NL 5.

Beskrivelse af konventionerne
for brug af
Indlæse- og Oversætterprogrammet NL 5
der læser
DASK ALGOL samt NL 5-KODE.

Chr. Gram og P. Mondrup

Febr. 1962

NL 5, C.G.
Febr. 1962

1. Indledning.

Ved NL 5 forstaas det indlæse- og oversætter-program, der baade kan oversætte programmer skrevet i ALGOL, programmer skrevet i en speciel maskinkode (NL 5-kode), samt programmer der er skrevet dels i ALGOL og dels i NL 5-kode.

Vi vil her beskrive konventionerne for NL 5-kode og for dennes brug i ALGOL-programmer, men hvad angaar konventionerne for ALGOL-programmer i-øvrigt henvises til Chr. Andersen: Lærebog i Algol, P.Naur: Report on the Algorithmic Language ALGOL 60, samt P. Naur m. fl.: A Manual of the DASK ALGOL Language.

Desuden er der til sidst en operationsliste for DASK, hvor virkningen af hver operation dels er beskrevet i text og dels (hvor det ikke er meget uoverskueligt) i ALGOL. Ogsaa den indledende adresseberegning er skrevet i ALGOL, idet dog de nødvendige erklæringer (declarations) overalt er udeladt, da det forhaabentlig er lykkedes at vælge betegnelser, der umiddelbart vækker de rigtige associationer hos læseren.

NL 5-koden er fastlagt saaledes, at programmer skrevet i NL 4-kode kun skal ændres paa ganske enkelte punkter for at kunne indlæses v.hj.a. NL 5.

For at udnytte forbindelsen mellem ALGOL-programmer og NL 5-kode, er det nødvendigt at vide ganske nøje, hvordan NL 5 oversætter et ALGOL-program til maskinkode, men hvad angaar dette, henvises til P. Naur m. fl.: A manual of the DASK ALGOL Language.

2. Ordrens opbygning.

2.1 Syntax.

Beskrevet med samme metasprog som det, der i rapporten om ALGOL 60 benyttes til beskrivelsen af selve ALGOLs syntax, kan den enkelte ordre i NL 5-kode defineres ved følgende:

<ciffer> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<bogstav> ::= a | b | c | d | e | ... | x | y | z | æ | ø
| A | B | C | D | E | ... | X | Y | Z | Æ | Ø

<fortegnløst heltal> ::= <ciffer>|<fortegnsløst heltal> <ciffer>

<identifikator> ::= <bogstav>|<identifikator> <bogstav>
|<identifikator> <ciffer>

<primær> ::= <fortegnsløst heltal>|<identifikator>| k | t

<operator> ::= + | - | × | /

<fortegnsløs adr.del>::= <primær>|<fortegnsløs adr.del> <operator> <primær>

<adr.del> ::= <fortegnsløs adr.del>|+ <fortegnsløs adr.del>
|- <fortegnsløs adr.del>

<oper.del> ::= <sedec.oper.del>|<mnemonic>|<macro>

<index> ::= a | b | c | d | A | B | C | D

<ordre med index> ::= <oper.del> <adr.del>,<index>|<oper.del>, <index>

<ordre> ::= <oper.del>|<oper.del> <adr.del>|<ordre med index>

2.2 Betydningen af ordrens bestanddele.

Adressedelen: De i en adressedel optrædende identifikatorer vil under indlæsningen faa tillagt heltalsværdier. Derpaa udregnes heltalsværdien modulo 2048 af hele adressedelen under benyttelse af de sædvanlige regneregler (efter hver division afrundes kvotienten korrekt ifølge reglen q afrundet:= entier (q+0.5)), og dette tal indgaar som adressetal for den paagældende ordre.

Herunder faar k værdien af den løbende adresse, d.v.s. adressen paa den halvcelle, hvor den aktuelle ordre lagres; k kan altsaa bruges ved relativ adressering, d.v.s. adressering i forhold til den aktuelle ordre.

Værdien af t bliver nummeret paa den første frie tromlekanal, d.v.s. den første tromlekanal (efter kanal nr. 100), som ikke er beslaglagt af det aktuelle program.

Har en indlæst ordre ingen adressedel, faar den tilsvarende (i maskinen lagrede) ordre adressetallet 0, undtagen i det tilfælde, hvor ordren bestaar af en macro alene (d.v.s. at ordren hverken har adressedel eller indexmærke). I dette tilfælde oversættes den indlæste macro til en 17-ordre; se iøvrigt operationslisten.

Indexmærket: Har en ordre ingen indexmærke eller indexmærket a eller indexmærket A, angiver ordrens adressetal den effektive adresse (og den i maskinen indlæste ordre vil iøvrigt have samme udseende i de tre tilfælde; motivet til at skrive et indexmærke a eller A er kun, at det kan fungere som skilletegn mellem ordrer; se nedenfor).

Indexmærkerne b og B bevirker, at den effektive adresse under udførelsen af ordren beregnes som summen af adressetallet og det øjeblikkelige indhold i indexregister B. Indexmærkerne c og C samt d og D har ganske tilsvarende virkninger.

Operationsdelen: De tilladte kombinationer i <sedec.oper.del>, <mnemonic> og <macro> fremgaar ikke af ovenstaaende syntax, men kan findes i operationslisten.Da der til hver mnemonic svarer en sedecimal operationsdel med samme virkning, er disse beskrevet under eet; en sedecimal operationsdel skrives som to sedecimale cifre, og under hver sedecimal operationsdel i operationslisten finder man den tilsvarende mnemonic (der bestaar af 3 eller 4 symboler). Betegnelserne kan i et program bruges helt i flæng.

En ordre, hvis operationsdel er sedecimal eller er en mnemonic, vil blive oversat til een maskinordre og altsaa netop fylde een halvcelle i lageret. En ordre, hvis operationsdel er en macro, vil blive oversat til een, to eller tre maskinordrer afhængig af, hvilken macro der er tale om (og om ordren har adressedel eller ej).

Mens de sedecimale operationsdele (og dermed ogsaa mnemonics) har samme virkninger som operationsdelene i det gammelkendte NL 1-sprog (se Lærebog i kodning for DASK), udfører de fleste af macroerne de sædvanlige aritmetiske operationer med flydende regning; her udnyttes de faste sekvenser i sekvenslageret, hvorfor der i oversættelsen af en macro altid indgaar en 17-ordre. To af macroerne udfører tromleadministrations-funktioner. Iøvrigt henvises til operationslisten.

Eksempel 2.1
Følgende er ordrer:

```
00  5      ,a
00  5 + k  ,B
01  a4 - 5
```

3. Opbygning af et program.

3.1 Indledning.

Da ALGOL og NL 5-kode er saa intimt sammenknyttet, er det uhensigtsmæssigt at opskrive en fuldstændig syntaktisk beskrivelse af NL 5-koden for sig. Derfor er nedenstaaende syntax uafsluttet, men ser man bort fra den tredie mulighed under <kodesætning> og den anden mulighed i definitionen af <kodehale> (det er netop dem, der knytter forbindelsen til ALGOL), indeholder syntaxen alt om NL 5-kode undtagen beskrivelsen af færdige programmer. Dette findes i næste afsnit.

3.2 Ordrer, konstanter, ordretællere og definition af etiketter.

Den første del af syntaxen definerer, hvordan man jævnsides med ordrer kan indlæse forskellige typer af konstanter samt oplysninger til indlæseprogrammet.

Syntaxen lyder saaledes:

<uafsl. enhed> ::= <ordre>|<k-tæller>|<t-tæller>|<etik.def.>|<indhop def.>
|h <halvordsgruppe>|f <helordsgruppe>|r <talgruppe>
|s <sedec. gruppe>

<afsl. enhed> ::= <ordre med index>| {<< <tekststreng> } { <layout> }

<sum> ::= <fortegnsløst heltal>|< fortegnsløst heltal> + <sum>

<k øget> ::= k + <sum>|<sum> + k|<sum> + k + <sum>

<k-tæller> ::= k:= <k øget>

<t øget> ::= t + <sum>|<sum> + t|<sum> + t + <sum>

<t-tæller> ::= t:= <t øget>

<etik.def.> ::= <identifikator>:= <adr.del>

etiketteliste> ::= <identifikator>|<identifikator> , <etiketteliste>

<indhop def.> ::= in <etiketteliste>

<talgruppe> ::= <number>|<number> , <talgruppe>

<halvordsgruppe>::= <talgruppe>

<helordsgruppe> ::= <talgruppe>

<5-gruppe> ::= <sedec. cif.><sedec. cif.><sedec. cif.>
<sedec. cif.><sedec. cif.>

<sedec. gruppe>::= <5-gruppe>|<5-gruppe><sedec. gruppe>

<sedec. cif.> ::= o|O|0|1|2|3|4|5|6|7|8|9|a|b|c|d|e|f|A|B|C|D|E|F

3.3 Betydningen af bestanddelene:

Af de ovenstaaende bestanddele af en enhed er <k-tæller>, <t-tæller>, <etik.def.> og <indhop def.> oplysninger til indlæseprogrammet, som ikke optager nogen plads i det oversatte program.

<k-tæller> øger ordretælleren k med summen af de omkring k staaende tal; dette medfører at det tilsvarende antal halvceller nulstilles under indlæsningen. Det er ulovligt at formindske k eller at sætte k til en eller anden talværdi (f.ex. er k:= 100 ikke tilladt). Indlæseprogrammet giver automatisk k en passende begyndelsesværdi, og derefter maa den kun øges.

<t-tæller> øger ganske tilsvarende tromlekanaltælleren t med summen af de omkring t staaende tal, og det medfører at det tilsvarende antal tromlekanaler ikke benyttes til lagring af tromleblokke i programmet. Ogsaa t faar automatisk en begyndelsesværdi (= 100) og maa derefter kun øges.

<etik.def.> definerer en etikette, d.v.s. tildeler den aktuelle identifikator den talværdi, som adressedelen paa højre side giver ved udregning efter de sædvanlige regler. Identifikatorer, som i NL 5-kode kun kan optræde som etiketter samt i adressedele, kan faa tillagt talværdier paa to maader: Enten ved en <etik.def.> eller ved at blive skrevet som etikette foran en ordre (se nedenfor under <ufuldst.kodesætning>).

Virkningen af en <indhop def.> beskrives i afsnit 3.5.

Exempel 3.1 <k-tæller>.

<k-tæller>:	Virkning:
k:= k + 5	5 hac nulstilles under indlæsning.
k:= 7 + k + 1 + 2	10 hac nulstilles under indlæsning.

NB: Man maa selv kode en hopordre for at komme udenom de oversprungne celler, hvis de ligger midt i en beregning.

Exempel 3.2 <t-tæller>.

<t-tæller>:	Virkning:
t:= t + 5	2 tromlekanaler overspringes.
t:= 7 + t + 1 + 2	5 tromlekanaler overspringes.

Exempel 3.3 <etik.def.>.

<etik.def.>:	Virkning:
L1:= 2046	Overalt i den indlæste NL5-kode hvor identifikatoren L1 optræder, erstattes den med tallet 2046.
Etik 2:= AB - USG + 7	Under forudsætning af at identifikatorerne AB og USG defineres andetsteds i programmet (gerne senere), tillægges identifikatoren Etik 2 den talværdi, som højre side giver.

NL 5 protesterer, hvis en identifikator ikke defineres i det indlæste program, eller hvis nogle identifikatorer definerer hinanden i ring.

Af syntaxen ses, at en <halvordsgruppe>, en <helordsgruppe> og en <talgruppe> er helt det samme, nemlig blot en gruppe af tal (i ALGOL-forstand) adskilt med komma. Forskellen ligger i den maade, hvorpaa indlæseprogrammet behandler disse grupper:

r bevirker, at de følgende tal lagres som flydende pakkede tal i overensstemmelse med beskrivelsen i A Manual of the DASK ALGOL Language, afsnit 11.4.2: Real number in store.

f bevirker, at de af de følgende tal, der har et decimalpunkt, lagres som maskintal (DASK-tal) i helceller; tal uden decimalpunkt lagres som heltal med enhed i pos. 39.

h har samme virkning som f, blot sker lagringen i successive halvceller (heltallene med enhed i pos. 19 eller pos. 39).

For baade r, f og h gælder, at hvis kapaciteten sprænges, d.v.s. hvis et tal er for stort (absolut) til at det kan lagres paa den forlangte form, protesterer indlæseprogrammet.

s bevirker at de følgende sedecimale cifre lagres i rækkefølge i konsekutive halvceller med 5 cifre i hver. NL 5 kræver, at antallet af cifre er et multiplum af 5.

De afsluttede enheder ⟨< <tekststreng> ⟩ og ⟨ <layout> ⟩ er ikke definerede i ovenstaaende syntax, men definitionen paa et <layout> kan findes i A Manual of the DASK ALGOL Language, afsnit 8.3.1 . Hvis man benytter DASK ALGOL-procedurerne tryk eller skrv til taludlæsning, skal tallenes typgrafiske opstilling fastlægges ved et <layout>, som ved indlæsningen anbringes i en helcelle, hvorfor det kræves, at k er lige ved indlæsning af et layout (se nedenfor hvordan k med tegnet = kan gøres lige). I det ovennævnte Manual, afsnit 11.4.7, findes en beskrivelse af, hvordan et layout lagres i DASK.

En <tekststreng> er en vilkaarlig symbolfølge, der blot ikke maa omfatte tegnene ⟨ og ⟩ . En saadan tekststreng kan atter udskrives (som en kopi af de indlæste symboler) v.hj.a. DASK ALGOL-procedurerne tryktekst eller skrvtekst. Enhver tekststreng fylder et antal helceller (antallet afhænger af tekststrengens længde), og det kræves at k er lige ved indlæsning af en tekststreng (se nedenfor, hvordan dette opnaas med tegnet =). I Manual, afsnit 11.4.8, findes en beskrivelse af, hvordan en tekststreng lagres i DASK.

3.4 Blokke og programmer.

I nedenstaaende syntax defineres, hvordan ordrer og de øvrige enheder kan sammensættes til blokke og programmer, samt hvordan NL 5-kode kan indgaa i et ALGOL-program og omvendt:

```
<ufuldst. kodesætning>::= <tom> | 0 | <uafsl. enhed>
                        |<afsl. enhed> <kodesætning>
                        |<identifikator>:<ufuldst. kodesætning>
                        |in <identifikator>:<ufuldst. kodesætning>

<stedbest.>           ::= <tom> | = | ≠

<kodesætning>         ::= <stedbest.> <ufuldst. kodesætning> | <kodeblok>
                        |algol;<statementfølge> code

<ordrefølge>          ::= <kodesætning> CR | <kodesætning> CR <ordrefølge>

<kodeblok>            ::= begin <kodehale>
                        |drum <identifikator> begin <kodehale>

<kodehale>            ::= <kodesætning> end | algol;<compound tail>
                        |<kodesætning> CR <kodehale>
```

Desuden udvides syntaxen for ALGOL 60 (i P. Naur m.fl.: Report on the Algorithmic Language ALGOL 60) i afsnit 4.1, der handler om Compound Statements and Blocks, paa følgende punkter:

1. Begrebet <compound statement> udvides, saa definitionen lyder:

```
<compound statement> ::= <unlabelled compound> | <label>:<compound statement>
                       |code CR <ordrefølge> algol
```

2. Begrebet <statementfølge> indføres ved følgende definition:
<statementfølge>::= <statement>;| <statement>; <statementfølge>

3. Begrebet <compound tail> udvides, saa definitionen lyder:
<compound tail> ::= <statement> end | code CR <kodehale>
| <statement>; <compound tail>

Hermed er den syntaktiske beskrivelse af NL 5-kode fuldstændig, idet et program i NL 5-kode er nøjagtig det samme som et program i ALGOL-forstand, nemlig et <compound statement> der ikke henviser til noget udenfor sig selv, og ikke er del af noget <compound statement>. Paa de næste sider beskrives disse ting mere detailleret, og her skal blot nævnes, at et program skrevet helt i NL 5-kode skal indledes med

begin code

og afsluttes med

end

idet indlæseprogrammet NL 5 altid starter klar til indlæsning af et ALGOL-program og derfor straks skal tvinges over i indlæsning af NL 5-kode med symbolet code.

3.5 Betydningen af de forskellige bestanddele.

1) Kodesætning: I almindelighed skal enhederne, ordrerne, adskilles med Carriage Return (der i syntaxen er betegnet med CR) som det fremgaar af definitionen paa <ordrefølge>, og dette tegn har omtrent samme adskillende virkning som semikolon i ALGOL; syntaktisk set svarer en <kodesætning> nemlig nogenlunde til et <statement>.
Som det ses af definitionen paa <ufuldst. kodesætning> er det imidlertid tilladt at anbringe flere ordrer (eller generelt enheder) paa samme linie, hvis blot de udgør afsluttede enheder, d.v.s. at det enten er ordrer med indexmærke eller tekststrenge eller layouts.

Er en kodesætning tom, sætter den sig overhovedet ingen spor i det oversatte program. Det betyder altsaa, at eet CR-tegn har nøjagtig samme virkning som flere CR-tegn efter hinanden.
Bestaar en kodesætning af tallet 0 alene, nulstilles den aktuelle halvcelle, som NL 5 er i færd med at indlæse til.

2) Etiketter: Sættes en etikette, d.v.s. en identifikator efterfulgt af et kolon, foran en kodesætning, tillægges identifikatoren den talværdi som k har i øjeblikket, altsaa adressen paa den halvcelle, eller den første af de halvceller, som kodesætningen anbringes i. Man kan da inden for den samme kodeblok, men ogsaa kun her, benytte denne identifikator i adressedele, og overalt hvor den optræder, vil den blive erstattet med det samme tal. Derimod er identifikatoren udefineret uden for den kodeblok, hvori den staar som etikette. Hvis man har en kodeblok, der udgør en del af en større kodeblok, er det tilladt at benytte den samme identifikator som etikette i den store og i den lille kodeblok. Virkningen vil være at overalt, hvor identifikatoren optræder i de ordrer, der udgør den inderste kodeblok, erstattes identifikatoren med det adressetal, den er defineret ved i den inderste kodeblok. I den yderste (den store) kodeblok erstattes identifikatoren med det adressetal, hvorved den er defineret i denne kodeblok.
Hver ny kodeblok giver saaledes anledning til et nyt niveau, hvor man har frit spillerum m.h.t. valg af etikettebetegnelser. Dette svarer nøje til forholdene ved en <block> i ALGOL, og ligesom i ALGOL gælder der her

følgende udvidelse af reglen ovenfor: Hvis man bruger den samme identifikator i en ydre og en indre kodeblok og identifikatoren kun er defineret een gang i den ydre kodeblok, bliver identifikatoren overalt (i begge niveauer) erstattet med det samme tal.

Bemærk, at til forskel fra ALGOL udgør enhver samling af ordrer omgivet af begin og end en kodeblok, der begrænser rækkevidden af de benyttede etiketter (i ALGOL er en samling af statements omgivet af begin og end først en block, hvis der er en declaration som indledning).

Forstavelsen in foran en etikette har den virkning, at den paagældende etikette faar mening (d.v.s. at identifikatoren faar samme talværdi) ikke blot i den kodeblok, hvori den defineres, men ogsaa i den omgivende kodeblok, altsaa i niveauet uden for den aktuelle kodeblok. Et in har kun virkning for den umiddelbart omgivende blok (enten en kodeblok eller en ALGOL-block), og er kun lovligt hvis den aktuelle blok er en kodeblok.

Denne virkning er den samme, hvad enten in optræder i en kodesætning eller i en <indhop def.> (afsnit 3.2); dog har et in i en <indhop def.> virkning for alle identifikatorerne i etikettelisten (adskilt med kommaer), mens et in foran en etikette til en ordre kun har virkning for den umiddelbart følgende identifikator.

De følgende exempler illustrerer etiketternes rækkevidde og virkningen af in (som ikke har nogen parallel i ALGOL 60, hvor man kun kan komme ind i en block forfra).

Exempel 3.4.

I dette og de følgende exempler staar <ordre> overalt for en vilkaarlig ordre og <statement> tilsvarende for et vilkaarligt statement. Lad nu et program se saaledes ud:

```
begin code
  begin   <ordre>
    in E1:<ordre>
          <ordre> end
  <ordre>
      begin <ordre>
      E2:   <ordre> end
  E3:<ordre> end program;
```

Her er E1 og E3 defineret og maa bruges overalt i programmet, mens E2 kun har en mening inden for den sidste af de smaa kodeblokke.

Exempel 3.5.

Lad et program se saaledes ud:

```
begin <statement>;
  begin <statement>;
  code
  E1:<ordre>
     <ordre> end;
<statement> end program;
```

Her er E1 defineret i hele programmet (da det er den eneste blok, der forekommer).
Hvis programmet i stedet er:

```
begin <statement>;
    begin code
       E1: <ordre>
           <ordre> end;
<statement> end program;
```

vil E1 stadig være defineret i hele programmet, da det inderste begin-end ikke udgør en blok (hverken en kodeblok eller en ALGOL-block).
Hvis programmet derimod hedder:

```
begin <statement>;
code
    begin E1: <ordre>
              <ordre> end
algol; <statement> end program;
```

udgør det inderste en kodeblok, hvorfor E1 kun er defineret herinden-for.

Exempel 3.6

Under indlæsning af følgende program:

```
begin code
<ordre>
begin  <ordre>
 in E1:<ordre>
       <ordre> end
algol;
begin <declaration>;
      <statement>;
code
in  E2:<kodesætning> end
end program;
```

vil NL 5 protestere, fordi in benyttes i en ALGOL-block.

Exempel 3.7

I følgende program (som er syntaktisk korrekt),

```
begin <statement>;
  <statement>;
  code
  begin <ordre>
     E1:<ordre>
        <ordre> end
  algol;
E1: <statement>;
  begin <declaration>;
      E1:<statement>
        <statement> end
  end program;
```

vil E1 faa tillagt tre forskellige værdier afhængig af i hvilken blok den benyttes.

3) Stedbestemmelse: Stedbestemmelsen = medfører, at ordretælleren k gøres lige før indlæsning af den efterfølgende kodesætning: Hvis k allerede er lige, har stedbestemmelsen = ingen virkning, men hvis k er ulige, øges k med 1 og i den oversprungne halvcelle sættes blindordren 130,a.
Stedbestemmelsen ≠ har den ganske analoge virkning, idet ordene lige og ulige ombyttes.

4) Begreberne <ordrefølge> og <kodehale> overdækker tildels hinanden, idet hver af dem i det væsentlige er en række af kodesætninger adskilt med CR, men medens en <ordrefølge> ogsaa afsluttes med et CR-tegn, afsluttes en <kodehale> med symbolet end. Derudover kan en <kodehale> bestaa af ALGOL-statements, der indledes med algol, afsluttes med end og som iøvrigt adskilles med semikolon som sædvanlig.

5) En <kodeblok> kan enten være en ferritlagerblok eller en tromleblok. En ferritlagerblok er en ordrefølge omgivet af begin og end, og disse symboler har kun den virkning at begrænse rækkevidden af etiketter (se ovenfor). Som navnet siger, lagres bestanddelene i ferritlageret. En tromleblok er en kodeblok, hvor der foran det indledende begin staar drum efterfulgt af en identifikator (tromleblokkens navn eller etikette).
NB: En tromleblok skal være del af en kodeblok, d.v.s. den umiddelbart omgivende blok skal være en ferritlager-kodeblok eller en tromle-kodeblok og maa ikke være en ALGOL-block.
For rækkevidden af etiketter gælder helt de samme regler som for kodeblokke, men lagringen i DASK sker efter følgende princip:
Tromleblokkene i et program lagres alle paa tromlen (og der benyttes her et helt antal kanaler til hver), men desuden reserveres der plads i ferritlageret saaledes, at til alle de tromleblokke, der hører ind under een og samme ferritlagerblok, reserveres kun saa meget plads at den længste af tromleblokkene netop kan være der. Det betyder bl.a. at man kun kan benytte een af disse tromleblokke ad gangen og har man brugt een og ønsker en anden fat, maa denne hentes fra tromlen og anbringes paa samme plads som den forrige; dette sker med en af macro-erne (se i operationslisten under TIN). Tromleblokkens etikette har derfor to funktioner: Den kan baade henvise til det sted i ferritlageret, der er reserveret til tromleblokken, og til de tromlekanaler, hvor tromleblokken er lagret. Dette realiseres ved at NL 5 knytter to adresser til den paagældende identifikator, og for alle de tromleblokke, der deler en plads i ferritlageret, er den ene adresse fælles og henviser til den første af de reserverede halvceller i ferritlageret, mens den anden adresse i hver af tromleblokkens etiketter henviser til den paagældende bloks første tromlekanal.
Benyttes nu en tromlebloks etikette som adressedel i en ordre andetsteds i programmet (f.ex. i en hopordre, der skal bevirke et hop til begyndelsen af tromleblokken, bliver dette af NL 5 oversat som et hop til den første af de reserverede celler, og koderen maa selv (ved i forvejen at bruge macroen TIN) sørge for at tromleblokken er kommet paa plads i ferritlageret.
Denne lagring af tromleblokke skal forstaas rekursivt, saaledes at hvis flere tromleblokke er dele af een større tromleblok, vil der i den store tromleblok være reserveret saa meget plads, at den længste af de smaa tromleblokke netop kan være der o.s.v.

4. Sammenknytningen mellem programdele skrevet i ALGOL og i NL 5-kode.

4.1 Programlagring.

Paa grund af den nære forbindelse mellem ALGOL-programmer og NL 5-kode, kan man ikke bestemme selv, hvor i ferritlageret den oversatte kode lagres, men uanset om programmet er en blanding af ALGOL og NL 5-kode eller kun bestaar af en af delene gælder følgende:

Den disponible del af ferritlageret (til program og data) er rundt regnet de sidste 1950 halvceller, idet knap 100 halvceller fra celle 0 og fremad benyttes som arbejdsceller for sekvenslageret (arbejdscellerne er celle 0-63) samt til administration af NL 5. Programmet lagres altid forfra, d.v.s. omtrent fra celle 90, og fremad.

Benyttes ALGOL er den sidste del af ferritlageret (fra celle 2047 og bagud) reserveret til den saakaldte stak. Hvor meget plads stakken optager afhænger af programmet, men i det væsentlige benyttes stakken til lagring af alle arrays samt til lagring af mellemresultater i de aritmetiske beregninger. Da indexregister B under kørsel af ALGOL-programmer benyttes som stak-viser (d.v.s. B altid indeholder adressen paa den sidste benyttede celle i stakken), stilles følgende krav til NL 5-kode der benyttes sammen med ALGOL:

Hvis indexregister B bruges, skal det retableres.

Paa tromlen er den disponible del kanalerne 100-510, og her lagres eventuelle tromleblokke fra kanal 100 og fremad (i den rækkefølge hvori de optræder i programmet); hver blok optager et helt antal kanaler, men den plads der ikke bruges til tromleblokke er disponibel for datalagring.

4.2 Fælles benyttelse af identifikationer.

I det følgende forudsættes overalt, at de omtalte identifikatorer er defineret paa de steder hvor de benyttes (smlg. afsnit 3.5). Det bør ogsaa nævnes, at en identifikator i NL 5-syntaxen er helt det samme som en <identifier> i DASK ALGOL, d.v.s. et vilkaarligt antal bogstaver og cifre (hvoraf det første symbol skal være et bogstav), men kun de første seks tegn er informationsbærende. Det betyder at to identifikatorer, der har de seks første tegn ens, opfattes og behandles som den samme identifikator af NL 5.

En identifikator kan defineres paa tre forskellige maader:

1) Ved en erklæring (declaration) i ALGOL,
2) ved en etikettedefination i NL 5-kode, eller
3) ved at blive brugt som etikette i NL 5-kode.

Ser vi først paa variable, der er erklæret i ALGOL, gælder der følgende regler:

1) For simple variable, der er erklæret i ALGOL, gælder det, at bruges en saadan identifikator i NL 5-kode, bliver den (under oversættelsen) erstattet med adressen paa den celle, der indeholder den variable.

Exempel 4.1.

Forudsat, at n, m1 og m2 er erklæret som heltal, vil følgende ALGOL-statement

```
n:= m1 + m2
```

og NL 5-ordrerne

```
60  m1
20  m2
28  n
```

udføre nøjagtig det samme.

For indicerede variable maa det nødvendigvis forholde sig lidt anderledes: Lad et array arr[i,j] være erklæret i ALGOL-programmet. Benyttes identifikatoren arr nu i en NL 5-ordre, bliver den oversat ved adressen paa en intern array-identifikator der fylder en helcelle; i denne helcelle staar der to ordrer, hvor den førstes adressetal er adressen paa den første komponent i arrayet, mens den anden ordres adressetal fortæller hvor indexgrænserne kan findes (se iøvrigt A Manual of the DASK ALGOL Language, afsnit 11.7.2.3, hvor en mere detailleret beskrivelse findes).

Exempel 4.2.

Lad et array være defineret paa ALGOL-vis ved:

```
real array arr [1: 10, 1: 10];
```

da vil NL 5-koden

```
35  2
60  arr
29  k + 1
40  0,b
```

bevirke, at komponenten arr[1,2] anbringes i AR, idet dette array er lagret i rækkefølen

arr[1,1], arr[1,2],..., arr[1,10], arr[2,1],..., arr[10,10],

og der i ovenstaaende kode lægges 2 til begyndelsesadressen v.hj.a. indexregister B.

Bruges en procedure-identifikator i NL 5-kode, oversættes den ved adressen paa den første af de halvceller, som proceduren optager (det er den adresse der skal hoppes til ved et procedure-kald). Skal dette benyttes for procedurer med formelle parametre, maa koderen sørge for, at disse parametre er anbragt korrekt som beskrevet i A Manual of the DASK ALGOL Language, afsnit 11.5.4.

De i dette Manuals afsnit 11.5.4.1 nævnte identifikatorer for standard procedurer er ogsaa i NL 5-kode reserveret, og de oversættes med de i Manual opførte adresser.

Exempel 4.2.

Forudsat, at x er erklæret som reel variabel, vil følgende ALGOL-expression

sin (x)

og NL 5-ordrerne

17 sin
40 x,c

være ækvivalente (smlgn. Manual, afsnit 11.5.4.2).

Bruges en switch-identifikator i NL 5-kode, oversættes den ved adressen paa den første af de halvceller, som switch-declarationen optager (se iøvrigt Manual afsnit 11.7.3 og 11.6.4).

Bruges en label i NL 5-kode, oversættes den ved den samme adresse som i ALGOL-programmet.

NB: Det er forbudt med en hopordre (i NL 5-kode) at hoppe ud af en ALGOL-block. Det er saaledes ikke ethvert go to-statement, der direkte kan erstattes af en hopordre.

Exempel 4.3.

Forudsat at E7 er en label inden for den aktuelle ALGOL-block (E7 er en lokal label), har et ALGOL-statement

go to E7

og NL 5-ordren

10 E7

samme virkning.

2) og 3). Er en identifikator defineret i NL 5-kode, maa den i ALGOL kun bruges som et procedure statement, og den vil, hvor den optræder i ALGOL, blive oversat ved et sekvenshop, en 16-ordre, (til den adresse, der definerer identifikatoren) efterfulgt af ordrer til behandling af eventuelle parametre.

Exempel 4.4.

Er etiketten Vanvid defineret i NL 5-kode (enten ved en etikettedefinition eller ved at Vanvid staar som etikette til en kodesætning), vil et ALGOL-statement

Vanvid

i det oversatte ALGOL-program være erstattet af hopordren

16 Vanvid

og Vanvid maa i ALGOL-programmet kun benyttes som et statement (den kan ikke indgaa i aritmetiske expressions og lignende).

4.3 Symbolerne algol og code.

Symbolerne algol og code har for indlæseprogrammet nærmest karakter af en switch, som bringer det fra at indlæse ALGOL til at indlæse kode og omvendt. Konventionerne er iøvrigt følgende:

Naar man starter indlæsningen af et program, er NL 5 altid klar til at indlæse ALGOL, og NL 5 fortsætter med at indlæse ALGOL indtil det møder symbolet code. Derefter indlæses NL 5-kode indtil enten 1) symbolet algol optræder, eller 2) det compound statement, hvori symbolet code optraadte, afsluttes. Derefter fortsættes med indlæsning af ALGOL, og dette afsluttes paa de samme to maader som kode-indlæsning.

Generelt formuleret dirigeres indlæsningstypen altsaa af de to symboler algol og code med følgende tilføjelse:1) Et end bevirker overgang til den indlæsningstype, der raadede ved det tilsvarende begin. 2) Ved begyndelsen af et program er indlæsningstypen altid ALGOL.

Regel 1 betyder, at ved udgangen af en blok eller et compound statement er indlæsningstypen altid den samme som ved den tilsvarende indgang.

Exempel 4.3.

I de følgende exempler betegner <statement> og <kodesætning> vilkaarlige ALGOL-statements og kodesætninger, der hver for sig gerne kan indeholde underblokke, baade kodeblokke og ALGOL-blocks.

Lad et program bestaa af

```
begin <statement>;
      <statement>;
   begin <declaration>;
         <statement>;
         code
         <kodesætning>
         <kodesætning> end;
      <statement>;
      <statement> end program;
```

Efter det første end bliver indlæsningstypen ALGOL, fordi det er en blokafslutning. Ønsker man at fortsætte med kode-indlæsning, maa man efter end (og efter semikolon) skrive et nyt code.

I det følgende program er det først det andet end efter code, der er afslutning paa det compound statement, hvori code staar, og kodeindlæsningen fortsætter derfor til næste end:

```
begin <statement>;
      <statement>;
   begin <statement>;
      code
      begin <kodesætning> end
            <kodesætning> end ;
      <statement> end program;
```

Exempel 4.4.

Da et ALGOL-statement bl.a. kan bestaa af en ordrefølge omgivet af code og algol, er følgende konstruktion tilladt i et ALGOL-program:

```
if <Boolean expression> then
   code
   <kodesætning>
   <kodesætning>
   algol else <statement>;
```

Omvendt kan en kodesætning bl.a. bestaa af en samling af statements omgivet af algol; og code, hvorfor følgende konstruktion kan forekomme i NL 5-kode:

```
<kodesætning>
algol;
<statement>;
<statement>; code
<kodesætning>
```

5. Kommentarer.

I ALGOL-programmer kan kommentarer anbringes som nævnt i Report on the Algorithmic Language ALGOL 60, afsnit 2.3.

I NL 5-kode kan kommentarer anbringes i det væsentlige paa to forskellige maader:

1) Efter et semikolon som afslutning paa en linie, idet kombinationen

; <vilkaarlig symbolfølge uden CR > CR

er ækvivalent med CR, og alt hvad der staar mellem semikolon og CR negligeres af NL 5. Kommentarer af denne art maa naturligvis kun anbringes paa de steder i koden hvor et CR-tegn er syntaktisk korrekt.

2) En venstreparentes, rund eller kantet, paa et vilkaarligt sted i NL 5-kode bevirker, at det følgende indtil den tilsvarende højreparentes negligeres af NL 5. Ved udtrykket den tilsvarende højreparentes forstaas for det første en højreparentes af samme type som den indledende venstreparentes, men desuden holder NL 5 regnskab med hvormange venstreparenteser og hvormange højreparenteser af samme type der er indlæst, og først naar der er indlæst lige mange, opfatter NL 5 kommentaren som afsluttet.

Exempel 5.1.

Det følgende er 4 exempler paa kommentarer i NL 5-kode, som vil blive negligeret under indlæsning:

```
; dette er kommentar til den usynlige ordre 13 E1,a.
[ runde halvparenteser ()( gør ingen skade]
[ en hel, kantet parentes [i,j] gør ingen skade]
( tilsvarende gør her kantede halv-parenteser ]][ og hele,
  runde parenteser ((NB)) ingen skade).
```

6. Operationsliste.

6.1 Betegnelser.

I operationslisten nedenfor findes alle de operationer, der kan bruges i NL 5-kode. I afsnit 6.3 beskrives de sedecimale operationsdele og mnemonics, og i afsnit 6.4 følger saa beskrivelsen af de 17 macro-er. Overalt hvor der i operationsbetegnelsen indgaar bogstaver, kan man i flæng skrive store og smaa bogstaver, og cifferet 0 og bogstaverne o og O skelnes der heller ikke imellem (dette gælder kun operationsdele og ikke adressedele).

Virkningen af hver operation beskrives dels i ALGOL (hvor det ikke er meget besværligt) og dels i text. Hvor texten maatte synes lidt kortfattet, henvises til Lærebog i kodning for DASK.

Vi har forsøgt at vælge betegnelser, hvis betydning skulle være umiddelbart forstaalig, men iøvrigt er i hvert fald følgende betegnelser benyttet:

AR	:	Akkumulatorregisteret eller dettes indhold opfattet som maskintal.
ARv,ARh	:	Akkumulatorregisterets venstre (højre) halvdel eller dennes indhold opfattet som maskintal.
ARpos00	:	Indholdet af pos. 00 i akkumulatorregisteret.
AR[j]	:	Indholdet af pos. j i akkumulatorregisteret.
ARvadr, ARhadr, ARvoper og ARhoper:		Indholdene af adresse- og operationsdelene i AR, opfattet som heltal.
Tilsvarende betegnelser bruges for multiplikatorregisteret M.		
ARMR	:	Indholdet af det lange akkumulatorregister, der bestaar af AR samt pos. 1-39 i M.
hec[m]	:	Indholdet af helcelle m, opfattet som maskintal.
hac[m]	:	Indholdet af halvcelle m, opfattet som maskintal.
pos[j,m]	:	Indholdet af pos. j i celle m.
adr[m], oper[m]	:	Indholdet af adresse-og operationsdelene i halvcelle m.
m	:	Den resulterende adresse, d.v.s. summen af adressetallet og indholdet af et indexregister (evt. 0).
B,C,D	:	Indholdet af indexregistrene, opfattet som heltal.
AS	:	Adresseregisterets indhold.
OP	:	Operationsregisterets indhold.
KR	:	Ordretællerens indhold.

6.2 Adresseberegning.

Udførelsen af den enkelte ordre indledes med at ordretælleren øges med 1, hvorefter den resulterende adresse m beregnes udfra adressetallet og indexmærket i den aktuelle ordre:

```
    AS := KR + 1;
HOP:KR := AS;
  L:OP := oper[AS];
     m := AS := adr[AS] + (if indexmærke[AS]=b then B else
                           if indexmærke[AS]=c then C else
                           if indexmærke[AS]=d then D else
                           0);
```

Derefter udføres selve operationen i den aktuelle ordre i overensstemmelse med operationslisten paa de følgende sider.

6.3 Sedecimale operationer og mnemonics.

Til hver mnemonic findes en sedecimal operation med helt samme virkning, og de er derfor beskrevet under eet i operationslisten. De enkelte steder, hvor en sedecimal operation staar alene, findes der ikke nogen mnemonic med samme virkning.

00 A+F	m lige: AR := AR + hac[m]; hac[m] adderes til AR; resultatet anbringes i AR. m ulige: AR := AR + (hac[m] + if hac[m]<0 then 2 else 0)× 2↑(-20); hhac[m] adderes til højre del af AR; resultatet anbringes i AR.
40 0+F	AR nulstilles; derefter som 00.
20 A+H	ARv := ARv + hac[m]; hac[m] adderes til AR og resultatet anbringes i AR. ARh er uændret.
60 0+H	AR nulstilles. Derefter som 20.
01 A-F	m lige: AR := AR - hac[m]; hac[m] subtraheres fra AR og resultatet anbringes i AR. m ulige: AR := AR - (hac[m]+if hac[m]<0 then 2 else 0)× 2↑(-20); hhac[m] subtraheres fra højre del af AR og resultatet anbringes i AR.
41 0-F	AR nulstilles. Derefter som 01.
21 A-H	ARv := ARv - hac[m]; hac[m] subtraheres fra AR; resultatet anbringes i AR. ARh er uændret.
61 0-H	AR nulstilles. Derefter som 21.

Absolut addition og subtraktion.

02 APF	m lige: AR := AR + abs(hac[m]); |hac[m]| adderes til AR; resultatet anbringes i AR. m ulige: AR := AR + (hac[m] + if hac[m]<0 then 2 else 0)× 2(-20); hhac[m] adderes til højre del af AR; resultatet anbringes i AR.
42 OPF	AR nulstilles. Derefter som 02.
22 APH	ARv := ARv + abs(hac[m]) ; |hac[m]| adderes til AR; resultatet anbringes i AR. ARh er uændret.
62 OPH	AR nulstilles. Derefter som 22.
03 ANF	m lige: AR := AR - abs(hac[m]); |hac[m]| subtraheres fra AR; resultatet anbringes i AR. m ulige: AR := AR -(hac[m] + if hac[m]<0 then 2 else 0)× 2↑(-20); hhac[m] subtraheres fra højre del af AR; resultatet anbringes i AR.
43 ONF	AR nulstilles. Derefter som 03.
23 ANH	ARv := ARv - abs(hac[m]); |hac[m]| subtraheres fra AR; resultatet anbringes i AR. ARh er uændret.
63 ONH	AR nulstilles. Derefter som 23.

Addition og subtraktion til MR.

04 M+F	n lige: MR := AR := AR + hac[n]; hac[n] adderes til AR; resultatet anbringes i AR og i MR. n ulige: MR := AR := AR +(hac[n] + if hac[n]<0 then 2 else 0)× 2↑(-20); hhac[n] adderes til højre del af AR; resultatet anbringes i AR og i MR.
44 F>M	AR nulstilles. Derefter som 04.
24 M+H	MR := AR := AR + hac[n]; hac[n] adderes til AR; resultatet anbringes i AR og i MR. ARh er uændret.
64 H>M	AR nulstilles. Derefter som 24.
05 M-F	n lige: MR := AR := AR - hac[n]; hac[n] subtraheres fra AR; resultatet anbringes i AR og i MR. n ulige: MR := AR := AR -(hac[n] + if hac[n]<0 then 2 else 0)× 2↑(-20); hhac[n] subtraheres fra højre del af AR; resultatet anbringes i AR og i MR.
45 -FM	AR nulstilles. Derefter som 05.
25 M-H	MR := AR := AR - hac[n]; hac[n] subtraheres fra AR; resultatet anbringes i AR og i MR. ARh er uændret.
65 -HM	AR nulstilles. Derefter som 25.

Addition til celle.

06
F+A

m lige: hec[m] := AR := AR + hec[m];
hec[m] adderes til AR, resultatet anbringes i AR og i hec[m].
m ulige: AR := AR + (hac[m] + if hac[m]<0 then 2 else 0)× 2↑(-20) ; hac[m] := ARh;
hhac[m] adderes til højre del af AR; resultatet anbringes i hhac m.

26
H+A

hac[m] := ARv := ARv + hac[m];
hac[m] adderes til ARv, resultatet anbringes i ARv og i hac m. ARh er uændret.

46
F+2

m lige: hec[m] := AR := hec[m] + 2↑(-10) + 2↑(-30);
adressetallene i begge halvord i hec m øges med 2; AR faar samme indh old som hec m.
m ulige: hac[m] := ARh := hac[m] + 2↑(-10); ARv := 0;
Adressetallet i hhac m øges med 2. ARh faar samme indhold som hac m, og ARv nulstilles.

66
H+2

hac[m] := ARv := hac[m] + 2↑(-10) ; ARh := 0;
Adressetallet i hac m øges med 2. ARv faar samme indhold som hac m, og ARh nulstilles.

Gem MR. Gem AR. Logisk Produkt.

07 el. 27 M>A	AR := MR; ARpos00 := C; MRs indhold anbringes i AR; ARoo nulstilles.
08 A>F	m lige: hac[m] := AR; AR anbringes i hac m. m ulige: hac[m] := ARh; ARh anbringes i hhac m.
48 0>F	AR nulstilles. Derefter som 08.
28 A>H	hac[m] := ARv; ARv anbringes i hac m.
68 0>H	AR nulstilles. Derefter som 28.

09
P>F
	m lige: adr[m] := ARvadr; pos[0,m] := AR[0];
	adr[m+1] := ARhadr; pos[20,m] := AR[20];
	AR_{0-11} og AR_{20-31} anbringes i de samme positioner i hac m.
	m ulige: adr[m] := ARhadr; pos[0,m] := AR[20];
	AR_{20-31} anbringes i de tilsvarende positioner i hhac m.

49 0FP	AR nulstilles. Derefter som 09.
29 P>H	adr[m] := ARvadr; pos[0,m] := AR[0]; AR_{0-11} anbringes i de tilsvarende positioner i hac m.
69 0HP	AR nulstilles. Derefter som 29.
47 el. 67 M∧A	for j := 0 step 1 until 39 do AR[j] := AR[j]∧MR[j]; ARpos00 := 0; Det logiske produkt af AR-s bits og MR-s bits anbringes i AR. AR_{oo} nulstilles.

Multiplikation.

0A M × F

m lige: AR := afkort (MR×hac[m] + 2↓(-40));
hac[m] multipliceres med MR, og det afrundede resultat anbringes i AR.

m ulige: AR := afkort (MR ×(hac[m] + if hac[m]<0 then 2 else 0) × 2↓(-20) + 2↓(-40)) ;
hhac[m] × 2^{-20} (omtrent) multipliceres med MR, og det afrundede resultat anbringes i AR.

2A M × H

AR := afkort (MR × hac[m] + 2↓(-40));
hac[m] multipliceres med MR, og det afrundede resultat anbringes i AR.

4A L × F

m lige: ARMR := MR × hac[m]; MR[0] := 0;
hac m multipliceres med MR, og resultatet anbringes i AR samt MR_{1-39}, mens MR_0 nulstilles.

m ulige: ARMR := MR ×(hac[m] + if hac[m]<0 then 2 else 0) × 2↓(-20); MR[0] := 0;
hhac[m] × 2^{-20} (omtrent) multipliceres med MR, og resultatet anbringes i AR, MR_{1-39}, mens MR_o nulstilles.

6A L × H

ARMR := MR × hac[m]; MR[0] := 0;
hac[m] multipliceres med MR, og resultatet anbringes i AR, MR_{1-39}, mens MR_o nulstilles.

Division.

KRAV: |dividend| $\leq$ | divisor |.

0B
A/F

m lige: MR := AR/hec[m]; AR := rest × 2↑39;
AR divideres med hec[m]. Resultatet anbringes i MR, og divisionsresten × 2^{39} i AR.

m ulige: MR := AR/(hac[m] + if hac[m]<0 then 2 else 0) × 2↑-20; AR := rest × 2↑39;
AR divideres med hhac[m] × 2^{-20} (omtrent). Resultatet anbringes i MR, og divisionsresten × 2^{39} i AR.

2B
A/H

MR := AR/hac[m]; AR := rest × 2↑39.
AR divideres med hac[m]. Resultatet anbringes i MR, og divisionsresten × 2^{39} i AR.

4B
L/F

m lige: MR := ARMR/hec[m]; AR := rest × 2↑39;
AR,MR_{1-39} divideres med hec[m]. Resultatet anbringes i MR og divisionsresten × 2^{39} i AR.

m ulige: AR,MR_{1-39} divideres med hhac[m] × 2^{-20} (omtrent). Resultatet anbringes i MR og divisionsresten × 2^{39} i AR.

6B
L/H

MR := ARMR/hac[m]; AR := rest × 2↑39;
AR,MR_{1-39} divideres med hac[m].
Resultatet anbringes i MR og divisionsresten × 2^{39} i AR.

Vedrørende den nøjere sammenhæng mellem dividend, divisor og kvotient henvises iøvrigt til Lærebog i Kodning for DASK. Specielt bemærkes, at p.g.a. divisionsprocessens forløb i DASK faar enhver kvotient et 1-tal som sidste (39-te) bit; det bevirker bl.a. at hvis divisionen undtagelsesvis "gaar op" inden for de 39 cifre, vil DASK ikke altid give den korrekte kvotient.

Skift, normalisering.

0C el. 2C VSK	AR := AR × 2↑(m - 128×entier(m/128)); comment AR inklusive ARpos00; AR_{00-39} skiftes m positioner til venstre. I AR_{39} indskiftes nuller.
4C el. 6C VSL	ARMR := ARMR × 2↑(m - 128×entier(m/128)); comment inklusive ARpos00; AR_{00-39}, MR_{1-39} skiftes m positioner til venstre. I MR_{39} indskiftes nuller. MR_0 nulstilles.
0D el. 2D HSK	AR := afkort(AR × 2↑(-m + 128×entier(m/128))); ARpos00 := 0; AR_{0-39} skiftes m positioner til højre. I AR_0 indskiftes oprindeligt fortegn. AR_{00} nulstilles.
4D el. 6D HSL	ARMR := ARMR × 2↑(-m+ 128×entier(m/128)); ARpos00 := MR[0] := 0; AR_{0-39}, MR_{1-39} skiftes m positioner til højre. I AR_0 indskiftes oprindeligt fortegn. AR_{00} og MR_0 nulstilles.
0F el. 2F LHK	AR_{0-39} skiftes m positioner til højre. I AR_0 indskiftes nuller. AR_{00} = 0.
4F el. 6F LHL	AR_{0-39}, MR_{1-39} skiftes m positioner til højre. I AR_0 indskiftes nuller. AR_{00} = MR_0 = 0.
0E el. 2E NOK	AR_{00-39} skiftes til venstre, indtil indholdet er normaliseret (der udføres 40 skift hvis AR = 0). I AR_{39} indskiftes nuller. Antallet af skift placeres i adressepositionerne 0-11 (eller 20-31) i hac m.
4E el. 6E NOL	AR_{00-39}, MR_{1-39} normaliseres som ved 0E (der udføres 80 skift, hvis indholdet er nul). Antallet af skift placeres i adressepositionerne 0-11 (eller 20-31) i hac m.

Hop.

10 HOP	go to Hop; comment: Hop er adressen paa den anden sætning i adresseberegningen Næste ordre hentes i hac m.
30 SHP	stop; go to Hop; Stop. Ved start hentes næste ordre i hac m.
50 OPH	AR := 0; go to Hop; AR nulstilles. Derefter som 10.
70 OSH	AR := 0; stop; go to Hop; AR nulstilles. Derefter som 30.
11 HP+	if AR $\geq$ 0 then go to Hop; AR $\geq$ 0: Næste ordre hentes i hac m. AR < 0: DASK fortsætter uanfægtet.
31 SH+	Stop. Ved start samme virkning som 11.
51 HP-	if AR <0 then go to Hop; AR $\geq$ 0: DASK fortsætter uanfægtet. AR < 0: Næste ordre hentes i hac m.
71 SH-	Stop. Ved start samme virkning som 51.
12 HPS	Ved spild: Næste ordre hentes i hac m. Ellers fortsættes normalt.
32 SHS	Stop. Ved start samme virkning som 12.
52 HPI	Ved spild: AR_{00-39} skiftes 1 position til højre; AR_{00} = 0; Derefter fortsættes normalt. Ved ikke spild: Næste ordre hentes i hac m.
72 SHI	Stop. Ved start samme virkning som 52.

Index-operationer.

13 TOM	Blind operation.
33 HPB	if B ≠ 0 then go to Hop; B ≠ 0: Næste ordre hentes i hac m. Ellers fortsættes uanfægtet.
53 HPC	if C ≠ 0 then go to Hop; C ≠ 0: Næste ordre hentes i hac m. Ellers fortsættes uanfægtet.
73 HPD	if D ≠ 0 then go to Hop; D ≠ 0: Næste ordre hentes i hac m. Ellers fortsættes uanfægtet.
14 0>P	adr[m] := 0; Pos. 1-11 eller 21-31 i hac. m nulstilles.
34 B>P	adr[m] := B; B anbringes i pos. 1-11 (eller 21-31) i hac m.
54 C>P	adr[m] := C; C anbringes i pos. 1-11 (eller 21-31) i hac m.
74 D>P	adr[m] := D; D anbringes i pos. 1-11 (eller 21-31) i hac m.
35 N>B	B := m; m anbringes i B.
55 N>C	C := m; m anbringes i C.
75 N>D	D := m; m anbringes i D.

Specielle hop.

16 SEK	D := KR; go to Hop; C(KR) anbringes i D. Derefter hentes næste ordre i hac m.
36 SSE	D := 0; stop; D := KR; go to Hop; D nulstilles, hvorefter DASK stopper. Ved start samme virkning som 16.
56 BSE	if hop 56 aktiv then begin D := KR; go to Hop end; Hvis omskifteren "56-hop" paa kontrolbordet er i aktivstilling, er virkningen som ved 16; ellers fortsætter DASK normalt.
76 SBS	if hop 56 aktiv then begin D := 0; stop; D := KR; go to Hop end else stop; Hvis omskifteren er i aktiv-stilling, er virkningen som ved 36. Ellers stopper DASK blot og fortsætter normalt ved start.
17 SKL	D := KR; go to Hop; comment: KR henviser til Sekvenslageret; KR anbringes i D. Derefter hentes næste ordre i hac m i sekvenslageret.
57 OSK	AR nulstilles. Derefter som 17

Udfør-ordrer.

37
UDF

go to L; comment L er adressen paa 3. sætning i adresseberegningen (se side 17);
Ordren i hac m udføres, men KR er i reglen uændret herefter, hvilket betyder, at DASK fortsætter med ordren efter 37-ordren. Der gælder følgende undtagelser.
hac m indeholder en hopordre:
Hoppet udføres normalt, og DASK fortsætter med ordrerne i den celle, som hopordrens adresse angiver, og de følgende celler. Dette gælder ogsaa hvis hac m indeholder en betinget hopordre, hvor betingelsen er opfyldt; ellers fortsættes med ordren efter 37 -ordren.
hac m indeholder en 16-ordre eller en variant heraf:
Hoppet udføres som beskrevet under 16-ordren, men D indeholder derefter adressen paa 37-ordren.
hac m indeholder en 37-ordre:
DASK udfører ordren i den halvcelle som adresse[m] henviser til, og fortsætter derefter med ordren efter den første 37-ordre.

77
OUD

AR nulstilles. Derefter som 37.

Gem adresse. Gem operation.

18
T>F

m lige: hec[m] := AR; oper[m+1] := 0;
AR_{0-31} anbringes i de tilsvarende positioner i hec m, mens pos.32-39 i hec m nulstilles.

m ulige: adr[m] := ARhadr; oper[m] := 0;
AR_{20-31} anbringes i de tilsvarende positioner i hhac m, mens dennes pos. 32-39 nulstilles.

38

m lige: vhac[m] := ARv;
ARv anbringes i vhac m (som ved 28-operationen).

m ulige: adr[m] := ARvadr; oper[m] := 0;
AR_{0-11} anbringes i pos. 32-39 nulstilles.

58
X>F

m lige: adr[m] := adr[m+1] := 0;
oper[m]:=ARvoper; oper[m+1] := ARhoper;
De to operationsdele i AR, AR_{12-19} og AR_{32-39}, anbringes i de tilsvarende positioner i hec m, mens dennes adressedele, pos. 0-11 og pos. 20-31, nulstilles.

m ulige: adr[m] := 0; oper[m] := ARhoper;
I hhac m nulstilles adressedelen, mens AR_{32-39} anbringes i de tilsvarende positioner i halvcellen.

78
X>H

adr[m] := 0; oper[m] := ARvoper;
adressedelen, pos.0-11 (eller 20-31), i hac m nulstilles, mens AR_{12-19} anbringes i operationsdelen i hac m.
Ved disse operationer bruges adr for indholdet i pos. 0-11 eller 20-31.

Indlæsning.

19
INF

m lige: hec[m] := AR := strimmelhelord;
Fra 5-hulstrimmel indlæses 10 tegn (sedecimale cifre), og anbringes i hec m og i AR i den orden, hvori de indlæses (med 4 bits/tegn).

m uligc: AR := strimmelhelord; hac[m] := ARh;
Fra 5-hulstrimmel indlæses 10 tegn til AR som ovenfor, derefter anbringes ARh i hhac[m].
Omskifteren paa 5-hulstrimmellæseren skal staa paa 4-kanal-indlæsning.

39
INH

Bør ikke benyttes, da den er reserveret til eventuelle udvidelser af indlæseapparatur. Dens virkning er i øjeblikket som beskrevet i Lærebog i kodning for DASK.

59
IN5

Først nulstilles AR. Derefter indlæses et tegn fra 5-hulstrimmel til den bagerste del af AR; virkningen afhænger af omskifteren paa 5-hulstrimmellæseren, der kan staa paa 4-kanal- eller 5-kanal-indlæsning:

	4-kanal:	5-kanal:
strimmel	o o . o o o	o o . o o o
AR pos.	36 37 38 39	36 37 38 39 35

79
IN8

Først nulstilles AR. Derefter indlæses et tegn fra 8-hulstrimmel til den bagerste del af AR; virkningen afhænger af hvilken sko, der er paasat 8-hulstrimmellæseren, men med den (normale) sko, som er mærket >>8-hul-indlæsning med paritetsfejl i AR0.<<, sker indlæsningen saaledes:

strimmel	o	o	o	o	o	.	o	o	o
AR pos.	33	34	35	36		37	38	39	

Hvis antallet af huller er ulige sker der ikke mere, men er antallet lige, etstilles AR[0] (dette kaldes paritetsfejl).

Udlæsning.

Valget af udskrift-medium styres dels af to omskiftere paa kontrolbordet og dels af ordrevalget (grundoperation eller variant). For omskifterne gælder: 1) Staar omskifterpilen paa P eller S faas al udskrift paa henholdsvis perforator eller skrivemaskine. Staar omskifterpilen paa stilling -, faas ingen udskrift (udlæsordrer virker som blindordrer). Staar omskifteren i stilling S + P, faas en del af udskriften paa begge medier (se iøvrigt Lærebog i kodning for DASK). I stilling 0 giver grundoperationerne udskrift paa skrivemaskine og varianterne udskrift paa perforator. 2) A-knappen har tre stillinger: S, P og -. Det medium, som A-knappen peger paa, erstattes af ANELEX-linieskriveren. Beskrivelsen nedenfor dækker kun det tilfælde, hvor omskifterpilen staar i stilling 0 og A-knappen paa -:

1A SA5

AR_{36-39} trykkes som et tegn paa 5-kanal-skrivemaskinen.

3A TA5

AR_{36-39} stanses i de øverste 4 positioner af et tegn paa 5-hulstrimmel (den 5. position stanses altid).

5A SA8

Forudsat at omskifteren paa kontrolbordet staar paa 7 sker følgende:

AR_{33-39} trykkes som et tegn paa 8-kanal-skrivemaskinen.

7A TA8

Forudsat at omskifteren paa kontrolbordet staar paa 7 (det normale) sker følgende:

AR_{33-39} stanses som et tegn paa 8-hulstrimlen (og eventuelt paritetsmærke stanses automatisk).

Staar omskifteren paa 8, stanses AR_{32-39} som et tegn paa 8-hulstrimlen.

Angaaende sammenhængen mellem AR-s indhold og det udskrevne tegn, henvises til oversigten nedenfor.

Udlæsning.

1B SN5	Paa 5-kanal-skrivemaskinen udskrives et tegn bestemt af den resulterende adresse m (modulo 16).
3B TN5	Paa 5-hulstrimlen stanses den resulterende adresse (beregnet modulo 16) i de 4 øverste positioner.
5B SN8	Forudsat at omskifteren paa kontrolbordet staar paa 7 sker følgende: Den resulterende adresse m (modulo 128) trykkes som et tegn paa 8-kanal-skrivemaskinen.
7B TN8	Forudsat at omskifteren paa kontrolbordet staar paa 7 (det normale) sker følgende: Den resulterende adresse m (modulo 128) stanses som et tegn paa 8-hulstrimlen (og eventuelt paritetsmærke stanses automatisk). Staar omskifteren paa 8, stanses m modulo 256 som et tegn paa 8-hulstrimlen.

Angaaende sammenhængen mellem m og det udskrevne tegn, henvises til oversigten nedenfor.

I de to skemaer er angivet hvilke tegn der udskrives paa 5-kanal-apparaturet med 1A- og med 1B-operationer (og disses varianter 3A og 3B), samt paa 8-kanal-apparaturet med 5A- og 5B-operationer (og varianterne 7A og 7B). De to midterste kolonner i hvert skema angiver, hvad der trykkes (eller stanses), naar apparaturet er i henholdsvis Lower Case LC og Upper Case UC; de to kolonner til højre i hvert skema angiver paa hvilke punkter ANELEX-ens funktion afviger fra flexowriternes og skrivemaskinernes funktioner.

5-kanal apparatur

AR[36-39]	1A-operation i LC	i UC	Afvig. ANELEX i LC	i UC	m mod. 16	1B-operation i LC	i UC	Afvig. ANELEX i LC	i UC
0	P	0			0	ingen			
1	Q	1			1	Car. return			
2	W	2			2	mellemrum			
3	"	3			3	tabul.			
4	R	4			4	:	-		
5	T	5			5	Z	+		
6	Y	6			6	=	.		ingen
7	U	7			7	M	¯		
8	I	8			8	V	±		
9	O	9			9	L	)		
10	a	Λ	A		10	K	(		
11	b	B	B		11	G	X		
12	c	C	C		12	H	S		
13	d	D	D		13	sæt LC			
14	e	E	E		14	sæt UC			
15	N	F			15	J	stop		

8-kanal-apparatur

AR[33-39] eller m mod. 128	5A- el. 5B-oper. i LC	i UC	Afvig. ANELEX i LC	i UC	AR[33-39] eller m mod. 128	5A- el. 5B-oper. i LC	i UC	Afvig. ANELEX i LC	i UC
0	mellemrum				33	j	J	J	
1	1	∨		£	34	k	K	K	
2	2	×			35	l	L	L	
3	3	/			36	m	M	M	
4	4	=			37	n	N	N	
5	5	;			38	o	O	O	
6	6	[			39	p	P	P	
7	7	]			40	q	Q	Q	
8	8	(			41	r	R	R	
9	9	)			42	ingen		formularsk.	
10	ingen		$	±	43	ø	Ø	Ø	
11	stop				44	Punch On		ingen	
13	ingen		≠	'	45	ingen		mellemrum	
14		|	ingen		46	ingen		mellemrum	
15	ingen		%	&	47	ingen		mellemrum	
16	0	^		↑	48	æ	Æ	Æ	
17	<	>			49	a	A	A	
18	s	S	S		50	b	B	B	
19	t	T	T		51	c	C	C	
20	u	U	U		52	d	D	D	
21	v	V	V		53	e	E	E	
22	w	W	W		54	f	F	F	
23	x	X	X		55	g	G	G	
24	y	Y	Y		56	h	H	H	
25	z	Z	Z		57	i	I	I	
26	ingen		Å	Å	58	¯	sæt LC		
27	,				59	.	:		
29	ingen	10	∝	∝	60	sæt UC			
30	tabul.				61	ingen		mellemrum	
31	Punch Off		ingen		62	ingen		mellemrum	
32	-	+			64	Car Return			
					65 - 127	ingen			

Vælg ydre enhed.

1C el. 3C. N>Y	YE := m; Ydre enhed nr. m tilkobles den aritmetiske enhed, men der sker ingen data-transport.
5C YAY	AR := 0; ARvadr := YE; YE := m; Nummeret paa den i forvejen valgte ydre enhed anbringes i adressepositionerne i ARv, mens resten af AR nulstilles. Derpaa tilkobles ydre enhed nr. m.
7C Y>A	AR := 0; ARvadr := YE; Nummeret paa den valgte ydre enhed anbringes i adressepositionerne af ARv, mens resten af AR nulstilles.

De ydre enheder har følgende numre:

m	Ydre enhed
0 el. 1	Tromlekanal nr. 0
2 el. 3	- - - 2
. . .	. . .
510 el. 511	Tromlekanal nr. 510
1280 1292	Hulkortenheder og funktioner i forb. med hulkortapparatur. Maa kun bruges i standardprogrammer for hulkortadministration.
2001	Baandstation nr. 1
2002	- - 2
2003	- - 3
2004	- - 4

De resterende numre er reserveret til udvidelser af de ydre enheder.

Læs fra ydre enhed.

1D LÆS

Fra den i forvejen valgte ydre enhed overføres indholdet af en kanal eller en blok eller et hulkort via AR til konsekutive celler i ferritlageret, saaledes at m er adressen paa den første celle. Normalt bør m være lige, idet overførslen kun da sker til helceller i ferritlageret. Iøvrigt gælder:

1. Fra tromlen læses 1 kanal = 32 helceller (1C-ordre maa være benyttet).
2. Fra baandstations bufferregister læses 1 blok = 65 helceller, hvoraf den første helcelle indeholder bloknummeret (baade 1C-ordre og 1E-ordre maa være benyttet).
3. Fra hulkort-bufferregister læses 1 blok = 82 halvceller, hvoraf de to første halvceller indeholder diverse kontroloplysninger. (1C-ordre maa være benyttet).

3D, 5D og 7D

bør ikke bruges, men har iøvrigt de i Lærebog i kodning for DASK beskrevne virkninger.

Se iøvrigt beskrivelserne af magnetbaandapparatur og af hulkortapparatur.

Skriv paa ydre enhed.

1F SKR

Til den i forvejen valgte ydre enhed overføres indholdet af en informationsenhed fra konsekutive celler i ferritlageret, saaledes at m er adressen paa dem første celle. Normalt bør m være lige, idet overførslen kun da sker fra helceller i ferritlageret. I øvrigt gælder:

1. Paa tromlen skrives 1 kanal = 32 helceller (1C-ordre maa være benyttet).
2. Til baandstation-bufferregister overføres 1 blok = 64 helceller (1C-ordre maa være benyttet).
3. Til hulkort-bufferregister overføres 1 blok = 80 halvceller (1C-ordre maa være benyttet).

3F, 5F og 7F

bør ikke bruges, men har iøvrigt de i Lærebog i kodning for DASK beskrevne virkninger.

Se iøvrigt beskrivelserne af magnetbaandapparatur og af hulkortapparatur.

Baandmanøvrering.

1E T>B	1. Hvis ARv = 0, sker der intet. 2. Hvis ARv = b × 2↑(-19), og hvis blok nr. a er den sidst benyttede, overføres blok nr. a + b til bufferregisteret. 3. Hvis ARv = -1, og hvis blok nr. a er den sidst benyttede, overføres samme blok til bufferregisteret. Før ovenstaaende udføres, afventer DASK dog evt. klarsignal fra den valgte baandstation (gælder ogsaa punkt 1).
5E SYN	AR nulstilles; derefter som 1E (punkt 1).
3E B>T	1. Hvis ARv = b × 2↑(-19), og hvis blok nr. a er den sidst benyttede, overføres bufferregisterets indhold til blok nr. a + b. Bufferregisterets indhold er uændret paa nær bloknummeret, der nu er a + b. 2. Hvis ARv = -1 faas samme virkning som hvis ARv = 0. Før ovenstaaende udføres, afventer DASK dog evt. klarsignal fra den valgte baandstation.
7E	AR nulstilles; derefter som 3E.

Alle ordrerne forudsætter, at en baandstation er valgt med en 1C-ordre. Hvad angaar nærmere detailler henvises til beskrivelsen af magnetbaandapparatur.

6.4 Macro-s.

For at forstaa virkningen af de første 15 macro-s maa man vide, at et oversat DASK ALGOL-program benytter to forskellige lagringsformer for reelle (flydende) tal og heltal. Den ene, store-repræsentation, benyttes for alle variable og konstanter, mens aritmetiske mellemresultater lagres i stack-repræsentation. Begge repræsentationer bestaar af en taldel og en exponentdel, og begge benytter en helcelle for hvert reelt tal, men i store-repræsentation bruges kun en halvcelle til hvert heltal (og iøvrigt ogsaa til hver boolesk variabel); forskellen mellem de to repræsentationer ligger iøvrigt i valget af exponent-repræsentation (se Manual, afsnit 11.4).

I følgende liste over macros betegner store[c] og stack[c] indholdet af celle c opfattet som et tal i henholdsvis store- og stack-repræsentation. Indholdet i AR er altid i store-repræsentation.

Tal i stack-repræsentation (samt indicerede variable) lagres i den saakaldte stack, der er den sidste del af ferritlageret. Indexregister B peger under oversættelse og kørsel af et ALGOL-program altid paa den sidst benyttede helcelle i stack-en, og ved indlæsning af et program starter NL 5 med B = 2048 (=0).

Vedrørende oversættelsen af en macro til maskinordrer anbefales det at sammenligne med Manual, afsnit 11.5.7, der handler om Simple arithmetic expressions in DASK ALGOL.

Betegnelsen uden adresse betyder, at den paagældende ordre hverken har adressedel eller indexmærke. Findes blot en af disse bestanddele, opfattes ordren som en macro med adresse.

Store til stack og omvendt.

O+V

Uden adresse: B := B - 2; stack[B] := AR;
Efter at indexregister B er mindsket med 2, omformes AR fra store-repræsentation til stack-repræsentation og lagres i celle B. Ordren oversættes til en maskinordre:

17 594

Med adresse: B := B - 2; stack[B] := store[c];
Efter at indexregister B er mindsket med 2, omformes celle[c] fra store-form til stack-form, og lagres i celle B. Ordren oversættes til to maskinordrer:

40<adr.del>,<indexmærke>
17 594

Herunder er k = adressen paa 40-ordren.

O-V

Uden adresse: B := B - 2; stack[B] := - AR;
Efter at indexregister B er mindsket med 2, omformes AR fra store-form til stack-form og lagres med modsat fortegn i celle B. Ordren oversættes til en maskinordre.

17 242

Med adresse: B := B - 2; stack[B] := - store[c];
Efter at indexregister B er mindsket med 2, omformes celle[c] fra store-form til stack-form og lagres med modsat fortegn i celle B. Ordren oversættes til to maskinordrer:

40<adr.del>,<indexmærke>
17 242

Herunder er k = adressen paa 40-ordren.

A>V

Uden adresse: AR := stack[b]; B := B + 2;
Indholdet i celle B omformes fra stack-form til store-form og anbringes i AR. Samtidig tælles stacken ned. Ordren oversættes til en maskinordre:

17 361

Med adresse: store[c] := stack[B]; B := B + 2;
Indholdet i celle B omformes fra stack-form til store-form og anbringes i celle c. Samtidig tælles stacken ned. Ordren oversættes til to maskinordrer:

17 361
08 <adr.del>,<indexmærke>.

Herunder er k = adressen paa 08-ordren.

HEL

Uden adresse: ARv := afrund(stack[B]); B := B + 2;
Indholdet i celle B afrundes til det nærmeste heltal, omformes til store-form og anbringes i ARv (d.v.s. med enhed i pos. 19). Ordren oversættes til en maskinordre:

17 554.

Med adresse: halvcelle[c] := afrund(stack[B]); B:= B + 2;
Indholdet i celle B afrundes til det nærmeste heltal, omformes til store-form og anbringes i halvcelle c. Samtidig tælles stacken ned. Ordren oversættes til to maskinordrer:

17 554
28 <adr.del>,<indexmærke>.

Herunder er k = adressen paa 28-ordren.

Addition og subtraktion i stack.

A+V

Uden adresse: stack[B] := stack[B] + AR;
AR omformes fra store-form til stack-form og adderes til indholdet i celle B. Ordren oversættes til en maskinordre:

17 257

Med adresse: stack[B] := stack[B] + store[c];
Indholdet i celle c omformes fra store-form til stack-form og adderes til indholdet i celle B. Ordren oversættes til to maskinordrer:

40 <adr.del>,<indexmærke>
17 257

Herunder er k = adressen paa 40-ordren.

S+A

Maa kun bruges uden adresse: B := B + 2; stack[B] := stack[B] + stack[B-2];
De to øverste tal i stacken adderes, og resultatet anbringes hvor det næstøverste tal stod; desuden tælles stacken ned. Ordren oversættes til en maskinordre:

17 254

A-V

Uden adresse: stack[B] := stack[B] - AR;
AR omformes fra store-form til stack-form og subtraheres fra indholdet i celle B. Ordren oversættes til en maskinordre:

17 264

Med adresse: stack[B] := stack[B] - store[c];
Indholdet i celle c omformes fra store-form til stack-form og subtraheres fra indholdet i celle B. Ordren oversættes til to maskinordrer:

40 <adr.del>,<indexmærke>
17 254

Herunder er k = adressen paa 40-ordren.

S-A

Maa kun bruges uden adresse: B := B + 2; stack[B] := stack[B] - stack[B-2];
Det øverste tal i stacken subtraheres fra det næstøverste tal, og resultatet anbringes hvor det næstøverste tal stod. Desuden tælles stacken ned.
Ordren oversættes til en maskinordre:

17 261

Multiplikation i stack.

A×V

Uden adresse: stack[B] := stack[B] × AR;

AR omformes fra store-form til stack-form og multipliceres med indholdet i celle B. Resultatet anbringes i celle B. Ordren oversættes til en maskinordre:

17 330

Med adresse: stack[B] := stack[B] × store[c];

Indholdet i celle c omformes fra store-form til stack-form og multipliceres med indholdet i celle B. Resultatet anbringes i celle c. Ordren oversættes til to maskinordrer:

40 <adr.del>,<indexmærke>

17 330

Herunder er k = adressen paa 40-ordren.

S×A

Maa kun bruges uden adresse: B := B + 2; stack[B] := stack[B] × stack[B-2];

De to øverste tal i stacken multipliceres, og resultatet anbringes hvor det næstøverste tal stod. Desuden tælles stacken ned. Ordren oversættes til en maskinordre:

17 327

Division i stack.

A/V

Uden adresse: stack[B] := stack[B]/AR;
AR omformes fra store-form til stack-form og divideres op i indholdet i celle B. Resultatet anbringes i celle B. Ordren oversættes til en maskinordre:

17 340

Med adresse: stack[B] := stack[B]/store[c];
Indholdet i celle c omformes fra store-form til stack-form og divideres op i indholdet i celle B. Resultatet anbringes i celle B. Ordren oversættes til to maskinordrer:

40 <adr.del>,<indexmærke>
17 340

Herunder er k = adressen paa 40-ordren.

S/A

Maa kun bruges uden adresse: B := B + 2; stack[B]:= stack[B]/stack[B - 2];
Det øverste tal i stacken divideres op i det næstøverste, og resultatet anbringes, hvor det næstøverste stod. Desuden tælles stacken ned. Ordren oversættes til en maskinordre:

17 337

Eksponentialfunktion.

A↑V

Uden adresse: stack[B] := stack[B] ↑ AR;
AR omformes fra store-form til stack-form, og indholdet i celle B opløftes til den exponent, indholdet i AR angiver. Resultatet anbringes i celle B. Ordren oversættes til en maskinordre:

17 368

Med adresse: stack[B] := stack[B] ↑ store[c];
Indholdet i celle c omformes fra store-form til stack-form, og indholdet i celle B opløftes til den exponent, som indholdet i celle c angiver. Resultatet anbringes i celle B. Ordren oversættes til to maskinordrer:

40 <adr.del>,<indexmærke>
17 368

Herunder er k = adressen paa 40-ordren.

S↑A

Maa kun bruges uden adresse: B := B + 2; stack[B]:=
stack[B] ↑ stack[B-2];
Det næstøverste tal i stacken opløftes til den exponent, som det øverste tal angiver. Resultatet anbringes hvor det næstøverste tal stod. Samtidig tælles stacken ned. Ordren oversættes ved en maskinordre:

17 365

Tromleadministration.

TIN

Som adresse skal bruges identifikatoren for en tromleblok (og indexmærke b, c eller d er ikke tilladt). Da er virkningen følgende: Den paagældende tromleblok overføres fra tromlen til ferritlageret (paa det i forvejen reserverede sted). Ordren oversættes til tre maskinordrer:

55 <identifikator>-1
17 713.
17 222.

TUD

Som adresse skal bruges identifikatoren for en tromleblok (og indexmærke b, c eller d er ikke tilladt). Da er virkningen følgende: Den paagældende tromleblok overføres fra ferritlageret til tromlen (paa de i forvejen reserverede kanaler). Ordren oversættes til tre maskinordrer:

55 <identifikator>-1
17 713
17 220

Naar en tromleblok med ordren TIN<identifikator> er bragt paa plads i ferritlageret, kan man f.ex hoppe til den j-te ordre i tromleblokken med hopordren:

10 <identifikator> + j - 1

eller i det hele taget henvise til celler i tromleblokken, idet identifikatoren peger paa den første celle i blokken.

Macro-en TUD skal bruges, naar man efter at have ændret i en tromleblok ønsker at gemme den ændrede blok paa tromlen. Iøvrigt bemærkes, at tromleblokkens identifikator oversættes til to forskellige adresser eftersom den optræder i TIN og TUD eller i en anden ordre: I TIN og TUD henviser identifikatoren til tre halvceller, der indeholder tromleblokkens kanalnummer og oplysning om dens placering i ferritlageret; disse tre halvceller ligger paa det sted i det oversatte program, der svarer til tromleblokkens placering i NL 5-koden. I alle andre ordrer henviser identifikatoren til den første af de fælles celler der er reserveret alle tromleblokkene under en og samme kodeblok; disse celler ligger umiddelbart efter denne kodeblok.

7. Exempler og øvelser.

Exempel 7.1

Skal fire maskintal i helcellerne A, A + 2, A + 4 og A + 6 adderes, og skal resultatet anbringes i celle A + 8, kan det ske med følgende kode (idet det antages at spild ikke kan forekomme)

```
start:  40 A        ; Her hentes det første tal
        00 A + 2
        00 A + 4
        00 A + 6
        08 A + 8    ; Nu gemmes resultatet
        30 start    ; Stop. Ved start gentages beregningen.
```

Er de aktuelle talværdier 0.0123, 0.0145, - 0.000156 og 0.0078, kan koden udbygges til følgende færdige program:

```
begin code
    start: 40 A
           00 A + 2
           00 A + 4
           00 A + 6
           08 A + 8
           30 start
= A :f .0123, .0145, - 1.56lo- 4, .0078 end;
```

Exempel 7.2

Idet komponenterne af to N-dimensionale vektorer er lagret som heltal i konsekutive halvceller, den første vektors første komponent i celle AA og den anden vektors komponenter i fortsættelse heraf, skal vektorernes skalarprodukt dannes som et heltal i helcelle BB; tallet N står som adressetal i celle Dimension:

```
start:  48 BB       ; nulstil celle BB
        60 Dimension
        0c 1
        0F 1        ; ARvadr = N. Evt. indexmærke fjernet
        29 k + 1
        35          ; B := N. Tælling startes
        75 AA, b
        74 k + 3    ; sæt adresse = AA + N
løkke:  35 2047, b; tælling
        64 AA, b
        2A   , b    ; AR := a[i] × b[i]
        06 BB       ; summen dannes i BB
        33 løkke    ; hop så længe B ≠ 0
        30 start    ; stop
```

Exempel 7.3

Tallene a, b, c, og d er reelle tal lagret på store-form. Tallet ab/(c+d) skal dannes og lagres i cellen efter d:

```
        e := d + 2    ; etikettedefinition for resultatcellen
start:  0 + V   a     ; a anbringes i stakken
        A × V   b     ; a × b anbringes i stakken
        0 + V   c
        A + V   d     ; c + d anbringes i stakken
        S / A         ; a × b/(c + d) anbringes i stakken
        A> V    e     ; resultatet anbringes i celle e
        30      start ; stop
```

Bemærk, at denne kode vil blive oversat til 12 maskinordrer. Iøvrigt vil den (på nær stopordren) være identisk med oversættelsen af ALGOL - sætningen

```
e := a × b/(c + d);
```

Den indledende etikettedefinition e := d + 2 kunne undværes ved at ændre den næstsidste ordre til A × V d + 2.

Exempel 7.4

I et ALGOL-program er det ikke muligt at udnytte det kontrolbords-styrede 56-hop, men det kan f.ex. lade sig gøre ved at erklære følgende procedure:

```
procedure  Kontrolhop ( L );
           label L;
           begin code
              56 k + 2
              10 END
              algol; go to L code
           in END: end Kontrolhop;
```

Bruges denne i et ALGOL-program som en normal procedure, vil den - såfremt 56-omskifteren er i aktivstilling - bevirke hop til etiketten L; L kan f.ex. henvise til en kontroludskrift eller lignende, som må afsluttes med ordren

```
10 END
```

Er omskifteren i passivstilling, har Kontrolhop (L) ingen virkning.

Exempel 7.5

Skal to reelle tal indlæses og sumnen derefter trykkes kan det ske med følgende program:

```
begin  code
start:  17 læs
        40 a          [1. programparameter for læs-proceduren]
        40 a + 2, c   [2. programparameter; som den sidste skal den c-mærkes.
                       Herved læses to tal fra strimmel til celle a og
                       celle a + 2 ]
        0 + V     a
        A + V     a + 2   ; i staktoppen står nu summen af tallene
        A > V     a       ; resultatet lagres i celle a
        17 tryk           ; hop til tryksekvens
        40 layout         ; layout-parameter
        40 a,c            [ sidste programparameter, som henter det tal der
                            skal trykkes. C-mærkningen skyldes at det er den
                            sidste parameter ]
        30 start          ; stop
 = layout: {-n.ddd10-dd}  ; det ønskede layout lagres i en helcelle
        a:  end           ; arbejdscellen a defineres.
```

Iøvrigt henvises til Manual, hvor man i afsnit 11 finder konventionerne for brug af parametre ved procedure-kald.