ALICE

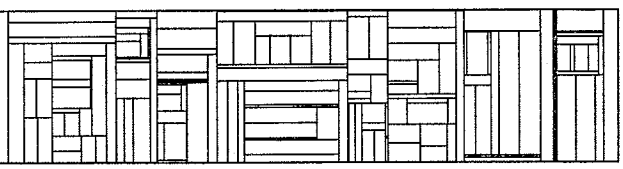
A Language

Improving Computer Education

Juli 1972

Frede Dybkjær.

Matematisk Institut Aarhus Universitet
DATALOGISK AFDELING
Ny Munkegade - 8000 Aarhus C - Danmark
Tlf. 06-12 83 55



DAIMI PB 5.

INDHOLD.

INTRODUKTION

I det følgende beskrives et system, der kan bruges som illustrationsmiddel i en datalogiundervisning.

Systemet er et sprog og et operativsystem - i det følgende samlet kaldet ALICE (A Language Improving Computer Education) -, der er implementeret på en computer.

Jeg forestiller mig, at en undervisning af studerende med ALICE som illustrationsmiddel opfylder følgende:

1) De studerende har på forhånd erhvervet sig et kendskab til brug af edb - mindst svarende til bogen "Arbejde med datamater" [12].

2) De studerende har nogenlunde uhindret adgang til en computer, hvor ALICE er implementeret.

3) ALICE bruges til illustration af et eller andet notestof, der behandler grundlæggende principper i datalogi.

4) Dette notestof behandler også områder, som ALICE er uegnet til at illustrere (f. eks. samfundsmæssige aspekter, analogsystemer).

5) Der bruges også andet illustrationsmateriale end ALICE.

Jeg vil gerne understrege, at ALICE er uafprøvet i undervisningssituationer. Dette skrift skal derfor betragtes som et oplæg til en diskussion af, hvordan en undervisning i datalogi kan og bør laves.

INSPIRATION

Bj. Svejgaard satte med nogle inspirerende grundlæggende ideer - eksemplificeret i [1] - det hele i gang.

Kendskab til CDC 6400 editor-system [2] og kendskab til BASIC-sprogets udbredelse i undervisningssammenhænge [3], [4], [12] har betydet meget.

Resten af referencerne har også på hver sin måde givet en vis inspiration.

ET UNDERVISNINGSFORLØB

Jeg vil herefter introducere ALICE ved i 13 punkter at beskrive den (ret store) del af et fornuftigt undervisningsforløb i datalogi, hvor ALICE kan bruges som illustrationsmiddel.

De understregede ord forestiller jeg mig, at det tilhørende notemateriale forklarer.

I eksemplerne har jeg med pile peget på nogle af de mest væsentlige linier.

PUNKT 1

Operativsystem, programmeringssprog, algoritmer, input og output skal være 100% forklaret. *)

Registre, boolske algebra, dyadiske og monadiske operationer illustreres ved hjælp af et logisk elektrisk undervisningssystem [5], [6], [7], [8].

ALICE er for de studerende nu et system, der kan udføre programmer med boolske operationer på 6-bits registre og med input-output i bitform på registrene.

Af operativsystemordrer kendes LIST, RUN, DELETE, ZERO (det eneste der nulstiller variable).

Desuden forklares operativsystemordrerne SAVE og GET og sprogordren UDFØR. Baggrundslager forklares. Fejlmeddelser.

(Det meste af ovenstående bør de studerende kende på forhånd, så at kun ALICE er nyt. (Jvf. introduktion, pkt. 1)).

```
..10INPUTBITREG(1)
..30PRINTBITREG(1)
..40END
..RUN
:110101
110101
..
```

*) I systemet ALICE skriver man sine ordrer, hver gang ALICE har skrevet "..". Sprogordrer, med linienumre først, listes i en lokal file. Operativsystemordrer udføres øjeblikkeligt. Inddata til sprogordrer skrives efter at systemet har skrevet ":".

```
..LIST
          40 PRINTBITREG(1)
          50 PRINTBITREG(2)
          60 PRINTBITREG(3)
          70 END
          IKKE FLERE LINIER
..SAVEOUTPUTREG10G20G3                    <--
..
..DELETE
..10REG(3)=REG(1) ANDREG(2)
..20UDFØROUTPUTREG10G20G3
..30END
..RUN

          OUTPUTREG10G20G3
         DETTE NAVN OVERSÆTTES NU
         DET SVARER HER TIL NR.
          39

 110101
 110011
 110001
..
..10REG(3)=REG(1) ORREG(2)
..LIST
          10 REG(3)=REG(1) ORREG(2)
          20 UDFØROUTPUTREG10G20G3
          30 END
          IKKE FLERE LINIER
..RUN

 110101
 110011
 110111
..
```

64729

```
..
..10REG(3)=((REG(1) ORREG(2)
..LIST
          10 REG(3)=((REG(1) ORREG(2)
          20 UDFØROUTPUTREG10G20G3
          30 END
          IKKE FLERE LINIER
..RUN
          FORKERT ANTAL )    I LINIE      10    <--

         REKLAGER
..
```

```
..LIST
     5
    10
    30 PRINTBITREG(1)
    40 PRINTBITREG(2)
    50 END
    IKKE FLERE LINJER
..RUN

110011
110101
..ZERO
..LIST
     5
    10
    30 PRINTBITREG(1)
    40 PRINTBITREG(2)
    50 END
    IKKE FLERE LINJER
..RUN

000000
000000
..
```

PUNKT II

Simulation. Gennem mere eller mindre indviklede primitive cellulære automater gøres de studerende familiære med det under punkt I nævnte.

```
..
..DELETE
..10INPUTBITREG(1)
..20INPUTBITREG(2)
..30INPUTBITREG(3)
..40INPUTBITREG(4)
..50INPUTBITREG(5)
..60INPUTBITREG(6)
..70END
..SAVEINPUTREG123450G6
..DELETE
..10PRINTBITREG(1)
..30PRINTBITREG(3)
..20PRINTBITREG(2)
..40PRINTBITREG(4)
..50PRINTBITREG(5)
..60PRINTBITREG(6)
..70END
..SAVEOUTPUTREG123450G6
..
```

```
..
..10UDFØRØUTPUTREG123450G6
..LIST
        10 UDFØRØUTPUTREG123450G6
        15 REG(2)=REG(1) ANDREG(3)
        20 REG(4)=REG(3) ANDREG(5)
        30 REG(3)=REG(2)ORREG(4)
        35 REG(5)=REG(4) ORREG(6)
       100 UDFØRØUTPUTREG123450G6
        IKKE FLERE LINIER
..RUN
      ADVARSEL: PROGRAM IKKE AFSLUTTET AF END I LINIE 10
```

```
010101
010101
111111
101010
101010
111000
010101
010101
111111
101010
111010
111000
..
```

```
..
..RUN
      ADV
010101
010101
111111
101010
111010
111000
010101
010101
111111
111010
111010
111000
..
```

```
..
..RUN
      ADVARSEL:
010101
010101
111111
111010
111010
111000
010101
010101
111111
111010
111010
111000
..
```

PUNKT III

Hurtiglager forklares. Transporter til/fra registre fra/til variable introduceres i ALICE. Herefter kan registre og variable bruges uden smålig hensyntagen til, hvordan det rent faktisk foregår.

Bitkonfigurationer kan nu opfattes som text. Der laves textbehandling, og det er nu muligt, at få registrene og de variable ud/ind skrevet som text (ordrene INPUTTEXT, PRINTTEXT).

Sprogordrene GOTO og IF-THEN introduceres (IF-THEN tester på den sidste (mindst betydende) bit i udtrykket).

Der laves simple gentagemekanismer, der for eksempel kan udskrive en tabel over bitkonfiguration/bogstavbetydning.

```
..
..DELETE
..10 REG(7)=REG(7) NONEQUI REG(1)
..20 IF REG(7) THEN 200
..30 REG(7)=REG(7) NONEQUI REG(2)
..40 IF SHIFT REG(7) THEN200
..50 REG(7)=REG(7) NONEQUI  REG(3)
..60 IF SHIFT(SHIFT(REG(7)))THEN 200
..70 REG(7)=REG(7) NONEQUI  REG(4)
..80 IF SHIFT(SHIFT(SHIFT(REG(7)))) THEN 200
..90 REG(7)= REG(7) NONEQUI REG(5)
..100 IF SHIFT(SHIFT(SHIFT(SHIFT(REG(7))))) THEN 200
..110 REG(7)=REG(7) NONEQUI REG(6)
..200 END
..SAVE ADDER
..
```

```
..LIST
        10 INPUTBIT REG(7)
        15 PRINTBIT REG(7)
        17 PRINTTAL REG(7)
        30 PRINT =ANDRINGER I REG(7)=
        40 UDFOR ADDER
        50 PRINTBIT REG(7)
        52 PRINTTAL REG(7)
        55 PRINTTEXT REG(7)
        60 IF SHIFT(SHIFT(SHIFT(SHIFT(SHIFTREG(7))))) THEN 100
        70 GOTO 40
       100 END
        IKKE FLERE LINIER
```

```
..RUN

:
      ADVARSEL: BITINPUT, IKKE ANDET  I LINIE     10
:000000
000000
   0
ANDRINGERIREG(7)
000001
   1
1
000010
   2
2
000011
   3
3
000100
   4
4
000101
   5
5
000110
   6
6
000111
   7
7
001000
   8
8
001001
   9
9
001010
  10
A
001011
  11
B
001100
  12
C
001101
  13
D
001110
  14
E
001111
  15
F
010000
  16
G
010001
  17
H
010010
  18
I
010011
  19
J
010100
  20
K
010101
  21
L
010110
  22
M
010111
  23
N
011000
  24
O
011001
  25
P
011010
  26
Q
011011
  27
R
011100
  28
S
011101
  29
T
1000 ORDRER UDFORT, TRYK KAON FOR STOP
011110
  30
U
011111
  31
V
100000
  32
W
..
```

(besvar med stumme-input).

PUNKT IV

Operativsystemordren PARAM introduceres. Den muliggør bl. a. ændring af parameteren wordlength (antal bit pr. ord) og af parametrene bittrue og bitfalse, der er udskrivningsværdier af bits. Da param også bruges til styring af input-outputmedier, foregår ændringerne altid på skrivemaskine. I dette eksempel har jeg brugt PARAM til at ændre ordlængden fra 6 til 36 bit.

```
..
....PARAM
..DELETE
..10INPUTTEXTA
..12INPUTTEXTB
..20C=AANDB
..30D=AORB
..40PRINTTEXTA
..50PRINTTEXTB
..60PRINTTEXTC
..70PRINTTEXTD
..80END
..RUN

:OFREDE
:MAJLIS
OFREDE
MAJLIS
0AJ40C
MFRVVU
..
```

PUNKT V

Bitkonfigurationer kan betyde tal. Input/output som tal introduceres i ALICE. En additionsalgoritme laves. (På 6-bit ord). Her bør komme en undervisning i datamaters kredsløbsopbygning (rent teknisk), og man bør komme med en bemærkning om, at når man bygger en datamat, må man vælge, hvilke simple algoritmer man vil have som maskincode (ordrer) (f. eks. normalt addition), og hvilke man må konstruere derudfra. Altså software kontra hardware. Mikroordrer, integrerede kredse og ferritkerner nævnes.

```
..
..DELETE
..ZERO
..10 INPUTBIT A(25),A(24)
..20 PRINTBIT A(25),A(24)
..SAVE INPUTA25OGA24
..RUN
      ADVARSEL: PROGRAM IKKE AFSLUTTET AF END  I LINIE     20

:111111
:
      ADVARSEL: BITINPUT, IKKE ANDET  I LINIE     10
:000000
111111000000
..
..DELETE
..
..
..5 A(23)=A(24)
..1
..10 IF A(22) THEN 100
..20 IF SHIFT A(22) THEN 100
..30 IF SHIFT(SHIFT A(22)) THEN 100
..
..40 IF SHIFT SHIFT SHIFT A(22) THEN 100
..50 IF SHIFT SHIFT SHIFT SHIFT A(22) THEN 100
..60 IF SHIFT SHIFT SHIFT SHIFT SHIFT A(22) THEN 100
..70 REM NU ER ALLE SEKS BIT UNDERSOGT
..80 PRINT =A(22) ER NUL=
..90 A(23)=A(25)
..100 END
..SAVE ER A22 LIG NUL RESULT I A23
```

(Besvar med strimmelinput)

```
..RUN

SKRIVTOGANGEBIT,OG
JEGLAGGERDEMSAMMEN
:000110
:
      ADVARSEL: BITINPUT, IKKE ANDET  I LINIE      40
:001100
000110
001100
   6  12
A(22)ERNUL
RESULTATETER
010010
  18
..
..LIST
        10 PRINT = SKRIV TO GANGE BIT, OG=
        20 PRINT =JEG LAGGER DEM SAMMEN=
        30 INPUTBIT A
        40 INPUTBIT B
        50 PRINTBIT A
        60 PRINTBIT B
        70 PRINTTALA,B
        80 C=A NONEQUI B
        90 A(22)= A AND B
       100 UDFOR ER A22 LIG NUL RESULT I A23
       110 IF A(23) THEN 180
       120 A=C
       130 B=SHIFT SHIFT SHIFT SHIFT SHIFT A(22)
       140 GOTO 80
       180 PRINT =RESULTATET ER=
       190 PRINTBIT C
       200 PRINTTAL C
       210 END
        IKKE FLERE LINIER
..RUN

SKRIVTOGANGEBIT,OG
JEGLAGGERDEMSAMMEN
:010011
:
      ADVARSEL: BITINPUT, IKKE ANDET  I LINIE      40
:001011
010011
001011
  19  11
A(22)ERNUL
RESULTATETER
011110
  30
..
```

PUNKT VI

Nødvendigheden af interrupts forklares f. eks. ved processtyring eller simpelthen ved ydre enheder.

I ALICE kan interruptes ved / i input eller ved en særlig knap.

```
..
..LIST
        10 GOTO 20
        20 GOTO 10
        30 END
        IKKE FLERE LINIER
..RUN

      100 ORDRER UDFORT, TRYK KAON FOR STOP
      100 ORDRER UDFORT, TRYK KAON FOR STOP
      100 ORDRER UDFORT, TRYK KAON FOR STOP
..
DELETE
```

```
..RUN

:010001
:
      ADVARSEL: BITINPUT, IKKE ANDET  I LINIE     10
:0010/..
..LIST
        10 INPUTBIT A
        20 GOTO 10
        30 END
        IKKE FLERE LINIER
..
```

PUNKT VII

Med den begrundelse, at den studerende nu vil få et indblik i arbejdet for at få en datamat (fra maskincode, maskinordrer) til at forstå et højere programmeringssprog introduceres datastrukturproblemer. Først ved indirekte adressering, en variabel i ALICE kan nu betyde en adresse.

```
..RUN

:19


:001110
001110
..
..RUN

:7


:011101
000000
..
..LIST
        10 INPUTTAL A(3)
        20 INPUTBIT B(19)
        30 PRINTBIT B(A(3))
        40 END
        IKKE FLERE LINIER
..RUN

:19


:011010
011010
..
```

PUNKT VIII

lister, træstruktur illustreres nu ved hjælp af nogle ALICE-operativsystemordrer. Garbagecollection.

```
..DELETE
..10INPUTTEXTREG(6)
..TESTD
          1 000000      0
          2 000003      3
          3 00000A     10
          4 000000      0
          5  INPUT
          6 TEXTRE
          7 G(6)_0
          8 TESTD_
..20PRINTTEXTREG(8]]]]]]]
..TESTD
          1 000000      0
          2 000003      3
          3 00000A     10
          4 000008      8
          5  INPUT
          6 TEXTRE
          7 G(6)_0
          8 00000K     20
          9 000000      0
         10  PRINT
         11 TEXTRE
         12 G(8]]]
         13 ]]]]]_
         14 TESTD_
..20PRINTTEXTREG(8)
..TESTD
          1 000000      0
          2 000003      3
          3 00000A     10
          4 00000E     14
          5  INPUT
          6 TEXTRE
          7 G(6)_0
          8 00000K     20
          9 000000      0
         10  PRINT
         11 TEXTRE
         12 G(8]]]
         13 ]]]]]_
         14 00000K     20
         15 000000      0
         16  PRINT
         17 TEXTRE
         18 G(8)_0
         19 TESTD_
..15REG(8)=REG(6) ORREG(8)
..TESTD
```

```
          1 000000      0
          2 000003      3
          3 00000A     10
          4 00000I     19
          5  INPUT
          6 TEXTRE
          7 G(6)_0
          8 00000K     20
          9 000000      3
         10  PRINT
         11 TEXTRE
         12 G(8]]]
         13 ]]]]]_
         14 00000K     20
         15 000000      0
         16  PRINT
         17 TEXTRE
         18 G(8)_0
         19 00000F     15
         20 00000E     14
         21  REG(8
         22 )=REG(
         23 6) ORR
         24 EG(8)_
         25 TESTD_
..30END
..TESTD
          1 000000      0
          2 000003      3
          3 00000A     10
          4 00000J     19
          5  INPUT
          6 TEXTRE
          7 G(6)_0
          8 00000K     20
          9 000000      0
         10  PRINT
         11 TEXTRE
         12 G(8]]]
         13 ]]]]]_
         14 00000K     20
         15 00000P     25
         16  PRINT
         17 TEXTRE
         18 G(8)_0
         19 00000H     15
         20 00000E     14
         21  REG(8
         22 )=REG(
         23 6) ORR
         24 EG(8)_
         25 000000     30
         26 000000      0
         27  END_0
         28 TESTD_
```

De seks søjler betyder: 1) nummer, 2) og 3) symbol, 4) adresse ved uoverensstemmelse, 5) adresse ved overensstemmelse og 6) værdi ved udhop.

```
..10INPUTBITREG(1)
..20PRINTBITREG(1)
..SAVEPORGRAM1
..TESTC
          1 A    10      3      2      0
          2 A    10      0      0      0
          3 B    11      4      0      0
          4 P    25      0      5      0
          5 O    24      0      6      0
          6 R    27      0      7      0
          7 G    16      0      8      0
          8 R    27      0      9      0
          9 A    10      0     10      0
         10 M    22      0     11      0
         11 1     1      0      0      1
         12 0     0      0      0      0
..LIST
         10 INPUTBITREG(1)
         20 PRINTBITREG(1)
         IKKE FLERE LINIER
..DELETE
..LIST
         IKKE FLERE LINIER
..GETPORGRAM1
..LIST
         10 INPUTBITREG(1)
         20 PRINTBITREG(1)
         IKKE FLERE LINIER
..15REG(1)=REG(1)ORREG(18)
..LIST
         10 INPUTBITREG(1)
         15 REG(1)=REG(1)ORREG(18)
         20 PRINTBITREG(1)
         IKKE FLERE LINIER
..SAVEPROGRAM2
..TESTC
          1 A    10      3      2      0
          2 A    10      0      0      0
          3 B    11      4      0      0
          4 P    25      0      5      0
          5 O    24     12      6      0
          6 R    27      0      7      0
          7 G    16      0      8      0
          8 R    27      0      9      0
          9 A    10      0     10      0
         10 M    22      0     11      0
         11 1     1      0      0      1
         12 R    27      0     13      0
         13 O    24      0     14      0
         14 G    16      0     15      0
         15 R    27      0     16      0
         16 A    10      0     17      0
         17 M    22      0     18      0
         18 2     2      0      0      3
         19 0     0      0      0      0
..DELETE
```

T92901

```
..LIST
        IKKE FLERE LINJER
..GETPORGRAM1
..LIST
        10 INPUTBITREG(1)
        20 PRINTBITREG(1)
        IKKE FLERE LINJER
..30END
..LIST
        10 INPUTBITREG(1)
        20 PRINTBITREG(1)
        30 END
        IKKE FLERE LINJER
..SAVEPROGRAM3                                    <--
..TESTC
          1 A    10     3     2     0
          2 A    10     0     0     0
          3 B    11     4     0     0
          4 P    25     0     5     0
          5 O    24    12     6     0
          6 R    27     0     7     0
          7 G    16     0     8     0
          8 R    27     0     9     0
          9 A    10     0    10     0
         10 M    22     0    11     0
         11 1     1     0     0     1
         12 R    27     0    13     0
         13 O    24     0    14     0
         14 G    16     0    15     0
         15 R    27     0    16     0
         16 A    10     0    17     0
         17 M    22     0    18     0
         18 2     2    19     0     3
         19 3     3     0     0     5
         20 0     0     0     0     0
..SAVEPROGRAM2                                    <--
      NAVNET FINDES ALLEREDE                      <--
..NONAMEPROGRAM2
..TESTC
          1 A    10     3     2     0
          2 A    10     0     0     0
          3 B    11     4     0     0
          4 P    25     0     5     0
          5 O    24    12     6     0
          6 R    27     0     7     0
          7 G    16     0     8     0
          8 R    27     0     9     0
          9 A    10     0    10     0
         10 M    22     0    11     0
         11 1     1     0     0     1
         12 R    27     0    13     0
         13 O    24     0    14     0
         14 G    16     0    15     0
         15 R    27     0    16     0
         16 A    10     0    17     0
         17 M    22     0    18     0
         18 2     2    19     0     0             <--
         19 3     3     0     0     5
         20 0     0     0     0     0
```

```
..SAVEPROGRAM2
..TESTC
 1  A   10    3    2    0
 2  A   10    0    0    0
 3  P   11    4    0    0
 4  P   25    0    5    0
 5  O   24   12    6    0
 6  R   27    0    7    0
 7  G   16    0    8    0
 8  R   27    0    9    0
 9  A   10    0   10    0
10  M   22    0   11    0
11  1    1    0    0    1
12  R   27    0   13    0
13  O   24    0   14    0
14  G   16    0   15    0
15  R   27    0   16    0
16  A   10    0   17    0
17  M   22    0   18    0
18  2    2   19    0    3
19  3    3    0    0    5
20  0    0    0    0    0
..DELETE
..LIST
        IKKE FLERE LINJER
..GETPROGRAM2
..LIST
        10 INPUTBITREG(1)
        20 PRINTBITREG(1)
        30 END
        IKKE FLERE LINJER
..
..TESTB
 1  A   10    3    2    0
 2  A   10  184    0    0
 3  B   11    4  162    0
 4  O   24   14    5    0
 5  U   30  148    6    0
 6  T   29    0    7    0
 7  S   28    0    8    0
 8  I   18    0    9    0
 9  T   29    0   10    0
10  U   30    0   11    0
11  A   10    0   12    0
12  T   29    0   13    0
13  E   14    0    0  101
14  I   18   23   15    0
15  N   23   88   16    0
16  S   28  129   17    0
17  I   18    0   18    0
18  T   29    0   19    0
19  U   30    0   20    0
20  A   10    0   21    0
21  T   29    0   22    0
22  E   14    0    0  10
23  S   28   27   24
24  A   10  159   25
25  V   31    0
26  E   14    0
27  G   16
28  E   14
29  T
30  R
31
```

6622901

PUNKT IX

Stak forklares. Compilers formål forklares endnu en gang.

Simple oversatte algoritmer demonstreres ved ALICE operativsystem-ordren TRANSLATE.

Da indeks i den aktuelle implementation er mindre end 25 beregnes en variabels adresse som bogstavnummer * 26 - indeks.
De fire søjler betyder: 1) nummer, 2) ordrebetegnelse, 3) variabel 1 og 4) variabel 2.

```
..10INPUTBITREG(1)
..30PRINTBITREG(1)
..40END
..TRANSLATE

          1        1        0       10
          2        7        0        1
          3        1        0       30
          4        3        0        1
          5        6     8188        0
          6        1        0       40
          7        2        0        0
..

..LIST
        10 INPUTTAL A(3)
        20 INPUTBIT B(19)
        30 PRINTBIT B(A(3))
        40 END
        IKKE FLERE LINIER
..TRANSLATE

          1        1        0       10
          2        9        0       29
          3        1        0       20
          4        7        0       71
          5        1        0       30
          6       24       52       29
          7        3        0      999
          8        6     8188        0
          9        1        0       40
         10        2        0        0
```

ALICE maskinkode.

Ordre	betydning	variabel 2	variabel 1
1	opdater linienr	linienr	
2	afslue beregning af program		
3	print som bit	variabel	
4	print som tal	variabel	
5	print som text	variabel	
6	print som text	------text-----	
7	input som bit	variabel	
8	input som text	variabel	
9	input som tal	variabel	
10	udfør program i navngivet område	adresse på adresse	
11	hop til oversat ordre	nummer	
12	hvis sidstebit(variabel) så over-spring næste ordre	variabel	
13	not	variabel	
14	shift	variabel	
15-22	or,.....,and	variabel 2	variabel 1
23	placer i adresse	adresse	variabel
24	beregn adresse	adresse på indeks	adressebase
25	shiftleft	variabel	
26	udfør den variable som ordre	variabel	

Ordrerne placerer et eventuelt resultat i staktoppen.

Hvis en variabel er 999 hentes aktuel variabel i staktoppen, ellers hentes aktuel variabel i den adresse variabel angiver.

PUNKT X

Oversættelse og udførelse af (aritmetiske og boolske) udtryk. Omvendt polsk notation forklares. Oversættelse af hoppeordrer, to pass er nødvendige (eller i hvert fald en efterberegning).

```
..LIST
        10 REG(3)=REG(2)OR (REG(2)ANDREG(1))
        20 UDFØROUTPUTREG10G20G3
        30 END
        IKKE FLERE LINIER
..TRANSLATE

        1       1       0       10
        2       22      2       1
        3       15      2       999
        4       23      999     3
        5       1       0       20
        6       10      0       39
        7       1       0       30
        8       2       0       0
..
```

```
..LIST
        10 INPUTBIT A
        20 GOTO 10
        30 END
        IKKE FLERE LINIER
..TRANSLATE

        1       1       0       10
        2       7       0       26
        3       1       0       20
        4       11      0       1
        5       1       0       30
        6       2       0       0
..LIST
```

PUNKT XI

Et ords indhold opfattes nu som ordre (40 bit).(NB! Dette er en variabels femte betydning, de andre var bit, tal, tekst og adresse).

I ALICE kan sprogordren FINDORDER ukritisk forsøge at udføre en variabel.

Hele problematikken om betydningen af et lager gennemdiskuteres.

```
..TRANSLATE

          1        1        0       10
          2        7        0       52
          3        1        0       20
          4       26        0       52
          5        1        0       30
          6        2        0        0
..LIST
        10 INPUTBIT B
        20 FINDORDER B
        30 END
        IKKE FLERE LINIER
..RUN

:0001100000001111001011000110001110000011    PRINTTEXT
FB6E3..
..LIST
        10 INPUTBIT B
        20 FINDORDER B
        30 END
        IKKE FLERE LINIER
..RUN

:0000110000000000000000000000000000110100    PRINTBIT B
0000110000000000000000000000000000110100..
..
```

PUNKT XII

Begrebet matematisk maskine - i første omgang regulær automat forklares.

Selve programmet bag ALICE fremvises og diskuteres, specielt måden et boolsk udtryk oversættes (pr. regulær automat).

input:	1	2	3	4	5	6
forr. input:						
1: (,start	1	0	0	3	6	4
2:)	0	2	5	3	0	0
3: dyad. op	1	0	0	3	6	0
4: monad. op	1	0	0	3	6	0
5: variabel	1	2	5	0	0	0
6: tal	0	7	0	0	0	0

Aktionstabel for udtryk.

PUNKT XIII

Forskellige beskrivelsesformer for sprog diskuteres. Backus-Naur-form, syntaks.

SYNTAX FOR ALICE

- Det med + markerede findes ikke i den beskrevne udgave:

```
<ALICE>::=<operativsystemordre>|
          <sprogordre          >|
          <ALICE><CR><ALICE>
<operativsystemomdre>::=
   DELETE<+ intervaller>|
   LIST   <intervaller>|
   GET    <navn>|
   SAVE   <navn>|
   NONAME <navn>|
   +RENAME <navn>,<navn>|
   TRANSLATE|RUN|PARAM|
   CHANGELIBRARY|ZERO|+ FILES|
   OUTSITUATE|INSITUATE|+ TEACH|
   TESTE|TESTD|
   TESTC|TESTB|TESTA|
<intervaller>::=ε|<linienr>. <linienr>|
                 <linienr>|
                 <intervaller>, <intervaller>
<linienr>::=<ciffer>|<linienr><linienr>
<navn>::=<bogstav>|<ciffer>|
         <navn><navn>
<ciffer>::=0|1|2|3|4|5|6|7|8|9
<bogstav>::=A|B|C|D ..... |Æ|Ø|Å
<sprogordre>::=<linienr><ordre>
```

```
<ordre>::=
    <variabel>=<udtryk>|
    GOTO<linienr>|
    IF<udtryk>THEN<linienr>|
    REM<TEXT>
    PRINT=<TEXT>=<outputafslutning>|
    <outputordre><variabelrække><outputafslutning>|
    <inputordre><variabelrække>|
    FINDORDER<udtryk>|
    EXE Q<navn>|
    + WHILE<udtryk>|
    + ENDWHILE|
    END


<outputordre>::=PRINT<art>
<inputordre>::=INPUT<art>
<art>::=BIT|TEXT|INTEGER
<variabel>::=REG<index>|<bogstav><index>
<index>::=ε|(<linienr>)|(<udtryk>)
<udtryk>::=<variabel>|
            <udtryk><dyadisk op.><udtryk>|
            <monadisk op><udtryk>|
            (<udtryk>)
<dyadisk op>::=AND|OR|NOR|NAND|EQUI|NONEQUI|
                EXCL|INCL|+ ADD|+ MULT|+ SUB|+ DIV
<monadisk op>::=NOT|SHIFT|+ SHIFT LEFT
<variabelrække>::=<variabel>|<variabelrække>,<variabelrække>
<outputafslutning::=ε|,
<text>::={alt undtagen CR,=og/}
```

Implementationen er (også) smuk og generel på følgende områder:

Forud definerede navne (sammensatte symboler, f. eks. RUN, DELETE, INPUTBIT og EXEQ) kan ændres ved kald af ordren CHANGELIBRARY. Derved kan man f. eks. få danske navne i stedet for de engelske.

Tilføjelse af en ordre i ALICE (sprog- eller operativsystemordre) klares let ved følgende proces: Skriv de (få) linier kode ordren kræver og indføj dem det rigtige sted i programmet. Tilføj navnet på den nye ordre v. hj. a. CHANGELIBRARY. Eksempel: Tilføjelse af sprogordren FINDORDER (udfører en variabel) tog 5 minutter.

KONKLUSION

Jeg mener her at have demonstreret et kraftfuldt system til illustration af en væsentlig del af en fornuftig datalogiundervisning.

BEGRUNDELSE FOR VALG AF MASKINE TIL AKTUEL IMPLEMENTATION

NB! Dette punkt er forholdsvis uvæsentligt for diskussion af ALICE's muligheder i undervisningssituationer.

Implementationen er foretaget på GIER i algol IV [13].

Begrundelsen herfor er:

1) GIER er den datamat jeg kender bedst.

2) Jeg har haft uhindret adgang til at bruge GIER.

3) GIER har en algol, der gør, at implementationen nogenlunde nemt kan overføres til andre maskiner f. eks. RC 7000 (med disk), RC 4000 eller ved at udnytte PASCAL til CDC's og ICL's maskiner.

Selvfølgelig vil det kræve en indsats at overføre ALICE til andre datamater, så det bliver helt compatibelt med GIER-udgaven. For at afhjælpe besværet har jeg forsøgt at lave implementationen meget procedureorienteret. Derfor er det kun nogle få procedurer, der laver input, output, pakning af ord, oppakning af ord og gemning på baggrundslagerområde.

MANGLER I AKTUEL IMPLEMENTATION

NB! Også dette punkt er forholdsvis uvæsentligt for diskussion af om ALICE kan bruges.

1) DELETE bør kunne tage linienumre som parametre (som nu LIST).

2) TEACH - til undervisning - og FILES - angivelse af navngivne områder - bør laves.

3) Visse utrerede situationer (brug af mere end 140 baggrundslagerområder og lign.) skal der tages højde for.

4) Operationerne +, -, :, * bør indføres (?)

Men alt dette (bortset fra TEACH) er ret nemt at indføje; her ikke gjort på grund af tidspres.

5) På visse områder er implementationen mere large i brug af plads end andre implementationer måske kan tåle.

6) Implimentationen bør laves så flere brugere fra forskellige terminaler samtidigt kan udnytte sproget.

7) Ordren RUN bør måske kunne have et navn som parameter.

8) De lokale files til oversat og uoversat program kan ikke blive mere end 1200 40-bits ord, dvs. 1200 oversatte ordrer, ca. 6000 tegn uoversatte sprogordrer excl. linienumre. (Se også pkt. 3).

9) I det hele taget er der en del i den aktuelle implimentation, som ikke er 100% godt.

LITTERATUR

[1] Bj. Svejgaard: "A letter to a Computer Manufacturer" (Nordisk Conference om programmeringssprog, 1971)

[2] TEACH, (RECAU-publikation, 1971)

[3] Leif Druedahl og Knud Jøring: BASIC-lærebogen (Gjellerup, 1972)

[4] H. B. Hansen: "BASIC for begyndere" (Regnecentralen, 1972)

[5] Artikelserie om basic computer principles, (Tidsskriftet Orbit, nov. 1970-71)

[6] Beskrivelser af firmaerne Digital, Metric, Uddata og skoleforlaget Gonge's logiske undervisningssystemer.

[7] Einar Bertelsen: "Undersøgelse af eksisterende elementært udstyr til brug ved undervisning i datalogi". (DAIMI, 1972)

[8] Diverse litteratur om datamaters logik. (DAIMI's bibliotek)

[9] Rapport om datalogiforsøg ved Frederiksberg gymnasium, (1970). Dagmarskolen, Ringsted (1970). Løkken Centralskole (1969-70).

[10] Diverse papirer fra udlandet, bl. a.:
OECD-rapport: Computer Sciences in Secondary Education (1971).
Computers and the Schools (Scottish Education Department, reprint 1971).

Diverse noter fra:
Forsøksgynmasiet, Oslo (1971)

[11] Tidskriftet: Computer Education 1-10

[12] Frede Dybkjær: "Arbejde med datamater",
(G. E. C. Gad, August 1972)

[13] Version 127/84 af GIER-algol IV-program
(Frede Dybkjær, 1972)

[14] Anders Andersen, Mogens Winther og Frede Dybkjær:
Rapport om studiegruppearbejde: Undervisnings-
situationen i datalogi (juni 1972, Økonomisk Inst.,
Århus)

[15] For lidt pædagogisk litteratur

[16] Al (?) litteratur udgivet på dansk under betegnelse:
"Introduktion til edb". (Se bogliste på bibliote-
ket)