De architectuur van de machines van de derde generatie, die geconcipieerd is in het begin van de zestiger jaren, is heel bewust toegespitst -met behulp van simulatie van bestaande, representatief geachte programmapakketten, met stochastische modellen en statistische criteria als bv. de "Gibson mix"- op de belasting van de machine, zoals men zich die toen voorstelde. (Als altijd is daarbij ongetwijfeld aanwezige toekomstvisie gecompenseerd door het feit, dat alle statistische materiaal uit het verleden stamde.) Om deze architectuur te begrijpen moeten we daarom trachten het beeld van de machine-belasting, dat toen prevaleerde, in grote trekken te reconstrueren.

Het belangrijkste kenmerk van die periode is in terugblik, dat de zogenaamde "hogere programmeertalen" in het hele veld van automatische informatieverwerking pas aarzelend burgerrecht verkregen. FORTRAN begon toen voor technisch-wetenschappelijke berekeningen redelijk erkenning te krijgen (hoewel vele professionelen het toen nog beschouwden als een tweederangs hulpmiddel voor hen "die niet echt konden programmeren), ALGOL 60 was slechts een hoofdzakelijk Europese, academische hobby en in de administratieve sector, waar toen al de grootste markt gezien werd, was van erkenning van COBOL nog nauwelijks sprake en was "assembly code" nog veruit het voornaamste programmeergereedschap. Zoals wij in het volgende zullen zien, heeft de toenmalig beperkte rol van hogere programmeertalen op allerlei aspecten een diepgaande invloed gehad.

Daarnaast is de architectuur van de machines van de derde generatie in tweeerlei opzicht een reactie op ervaringen opgedaan met machines van de tweede generatie. Dat waren machines met random access geheugens van zeer beperkte omvang, aangevuld met magnetische banden als massa-geheugen, configuraties die door de beperkte toegankelijkheid van de informatie op de magnetische banden steeds moeilijker hanteerbaar werden. Enerzijds is de derde generatie hierop een duidelijke reactie: de XXXXXXXXXXX maximaal toelaatbare omvang van kerngeheugens is drastisch vergroot en de magnetische schijven hebben als massa-geheugens hun intrede gedaan. (Dit waren schijven met bewegende arm: hoewel "randam" toegankelijkheid hierbij nog een stuk lager ligt dan bij trommels of dan bij schijven met vaste koppen, is het vergeleken bij magnetische banden alleszinds fair deze als "random access" geheugens te beschouwen.) Anderzijds zijn de machines van de derde generatie duidelijk beinvloed door het programma-aanbod, zoals zich dit bij de tweede generatie heeft ontwikkeld, in het bijzonder met betrekking tot sorteren. Wij doelen

hier op de uitgebreide kanaalfuncties .zoals "scatter read/write", die ten duidelijkste geintroduceerd zijn om sorteerwerk op magnetische banden te versnellen, er wellicht aan voorbijgaand, dat de behoefte aan sorteren met de vervanging van banden door schijven aanmerkelijk is afgenomen. (De barokheid van deze kanaalfuncties lijkt echter mede-verantwoordelijk voor de complexiteit, die de operating systems voor de machines van de derde generatie heeft aangenomen.)

Wij keren terug tot de directe gevolgen, die de toen nog onderschikte positie van hogere programmeertalen gehad heeft.

1) Het vermogen nieuwe programma's te creeeren is hierdoor sterk onderschat: men heeft zich de belasting voorgesteld als opgebouwd uit minder verschillende programma's die ieder voor zich langer zouden lopen.

2) In het licht van deze verwachting heeft men gekozen voor een wijze van programmaverwerking, waarin meer geoptimaliseerd is op maximale "throughput" dan op minimale "turn around time".

3) In het licht van deze verwachting is in operating systems veel aandacht geschinken aan "program scheduling": als het om weinig, grote programma's gaat is de tijd besteed aan program scheduling (vanwege het gering aantal) beperkt in omvang en (vanwege de grootte van de individuele programma's) hopelijk welbesteed. (Daar kwam bij, dat program scheduling, dat bij de tweede generatie vanwege de behandeling van magnetische bandeenheden zoveel had opgeleverd, een nog onverwelkte aureool had.)

4) In het licht van deze verwachting is de neiging ontstaan grote "initiele kosten" bij programmaverwerking wat makkelijk door procentuele verkorting van de eigenlijke rekentijd te rechtvaardigen.

5) Een microscopisch gevolg van de bereidheid bij programmaverwerking grote initiele kosten op te brengen ten bate van de snellere verwerking is te vinden in de structuur van de rekenorganen, die om der snelheids wil met een groot aantal, in de programmatekst expliciet benoemde rekenregisters zijn uitgerust. Dit echter vraagt om grotere initiele kosten bij het vertaalproces, dat nu de additionele plicht van "register allocation" heeft, wat de vertaaltijd vergroot.

6) Een tweede microscopisch gevolg van de bereidheid bij programmaverwerking grote initiele kosten op te brengen ten bate van snellere verwerking is te vinden in de betrekkelijk directe adressering van het geheugen, die geleid heeft tot de noodzaak van "linkage editing". Dit probleem ontstaat doordat procedures (hetzij eigen, voorvertaalde procedures, hetzij procedures uit de

bibliotheek) in objectcode geformuleerd diene te worden op een wijze, die bepaald wordt door het programma waarvan ze dit keer deel uit maken. Een en ander is noch des te ernstiger, omdat men niet aan de indruk kan ontkomen, dat de kosten die aan "linkage editing" verbonden zijn, aanvankelijk ernstig onderschat lijken.

7) De genoemde microscopische aspecten -groot aantal rekenregisters en betrekkelijk directe adressering van het geheugen- hebben niet alleen de macroscopische gevolgen ten aanzien van de XXXXXXX initiele (tijd)kosten gehad. Door het grote aantal expliciet benoemde rekenregisters is het mechanisme van de zg. "gesloten subroutine" belast met een zodanige "overhead" (ontstaan door red- en herstelplichten van registerinhouden) dat de gesloten subroutine in discrediet is geraakt ten gunste van de zg. "macro facility", die de neigin heeft de lengte van de programmatekst in object code te verlengen. Daarnaast heeft ieder programma van elke bibliotheekprocedure die het gebruikt een eigen, door linkage editing aan hem aangepaste, versie. Ook dit heeft geleid tot veeleisendheid ten aanzien van geheugengebruik. Ten gevolge van beide factoren moeten machines van de derde generatie, willen ze enigszins redelijk functioneren, met grote kerngeheugens uitgerust worden (in elk geval aanmerkelijke groter dan aanvankelijk voorzien; naar verluid is deze ontwikkeling de fabrikanten niet onwelgevallig geweest).

Tekenend is tenslotte, dat vertalers en operating systems as regel niet in een hogere programmeertaal geschreven zijn maar in assembly code, een keuze die onveranderlijk door de fabrikant gemotiveerd is met de uitspraak, dat anders deze "harde software" onvoldoende efficient zou zijn!

De aard van machinegebruik is in de afgelopen tien jaar echter geenszinds constant gebleven. De belangrijkste ontwikkelingen zijn de volgende.

1) Hogere programmeertalen hebben, zelfs bij machines die voor hun verwerking maar matig geschikt zijn, burgerrecht verkregen. Er zijn allerlei installaties waarvoor veel geprogrammeerd wordt, maar geen programma meer in assembly code.

2) Tengevolge hiervan heeft het programmaaanbod zich verschoven naar meer (soms ook kleinere) programma's.

3) Tengevolge hiervan is de waardering voor korte "turn around time" en bij werk met terminals "short response time" aanzienlijk toegenomen.

4) Tengevolge hiervan zijn de hoge initiele kosten bij programmaverwerking steeds moeilijker te accepteren.

5) Tengevolge hiervan is het steeds noodzakelijker, dat de installatie voor de programmeur "soepel in de hand ligt". Als een installatie een gebruiker met "overlast" confronteert, doet hij dat nu in veelvoud, hoe meer mensen voor een machine programmeren, des te essentieler is het, dat dit met een minimum aan haken en ogen gepaard gaat.

6) Multiprogrammering heeft als middel ter verhoging van de toegankelijkheid van de machine zijn waarde onomstotelijk bewezen.

7) De realistische implementeerbaarheid van "virtual stores", waarin de gebruiker zich niet hoeft te bekommeren over de hoeveelheid voor hem beschikbaar primair geheugen (en waarin deze hoeveelheid door het systeem aan de omstandigheden wordt aangepast) is voor een grote klasse van problemen onomstotelijk aangetoond.

8) Dat het concept van het "virtual store" de gebruiker een hoop overlast bespaart is eveneens onomstotelijk aangetoond.

9) De implementatie van virtual stores heeft een halt toegeroepen aan de ontwikkeling van allerlei (standaard-)programma's in een veelvoud van versie's -gerangschikt naar opklimmende behoefte aan primair geheugen; hiermee is een van de schadelijke groeifactoren effectief bestreden.

Op het Rekencentrum van de THE is deze ontwikkeling, waarin we een essentieel aspect van de dienstverlening zien, voorzien, toegejuicht en tot grote tevredenheid alom daadwerkelijk bevorderd. Het is in dit licht moeilijk om de aanschaf van apparatuur te rechtvaardigen, die ons zou dwingen in dezen de klokl een kleine tien jaar terug te zetten.

* * *

Voorgesteld is een configuratie van een P1400 met daaromheen 4 P880's as satelietcomputers. De P1400 is qua conceptie een machine van de derde generatie, om deze als centrale machine van de configuratie op te stellen is daarom niet aantrekkelijk. De P88o is qua opzet iets moderner: het rekenorgaan is bv. met veel minder expliciet benoemde rekenregisters uitgerust. De wijze, waarop het geheugen geadresseerd wordt is echter nog heel klassiek, wat blijkt uit de wijze, waarop de gebruiker nog met "overlays" te maken heeft. De netto indruk van de P880 is, dat er wel voorzieningen zijn ten bate van het aspect van "processor sharing" zoals dit bij multiprogrammering optreedt, dat de rolverdeling van primair geheugen echter nog zo star is, dat de P880 moeilijk onder "general purpose" omstandigheden is in te schakelen.

In het voorstel zitten drie aspecten:

1) verhogen van de rekencapiciteit door 4 identieke "kleine" computers (voor "kleine" programma's)

2) verhoging van de rekencapaciteit door toevoeging van een grotere machine (voor "grotere" programma's)

3) koppeling van dit vijftal

ad 1) Verhoging van de rekencapaciteit door een aantal identieke computers baadt niet. Hiermee bestrijdt je nl. alleen maar overbezetting, maar niet configurationele tekortkomingen, die ons nu de pas naar ambitieuzere computertoepassingen afsneiden. Hoe tevreden we ook over de X8 zijn, het rekencentrum zou door bv. 4 X8-en onvoldoende geholpen zijn: je kan dan wel meer opgaven van hetzelfde allooi aan, maar niet de opgaven die je nu "zelfs in het weekend" niet aankunt. Het is een beetje alsof je vier ovens tot 250 graden krijgt inplaats van 1 oven tot 1000 graden.

ad 2) Aan het bovenstaande wordt gesuggereerd tegemoet te komen door toevoeging van een grotere machine. Het onderhavige voorstel is moeilijk verdedigbaar omdat de P1400 qua capaciteit wat te weinig van de P880 verschilt, het is absoluut onverdedigbaar omdat de P880 en de P1400 incompatibel zijn. Niet alleen, dat dit zou betekenen dat de gebruiker twee ALGOL systemen zou moeten kennen, het betekent ook, dat hij voor elk programma afhankelijk van de "grootte" van het programma zou moeten kiezen. Dit gaat er van uit, dat de "grootte" een constante programmaeigenschap is, quod non: een toekomstig "groot" programma geeft in zijn testphase tot vele "kleine" runs aanleiding. Verder zou een "klein" programma op de grote machine kunnen moeten draaien vanwege aldaar beschikbare faciliteiten als een procedurebibliotheek.

ad 3) De vermeende voordelen van koppeling zijn mysterieus. Koppeling is in twee omstandigheden duidelijk verdedigbaar:

a) Indien een uiterste aan "beschikbaarheid" geeist wordt (zeg bij "air traffic control"). In zo'n geval koppel je twee identieke machines, die als ze beiden werken identiek hetzelfde doen om ook je vitale informatiebestanden in duplo te hebben. De koppeling dient dan om in geval van tijdelijke uitschakeling van de ene na afloop die weer in staat te stellen de draad op te pakken.

b) Indien een randapparaat of een toepassing zo urgente real time verplichtingen of een zo intensief informatieverkeer met zich meebrengt, dat dit een stuk "dedicated equipment" rechtvaardigt - denk aan het drijven van een visual display.

Maar geen van beide rechtvaardigingen zijn in het onderhavige voorstel van toepassing. Het is absoluut misleidend om de zaak voor te stellen, alsof we er door koppeling "een grote installatie" van maken: het blijven vijf kleine machines elk met hun zeer beperkte capaciteit, daarenboven belast met de koppelingssoftware, die bv. zeker niet in staat zal zijn een door de P880 begonnen berekening "omdat die te groot wordt" door de P1400 te laten overnemen. Kortom het tegendeel van een installatie die lekker in de hand ligt.

Edsger W.Dijkstra