PROGRAMMERINGSHJÄLPMEDEL FÖR INTEL 4040

PER WOLGAST
JANUSZ MISZCZUK

RE-172 Januari 1976
Inst. för Reglerteknik
Lunds Tekniska Högskola

I N N E H Å L L S F Ö R T E C K N I N G

A B S T R A C T

At the department of Automatic Control has been developed a programming tool for INTELS microcomputors, mainly for INTEL 8008 (1).

When INTEL 4040 became available on the market it seemed natural to adapt this programming tool the new microcomputor.

Thus the report is about developing a programming tool for INTEL 4040 and the departments PDP:15 has been used as a host computor.

That means:

1. A macrodefinition file has been written in order to be able to use the PDP:15 assembler.

2. We have modified the loader (load 8 - load 40) by:
   A. Making some changes in the main program (MAIN8 - MAIN40).
   B. Writing a new relocating subroutine. (REL8 - REL40)

3. We have written 2 routines which we consider necessary for many applications namely:
   A. A routine, MULT, which multiplies two 8-digit numbers with each other.
   B. A routine, DIV, which divide a 16-digit number a 8-digit number.

S A M M A N F A T T N I N G

Vid institutionen för reglerteknik har någått ett arbete med utveckling av programmeringshjälpmedel för INTELs mikrodator huvudsakligen INTEL 8008 (1).

När INTEL 4040 kom ut på marknaden föreföll det naturligt att anpassa detta programmeringshjälpmedel till den nya microdatorn.

Examensarbetet omfattar alltså framtagande av programmerings-hjälpmedel för INTEL 4040 och institutionens PDP:15 har använts som värddator. Det innebär:

1. En macrodefinitionsfil har skrivits för att man skall kunna använda PDP:15s assembler.

2. Vi har modifierat laddaren (LOAD8 - LOAD40) genom att
   A. Göra vissa ändringar i huvudprogrammet (MAIN8 - MAIN40).
   B. Skriva en ny relokeringssubrutin (REL8 - REL40).

3. Vi har skrivit två rutiner som vi anser vara nödvändiga i många tillämpningar nämligen:
   A. En rutin MULT, som multiplicerar två stycken 8-bits tal med varandra.
   B. En rutin DIV, som dividerar ett 16-bits tal med ett 8-bits tal.

1. INTRODUKTION

Det är en avsevärd fördel att kunna använda en stor dator som värddator när man skall kompilera program för en mikrodator. Man kan bl. a. använda värddatorns kraftfulla editor och överlägsna minneskapacitet. För att kunna utnyttja PDP:15:s assembler till att generera kod för INTEL 4040 har vi skrivit en macrodefinitionsfil ASS40, där mikrodatorns instruktioner definieras och som alltid assembleras tillsammans med mikrodatorprogrammet. Den relokerbara kod som blir resultatet av assemblerns arbete tas om hand av programmet LOAD40 som relokerar och länkar ihop subrutiner. LOAD40 producerar relokerad kod på hålremsa som direkt kan matas in i mikrodatorn. Man får även en utskrift på radskrivare av den binära koden.

2. ASSEMBLER FÖR INTEL4040

Uppgiften för assemblern är att översätta instruktionssymboler och adressymboler till numeriska värden. Adresserna behandlas automatiskt korrekt och om instruktionssymbolerna förs in i en User's Symbol Table så känner assemblern igen dem och lägger ut rätt kod. Vissa instruktioner kan definieras genom direkt assignement statements medan mera komplicerade anges genom macrodefinition. PDP:15:s assembler letar igenom användarens symbol tabell före den permanenta symbol tabellen där PDP:15:s instruktioner är lagrade. Det gör alltså inget om man använder samma symboler, vilket inte är fallet på alla datorer. Pseudooperations- och systemmacrodefinitions tabellen söks visserligen igenom före användarens symboltabell men dessa instruktioner inleds alltid med en punkt så det är ingen risk att de förväxlas med mikrodatorinstruktioner. (En pseudoinstruktion används för att modifiera assemblerns arbete. En sådan kan t. ex. avgöra om vissa instruktioner skall läggas ut beroende på om en symbol har värdet mindre än noll.)

PDP:15:s instruktionslängd är 18 bitar medan mikrodatorns bara är 8. En av laddarens uppgifter är därför att plocka ut de sista 8 bitarna.

2.1 M A C R O D E F I N I T I O N S F I L E N A S S 4 0

En listning av ASS40 finns i Appendix 1. För en komplett sammanställning av instruktionsrepertoaren se (2).

1. Ettordsinstruktion.

   A. Instruktionerna som alltid har samma kod tilldelas värde genom direkt assignement statement. T. ex.

      CLC = 360 (clear carry)

   B. Instruktioner som innehåller data eller referenser till något register i de 4 sista bitarna definieras med en macro. För en komplett sammanställning av PDP:15:s assembler se (3). Ex.

      .DEFIN ADD,R

      17&R+200

      .ENDM

      200 är instruktionskoden med noller i registerbitarna. Macrons namn är ADD och R är parameter. Macron anropas genom att skriva t. ex.

      ADD R8

      där R8 tidigare har tilldelats det oktala värdet 10. (Assemblern tolkar alla tal som oktala om det inte finns en pseudooperation som ändrar på detta).

      Det finns inga prioritetsregler utan evalueringen av uttrycken sker från vänster. Tecknet & betecknar logiskt OCH i varje position.

Ex.

```
.DEFIN   LDM,DATA
17GDATA+320
.ENDM
```

I anropet kan parametern vara en symbol men också ett tal.

C. Vissa instruktioner innehåller referenser till ett registerpar. Detta anges ömsom med ett udda tal och ömsom med ett jämnt. För att programmeraren skall slippa hålla reda på detta han en macro definierats där sista biten i R ignoreras. Ex.

```
.DEFIN   FIN,R
16GR+60
.ENDM
```

2. Tvåordsinstruktioner.

A. Två instruktioner, JUN och JMS, innehåller en 12 bits adress. Operationskoden för JUN är:

```
Ord 1      0100   AAAA
Ord 2      AAAA   AAAA
```

Macro definieras

```
.DEFIN   JUN,A
100
A
.ENDM
```

Här läggs alla 12 bitarna för adressen A i ett ord. Det är sedan laddarens uppgift att föra över de 4 mest signifikanta bitarna till första ordet.

8. Instruktionerna ISZ och JCN innehåller en adress på 8 bitar. De kan alltså endast adressera en av 16 sidor (=256 ord) av minnet, dvs samma sida som instruktionen ligger i. Ett anrop av respektive macro lägger dock ut 12 bitar som laddaren sedan korta ner till 8.

Laddaren kan då kontrollera om adressen verkligen ligger inom samma sida som instruktionen och ger annars varningsutskrift på teletype. Programmeraren får i så fall stuva om i programmen, framförallt ändra läge på subrutinerna. Detta kan dock ge upphov till nya varningsutskrifter. Programmeraren bör i stället se till från början att hoppadressen inte ligger för långt från instruktionen och även försöka att inte använda dessa instruktioner för mycket. Annars kan man alltid skjuta in 2 stycken villkorliga hoppinstruktioner med 12 bits adress. I stället för följande instruktion

```
                ISZ   ADRESS1
```

används

```
                ISZ   ADRESS2
                JUN   ADRESS2+2
ADRESS2         JUN   ADRESS1
```

Kommentar: Om ett visst villkor är uppfyllt så hoppar programmet till ADRESS1

JUN står för Jump UNconditional och är en tvåordsinstruktion.

Instruktionen JCN är egentligen uppdelad i 16 st där ett hopp sker om ett valt villkor är uppfyllt. Man kan alltså välja mellan 16 olika villkor som specificeras i bitarna som här har separerats med C C C C

	Operationskod	
JCN	0001	C C C C
	AAAA	AAAA

Innebörden av instruktionen är: hoppa till adressen inom samma sida om villkoret C C C C är uppfyllt. Detta villkor består egentligen av 3 st andra villkor som är: är accumulatorn = 0, är carryn = 1, är testsignalen = 0. C C C C kan sedan specificera ett av dessa tre villkor och ett hopp utförs om detta är uppfyllt. C C C C kan även specificera 2 villkor och hopp utförs då om något av dessa är uppfyllt och samma sak med tre villkor. Detta ger $\binom{3}{1} + \binom{3}{2} + \binom{3}{3} = 7$ st villkor. Man kan även komplementera dessa villkor. Det innebär att hopp sker om ett specifierat villkor (av tre möjliga) inte är uppfyllt, om två specifierade bägge inte är uppfyllda och slutligen om alla tre villkoren inte är uppfyllda. Detta ger 7 nya villkor. De två återstående instruktionerna är No Operation som redan finns och Jump UNconditional som också finns tidigare men med 12 bitars adress vilket är betydligt bättre (för samma längd på instruktionen = 2 ord)
Programmeraren skall naturligtvis inte behöva hålla reda på vilken kod som ger vilket villkor så därför har vi skrivit 15 st macrodefinitioner där den mnemoniska koden i macroprogrammet avgör villken macro som anropas och där-

med vilken kod som läggs. Villkoren: är testsignalen = 0 får då beteckningen T, är carryn = 1 betecknas med C och är accumulatorn = 0 skrivs med Z. Att man har eller-operationer mellan dessa villkor anges med T som står för True och att man komplementerar villkoren betecknas med F (False). Detta tecken följer direkt efter J (Jump). Därefter skrivs de villkor man vill ha med i nämnd ordning. Ex. anropet

JTTZ A

lägger ut instruktionen som medför hopp till A om testsignalen = 0.

JFTZ A

lägger ut instruktionen som medför hopp till A om testsignalen inte är 0 och accumulatorn inte är 0.

3. L A D D A R E F Ö R I N T E L 4 0 4 0.

Vid framtagande av progammeringshjälpmedel för INTEL 4040 hade vi stor användning av redan utvecklade progammeringshjälpmedel för INTEL 8008 (1).

Vi modifierade laddaren (LOAD8 - LOAD40) genom att göra vissa ändringar i huvudprogrammet (MAIN8 - MAIN40) och genom att skriva en ny relokerings subrutin (REL8 - REL40). LOAD40 utför laddning (relokring och länkning) av önskade program genom att programmeraren skriver ett laddningskommando följt av startadress och de programfiler som skall laddas. Det är:
LOAD XXX:YYY NAMN1, NAMN2. X är sidans nummer och Y är adressen inom sidan. Det förutsättes att alla program som skall laddas, är tillgängliga i binär, relokerbar form (detta åstadkommer tidigare PDP:15:s assembler).

3.1 L A D D A R E N S H U V U D P R O G R A M

En listning av MAIN40 återfinns i appendix 2. MAIN40 läser igenom binära filer två gånger. Första gången samlar den alla nödvändiga informationer om globala symboler., interna referenser, storleken på program osv. Dessa samlas i nedanstående vektorer:

- Filnamn - Namn på alla förekommande program
- Glob - Alla globala symboler
- Iadr - Slutliga adresser av globala symboler
- Extref - Namn på alla externa referenser
- Name - Namn på alla program

Andra gången däremot, genom att utnyttja alla dessa informationer, utför laddaren relokering och länkning. Som resultat av laddning får vi en stansad remsa färdig att matas in i INTEL4040:s programmeringsmodul. Dessutom skrivs det ut en lista på radskrivaren och eventuella varningsutskrifter på teletypen. Listan består av:

1. "File names" - namn på alla program som finns laddade
2. "Library" - namnet på eventuell biblioteksfil.
3. "Memory required" - startadress och slutadress.
4. "Global symbol table" - förteckning över alla globala symboler och deras adresser.
5. Förteckning över alla adresser och instruktioner i oktal form i samma ordning som på remsan. (Adresserna finns längst till vänster och visar var i programminnet instruktionerna kommer att hamna) se appendix 3.

3.2 R E L 4 0

En listning av REL40 och flödesschema återfinns i appendix 2. REL40 är ett maskinberoende program som utför den slutliga laddningen och relokeringen. REL40 anropas av MAIN40 varje gång ett nytt program skall laddas. Den behandlar information samlad i informationsblock. Genom att anropa en subrutin "DATINF" hämtas ett informationsblock och packas upp. Beroende på om innehållet är en absolut instruktion eller relokerbar adress behandlas det på olika sätt. Den absoluta instruktionen skrives ut oförändrad. För den relokerbara adressen däremot kontrolleras om den ligger inom samma program där den anropas eller utanför detta. I det sista fallet är det en extern referens. När alla instruktioner har blivit behandlade stänges filen.

I N F O R M A T I O N S B L O C K

PDP:15 assemblern producerar informationsenheter i form av block om 4 ord. Varje block består av en identifikationskod (6 bitar och ett dataord 18 bitar. Ett informationsblock är uppbyggt på följande sätt:

ORD 1	KOD1	KOD2	KOD3
ORD 2	DATAORD 1		
ORD 3	DATAORD 2		
ORD 4	DATAORD 3		

Koderna anger vilken typ motsvarande dataord är. Koderna kan anta värden från 1 till 24. För närmare beskrivning se (5).

De enda koderna som är relevanta i vårt fall är kod 04 och 06 och därför ignorerar REL40 alla dataord med annan kod.

A. Om koden är 04 så innebär detta att dataordet är en icke minnesrefererande instruktion, en absolut adress eller en konstant. Dataordet matas alltså ut oförändrat.

B. Om koden är 05 så är dataordet en relokerbar adress.

1. En adress med adresseringsmöjlighet i samma sida där instruktionen ligger (8 bit). Här adderas IST (IST anger begynnelseadressen till programmet) till innehållet av dataordet och avkortat resultat (de 8 minst signifikanta bitarna) matas ut.

2. En adress - (12 bit). Här adderas IST till dataordet som i föregående fall. Resultatet delas i två delar. Första delen de 4 mest signifikanta bitarna adderas till föregående dataord (med kod 04) och detta matas ut. Andra delen de 8minst signifikanta bitarna placeras i det aktuella dataordet.

3. En adress - extern referens. I detta fall är innehållet av dataordet (redan ökat med IST) större än ISN. Det innebär att vi refererar till en adress utanför vårt program. (1).

E X T E R N A R E F E R E N S E R

Om vi refererar till en adress utanför vårt program., markerar assemblern detta genom att reservera en extra cell som ligger omedelbart efter programmet. Cellens adress ersätter den externa adressen men med dess hjälp kan man senare få reda på den riktiga sdressen. Antalet extra celler bestäms av hur många externa adresser som finns i vårt program. BEL40 upptäcker inte de extra cellerna, istället träffar den på ett nytt program omedelbart efteråt. Den jämför alla adresser i vårt ursprungliga program

med begynnelse adressen för nästa program och om det visar sig att något av talen är större än den så är det en extern referens. Den externa referensen har blivit upptäckt, och nu måste REL40 finna en slutlig adress för den. Detta sker genom att flera vektorer, redan bildade av MAIN40 (huvudprogram för LOAD40), genomsökes. I EXTREF hittar REL40 namnet på vår externa referens. Namnet ligger på en plats som beräknas på följande sätt: den extra cellens adress (IA) minskas med värdet på begynnelse adressen för nästa program (ISN). Resultatet adderas till ett värde som anger begynnelse positionen i vektorn " EXTREF" för vårt extra fält (NEX).

IA = IA - ISN + NEX

Därefter läser REL40 igenom en annan vektor Glob, som innehåller alla globala symboler, och leter efter samma namn. Detta finns nämligen både i "EXTREF" och "GLOB". På samma adress som namnet ligger i "GLOB" fast i en annan vektor "IADR" finns den slutliga adressen för vår externa referens.

3.3 V A R N I N G S U T S K R I F T E R

REL40 utför också en del tester för att försäkra sig att alla instruktioner hamnar rätt vid laddningen. Det är nämligen så att 2 av INTEL 4040:s instruktioner förorsaka fel. Här följer en beskrivningen av de möjliga fallen:

1. Någon av instruktionerna JCN eller ISZ ligger på sista adressen (254 och 255) i en sida och deras destinationsadress pekar inte på nästa sida.
2. Destinationsadressen för instruktionen JCN eller ISZ ligger i en annan sida än själva instruktionerna (och inte på plats 254 och 255).

   Varningsutskrift på teletypen blir:

   Fall 1.

   Adress error for JCN or ISZ at XXX:377 CODE = XXX

   destination at XXX:XXX

   Fall 2.

   Adress error for JCN or ISZ at XXX:XXX CODE = XXX

   destination at XXX:XXX.

   Tack vare dessa utskrifter kan programmeraren göra nödvändiga ändringar.

4. RUTINEN MULT

Vi har skrivit ett program MULT, som multiplicerar 2 8-bits tal med varandra. Rutinen utnyttjar även subrutinerna ABSOL och ROT. En listning av programmen finns i appendix 2. Talen ligger i registren R0 - R3 och resultatet läggs i R4 - R7. Rutinen utnyttjar även registren R8 - R15. Talen ligger från början i 2-komplementform men att utnyttja en algoritm för denna form ger ett betydligt längre program än om man först tar absolut beloppet och sedan använder en algoritm där talen förutsättes vara positiva. Resultatets tecken lagras då i ett register och konvertering tillbaka till 2- komplementform sker senare. Resultatet skall eventuellt senare divideras i programmet DIV med ett 8-bits tal i 2-komplementform. Eftersom det då redan finns en rutin i MULT som tar absolutbeloppet utnyttjar man denna och divisionen utförs alltså också med positiva tal. Efter divisionen konverteras sedan resultatet till 2-komplementform.

Programmets utseende beror något på om talen är bråktal eller heltal. Första biten är en teckenbit, i vårt fall alltid en nolla och därefter följer 7 binära siffror. Resultatet av multiplikationen blir bara 14 binära siffror plus teckenbiten vilket visas på följande sätt:

7 bitar per maximala talet:

$$\sum_{i=1}^{7} 2^{i-1} = 127$$

$$127 * 127 = 16.129 < 16.383 = \sum_{i=1}^{14} 2^{i-1} \qquad (= 2^{14} - 1)$$

Är talen heltal skall en nolla skjutas in i första positionen, men eftersom talen i vårt fall är bråktal läggs en nolla in i sista positionen.

Talen som skall multipliceras består av 8 bitar.

Efter registren endast har fyra bitar får man dela upp multiplikationen enligt följande formel:

$$(A*2^4 + B) * (C*2^4 + D) = 2^8 * A*C + 2^4 * (A*D + B*C) + B*D$$

där A och C är de 4 första bitarna av talen och B och D de fyra sista bitarna.

Multiplikationen av 4-bitstalen utförs i subrutinen MULT4. Resultaten på 8 bitar läggs sedan in i 4 register där A*C ligger förskjutet 4 bitar till vänster i förhållande till A*D + B*C som i sin tur ligger 4 bitar till vänster om B*D. Eftersom talen är överlappade utförs addition med tillvaratagande av carryn.

Subrutinen MULT4 utförs på i prindip samma sätt som vid vanlig multiplikation vid handräkning, dvs man utgår från ena talet och adderar detta om lägsta siffror i det andra talet är 1. (I 10-systemet multiplicerar man första talet med lägsta siffran men i binära systemet finns ju bara 0 och 1, dvs addition eller inte addition)

Därefter skifter man första talet ett steg åt vänster och adderar detta om närmast högre position i tal 2 är en etta. Så fortsätter man tills man har gått igenom hela tal 2.

Ex. Multiplikation av talen 0011 och 1001 utförs på följande sätt:

```
 0011  *  1  =        0011
 0011  *  0  =       ----
 0011  *  0  =      -----
 0011  *  1  =   0011
                 --------
                 00110011
```

Observera att resultatet endast har 7 bitar.

Nu är ju registren endast på 4 bitar så därför läggs resul-

taten av additionerna in i två register. Tal 2. i detta fall 1001, högerskiftas och om en etta skiftas ut så adderas tal 1, dvs 0001, till eventuella tidigare delresultat i register R9. Sedan vänsterskiftas tal 1. en nolla förs då in i högra positionen och om högerskift av tal 2 ger en etta så adderas tal 1 igen. På det viset går man igenom gela tal 2. Om det vid additionen blir en carry så ökas R8 med 1. (I R8 läggs de fyra mestsignifikanta bitarna). När tal 1 vänsterskiftas så sparas eventuella ettor i ett register (R13). Detta vänsterskiftas efterhand också och adderas till de 4 mest signifikanta siffrorna (i R8) om vald position i tal 2 är 1. När MULT4 har anropats 4 ggr läggs talen in i R4 - R7.

Rutinen har testats på INTEL4040. Då användes även programmen INUTI och DISP se appendix 4.

5. R U T I N E N D I V

Rutinen utnyttjar även subrutinerna ABSOL och ROT se appendix 4. Förutsättningarna för divisionen är att ett 16-bits positivt bråktal, dividenden, ligger i registren R4 - R7 och ett 8-bits bråktal i 2-komplementform ligger i register R0 - R1.

Vi använder en algoritm för division utan återställning av Burks, (4).

Enligt denna algoritm kan man använda tal i 2-komplementform. Vi har dock omvandlat även divisorn till ett positivt tal först eftersom programmet blir kortare då. (Annars får man bl.a. ta exklusivt eller av teckenpositionerna och komplementera detta). Det finns för övrigt redan en rutin i MULT som tar absolutbeloppet. Ett eventuellt minustecken lagras i ett register och resultatet konverteras senare till 2-komplementform. Algoritmen förutsätter att dividenden är mindre än divisorn . Programmet kontrollerar först detta och om villkoret inte är uppfyllt så blir resultatet talet närmast under 1. Division med 0 ger också detta resultat.

Algoritmen går ut på att man först roterar dividenden ett steg till vänster (dvs. fördubblar talet) och sedan subtraherar de mest signifikanta bitarna med divisorn. Om resultatet är positivt är kvoten alltså större än 1/2 (och mindre än 1) och innehåller en etta i den positionen.

Därefter roteas talet till vänster igen, divisorn subtraheras och om resultatet är positivt så innehåller kvoten alltså även

1/4 osv. Om resultatet blev negativt i stället skall det vara en nolla i motsvarande position. Då skulle man egentligen addera divisorn, rotera dividenden och subtrahera divisorn (som nu är hälften så stor relativt dividenden). Samma sak utförs emellertid enklare genom att rotera dividenden och lägga till divisorn. Enligt algoritmen fortsätter man tills dividenden är lika lång som divisorn. Vårt program fortsätter dock ytterligare en gång. Är sista biten en etta så är resten mindre än 1/2 och kvoten höjs. Sedan högerskiftas talet. Ingen siffra går förlorad eftersom första biten alltid är noll.

Dividenden minskar från 16 bitar till 8 bitar efter divisionen och roteras hela tiden ett steg till vänster. Roterarett steg gör även resultatet så programmet blir kortare om vi lägger in detta i de bitar i dividenden som blir lediga efter hand.

Rutinen har testats på INTEL 4040. Då användes även programmen INUT2 och DISP se appendix 4.

R E F E R E N S E R

1. L. Andersson "Assembling and relocating programs for INTEL 8008 using PDP:15 as a host computer".

2. INTEL MCS-40 "Users manual".

3. "MACRO-15 Assembler".

4. P.E. Danielsson "Datorteknik,Binär Aritmetik".

5. Linking Loader, Chapter 4

APPENDIX 1

```
        .NOLST
HLT=1
BBS=2
LCR=3
OR4=4
OR5=5
AN6=6
AN7=7
DB0=10
DB1=11
SB0=12
SB1=13
EIN=14
DIN=15
RPM=16
WRM=340
WMP=341
WRR=342
WPM=343
WR0=344
WR1=345
WR2=346
WR3=347
SBM=350
RDM=351
RDR=352
ADM=353
RD0=354
RD1=355
RD2=356
RD3=357
CLB=360
CLC=361
IAC=362
CMC=363
CMA=364
RAL=365
RAR=366
TCC=367
DAC=370
TCS=371
SIC=372
DAA=373
KBP=374
DCL=375
NOP=0
R0=0
R1=1
R2=2
R3=3
R4=4
R5=5
R6=6
R7=7
R8=10
R9=11
R10=12
R11=13
R12=14
R13=15
R14=16
R15=17
        .DEFIN  INC,R
```

```
17&R+140
.ENDM
.DEFIN  ADD,R
17&R+200
.ENDM
.DEFIN  SUB,R
17&R+220
.ENDM
.DEFIN  LD,R
17&R+240
.ENDM
.DEFIN  XCH,R
17&R+260
.ENDM
.DEFIN  SRC,R
16&R+41
.ENDM
.DEFIN  FIN,R
16&R+60
.ENDM
.DEFIN  JIN,R
16&R+61
.ENDM
.DEFIN  ISZ,R,A
R&17+160
A
.ENDM
.DEFIN  JMS,A
120
A
.ENDM
.DEFIN  JUN,A
100
A
.ENDM
.DEFIN  FIM,R,DATA
R&16+40
DATA
.ENDM
.DEFIN  BBL,DATA
DATA&17+300
.ENDM
.DEFIN  LDM,DATA
DATA&17+320
.ENDM
.DEFIN  JTT,A
21
A
.ENDM
.DEFIN  JTC,A
22
A
.ENDM
.DEFIN  JTTC,A
23
A
.ENDM
.DEFIN  JTZ,A
24
A
.ENDM
.DEFIN  JTTZ,A
25
A
```

```
.ENDM
.DEFIN   JTCZ,A
26
A
.ENDM
.DEFIN   JTTCZ,A
27
A
.ENDM
.DEFIN   JFT,A
31
A
.ENDM
.DEFIN   JFC,A
32
A
.ENDM
.DEFIN   JFTC,A
33
A
.ENDM
.DEFIN   JFZ,A
34
A
.ENDM
.DEFIN   JFTZ,A
35
A
.ENDM
.DEFIN   JFCZ,A
36
A
.ENDM
.DEFIN   JFTCZ,A
37
A
.ENDM
.DEFIN   JMP,A
30
A
.ENDM
.DEFIN   CHANGE,RX,RY,RZ,RW
LD       RX
XCH      RZ
LD       RY
XCH      RW
JMS      MULT4
.ENDM
.LST
```

APPENDIX 2

```
C       PROGRAM MAIN40
C
C       MAIN PROGRAM FOR LOAD40
C       AUTHOR LEIF ANDERSSON 1974-11-13
C       REVISED JANUSZ MISZCZUK 1975-09-03
C
C       SUBROUTINES REQUIRED
C               REL40
C               SNAM
C               COMND
C               MATCH
C               GLOBLS
C               DATINF
C               RADASC
C               ASCRAD
C               LPOUT
C               PPOUT
C               DKOUT
C               RADR
C               RID
C               PUTCHA
C               RLINE
C               FAC
C               INIT
C               READ
C               WRITE
C
        DOUBLE INTEGER FILNAM,GLOB,EXTREF,NAME,NOMTCH,FN,DNAM,
        1                BUFF,LIBR,A,B,ALTM
        DIMENSION FILNAM(25),GLOB(100),IADR(100),EXTREF(100),
        1          NAME(100),IST(100),NEX(100),NOMTCH(100),FN(2),
        2          DNAM(2),BUFF(16)
        COMMON FILNAM,GLOB,IADR,EXTREF,NOMTCH,NAME,IST,NEX
        DATA FN(2)/' BIN'/,
        1    DNAM(2)/' ABS'/,
        2    LIBR/'NONE'/,
        3    ALTM/#D764000000000/
C
C       GET COMMANDS
C
 10     WRITE(9,100)
 100    FORMAT(1X'LOAD40 V1A')
        IGLOB=0
        IEX=0
        CALL COMND(LIBR,GLOB,IADR,IGLOB,IST1,FILNAM,NFIL,DNAM(1),IND)
        IG1=IGLOB+1
        IST(1)=IST1
        IU=1
        NEX(1)=1
C
C       GET GLOBAL SYMBOLS AND EXTERNAL REFERENCES
C
        DO 20 I=1,NFIL
        FN(1)=FILNAM(I)
        CALL SEEK(2,FN)
 15     CALL GLOBLS(IST1,NAME(IU),GLOB,IADR,IGLOB,EXTREF,IEX,IEOF)
        IF(IEOF)99,20,16
 16     IU=IU+1
        NEX(IU)=IEX+1
        IST(IU)=IST1
        GO TO 15
 20     CALL CLOSE(2)
        IG2=IGLOB
```

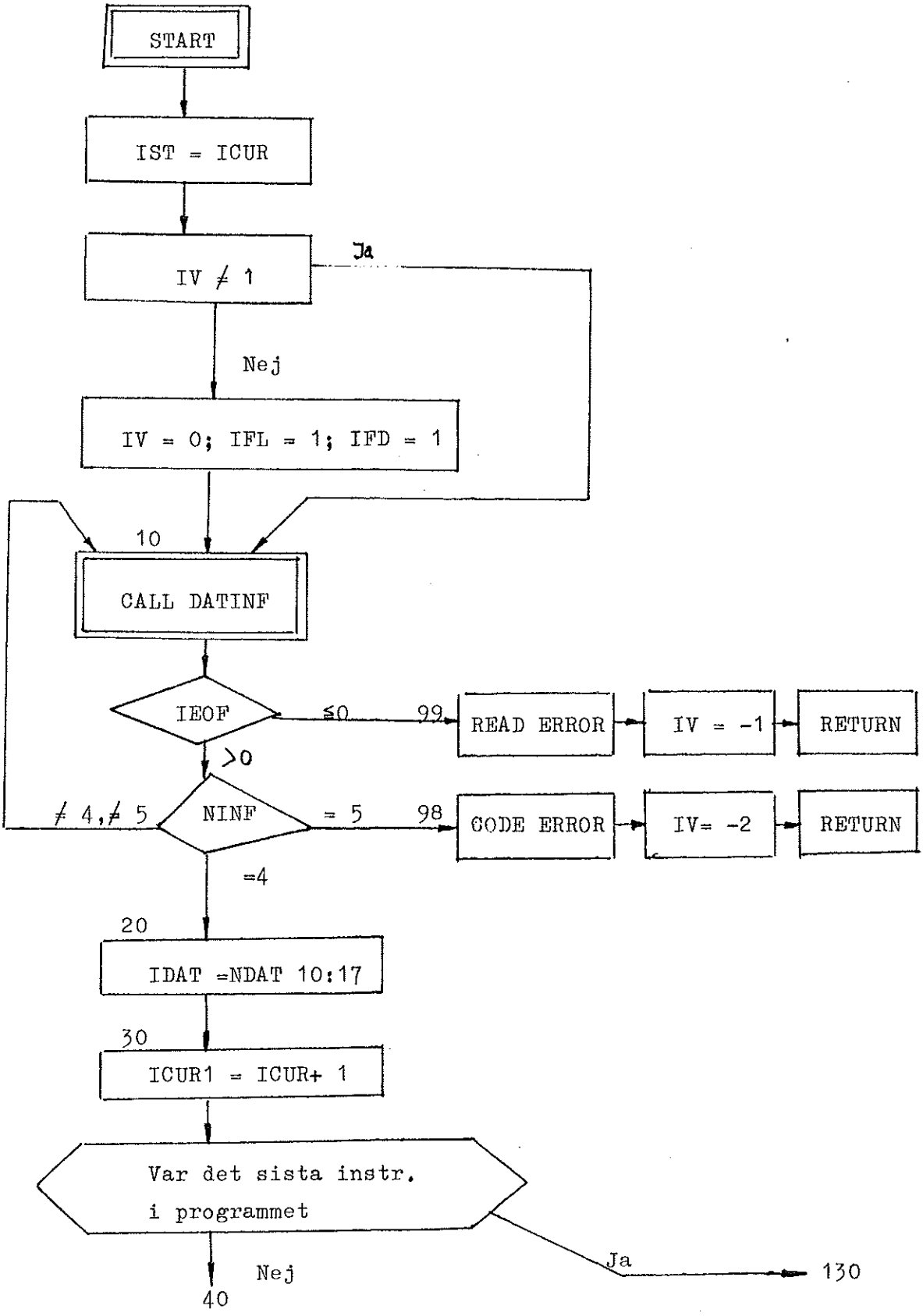
```
C
C       GET THE GLOBALS FOR THE LIBRARY PROGRAMS IF ANY ARE NEEDED
C
        CALL MATCH(EXTREF,IEX,GLOB,IGLOB,NOMTCH,IN,M)
        IF(IN .EQ. 0)GO TO 50
        IF(LIBR .EQ. 'NONE')GO TO 40
        NFIL=NFIL+1
        FILNAM(NFIL)=LIBR
        FN(1)=LIBR
        CALL SEEK(2,FN)
C
 25     IG=IGLOB
        IE=IEX
        IST1=IST(IU)
 26     CALL GLOBLS(IST1,NAME(IU),GLOB,IADR,IG,EXTREF,IE,IEOF)
        IF(IEOF)99,29,27
 27     IN1=IN
        CALL MATCH(NOMTCH,IN1,GLOB,IG,NOMTCH,IN,M)
        IF(M .EQ. 0)GO TO 25
        CALL MATCH(EXTREF,IE,GLOB,IG,NOMTCH,IN,M)
        IU=IU+1
        IEX=IE
        IGLOB=IG
        NEX(IU)=IEX+1
        IST(IU)=IST1
        IF(IN .NE. 0)GO TO 26
 29     CALL CLOSE(2)
        IG2=IGLOB
        IF(IN .EQ. 0)GO TO 50
C
C       GET THE UNRESOLVED GLOBALS IF ANY
C
 40     WRITE(9,110)
 110    FORMAT(1X'UNRESOLVED GLOBALS')
        DO 45 II=1,IN
        CALL RADASC(NOMTCH(II),A,B)
        WRITE(9,120)A,B,ALTM
 120    FORMAT(1X,A5,A5,A1)
 43     CALL RLINE(9,8,BUFF,0)
        I=1
        CALL RADR(I,IAD,IA)
        IF(IA .EQ. 1)GO TO 44
        WRITE(9,130)
 130    FORMAT(1X'SYNTAX ERROR, RETYPE ADDRESS,')
        GO TO 43
 44     IGLOB=IGLOB+1
        GLOB(IGLOB)=NOMTCH(II)
 45     IADR(IGLOB)=IAD
C
C       WRITE LOAD MAP AND GLOBAL SYMBOL TABLE
C
 50     WRITE(6,200)
 200    FORMAT(1X'FILE NAMES')
        IF(LIBR .NE. 'NONE') NFIL=NFIL-1
        DO 55 I=1,NFIL
 55     WRITE(6,210)FILNAM(I)
 210    FORMAT(5XA5)
C
        WRITE(6,220)LIBR
 220    FORMAT('0'/1X'LIBRARY:  ',A5)
C
        IL1=MOD(IST(1),256)
        IH1=IST(1)/256
        IST1=IST(IU)-1
```

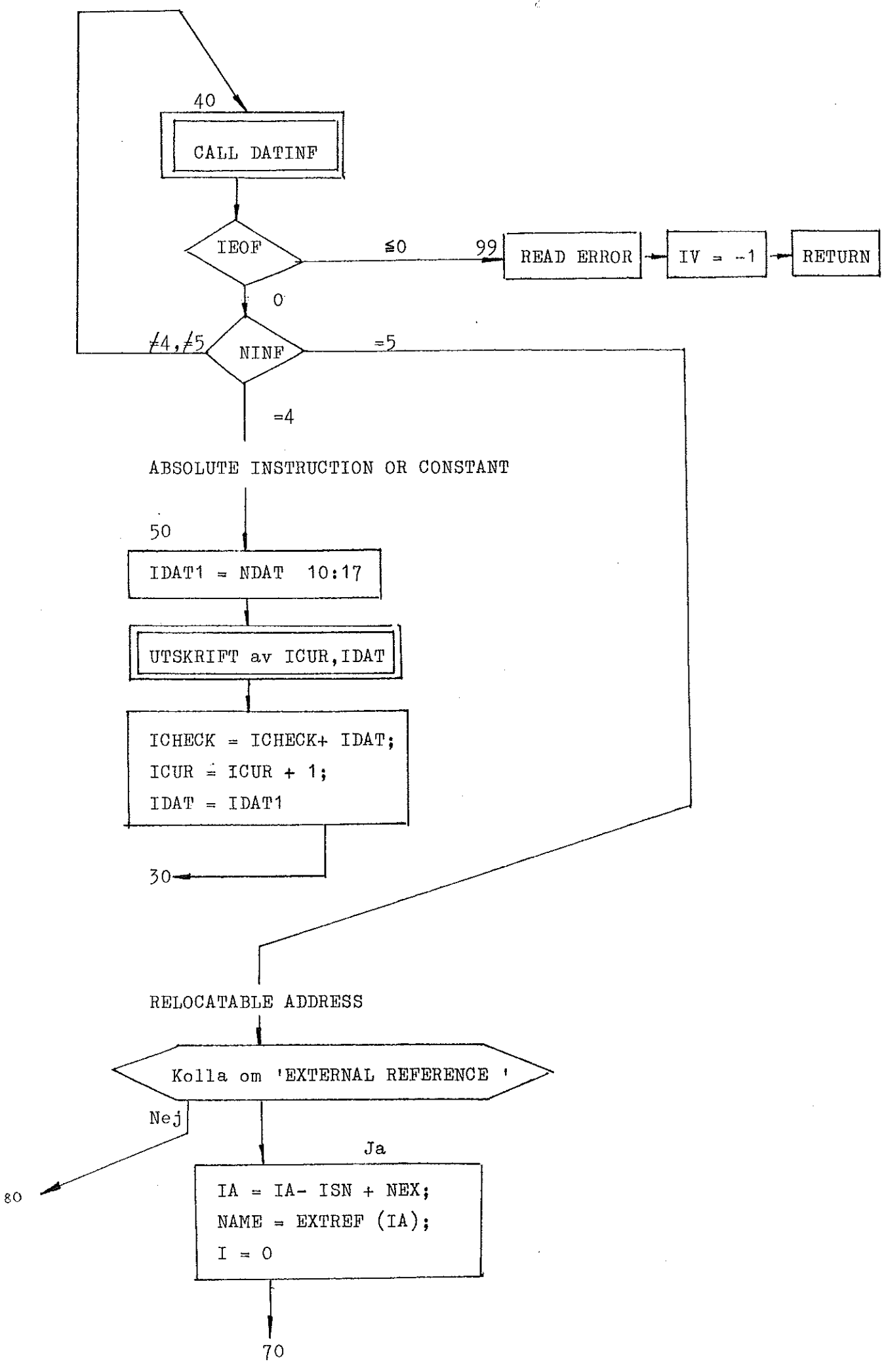
```
      IL2=MOD(IST1,256)
      IH2=IST1/256
      WRITE(6,230)IH1,IL1,IH2,IL2
 230  FORMAT('0'/1X'MEMORY REQUIRED:  ',O3,':',O3,' - ',
     1      O3,':',O3)
C
      WRITE(6,240)
 240  FORMAT('0'/1X'LOAD MAP')
      IU=IU-1
      DO 60 I=1,IU
      CALL RADASC(NAME(I),A,B)
      IL=MOD(IST(I),256)
      IH=IST(I)/256
 60   WRITE(6,250)A,B,IH,IL
 250  FORMAT(5XA5,A5,O3,':',O3)
C
      IF(IG1 .GT. IG2)GO TO 70
      WRITE(6,260)
 260  FORMAT('0'/1X'GLOBAL SYMBOL TABLE')
      DO 65 I=IG1,IG2
      CALL RADASC(GLOB(I),A,B)
      IL=MOD(IADR(I),256)
      IH=IADR(I)/256
 65   WRITE(6,250)A,B,IH,IL
C
 70   WRITE(6,270)
 270  FORMAT('1')
C
C     START RELOCATING
C
      IF(IND .NE. 2)GO TO 73
      CALL ENTER(1,DNAM)
      WRITE(1,230)IH1,IL1,IH2,IL2
 73   IF(IND .NE. 1)GO TO 75
      CALL INIT(7,1)
      IFP=1
      CALL PPOUT(O377,IFP)
C
 75   IV=1
      IFIL=0
      ICUR=IST(1)
      ICHECK=0
      DO 80 I=1,IU
      II=I+1
      CALL SNAM(FILNAM,IFIL,NAME(I),ITRIP,IERR)
      IF(IERR .LT. 0)GO TO 99
      CALL REL40(ICUR,IST(II),GLOB,IADR,EXTREF,NEX(I),
     1          IND,IV,ICHECK,ITRIP,IFP)
      IF(IV .EQ. -1)GO TO 99
      IF(IV .EQ. -2)GO TO 98
 80   CONTINUE
C
      CALL CLOSE(2)
      IFL=-1
      IFD=-1
      CALL LPOUT(I,ICUR,IFL)
      IF(IND .NE. 2)GO TO 85
      CALL DKOUT(I,IFD)
      CALL CLOSE(1)
 85   IF(IND .NE. 1)GO TO 90
      CALL PPOUT(ICHECK,IFP)
      IFP=-1
      CALL PPOUT(I,IFP)
      CALL CLOSE(7)
```

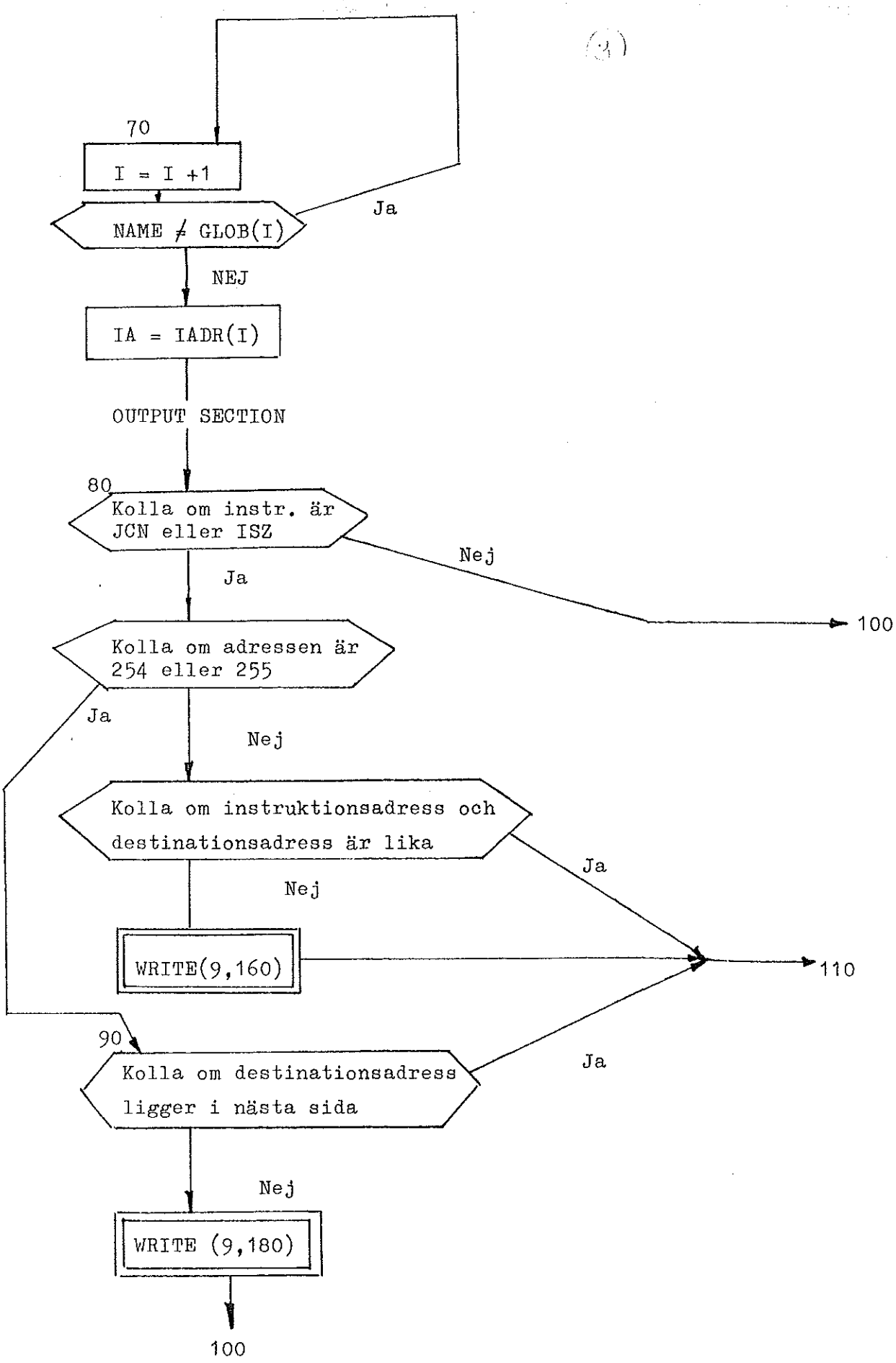
```
 90      WRITE(6,270)
         GO TO 10
C
         READ(2)I
         WRITE(7)I
 99      WRITE(9,300)
 300     FORMAT(1X'READ ERROR')
 98      WRITE(9,310)
 310     FORMAT(1X'CODE ERROR')
         GO TO 10
         END
```

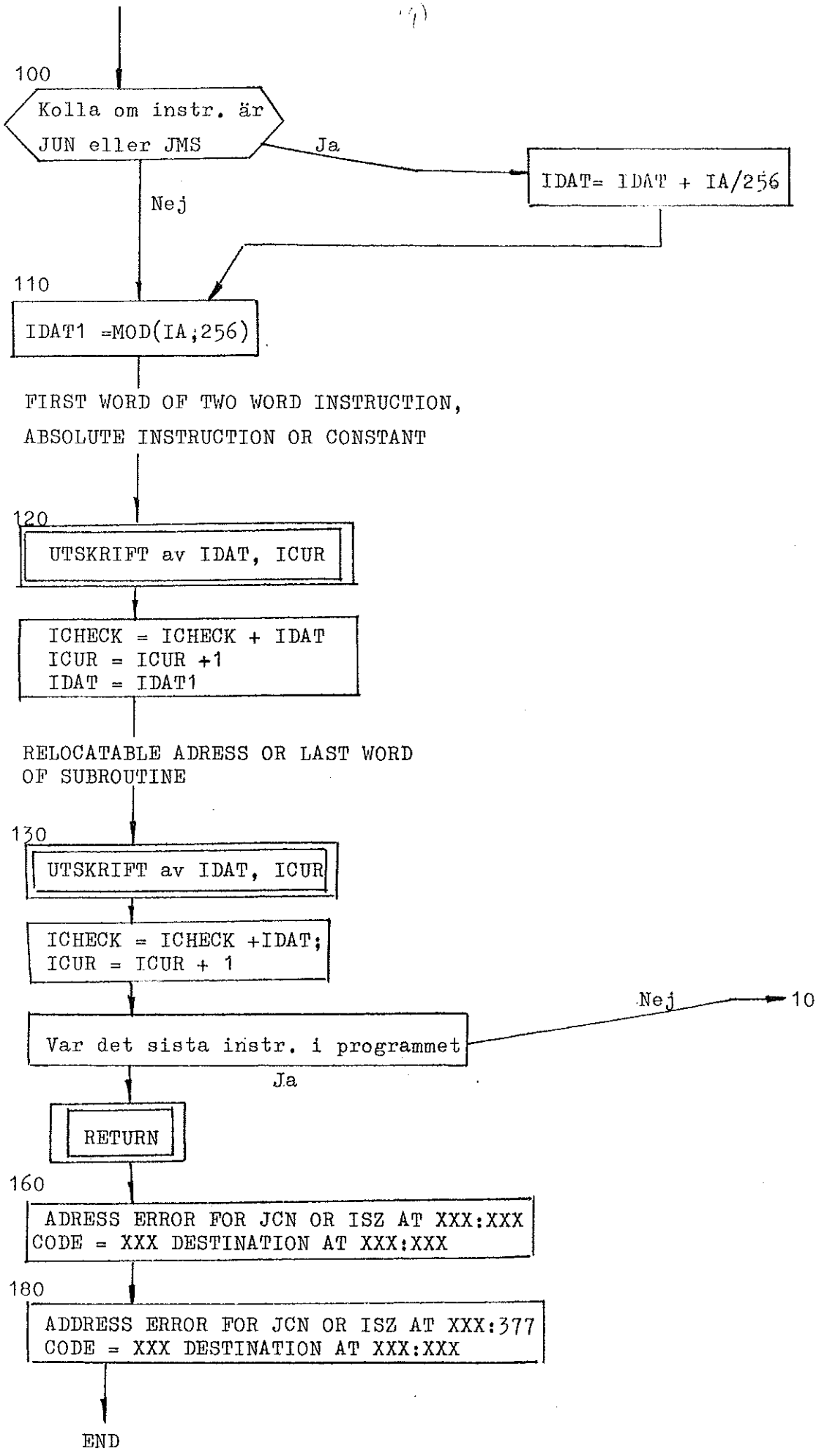
APPENDIX 2

FLÖDESCHEMA - REL40




40
CALL DATINF
IEOF
≤0
99
READ ERROR
IV = -1
RETURN
0
≠4,≠5
NINF
=5
=4
ABSOLUTE INSTRUCTION OR CONSTANT
50
IDAT1 = NDAT 10:17
UTSKRIFT av ICUR,IDAT
ICHECK = ICHECK+ IDAT;
ICUR = ICUR + 1;
IDAT = IDAT1
30
RELOCATABLE ADDRESS
Kolla om 'EXTERNAL REFERENCE '
Nej
Ja
IA = IA- ISN + NEX;
NAME = EXTREF (IA);
I = 0
70
80


70
I = I +1
NAME ≠ GLOB(I)
Ja
NEJ
IA = IADR(I)
OUTPUT SECTION
80
Kolla om instr. är JCN eller ISZ
Nej
Ja
100
Kolla om adressen är 254 eller 255
Ja
Nej
Kolla om instruktionsadress och destinationsadress är lika
Ja
Nej
WRITE(9,160)
110
90
Kolla om destinationsadress ligger i nästa sida
Ja
Nej
WRITE (9,180)
100


100
Kolla om instr. är JUN eller JMS
Ja
IDAT= IDAT + IA/256
Nej
110
IDAT1 =MOD(IA;256)
FIRST WORD OF TWO WORD INSTRUCTION, ABSOLUTE INSTRUCTION OR CONSTANT
120
UTSKRIFT av IDAT, ICUR
ICHECK = ICHECK + IDAT
ICUR = ICUR +1
IDAT = IDAT1
RELOCATABLE ADRESS OR LAST WORD OF SUBROUTINE
130
UTSKRIFT av IDAT, ICUR
ICHECK = ICHECK +IDAT;
ICUR = ICUR + 1
Var det sista instr. i programmet
Nej
10
Ja
RETURN
160
ADRESS ERROR FOR JCN OR ISZ AT XXX:XXX
CODE = XXX DESTINATION AT XXX:XXX
180
ADDRESS ERROR FOR JCN OR ISZ AT XXX:377
CODE = XXX DESTINATION AT XXX:XXX
END

```
      SUBROUTINE REL40(ICUR,ISN,GLOB,IADR,EXTREF,NEX,
     1                 IND,IV,ICHECK,ITRIP,IFP)
C
C     RELOCATION SUBROUTINE FOR LOAD40
C     AUTHOR JANUSZ MISZCZUK 1975-09-03
C
C     THE INPUT AND OUTPUT FILE MUST BE OPENED BEFORE THE CALL
C     AND INIT MUST BE DONE ON THE PP
C
C     ICUR -  CURRENT RELOCATION ADDRESS
C     ISN -   START ADDRESS OF NEXT PROGRAM UNIT
C     GLOB -  DOUBLE INTEGER VECTOR WITH GLOBAL NAMES
C     IADR -  VECTOR WITH THE ADDRESSES OF THE GLOBALS
C     EXTREF- DOUBLE INTEGER VECTOR WITH THE NAMES OF EXTERNAL REFS
C     NEX -   INDEX OF THE FIRST EXTERNAL REFERENCE FOR THE CURRENT
C             PROGRAM UNIT
C     IND -   OUTPUT INDICATOR          0: LP ONLY
C                                       1: PP+LP
C                                       2: DK+LP
C     IV -    SET IV=1 ON THE FIRST CALL, IV IS RETURNED 0
C             NORMALLY, -1 IF READ ERROR AND -2 IF CODE ERROR
C     ICHECK- CHECKSUM
C     ITRIP-  PARAMETER FOR DATINF, DO NOT CHANGE IT.
C     IFP -   THIS IS THE PARAMETER IFP OF PPOUT.
C
C     SUBROUTINES REQUIRED
C             DATINF
C             (READ)
C             LPOUT
C             DKOUT
C             PPOUT
C             (WRITE)
C
      DOUBLE INTEGER GLOB,EXTREF,NAME
      DIMENSION GLOB(1),EXTREF(1),IADR(1)
C
      IST=ICUR
      IF(IV .NE. 1)GO TO 10
      IV=0
      IFL=1
      IFD=1
C
 10   CALL DATINF(NINF,NDAT,ITRIP,IEOF)
      IF(IEOF)99,99,15
 15   IF(NINF .EQ. 4)GO TO 20
      IF(NINF .EQ. 5)GO TO 98
      GO TO 10
 20   IDAT=NDAT(10:17)
 30   ICUR1=ICUR+1
C
C
      IF(ICUR1 .EQ. ISN) GO TO 130
 40   CALL DATINF(NINF,NDAT,ITRIP,IEOF)
      IF(IEOF)99,99,45
 45   IF(NINF .EQ. 4) GO TO 50
      IF(NINF .EQ. 5) GO TO 60
      GO TO 40
C
C     ABSOLUTE INSTRUCTION OR CONSTANT
C
 50   IDAT1=NDAT(10:17)
C
C
```

```
      IF(IND .EQ. 1)CALL PPOUT(IDAT,IFP)
      IF(IND .EQ. 2)CALL DKOUT(IDAT,IFD)
      CALL LPOUT(IDAT,ICUR,IFL)
C
      ICHECK=ICHECK+IDAT
      ICUR=ICUR+1
      IDAT=IDAT1
      GO TO 30
C
C     RELOCATABLE ADDRESS
C
 60   IA=NDAT+IST
      IF(IA .LT. ISN) GO TO 80
C
C     EXTERNAL REFERENCE
C
      IA=IA-ISN+NEX
      NAME=EXTREF(IA)
      I=0
 70   I=I+1
      IF(NAME .NE. GLOB(I)) GO TO 70
      IA=IADR(I)
C
C     OUTPUT SECTION
C
 80   ID=IDAT/16
      IC1=MOD(ICUR1,256)
      MSIC1=ICUR1/256
      IA1=IA/256
      IA2=MOD(IA,256)
      IF(ID .NE. 1 .AND. ID .NE. 7) GO TO 100
      IF(IC1 .EQ. 255) GO TO 90
      IF (IA1-MSIC1 .EQ. 0) GO TO 110
      WRITE(9,160) MSIC1,IC1,IDAT,IA1,IA2
      GO TO 110
 90   IF(IA1-MSIC1 .EQ. 1) GO TO 110
      WRITE(9,180) MSIC1,IDAT,IA1,IA2
C
C
 100  IF(IDAT .EQ. ¤100 .OR. IDAT .EQ. ¤120)IDAT=IDAT+IA/256
 110  IDAT1=MOD(IA,256)
C
C     FIRST WORD OF TWO WORD INSTRUCTION , ABSOLUTE
C     INSTRUCTION OR CONSTANT
C
 120  IF(IND .EQ. 1)CALL PPOUT(IDAT,IFP)
      IF(IND .EQ. 2)CALL DKOUT(IDAT,IFP)
      CALL LPOUT(IDAT,ICUR,IFL)
C
      ICHECK=ICHECK+IDAT
      ICUR=ICUR+1
C
C     RELOCATABLE ADDRESS OR THE LAST WORD OF
C     SUBROUTINE
C
      IDAT=IDAT1
 130  IF(IND .EQ. 1)CALL PPOUT(IDAT,IFP)
      IF(IND .EQ. 2)CALL DKOUT(IDAT,IFP)
      CALL LPOUT(IDAT,ICUR,IFL)
      ICHECK=ICHECK+IDAT
      ICUR=ICUR+1
      IF(ICUR .EQ. ISN) RETURN
      GO TO 10
C
```

```
C         CODE ERROR
C
 98       IV=-2
          RETURN
C
C         READ ERROR
C
 99       IV=-1
          RETURN
C
 160      FORMAT(1X'ADDRESS ERROR FOR JCN OR ISZ AT  '
          1       ,03,':',03,'  CODE=',03/1X'DESTINATION',
          2       ' AT  ',03,':',03)
 180      FORMAT(1X'ADDRESS ERROR FOR JCN OR ISZ AT  '
          1       ,03,':377  CODE=',03/1X'DESTINATION AT '
          2       ,03,':',03)
          END
```

APPENDIX 3

```
FILE NAMES
     MULT
     ABSOL
     ROT


LIBRARY:  NONE


MEMORY REQUIRED:  000:000 - 000:243


LOAD MAP
     MULT       000:000
     ABSOL      000:204
     ROT        000:226


GLOBAL SYMBOL TABLE
     MULT       000:000
     ABSOL      000:204
     ROT        000:226
```

```
000:000   044   000   046   000   056   000   240   034
000:010   014   241   024   132   242   034   022   243
000:020   024   132   120   204   242   260   262   243
000:030   261   263   120   204   241   272   243   273
000:040   120   133   251   267   250   266   241   272
000:050   242   273   120   133   250   265   246   361
000:060   211   266   032   065   145   240   272   243
000:070   273   120   133   246   361   211   266   032
000:100   103   361   145   245   210   265   032   111
000:110   144   240   272   242   273   120   133   250
000:120   264   245   361   211   265   032   130   144
000:130   120   226   300   050   000   054   000   335
000:140   274   100   155   255   361   365   275   252
000:150   365   272   032   155   155   253   366   273
000:160   032   175   361   250   215   270   251   361
000:170   212   271   032   175   150   254   024   203
000:200   154   100   143   300   330   361   200   032
000:210   225   156   240   364   260   241   364   361
000:220   362   261   032   225   140   300   361   247
000:230   365   267   246   365   266   245   365   265
000:240   244   365   264   300
```

APPENDIX 4

```
PAGE    1      MULT    SRC      MULT

    1                                       .TITLE MULT
    2                                       .GLOBL MULT,ABSOL,ROT
    3                                       /
    4                                       /
    5                                       /PROGRAMMET MULTIPLICERAR TVA 8-BITS TAL I 2-KOMPL.
    6                                      /SOM LIGGER I R0-R3.
    7                                      /RESULTATET LAGGS I R4-R7. RUTINEN UTNYTTJAR AVEN
    8                                      /R8-R15. TALEN DELAS UPP I 4-BITS TAL ENLIGT FORMELN:
    9                                      / (A*2**4+B)*(C*2**4+D)=A*C*2**8+2**4*(A*D+B*C)+B*D
   10                                      /MAN UTNYTTJAR SURRUTINEN "MULT4" SOM MULTIPLICERAR
   11                                      /TVA 4-BITS TAL MED VARANDRA. ("MULT4" HOGERSKIFTAR
   12                                      /TAL 2 OCH OM DEN UTSKIFTADE SIFFRAN AR 1 SA ADDERAS
   13                                      /TAL 1 . DAREFTER VANSTERSKIFFAS TAL 1 OCH RUTINEN
   14                                      /GENOMLOPS IGEN TILLS MULTIPLIKATIONEN AR SLUT).
   15                                      /RESULTATET BLIR ALLTID POSITIVT.ANTALET MINUSTECKEN
   16                                      /I FAKTORERNA LAGRAS I R14.PROGRAMMET DIV AR AVSETT
   17                                      /ATT KORAS EFTER MULT OCH DETTA PROGRAM AVSLUTAS MED
   18                                      /ATT GORA RESULTATET NEGATIVT OM R14 AR UDDA.
   19                                      /
   20      00000 R              MULT       FIM     R4,0
           00000 R 000044 A *G
           00001 R 000000 A *G
   21                                      FIM     R6,0
           00002 R 000046 A *G
           00003 R 000000 A *G
   22                                      FIM     R14,0
           00004 R 000056 A *G
           00005 R 000000 A *G
   23                                      LD      R0      /TESTA OM NAGOT 8-BITS TAL AR 0.
           00006 R 000240 A *G
   24                                                      /I SA FALL HOPPA UT
   25                                      JFZ     FZERO1
           00007 R 000034 A *G
           00010 R 000014 R *G
   26                                      LD      R1
           00011 R 000241 A *G
   27                                      .EJECT
```

```
                MULT    SRC     MULT
28                                      JTZ     OUT
        00012 R 000024 A *G
        00013 R 000132 R *G
29      00014 R                 FZERO1  LD      R2
        00014 R 000242 A *G
30                                      JFZ     FZERO2
        00015 R 000034 A *G
        00016 R 000022 R *G
31                                      LD      R3
        00017 R 000243 A *G
32                                      JTZ     OUT
        00020 R 000024 A *G
        00021 R 000132 R *G
33      00022 R                 FZERO2  JMS     ABSOL
        00022 R 000120 A *G
        00023 R 000204 E *G
34                                      LD      R2      /TALEN BYTS HAR SA ATT ABSOL KAN TA
        00024 R 000242 A *G
35                                      XCH     R0      /ABSOLUTBELOPPET AV NASTA TAL.
        00025 R 000260 A *G
36                                      XCH     R2
        00026 R 000262 A *G
37                                      LD      R3
        00027 R 000243 A *G
38                                      XCH     R1
        00030 R 000261 A *G
39                                      XCH     R3
        00031 R 000263 A *G
40                                      JMS     ABSOL   /TA ABSOLOLUTBELOPPET AV NASTA TAL.
        00032 R 000120 A *G
        00033 R 000204 E *G
41                                      .EJECT
```

```
 42                                                  CHANGE  R1,R3,R10,R11   /R1 OCH R3 LAGGS I R10 OCH R11
                                    *G
          00034 R 000241 A          *G
                                    *G
          00035 R 000272 A          *G
                                    *G
          00036 R 000243 A          *G
                                    *G
          00037 R 000273 A          *G
                                    *G
          00040 R 000120 A          *G
          00041 R 000133 R          *G
 43                                                                 /OCH MULT4 ANROPAS SOM MULTIPLICERAR
 44                                                                 /R10 OCH R11 OCH LAGGER RESULTATET I
 45                                                                 /R8 OCH R9.
 46                                                  LD      R9
          00042 R 000251 A          *G
 47                                                  XCH     R7      /LAGG IN PRODUKTEN I RATT REGISTER.
          00043 R 000267 A          *G
 48                                                  LD      R8
          00044 R 000250 A          *G
 49                                                  XCH     R6
          00045 R 000266 A          *G
 50                                                  CHANGE  R1,R2,R10,R11   /B*C ADDERAS TILL R5 OCH R6
                                    *G
          00046 R 000241 A          *G
                                    *G
          00047 R 000272 A          *G
                                    *G
          00050 R 000242 A          *G
                                    *G
          00051 R 000273 A          *G
                                    *G
          00052 R 000120 A          *G
          00053 R 000133 R          *G
 51                                                                 /OCH MULT4 ANROPAS.
 52                                                  .EJECT
```

```
 53                                          LD      R8
          00054 R 000250 A *G
 54                                          XCH     R5
          00055 R 000265 A *G
 55                                          LD      R6
          00056 R 000246 A *G
 56       00057 R 000361 A                   CLC
 57                                          ADD     R9
          00060 R 000211 A *G
 58                                          XCH     R6
          00061 R 000266 A *G
 59                                          JFC     NOCY1
          00062 R 000032 A *G
          00063 R 000065 R *G
 60                                          INC     R5      /INGEN CARRY TILL R4 MOJLIG.STORSTA
          00064 R 000145 A *G
 61                                                          /MOJLIGA TAL I R5 OCH R6 AR 1110 1111.
 62       00065 R                    NOCY1   CHANGE  R0,R3,R10,R11   /A*D ADDERAS TILL R5 OCH R6.
                          *G
          00065 R 000240 A *G
                          *G
          00066 R 000272 A *G
                          *G
          00067 R 000243 A *G
                          *G
          00070 R 000273 A *G
                          *G
          00071 R 000120 A *G
          00072 R 000133 R *G
 63                                          LD      R6
          00073 R 000246 A *G
 64       00074 R 000361 A                   CLC
 65                                          ADD     R9
          00075 R 000211 A *G
 66                                          .EJECT
```

```
67                                          XCH     R6
          00076 R 000266 A *G
68                                          JFC     NOCY2
          00077 R 000032 A *G
          00100 R 000103 R *G
69        00101 R 000361 A                  CLC
70                                          INC     R5      /INGEN RISK ATT R5 BLIR 0.
          00102 R 000145 A *G
71        00103 R                   NOCY2   LD      R5
          00103 R 000245 A *G
72                                          ADD     R8
          00104 R 000210 A *G
73                                          XCH     R5
          00105 R 000265 A *G
74                                          JFC     NOCY3
          00106 R 000032 A *G
          00107 R 000111 R *G
75                                          INC     R4
          00110 R 000144 A *G
76        00111 R                   NOCY3   CHANGE  R0,R2,R10,R11   /A*C ADDERAS TILL R4 OCH R5.
                                    *G
          00111 R 000240 A *G
                                    *G
          00112 R 000272 A *G
                                    *G
          00113 R 000242 A *G
                                    *G
          00114 R 000273 A *G
                                    *G
          00115 R 000120 A *G
          00116 R 000133 R *G
77                                          LD      R8
          00117 R 000250 A *G
78                                          XCH     R4
          00120 R 000264 A *G
79                                          .EJECT
```

```
60                                          LD      R5
          00121 R 000245 A *G
61        00122 R 000361 A                  CLC
62                                          ADD     R9
          00123 R 000211 A *G
63                                          XCH     R5
          00124 R 000265 A *G
64                                          JFC     NOCY4
          00125 R 000032 A *G
          00126 R 000130 R *G
65                                          INC     R4
          00127 R 000144 A *G
66        00130 R                 NOCY4     JMS     ROT
          00130 R 000120 A *G
          00131 R 000205 E *G
67        00132 R                 OUT       BBL     0
          00132 R 000300 A *G
68        00133 R                 MULT4     FIM     R8,0      /MULT4 MULTIPLICERAR 2 4-BITARS TAL
          00133 R 000050 A *G
          00134 R 000000 A *G
69                                                            /SOM FINNS I R10 OCH R11.
70                                                            /RESULTATET LAGRAS I R5 OCH R9.
71                                          FIM     R12,0
          00135 R 000054 A *G
          00136 R 000000 A *G
72                                          LDM     -3        /RAKNAREN R12 SATTS = -3.
          00137 R 000335 A *G
73                                          XCH     R12
          00140 R 000274 A *G
74                                          JUN     NOSHIFT   /INGA VANSTERSKIFT FORSTA GANGEN.
          00141 R 000100 A *G
          00142 R 000155 R *G
75        00143 R                 SHIFT     LD      R13       /I R13 LAGRAS DE 4 MEST SIGNIFIKANTA
          00143 R 000255 A *G
76                                          .EJECT
```

```
 97                                                  /BITARNA SOM  UPPKOMMER GENOM VANSTER-
 98                                                  /SKIFTEN.AR 0 FRAN BORJAN.
 99                                                  /INGEN CARRY VID VANSTERSKIFT.
100        00144 R 000361 A             CLC
101        00145 R 000365 A             RAL
102                                     XCH     R13
           00146 R 000275 A *G
103                                     LD      R10
           00147 R 000252 A *G
104        00150 R 000365 A             RAL
105                                     XCH     R10
           00151 R 000272 A *G
106                                     JFC     NOSHIFT /TA VARA PA CARRYN
           00152 R 000032 A *G
           00153 R 000155 R *G
107                                     INC     R13
           00154 R 000155 A *G
108        00155 R              NOSHIFT LD      R11     /HOGERSKIFTA TAL 2.
           00155 R 000253 A *G
109
110                                                 /OM VALD TECKENBIT AR 0
111                                                 /INGEN ADDITION.
111        00156 R 000366 A             RAR
112                                     XCH     R11
           00157 R 000273 A *G
113                                     JFC     NOADD   /OM CARRYN AR 0 INGEN ADDITION.
           00160 R 000032 A *G
           00161 R 000175 R *G
114        00162 R 000361 A             CLC
115                                     LD      R8      /ADDERA TILL DE 4 MEST
           00163 R 000250 A *G
116                                                 /SIGNIFIKANTA BITARNA.
117                                     ADD     R13
           00164 R 000215 A *G
118                                     XCH     R8
           00165 R 000270 A *G
119                                     .EJECT
```

```
120                                          LD      R9      /ADDERA TILL DE 4 MINST
          00166 R 000251 A *G
121                                                          /SIGNIFIKANTA BITARNA.
122       00167 R 000361 A                   CLC
123                                          ADD     R10
          00170 R 000212 A *G
124                                          XCH     R9
          00171 R 000271 A *G
125                                          JFC     NOADD   /LAGG TILL CARRYN.
          00172 R 000032 A *G
          00173 R 000175 R *G
126                                          INC     R8
          00174 R 000150 A *G
127       00175 R                NOADD       LD      R12     /FARDIGT ?
          00175 R 000254 A *G
128                                          JTZ     OUTM
          00176 R 000024 A *G
          00177 R 000203 R *G
129                                          INC     R12
          00200 R 000154 A *G
130                                          JUN     SHIFT
          00201 R 000100 A *G
          00202 R 000143 R *G
131       00203 R                OUTM        BBL     0
          00203 R 000300 A *G
132             000000 A                     .END
          00204 R 000204 E *E
          00205 R 000205 E *E
              SIZE=00206     NO ERROR LINES
```

```
 1                                          .TITLE  DIV
 2                                          .GLOBL  DIVI,ROT,ABSOL
 3                                          /
 4                                          /
 5                                          /PROGRAMMET DIVIDERAR ETT 16-BITS TAL I R4-R7
 6                                          /MED ETT 8-BITS TAL I R0 OCH R1.RESULTATET LAGGS
 7                                          /I R6 OCH R7.TALET I R4-R7 ANTAS VARA POSITIVT
 8                                          /EFTERSOM DET AR MENINGEN ATT DET SKALL UTNYTTJA
 9                                          /RESULTATET FRAN PROGRAMMET MULT OCH ANTALET MINUS-
10                                          /TECKEN LIGGER DA I R14.ABSOLUTBELOPPET TAS AV R0
11                                          /OCH R1 OCH VAR TALET NEGATIVT OKAS R14 MED 1.
12                                          /PROGRAMMET AVSLUTAS MED ATT KONVERTERA TALET TILL
13                                          /2-KOMPLEMENTFORM.
14                                          /
15                                          /
16      00000 R                 DIVI        FIM     R14,0   /OBS.OM MULT4 HAR KORTS INNAN FAR
        00000 R 000056 A *G
        00001 R 000000 A *G
17                                                          /INTE R14 NOLLSTALLAS UTAN
18                                                          /INSTRUKTIONEN STRYKS.
19                                          SRC     R15
        00002 R 000057 A *G
20                                          JMS     ABSOL
        00003 R 000120 A *G
        00004 R 000132 E *G
21                                          JMS     SUBT
        00005 R 000120 A *G
        00006 R 000121 R *G
22                                          JTC     MAX     /HOPPA OM DIVIDENDEN AR > ELLER =
        00007 R 000022 A *G
        00010 R 000042 R *G
23                                                          /DIVISORN.
24                                          LDM     -11     /OKTALT
        00011 R 000327 A *G
25                                          .EJECT
```

```
26                                          XCH     R8      /R8 AR RAKNAREN
        00012 R 000270 A *G
27      00013 R                 ROTADD      INC     R8
        00013 R 000150 A *G
28                                          LD      R8
        00014 R 000250 A *G
29                                          JTZ     ROTR
        00015 R 000024 A *G
        00016 R 000050 R *G
30                                          JMS     ROT
        00017 R 000120 A *G
        00020 R 000133 E *G
31                                          JMS     AD      /OM RESULTATET AV ADDITIONEN AR
        00021 R 000120 A *G
        00022 R 000111 R *G
32                                                  /POSITIVT ELLER 0 AR CARRYN=1.
33                                          JFC     ROTADD
        00023 R 000032 A *G
        00024 R 000013 R *G
34      00025 R                 ROTSUB      INC     R7      /I R7 LAGGS RESULTETET IN.
        00025 R 000147 A *G
35                                                  /R6 OCH R7 SKIFTAS EFTER VARJE GANG.
36                                          INC     R8
        00026 R 000150 A *G
37                                          LD      R8
        00027 R 000250 A *G
38                                          JTZ     ROTR
        00030 R 000024 A *G
        00031 R 000050 R *G
39                                          JMS     ROT
        00032 R 000120 A *G
        00033 R 000133 E *G
40                                          JMS     SUBT    /OM RESULTATET AV SUBTRAKTIONEN AR
        00034 R 000120 A *G
        00035 R 000121 R *G
41                                                  /POSITIVT ELLER 0 SA AR CARRYN=1.
42                                          .EJECT
```

```
                DIV     SRC     DIV

43                                              JTC     ROTSUB
        00036 R 000022 A *G
        00037 R 000025 R *G
44                                              JUN     ROTADD
        00040 R 000100 A *G
        00041 R 000015 R *G
45      00042 R                 MAX     LDM     7       /MAX STALLER UT MAXIMALT TAL,
        00042 R 000327 A *G
46                                                      /DVS TALET NARMAST UNDER 1.
47                                              XCH     R6
        00043 R 000266 A *G
48                                              LDM     17
        00044 R 000337 A *G
49                                              XCH     R7
        00045 R 000267 A *G
50                                              JUN     TWOSC
        00046 R 000100 A *G
        00047 R 000072 R *G
51      00050 R 000361 A        ROTR    CLC             /R6 OCH R7 HAR ROTERATS ETT STEG FOR
52                                                      /MYCKET AT VANSTER.DARFOR ROTERAS
53                                                      /DE NU ETT STEG AT HOGER OCH OM UTSKIFTAD
54                                                      /SIFFRA AR 1 HOJS SISTA BITEN.
55                                              LD      R6
        00051 R 000246 A *G
56      00052 R 000366 A                        RAR
57                                              XCH     R6
        00053 R 000266 A *G
58                                              LD      R7
        00054 R 000247 A *G
59      00055 R 000366 A                        RAR
60                                              XCH     R7
        00056 R 000267 A *G
61                                              JFC     TWOSC   /OM DEN UTSKIFTADE BITEN I R7 VAR
        00057 R 000032 A *G
        00060 R 000072 R *G
62                                              .EJECT
```

```
63
64
65                                                      /1 SA AR RESTEN > ELLER = 1/2 OCH
                                                        /DARFOR OKAS TALET MED 1.
                                              INC       R7
          00061 R 000147 A *G
66                                            LD        R7
          00062 R 000247 A *G
67                                            JF2       TWOSC
          00063 R 000034 A *G
          00064 R 000072 R *G
68                                            INC       R6
          00065 R 000146 A *G
69                                            LD        R6
          00066 R 000246 A *G
70        00067 R 000365 A                    RAL
71                                            JTC       MAX       /HOPPA TILL MAX OM TALET BLEV FOR STORT
          00070 R 000022 A *G
          00071 R 000042 R *G
72
                                                        /DVS > ELLER = 1.
73        00072 R                     TWOSC   LD        R14       /TWOSC KONVERTERAR RESULTATET I R6
          00072 R 000256 A *G
74                                                      /OCH R7 TILL 2-KOMPLEMENTFORM.
75        00073 R 000366 A                    RAR
76                                            JFC       OUT       /HOPPA OM TALET AR JAMNT
          00074 R 000032 A *G
          00075 R 000110 R *G
77                                            LD        R6        /KOMPLEMENTERA R6 OCH R7
          00076 R 000246 A *G
78        00077 R 000364 A                    CMA
79                                            XCH       R6
          00100 R 000266 A *G
80                                            LD        R7
          00101 R 000247 A *G
81        00102 R 000364 A                    CMA
82        00103 R 000362 A                    IAC
83                                            .EJECT
```

```
                DIV     SRC     DIV
 84                                         XCH     R7
         00104 R 000267 A *G
 85                                         JFC     OUT
         00105 R 000032 A *G
         00106 R 000110 R *G
 86                                         INC     R6
         00107 R 000146 A *G
 87      00110 R                    OUT     BBL     0
         00110 R 000300 A *G
 88      00111 R 000361 A           AD      CLC             /SUBRUTINEN AD ADDERAR TALET I R0
 89                                                         /OCH R1 TILL TALET I R4 OCH R5.
 90                                         LD      R5
         00112 R 000245 A *G
 91                                         ADD     R1
         00113 R 000201 A *G
 92                                         XCH     R5
         00114 R 000265 A *G
 93                                         LD      R4
         00115 R 000244 A *G
 94                                         ADD     R0
         00116 R 000200 A *G
 95                                         XCH     R4
         00117 R 000264 A *G
 96                                         BBL     0
         00120 R 000300 A *G
 97      00121 R 000361 A           SUBT    CLC             /SUBRUTINEN SUBT SUBTRAHERAR TALET I
 98                                                         /R0 OCH R1 FRAN TALET I R4 OCH R5.
 99                                         LD      R5
         00122 R 000245 A *G
100                                         SUB     R1
         00123 R 000221 A *G
101                                         XCH     R5
         00124 R 000265 A *G
102     00125 R 000363 A                    CMC
103                                         .EJECT
```

```
     DIV     SRC      DIV

104                                  LD       R4
          00126 R 000244 A *G
105                                  SUB      R0
          00127 R 000220 A *G
106                                  XCH      R4
          00130 R 000264 A *G
107                                  BBL      0
          00131 R 000300 A *G
108                                  .END
                  000000 A
          00132 R 000132 E *E
          00133 R 000133 E *E
              SIZE=00134     NO ERROR LINES
```

```
 1                                          .TITLE  ABSOL
 2                                          .GLOBL  ABSOL
 3                                          /
 4                                          /
 5                                          /PROGRAMMET TAR ABSOLUTBELOPPET AV TALEN I
 6                                          /R0 OCH R1.R14 OKAS MED ANTALET NEGATIVA
 7                                          /TAL I R0 OCH R1 (0,1 ELLER 2).
 8                                          /
 9                                          /
10      00000 R              ABSOL  LDM     10
        00000 R 000330 A *G
11      00001 R 000361 A            CLC
12      00002 R 000200 A            ADD     R0
13                                  JFC     POS
        00003 R 000032 A *G
        00004 R 000021 R *G
14      00005 R 000156 A            INC     R14
15      00006 R 000240 A            LD      R0
16      00007 R 000364 A            CMA
17      00010 R 000260 A            XCH     R0
18      00011 R 000241 A            LD      R1
19      00012 R 000364 A            CMA
20      00013 R 000361 A            CLC
21      00014 R 000362 A            IAC
22      00015 R 000261 A            XCH     R1
23                                  JFC     POS
        00016 R 000032 A *G
        00017 R 000021 R *G
24      00020 R 000140 A            INC     R0
25      00021 R              POS    BBL     0
        00021 R 000300 A *G
26              000000 A            .END
    SIZE=00022     NO ERROR LINES
```

```
                                        .TITLE  ROT
                                        .GLOBL  ROT
                                        /
                                        /
                                        /PROGRAMMET ROTERAR R4-R7 ETT STEG TILL VANSTER.
                                        /I SISTA BITEN SKJUTS EN NOLLA IN.
                                        /
                                        /
00000 R 000361 A    ROT     CLC
00001 R 000247 A            LD      R7
00002 R 000365 A            RAL
00003 R 000267 A            XCH     R7
00004 R 000246 A            LD      R6
00005 R 000365 A            RAL
00006 R 000266 A            XCH     R6
00007 R 000245 A            LD      R5
00010 R 000365 A            RAL
00011 R 000265 A            XCH     R5
00012 R 000244 A            LD      R4
00013 R 000365 A            RAL
00014 R 000264 A            XCH     R4
                            BBL     0
00015 R 000300 A *G
      000000 A              .END
    SIZE=00016      NO ERROR LINES
```

```
 1                                          .TITLE  INUT1
 2                                          .GLOBL  MULT,DISP
 3                                          /
 4                                          /
 5                                          /PROGRAMMET GER BEGYNNELSEVARDEN TILL MULT
 6                                          /OCH VISAR RESULTATET PA DISPLAY.
 7                                          LDM     0
        00000 R 000320 A *G
 8      00001 R 000375 A                    DCL
 9                                          FIM     R8,0
        00002 R 000050 A *G
        00003 R 000000 A *G
10                                          SRC     R8
        00004 R 000051 A *G
11                                          FIM     R0,0    /I R0 OCH R2 LAGGS TALEN SOM
        00005 R 000040 A *G
        00006 R 000000 A *G
12                                                          /SKALL MULTIPLICERAS.
13                                          FIM     R2,0
        00007 R 000042 A *G
        00010 R 000000 A *G
14                                          JMS     MULT
        00011 R 000120 A *G
        00012 R 000031 E *G
15      00013 R 000244 A                    LD      R4
16                                          JMS     DISP
        00014 R 000120 A *G
        00015 R 000030 E *G
17      00016 R 000245 A                    LD      R5
18                                          JMS     DISP
        00017 R 000120 A *G
        00020 R 000030 E *G
19      00021 R 000246 A                    LD      R6
20                                          JMS     DISP
        00022 R 000120 A *G
        00023 R 000030 E *G
21      00024 R 000247 A                    LD      R7
22                                          JMS     DISP
        00025 R 000120 A *G
        00026 R 000030 E *G
23      00027 R 000001 A                    HLT
24              000000 A                     END
```

```
 1                                      .TITLE  INUT2
 2                                      .GLOBL  DIVI,DISP
 3                                      /
 4                                      /
 5                                      /PROGRAMMET GER BEGYNNELSEVARDEN TILL DIVI
 6                                      /OCH VISAR RESULTATET PA DISPLAY.
 7                                      /
 8                                      /
 9                                      LDM     0
        00000 R 000320 A *G
10      00001 R 000375 A                DCL
11                                      FIM     R8,0
        00002 R 000050 A *G
        00003 R 000000 A *G
12                                      SRC     R8
        00004 R 000051 A *G
13                                      FIM     R0,0    /I R0 OCH R1 LAGGS DIVISORN.
        00005 R 000040 A *G
        00006 R 000000 A *G
14                                      FIM     R4,0    /I R4-R7 LAGGS DIVIDENDEN.
        00007 R 000044 A *G
        00010 R 000000 A *G
15                                      FIM     R6,0
        00011 R 000046 A *G
        00012 R 000000 A *G
16                                      JMS     DIVI
        00013 R 000120 A *G
        00014 R 000025 E *G
17      00015 R 000246 A                LD      R6
18                                      JMS     DISP
        00016 R 000120 A *G
        00017 R 000024 E *G
19      00020 R 000247 A                LD      R7
20                                      JMS     DISP
        00021 R 000120 A *G
        00022 R 000024 E *G
21      00023 R 000001 A                HLT
22            000000 A                  .END
        00024 R 000024 E *E
        00025 R 000025 E *E
            SIZE=00026      NO ERROR LINES
```

```
                DISP    SRC     DISP

  1                                     .TITLE  DISP
  2                                     .GLOBL  DISP
  3                                     /
  4                                     /
  5                                     /PROGRAMMET VISAR INNEHALLET I ACCUMULATORN  PA
  6                                     /DISPLAY C:A 2 SEK.DAREFTER VISAS 1111 OCH SEDAN
  7                                     /0000 UNDER LIKA LANG TID.VID ATERHOPPET AR
  8                                     /ACCUMULATORN NOLL.
  9       00000 R 000341 A    DISP      WMP
 10       00001 R 000013 A              SB1
 11                                     FIM     R2,0
          00002 R 000042 A *G
          00003 R 000000 A *G
 12                                     FIM     R4,0
          00004 R 000044 A *G
          00005 R 000000 A *G
 13                                     FIM     R6,357  /R6=14=-2 OCH R7=1111
          00006 R 000046 A *G
          00007 R 000357 A *G
 14              000010 R    A=.
 15                                     ISZ     R2,A    /FORDROJNINGSLOOP C:A 2 SEK.
          00010 R 000162 A *G
          00011 R 000010 R *G
 16                                     ISZ     R3,A
          00012 R 000163 A *G
          00013 R 000010 R *G
 17                                     ISZ     R4,A
          00014 R 000164 A *G
          00015 R 000010 R *G
 18                                     ISZ     R5,A
          00016 R 000165 A *G
          00017 R 000010 R *G
 19                                     ISZ     R6,A
          00020 R 000166 A *G
          00021 R 000010 R *G
 20                                     .EJECT
```

```
                DISP    SRC     DISP
 21                                         FIM     R6,377  /R6=-1 OCH R7=1111
          00022 R 000046 A *G
          00023 R 000377 A *G
 22                                         LD      R7
          00024 R 000247 A *G
 23       00025 R 000341 A                  WMP
 24             000026 R    B=.
 25                                         ISZ     R2,B    /FORDROJNINGSLOOP C:A 2 SEK.
          00026 R 000162 A *G
          00027 R 000026 R *G
 26                                         ISZ     R3,B
          00030 R 000163 A *G
          00031 R 000026 R *G
 27                                         ISZ     R4,B
          00032 R 000164 A *G
          00033 R 000026 R *G
 28                                         ISZ     R5,B
          00034 R 000165 A *G
          00035 R 000026 R *G
 29                                         ISZ     R6,B
          00036 R 000166 A *G
          00037 R 000026 R *G
 30                                         FIM     R6,360  /R6=-1 OCH R7=0000
          00040 R 000046 A *G
          00041 R 000360 A *G
 31                                         LD      R7
          00042 R 000247 A *G
 32       00043 R 000341 A                  WMP
 33             000044 R    C=.
 34                                         ISZ     R2,C    /FORDROJNINGSLOOP C:A 2 SEK.
          00044 R 000162 A *G
          00045 R 000044 R *G
 35                                         ISZ     R3,C
          00046 R 000163 A *G
          00047 R 000044 R *G
 36                                         .EJECT
```

```
     DISP   SRC     DISP
 37                                     ISZ     R4,C
          00050 R 000164 A *G
          00051 R 000044 R *G
 38                                     ISZ     R5,C
          00052 R 000165 A *G
          00053 R 000044 R *G
 39                                     ISZ     R6,C
          00054 R 000166 A *G
          00055 R 000044 R *G
 40       00056 R 000012 A              SBO
 41                                     BBL     0
          00057 R 000300 A *G
 42             000000 A               .END
          SIZE=00060     NO ERROR LINES
```