8 juni 1964 EWD 89-0

Over formal locations

(Résumé van de besprekingen van de werkgroep, gehouden op 8 juni 1964)

Het volgende is een résumé, dat ik uit mijn blote hoofd opstel; in hoeverre het waarheidsgetrouw is, kan ik dus niet garanderen.

We hebben bij de kop gevat het geval, dat de formele parameter gespecificeerd is als real of integer. We zullen dit noemen "arithmetic": we hebben nl. besloten de vertaler tussen deze twee specificaties geen onderscheid te laten maken en altijd het type van de actuele parameter -mits natuurlijk real of integer- te laten prevaleren.

De prijs die we bereid zijn te betalen, bestaat zo op het oog uit vrij omvangrijke formal locations en een analyse van de meegegeven actuele parameters bij binnenkomst in de procedure.

Over de omvang van de formal locations: wij komen zo op het eerste gezicht op vier woorden per parameter.

Over de test: wij stellen ons voor, dat de meegegeven actuele parameters getest worden, d.w.z. aan een controle onderworpen worden, die op grond van de specificaties aan gelegd kan worden (wij stellen specificaties verplicht). Tot nog toe hebben wij ons geen zorgen gemaakt over de snelheid, waarmee deze test aangelegd kan worden ("Is het dit, of dat, of dat etc." Als je eerst vraagt naar het meest voorkomende type actuele parameter, dan is het waarschijnlijk nog niet eens zo tijdrovend.)

Wij nemen ons voor om deze acceptabiliteitstest zover door te voeren, dat daarna de tests, die nog dynamisch uitgevoerd moeten worden, een soort van sinecure zijn.

Omdat de type-overgangen speciale complicaties met zich meebrengen, gaan we het daar eerst over hebben.

We stellen ons voor, dat de arithmetiek altijd in het drijvend register F wordt uitgevoerd. (Uitgezonderd misschien n:= n + 1 of zoiets doms.)

Om een integer als operand, als primary, in een expressie te betrekken,

is dankzij de speciale representatie van drijvende getallen in de X8 geen probleem. Moeilijker is de assignment aan een integer variabele. Dit kan op twee manieren moeilijkheden opleveren: het getal kan niet geheel zijn, het getal kan (na evenetuele afronding!) te groot zijn.
Als F geïntegreerd moet worden, lassen we in het programma in:

```
U, A = M[57], Z          kop F = 0?
N, SUBCD(INTEGREER) →
```

Waarin de subroutine "INTEGREER" -hiervoor moeten we een kortere naam bedenken- luidt (begint met)

```
INTEGREER:    F + AFR              AFR = 3 * 2 ↑ 38
              F - AFR
              U, A = M[57], Z
              Y, GOTOR(MC[-1]) →   en terug
                 ----------------- capaciteitsoverschrijding.
```

Opm. De constante AFR is zo gekozen, dat mits F van te voren niet te groot is (abs. $< 2 \uparrow 38$) de som F + AFR met de binaire komma naar rechts geechelonneerd staat. En passant verzorgt dan het fatsoen van de rekencursist van de EL-X8 de correcte afronding. Vervolgens wordt AFR er weer afgetrokken. Nu is F in elk geval geheel. F kan nog te groot zijn (veel te groot zelfs); vandaar dat de kop van F weer aan een nultest onderworpen wordt.

Een en ander kost ons in het hoofdprogramma twee opdrachten per integer assignment aan plaatsruimte.
Aan tijd

in het normale geval (geen afronding nodig): 2 geh.cont.
als transferfunctie nodig (wel afronding): 12 geh.cont. + exces tijd voor drijvende optelling

Ik verwacht, dat afronding zo weinig voorkomt, dat ik er weinig voor voel om -als standaardpraktijk tenminste- ten koste van vier extra opdrachten in het objectprogramma de afronding vier geheugencontacten te versnellen.

```
              U, A = M[57], Z
              Y, JUMP (4)      →
                 F + AFR
                 F - AFR
              U, A = M[57], Z
              N, GOTO          →      capaciteitsoverschrijding
         →    ------------
```

Tot zover het probleem van afronding en range analyse. (In geval van capaciteitsoverschrijding zullen we de plaats waar ook nog wel een

beetje moeten aangeven. Daar mogen we later dan over denken)

We keren nu terug tot de formal locations.

Het type van de actuele parameter deert ons, zoals bekend, weinig, wanneer de formele in de expressie-situatie uitgewerkt moet worden. Als hij aan de linkerkant van de assignment voorkomt, treedt er nog meer ellende op, zodra we een multiple assignment hebben. Immers:

als een formele linkerkant in zijn eentje voorkomt, heeft een integer actuele tot gevolg, dat er afgerond kan moeten worden;

als een formele linkerkant, samen met een real linkerkant voorkomt, moet een integer actuele tot gevolg hebben, dat er een foutindicatie gegeven wordt, omdat in een multiple assignment de linkerkanten van hetzelfde type moeten zijn;

als in een multiple assignment alle linkerkanten formeel zijn -dankzij de binnenprocedure kunnen dit formelen van verschillende hoogte zijn!- moet de foutindicatie bij verschillende types gegeven worden, en afronding moet bij (homogeen) integer actuelen ingelost worden.

We hebben dus besloten om de multiple formal assignment voorlopig te laten voor wat hij is, d.w.z. het normale geval niet door het bestaan van de multiple formal assignment te laten vertragen. (We hebben voor de multiple formal assignment wel een oplossing, maar die is dan maar een beetje tijdrovend.)

Vast staat, dat we de assignment aan de nonformele scalar exceptioneel zullen behandelen. Als de vertaler in een scan van links naar rechts de assignment statement leest, worden de nonformele scalaire linkerkanten opgezouten en op homogeneiteit van type onderzocht (tezamen met de nonformele subscripted linkerkanten, die van hetzelfde type moeten zijn, maar wel meteen aanleiding tot "programma" geven.)
Als de vertaler het programma gegenereerd heeft, dat de expressie uitrekent in het F-register, test hij, of het verzamelde type van de linkerkant van type integer is, zo ja, dan wordt de integratie ingelast. Vervolgens genereert de vertaler voor elke opgezouten nonformele linkerkant een Mp[q]:= F of een Mp[q]:= G, afhankelijk van het gevonden type. Tenslotte komen de subscripted left hand sides aan bod. (waarover later vast nog een hele hoop meer). Tot zover was er niets formeel.

Nu gaan we bekijken het geval van de single formal assignment.

Hier staat de taal als actuele parameter toe:
1e een real scalar
2e een integer scalar
3e een subscripted real
4e een subscripted integer
5e een formal.

De laatste mogelijkheid confronteert ons echter met niets nieuws, want het meegeven van een reeds formele parameter als actuele is niet meer dan copieren van de formal locations; de laatste mogelijkheid geeft dus geen nieuwe vrijheidsgraad aan de formal locations.

De eerste opmerking is, dat we bij een formele linkerkant eerst deze linkerkant moeten analyseren en zelfs moeten uitwerken, in het geval het een subscripted variable is. De uitwerking van de expressie kan immers als neveneffect hebben, dat de subscript evaluation na afloop andere waarden op zou leveren. (Denk maar aan "R[read]:= read".)

Om die reden zullen we de formele linkerkant eerst moeten "uitwerken", d.w.z. identiteit van de variabele moeten vaststellen. (In geval van meer formele en/of subscripted linkerkanten werken we deze af in volgorde van links naar rechts.)

Het uitwerken van een formele linkerkant zal aanleiding geven tot een stapelvulling, onder controle waarvan de waarde van de expressie, wanneer deze eenmaal in het F-register gevormd is, zal worden weggeschreven. En het is van deze kant, dat we het systeem opbouwen: eerst stellen we vast, wat een plezierige stapelvulling is, opdat dit wegschrijven vlot en snel zal gaan. Hierna stellen we vast, wat geschikte vullingen van formal locations zijn, om bij de uitwerking van de formele linkerkant deze gewenste stapelvulling tot stand te brengen. De stapelvulling ten gevolge van uitwerking van een formele linkerkant noemen we een "linkerkantwaarde".

Op het moment, dat de wegschrijving moet plaatsvinden, staat de waarde in F en de linkerkantwaarde is het top-element van de stapel.

We zullen voor de verschillende gevallen verschillende gestructueerde linkerkantwaarden moeten introduceren. Deze verschillende structuren zijn echter gekoppeld door de eis, dat de feitelijke assignments moe-

ten geschieden op grond van precies dezelfde indicatie in het (aan de body ontleende) objectprogramma. We kiezen deze indicatie zo, dat

a) de in beslag genomen ruimte in het objectprogramma niet te groot is

b) het meest voorkomende geval -actuele een real in de stapel zo vlot mogelijk verwerkt wordt.

Eén opdracht in het objectprogramma is kennelijk wel het minimum en we gaan exploreren, wat de gevolgen zijn, als we hiervoor kiezen

"DO(MC[-1])"

d.w.z. een execute op de top van de stapel met impliciete aflaging van het B-register.

We gaan nu na, wat in de verschillende gevallen de aangemeten linkerkantwaarde moet zijn.

1. Een real op een vast physisch adres "a". Dit doet zich voor als de real een scalar in de stapel is, of een element van een klein array. Bladzijden van grote arrays zullen immers in het algemeen niet op heilige pagina's staan.

1.1. $a < 32768$

In dit geval bestaat de linkerkantwaarde uit een enkel woord (bij de weergave van stapels is de stapelbodem boven!):

```
B - 1: M[a]:= F
B    : --------
```

Tijdsduur 4 geheugencontacten.

1.2. $a \geq 32768$

Dit geval, dat zich alleen voordoet als men beschikt over een geheugen, dat groter is dan 32K behandelen we volledigheidshalve wel. We kunnen dan niet volstaan met een linkerkantwaarde van een enkel woord. Dan moet dus b.v. B met meer dan 1 worden afgelaagd, dit kan door het topwoord van de linkerkantwaarde een subroutinesprong te laten zijn. Een mogelijke structuur van de linkerkantwaarde is:

```
B - 2:       + a
B - 1: SUBCD(:q1)
B    : ----------
```

Waarbij adres :q1 de entry van het systeem is en wel

```
q1:   S = MC[-2]              S:= a, B:= B-1
      MS[0] = F               berg
      GOTOR(MC[0]) ⟹          B:= B-1 en terug.
```

Opm. Wij vestigen er de aandacht op, dat hier wel een stukje gemeen programmeren wordt weggeven. De DO-operatie laagt eerst B af; de uiteindelijke opdracht blijkt een stapelende subroutinesprong te zijn, die zichzelf met de link overschrijft en B weer met 1 ophoogt. De subroutine bestaat uit drie opdrachten, waarin twee stapelverwijzingen de twee nodige B-aflagingen impliciet bewerkstelligen.

Tijdsduur: 10 geheugencontacten.

2. Een integer op adres a

Ook bij integers kan het geval a < 32768 apart behandeld worden; het verschil is echter minder gemarkeerd.

2.1. a < 32768

De structuur van de linkerkantwaarde kan zijn

```
B-2 :  M[a] = G
B-1 : SUBCD(:q2)
B   : ----------
```

```
Met q2:    U, S = M[57], Z
           Y, DO(MC[-2])
           Y, GOTOR(MC[])  ⟶   zonder meer acceptabel
              F + AFR
              F - AFR
           U, S = M[57], Z
           Y, DO(MC[-2])
           Y, GOTOR(MC[0])  ⟶  na afronding acceptabel
              ------------      capaciteitsoverschrijding
```

Tijdsduur minimaal 9 geheugencontacten.

2.2. a ≥ 32768

De structuur van de linkerkant kan zijn:

```
B-2 :   + a
B-1 : SUBCD(:q3)
B   : ----------
```

```
met q3:      S = MC[-2]
             U, S = M[57], Z
             Y, MS[0]= G
```

Y, GOTOR(MC[0]) ⟶ zonder meer acceptabel
⋮ etc.

Tijdsduur minimaal 10 geheugencontacten.

3. Een real op een bladzijde

De linkerkantwaarde beslaat weer twee woorden

B-2 : pakking van physisch adres van de BZV en regelnummer
B-1 : SUBCD(:q4)
B : ----------

De wijze, waarop de subroutine, die op :q4 begint toegang heeft tot de gepakte gegevens is na de vorige routines nu wel duidelijk. Wij merken op, dat wanneer het uit tijdsoverwegingen voldoende gunstiger uitkomt, we hier een linkerkant ook wel drie woorden kunnen laten beslaan, door nl. het physisch adres van de BZV en het regelnummer gescheiden, ieder in een eigen woord op te bergen.

4. Een integer op een bladzijde

B-2 : pakking van physisch adres van de BZV en regelnummer
B-1 : SUBCD(:q5)
B : ---------

De routine, die bij q5 begint, begint (zie 2) af te ronden, en verloopt verder analoog aan die van q4. Ook hier kunnen desgewenst de hier gepakte gegevens in twee aparte woorden worden geborgen.

Uitwerking van formele variabelen

Wij moeten nu, voor elk type van actuele parameter gaan vaststellen, welke structuur de formal locations zullen vertonen. De formal locations bepalen de actuele parameter, die in het algemeen op twee wijzen kan moeten worden uitgewerkt, nl. als linkerkant en als rechterkant. In het laatste geval stellen we als eis, dat de waarde in het F-register verschijnt, in het eerste geval, dat de linkerkantwaarde, als boven beschreven, aan de stapel wordt toegevoegd.

We gaan uit van vier formal locations, in volgorde van opklimmend geheugenadres beschreven als f[0], f[1], f[2], en f[3]. We spreken af,

dat uitwerking als rechterkant in het objectprogramma van de body gecodeerd zal worden als

DOS (f[0])

en uitwerking als linkerkant als

DOS (f[1])

Opm.1. De formal locations zullen, als grootheden in de stapel, via de dynamische adressering geadresseerd worden.

Opm.2. Het meegeven van een formele parameter als actuele willen we effectueren, door "ongezien" copieren van de formal locations. We moeten er dus voor zorgen, dat de structuur van de formal locations zodanig is, dat ze tegen verplaatsing (nl.copiering elders) bestand zijn.

We volgen nu dezelfde indeling.

1. Een real op een vast physisch adres a

1.1. a < 32768

f[0]: F = M[a]

f[1]: SUBCD(:q6)

f[2]: M[a] = F

f[3]: voor alsnog ongebruikt.

Opm. f[3] hebbe wel de goede pariteit in verband met copiering van de formal locations.

De subroutine, die op q6 begint, is even wat gecompliceerd, omdat de link boven op de stapel komt en het juist deze plaats is, die gevuld moet worden. Dit impliceert "redden van de link".
De systeemroutine luidt

```
q6:  G = MS[1]
     S = M[B-1]              red link
     M[B-1] = G
     GOTOR(M[60])   ⇒        terug via S-register.
```

Tijdsduur voor linkerkantuitwerking: 10 geheugencontacten.

De rechterkantuitwerking spreekt voor zichzelf en duurt 4 geheugencontacten.

1.2. a ≥ 32768

Volledigheidshalve geven we de formal locations voor een vast gesitueerde real boven de 32 K:

```
f[0]   SUBCD(:q7)
f[1]   SUBCD(:q8)
f[2]   + a
f[3]   SUBCD(:q3)
```

Hier geeft q8 de systeemroutine aan:

```
q8:   F = MS[1]
      S = MC[-1]
      MC[0] = F
      GOTOR(M[60])
```

een routine, die grote overeenkomst vertoont met q6; de netto verhoging van B met 1 wordt door - 1 + 2 bewerkstelligd. De tijdsduur voor linkerkantuitwerking is nu 12 geheugencontacten.

De systeemroutine q7 luidt:

```
q7:   S = MS[2]
      F = MS[0]
      GOTOR(MC[-1])
```

en vergt 10 geheugencontacten.

2. Een integer op een vast physisch adres a

2.1. a < 32768

```
f[0]   G = M[a]
f[1]   SUBCD(:q8)
f[2]   M[a] = G
f[3]   SUBCD(:q2)
```

Hierbij wordt dezelfde systeemroutine gebruikt (nl.q8) als in geval 1.2. Uitwerking van een linkerkant kost 12 geheugencontacten, uitwerking van een rechterkant 3.

2.2. a ≥ 32768

Volledigheidshalve geven we dit geval weer

```
f[0]   SUBCD(:q9)
f[1]   SUBCD(:q8)
f[2]   + a
f[3]   SUBCD(:q3)
```

waarbij q9 (analoog aan q7) luidt:

```
q9:   S = MS[2]
      G = MS[0]
      GOTOR(MC[-1])
```

Uitwerking van een rechterkant duurt dan 9 geheugencontacten, die van een linkerkant als gebruikelijk 12.

3. Een real op een bladzijde

De actuele parameter is dan een subscripted element. We nemen dat deze gegeven is door een impliciete IS, die (zo ongeveer) het element localiseert in termen van physisch adres van de BZV en regelnummer. Aanwezigheidsanalyse van de bladzijde wordt door de impliciete subroutine nog niet gepleegd.

De impliciete subroutine wordt in de formal locations gekarakteriseerd door een "invariant beginadres". We laten in het midden of hierin de BZV van de pagina gegeven is door zijn physische adres, dan wel door zijn adressering als locale van het buitenste blok (eigen bladzijde) of de bibliotheeklijst. Physisch adres is waarschijnlijk het snelste.

De inhoud van de formal locations is dan

```
f[0]   SUBCD(q10 of q11)
f[1]   SUBCD(q12 of q13)
f[2]   invariant beginadres impliciete subroutine
f[3]   context D.
```

Routines q10 en q12 zijn bedoeld voor het geval, dat de IS simpel is, wat o.a. impliceert, dat de pagina van de body niet geprofaneerd hoeft te worden en er in de stapel per definitie voldoende anonieme ruimte zal zijn. In dit geval kan de machinelink als terugadres gehandhaafd worden en hoeft alleen de heersende D aan de terugkeergegevens toegevoegd te worden. Anders (q11 en q13) moet de link in invariant terugkeeradres terugvertaald worden en de pagina van de body geprofaneerd worden.

Onder controle van S hebben deze routines toegang tot de IS en de context D, die ingevuld moet worden om de IS correct te kunnen uitvoeren. Verder moet de bladzijde waar de IS begint heilig aangevraagd worden. (Als hij al heilig aanwezig is, betekent dit al, dat HHT = 2 wordt!)

Na afloop van de IS keert de besturing terug in de systeemroutine. Afhankelijk van linker of rechterkant wordt of alleen de pakking van BZV-adres en regelnummer -door de IS afgeleverd- op de top van de stapel gezet en dit afgedekt met de constante "SUBCD(:q4)" of wordt selectie gepleegd -inclusief aanwezigheidscontrole- en de gevraagde real in F gezet.

4. Een integer op een bladzijde

De inhoud van de formal locations is dan

f[0] SUBCD(q14 of q15)

f[1] SUBCD(q16 of q17)

f[2] invariant beginadres impliciete subroutine

f[3] context D.

Deze routines verschillen slechts van de vorige vier, doordat bij linkerkantuitwerking de pakking met"SUBCD(:q5)" wordt afgedekt en bij rechterkantuitwerking alleen het geselecteerde woord in G geplaatst wordt.

Opm! Zodra we ook"kleine arrays" in de stapel willen toestaan, dan zullen (kunnen) de laatste acht subroutines na terugkeer uit IS inspecteren, of het aangewezen element een van een klein array is. In dat geval is de linkerkantwaarde als van type 1 of 2, de rechterkantwaarde is dan triviaal.

Opm.2 Deze acht subroutines lijken verdacht veel op elkaar en het ligt dus voor de hand om ze van gemeenschappelijke routines gebruik te laten maken. Men lette erop, dat alle status informatie dan op de stapel moet komen, omdat ze genest en parallel gearchiveerd kunnen worden -d.w.z. door verschillende programma's door elkaar gebruikt kunnen worden.

Hiermede zijn alle actuele parameters, corresponderend met een specificatie arithmetic behandeld voorzover zij een bestaanbare linkerkantwaarde hebben. De overige zijn de constante, de expressie en de arithmetic procedure SS (Self Supporting, d.w.z. een procedure identifier behorend bij een procedure zonder parameter).

Bij binnenkomst in de procedure zullen de formal locations op aanvaardbaarheid getest moeten worden; hier wordt niet getest, of een arithmetic actual ook inderdaad een linkerkantwaarde heeft. Daarom zullen wij deze formal locations (met name f[1]) zo vullen, dat àls de body probeert, hun linkerkantwaarde uit te werken, dit alsnog tot protest aanleiding geeft.

5. De real constante

f[0] F = MS[2]
f[1] Alarmsprong
f[2]]
f[3]] waarde van de constante

Tijdsduur rechterkantuitwerking: 4 geheugencontacten.

6. De integer constante

(Dit is misschien een beetje te veel eer!)

f[0] G = MS[3]
f[1] Alarmsprong
f[2] invariant beginadres impliciete subroutine
f[3] waarde integer constante

Tijdsduur rechterkantuitwerking: 3 geheugencontacten.

7. De arithmetische expressie

f[0] SUBCD(q18 of q19)
f[1] Alarmsprong
f[2] invariant beginadres impliciete subroutine
f[3] contact D.

Het onderscheid tussen de systeemroutines q18 en q19 is weer of de impliciete subroutine simpel is of niet.

8. De enkele type procedure identifier, die als expressie op mag treden

De inhoud van de formal location wordt, zodra we goed en wel ons in de body bevinden

f[0] SUBCD(:q20)
f[1] Alarmsprong
f[2] invariant beginadres van de procedure
f[3] context D.

De moeilijkheid hier is, dat de procedure ook expliciet als procedure aangeroepen kan worden. Nu stellen wij ons voor om bij de call de formal locations van de parameters te vullen, vervolgens dit aantal mee te geven en dan naar de (nog "ongeziene") procedure te springen, die begint te testen of het aantal meegegeven parameters correct is. De body van de procedure zonder parameters begint dus te testen of hij inderdaad geen parameters heeft meegekregen.

De functie van de systeemroutine q20 is om het stukje calling sequence behelzende "nulparameters" dynamisch in te lassen, voordat volgens alle regels van de kunst de procedure geactiveerd wordt.

De formal location vulling met SUBCD(:q20) kan echter slechts op twee manieren tot stand komen

1e Door een arithmetic formal parameter -waarbij de actuele een procedure identifier was- als actuele parameter door te geven. Doorgeven is immers copieren van de formal locations.

2e Door een arithmetic procedure identifier als actuele parameter mee te geven, waar de formele als arithmetic gespecificeerd blijkt te zijn. Het meegeven van een procedure identifier als actuele parameter geschiedt door de vertaler:

1e zonder bewustzijn van het aantal parameters, dat deze procedure vergt

2e zonder bewustzijn of de overeenkomstige formele als procedure of als arithmetic gespecificeerd is.

Als de formele als procedure gespecificeerd is, dan inspecteert de body de formal locations om te kijken of er een procedure van het goede type is meegegeven. Bij gebruik van deze als procedure gespecificeerde parameter zal de callside het feitelijk aantal parameters (kan = 0 zijn) expliciet aangeven.

Als de formele parameter als arithmetic gespecificeerd is, dan wordt als overeenkomstige actuele een procedure identifier van het goede type geaccepteerd, maar de formal locations worden gewijzigd door de invulling van"SUBCD(:q20)" in f[0]. Dit proces vindt plaats ongeacht het aantal parameters, dat de meegegeven procedure graag zou willen hebben. Bij dynamisch gebruik wordt dit in de procedure getest; q20 zorgt nl. voor expliciete meegave van "nulparameters". (Ook dan hoeft de Alarmsprong pas in de formal location ingevuld te worden: bij een procedure, die als zodanig gespecificeerd is, is een alarmsprong immers een open deur!)