14- 7-1964 X8A 60

Als voorbereiding voor een discussie over een micro-monitor voor de X8 tussen de groepen Utrecht, PTT, en MC, lijkt het nuttig enige feiten en concepten over het op het MC in opbouw zijnde ALGOL-systeem samen te vatten en enige wensen voor een coördinator te formuleren.

Feiten:

1) het ALGOL-systeem heeft ruimte nodig voor:
   a) vast complex
   b) objectprogramma en bibliotheekselectie
   c) stapel
   d) contrastapel
   e) buffers
   f) vertaler
2) het complex en de vertaler staan vast in de machine, op vaste plaatsen; het objectprogramma is niet relocateerbaar en consecutief
3) de stapel is consecutief, niet relocateerbaar, van dynamisch onbekende lengte
4) de contrastapel is consecutief, van dynamisch onbekende lengte, relocateerbaar met dien verstande, dat de relocatie slechts mag plaats hebben op verzoek van het running system aan de monitor. Relocaties op andere tijdstippen zouden nl. oa. de eis stellen dat de complexroutines, die in de contrastapel werken, dan gedeeltelijk in doofheid zouden moeten opereren. Wij zullen de consekwenties hiervan niet gaan onderzoeken.
5) de stapel en de contrastapel groeien naar elkaar toe; bij een treffen zal geprobeerd worden de contrastapel te reorganiseren en zo nodig te verschuiven.
6) het complex, en een aantal macro-opdrachten uit het objectprogramma maken gebruik van een beperkt aantal vaste werkruimtes. Hieronder vallen ook een aantal lambda's en mogelijk enige tau's

Deze organisatie is een gevolg van de beslissing, te streven naar een uiterst efficient ALGOL-systeem, dat de beschikking heeft over de hele machinecode, en zo nodig bijna de hele machine voor zich mag opeisen.

Organisatie bij mono-bedrijf:

Na lange discussies op het MC is het volgende concept gegroeid voor het geheugengebruik bij mono-bedrijf:

1) de vertaler staat helemaal achterin in het geheugen, het complex en de monitor helemaal voorin
2) complex en monitor zijn sacrosanct, de vertaler mag door de monitor vrijgegeven worden als daar dringend behoefte aan is. Van zo'n besluit zal de monitor melding maken
3) de overgebleven ruimte wordt door de monitor in tweeën gedeeld: het laatste stuk is bestemd voor de buffers, het voorste bevat objectprogramma, daaraan aansluitend de stapel, en na een niemandsland de contrastapel tot aan het buffer-territorium.

De grens tussen de twee gebieden noemen we even "plafond"

14- 7-1964 X8A 61

4) de bufferruimte is in eerste instantie bestemd voor output. De organisatie is pagina-gewijs (maar misschien met tamelijk kleine pagina's), gepakt, en in kant en klare outputcode. De pagina's zijn geketend; de eerste schakel wordt asynchroon ontpakt naar een van twee miniatuur-buffers van de monitor, die in wiggel-waggel bedreven worden; de laatste schakel wordt door de monitor synchroon gevuld met de door het running systeem aangeboden symbolen
5) de monitor houdt de in gebruik zijnde bufferpagina's angstvallig zo ver mogelijk van het "plafond" verwijderd, dus zo dicht mogelijk bij de vertaler
6) als de monitor een nieuwe outputpagina nodig heeft, en er is er geen beschikbaar, kan de monitor in een wachtcyclus het vrijkomen van een pagina afwachten. Gebeurt dit geregeld en wordt een door de operateur instelbare drempelwaarde overschreden (deze waarde mag 0 of oneindig zijn!), dan zal de monitor kijken of bufferruimte geschapen kan worden door het "plafond" te verlagen (we rekenen dit tot een relocatie van de contrastapel op verzoek van het running system, omdat dit de monitor aanriep met een outputsymbool). Voor een klein programma met veel output kan de bufferruimte op deze wijze tot duizenden plaatsen aangroeien
7) als de stapel in de contrastapel groeit, zal de monitor de aanvraag ontvangen, het "plafond" te verhogen, en de contrastapel te relocateren. Door de strategie van bufferkeuze-met-ingebouwde-plafondvrees zal zo'n aanvraag meestal direct of gemakkelijk gehonoreerd kunnen worden. Zo mogelijk zal de monitor een ongunstig liggende bufferpagina verplaatsen, zo nodig zal de monitor wachten tot de uitvoer voldoende gevorderd is. Voor deze wacht zal ook een telling bijgehouden worden.
8) als het wachten al te lang duurt (en ook deze drempel zal door de operateur ingesteld mogen worden) zal de monitor mogen besluiten de vertaler op te offeren, en aldus weer veel buffer- en stapelruimte te winnen. Dit gaat ten koste van de plicht na afloop de vertaler opnieuw in te lezen
9) voor imputbuffers geldt, dat ze niet verdwijnen door wachten. De volgende strategie is in ieder geval voorzichtig. Als het systeem moet wachten, omdat een aangevraagd imputsymbool er nog niet is, zal getuffd worden. Afhankelijk van een hoge drempelwaarde zal de monitor kunnen besluiten, meer imputpagina's ter beschikking te stellen. Deze zullen zeker zeer ver van het "plafond" gekozen moeten worden. Krimpen van het aantal outputpagina's legt een druk op de imput en zal het bovengenoemde turven negatief beïnvloeden; vrijkomende imputpagina's kunnen weer van de imput afgenomen worden
10) zodra een programma afgelopen is of om andere reden overlijdt, zal de vertaler in actie komen, terwijl de output nog rustig doorloopt. De vertaler beschikt dan over de ruimte onder het "plafond", maar ook hier zal de monitor weer een bufferspel in gaan spelen. De organisatie hiervan hangt sterk van de aard van de vertaler af; in het volgende zullen we enkele van onze denkbeelden hieromtrent toelichten.

15- 7-1964 X8A 62

Speculaties over de vertaler:

1) het is ons streven, de vertaler geheel of grotendeels in ALGOL te schrijven, en dan machinaal in X8-objectcode om te zetten. De belangrijkste gevolgtrekking zou dan zijn, dat de vertaler gebruik maakt van het running system, en werkt met een stapel en een contrastapel.
2) het grootste probleem hier is dan wel de volledig onbekende lengte van het programma, en het feit, dat het objectprogramma ten slotte aan de voet van het geheugen moet komen te staan. Dit zou gerealiseerd kunnen worden door in blok 0 van de vertaler een zeer groot array te declareren, en daar alle tussenfasen en het eindresultaat in te stouwen. Het nare is dan, dat als dit array te klein blijkt, het spel afgelopen is, want het array is niet uitbreidbaar (en niet own, want dan zou het in de contrastapel komen en niet aan de voet van de stapel)
3) een geheel andere oplossing zou zijn, abstracte transput-kanalen in te voeren, zodanig dat de vertaler via outsymbol (symbol, channel) symbolen aan de monitor kan lozen, terwijl ze later met insymbol (symbol, channel) in de zelfde volgorde terug te winnen zijn. De monitor moet de symbolen dan maar in buffers bergen, en deze kunnen dan paginagewijs en geketend georganiseerd zijn. Eventueel zou dan iedere scan zijn resultaten kunnen dumpen en derhalve als zelfstandig programma kunnen bestaan. Bij 3 scans dus drie programma's. Na de eerste scan moet het al mogelijk zijn een raming te maken voor de uiteindelijke lengte van het objectprogramma, en bij het begin van de laatste scan kan de monitor dan één grote 'outputbuffer' kunnen reserveren voor het objectprogramma en de stapel voor dat vertalerdeel daarboven kunnen laten beginnen.

Hier is dus sprake van een nieuw element voor de discussies over een monitor, nl. een transputkanaal, waarvan de buffers niet asynchroon, gestuurd door ingrepen, geleegd worden, maar door een later programma(deel). Bovendien is de aanvraag van een segment van grote lengte op een gespecificeerde plaats ook tot nu toe niet ter sprake geweest.

De algehele indruk is, dat het ALGOL-systeem hele speciale eisen aan een monitor zal stellen, en deze zeker toegespitst zal moeten zijn op het ALGOL-bedrijf. Voorts zal een monitor, die in solobedrijf bovenstaande taken naar behoren uitvoert, zeker geen kinderwerk zijn. In ieder geval lijkt het me haast ondoenlijk,
1) twee ALGOL-programma's te laten samenspelen,
2) een ALGOL-programma en de vertaler tegelijk aan bod te laten
(strikt genomen zou dit laatste kunnen vallen onder het eerste).
Er is echter geen enkel bezwaar tegen de overlap van output van een voltooid programma met vertalen en draaien van een volgend.