Enige nieuwe ideeën over de assembler en een operatoringreep

Bij enige experimenten met de assembler is bij mij het volgende idee opgekomen hetgeen ik bij deze als voorstel aan de Z8 commissie zou willen voorleggen.

De directieven 'LOC' en 'DEF' zijn overbodig en kunnen vervangen worden door een veel eleganter en logischer mechanisme nl.:
Een label geeft een naam aan een plaats; waarom dan ook geen label gebruikt om aan te duiden op welke plaats wij zijn of op welke plaats wij willen gaan inzetten? Regel: Het verschijnen van een label (iets van de vorm STAT:) die nog niet een bekende plaats aanduidt definieert daarmee de plaats. Deze wordt dus van nu af bekend. Het verschijnen van een reeds bekende label duidt aan dat vanaf dit punt ingezet moet worden. Hierbij mag nu de label ook geindiceerd zijn.
Bovenstaande regel geeft een buitengewoon elegante en duidelijke notatie voor de invoeraanwijzingen. Voorbeeld:

```
M(1000):        A+PIET              "Ga inzetten vanaf 1000
                  :
                  :
PIET:           +12345              "Plaats PIET wordt bekend
                  :
                  :
PIET:           +23456              "Vul nu op PIET iets anders in
```

Dit mechanisme maakt DEF ook overbodig. De assembler kan herkennen dat een dynamisch adres als label wordt uitgedeeld en van dat moment af het werkelijke opbergen onderdrukken. Een label M3(5): is nu helemaal geen bezwaar.
Ook voor het herbenoemen van bijv. A kan men rustig geven:

```
A:                                  "Ga inzetten vanaf 59, het opbergen
                                    "vindt toch niet plaats.
ACCUA:                              "Geef A nu de naam ACCUA
PIET: 'BEGIN' A                     "Ga verder op PIET. Declareer nieuwe A
A:                                  "Nieuwe A ligt hier.
```

Inde regel zal de assembler het beginadres van een program ter beschikking moeten stellen bijvoorbeeld in de vorm

```
M(BEGIN):                           "Ga inzetten vanaf het in BEGIN gegeven
                "adres. Let wel! Niet  BEGIN: zetten want anders wordt dit
```

gegeven adres vervangen door wat anders.

De aanwijzing SKIP zal nog wel nodig blijven daar anders altijd een label gegeven moet worden aan het beginadres van een op te laten ruimte. Deze label hoeft evenwel niet bekend te blijven. Men kan zeggen
'BEGIN' P; P: P(100): 'END'; wat equivalent is met 'SKIP' 100;

Bij het beschouwen van de vorm van enige ingreep programs kwam duidelijk naar voren dat er bij elk zelfstandig program een aantal vaste plaatsen moet zijn waarin de statusgegevens van dat program opgeborgen zitten of opgeborgen kunnen worden. Van deze zelfde plaatsen zal ook de assembler gebruik maken als werkregisters gedurende het assembleren. Het is mij nog niet geheel duidelijk wat in deze als zelfstandige programs aangemerkt zal moeten worden. M.a.w. of ook de "eigen kinderen" van de coördinator in deze zelfstandig zijn. Ook is het nog een open vraag of partieel of totaal gered moet worden.

Elk program wordt gekenmerkt door zijn sleutel, dat is zijn beginadres in het geheugen. Alleen deze sleutel wordt door de coordinator vastgehouden. Het enige actieve program heeft zijn sleutel in SLEUTEL staan. Een tentatieve indeling is:

0	GOTO(:STARTING POINT)	Inspringen op 0 start het program
1	Aantal bezette plaatsen (totaal)	
2	Redplaats voor F bij interrupties	
3	Redplaats voor G	
4	Redplaats voor A	
5	Redplaats voor S	Eventueel zijn deze plaatsen tijdens
6	Redplaats voor B	het assembleren in gebruik voor opberg-
7	Redplaats voor T	instructies en de zes stapel pointers
8	Redplaats voor D	
9	Redplaats voor M(7)	
10	Redplaats voor M(8)	
11	Senseswitch woord	Elk program zijn eigen "consolewoord" veranderbaar via toetsenbord van telex.
12	Plaats van overgebleven vaste naamlijst voor evtl. verder assembleren	
13	Evtl. begin van stack	
14	Bits van dit woord geven de behoefte aan transputapparaten + in hoeverre aan deze behoefte op een bepaald ogenblik is voldaan.	
15	Prioriteitsgetal	
16	Gegevens omtrent inruilbare bufferruimtes	

Het lijkt mij een dwingende eis dat de operateur een program dat op een of andere manier in doofheid of in een DO cyclus is blijven hangen kan verlossen. Knutselen aan de handschakelaars is clumsy en niet status reddend en bovendien niet altijd afdoende want de machine kan niet stoppen in een DO cyclus. Daarom zou ik het volgende voorstel willen doen.
Er komt op de normale console een toets die een enkele gesimuleerde pariteits-fout maakt. In ieder geval zal, ook bij een echte geïsoleerde pariteitsfout het onderzoekingsprogram aan het werk moeten gaan om te kijken of en waar de fout plaats vond en voorlopig het in behandeling zijnde program moeten afmelden op de schrijfmachine en overgaan op een ander program. Hetzelfde gebeurt nu bij het drukken op deze "parity interrupt" toets, alleen weet nu de operateur dat hij na het afmelden en uittikken zelf de oorzaak is geweest van het afmelden. Het voordeel is evenwel dat de machine op geen enkel punt zijn andere werkzaamheden heeft onderbroken en dat zelfs van het foute program de status zo volledig als dat mogelijk was gered is. Misschien dat de coördinator het program naar een zeer lage prioriteit kan verwijzen. Herstarten zou hoogstwaarschijnlijk weer in dezelfde fout loop terecht komen. In ieder geval heeft de operateur de gelegenheid op zijn gemak aan de hand van de geredde gegevens (zie vorige pagina) de aard van de fout op te sporen zonder dat de machine noemenswaard gestoord wordt. De methode om de machine geheel uit te schakelen en Normaal Begin te drukken is hierbij vergeleken zeer dom en grof.

Deze bovengeschetste methode is zelfs bestand tegen het per ongeluk schrijven op 24. Alle ingrepen worden dan onmogelijk. Als alleen 26 maar behouden blijft is de "parity interrupt" mogelijk. Het argument dat dit een gevaarlijke schakelaar zou zijn is dwaasheid. De programmeur is een veel gevaarlijker dier dan de operateur. De programmeur kan alle mogelijke narigheid plegen, zoals doofmaken, DO loops maken, 24 vernietigen, de coördinator overschrijven enz. Daartegenover is deze statusreddende manuele interrupt een zeer ongevaarlijke en in het normale geval zelfs herstelbare handeling. Ik zou dan ook sterk willen pleiten voor een dergelijke toets op de console. De technische implementatie brengt nauwelijks kosten mee. Alles is er al. Alleen moet gezorgd worden dat er slechts één keer een parity fout gesimuleerd wordt en niet fouten zolang de knop gedrukt is.