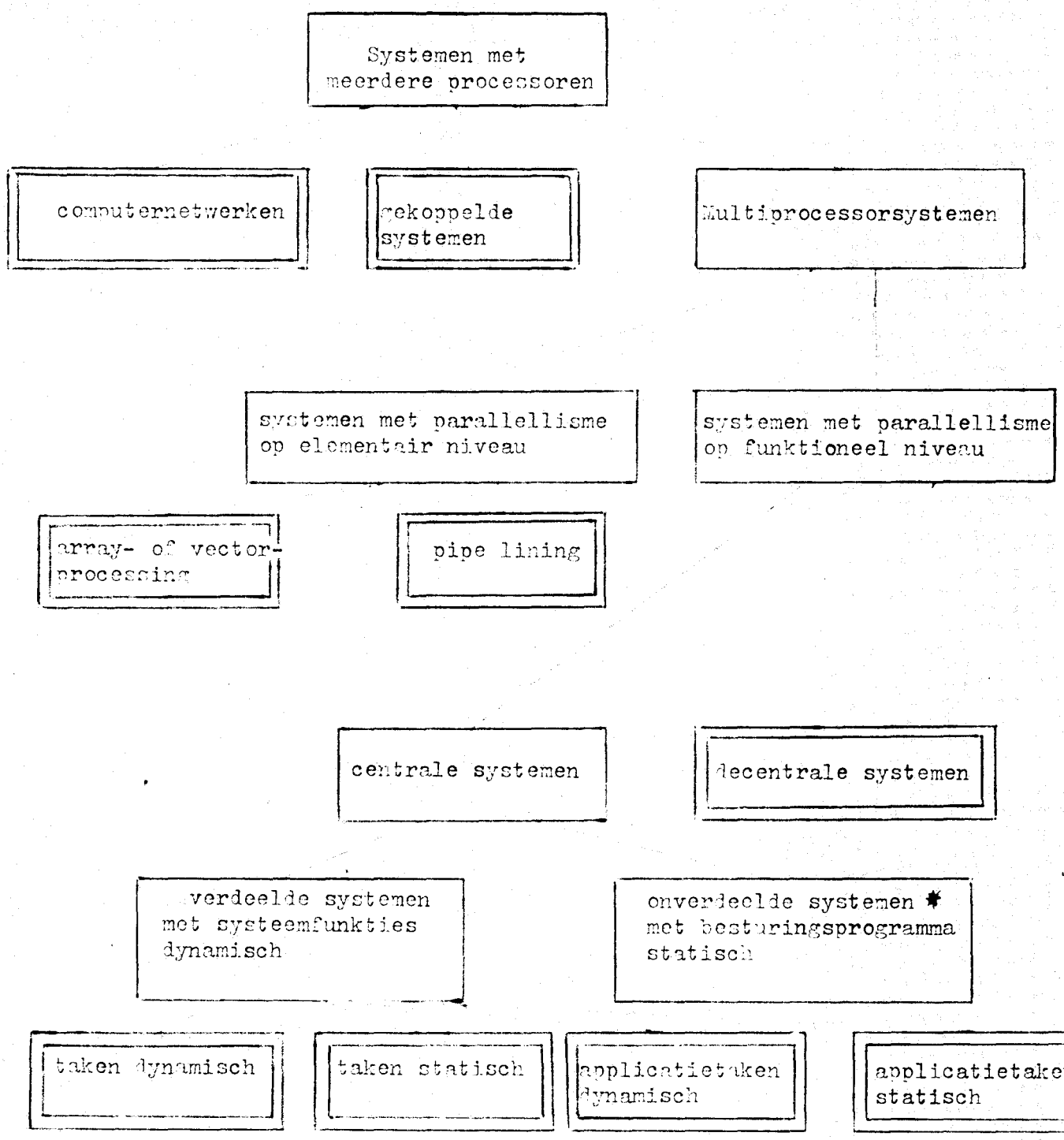


fig. 2.16 Overzicht van systemen, waarin meerdere processoren werkzaam zijn.

* De verdeelde systemen met systeemfunkties statisch bleken geen zinvolle systeemopzetten op te leveren. (zie par. 2.2.4).

HOOFDSTUK 3 BESTURINGSSYSTEMEN IN EEN MULTIPROCESSOR-OMGEVING

3.1 Inleiding

Zoals uit hoofdstuk 2 (systeemopzetten) is gebleken wordt er bij de indeling van (typen) multiprocessorsystemen een onderscheid gemaakt tussen

- het pakket applicatieprogrammatuur.
- het pakket systeemprogrammatuur (= besturingssysteem).

Applicatieprogrammatuur is dat deel van de programmatuur dat de gebruiker als taak uitgevoerd wil hebben. De algemene funktie van systeemprogrammatuur is: de verdeling van de resources (gemeenschappelijke systeemfaciliteiten) welke nodig zijn voor de afhandeling van onderling concurerende programmas. Deze recources zijn:

- processoren
- werkgeheugen
- hulpgeheugen
- in/uitvoerapparaten
- sommige kritische tabellen of data sets.

Gezien het tijdgebonden karakter van procescomputer control toepassingen wordt aan het besturingssysteem een strenge voorwaarde gesteld:

- het moet real-time zijn; de resultaten van de taakprogrammas moeten op tijd aan het te regelen proces worden toegevoerd.
  Dit betekent dat iedere aangevraagde taak zo snel moet worden uitgevoerd dat het resultaat ervan nog bruikbaar is.
  Het spreekt voor zich dat voor sommige taken de real-time eis zwaarder is dan voor andere, anders gezegd: sommige taken zijn "urgenter" dan andere. Iedere aangevraagde taak moet daarom op zijn mate van urgentie worden beoordeeld. Dit wordt verwezenlijkt door aan de programmas een prioriteit toe te kennen en de aangevraagde programmas op volgorde van prioriteit uit te voeren.

In een uniprocessorsysteem moeten applicatie- en systeem-programmas noodzakelijkerwijze door dezelfde processor worden uitgevoerd. We zullen in dit hoofdstuk aan de hand van een in onze groep ontwikkeld besturingssysteem voor procescomputers nagaan, waaruit het pakket systeemprogrammatuur moet bestaan. Zie paragraaf 3.2.

Dit besturingssysteem, EROS genaamd, is een real-time systeem met een

strak modulaire opbouw. In elke module wordt een zeer nauwkeurig omschreven organisatietaak uitgevoerd, zoals bijvoorbeeld opbouwen van werklijst (Scheduler), "afbouwen" van werklijst (Executive), geheugenbeheer (Diskmodule).

In een multiprocessorsysteem kunnen de organisatieprogrammas en de applicatieprogrammas op vele manieren over de aanwezige processoren verdeeld worden (Zie hoofdstuk 2). Dit heeft gevolgen voor het besturingssysteem in een multiprocessorsysteem. In paragraaf 3.3 wordt aan de orde gesteld welke funkties men in een multiprocessorbesturingssysteem kan aan treffen.

In de vierde paragraaf van dit hoofdstuk worden enkele mogelijke opzetten van besturingssystemen in een multiprocessor-omgeving gegeven. Deze paragraaf is onderverdeeld in twee subparagrafen, waarin een mogelijke realisatie van dynamische toewijzing van de organisatie (par. 3.4.1.) respectievelijk twee mogelijke opzetten van een statische toewijzing van de organisatie (par. 3.4.2.) worden behandeld.

3.2 De organisatie in een uniprocessorsysteem

De buitenwereld (een randapparaat, een geregelde grootheid van een proces, etc.) kan op elk willekeurig moment aandacht van het computersysteem vragen door het opsteken van een vlag. Op deze vlag kan op twee manieren worden gereageerd:

- "Op interruptie" basis. Karakteristiek voor deze manier is dat de buitenwereld het tijdstip van onderbreking bepaalt.

  Treedt er een interruptie-aanvraag op, dan maakt de processor de lopende instruktie af, waarna naar een vaste plaats in het geheugen wordt gesprongen (aangenomen dat er maar één interruptieniveau is). Op deze plaats moet de "interruptie-routine" staan. Omdat het nieuw aan te lopen programma de processorstatus kan vernietigen, moet deze gered worden. Daarom begint de interruptieroutine met een redprocedure. Hierna wordt de herkomst van de interruptie geïdentificeerd.
- Door de "vlagtest". Karakteristiek voor deze manier is dat de binnenwereld (de programmatuur) het tijdstip van onderbreking bepaalt.

  De processor reageert nu niet instantaan op een vlag, maar test

de vlaggen (van de randapparaten) op tijdstippen, die voor het lopende programma gunstig zijn (bijvoorbeeld na beëindiging van programma). Gezien het tijdgebonden karakter van computer control toepassingen bestaat de kans dat een programma c.q. commando niet snel genoeg uitgevoerd wordt. Dit kan te wijten zijn aan een programma, dat de processor te lang bezighoudt. Daarom wordt bij deze werkwijze van de programmeur geëist, dat hij langdurige programmas in kleine eenheden verdeeld. Tussen twee eenheden moet het testen van vlaggen kunnen plaats vinden. Evident is dat deze gang van zaken de programmeur in zijn vrijheid beperkt.

In EROS is voor de eerste manier gekozen. Nadat de herkomst van de interrupties geïdentificeerd is, worden de taakaanvragen verzameld in een werklijst (in EROS is deze werklijst ProcesVerzamelBuffer genoemd; afkorting P.V.B.). Dit is noodzakelijk omdat er tijdelijk meer aanvragen voor programmas geplaatst kunnen worden, dan er uitgevoerd kunnen worden.

Het plaatsen van aanvragen in het procesverzamelbuffer gebeurt door de Scheduler, die voor dit doel de volgende modulen bevat:

- een Prioriteitsprocessor
- een Eventprocessor
- een Klokroutine

De prioriteitsprocessor kent aan de geïdentificeerde interruptie (de taakaanvraag), door middel van een tabel (prioriteitentabel genoemd) een prioriteit toe. Deze prioriteitswaarde (dit is in EROS een systeemgegeven) manifesteert zich door zijn plaats in de werklijst: hoe hoger de plaats van de aanvraag in de werklijst, hoe groter de prioriteit, des te eerder de taak aan de beurt komt.

De Eventprocessor kan actie nemen onder controle van een te detekteren gebeurtenis, zoals het aanvragen (via de prioriteitsprocessor) van programmas. Voorbeeld van eventprocessing: meld dat de interruptie i heeft plaats gevonden en kijk of er iemand op wacht.

De Klokroutine is een bijzonder geval van de Eventprocessor. Akties kunnen hierbij ondernomen worden op basis van tijd. Met behulp van de Klokroutine is het onder andere mogelijk de maximaal toegestane uitsteltijd van een (real-time) taak te bewaken. Indien deze tijd wordt overschreden is adequate aktie vereist (bijvoorbeeld genereren van alarmboodschap op teletype).

Nadat de aanvragen zijn verzameld gaat de controle van de Scheduler over naar het organisatieprogramma dat de werklijst afzoekt (de Executive).

Het afzoeken begint bovenaan (hoogste prioriteit). De gevonden taakaanvraag verwijst indirekt naar een bij de taak horende lijst, de jobhead, welke alle informatie (zoals beginadres, processorstatus, programmastatus, core resident? , etc.) bevat, die noodzakelijk is voor het aanlopen c.q. hervatten van het taakprogramma.

Na beëindiging van de taak gaat de controle weer terug naar de Executive. Deze veegt de taakaanvraag uit de werklijst en zoekt de volgende taakaanvraag op.

Het opbouwen en "afbouwen" van werklijsten voor alle gemeenschappelijke systeemfaciliteiten vormt de basis van het operating systeem EROS. Deze "Opbouw-Afbouw-Werklijst"-struktuur (afkorting: O-A-W struktuur) is algemeen. Zij kan gebruikt worden om alle konflikten, die ontstaan met betrekking tot het gemeenschappelijk gebruik van een bepaalde systeemfaciliteit in een uniprocessorsysteem op te lossen. Zie figuur 3.1.

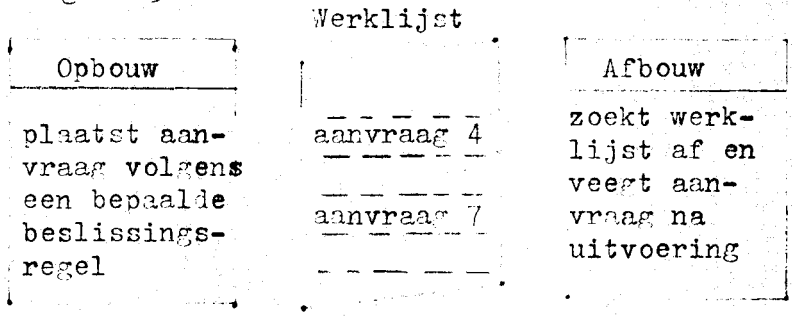


fig. 3.1 Opbouw-Afbouw-Werklijst-struktuur

Bij elke werklijst hoort een beslissingsregel, welke de volgorde bepaalt, waarin aanvragen worden afgehandeld. Voorbeelden van zulke beslissingsregels zijn:

- afhandeling op basis van prioriteit
- cyclische afhandeling (round robin)
- first in. first out
- last in, first out

Bij de verdeling van de processortijd door middel van het P.V.B. is de beslissingsregel de prioriteit van het programma.

In bijna alle toegevoegde organisatieprogrammas (systeemmodulen) van EROS komt de O-A-W struktuur terug. Zie figuur 3.2. In de Diskmodule die alle organisatie rond de disktransporten verzorgt wordt met behulp van de O-A-W struktuur de volgorde van de transporten

geregeld. Er kunnen namelijk tijdelijk meer disktransporten aangevraagd zijn, dan er uitgevoerd kunnen worden. In de Typmodule wordt met behulp van deze struktuur de afhandeling van te typen teksten geregeld. En ook de communicatiemodule waarmee het mogelijk is om op conversationele wijze, voor maximaal vier gebruikers, met het systeem te communiceren gebruikt deze struktuur om de afhandeling van de verschillende gebruikers te regelen.

De bovengenoemde drie systeemmodulen (ook wel systeemtaken genoemd) worden op dezelfde wijze als taakprogrammas aangevraagd in het P.V.B. In dit P.V.B. wordt de processortijd verdeeld onder alle in het systeem te verrichten werkzaamheden

Omdat zonder processor het systeem geen programmas kan uitvoeren is het evident dat in een uniprocessorsysteem de verdeling van de processortijd op het hoogst bestuurlijke niveau geschiedt.

Er zijn in EROS enkele modulen, welke niet via het P.V.B. aangevraagd worden. Dit zijn modulen met een zeer korte verwerkingstijd (Keyboardmodulen) of met een zeer hoge prioriteit (Klokroutine)

Tot slot van deze paragraaf volgt in onderstaande figuur een overzicht van de struktuur van het operating systeem EROS.

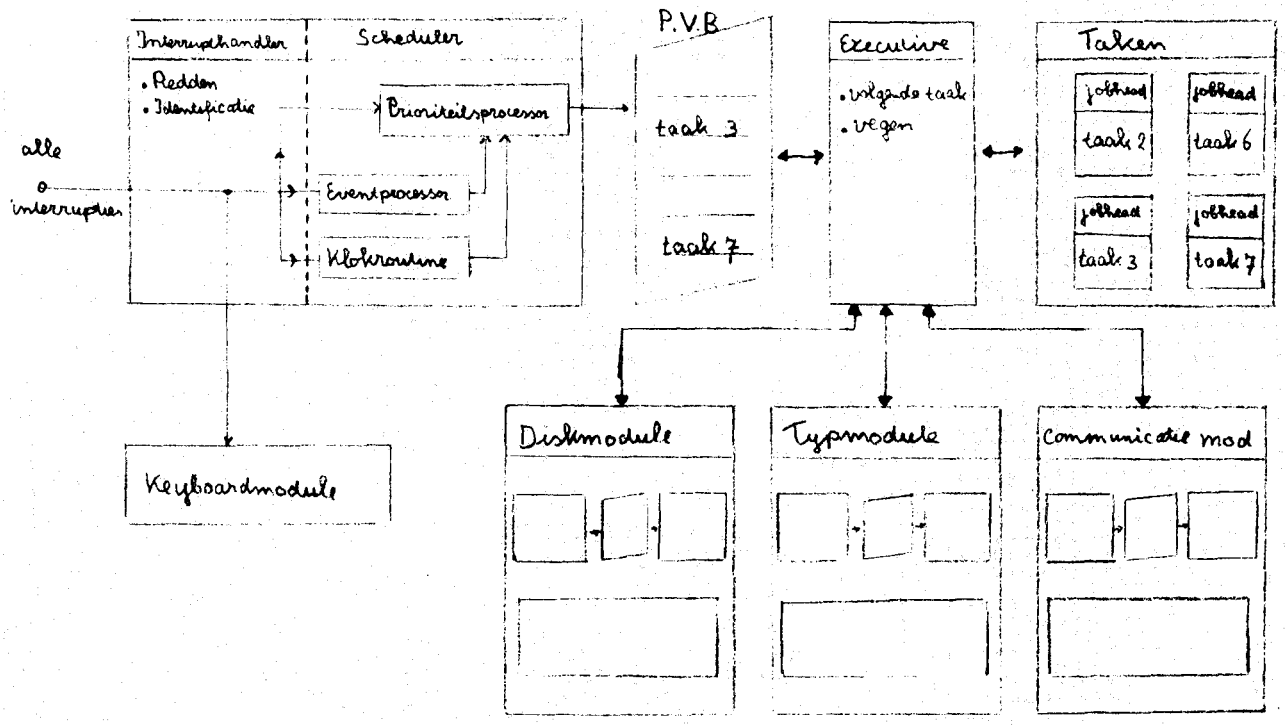


fig. 3.2 Schematisch overzicht van de struktuur van het operatingsysteem EROS

3.3 Organisatie-funkties in een multiprocessorsysteem

In de vorige paragraaf (3.2) is behandeld welke funkties een real-time besturingssysteem in een uniprocessorsysteem moet bezitten. In een multiprocessorsysteem is het mogelijk de programmatuur op vele manieren over de aanwezige processoren te verdelen. Dit levert veel verschillende systeemontwerpen op (zie hoofdstuk 2), welke aanleiding geven tot even veel verschillende besturingssystemen, die we hier uiteraard niet uitgebreid kunnen behandelen.

We moeten ons daarom beperken; we zullen in deze paragraaf nagaan welke extra funkties nodig zijn in een multiprocessor-omgeving en welke funkties gewijzigd dienen te worden in vergelijking met een uniprocessorsysteem. Tevens introduceren we enkele nieuwe begrippen.

In een later stadium (par. 3.4, hoofdstuk 5 en 6) wordt de onderlinge samenhang van de in deze paragraaf genoemde funkties en begrippen nader aan de orde gesteld.

Enkele begrippen

Bij de beschrijving van multiprocessorprogrammatuur kunnen we onderscheid maken tussen "taakprocessoren" en "organisatieprocessoren"; wanneer een processor een applicatieprogramma uitvoert noemen we hem een taakprocessor; op het moment dat hij een organisatieprogramma (of een deel ervan) uitvoert heet hij organisatieprocessor(*)

Overeenkomstig het bovenstaande kunnen we een funktionele scheiding maken tussen een :

- taakwerklijst
- systeemwerklijst.

In de taakwerklijst staan aanvragen voor applicatietaken op uitvoering te wachten. Deze taken worden door taakprocessoren in behandeling genomen. In de systeemwerklijst staan de systeemtaken aangevraagd. Deze worden door de organisatieprocessor uitgevoerd. In het geval dat zowel systeemtaken als applicatietaken door middel van dezelfde werklijst worden aangevraagd spreken we van de werklijst. (vergelijk uniprocessorsysteem: P.V.B.)

(*) Deze naamgeving zal niet in alle gevallen even streng worden toegepast.

In een systeem, waarin vele processoren werkzaam zijn, is een lijst noodzakelijk, waarin wordt bijgehouden met welk programma elke processor zich bezighoudt. Deze lijst geven we de naam processor-bezettingstabel (kortweg: bezettingstabel).

Funkties binnen een multiprocessor-besturingssysteem

Een multiprocessor-besturingssysteem kan of moet de volgende funkties bevatten:

1. het verwerken van alle interrupties en/of vlaggen.
2. het bedienen van alle randapparaten.
3. het beheren van het gemeenschappelijk geheugen (voor-en achtergrond).
4. het onderhouden van de door de gebruiker gewenste communicatie.
5. het vastleggen van de volgorde waarin taken moeten worden uitgevoerd.
6. het "afbouwen" van (een) werklijst(-en)
7. het overdragen van het besturingssysteem.

De punten 1, 2, 3 en 4 treft men in nagenoeg ongewijzigde vorm in een uniprocessorsysteem aan en verdienen daarom hier geen nadere aandacht. Zie hiervoor paragraaf 3.2 en de literatuur (*).

Bij punt 5 kan het volgende worden opgemerkt. In het geval dat het totale pakket applicatieprogrammatuur dynamisch aan de processoren is toegewezen stelt de organisatieprocessor één (taak-) werklijst voor alle processoren op; wanneer elke processor een vaste taak of groep taken afhandelt (Statische toewijzing), is het nodig dat de organisatieprocessor meerdere (taak-) werklijsten opstelt.

We gaan er bij de bespreking van punt 6 van uit dat elke processor elke taak kan uitvoeren, zodat de organisatieprocessor dus één (taak-)werklijst opstelt.

In het algemeen kan gesteld worden dat het "afbouwen" van de (taak-)werklijst in twee situaties geaktiveerd wordt:

- nadat een processor met een taak "klaar" is (en een volgende in uitvoering wil nemen)
- nadat een nieuwe taakaanvraag in de werklijst is geplaatst (en onderzocht moet worden of deze onmiddellijk in uitvoering genomen moet worden)

(*) Afstudeerverslagen van J.M.L. Kouwenberg en F.P.M. Sopers, R.Thijs en J. Nieron.

We kunnen met betrekking tot het afbouwen van de werklijst drie mogelijkheden onderscheiden:

1. het afbouwen van de (taak-)werklijst geschiedt door de organisatieprocessor ten behoeve van een taakprocessor.
2. het afbouwen van de (taak-)werklijst geschiedt door de taakprocessor.
3. het afbouwen van de (taak-)werklijst geschiedt door de organisatieprocessor ten behoeve van zichzelf.

ad 1 In dit geval is de organisatieprocessor de enige die de (taak-)werklijst bewerkt. De in de "afbouw"-routine gevonden taakaanvraag moet aan een taakprocessor worden toegewezen (hoe dit in zijn werk gaat wordt behandeld in hoofdstuk 6): In het geval een taakprocessor zich klaar meldt is het duidelijk aan wie toegewezen moet worden; wanneer echter een taak met hoge prioriteit aangevraagd wordt, moet volgens een bepaald allocatie criterium (zie blz 3.9) worden toegewezen.

Wanneer de organisatieprocessor nooit de taakprocessoren opdraagt een reeds in behandeling zijnde taak te onderbreken ten gunste van een taak met hogere prioriteit spreken we van vrije taakprocessoren (zie blz. 3.19). Het initiatief om een nieuwe taak te krijgen gaat in dit geval altijd van de taakprocessoren uit (klaarmelding). In het andere geval spreken we van gedwongen taakprocessoren.

ad 2 In dit geval wordt de taakwerklijst door meerdere processoren bewerkt: de organisatieprocessor stelt hem op en de taakprocessoren breken hem zelfstandig en op eigen initiatief af. Zie par. 3.4.1 en 3.4.2.

Omdat op elk tijdstip maar één processor de werklijst mag bewerken, introduceren we in dit geval een "kritisch" tabel. De toegang tot deze kritische tabel moet met behulp van een lock/unlock mechanisme geregeld worden. Voor de beschrijving van dit mechanisme wordt verwezen naar appendix E.

ad 3 In dit geval stelt een organisatieprocessor de werklijst op en breekt een organisatieprocessor deze af. Dit kunnen telkens andere organisatieprocessoren zijn, doch op elk tijdstip is er slecht één; deze organisatieprocessor neemt de in de werklijst gevonden taakaanvraag zelf in behandeling. Dit is een analoge

situatie als in een uniprocessorsysteem. Op deze wijze omzeilen we de kritische tabel uit het vorige geval (zie ad 2).

Een konsekwentie van deze gang van zake in een multiprocessorsysteem is, dat de organisatieprocessor, nadat hij een taak in behandeling neemt, het besturingssysteem aan een andere processor moet kunnen overdragen (zie behandeling punt 8), ten einde een volgende processor in staat te stellen een taakaanvraag uit de werklijst in uitvoering te nemen. We hebben in dit geval dus te maken met een dynamische toewijzing van de organisatie.

Deze manier van werken wordt in hoofdstuk 5 nader uitgewerkt.

Wanneer in een multiprocessorsysteem een taak in behandeling wordt genomen is het noodzakelijk de corresponderende aanvraag uit de werklijst onmiddellijk te markeren (blokkeren of vegen), omdat anders de mogelijkheid bestaat dat eenzelfde taak door meerdere processoren wordt uitgevoerd.

Als we over "het overdragen van het besturingssysteem" (punt 8) spreken, zijn we vanzelf beland in een systeem met een "dynamische toewijzing van de organisatie". Bij een dynamische toewijzing van de organisatie wordt telkens een andere processor met het besturingssysteem belast, echter te allen tijde slechts één. Het besturingssysteem moet nu over een toegevoegde systeemfunktie ("Gidsroutine") beschikken, die volgens een allocatie criterium uitzoekt aan welke processor het besturingssysteem moet worden toegewezen.

Het meest voor de hand liggende allocatie criterium is, dat aan die processor wordt toegewezen, die op dat moment de taak met de laagste prioriteit in behandeling heeft. Meer geavanceerde allocatie citeria ten einde de meest optimale werking van het systeem te bereiken zijn mogelijk.

Allocatie is een gevolg van dynamische toewijzing en is dus zowel nodig bij - het toewijzen van urgente taken aan "gedwongen taakprocessoren".

als bij - het overdragen van het besturingssysteem.

Het moment waarop de "Gidsroutine" wordt aangelopen verschilt in de nog te behandelen systeemopzetten. Dit moment kan namelijk zijn:

- telkens nadat een nieuwe taak door de organisatieprocessor is gevonden; zie voor de uitwerking van deze mogelijkheid hoofdstuk 5 "Symmetrisch of anoniem systeem"

- nadat alle aangevraagde systeemtaken (organisatie) zijn afgehandeld en de organisatieprocessor daarna de taakwerklijst gaat afzoeken en een taak vindt; deze gang van zake wordt nader aan de orde gesteld in paragraaf 3.4.1. "Dynamische toewijzing van de organisatie".

Een konsekwentie van het overdragen van het besturingssysteem is dat alle interruptie-, data-, en adreslijnen van alle randapparaten naar de nieuwe organisatieprocessor geleid moet worden. Dit wordt verwezenlijkt door een schakeling, welke de naam "gids" draagt.

3.4 Mogelijke opzetten van besturingssystemen in een multiprocessorsysteem

In deze paragraaf wordt aan de onderlinge samenhang van de in de vorige paragraaf gegeven funkties en begrippen aandacht besteed. Dit gebeurt aan de hand van mogelijke opzetten voor besturingssystemen.

Deze paragraaf is onderverdeeld in twee subparagrafen. In de eerste subparagraaf (3.4.1) wordt een mogelijke opzet van dynamische toewijzing van de organisatie gegeven. Bij een dynamische toewijzing van de organisatie is elke processor in staat het besturingssysteem (of een (hoofd-)deel ervan) in uitvoering te nemen. In de tweede subparagraaf worden twee mogelijke opzetten van statische toewijzing van de organisatie gegeven. Hierbij is steeds dezelfde processor (in een centraal systeem) of dezelfde groep processoren (in een decentraal systeem) met het besturingssysteem belast. Er zal aandacht worden besteed aan saillante voor- en nadelen van deze opzetten.

In beide subparagrafen gaan we ervan uit dat de processoren dynamisch aan de applicatieprogramma's zijn toegewezen, met andere woorden de organisatieprocessor stelt één taakwerklijst op voor alle processoren.

Tot slot zij nog opgemerkt dat de hier gepresenteerde opzetten géén uitwerkingen zijn; noch wordt gepretendeerd dat het alle mogelijke opzetten van besturingssystemen zijn. Voor de uitwerking van enkele systemen wordt verwezen naar de hoofdstukken 5 en 6.

3.4.1 Dynamische toewijzing van de organisatie

Noodzakelijke systeemopbouw

Dynamische toewijzing van de organisatie is mogelijk indien het systeem symmetrisch van opbouw is dat wil zeggen dat alle aanwezige

processoren identiek zijn en toegang hebben tot een (groot) gemeenschappelijk geheugen, tot alle input/output kanalen en alle randapparaten. Een "omvangrijke" interface (gids) tussen randapparaten en processoren zorgt ervoor dat van alle mogelijke wegen van elk randapparaat naar alle processoren op elk tijdstip slechts één weg (deze bestaat uit adres-, data-, en statuslijnen) geldig is. In de appendix E is deze interface voor dit geval schematisch weergegeven. In het gemeenschappelijk geheugen bevinden zich op vaste plaatsen alle organisatieprogrammas. Voor het gemeenschappelijk gebruik van geheugens zie hoofdstuk 4.

Dynamische toewijzing

Het idee achter de dynamische toewijzing van de organisatie is hier het volgende: Elke processor voert zélf die systeemprogrammas uit, die onafscheidelijk zijn verbonden met de uitvoering van een taak en die programmas die nodig zijn voor het bemachtigen van een nieuwe taak (zoals onder andere de "Afbouw"-routine), als de oude taak beëindigd is of onderbroken wordt. Er zijn echter vele systeemprogrammas die door elke processor uitgevoerd zouden kunnen worden, zoals onder andere het verwerken van interrupties of het bedienen van input-/output apparaten. De processor die laatstgenoemde funkties uitvoert wordt de organisatieprocessor genoemd.

Elke processor in het systeem is in staat organisatieprocessor te zijn. Op elk tijdstip mag er slechts één organisatieprocessor zijn. Deze bestaat op die momenten waarop er behoefte is aan een processor die zich bezighoudt met organisatie; de rest van de tijd houdt deze processor zich bezig met het uitvoeren van taakprogrammas, evenals de andere processoren. Iedere andere processor kan echter op elk "willekeurig" moment de organisatie toegewezen krijgen. (Zie blz. 3.14)

In het nu volgende wordt puntsgewijs een aantal belangrijke zaken toegelicht, waarna in een afzonderlijk deel (Zie blz. 3.14) het principe van de werking van de dynamische toewijzing aan de orde wordt gesteld.

De funkties van de organisatieprocessor in dit systeem

- het verwerken van alle interrupties c.q. vlaggen (zie par. 3.2)
- het bedienen van alle randapparaten (zie par. 3.2)
- het onderhouden van de door de gebruiker gewenste communicatie (zie par. 3.2)

- het beheren van het gemeenschappelijk geheugen (zie par. 3.2)
- het opstellen van een werklijst (taakwerklijst) voor de taakprocessoren.
- het opstellen van een werklijst (systeemwerklijst) voor de organisatieprocessor (zichzelf)
- het overdragen van de funktie van organisator aan een ander processor.

In deze opzet wordt een scheiding gemaakt tussen systeemtaken en applicatietaken. De systeemtaken worden door middel van de systeemwerklijst aangevraagd en door de organisatieprocessor sequentiëel afgehandeld. De applicatietaken worden door middel van de taakwerklijst aangevraagd en parallel door de taakprocessoren uitgevoerd. De taakwerklijst is nu een gemeenschappelijke tabel voor alle processoren. Daarom is deze tabel "kritisch". Hoe deze tabel bewerkt wordt, wordt behandeld op bladzijde 3.13. Hoe en op welke momenten de funktie van organisator aan een andere processor wordt overgedragen wordt behandeld bij "het principe van de werking van dynamische toewijzing van de organisatie" op bladzijde 3.14

De taakprocessor

De term taakprocessor wordt hier gebruikt als tegenhanger van organisatieprocessor. Het klinkt daarom paradoxaal dat "taakprocessoren" in dit systeem zelf die systeemfunkties uitvoeren, die onafscheidelijk verbonden zijn met de uitvoering van een applicatietaak of met het zoeken naar een nieuwe taak. In het nu volgende wordt dit toegelicht.

Nadat een "taakprocessor" toegang heeft gekregen tot de taakwerklijst (via een lock/unlock mechanisme) zoekt hij deze met behulp van een "Afbouwroutine" af. Wanneer een potentiële taak gevonden is, wordt de corresponderende aanvraag onmiddellijk geveegd. Hierna geeft de "taakprocessor" de toegang tot de werklijst vrij voor een andere processor (onder voorbehoud: zie bezettingstabel op blz. 3.13) en voert zelf de volgende handelingen uit:

- uitzoeken waar de taak zich bevindt (start-adres)
- onderzoeken van de status van de taak en afhankelijk hiervan:
- uitvoeren van juiste aktie (bijvoorbeeld aanlopen, ontredden).

Na het uitvoeren van bovenstaande handelingen wordt de betreffende processor een echte taakprocessor. Verder moet nog opgemerkt worden, dat een taakprocessor, zoals later zal blijken, in staat moet zijn een taak te redden (= het op een bepaalde geheugenplaats bewaren van

van de programmastatus)

De taakwerklijst en de bezettingstabel

Een van de bezigheden van de organisatieprocessor is het opstellen van een zogenaamde taakwerklijst. Hierin staan de aanvragen voor applicatieprogrammas op volgorde van prioriteit. Deze werklijst moet toegankelijk zijn voor alle processoren en staat daarom in het gemeenschappelijk geheugen. Het is belangrijk op te merken dat te allen tijde slechts één processor toegang tot de lijst mag hebben. Van de kant van de taakprocessoren gezien voorkomt deze enkelvoudige toegang dat verschillende taakprocessoren eenzelfde taak uitvoeren (Bij het werken met een pointer in de werklijst kan zelfs een taakaanvraag verloren gaan, indien de tabel niet gelockt wordt). De organisatieprocessor neemt de beslissing het besturingssysteem aan een andere processor over te dragen. Deze beslissing berust op de stand van zaken in de werklijst én de bezettingstabel. Om te voorkomen dat een foutieve beslissing genomen wordt mag deze stand van zaken dan ook niet veranderen terwijl de beslissing genomen wordt.

In de bezettingstabel is elke processor benoemd. Hierin wordt bijgehouden met welke taak (prioriteit) elke processor bezig is. Nadat een "taakprocessor" een potentiële taak in de werklijst heeft gevonden, plaatst hij direkt in de bezettingstabel op zijn plaats de prioriteit van de gevonden taakaanvraag. Er staat hierdoor in de bezettingstabel altijd up-to-date informatie.

Zoals uit het bovenstaande blijkt komen de toegang tot de werklijst en de toegang tot de bezettingstabel altijd in combinatie voor. Het is om deze reden dat de toegang voor beide tabellen door dezelfde lock-bit geregeld moet worden. Zie figuur 3.3

Omdat processoren , op gelijkwaardige basis, de toegang tot een kritische tabel niet

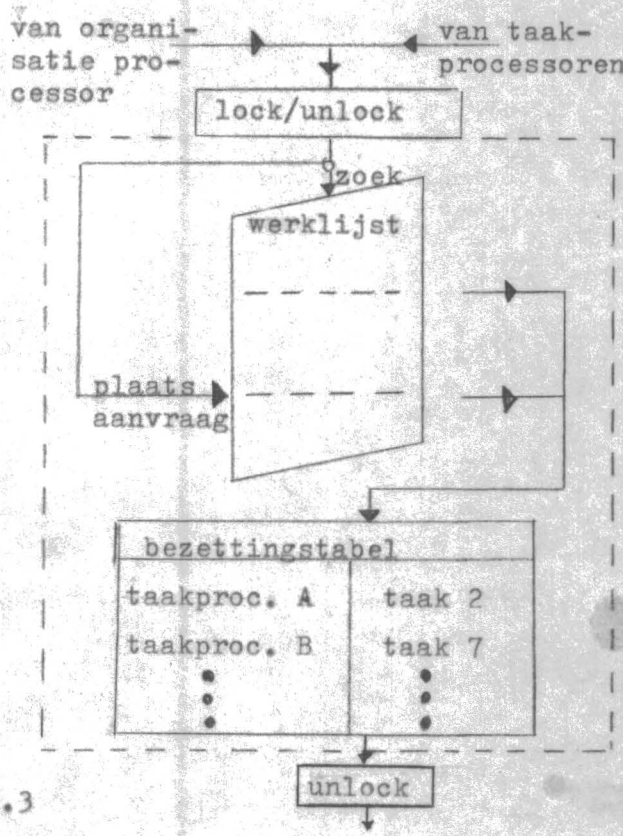


fig. 3.3

zelfstandig kunnen coördineren, moet de hulp worden ingeroepen van extra hardware. Deze hardware zorgt ervoor dat te allen tijden slechts één processor de lock-bit mag testen, die toegang geeft tot het testen van de lock-bit van de werklijst. Zie appendix D.

Processor communicatie

Het is in een multiprocessorsysteem noodzakelijk dat processoren met elkaar kunnen communiceren. In het bijzonder moeten organisatieprocessor en taakprocessor in staat zijn opdrachten/data naar elkaar te sturen. Deze communicatie gaat via het gemeenschappelijk communicatiegeheugen. Een gedeelte van het gemeenschappelijk geheugen kan als zodanig dienen. In dit communicatiegeheugen heeft elke processor een vast LEES-en SCHRIJF-veld. Een uitwerking van dit statisch gealloceerde communicatiegeheugen wordt gegeven in hoofdstuk 6

"Interne" interrupties

Even belangrijk in deze opzet is het, dat taakprocessoren de organisatieprocessor moeten kunnen interrumperen (bijvoorbeeld om te melden dat er data in het veld SCHRIJF klaar staan, die via randapparaat x naar buiten moeten worden gebracht of om vanuit een taak een andere taak aan te vragen etc). De interrupties die door de taakprocessoren worden gegenereerd, moeten door de interface tussen de processoren automatisch naar de momentane organisatieprocessor geleid worden. Na de interruptieverwerking haalt de organisatieprocessor de code uit het veld SCHRIJF van de interrumperende processor op en bepaalt hieruit zijn opdracht.

De organisatieprocessor is in staat met behulp van één output-instruktie acht taakprocessoren rechtstreeks te interrumperen. Wat met de interruptie bedoeld wordt, kan in het veld LEES van de betreffende processor gespecifieerd worden.

Het principe van de werking van dynamische toewijzing van de organisatie

Bij de dynamische toewijzing kan de organisatie door elke processor gedaan worden. Dit houdt in dat één van de taken van de organisatieprocessor is: het overdragen van de organisatie aan een andere processor.

Nadat alle interrupties/vlaggen en alle aangevraagde systeemtaken (zoals Typmodule, Communicatiemodule, etc.) verwerkt zijn, zoekt de

organisatieprocessor de taakwerklijst af (na toegang te hebben verkregen via het lock/unlock mechanisme) en neemt de taak met de hoogste prioriteit in behandeling. Alvorens tot daadwerkelijk uitvoering over te gaan, wordt volgens een bepaald allocatie criterium uitgezocht wie de funktie van organisator opgedragen krijgt. Een mogelijk criterium is:

- als de prioriteit van de gevonden taak groter is dan de laagste prioriteit uit de bezettingstabel, draag dan de funktie van organisator over aan de processor, die de laagst priore taak uitvoert.
- als de prioriteit van de gevonden taak kleiner is dan de laagste prioriteit uit de bezettingstabel, neemt de organisatieprocessor de taak wel in uitvoering, en blijft organisatieprocessor.

Het uiteindelijke overdragen van het besturingssysteem bestaat hierin, dat de interface (gids) tussen randapparaten en processoren én tussen processoren onderling zodanig gezet wordt door de organisatieprocessor dat alle adres-, data- en interruptielijnen naar de nieuwe organisatieprocessor leiden.

De toekomstige organisatieprocessor werkt voorlopig nog ongestoord aan zijn taak, totdat na verloop van tijd de eerste interruptie binnenkomt (bijvoorbeeld van I/O apparaat). Op dat moment ondergaat deze processor de gedaantewisseling van taakprocessor naar organisatieprocessor. De gang van zaken bij deze processor is nu alsvolgt:

- hij redt de taak (status onderbroken)
- hij herstelt de aanvraag in de taakwerklijst en
- veegt de prioriteit uit de bezettingstabel.

Deze nieuwe organisatieprocessor kan bovenstaande handelingen gemakkelijk uitvoeren omdat hij zelf de processor is, die de onderbroken taak in behandeling had, met andere woorden hij beschikt zelf over de juiste gegevens omtrent de taak. Daarna houdt deze processor zich bezig met de noodzakelijke organisatie, totdat zijn "tijd van gaan" aanbreekt.

Een belangrijke faciliteit van de organisatieprocessor is het ogenblikkelijk in behandeling nemen van zeer urgente taakaanvragen. Het begrip "zeer urgente taakaanvraag" wordt door de proceskundige vertaald in een bepaalde prioriteitswaarde. Wanneer een taak aangevraagd wordt, waarvan de prioriteit groter is dan deze prioriteitswaarde, werkt de organisatieprocessor niet langer aan het besturings-

systeem, maar begint terstond (na aanpassing van de bezettingstabel en het overdragen van de organisatie) aan de uitvoering van deze taak. In dit geval moet de nieuwe organisatieprocessor tegelijkertijd met het overdragen van het besturingssysteem geïnterrumpeerd worden, opdat direkt met de organisatie wordt verder gegaan.

Voordelen van deze opzet

1. Opzet met meest evenwichtige verdeling van de belasting over de aanwezige processoren.(*)
2. Opzet met de grootst mogelijke doorzet omdat elke processor tot het maximale benut wordt.(*)
3. Zeer urgente taakaanvragen worden direkt verwerkt.
4. Taken met hoge prioriteit worden niet onderbroken

Nadelen van deze opzet

1. "Uitgebreide" interface tussen randapparaten en processoren en tussen processoren onderling is noodzakelijk.
2. Het coördineren van de toegang tot de gemeenschappelijke taakwerklijst eist toevoeging van extra hardware en software.
3. Sommige routines moeten reëntrant zijn of anders moeten meerdere kopieën aanwezig zijn.
4. Omdat elke processor de organisatie opgedragen kan krijgen moet elke processor zijn taak kunnen redden.

3.4.2. Statische toewijzing van de organisatie

In deze subparagraaf worden twee mogelijkheden van statische toewijzing van de organisatie besproken. De eerste mogelijkheid ligt voor de hand. We vragen ons namelijk af wat de gevolgen zijn, als we de opzet uit de vorige subparagraaf ("dynamische toewijzing van de organisatie") statisch maken (mogelijkheid 1). De tweede mogelijkheid volgt als het ware "logischerwijze" uit de eerste mogelijkheid (zie mogelijkheid 2 op bladzijde 3.18). We gaan er in deze subparagraaf weer vanuit dat de processoren dynamisch aan de applicatietaken zijn toegewezen.

Mogelijkheid 1

In deze opzet is de funktie van de organisatieprocessor dezelfde als bij de dynamische toewijzing (zie bladzijde 3.11) met dien verstande dat de taak "het overdragen van de funktie van organisator aan de proces-

(*) aangenomen dat de organisatieprocessor geen bottle-neck vormt.

sor met de minst priore taak" en alle daarbij horende hulptaken komen te vervallen. De organisatieprocessor verwerkt alle interrupties en bedient alle input/output apparaten. De "omvangrijke" interface tussen randapparaten en processoren, die ervoor zorgt dat elk randapparaat met elke processor verbonden kan worden, komt hierdoor te vervallen.

Een zeer belangrijke funktie van de organisatieprocessor is het verdelen van het werk door middel van het opstellen van de taakwerklijst welke toegankelijk is voor alle processoren. De taakprocessoren voeren ook in deze opzet zelfstandig die routines uit die nodig zijn voor het zoeken naar een nieuwe taak. Zij zoeken op eigen initiatief, na het beëindigen van een taak, de taakwerklijst af (met behulp van de "afbouw" routine). Daarom moet ook hier de toegang tot deze werklijst gecoördineerd worden. Deze coördinatie kan op dezelfde wijze geschieden als bij de dynamische toewijzing (hardware oplossing). Een andere oplossing is: Laat de organisatieprocessor de toegang tot de werklijst regelen (software oplossing). De procedure is dan als volgt: Als een processor in de werklijst wil, wordt aan de organisatieprocessor toegang gevraagd. Laatstgenoemde houdt nu een "Toegangstabel" bij. In deze tabel staan eventuele aanvragen te wachten. Er zijn nu twee mogelijkheden:

- er zijn geen wachtenden; de taakprocessor krijgt nu direkt toestemming tot toegang.
- er zijn wel wachtenden; de aanvraag wordt volgens een bepaalde beslissingsregel (bijvoorbeeld fifo) in de tabel geplaatst. Te zijner tijd krijgt deze processor toegang.

Bovenbeschreven software oplossing is niet aanbevelenswaardig. Wat is nog het nut van het centraal opstellen van de werklijst als de toegang decentraal geregeld wordt? De vraag komt op "Kan de organisatieprocessor in de tijd dat hij de toegang tot de tabel regelt niet beter de tabel zèlf bewerken en de nodige informatie aan de betreffende taakprocessor doorsturen?" We ontlasten op deze wijze de taakprocessoren van organisatorische werkzaamheden. Dit blijkt voordelen te bezitten en leidt tot een aparte opzet (mogelijkheid 2), waaraan in het tweede deel van deze paragraaf aandacht wordt besteed.

De bezettingstabel is in deze opzet niet noodzakelijk. Hij kan eventueel van nut zijn als de operateur een overzicht van de werkzaamheden van de processoren wil hebben (bijvoorbeeld als onderdeel van de Communicatie module).

Een gevolg van het statisch maken van de organisatie is dat een processor, die aan een taak begonnen is, deze taak ononderbroken afmaakt. (Bij dynamische toewijzing kan een processor onderbroken worden doordat deze de uitvoering van het besturingssysteem opgedragen krijgt). Hierdoor komen de routine, welke de programmastatus redt en de redruimte te vervallen.

Een gevolg van bovenstaande is dat als een taak met hoge prioriteit wordt aangevraagd deze op uitvoering moet wachten, totdat één van de taakprocessoren een taak beëindigt. Dit moet geaccepteerd worden. Men beseffe dat dit niet altijd betekent dat er langer gewacht moet worden, dan in het geval een taakprocessor tot het redden (processor- en programmastatus) van zijn taak gedwongen wordt.

De keuze is vrij om in deze opzet de organisatieprocessor een taakprogramma te laten uitvoeren in de tijd dat er geen organisatorische werkzaamheden te verrichten zijn. De responsietijd zal dan gemiddeld iets verslechteren.

Ook is het mogelijk de organisatieprocessor zéér urgente, aangevraagde, taakprogrammas (bijvoorbeeld alarmprogramma) te laten uitvoeren. Dit verbetert de responsietijd van betrokken programma maar verslechtert die van andere programmas.

Voordelen van deze opzet

1. "Omvangrijke" interface tussen randapparaten en processoren komt te vervallen.
2. De taakprogrammas worden "ononderbroken" uitgevoerd, hetgeen tot gevolg heeft dat er geen "programma-status" redroutine aanwezig hoeft te zijn.

Nadelen

1. Het coördineren van de toegang tot de taakwerklijst heeft de hulp nodig van extra hardware en software.
2. Sommige routines moeten reëntrant zijn of anders moeten meerdere kopieën aanwezig zijn.

Mogelijkheid 2

In de vorige opzet werden nog enkele organisatieprogrammas door de taakprocessoren uitgevoerd. In de nu gepresenteerde opzet wordt de mogelijkheid onderzocht de taakprocessoren hiervan zoveel mogelijk

te ontlasten. We creëren op deze wijze échte taakprocessoren en een superorganisator of "supervisor". Met name het afzoeken van de werklijst en red-akties kunnen de taakprocessoren bespaard blijven. Wat de konsekwenties hiervan zijn wordt in het volgende behandeld

Als de organisatieprocessor de enige is, die toegang tot de taakwerklijst heeft, wordt het lock/unlockmechanisme, alsmede het reëntrant zijn van de "Afbouw"-routine overbodig. Het is nu niet langer noodzakelijk dat de taakwerklijst in het gemeenschappelijk geheugen staat en wordt daarom in het privégeheugen van de organisatieprocessor geplaatst.

De funktie van de organisatieprocessor in deze opzet is hetzelfde als in mogelijkheid 1 (zie blz. 3.16) met dien verstande dat het volgende onderdeel er aan toegevoegd wordt:

- het "afbouwen" van de taakwerklijst en de sturing van de taakprocessoren.

Vanwege de aanwezigheid van een superorganisator en min of meer slaafse taakprocessoren spreekt men van een Master/Slave-systeem. De master geeft opdrachten aan de slaven. Er zijn twee gradaties van "slavernij" te onderkennen. In beide gevallen bepaalt de master <u>welk</u> programma uitgevoerd moet worden :

1. de organisatieprocessor kan een taakprocessor op elk moment dwingen een andere taak in uitvoering te nemen ("gedwongen taakprocessoren")
2. een taakprocessor bepaalt zelf het moment (gewoonlijk na het beeindigen van een programma) waarop aan de organisatieprocessor een nieuwe opdracht wordt gevraagd ("vrije taakprocessoren")

ad 1 Omdat de organisatieprocessor op elk willekeurig moment een nieuwe opdracht aan een taakprocessor kan verstrekken (doordat een taak met hoge prioriteit is aangevraagd) moeten de taakprocessoren in staat zijn het programma te redden. Hiervoor moeten de taakprocessoren over tenminste een deel van de jobhead beschikken. In dit deel van de jobhead worden na een onderbreking de gegevens gered (vervolgadres, tussenresultaten) en de status "onderbroken" ingevuld, zodat de taak op een later tijdstip, eventueel door een andere processor op de juiste wijze vervolgd kan worden.

De organisatietaak "het afbouwen van de taakwerklijst en de sturing van de taakprocessoren" wordt verricht door een routine,

waaraan de naam Dispatcher is gegeven. In de literatuur(*) wordt onder dispatching verstaan: de toewijzing van een processor aan de uitvoering van een taak.

De funktie van de Dispatcher bestaat uit de volgende onderdelen:
- het afzoeken van de taakwerklijst
- het vegen van de aanvraag na toewijzing

De Dispatcher bezit van elke taak een lijst met gegevens (joblist) aan de hand waarvan hij:
- kan beslissen of de taak mag draaien i.v.m. blokkering
- kan beslissen of de taak kan draaien (staat taak in geheugen?)
- de taakprocessor van de juiste gegevens (beginadres) kan voorzie

Tevens beschikt de Dispatcher over de bezettingstabel aan de hand waarvan hij:
- kan beslissen welke taakprocessor onderbroken moet worden, in het geval dat er een urgente taak wordt aangevraagd.
- de aanvraag van een onderbroken taak kan herstellen

De Dispatcher wordt in dit geval in twee situaties geaktiveerd, namelijk nadat - door middel van een interruptie een aanvraag in de taakwerklijst is geplaatst.
- een taakprocessor met een taak klaar is en dit meldt bij de organisatieprocessor.

De funktie en struktuur van dit besturingssysteem wordt in hoofdstuk 6 nader aan de orde gesteld.

ad 2 Een door een taakprocessor in behandeling genomen taak wordt nu niet onderbroken; anders gezegd: de prioriteit van een taak in uitvoering wordt oneindig hoog.

De Dispatcher wordt nu slechts geaktiveerd in de situatie dat een taakprocessor met een taak klaar is en dit meldt bij de organisatieprocessor. Deze behoeft nu niet bij elke nieuwe aanvraag te controleren of de taak direkt toegewezen moet worden. Er hoeft in dit geval niet meer gered en "ontred" te worden.

Het zij hier opgemerkt dat als een taak met hoge prioriteit wordt aangevraagd deze op uitvoering moet wachten totdat één van de aanwezige taakprocessoren een taak beëindigt. Dit hoeft in een multipprocessorsysteem echter niet altijd te betekenen dat er ook langer gewacht moet worden, dan in het geval een taakprocessor deze taak direkt opgedragen krijgt.

(*) "Multiprocessors and Parallel Processing" P.H. Enslow

In de tot nu toe beschreven opzet is steeds maar één processor met de organisatie belast (centraal systeem). Bij aanwezigheid van vele taakprocessoren is het mogelijk dat de organisatie te omvangrijk wordt, waardoor de organisatieprocessor de "bottle-neck" van het systeem wordt. In deze situatie is het zinvol een of meerdere systeemtaken naar een aparte processor te delegeren (bijvoorbeeld een Disk-module of een input/outputprogramma voor snelle periferie). Als de gedelegeerde systeemtaak betrekking heeft op de bediening van een randapparaat, verwerkt de processor alle interrupties van het betrokken randapparaat. Deze processor krijgt zijn opdrachten van de supervisor, maar volbrengt deze zelfstandig.

Omdat het mogelijk is dat de opdrachten van de supervisor sneller komen dan ze uitgevoerd kunnen worden, ontstaat in dit subsysteem een O-A-W struktuur (zie bladzijde 3.4). De opdrachten kunnen in de vorm van interrupties plus een code in het communicatie geheugen worden door gegeven (zie hoofdstuk 6 "het gemeenschappelijk communicatie geheugen"). In het geval dat de supervisor een of meer organisatie "slaven" heeft spreken we van een decentraal systeem.

In bovenbeschreven systemen zijn alle taakprocessoren in staat elk programma uit te voeren. Een andere mogelijkheid is dat elke processor een vaste taak of een vaste groep taken krijgt toegewezen. De applicatietaken zijn nu statisch aan een processor toegewezen. De organisatieprocessor stelt in dit geval voor elke taakprocessor een aparte werklijst op.

Deze mogelijkheid zou met vrucht kunnen worden toegepast in het geval het computersysteem meerdere onafhankelijke processen tegelijkertijd moet regelen. Elke taakprocessor krijgt een bepaald proces toegewezen, waardoor een minimum aan interferentie tussen de af te handelen programmas van de verschillende processen ontstaat.

Een algemeen aspekt is dat hettoekennen van prioriteiten aan programmas een erg lastig karwei is. Dit wordt nog moeilijker indien meerdere te regelen processen in het geding zijn. Wat is het criterium dat programma i van proces a urgenter is dan programma j van proces b? Door het regelen van één proces aan één processor toe te wijzen wordt het toekennen van prioriteiten voor een deel vergemakkelijkt.

In het geval van één taakwerklijst kan deze moeilijkheid vergemakkelijkt worden door per prioriteitsniveau meerdere taakaanvragen toe te laten. Programmas met dezelfde prioriteit (van verschil-

lende processen) worden dan bijvoorbeeld op basis van first-in-first-out afgehandeld. De Scheduler zal in dit geval uitgebreid moeten worde met een "Linking"-routine. Uiteraard moet ook de Dispatcher aangepast worden.

Voordelen van mogelijkheid 2

1. Het lock/unlock mechanisme vervalt, omdat alleen de organisatie-processor toegang tot de taakwerklijst heeft.
2. Systeemprogrammas hoeven niet reëntrant te zijn.

Nadelen

1. Bij statische toewijzing van de processoren bestaat de kans dat sommige processoren overbelast worden, terwijl andere niets te doen hebben.