Enige mededelingen en voorstellen betreffende ELAN

Het gebruik van het directief 'START'

'START' wordt gebruikt om een stuk van een program of een geheel program in werking te stellen vanuit de assemblagefase. Het directief 'START' kan gevolgd worden door een absoluut adres, een :NAAM, een NAAM, een :NAAM[INDEX] of een NAAM[INDEX]. Voorbeeld:

```
'START' :PIET          ''START OP ADRES PIET. PIET MOET BEKEND ZIJN.
'START' PIET           ''START OP ADRES VERMELD IN PIET.
```

Starten mag dus tussentijds (voor interludes) of op het eind. Men moet echter wel bedenken dat tussentijds starten met zich mee kan brengen dat nog niet alle blokken beeindigd zijn door 'END' en dat er nog onbekende namen kunnen zijn. De programmeur is geheel verantwoordelijk hiervoor.

Met interludes is geassocieerd een standaard replica

GOASS: GOTO(:<juiste punt van de assembler>)

om terug te keren naar de assemblagefase. Voorbeeld:

```
L:      * S = N
        MULS(M)                ''VERMENIGVULDIG N MET M
        L = S
        GOASS
        'START' :L             ''START INTERLUDE EN PLAATS N x M IN L
L[0]:   'SKIP' L               ''NA TERUGKEER IN ASSEMBLER: SKIP N x M
                               ''OM ARRAY VAN NM TE RESERVEREN.
```

Voorstel voor standaardafspraak: Bij ieder program dat later geherstart moet kunnen worden (of dat niet direct door de programmeur zelf gestart wordt) wordt het benodigde inspringpunt geplaatst als laatste opgeborgen adres in het program. Het standaard inspringpunt bevindt zich dus op STACK - 1 . De coordinator kan na aanwijzing van de operateur het betreffende program dus starten door te eindigen met

```
A = STACK
GOTO(MA[-1])
```

Zie voorts voorbeelden verderop.

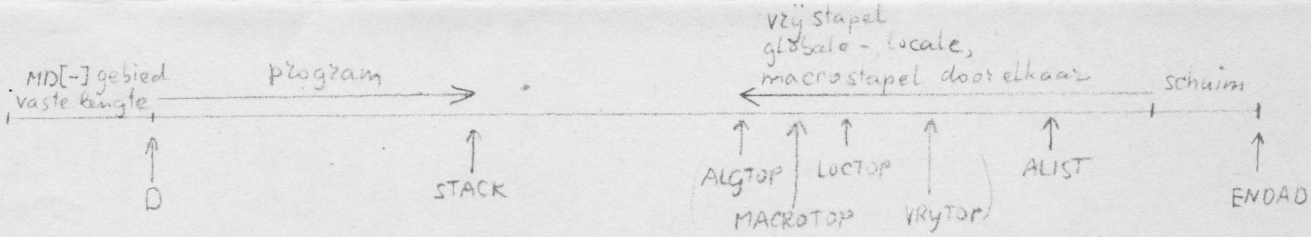
De volgende standaardhandelingen van de assembler zijn bereikbaar met behulp van 'START' :

'START' COORD Verlaat het assembleren en geef de controle terug aan de coordinator. Sluit voorts de AM waarop het program geassembleerd werd. Het program staat nu inactief maar klaargeassembleerd in het geheugen en wacht op een "herstart" van de operateur om verder te gaan werken.

'START' CLENA Clear namelist. Aan alle namen waarvoor nog geen label verschenen is wordt een plaats toegewezen. Verder wordt de naamlijst (die alleen nog supernonlocale namen bevat)aangeplempt en achter het program geschreven. Het begin van de globale lijst is weer te vinden in ALIST evenals voor het aanplempen.

Er volgt een foutsignalering als men tracht 'START' CLENA te geven als het aantal gelezen 'BEGIN's ≠ het aantal gelezen 'END's. Men mag wel tussentijds aanplempen en niet gelabelde plaatsen toewijzen. Alle verantwoording berust weer bij de programmeur.

Heeft men vrije namen gebruikt die nergens als label verschijnen dan is men <u>verplicht</u> tenminste eenmaal aan het eind 'START' CLENA te geven.
Voor 'START' CLENA



Na 'START' CLENA

Stelt men verder geen prijs op de aangeplempte naamlijst dan kan men geven
M[ALIST]: '' MAAK STACK GELIJK AAN ALIST

'START' ATYPE Hiermede wordt het op een of andere manier uitvoeren van de aangeplempte naamlijst verzorgd ter controle van spelfouten bij gedeclareerde namen of om de naam-adres correspondentie te hebben voor alle globale namen.

'START' AFREE Hiermede wordt alle ruimte die nog over is na STACK in gehele bloklengten weer ter beschikking van de coordinator gebracht. Men zal dit kunnen gebruiken bij het assembleren van programs met veel locale namen en veel macros maar zonder stackbehoefte. Tijdens het assembleren is dan een enorme ruimte nodig geweest voor de locale lijst die gerust na assemblage teruggegeven kan worden.

Na 'START' CLENA, 'START' ATYPE en 'START' AFREE komen deze routines weer vanzelf terug in de assembler (met GOASS). Onmiddellijk voor het geven van 'START' CLENA kan men door het geven van een nieuw inzetadres de gepakte naamlijst terecht laten komen waar men wil. Ook kan men door het geven van een nieuw inzetadres vlak voor het einde STACK zetten waar men wil of er zelfs helemaal geen gebruik van maken. Bedenk echter wel dat 'START' AFREE alles teruggeeft achter STACK en verder dat het herstartpunt gelegen is op STACK - 1.
Voorbeeld van een mogelijk einde van een program:

```
EIGENSTACK:'SKIP' 100              ''LAAT EEN EIGEN STACK VRIJ
           'START' CLENA           ''PAK NAAMLIJST EN GEEF ONGELABELDEN PLAATS
           'START' ATYPE           ''TYPE NAAMLIJST
M[ALIST]:                          ''MAAK STACK = ALIST. DUS RUIM LIJST OP
           'START' AFREE           ''GEEF ALLE VERDERE RUIMTE TERUG
           :PIET                   ''VUL ALS LAATSTE HET HERSTARTADRES IN
           'START' COORD           ''HAD OOK 'START':PIET MOGEN ZIJN.
```

Na het gebruik van 'START' COORD is het apparaat waarover het program geassembleerd is afgemeld met CLOSE en zal elk nieuw apparaat bij het herstarten van het program opnieuw met OPEN aangevraagd moeten worden. Start men daarentegen met 'START' <adres van het program zelf> dan blijft het apparaat waarover het program binnen is gekomen beschikbaar met NEXT of LOOK en behoeft dit apparaat niet geOPENd te worden. Op deze wijze kan het voer direct achter het program volgen.

Na een geaccepteerde taakmelding geeft de coordinator het beginadres van het toegestane traject mee in de vorm D := <beginadres> + <vaste lengte van het MDmin gebied> en het eindadres (nalaatste adres) in ENDAD (=MD[-1]). Er wordt daarna automatisch de eerste inzetadres aanwijzing MD[0]: gegeven (dus STACK := D). Wil men zelf indexregisters vrijhouden of blokken in dynamische vorm inlezen dan kan men dit altijd herroepen door MD[n]:. Vanzelfsprekend heeft men geen enkele verplichting ook later tijdens de werking van het program gebruik te maken van deze display. Wel verdient het aanbeveling om de waarde van D bij het begin van het uitvoeren even vast te leggen zodat men altijd door het herinstalleren van deze waarde in D de grootheden ENDAD. STACK, ALIST en alle andere MDmin grootheden kan raadplegen. Tijdens de assemblage is alles buiten het interval D tot ENDAD niet beschrijfbaar. Ook in een niet-geprotegeerde machine is het dus niet mogelijk om zonder meer het toegestane interval te overschrijden. Dit is alleen dynamisch mogelijk door een interlude en dan nog slechts zover als de hardware protectie toelaat.

De standaardnamen van de assembler

De vaste naamlijst van de assembler kent naast de bekende standaardnamen voor variabelen (A = M[59] , S = M[60] enz.) en standaardfuncties (MULAS, GOTO, LUS enz.) de volgende namen:

ENDAD = MD[-1] . In ENDAD staat het nalaatste adres van het traject vermeld.

ALIST = MD[-2] . In ALIST staat de stapelwijzer van de supernonlocale naamlijst voor of na het aanplempen.

STACK = MD[-3] . Stack bevat het lopend bergadres

Andere werkregisters van de assembler die in het MD min gebied zijn gelegen zijn wel bereikbaar maar uitsluitend onder de namen MD[-n] . Men is natuurlijk vrij om zelf hiervoor namen in te voeren door MD[-8] : KAREL: M[STACK] :

De vaste namen van de apparaten zijn:

TARE1 ... TARE9,TAR10 ..	voor tapereaders
TAPU1 ... TAPU9,TAP10 ..	voor tapepunches
CARE1 ... CARE9,CAR10 ..	voor cardreaders
CAPU1 ... CAPU9,CAP10 ..	voor cardpunches
MATA1 ... MATA9,MAT10 ..	voor magnetic tapes
LIPR1 ... LIPR9,LIP10 ..	voor lineprinters
TEPR1 ... TEPR9,TEP10 ..	voor teleprinters
TEKE1 ... TEKE9,TEK10 ..	voor teleprinter keyboards

Ten behoeve van de standaard invoer en uitvoer zijn beschikbaar de standaardmacro's : OPEN, NEXT, LOOK, CLOSE, INDIT en EXDIT. Zolang de assembler de macrofaciliteiten nog niet heeft (laat staan de faciliteit van standaardmacro die voorlopig naast gedeclareerde macro's nog niet ontwikkeld is) worden deze routines bereikt met: SUBC(:OPENX), SUBC(:NEXTX), SUBC(:LOOKX), SUBC(:CLOSX), SUBC(:INDIX) en SUBC(:EXDIX).

De bijzondere adressen van de assembler

Tijdens het evalueren van indices, het opbergen van een geassembleerd woord of vlak voor het inspringen in een program na 'START' zijn de volgende registers gevuld:

A bevat het beginadres van de assembler

S bevat het beginadres van de coordinator

B bevat het bergadres STACK

Men kan van deze feiten gebruik maken om in de assembler zelf veranderingen aan te brengen. Het spreekt vanzelf dat dit alleen mogelijk is als men

met behulp van geschikte interludes eerst de software-(en eventueel ook de hardware-) protectie heeft opgeheven. Een en ander is dus wel voorbehouden aan de zeer deskundige goedwillenden. Ongeacht de werkelijke plaats van assembler en coordinator kan men dus met inzetaanwijzingen M[A + n]: iets veranderen op adres n van de assembler.
N.B. het gebruik van MA[n]: voor dit doel is alleen goed als n < 256 en bovendien worden dan alle labels binnen het daaropvolgende blok dynamisch van het type MA. Wel kan men gebruiken MA[0]:'SKIP' n

Men kan soms met vrucht gebruik maken van het feit dat B het bergadres bevat. In een interlude kan men zonder meer een op een bijzondere manier in elkaar gezet woord opbergen met MC = A; GOASS of iets dergelijks. M[B + x]: is in zoverre equivalent met 'SKIP' x dat het inzetten ook x plaatsen opschuift, maar bovendien wordt er nu een regel (of blok) van het M[B] type ingevoerd.

Een soms wel handige truc voor het uitdelen van een enkele statische label in een dynamisch geadresseerd blok kan zijn:

```
M3:'BEGIN'
    ¦
  X: ¦            ''X WORDT EEN ADRES VAN DE VORM M3[n]
    ¦
  M[STACK]: Z:    ''Z WORDT EEN STATISCH ADRES. TYPE WISSELT SLECHTS VOOR
    ¦             ''EEN REGEL EN GAAT DAN WEER DYNAMISCH DOOR.
    ¦
  Y: ¦            ''Y WORDT EEN ADRES VAN DE VORM M3[m]
    ¦
  'END'M3
```

BINAIRE BANDEN VOOR DE X8

Binaire banden kunnen geponst worden in verschillende media nl. 5-gats en 7-gats ponsband. In beide gevallen heeft alleen de bitrij die men verkrijgt door alle bandsymbolen als bitrij te beschouwen met het hoogste orde bit voorop en vervolgens naast elkaar te zetten. Dat dan bij een 5-gats band een geheel andere verdeling over de symbolen tot stand komt dan bij de 7-gats band doet hierbij niet terzake. Wel is een verschillende aankondiging nodig voor 7-gats en 5-gats binaire band. Voorgesteld wordt als standaardnamen voor de soort code te gebruiken: BIBA5 en BIBA7
Men kan op de binaire band verschillende bitaggregaten als syntactische eenheden onderscheiden. Elk van deze syntactische eenheden hoewel bestaand uit een verschillend aantal bits heeft een bijbehorend pariteitsbit.
De volgende syntactische eenheden worden onderscheiden:
De primaire stuurinformatie s bestaande uit twee bits plus pariteitscheck
De secundaire stuurinformatie x bestaande uit drie bits plus pariteitsbit
Een woord d bestaande uit 27 bits plus pariteitsbit.
Een adres a bestaande uit 18 bits plus pariteitsbit.
Een index i bestaande uit 6 bits plus pariteitsbit.
Deze syntactische eenheden kunnen in de volgende structuren optreden (de semantiek zal er in een soort ELAN ALGOL bijgegeven worden):

s d	met s = 0	M[STACK] := d ; STACK := STACK + 1
s d	met s = 1	M[STACK] := d + PREGIFT ; STACK := STACK + 1
s i	met s = 2	PREGIFT := DISPLAY i

Alle volgende structuren hebben s = 3 en hun functie wordt nader bepaald door x:

s x a	met x = 0	STACK := a
s x a	met x = 1	GOTO(:a) en synchroniseer op eerstvolgende n-ade
s x a	met x = 2	:DISPLAY := a
s x a	met x = 3	synchroniseer op eerstvolgende n-ade
s x a i	met x = 4	STACK := a + DISPLAY[i]
s x a i	met x = 5	GOTO(:a + DISPLAY[i]) en synchroniseer.
s x a i	met x = 6	DISPLAY[i] := a + PREGIFT
s x a i	met x = 7	nog ongebruikt

Blank aan het begin van een binaire band wordt geskipt. De band moet beginnen met s = 2 of s = 3 (het begin na coordinatorinformatie wel te verstaan). Met s = 3, x = 0 kan men een beginbergadres zetten. Met x = 2 kan men het beginpunt van een display van referentieadressen vastleggen. Met x = 6 kan deze display gevuld worden. Met s = 0 worden absolute woorden gelezen. Met s = 2 wordt een referentiepunt "heersend" gemaakt. Met s = 1 wordt relatief t.o.v dat referentiepunt getransformeerd. De rest spreekt voor zichzelf.