Over paginaadministratie.

0. Enige terminologie.

Om spraakverwarring te voorkomen, voeren we in de begrippen "pagina" en "bladzijde". Een bladzijde is een hoeveelheid informatie van 512 woorden, een pagina is een stuk van een van de geheugens, bestemd om, indien gebruikt, een bladzijde te herbergen. Een pagina beslaat dus 512 woorden, en wel consecutief geadresseerde woorden. De bladzijde isdus de informatiehoeveelheid, die in een pagina "past".

In de programma's zijn het de bladzijden, die een duidelijke identiteit hebben. De text van het objectprogramma wordt in bladzijden onderverdeeld, bladzijden, wier bestaan begint in de loop van het compilatieproces en eindigt na voltooiing van het programma. Grote arrays zullen hun elementen eveneens in bladzijden ingedeeld krijgen, in principe beginnen deze bladzijden hun bestaan bij de uitvoering van de arraydeclaratie, ze eindigen hun bestaan bij de definitieve verlating van het blok in kwestie.

De stapel wordt enigszins anders behandeld; deze zal wel steeds bestaan uit een aantal bladzijden, maar om de elementen van deze bladzijden te addresseren zal een ander mechanisme geschapen worden dan het mechanisme, nodig om een element te selecteren uit een algemene bladzijde (programmatext of groot array). De elementen in de stapel zullen nl. tijdens hun levensduur slechts op een vaste plaats in de kernen kunnen staan, zodat we deze elementen gedurende hun bestaan kunnen karakteriseren door hun physisch kernenadres.

Elke bladzijde (althans de non-stapelbladzijden) ontleent zijn identiteit aan een hem toegevoegde grootheid, een zg. BZV (BladZijde Variabele). Doorgaans zijn de BZV's gesitueerd in de stapel van het programma, waarin deze bladzijden voorkomen. (Dit geldt ten duidelijkste voor de bladzijden, die de programmatext uitmaken; de bijbehorende BZV's kunnen we beschouwen als globale variabelen van dit programma. Dit geldt ook voor de BZV's voor array-bladzijden: deze zijn lokale grootheden van het blok, waarin het array gedeclareerd is.) Er komen in het samenspel ook andere bladzijden voor, bv. de bladzijden, die de bibliotheek van standaardprocedures, die permanent op de trommel aanwezig geacht wordt;deze kunnen kennelijk niet exclusief in de stapel van 1 programma ondergebracht worden. Immers, de bibliotheek is een gedeelte van het universum, waarin de verschillende programma's samen in ingebed worden, de bibliotheek overtreft , omvat, in levensduur ook de individuele programma's. Dit vindt zijn neerslag daarin, dat er naast wat de programma's prive nodig hebben, op een vaste plaats in het kerngeheugen een rijtje BZV's ondergebracht zal worden, de "bibliotheek BZV's", waartoe ieder individueel programma, bij gratie van het feit, dat ze permanent op een vaste plaats te vinden zullen zijn, via directe adressering toegang toe heeft. Een soortgelijk arrangement zal gemaakt worden om het mogelijk te maken bladzijden van het ene proces over te hevelen naar een ander proces; ik heb hier het oog op input-output-buffers.

Pagina's hebben we uit de aard der zaak in twee soorten, nl. "kernpagina's" en "Trommelpagina's". In beide media beslaan ze 512 opeenvolgende geheugenwoorden. De BZV's heten "variabelen" omdat ze onder andere vastleggen, in welke pagina(s) de bladzijde in kwestie te vinden is. Wij zullen straks uitgebreid en in detail beschrijven, welke informatie in de BZV's vastgelegd wordt en hoe. Bij die afspraak hebben twee dingen als leidraad gediend: het meest voorkomende geval moet zo snel mogelijk herkend kunnen worden (een tijdsoverweging dus) en de BZV moet zo min mogelijk ruimte in beslag nemen (een overweging van geheugenruimte dus). Dit laatste is belangrijk, omdat de BZV's -zoals wij later zullen zien- permanent in het kerngeheugen zullen staan en aan elke voorkomende bladzijde een BZV is toegevoegd.

Ook de pagina's worden door additionele "variabelen" nader beschreven. Voor de trommelpagina's -wat er heel veel zijn- is dit slechts een enkel bit, dat aangeeft of deze trommelpagina vrij, dan wel bezet is. Voor de kernpagina is deze sta-

tusinformatie aanmerkelijk omvangrijker, maar daar deert dit minder, omdat het maximale aantal kernpagina's zoveel kleiner is. De informatie, die de status van een kernpagina beschrijft heet een KIE (Kern Inhoud Element). Een KIE beslaat tot op heden vier woorden. (We merken en passant op, dat dit bijna een procent is van de geheugenruimte, die door de kernpagina zelf wordt ingenomen; bij veel kleinere kernpagina's zou de voor de KIE'sbenodigde geheugenruimte zich gaan ¥ wreken.) Wat een KIE precies allemaal aangeeft en hoe, zullen we eveneens in detail bespreken.

1. Classificatie van kernpagina's.

In het volgende zal een classificatie gegeven worden van de hoofdtoestanden, die we bij een kernpagina onderscheiden. Deze classificatie geeft aan (steeds), welke rol deze kernpagina op dit moment speelt. In de individuele programma's is deze rol vrij duidelijk, we moeten ons echter realiseren, dat een kernpagina ook betrokken kan zijn in een trommeltransportopdracht, of om de bladzijde, die hij herbergde op de trommel te dumpen, of om de inhoud van een trommelpagina over te nemen. Deze actie's worden niet zozeer door de individuele programma's als door de coordinator ondernomen. De tijdsduur van deze transporten is geenszins verwaarloosbaar en we moeten de status van een kernpagina "in een transport betrokken" dus expliciet invoeren. Hoeveel onderscheidingen we hierin willen aanbrengen, welke facetten tijdens zo'n transport van belang zijn, hangt van de coordinator af. Dat de hieronder te beschrijven indeling van kernpagina's zinvol is, is dus eigenlijk niet aan te tonen zonder ook in detail dat stuk van de coordinator te beschrijven, dat hier relevant is. We zullen dit toch proberen, dit rapport moge dan dienen als "underlying information" voor de beschrijving van de betrokken coordinatormechanismen. (Zoals beschreven zijn in de blokschema's van het document "bladzorg" d.d. 1 - 3 mei 1964.)

Ik ben mij bewust, dat de onder te geven indeling niet dwingend is; zij is de neerslag van weloverwogen (zo men pejoratief wil zijn "willekeurige") beslissingen. Het beste, waarnaar ik dus naar streven kan is waarschijnlijk maken, dat de genomen beslissingen de weg banen tot een hanteerbare organisatie!

Ter introductie tot het soort moeilijkheden moge de volgende "PTT-gelijkenis" dienen. De PTT heeft op zich genomen, om brieven te bezorgen, bezorgt deze brieven niet instantaan maar weet drommels goed, dat deze brieven een eindige reistijd hebben. Als alle geadresseerden het eeuwige leven hadden en nooit verhuisden, had de PTT geen probleem. Gegeven is echter het feit, dat geadresseerden slechts per adres slechts een eindige verblijfstijd doorbrengen. Het probleem van briefbezorging zou heel andere vormen aannemen, als de gemiddelde verblijfstijd op eenzelfde adres eens klein was ten opzichte van de gemiddelde reistijd van een brief! (Soms vraag ik me af, of de PTT zich wel bewust is, hoezeer ze in dezen door het oog van de naald gekropen zijn.)

Soortgelijke moeilijkheden hebben wij, doordat trommeltransporten niet instantaan plaats vinden, ook. Tijdens de periode, dat een bladzijde tussen langzaam en snel geheugen getransporteerd wordt, kunnen er allerlei dingen mee gebeuren. De bladzijde kan opbouden te bestaan. Het kan ook gebeuren, dat terwijl we bezig zijn om deze bladzijde op de trommel te dumpen, er hernieuwd interesse in ontstaat. Mww. een trommeltransport wordt ondernomen op grond van een of andere beslissing (dat een bladzijde naar het schijnt niet erg interessant is bv.) maar tijdens de uitvoering van een dergelijk transport kunnen zich omstandigheden voordoen, waardoor men deze beslissing eigenlijk wil herroepen, resp. aan het transport een andere betekenis wil geven. De eerste beslissing, die wij daarom genomen hebben is, dat wij voor de roltoekenning van een kernpagina, die in een transport betrokken gaat worden, geen beslissing zullen nemen, die verder in de toekomst reikt dan de voltooing van dit transport.

Het komt er op neer, dat we niet het volgende arrangement gevolgd hebben. Als een van de programma's een levendige belangstelling in een bladzijde, zeg bladzijde A , toont, die op dat moment zich niet in de kernen bevindt, dan zoekt men uit, welke bladzijde, die zich wel op de kernen bevindt, het oninteressantst lijkt. Dit zij bladzijde B. Men verwijdert nu bladzijde B uit de kernen (als hij niet ook nog op de trommel staat, impliceert dit een dumpend transport naar de trommel) en geeft vervolgens een transportopdracht om bladzijde A op de vrijgemaakte kerpagina over te nemen. Dit arrangement dwingt ons dus om ten aanzien van de kernpagina in kwestie twee transporten in de toekomst te kijken en dit hebben wij niet gekozen.

In plaats daarvan maken wij het systeem zo, dat er inpprincipe wel een vrije kernpagina is. Ontstaat er nu behoefte aan bladzijde A, dan wordt een transportopdracht naar deze vrije kernpagina gegeven; vervolgens wordt geanalyseerd of door deze bezetting het aantal vrije kernpagina's angstig laag geworden is, zo ja, dan wordt gekeken welke het oninteressantst is en deze wordt verwijderd uit de kernen; dit laatste heeft dus mogelijk een dumpend transport ten gevolge.

De redenen, dat wij de tweede methode gekozen hebben, zijn de volgende. Nogmaals, wij zijn ons bewust, dat ze niet dwingend zijn. In het eerste geval moet je eerst de minst interessante kiezen, zodoende ruimte maken voor wat je nodig hebt. Dit lijkt in twee opzichten ongunstig. Het keuzeproces, welke het minst interessant is, kon wel eens een beetje tijdrovend zijn en bovendien komt er (mogelijk) eerst het ruimtemakend transport voordat het transport van trommel naar kernen plaats kan vinden. Het programma, dat bladzijde A nodig had, wordt daardoor gegarandeerd vrij lang opgehouden. Dat is allemaal goed in multiprogrammering, als je veel andere programma's hebt, maar dat is natuurlijk niet iets om op te bouwen. In het gekozen arrangement kan het transport voor bladzijde A meteen gegeven worden, zodat het programma, dat bladzijde A nodig heeft, zo gauw mogelijk weer door kan gaan. Het is nu de vraag, of de machine nog een ander nuttig programma op de pennen heeft en dit is dus het geknipte moment om de "minstinteressante" er uit te kiezen en maar vast ruimte te maken.

Een ander gevolg is zuiver organisatorisch: in het startmagazijn voor de trommel zal op elk moment voor een bepaalde kernpagina hoogstens 1 startopdracht staan, in het transport is elke kernpagina hoogstens met 1 bladzijde betrokken. Dit betekent, dat we de informatie over wat op komst is, wat weggetransporteerd wordt etc. per kernpagina kunnen bijhouden, dwz. dat we de omvang van een KIE klein kunnen houden. Hoewel we alternatieven niet tot het eind toe hebben uitgewerkt (we hebben pogingen ondernomen, maar raakten, mogelijk door gebrek aan ervaring, vast), hebben we het gevoel, dat het hier voor te stellen systeem onder andere zijn eenvoud ontleend aan het feit, dat een kernpagina in de tijd, dat hij vrijgemaakt zit te worden, niet al vast een andere bestemming gekregen heeft.

Nu gaan we de classificatie van de kernpagina's geven. Een kernpagina kan zich in eerste instantie in twee elkaar uitsluitende toestanden bevinden, nl. wisselend en niet-wisselend. Een kernpagina is wisselend, wanneer voor deze kernpagina een trommeltransportopdracht gegeven is, van de voltooiing waarvan door de X8 nog geen kennis is genomen; anders is de kernpagina niet-wisselend.

Zolang een kernpagina wisselend is, zal geen X8-programma zijn kernen aanraken, noch voor deze kernpagina vast andere startopdrachten geven. Het aanbod van een startopdracht impliceert dus (ongeacht richting van het transport) altijd de overgang van niet-wisselend naar wisselend; de kennisname van de voltooing van het transport impliceert dankzij deze zelfde conventie dus altijd de inverse overgang.

(Wij zijn ons er van bewust, dat de regel, dat geen programma de kernen van een wisselende pagina zal selecteren, nodeloos stringent is. Als dit transport een transport van trommel naar kernen is, zou deze selectie onder bijkans alle omstandigheden zinloos zijn; in het inverse geval valt er nog over te denken: als we uit de kernpagina willen lezen, dan kan dit nog, willen we er in schrijven, dan heeft dit alleen maar ten ge-

volge, dat de copie, die op de trommel komt, tijdens dat proces al obsoleet an het worden is. Logisch zou dit wel kunnen, we hebben het echter niet toegestaan.

De eerste reden is, dat hierdoor bij inspectie van de toestand van een kernpagina de transportrichting plotseling interessant zou worden, terwijl, zoals we zullen zien, die verder nauwelijks een rol speelt. (Na afloop van een transport, tijdens de uitvoering waarvan we niet geinterfereerd hebben met de kernpagina, kunnen we altijd de kernpagina beschouwen als een copie van de trommelpagina, ongeacht in welke richting het transport heeft plaatsgevonden.) De tweede reden is een kwestie van voorzichtigheid: stel, dat het transport een pariteitfout detecteert! Om er dan nog uit te komen, als het programma de inhoud van de kernpagina, zoals die bedoeld was, inmiddels heeft mogen veranderen, lijkt een nodeloze verzwaring van deze toch al hachelijke opgave. Wisselende kernpagina's blijven dus op alle manieren onaangeroerd.)

Wisselende kernpagina's worden onderscheiden in PRAEVRIJE en PRAECOPIEEN.

Een wisselende kernpagina is PREAVRIJ, wanneer er -met inachtname van de momentane kennis van de getoonde interesse- na afloop van het transport geen interesse zal zijn in de inhoud van de kernen. Dit is dus ten duidelijkste het geval, wanneer de bladzijde, die in het transport betrokken is, tijdens dit transport zal ophouden te bestaan. Ook las een origineel gedumpt wordt om een vrije kernpagina te maken (zie onder), dan zullen we deze kernpagina als PRAEVRIJ beschouwen.

Een wisselende kernpagina is een PRAECOPIE, zodra is vastgesteld, dat er na voltooiing van het transport interesse in de kerneninhoud geacht wordt te ontstaan. Dit is ten duidelijkste het geval bij het aanhalen van een trommelpagina, die een bladzijde bevat, waarmee een van de programma's verder moet werken.

Een niet-wisselende kernpagina is in eerste aanleg een VRIJE, een COPIE of een ORIGINEEL.

Een niet-wisselende kernpagina is een VRIJE, wanneer zijn KIE niet met een BZV geassocieerd is: de inhoud van deze kernen is volledig "afgeschreven", behalve dat we wel correcte pariteit zullen eisen.(Maar dit is alleen een verplichting voor het tandenpoetsen.)

Een kernpagina is een ~~COPIE~~ COPIE of een ORIGINEEL, wanneer hij een bladzijde bevat; zijn KIE is dan geassocieerd met de BZV van deze bladzijde. Het verschil tussen een COPIE en een ORIGINEEL is, dat in het geval van de COPIE tevens een trommelpagina geassocieerd is, waar deze zelfde bladzijde te vinden is. Een origineel bestaat slechts enkel op de kernen. Kernpagina's, die programmatext bevatten zullen dus meestal COPIEEN zijn (nodig is dit niet: ze kunnen door de compiler in het kerngeheugen opgebouwd zijn en als er altijd voldoende ruimte in het kerngeheugen gebleven is, dan zijn ze nooit gedumpt en dus origineel gebleven).

Er zijn tussen deze vijf toestanden (hoofdtoestanden, we zullen ze later nog wat differentieren) 10 overgangen mogelijk.

1) VRIJE → PRAECOPIE.
Dit vindt plaats, als een vrije kernpagina gekozen is om een bladzijde over te nemen, die alleen op dat moment op de trommel staat. Dit is een heel gewone overgang.

2) PRAECOPIE - COPIE.
Dit vindt plaats als onderdeel van de kennisname van de voltooiing van een trommeltransport, waarna mogelijk interesse in de getransporteerde bladzijde ontstaat. Dit is de normale overgang, volgens welke de toestand PRAECOPIE verlaten wordt. Wij vestigen er nogmaals de aandacht op, dat deze overgang plaats kan vinden na afloop van een schrijfoodracht naar de trommel (zie overgang 10.)

3) COPIE → ORIGINEEL.

Deze overgang vindt plaats, zodra een programma schrijft op een bladzijde, die in een kernpagina geborgen werd, die als COPIE te boek stond; een neveneffect hiervan is, dat de trommelpagina, die in de KIE van de kernpagina vermeld stond, wordt vrijgegeven. Tevens wordt de verwijzing naar deze trommelpagina, ter voorkomig van verwarring, verwijderd. Dit is een normale overgang voor bladzijden van een groot array, met programmabladzijden zal dit niet gebeuren.

4) ORIGINEEL → PRAEVRIJE.

Deze overgang vindt plaats, als de keuze van de coordinator om ruimte te scheppen, gevallen is op een "oninteressant" ORIGINEEL. Er wordt, omdat in de kernen een ORIGINEEL staat, voor deze bladzijde een vrije trommelpagina gekozen en er wordt een (dumpende) startopdracht gegeven. Deze overgang (evenals overgang 1) vindt dus altijd plaats op het moment, dat een nieuwe startopdracht aan de trommel aangeboden wordt. Ook deze overgang moet normaal geacht worden.

5) PRAEVRIJE → VRIJE.

Deze overgang vindt plaats als onderdeel van de XXX kennisname van de voltooiing van een trommeltransport. Als de KIE van deze kernpagina nog aan een BZV gekoppeld was (en als aan het einde van het transport de bladzijde nog bestond, dan zal dit het geval zijn) dan wordt op dit moment de koppeling losgemaakt. Op dit moment is deze kerninhoud niet meer via deze BZV te bereiken.

6) COPIE → VRIJE.

Deze overgang vindt plaats als de keuze van de coordinator om ruimte te scheppen gevallen is op een "oninteressante" COPIE; de overgang vindt tevens plaats, als een bladzijde ophoudt te bestaan (bv. een bladzijde van een groot array bij verlating van het blok), die als COPIE in de kernen aanwezig is, een geval, waarin tevens een trommelpagina wordt vrijgegeven.

7) ORIGINEEL → VRIJE.

Deze overgang vindt plaats als een bladzijde, die als origineel in de kernen aanwezig is, ophoudt te bestaan, bv. een arraybladzijde bij blokverlating. Dit zal ook gebeuren bij inkrimping van een programmastapel, omdat, zoals we later zullen zien, de stapelbladzijden altijd als origineel in de kernen zullen staan.

8) VRIJE → ORIGINEEL.

Deze overgang vindt plaats bij stapelgroei en eveneens bij eerste referentie naar een van de elementen van een nieuw geintroduceerde arraybladzijde. Bij de array-declaratie wordt voor een groot array nog geen bladzijde geintroduceerd, alleen maar een stel "potentiele" bladzijden, dwz. een stel BZV's, die geinitialiseerd worden met de speciale indicatie "nuchter". Pas bij de eerste referentie naar een dergelijke bladzijde wordt er aan deze BZV een feitelijke (dwz. niet-lege) bladzijde toegevoegd. Deze introductie vindt uit de aard der zaak plaats in de kernen en geeft dus aanleiding tot een ORIGINEEL.

9) PRAECOPIE → PRAEVRIJE.

Deze overgang zal, zo hij voorkomt, exceptioneel zijn. Het betekent, dat we, van de veronderstelling, dat een bladzijde interessant was, afstappen en beslissen, dat we deze bladzijde toch maar niet in de kernen willen hebben. Dit zou zich voor kunnen doen, doordat het programma, dat deze bladzijde heeft aangevraagd, in die tussentijd ten gevolge van een foutmelding beeindigd wordt. Het meest waarschijnlijke geval, dat dit optreedt, is ten gevolge van de hint (zie onder); hier kan een programma van te voren vast aankondigen, dat het hoogstwaarschijnlijk straks interesse gaat tonen in een zekere bladzijde. Nodig is die interesse niet, de hint kan een arraybladzijde betreffen, die tijdens transport opboudt te bestaan.

10) PRAEVRIJE → PRAECOPIE

Deze overgang is weer veel waarschijnlijker dan de vorige. Hij treedt op, wanneer

interesse getoond wordt in een bladzijde, die op dat moment net in een transport betrokken was, waaraan op dat moment de bedoeling gehecht was, dat het een VRIJE kernpagina zou opleveren; in alle waarschijnlijkheid was dit dus een dumpend transport voor een arraybladzijde.

2. Onderverdeling van de hoofdclassificaties voor kernpagina's.

De VRIJE kernpagina's zijn het gemakkelijkste, omdat daarover het minste te vertellen valt. De belangrijkste eis is, dat we, op het moment, dat we een vrije kernpagina nodig hebben en er inderdaad eentje is, zo snel mogelijk ontdekken, welke kernpagina we daarvoor kunnen nemen. Voor dit doel zijn de VRIJEN in een ketting geregen, het schakelelement waarvan in de KIE van de kernpagina is opgenomen. Men kan voor dit doel volstaan met een enkelvoudige ketting, omdat je de VRIJEN -ze zijn je allemaal even lief- gebruiken kunt volgens het principe van een stapel. Een nieuwe vrije schakel je voor aan de ketting erbij, als je een vrije nodig hebt, neem je de eerste van de ketting.Het is de plicht van de tandepoetserij om aan het begin alle kernpagina's, die dan vrij zijn, in deze ketting op te nemen.

Voor de wisselende kernpagina's moeten we onderscheiden, of het transport betrekking heeft op een nog bestaande bladzijde of niet. Bij de PRAECOPIEEN is dit in elk geval zo, voor de PRAEVRIJEN hoeft de bladzijde echter niet meer te bestaan. In geval van wisselende kernpagina ten bate van een bestaande bladzijde zal tijdens het transport de koppeling van BZV en KIE bestaan. Wie vraagt naar een bladzijde, die in wisseling betrokken is, wordt via de BZV van die bladzijde (deze BZV is uniek voor die bladzijde en de enige manier om contact met die bladzijde op te nemen) naar de KIE verwezen worden, daar echter zal genoteerd staan, dat de kernpagina wisselend is. Tot deze "vroegtijdige" koppeling worden we gedwongen door het bestaan van de bibliotheek: het is immers heel goed mogelijk, dat een programma niet verder kan, vanwege de afwezigheid van een bibliotheekbladzijde; in de wachttijd van dit programma kan heel wel een ander programma, dat wel door kan gaan, ontdekken, dat het deze zelfde bibliotheekbladzijde nodig heeft. Zijn het de PRAEVRIJEEN, waarbij we het onderscheid moeten aangeven of de bladzijde nog bestaat, bij de PRAECOPIEEN kunnen we reeds betekenis toekennen aan de zg. heiligheidsteller. (Zie onder.)

De COPIEEN en de ORIGINELEN worden onderverdeeld in "profanen" en "heiligen". Een heilige kernpagina is een kernpagina, waaraan de coordinator niet op eigen gezag een andere rol mag toebedelen. Het is nl. wel mogelijk, de programma's zo te schrijven, dat een bladzijde niet noodzakelijkerwijze ergens in de kernen is, het is echter niet toegestaan, dat een bladzijde, waarvan de aanwezigheid door een programma geconstateerd is, op elk willekeurig microscopisch moment verdwijnt. Allicht niet: als een programma een element van een groot array wil selecteren en heeft vastgesteld, op welk kernadres dit element zich bevindt, dan mag xxxxxxxxxxxxxx tussen deze vaststelling en de feitelijke selectie dit element niet uit de kernen verdwijnen. Wel, heel vaak zal dit gegarandeerd worden, door het stuk van verificatie van de aanwezigheid en de feitelijke selectie in doofheid te doen uitvoeren. Als de X8 aan zo'n stukje begonnen is, dan wordt dat dus eerst afgemaakt, voordat mogelijk de coordinator over de rol van deze kernpagina anders beslist. (In multiprogrammering moet je zoals bekend voorzichtig zijn met doofheid: in de individuele programma's zal de X8 heel vaak eventjes doof zijn.)

Het mechanisme van de tijdelijke doofheid is echter niet machtig genoeg: de efficiency van de individuele programma's wordt aanzienlijk verhoogd, indien zij de mogelijkheid hebben om ten aanzien van een beperkt aantal pagina's over een beperkt verloop van tijd er op te mogen rekenen, dat deze bladzijden niet onder hun vingers worden weggegrist. Aan elk programma is de plicht, zo geconstrueerd te zijn, dat het op geen moment te veel kernpagina's simultaan heiligt, want anders zouden ze het kernegeheugen blokkeren. (Hiermee hebben we bij de behandeling van destapel een beetje de hand gelicht, zie later.)

xxxx

Heiliging van een kernpagina is iets, dat geschiedt op instigatie van een van de individuele programma's; dit programma laadt daarmee de plicht op zich om deze heiliging mettertijd weer ongedaan te maken, deze pagina weer te "profaneren." In het bijzonder zal een programma, dat een niet in de kernen aanwezige bladzijde voor zijn voortzetting nodig heeft, vragen om deze bladzijde in een geheiligde kernpagina. Met deze heiliging is gekoppeld, dat dit programma straks via een P-operatie op deze bladzijde kan gaan staan wachten. Hiermee voorkomen we de volgende situatie: een programma heeft een bladzijde, die enkel op de trommel staat, beslist nodig, er wordt een trommeltransport voor dit programma gestart en het programma komt op de wachtlijst der geblokkeerde programma's. Na voltooiing van dit transport wordt het wachtende programma overgeheveld naar de uitvoerbaren, maar het komt niet direct aan bod. Als we nu de pagina in kwestie niet geheiligd hadden, dan zou de mogelijkheid bestaan, dat dit programma, tegen de tijd, dat het wel aan bod komt, tot de ontdekking moet komen, dat de bladzijde, waarop het had staan wachten, inmiddels weer verdwenen is. Dit nu hebben we met behulp van de heiligheid uitgesloten: deze bladzijde blijft onaantastbaar in het kerngeheugen staan, totdat het programma, dat om deze bladzijde gevraagd heeft, de inhoud van deze bladzijde inderdaad heeft kunnen selecteren.

De bibliotheek schept hier een kleine complicatie. Uit het voorafgaande zou men kunnen concluderen, dat het voldoende is, om elke kernpagina een heiligheidsbit te geven, maar terwijl het ene programma staat te wachten op de aankomst van een bibliotheekbladzijde, kan een ander programma doorgaan en ook op deze bladzijde komen te wachten. En niet alleen in deze toestand. We zullen in elk programma de bladzijde, waaruit op dat moment de opdrachten gehoorzaamg worden, heiligen, dwz. heiligen bij binnenkomst en profaneren bij verlating. Twee verschillende programma's kunnen dus beide dezelfde bladzijde willen heiligen. Om die reden bevat de KIE niet een heiligheidsbit, maar een heiligheidsteller; heiliging betekent verhoging met 1, profanatie betekent aflaging met 1 en een kernpagina geldt als profaan, wanneer zijn heiligheidsteller = 0 is. (Naast de individuele programma's zullen wij in het "Hotel voor abstracte machines" ook stamgasten hebben ter verzorging van input-output bv.; dankzij de heiligheidsteller is het nu ook mogelijk, meer dan 1 kleine stamgast in de zelfde programmapagina te definieren, c.q. een stamgast parallel te activeren.)

Het heilig aanvragen van een bladzijde legt een onontkoombare verplichting op aan de coordinator (nl. om vroeg of laat met die bladzijde voor de draad te komen), het programma, op welks instigatie dit gebeurd is, heeft hiermee de verplichting op zich genomen, om vroeg of laat deze pagina weer te profaneren. Dit is het ene mechanisme, dit lijkt ons absoluut noodzakelijk. Daarnaast willen we in dit stadium niet de mogelijkheid uitsluiten, dat een (waarschijnlijk handgemaakt) programma de coordinator een zg. "hint" geeft, behelsend "Straks ben ik waarschijnlijk in die en die bladzijde geinteresseerd". Dit is het soort opmerkingen, dat de coordinator als niet opportuun naast zich neer kan leggen; het programma, dat de hint gegeven heeft, zal voor de feitelijke selectie toch de aanwezigheidstest weer uitvoeren. Als de coordinator de hint krijgt en de bladzijde is aanwezig, dan kan hij het interessegetal van deze kernpagina verhogen, daarmee de kans verkleinend, dat deze bladzijde er in de naaste toekomst uitgekeild wordt, als de bladzijde er niet is, dan mag een trommeltransport gegeven worden (als er een geschikte vrije is, als er ruimte in het startmagazijn is etc.) Deze verwerking van de hint gaat niet met heiliging gepaard. Hier zien we dus, dat ook bij de PRAECOPIEEN de heiligheidsteller al een betekenis gegeven kan worden, nl. de waarde, die de heiligheidsteller van deze kernpagina zal moeten gegeven worden, als de PRAECOPIE in COPIE overgaat. In de volgende sectie zullen we zien, dat dit gegeven op de keeper beschouwd, overbodig zal zijn.

3. Terugmelding van aankomst van bladzijden.

Als een programma vraagt om een heilige kernpagina, dan betekent dit vanuit de kant van het programma, dat het via de voltooiing van een P-operatie van de aanwezigheid van deze bladzijde in kennis gesteld wil worden, het betekent voor het sys-

teem, dat de aankomst van deze bladzijde in een kernpagina door de ophoging van een seinpaal bezegeld moet worden. De eerste vraag is, waar we deze seinpaal onderbrengen. Een eerste suggestie is, om deze seinpaal onder te brengen bij de startopdracht; we zouden dan vier van deze trommelseinpalen hebben, voor elke mogelijke startopdracht in het magazijn eentje. Bij aanbod van een nieuwe startopdracht zouden we deze seinpaal = 0 moeten zetten, in de heiligheidsteller van de PRAECOPIE houden we bij, hoeveel programma's echt op deze bladzijde staan te wachten en als reactie op de ingreep voeren we de V-operatie op deze seinpaal uit met de dan heersende waarde van de heiligheidsteller als increment. Een programma, dat op een bladzijde moet wachten, doet dit, omdat de bladzijde al wisselend is of omdat er een startopdracht voor die bladzijde gegeven moet worden, in beide gevallen is bekend, welke van de vier startopdrachten voor het transport zorgt en het programma weet op dat moment, op welke seinpaal het straks moet gaan wachten. Dit arrangement is echter uiterst link. Dit werkt nl. goed, wanneer het aanvragende programma al op deze seinpaal staat te wachten, tegen de tijd, dat de bladzijde arriveert; in dat geval bewerkstelligt de V-operatie nl. dat het programma van de lijst der geblokkeerden naar die van de uitvoerbaren wordt overgeheveld. Dit gaat echter scheef, wanneer op het moment van aankomst van de bladzijde het programma nog niet zover is, nog niet aan de P-operatie toe is. Tegen de tijd, dat dat nl. het geval is, kan diezelfde plaats in het startmagazijn (en daarmee diezelfde seinpaal) allang voor een ander transport gebruikt zijn.

Een weg uit deze moeilijkheden is, om de seinpaal niet onder te brengen bij de startopdracht maar bij de KIE van de kernpagina in kwestie. Dit is dan wel safe, maar heeft het nadeel, dat dit wel de nodige geheugenplaatsen zal gaan kosten. Een geprogrammeerde seinpaal kost nl. drie geheugenplaatsen en dit zou dus ongeveer een verdubbeling van de pakweg vijftih KIE's betekenen; dit si dee te triester, omdat er normaliter niet meer dan vier van deze seinpalen gebruikt worden.

Wij hebben besloten kool en geit te sparen en de seinpalen, die deze blokkade beheersen, bij de individuele programma's onder te brengen (er op gokkend, dat we aanmerkelijk minder programma's in actie zullen hebben dan kernpagina's). Ieder programma heeft dus zijn eigen seinpaal. Het aanvragen van een heilige kernpagina betekent nu (het = 0 zetten van de eigen seinpaal, als daar het tandenpoetsen niet al voor gezorgd heeft) en het melden aan de trommelorganisatie, via welke seinpaal dit keer de voltooiing van het transport teruggeseind moet worden. Bij de kennisname van de voltooiing van het transport moet nu de "trommelingreepreactor" niet een vaste seinpaal met mogelijk meer dan 1 ophogen, maar een lijstje van seinpalen (mogelijk leeg, mogelijk langer dan 1) moeten allemaal met 1 opgehoogd worden. Daartoe bevatten de eigen seinpalen een schakelelement en bij de startopdracht bevindt zich de beginschakel. En nu zien we, dat de heiligheidsteller bij de PRAECOPIE eigenlijk overbodig is; deze is nl. gelijk aan de lengte van de seinpalenketting, die hangt aan de betrokken startopdracht.

Een wezenlijke veronderstelling,die aan dit laatste arrangement ten grondslag ligt, is, dat per programma de aanvraag van een heilige pagina en de P-operatie op de eigenseinpaal in de tijd elkaar afwisselen: als je immers eerst de ene bladzijde heilig aanvraagt en dan de ander, en vervolgens 1 P-operatie doet, dan weet je niet meer, welke bladzijde nu gearriveerd is. (Dit staat ons tevens toe, de eigen seinpaal slechts in 1 enkel ketting opneembaar te kunnen maken. Mocht, wat ik niet veronderstel, dit "ongenest" gebruik van de eigen seinpaal tot ongewenste restricties leiden, dan kunnen we altijd nog ober gaan tot een seinpaal per kernpagina.

4.Coderingen.

Om der wille van de volledigheid zal ik nu beschrijven, welke afspraken wij gemaakt hebben over de wijze, waarop de BZV's en de KIE's de informatie, die ze moeten herbergen, vasthouden. Dit is helemaal niet spannend, de genomen beslissingen zijn soms volslagen willekeurig en als ze dat niet zijn, zijn ze ingegeven door het reinste opportunisme.

4.1. De structuur van de BZV.

Een BZV beslaat 1 woord (met zorg niet meer, omdat we er voor elke bestaande bladzijde eentje in het kerngeheugen moeten invoeren). Een BZV kent vier toestanden.

1) BZVi = -0

Dit zal betekenen "De bladzijde in kwestie is nuchter".

2) BZVi = | d[26] = 1
| d[25] = 0
| d[24]...d[0] = beginadres trommelpagina, links aangevuld met nullen

Dit zal betekenen, dat de bladzijde in kwestie niet nuchter is, te vinden is op de aangegeven plaats op de trommel en nergens anders; de bladzijde is tevens niet in een transport betrokken.

3) BZVi = :BKPk (dwz. Basis Kern Pagina nr. k), links aangevuld met nullen.

Dit zal betekenen, dat de bladzijde niet nuchter is, niet in een transport betrokken is, wel op de kernen aanwezig is en wel als

BKPk[0] t/m BKPk[511]

Opm.1.:BPKk zal een 512-voud zijn.
Opm.2. Het basisadres van de bijbehorende KIEk wordt gevonden door

:KIEk = :BPKk / 128 + KPC

(KPC = KIEtabel Positionerings Constante); een KIE beslaat dus vier woorden.
Opm.3. Toestand 3 zal gekarakteriseerd zijn door BZVi groter dan 0.

4) BZVi = -(:BKPk, links aangevuld met nullen)

Dit zal betekenen, dat de bladzijde in een transport betrokken is met betrekking tot kernpagina k.

Opm. Deze toestand onderscheidt zich van 1) doordat BZVi ≠ 0 is en van 2) doordat d[25] = 1 is.

Het gevolg van deze conventie's is, dat de vaste aanwezigheid van een bladzijde volledig uit de BZV volgt, zonder raadpleging van de KIE; bij het schrijven op zo'n bladzijde moet wel de KIE geraadpleegd worden om te kijken , of dit schrijven wellicht de overgang van COPIE tot ORIGINEEL ten gevolge heeft. Wij kunnen overwegen, om dit bij toestand 3 in bit d[25] weer te geven; ik voel hier op het ogenblik eigenlijk weinig voor, selectie van een arrayelement is anyhow een aanzienlijk rpoces en raadplegen van de KIE is nu ook weer niet onoverkomelijk (het testen van zo'n hoger bit kost ook wel wat).

4.2. De structuur van de KIE.

KIEk[0] = indien gekoppeld aan een BZV, dan = :BZVi, links aangevuld met nullen
anders = - 0

KIEk[1] = indien trommelverwijzing van toepassing (impliceert onvrij)
d[26] = 1,

d[25] = 0

d[24]...d[0] = beginadres trommelpagina, links aangevuld met nullen

anders (dwz. trommelverwijzing niet van toepassing)

indien onvrij, dan = - 0

anders (dwz. indien VRIJ)

indien laatste pagina uit vrije ketting, dan = + 0

anders = :KIEj ($\neq$ 0), verwijzend naar de KIE van de volgende pagina uit de vrije ketting

KIEk[2] = als COPIE of ORIGINEEL, dan HHTk (< 128)
(= + 0, 1, 2, 3, ...) HHT betekent "Heiligheidsteller".
als PRAECOPIE, dan 128 + HHTk + 1024 * startopdrachtnummer
als PRAEVRIJ, dan 256 + 1024 * startopdrachtnummer
als VRIJ, dan 512.

KIEk[3] = strategische informatie (ook wel interessegetal genoemd).

4.3. Opmerkingen.

Tijdens transport van een bestaande pagina zijn KIEk en BZVi dus aan elkaar gekoppeld, ook als de kernpagina PRAEVRIJ is. Tijdens transport van een niet meer bestaande bladzijde is de kernpagina uit de aard der zaak PRAEVRIJ, KIEk[0] = - 0 en ook KIEk[1] = - 0. Het vermoorden van een bladzijde, die in een transport betrokken is, heeft nl. toch onmiddellijke vrijgave van de trommelpagina tenGevolge. Dit kan, omdat de trommelorganisatie de transportopdrachten verwerkt in de volgorde van aanbod, elke latere bestemming van deze trommelpagina wordt dus pas effectief, als de eerste transporteerderij volledig afgelopen is.

Wij hebben de kernpagina's niet geinterspatieerd met de bijbehorende KIE's, hoewel dit ons de schuifopdracht voor de herleiding van KIEk tot BKPk of omgekeerd zou besparen. De redenen waren:

a) mocht ooit een systeem van memory protection zijn intrede doen, dan is het waarschijnlijk, dat een techniek, die kernpagina's op hoge tweemachten laat beginnen, meer aansluit bij wat gebruikelijk wordt
b) het zou voor testdoeleinden (misschien bedrijven wij aanvankelijk nog wel inspectie) wel eens makkelijk kunnen zijn
c) de herleiding van physisch adres tot kernpagina en regelnummer in deze pagina wordt aanmerkelijk vergemakkelijkt.

Van het feit, dat de schone schuifopdrachten in een enkel register snel zullen zijn, zullen wij dus dankbaar gebruik maken.