ASSEMBLER 800

Makroassembler till ABC 800

ABC 800®

Luxor Datorer AB ansvarar ej för att denna bruksanvisning till alla delar överensstämmer med andra programversioner än den till vilken bruksanvisningen levererats.

ASSEMBLER 800

Makroassembler till ABC 800

ASSEMBLER 800 omfattar

- EDIT.ABS — Textediteringsprogram för att skapa källfiler, som sedan assembleras med ASMZ.
- ASMZ.ABS ASMZ.OVL — Makroassembler för Z80, som klarar Zilog *och* Databoard mnemonic.
- OBJUPD.ABS — Program för skapande och uppdatering av OBJEKT-bibliotek.
- ESTAB.ABS — Program för länkning av objektfiler och minnes-allokering till en .ABS-fil.
- TRACE.ABS — Test och felsökningshjälpmedel för "debugging" av .ABS-filer.
- ABSBAS.BAC — Program för konvertering och inlänkning av .ABS-filer i basicprogram.

84:35:3:250
Art.nr 66 22904-14

INNEHÅLL

1. INTRODUKTION

Denna manual beskriver ASMZ, en assembler för ABC800.

Manualen är riktad till dem som skall använda ASMZ för att skriva assemblerprogram till Z80.

Manualen är inte avsedd att vara en lärobok i assemblerprogrammering, så läsaren förutsätts ha grundläggande kunskaper inom detta område.

Manualen beskriver inte heller de olika maskininstruktionerna i Z80 - här hänvisas istället till Zilogs manual.

Slutligen förutsätts läsaren ha viss förtrogenhet med ABC 800 och dess filhantering.

I följande kapitel beskrivs bl a:

- Hur ASMZ körs på ABC 800.
- Formatet på assemblerprogram.
- Tillgängliga assemblerdirektiv.
- Villkorlig assemblering.
- Hjälpprogrammet OBJUPD.
- Hjälpprogrammet ESTAB.

I ASMZ finns inbyggt två olika uppsättningar pseudoinstruktioner: dels en kompatibel med Databoard-assembler för Z80, och dels en kompatibel med Zilogs Z80-assembler. Vilken av dessa man vill använda väljer man, med hjälp av satsen PROG eller ZPROG.

Denna manual är i första hand avsedd att beskriva användning av ASMZ i Zilog-kompatibel version.

2. ASMZ PÅ ABC 800

För att köra ASMZ krävs en ABC800 med flexskiveenhet. Det är även önskvärt, men ingalunda nödvändigt, att man har tillgång till en skrivare för utmatning av assemblerlistningen.

ASMZ anropas från ABC800-DOS. Befinner man sig i Basic:en kommer man till DOS:et genom att ge kommandot

BYE

DOS:et svarar genom att skriva ut ett meddelande liknande detta:

```
ABC800 DISC OPERATING SYSTEM

VERS 1.01 Feb '81
* R E A D Y *
```

Här anropas ASMZ med följande kommando:

ASMZ, <optioner>,<källfil>,<objfil>,<copyfil>

Följande optioner är tillåtna:

D Lista assemblerlistan på skärmen.

L Lista assemblerlistan på printer (PR:). Man kan inte initiera printern i detta läge, utan det måste vara gjort tidigare.

O Generera objektkod.

T Skriv ut datum på assemblerlistan. Denna option följs alltid av en datumangivelse på formen:
"=åå-mm-dd,tt:mm:ss" (":ss" kan utelämnas).
Denna option kan bara ges som sista option.

Det får inte finnas blanktecken i optionslistan. Dessutom måste kommatecknet som inleder optionslistan placeras omedelbart efter ASMZ. Listning kan inte fås på både skärm och printer samtidigt; om optionerna D och L ges samtidigt kommer listningen på skärmen.

Om optionslistan, inklusive komma, utelämnas helt, antas optionen D.

Filangivelser (källfil, objfil, copyfil) måste vara skiljda från optionslistan med minst en blank. Inte heller i fillistan får det finnas några blanka inskjutna.

Den enda fil som måste anges är källfilen - den fil som innehåller det källprogram som skall assembleras. Anges ingen filtyp antas den vara .ASM.

Objektfilen - den fil där assemblern lagrar producerad kod - behöver inte anges, och i sådant fall får den samma namn som källfilen, men med filtypen .OBJ. Anges namn och filtyp får filen naturligtvis angivet namn.

Copyfilen behöver inte heller anges. Copyfilen innehåller assemblertext som skall placeras in i källtexten. Se beskrivningen av assemblerdirektivet INCLUDE.

Exempel på kommandorader:

```
ASMZ,DO APROG
ASMZ,DO PROG,PROG,ALIB
ASMZ,DOT=81-09-30,23:37 BLINK
```

3. ASSEMBLERPROGRAMMETS UTFORMNING

Assemblerprogrammet lagras på en textfil. För att skapa denna textfil används en lämplig texteditor. Med assemblern följer en EDITOR med separat manual.

3.1 Programmets uppdelning på rader

Ett assemblerprogram består av rader. Vi kan skilja på två typer av rader: kommentarrader och instruktionsrader.

En kommentarrad kännetecknas av att det står ett semikolon (;) i första kolumnen. Alla övriga rader betraktas som instruktionsrader.

Allting som står på en instruktionsrad hör ihop. Innehållet på en rad kallas för en instruktion. En assemblerinstruktion kan alltså inte börja på en rad och sluta på en annan.

En instruktionsrad uppdelas i ett antal fält. Följande kan ingå:

- Adressfält (även kallat 'label-fält').
- Instruktion.
- Operander.
- Kommentar.

Dessa fält måste vara åtskiljda med minst ett blanktecken.

Operandfältet får inte innehålla mer än 100 tecken. Om raden är längre ignoreras resten.

Instruktioner kan delas in i två grupper: maskininstruktioner och pseudoinstruktioner (assemblerdirektiv). En maskininstruktion översätts av assemblern till maskinkod. En pseudoinstruktion är en instruktion till assemblern.

Assemblerprogram får skrivas i fritt format, dvs det är inte nödvändigt att följa vissa kolumner för en viss del i instruktionen; dock med ett undantag - symboliska adresser måste börja i första kolumnen på en rad.

För att göra det lättare för programmeraren att läsa sitt program kan det dock vara lämpligt att följa särskilda kolumner för de olika instruktionsfälten. Ett förslag till sådan formatering är:

- Adressfältet börjar i kolumm 1
- Instruktion börjar i kolumm 11
- Operanden börjar i kolumm 17
- Kommentar börjar i kloumm 34

Om man väljer den Databoard-kompatibla varianten, kommer ASMZ att formatera assemblerlistningen så att de olika fälten positioneras enligt denna regel, oavsett hur programmeraren har skrivit koden från början.

Adressfältet

Om adressfältet anges skall det innehålla en symbol.

Om raden är en maskininstruktion så kommer symbolen att tilldelas värdet av adressräknaren vid instruktionens början, dvs adressen till den skapade instruktionen.

Olika pseudoinstruktioner tolkar adressfältet olika. I många situationer tolkas det på samma sätt som för maskininstruktioner. Vissa pseudoinstruktioner har en egen tolkning av fältet. Se respektive instruktion för mer information.

Operationsfältet

Operationsfältet består av en symbol som motsvarar en maskininstruktion eller en pseudoinstruktion. Detta är det enda fält som aldrig kan utelämnas.

Operandfältet

Operanderna identifierar de data som skall manipuleras av instruktionen. Typen av operand och antalet operander bestäms av instruktionen som angivits i operationsfältet. En operand består oftast av ett uttryck, med detta är beroende av instruktionen. Om flera operander anges, separeras de med komma.

Kommentarfältet

Kommentarfältet innehåller godtycklig text och finns med på assemblerlistningen.

3.2 Symboler

En symbol är ett namn på en adress eller ett värde. Den kan användas i adressfältet, i en instruktion eller i ett uttryck.

En symbol består av 1 - 8 tecken. Det första tecknet måste vara en bokstav - de följande tecknen kan vara bokstäver, siffror, punkt (.), valutatecken (¤) eller understrykning (_).

3.3 Konstanter

Flera typer av konstanter är tillgängliga.

Numeriska konstanter

Decimala konstanter består av ett decimalt tal i området 0 till 65535. För att särskilt ange att det är en decimal konstant får bokstaven D skrivas omedelbart efter konstanten, men kan också utelämnas.

Hexadecimala konstanter måste börja med en siffra och sluta med ett H. Om man vill ha en hexadecimal konstant som börjar med en bokstav sätter man en nolla (0) framför.

Oktala konstanter avslutas med ett Q.

Binära konstanter avslutas med ett B.

Följande konstanter representerar samma värde:

162	162D	242Q	0A2H	10100010B

Teckenkonstanter

Teckenkonstanter får bestå av ett eller två tecken, och måste inledas och avslutas med situationstecken ("). Om *ett* tecken anges får konstanten värdet av ISO-koden för tecknet. Om *två* tecken anges blir värdet 256 gånger koden för första tecknet plus koden för andra tecknet. Om man vill ge ett citationstecken i konstanten så skrivs tecknet dubbelt. (Exempelvis motsvaras "%" " " av de två tecknen % och ".)

3.4 Uttryck

Ett uttryck är en symbol, en konstant eller en följd av symboler och konstanter åtskilda av operatorer. Uttryck beräknas från höger till vänster. Parenteser får inte finnas i uttryck.

De tillåtna operatorerna är:

+ addition
- subtraktion
* multiplikation
/ heltalsdivision

! bitvis OR (inklusivt eller)
& bitvis AND (och)

Uttryck beräknas alltid med 16 bitar.

Relokerbara och absoluta uttryck

Ett uttryck är absolut om det har ett värde som inte beror på var någonstans i minnet som programmet placeras. Ett relokerbart värde beror på var programmet placeras i minnet och ändrar värde med skillnaden mellan adressen till gamla placeringen och nya placeringen när det flyttas.

De logiska operatorerna (! och &) kan bara utföras på absoluta värden och ger bara absoluta värden som resultat. De övriga operatorerna (+, -, * och /) kan användas i uttryck med absoluta eller relokerbara värden så länge som resultatet är absolut eller relokerbart en gång.

Importerade symboler (se pseudoinstruktionen EXTERNAL) får ingå i uttryck med operatorerna + och - och tolkas som absoluta värden.

Som en speciellt uttryck är det möjligt att få de övre åtta bitarna av ett relokerbart värde, genom att skriva REL/256, där REL är en relokerbar symbol.

Adressräknaren

Värdet av adressräknaren vid början av en viss instruktion kan fås genom att använda en asterisk (*).

3.5 Beskrivning av assemblerlistan

De olika fälten på listan betyder:

LOCATION	Värdet av adressräknaren.
PLC	Använd adressräknare.
CODE	Genererad objektkod.
ARG	Värdet av en symbol i ORG, DEFL eller EQU-sats.
LC	Radnummer.
SOURCE STATEMENT	Källtext för raden.

Efter assemblerlistan kommer en korsreferenslista med följande beteckningar:

SYMBOL	Namnet på symbolen. Prefixat med: (inget prefix) Symbolen använd. ? Symbolen ej använd. < Importerad symbol. > Exporterad symbol.
LOCATION	Värdet av symbolen (Senaste om DEFL använts).
DECL	Rad där symbolen deklarerats.
REFERENCES	Rader där symbolen refererats.

Fel i assemblerprogrammet

Om det finns ett fel i en instruktion anges detta med ett felmeddelande på nästa rad, och med en angivelse om var någonstans på raden som felet hittades.

4. PSEUDOINSTRUKTIONER

Pseudoinstruktioner används för att kontrollera assembleringen, för att definiera symboler och för att generera data. Pseudoinstruktionerna genererar inte alltid data på samma sätt som maskininstruktionerna gör.

4.1 Instruktioner för att definiera symboler

EQU - Tilldela en symbol ett värde

Adressfält	Operation	Operand
Symbol	EQU	Uttryck

Instruktionen tilldelar symbolen resultatet av uttrycket. Värdet av symbolen är absolut eller relokerbart beroende på uttryckets värde. De symboler som används i uttrycket skall vara definierade tidigare i programmet.

Exempel:

```
LOOP    EQU     LOOP1
TOP     EQU     END-64
HERE    EQU     *
START   EQU     0F00H
NO      EQU     2
POINT   EQU     BASE/256+1*256+OFFSET
```

DEFL - Tilldela en symbol ett värde

Adressfält	Operation	Operand
Symbol	DEFL	Uttryck

Instruktionen tilldelar en symbol ett värde. Den kan också användas för att ge en symbol ett nytt värde, men bara om symbolen första gången definierades i en DEFL-sats. Symboler definierade med DEFL kallas ibland assemblervariabler.

Exempel:

```
IGEN    DEFL    0
IGEN    DEFL    10H
IGEN    DEFL    IGEN+3
```

GLOBAL - Deklarera exporterade symboler

Adressfält	Operation	Operand
	GLOBAL	En eller flera symboler, separerade med komma.

GLOBAL-instruktionerna används för att deklarera de symboler i programmet som skall kunna användas av andra, externa program. Detta gör det möjligt för separatassemblerade program att kommunicera med varandra. Bara de symboler som angivits i GLOBAL-instruktionen är åtkomliga från andra, separatassemblerade program. Om en symbol deklareras med en asterisk (*) - exempelvis 'A EQU*' - så anses denna också vara en exporterad symbol.

Exempel:

```
          GLOBAL  SIN,COSIN
          .
          .
          .
SIN       LD      HL,TEMP1
          .
          .
          .
COSIN LD      A,0
ISIN  LD      IX,TEMP2
```

EXTERNAL - Deklarera importerade symboler

Adressfält	Operation	Operand
	EXTERNA	En eller flera symboler, separerade med komma

EXTERNAL-instruktionen anger symboler i andra, separatassemblerade program som detta program refererar till. Detta gör det möjligt för separatassemblerade program att kommunicera med varandra. EXTERNAL-instruktionerna måste komma före de instruktioner som refererar till de importerade symbolerna. Alla odefinierade symboler antas vara importerade och en varning ges på assemblerlistan.

Exempel:

```
EXTERNAL SIN, BITMASK ;deklarera SIN och BITMASK
.                     ;som importerade symboler
.
.
CALL SIN
AND BITMASK
```

Endast addition och subtraktion får användas på importerade symboler i uttryck.

GLOBAL- och EXTERNAL-deklarerade symboler möjliggör uppdelning av program i mindre moduler. Dessa moduler kan assembleras var för sig och refererar till varandra med hjälp av externa symboler. Vid ändring av en modul behöver endast denna assembleras om och sedan länkas ihop med övriga moduler.

4.2 Generering av data

DEFW - Definiera ord

Adressfält	Operation	Operand
Symbol	DEFW	En eller flera uttryck separerade med komma.

DEFW-instruktionen används för att generera 16-bitars ord. Den kan generera ett eller flera ord. De 16 bitar som genereras lagras med de lägre 8 bitarna först och de högre 8 bitarna sedan, dvs som en maskinadress i Z80. Uttrycken får innehålla relokerbara och externa symboler. Symbolen i adressfältet behöver inte anges.

Exempel:

```
TAB    DEFW 0F10H,0F11H
DATA   DEFW TAB
       DEFW 0,1,2,3,4,5
```

DEFB - Definiera byte

Adressfält	Operation	Operand
Symbol	DEFB	Ett eller flera uttryck, separerade med komma.

DEFB-instruktionen används för att generera 8-bitars data. Varje uttryck representerar ett absolut eller ett relokerbart värde. För varje uttryck genereras en byte data och adressräknaren ökas med ett. Symbolen i adressfältet behövar inte anges.

Exempel:

```
A       DEFB  *
        DEFB  10H,20Q
        DEFB  "1"
        DEFB  REL+EXT-10H
```

DEFM - Definiera n bytes

Adressfält	Operation	Operand
Symbol	DEFM	'S'

Definierar minnesinnehållet i ett antal bytes till ASCII-representationen av strängen 'S', där antalet bytes motsvarar längden av 'S'. Längden måste ligga i intervallet 0-63.

DEFS - Definiera minnesarea

Adressfält	Operation	Operand
Symbol	DEFS	Antal

DEFS-instruktionen reserverar ett antal bytes i minnet genom att räkna upp adressräknaren med samma belopp son anges i operandfältet till instruktionen. Inga data genereras och det reserverade minnesområdet sätts inte till något begynnelsevärde.

Exempel:

```
        DEFS    8       reserverar 8 byte
```

4.3 Instruktioner för kontroll av listning

ZPROG - Anger programnamn och listhuvud

ZPROG ska stå först i programmet (utan några rader före) och anger att ZILOG-mnemonics ska användas. Jfr PROG för Databoard assembler.

Adressfält	Operation	Operand
Symbol	ZPROG	Textsträng

Denna instruktion definierar den angivna symbolen till programmets namn.

Programnamnet skrivs tillsammans med textsträngen överst på varje sida av assemblerlistan. Textsträngen får innehålla 56 tecken och kan som exempel innehålla:

- Projektnummer.
- Datum på programmet och dess revisionsnummer.
- Längre version på programmet.

Endast en ZPROG-sats får finnas i ett program.

*HEADING - Ange undertitel på programavsnitt

Adressfält	Operation	Operand
	*HEADING	Textsträng

*HEADING-instruktionen gör det möjligt att ge en eller flera undertitlar. En assemblerlista som produceras av ASMZ innehåller ett listhuvud (bestäms i PROG-satsen) på varje sida följd av en eller flera undertitlar. *HEADING-satsens operand får bestå av maximalt 80 tecken. Alla tecken som följer efter operationskoden placeras i undertiteln på varje sida av listningen.

*EJECT - Utför sidframmatning

Adressfält	Operation	Operand
	*EJECT	

*EJECT-instruktionen medför att ny sida matas fram på assemblerlistan.

*LIST - Kontroll av listningen

Adressfält	Operation	Operand
	*LIST	Parameter

Där parametern är en av följande:

- ON/OFF
- DATA/NODATA
- IF/NOIF

*LIST ON/OFF

*LIST ON/OFF-instruktionen används för att kontrollera assemblerlistningen. En intern listvariabel utnyttjas. När variabeln är mindre än 0 undertrycks utskriften. *LIST ON ökar variabeln med 1 och LIST OFF minskar den med 1.

Startvärdet för den interna listvariabeln är 0. Rader som innehåller fel kommer att listas även om listningen för övrigt är avstängd.

*LIST DATA/NODATA

*LIST DATA: Lista alla data som DEFB,DEFW och DEFM genererar.

*LIST NODATA: (Default) Lista endast de data som får plats på en rad.

*LIST IF/NOIF

Denna instruktion har sin funktion när villkorlig assemblering används. I *LIST NOIF-mode listas inte de oassemblerade instruktionerna; i *LIST IF-mode listas de. Startvärde är *LIST NOIF.

4.4 Instruktion för hantering av adressräknaren

ORG - Tilldela värde till adressräknaren

Adressfält	Operation	Operand
Symbol	*PAGE	Antal

Denna sats bestämmer hur många rader det skall vara på varje sida. Antalet rader får inte vara mindre än 10 och inte större än 99. Startvärdet är 38 rader per sida (= liggande A4).

Instruktionen ORG används för att sätta adressräknaren till värdet som anges av uttrycket. Värdet kan vara relokerbart eller absolut.

Symboler som används i uttrycket måste ha definierats tidigare i programmet.

Om instruktionen innehåller en symbol i adressfältet får symbolen samma värde som adressräknaren, dvs värdet av det angivna uttrycket.

*PAGE - Sätt antal rader per sida

Adressfält	Operation	Operand
Symbol	ORG	Uttryck

4.5 Kontrollinstruktioner

END - avslutar assembleringen

Adressfält	Operation	Operand
Symbol	END	Uttryck

Denna instruktion avbryter assembleringen. Värdet av operanden skall vara ett absolut eller relokerbart uttryck och anger startadressen för programmet. Anges ingen operand får programmet ingen startadress.

I denna instruktion får ingen kommentar anges i kommentarfältet.

*INCLUDE - Kopiera assemblertexten från fil

Adressfält	Operation	Operand
Symbol1	*INCLUDE	Symbol2

Denna instruktion hämtar angivit programavsnitt och placerar det i programmet.

ASMZ söker igenom den fil från vilken kopiering ska ske (se kapitel 2) efter den symboliska adress som anges i operandfältet och kopierar från och med den symboliska adressen till nästa END-sats i filen. Denna END-sats kopieras inte. Därefter fortsätter ASMZ att hämta programtext från den ursprungliga programfilen.

På detta sätt kan alltså bara en programmodul hämtas (från den angivna symboliska adressen till nästa END-sats). Skall flera programmoduler kopieras måste en *INCLUDE-instruktion göras för varje sådan modul. Observera att endast EN fil kan sökas.

4.6 MACRO-definition

Med hjälp av MACRO-definitioner är det möjligt för användaren att definiera egna operationskoder.

Vid varje anrop av ett MACRO läggs, vid assembleringen, automatiskt macrokroppen in i källtexten.

4.6.1 MACRO-uppbyggnad

En MACRO-definition skall bestå av en MACRO-sats, en prototypsats, MACRO-kropp och en ENDM-sats.

SYNTAX:

```
<label>  MACRO
         MACRONAMN <parameter1,parameter2, . . ,>
         -
         -
         macrokropp
         -
         -
         ENDM
```

Uppgifter inom <> kan utelämnas.

Förklaring:

MACRO:	Deklarerar MACRO-definitionens startläge.
MACRONAMN:	Är en symbol.
<parametrar>:	Är symboler eller symboler föregångna av ”?”.
ENDM:	Deklarerar MACRO-definitionens slut.

Exemplet nedan visar en MACRO-definition utskrivet av Assemblern. Lägg märke till de expanderade raderna, som markeras med +.

```
ASMZ-4.41 MACROEXEMPEL                                    ERRORS:   0    11:30:00    82-03-24    PAGE   1


   LOCATION PLC CODE      ARG     LC   SOURCE STATEMENT ----------------------------------------------------

    0000                           1  MACROEX   ZPROG MACROEXEMPEL
    0000                           2            *PAGESIZE 62
    0000                 FE00      3            ORG   65024
                                   4  ;****************************************
                                   5  ;DEFINITIONER
    FE00         0C                6  TEXT1     DEFB  12           TöMMER BILDSKÄRM
    FE01         44455454          7            DEFM  'DETTA ÄR ETT'
    FE0D         0A0D              8            DEFW  0D0AH        .  CR+LF
    FE0F                 000F      9  TEXT1L    EQU   15              ;TEXTLÄNGD 15 TECKEN
    FE0F         535B5454         10  TEXT2     DEFM  'SÄTT ATT SKRIVA ETT'
    FE22         0A0D             11            DEFW  0D0AH
    FE24                 0015     12  TEXT2L    EQU   21
    FE24         4D204120         13  TEXT3     DEFM  'M A C R O'
    FE2D         0A0D             14            DEFW  0D0AH
    FE2F                 000B     15  TEXT3L    EQU   11
    FE2F                 C103     16  CMDINT    EQU   0C103H
    FE2F                 000B     17  DSPLY     EQU   0BH            UTSKRIFTSRUTIN
                                  18  ;****************************************
                                  19  ;MACRODEFINITION
    FE2F                          20            MACRO
    FE2F                          21            TEXT  UT,LÄNGD        ;UT=ADRESS TILL TEXT  LÄNGD=TEXTLÄNGD
    FE2F                          22            LD    HL,UT
    FE2F                          23            LD    BC,LÄNGD
    FE2F                          24            CALL  DSPLY
    FE2F                          25            ENDM
                                  26  ;****************************************
                                  27  ;HUVUDPROGRAM
    FE2F                          28  START     TEXT  TEXT1,TEXT1L      MACROANROP
    FE2F         2100FE           28+           LD    HL,TEXT1
    FE32         010F00           28+           LD    BC,TEXT1L
    FE35         CD0B00           28+           CALL  DSPLY
    FE38                          29            TEXT  TEXT2,TEXT2L      MACROANROP
    FE38         210FFE           29+           LD    HL,TEXT2
    FE3B         011500           29+           LD    BC,TEXT2L
    FE3E         CD0B00           29+           CALL  DSPLY
    FE41                          30            TEXT  TEXT3,TEXT3L      MACROANROP
    FE41         2124FE           30+           LD    HL,TEXT3
    FE44         010B00           30+           LD    BC,TEXT3L
    FE47         CD0B00           30+           CALL  DSPLY
    FE4A         C303C1           31            JP    CMDINT
    FE4D         FE2F             32            END   START

      ERRORS :    0   WARNINGS :    0
```

```
ASMZ-4.41 MACROEXEMPEL                          ERRORS:  0   11:30:00   82-03-24   PAGE  2

   CROSS REFERENCE LISTING

   SYMBOL      LOCATION   DECL   REFERENCES  ----------------------------------------------------------

    CMDINT      C103       16    31
    DSPLY       000B       17    28    29    30
    START       FE2F       28    32
    TEXT1       FE00        6    28
    TEXT1L      000F        9    28
    TEXT2       FE0F       10    29
    TEXT2L      0015       12    29
    TEXT3       FE24       13    30
    TEXT3L      000B       15    30
```

4.6.2 Generering av labels

Om en parameter föregås av ett "?" kommer en label att genereras om parametern har utelämnats vid anropet. Labeln får formen "??"+nr, där nr är ett nummer som uppräknas för varje label som skapas.

4.6.3 Symbolkonvertering och konkatenering

Om ett "%" sätts före en symbol ersätts symbolen med sitt decimala värde vid expansionen.

Karaktären ";" används för konkatenering. Om t ex parametrarna A och B skrivs på formen A;B i Macrokroppen kommer det att ,vid expansionen, att ersättas med AB.

4.6.4 MACRO-expansion

Vid assembleringen läggs MACRO-kroppens rader ut på de platser som MACRO-definitionen har anropat. Dessa expanderade rader erhåller samma radnummer som anropet men märks med ett "+".

Rekursiv expansion är tillåten, där enda begränsningen är tillgängligt minne.

4.6.5 MACRO-anrop

Syntax:

MACRONAMN <parameter,parameter,..,>

Samtliga parametrar som finns i definitionen måste anges om de i definitionen ej har föregåtts av ett "?".

MACRO-definitionen måste göras före första anrop.

5. VILLKORLIG ASSEMBLERING

5.1 COND - Starta villkorlig assemblering

Adressfält	Operation	Operand
Symbol	COND	1 eller 2 absoluta uttryck

Denna pseudoinstruktion gör det möjligt att undertrycka assembleringen av vissa programavsnitt, beroende på operandernas värden.

Operanderna kan vara ett absolut uttryck eller två absoluta uttryck skilda åt av en jämförelseoperator.

5.1.1 Ett argument

Om ett argument anges och dess värde är skilt från noll, assembleras den kod som följer. Om argumentet har ett värde lika med noll så kommer koden från och med "COND" till och med motsvarande "ENDC" att läsas men ej assembleras.

5.1.2 Två argument

Om två argument givits skall dessa åtskiljas med någon av jämförelseoperatorerna:

< mindre än
<= mindre än eller lika med
= lika med
>= större än eller lika med
> större än
<> skiljt från

De två uttrycken beräknas så som beskrivs i avsnitt 3.4 och den angivna jämförelsen utförs. Om villkoret är uppfyllt, dvs jämförelsen är sann, kommer efterföljande instruktioner att tas med i programmet. Är villkoret inte uppfyllt - jämförelsen är falsk - hoppar ASMZ över följande instruktioner fram till dess att motsvarande ENDC-instruktion påträffas.

5.2 ENDC - Slut på villkorlig assemblering

Adressfält	Operation	Operand
Symbol	ENDC	

Denna instruktion avslutar den villkorliga assemblering som påbörjats vid motsvarande COND-instruktion tidigare i programmet.

5.3 Exempel på villkorlig assemblering

```
A       DEFL    10H
B       DEFL    20H
        COND    A<>B
A       DEFL    B
        ENDC
        COND    A
A       DEFL    0
        ENDC
```

Appendix A

Assemblerinstruktioner, sammafattning

*EJECT	- Utför sidframmatning.
*HEADING	- Ange undertitel på programavsnitt.
*INCLUDE	- Kopiera assemblertext från fil.
*LIST	- Kontroll av listningen.
*PAGE	- Sätt antal rader xer sida.
COND	- Starta villkorlig assemblering.
DEFB	- Definiera byte.
DEFL	- Tilldela en symbol ett värde.
DEFM	- Definiera n bytes.
DEFS	- Definiera minnesarea.
DEFW	- Definiera ord.
END	- Avsluta assembleringen.
ENDC	- Slut på villkorlig assemblering.
EQU	- Tilldela en symbol ett värde.
EXTERNAL	- Deklarera importerade symboler.
GLOBAL	- Delkarera exporterade symboler.
ORG	- Tilldela värde till adressräknaren.
ZPROG	- Ange programnamn och listhuvud.

Appendix B

Assemblerdirektiv i Databoard-mod

ALIGN	
COPY	- Kopiera assemblertext från fil. Motsvarar *INCLUDE i ZILOG-mod.
CROSS	- Generera korsreferenslista.
DA	- Definiera adress. Motsvarar DEFW i ZILOG-mod.
DB	- Definiera byte. Motsvarar DEFB i ZILOG-mod.
DMA	- Definiera multipel adress.
DMB	- Definiera multipel byte.
DO	- Upprepad assemblering av en instruktion.
DS	- Definiera minnesarea. Motsvarar DEFS i ZILOG-mod.
EJECT	- Utför sidframmatning. Motsvarar *EJECT i ZILOG-mod.
END	- Avslutar assembleringen. Motsvarar END i ZILOG-mod.
ENDF	- Slut på villkorlig assemblering. Motsvarar END i ZILOG-mod.
ENDS	- Avsluta definition i datastruktur.
ENTRY	- Delkarera exporterade symboler. Motsvarar GLOBAL i ZILOG-mod.
EQU	- Tilldela en symbol ett värde. Motsvarar EQU i ZILOG-mod.
ERROR	- Generera fel.
EXTRN	- Delkarera importerade symboler. Motsvarar EXTERNAL i ZILOG-mod.
IF	- Starta villkorig assemblering. Motsvarar COND i ZILOG-mod.

LCNT	- Sätt antal rader per sida. Motsvarar *PAGE i ZILOG-mod.
LET	- Tilldelar en symbol ett värde. Motsvarar DEFL i ZILOG-mod.
LIST	- Kontroll av listning. Motsvarar *LIST i ZILOG-mod.
NCROS	- Generera ej korsreferenslista.
ORG	- Tilldela värde till adressräknaren. Motsvarar ZPROG i ZILOG-mod.
PLC	- Byte av adressräknare.
PROG	- Anger programnamn och listhuvud. Motsvarar PROG i ZILOG-mod.
RADIX	- Sätt bas för utskrivna tal.
STRUC	- Definiera datastruktur.
TARGT	- Sätt målmaskin (8080 eller Z80).
TITLE	- Anger undertitel på programavsnitt. Motsvarar *HEADING i ZILOG-mod.
TP	- Sätt adressräknaren till närmast högre 256-multipel.
TPL	- Sätt adressräknaren till närmast högre 256-multipel.

Notera att endast de angivna pseudo-instruktionerna har någon motsvarighet i ZILOG-mod.

Appendix C

ASCII-koden

CTRL	SHIFT TECKEN	ASCII-NAMN	DEC	OKT	HEX
X	E'	NUL	0	0	0
X	A	SCH	1	1	1
X	B	STX	2	2	2
X	C	ETX	3	3	3
X	D	EOT	4	4	4
X	E	ENQ	5	5	5
X	F	ACK	6	6	6
X	G	BEL	7	7	7
X	H	BS	8	10	8
X	I	HT	9	11	9
X	J	LF	10	12	A
X	K	VT	11	13	B
X	L	FF	12	14	C
X	M	CR	13	15	D
X	N	SO	14	16	E
X	O	SI	15	17	F
X	P	DLE	16	20	10
X	Q	DC1	17	21	11
X	R	DC2	18	22	12
X	S	DC3	19	23	13
X	T	DC4	20	24	14
X	U	NAK	21	25	15
X	V	SYN	22	26	16
X	W	ETB	23	27	17
X	X	CAN	24	30	18
X	Y	EM	25	31	19
X	Z	SUB	26	32	1A
X	Ä	ESC	27	33	1B
X	Ö	FS	28	34	1C
X	Å	GS	29	35	1D
X	Ü	RS	30	36	1E
X	0	US	31	37	1F
			32	40	20
		!	33	41	21
		”	34	42	22
		#	35	43	23
		¤	36	44	24
		%	37	45	25

CTRL	SHIFT	TECKEN	ASCII-NAMN	DEC	OKT	HEX
		&		38	46	26
		'		39	47	27
		(		40	50	28
		)		41	51	29
		*		42	52	2A
		+		43	53	2B
		,		44	54	2C
		-		45	55	2D
		.		46	56	2E
		/		47	57	2F
		0		48	60	30
		1		49	61	31
		2		50	62	32
		3		51	63	33
		4		52	64	34
		5		53	65	35
		6		54	66	36
		7		55	67	37
		8		56	70	38
		9		57	71	39
		:		58	72	3A
		;		59	73	3B
		<		60	74	3C
		=		61	75	3D
		>		62	76	3E
		?		63	77	3F
		É		64	100	40
		A		65	101	41
		B		66	102	42
		C		67	103	43
		D		68	104	44
		E		69	105	45
		F		70	106	46
		G		71	107	47
		H		72	110	48
		I		73	111	49
		J		74	112	4A
		K		75	113	4B
		L		76	114	4C
		M		77	115	4D
		N		78	116	4E
		O		79	117	4F

CTRL	SHIFT	TECKEN	ASCII-NAMN	DEC	OKT	HEX
		P		80	120	50
		Q		81	121	51
		R		82	122	52
		S		83	123	53
		T		84	124	54
		U		85	125	55
		V		86	126	56
		W		87	127	57
		X		88	130	58
		Y		89	131	59
		Z		90	132	5A
		Ä		91	133	5B
		Ö		92	134	5C
		Å		93	135	5D
		Ü		94	136	5E
		_		95	137	5F
		e'		96	140	60
		a		97	141	61
		b		98	142	62
		c		99	143	63
		d		100	144	64
		e		101	145	65
		f		102	146	66
		g		103	147	67
		h		104	150	68
		i		105	151	69
		j		106	152	6A
		k		107	153	6B
		l		108	154	6C
		m		109	155	6D
		n		110	156	6E
		o		111	157	6F
		p		112	160	70
		q		113	161	71
		r		114	162	72
		s		115	163	73
		t		116	164	74
		u		117	165	75
		v		118	166	76
		w		119	167	77
		x		120	170	78
		y		121	171	79

CTRL	SHIFT	TECKEN	ASCII-NAMN	DEC	OKT	HEX
			z	122	172	7A
			ä	123	173	7B
			ö	124	174	7C
			å	125	175	7D
			ü	126	176	7E
X		<>	■	127	177	7F

Appendix D

OBJUPD - Uppdatera objektmodulsbiblioteket

OBJUPD används för att skapa och underhålla bibliotek med objektmoduler. Dessa bibliotek kan sedan länkas med ESTAB för att generera en exekverbar absolut (ladd-) modul.

OBJUPD startas från ABC800-DOS med kommandot:

```
OBJUPD objlib,objfil,newlib
```

Där objlib är namnet på objektmodulsbiblioteket, objfil är namnet på objektprogrammet som skall adderas eller ändras i biblioteket och newlib (kan utelämnas) är namnet på det genererade biblioteket om det gamla ska bevaras oförändrat.

OBS!
OBJUPD kan inte användas på tomma objektfiler.

Appendix E

ESTAB-Länka objektmoduler

Inledning

Detta program används för att skapa taskfiler (.ABS) från en eller flera filer som skapats med ASMZ för ABC800.

Beskrivning

ESTAB startas från ABC800-DOS med kommandot:

ESTAB<,Optioner><Infil><,Utfil>

Uppgifter inom hakparenteser kan utelämnas. Om samtliga parametrar utelämnas startar ESTAB i kommandomod. Dessa kommandon beskrivs nedan.

Tillgängliga optioner:

D - Listning sker på bildskärmen.
L - Listning sker på skrivaren.
O - Alla värden listas i octal form (default hex).

Om infilen ej påträffas vid start, i ovan angivna operativa mod, startar ESTAB i kommandomod och en promt (>) skrivs på bildskärmen. Då promten framträder är ESTAB klar för att mottaga kommandon.

För att en task-fil (.ABS) skall skapas måste utfil anges.

Då kommandofil skall användas anges detta vid start enligt följande:

ESTAB<,optioner> CMD=kommandofil

Nedan beskrivna kommandon används i kommando-mod eller för att bygga upp en kommandofil.

Kommandon

ABORT	Programexekveringen avbryts. Utfilen raderas.
END <value>	Anges då kommandosekvensen är slut. Utfilen skapas och listning erhålls. Om <value> har angivits utgör den programmets startadress.

CHECK Alla odefinierade symboler listas på bildskärmen med angivande av läget för dess första referens.

EQU value,symbol,symbol, . . .
De uppräknade symbolerna erhåller angivet värde. Value kan vara symboliskt.

Om symbolerna ej tidigare har definierats, läggs de upp i symboltabellen.
Om symbolerna redan har ett värde anges de som definierade på flera ställen (multiple defined).

CEQU value,symbol,symbol, . . .
Samma som EQU men om symbolen redan har ett värde ges ej något felmeddelande. Symbolen får ej heller något nytt värde.

INCLUDE<,opt> filnamn <.modul>
Specificerade filer eller moduler inkjuderas. Om ej extension anges antas .OBJ.
Optioner används vid sökning efter en modul:

R (default) Sök från nuvarande position. Om modulen inte påträffas upprepas sökningen från filens början.

W Filen genomsöks en gång från filstart.

E Sökning efter modul till filslut.

LIBRARY<,opt> filnamn <.modul>
Den specificerade filen undersöks och de refererade modulerna inkluderas.
Om endast en modul specificeras antas senast använda fil som inmatningsfil. Default extention är .OBJ Optionerna är:

R (default) Sök från nuvarande position. Om end-of-file påträffas upprepas sökningen från filstart. Denna process upprepas till inga odefinierade symboler förekommer eller att inga fler referenser kan finnas inom filen.

W Samma som R men sökningen börjar från filstart.

E Objektfilen genomsöks endast en gång.

LIST UNUSED/ABSOLUTES
NOLIST UNUSED/ABSOLUTES

Dessa kommandon används för att välja om listning av oanvända symboler eller symboler med absolut värde skall utföras.

PLCNR plcnr — Kommandot används för att välja en PLC (Program Location Counter) under ett programavsnitt som ORGats inom annan minnesarea.

PLCCOMMON plcnr

Detta kommando används för att välja ett PLC inom vilken valfri Common-area kan vara allokerad. Default PLC=1.

PLCEND — Detta kommando ger upphov till att en PLC-end symbol genereras. Symbolen erhåller namn enligt "¤¤PLCE00-31" och tilldelas värdet av "plc-top". Referens kan ske till symbolen i användarprogrammet, men symbolen definieras ej förrän kommandot "END" utförs.

PLCSTART — Samma som PLCEND men symbolen erhåller namn enligt "¤¤PLCS00-31" och erhåller värdet "plc-bottom".

PLCORDER plcnr,plcnr, . . .

Anger i vilken ordning som plc-segmenten läggs in i minnet.

PLCBASE värde — Detta kommando används tillsammans med PLC-ORDER för att skapa den första PLC i en sekvens. Defaultvärdet är 0.

ORG värde — Detta kommando sätter aktuell plc, som specifiserats med föregående PLC-kommando, till angivet värde.

PRINT <enhet> — Print anger till vilken enhet som listning skall styras. Tillåtna enheter är PR: och CON:.

RADIX 8/16 — Anger om octal (8) eller hexadecimal (16) bas skall användas vid utskrift.

REMOTE	Om detta kommando ges avbryts processen om något fel uppkommer.
TASK filnamn	Kommandot specificerar namnet på filen.
OBJECT filnamn	Kommandot byter namn och sparar den temporära filen på nytt med sitt nya namn. Speciellt användbart i samband med skapande av nya objektfiler från gamla, i samband med användandet av "LIBRARY"- och "INCLUDE"-kommandona.

Teckenhantering

Alla värden kan anges i uttrycksform, som beräknas från vänster till höger som 16-bitars heltal. Operander kan vara numeriska (hex, octal, decimal eller ascii-strängar) eller tidigare definierade symboler. Tillåtna operatorer är + (plus), - (minus), * (multiplikation), / (division), & (AND), ! (OR), ? (XOR) och # (SHIFT).

OBS!
Endast + och - är tillåtna på icke absoluta värden. ORG och PLCBASE kräver absoluta värden. Värdet av det PLC som valts med PLCNR kan erhållas med symbolen '*'. Detta värde är aldrig ett absolutvärde.

Exempel

Exempel på kommandon som ges då ESTAB körs i kommandomod.

```
ESTAB                   Uppstart
>TASK    UTFIL          Filnamn UTFIL.ABS
>ORG     8000H          Startadress
>INC     DEL1
>INC     DEL2           Ingående filer
>INC     ALLEQU
>CHECK                  Letar odefinierade symboler
>END
```

Appendix F

TRACE-Debugger för ABS-filer

1 Inledning

Trace är ett interaktivt program som är gjort för att stödja felsökning i assemblerprogram (.ABS). Programmet har möjligheter att följa ett händelseförlopp i ett program under exekvering, i realtid eller interpretativ mod samt tillåter ändringar under körningen.

TRACE ligger på skiva och körs under DOS (via BYE).

2 Systemförutsättningar

Programmet TRACE laddas från skiva och kräver 14 kbyte minne och laddas från adress 32 kbyte till 46 kbyte. De program som skall debuggas får ej lagras på samma adress som Trace.

3 Start av TRACE

Sätt i programskivan och övergå till DOS-kontroll med hjälp av BYE-kommandot. Ladda och starta TRACE med kommandot:

TRACE<,optioner>< parameter,par,....>

Trace startar nu i kommandomod (interaktiv) och när en promt tänds på skärmen är programmet klart att ta emot kommandon. De optioner och parametrar som anges, används ej av TRACE utan överförs till det program som skall debuggas.

Då tracen har startats skall det program som skall debuggas laddas och startas. Se LOAD eller START/RUN. Trace-programmet kör nu det laddade programmet sats för sats och redovisar mnemonics, flaggor och minnesinnehåll för respektive läge.

4 Moder

Programmet TRACE arbetar i två moder, kommandomod och trace-mod. Kommandomoden används för att ange erforderliga värden för trace-moden, där trace-moden utgör den reella debuggningen.

I kommandomoden skall kommandon för laddning och start av testad fil ges, samt val av trace-villkor, minnesändringar och ändringar av registerinnehåll m m.

Programexekveringen styrs i trace-mod av en-tangentskommandon, som möjliggör specificering av flera trace-villkor.

4.1 Kommandomod

I kommandomod bestäms hur den kommande trace-moden skall arbeta. Det är möjligt att sätta brytpunkter och ange hur de skall hanteras samt att välja till en kopieringsfunktion och att sätta olika symboler m m.

4.1.1 Uttryck

Alla värden kan anges som uttryck vilka beräknas från vänster till höger som 16-bitars heltal. Operander kan vara tidigare definierade symboler eller numeriska värden i hex, oct eller dec- presentation samt ASCII-karaktärer. Tillåtna operatorer är + (plus) och - (minus). Det körda programmets läge kan refereras med '*' och senast inmatade adressvärde kan refereras med ' '. Numeriska värden skrivs i default talbas om ej annat uppgetts med kommando.

4.1.2 Filkommandon

Följande kommandon arbetar mot filen som skall debuggas:

END — Avbryter testen av programmet.

LOad <DRX:>filnamn — Laddar angiven fil från angiven enhet. Om enheten utelämnats söks på samtliga enheter.

STart <filnamn<.typ>>
RUN <filnamn<.typ>> — Laddar och startar angiven fil. Om filnamn ej anges antages filen tidigare laddad.
Default typ = ABS.

4.1.3 Allmänna kommandon

CONvert värde — Konverterar och skriver värdet i dec, oct och hex.

COPy ON — Medför att all text till bildskärm även kopieras på printer.

COPy OFF — Avbryter ovanstående funktion.

RADix 8/10/16 — Väljer talbas. Default = 16.

SET namn=adress — Tilldelar en symbol (namn) ett värde (adress). Namnet är en användardefinierad symbol med max 6 karaktärer. Denna symbol kan användas i alla uttryck. Om talbas 16 används får inte symbolen ge konflikt med något hexadecimalt tal, t ex AF, BAA osv.

SVc OFf/ON — Omöjliggör/Möjliggör (Disable/enable) tolkningen av RST 7-instruktionen som ett SVC. Default är möjlig (Enable).

ZILog — Mnemonics skrivs ut enligt Zilog (default).

ASMz — Mnemonics skrivs ut enligt Databoard-varianten av assemblern.

4.1.4 Minneskommando

Denna kommandogrupp används för att manipulera innehållet i minnet.

EXaminate <,A/D/H/0> Adress <,antal>
Skriver minnesinnehållet i ASCII (A), decimal form (D), hexadecimal form (H) eller oktal form (O). Utskriften börjar på adressen <Adress> och skriver antalet <antal> bytes. Om antal utelämnats skrivs en byte.

FILL<,D/H/O> adress,antal,värde
Fyller en minnesarea <antal> bytes med början på <adress> med värdet <värde>. Talbasen definieras med optionerna D, H eller O alltså Decimal, Hexadecimal eller Oktal.

MOdify adress — 16 bytes skrivs ut från adressen <adress> varefter de nya värdena matas in och separeras med komma (,). Om endast RETURN anges sker ingen ändring.

4.1.5 CPU-Kommandon

Följande kommandon används för att kontrollera CPU's olika faciliteter:

DISable — Stänger av processorns interruptlogik.

ENAble — Möjliggör interrupt till processorn.

INput portnr — Skriver ut datainnehållet i I/Oporten som angavs med portnr.

NMI — Möjliggör för Trace att omhänderta NMI till processorn. Eventuella NMI uppträder som brytpunkter.

OUT portnr,data — Lägger ut angivet data på angiven I/O-port.

REGister <register=värde>

Om parametern är utelämnad skrivs samtliga registers innehåll ut. Då parametern anges ändras värdet i angivet register till det nya värdet.
Registren specificeras enligt följande:

A = Register A.
F = Flaggregistret där flaggorna anges med
  Z = zero true
  C = carry true
  S = sign true
  P = parity true
  För att nollställa samtliga flaggor skall inget värde anges, alltså REG = <return-tangenten.>
BC = Registerpar BC
DE = Registerpar DE.
HL = Registerpar HL.
X = Indexregister X, 16 bitar.
Y = Indexregister Y, 16 bitar.
SP = Stackpekare, 16 bitar.

4.1.6 Brytpunktskommandon

Brytpunkter är, av användaren, definierade lägen där kontrollen av användarens program övertages av TRACE. Brytpunkter läggs in för att kunna köra användarprogrammet i 'free-run' till ett bestämt läge där TRACE-faciliteterna blir tillgängliga. Brytpunkter kan placeras på valfritt instruktionsläge i det testade programmet. Antalet brytpunkter är endast begränsat av minnesutrymmet. När en brytpunkt påträffas kontrolleras att brytpunkten är placerad på en riktig position (instruktionsläge).

Brytpunkter inplaceras i användarprogrammet då TRACE arbetar i interpreterande mod eller i single step-mod. När TRACE övergår i free-run-mod läggs samtliga brytpunkter ut på sina respektive lägen. Då någon brytpunkt påträffas tas den bort och aktuell instruktion läggs ut.

Följande brytpunktskommandon finns tillgängliga:

HAlt adress — Avbryter programexekveringen på angiven adress

JTAble adress — Specificerar basadressen till en 256 bytes stor hopptabell. Alla programinhopp som går genom en sådan tabell kommer ej att behandlas av TRACE.

LIst — Skriver ut den pågående TRACEns villkor och deras adresser.

REMove <adress> — Då adress utelämnas raderas alla villkor. Om adress anges raderas brytpunkten på angiven adress.

SUBroutine adress — Definierar en subrutin med angiven startadress och som ej kommer att behandlas av TRACE. Då subrutinen anropas från användarprogrammet skriver TRACE ut ett meddelande och subrutinen utföres i free-run.
Se varningen under TRACE-kommando 'X'.

TRace <,D/J/M/R/S> adress
Sätter ett tracevillkor på angiven adress Villkoren specificeras enligt följande:

Utelämnade parametrar

Övergår till single-step mod.

D(isable) Övergår till free-run. Programmet kommer att exekveras i free-run tills en ny brytpunkt påträffas.

J(ump) Övergår till interpreterande mod. Skriver endast ut JMP, CALL och RET instruktionerna.

M(emory) Övergår till interpreterande mod. Skriver ut ändringen av alla minnesceller, vid brytpunktsläget, som definierats med kommandot 'WATCH'.

S(can) Övergår i free-run mod. Skriver ut alla ändringar av de minnesceller, vid brytpunktstillfället, som definierats med 'WATCH'.

R(egister) Övergår i free-run mod. Skriver ut alla ändringar, vid brytpunktstillfället, av samtliga CPU-register.

4.1.7 Debugging-kommandon

Dessa kommandon används för att övergå till Tracemod, där programexekveringen kan kontrolleras.

DEBug adress Startar testningen vid angiven adress.

PROceed Fortsätter testningen vid det läge som testningen var vid då övergång senast skedde till kommandomod.

4.2 TRACE-mod

TRACE-moden är uppdelad i två undermoder, interpreterande och free-run mod.

I den intrerpreterande moden sköts exekveringen, av användarprogrammet, av TRACE och inte av användarens egna program. TRACE arbetar som en simulerad CPU.

I free-run mode överförs exekveringen så att användarprogrammet exekveras på vanligt sätt. Om inga brytpunkter har definierats återgår aldrig kontrollen till TRACE.

Att TRACE arbetar i TRACE-mod ses genom att en cursor uppträder efter den rad som skall testas. Den fortsatta exekveringen styrs med TRACE-kommandon. När kontrollen återgår till TRACE skrivs alltid alla registerändringar ut samt nästa instruktion som skall testas. Instruktionerna skrivs ut som mnemonics (Zilog eller Databoard-variant) och indirekta adresser skrivs ut med sina absoluta värden.

4.2.1 TRACE-kommandon

I detta kapitel beskrivs de en-tangentskommandon som används i TRACE-mod:

Mellanslagstangent (SPACE)
Gå till nästa instruktion (single step).

Disable Övergå till free-run mod.

Jump Sätter en brytpunkt i det läge som hoppinstruktionen skall hoppa till.

Kill Tar bort eventuell prytpunkt i nuvarande läge.

Loop Sätter en tillfällig brytpunkt i nuvarande läge + 2. Används t ex för att låta en DJNZ-instruktion gå i free-run till slut. Går att använda vid alla två-byte instruktioner.

Register Skriver ut innehållet i alla CPU-register.

Trace Specificerar en brytpunkt i nuvarande läge.

Watch Specificerar nuvarande läges argument för ändringsövervakning.

Xecute Kan anges då en subrutin har påbörjats och medför då att subrutinen utförs i free-run. Kommandot måste ges före eventuella PUSH och POP.

VARNING!
Observera att stacken inte innehåller användarprogrammets återhoppsadress, utan en adress till TRACE för att kunna återgå till TRACE-mod vid RET. Om subrutinen använder eller modifierar stackens toppadress ska ej X, SUB-rutin eller JTAble användas.

4.3 Felmeddelanden

ERR-TYPE:tttt, POS:ppppp

Då ett felaktigt kommando eller parameter anges skrivs ovanstående feltext ut. Positionsfältet anger var i kommandoraden felet finns, och typfältet anger feltypen:

- COMMAND, då ett okänt kommando använts.
- PARAM, då en otillåten parameter angivits.
- OPTION, då en otillåten option har angivits.
- NAME, då filen saknas eller filnamnet är felaktigt angivet.

END OF MEMORY

Minnet fullt vid definiering av symboler med SET-kommandot.

TAB-OVF

Kan ej specificera flera trace-lägen.

CAN'T FIND IT

Adressen som specificerats med REMOVE finns ej.

ILLEGAL BREAKPOINT

En RST-instruktion förekommer på en ospecificerad adress. Börja om tracen från början.

adress WAS oo IS nn

Minnescellen adress har bytt värde från oo till nn.

SUBROUTINE RUN ...

En subrutin som ej genomgår TRACE genomlöps.

TRACE STOP

En tidigare definierad HALT-adress har påträffats.

HALT TRACED

CPU-instruktionen HLT är påträffad. Exekveras aldrig.

TRACE ABORT

Ett omaskbart interrupt har inträffat.

ILLEGAL INSTRUCTION

En okänd CPU-instruktion har angivits i interpreterande mod.

CRC or FORMAT ERROR

Felaktigt filformat eller checksummafel.

ERROR. Program in TRACE.

Det laddade programmet kolliderar adressmässigt med TRACE.

Appendix G

ABC800 - CMDINT.SYS (DOS)

CMDINT.SYS upptar adresserna C000 till C6FF i ABC800.
CMDINT har några användbara funktioner.

1. CMDINT startadress - 0C100H.

2. CMDINT återstart - 0C103H.

3. Laddning och start av absolut-fil - OC106H.
   (Filnamn plus en vagnretur (ascii 13) ska ligga i 0C0B0H och framåt.)

4. Skriv text - 0C109H.
   HL ska innehålla adressen till texten. Texten avslutas med ett av följande tecken:

   - ascii 3 (EOT).
   - ascii 13 (CR).
   - ascii 19 (DC3).

5. Kommandobuffert - 0C0BOH.
   Här hamnar den kommandosträng man gav till CMDINT. De eventuella argument man har gett, tolkas dessutom som filnamn och läggs efter kontroll i 0FD40H och uppåt i block om 16 bytes. Ett block innehåller:

   - Drive-specifikation (1 byte).
     Möjliga värden:

     0-6 Drive-nummer.
     254 Felaktig Drive-specifikation.
     255 Ospecificierad drive.

   - 0FFH (1 byte).
   - Filnamn (8 byte).
   - Filtyp (3 byte).
   - Enhet (3 byte).

6. KEYIN - 0C000H.
   Läser text från tangentbordet och ekar denna på skärmen. HL innehåller adressen där texten ska lagras, C innehåller max antal tecken som ska läsas.

7. DSPLY - 0C006H.
   Skriver text på bildskärmen. Se beskrivning av 0C109H. För att kunna utnyttja dessa adresser krävs att programmet inte ligger över CMDINT.

Appendix H

BASIC och Assembler-rutiner

Assemblerrutiner anropas från ett BASIC-program med BASIC-funktionen CALL. Innan en assemblerrutin anropas måste den dock laddas, vilket kan ske på flera sätt:

1. POKE-satser i programmet.

2. En sträng innehållande rutinen anropas. Anropet sker på följande sätt (rutinen antas ligga i A¤):

   ```
   X=CALL(VARPTR(A¤),argument).
   ```

3. En absolut-fil laddas in från disken.
   Detta gör man enklast så här:

   ```
   10 A¤=CHR¤(14,255,205,27,96,201) !Lägga en liten
   assemblerrutin i A¤
   20 B¤="FILNAMN EXT" ! 8+3 Tecken
   30 H=CALL(VARPTR(A¤),VARPTR(B¤)) !Laddar
   programmet och returnerar startadressen
   40 H=CALL(H,D) !startar den laddade rutinen,
   D är data
   ```

Appendix I

ABC800 - adresser

Här följer en lista över användbara adresser i ABC800.

- Rutiner i ROM:

  1. OH : Reset ABC800.

  2. 2H : Inchar, läs ett tecken från tangentbordet.

  3. 5H : Inline, läs en rad från tangentbordet.

  4. OBH : Outtext, skriv text på bildskärmen. HL innehåller adressen, BC antalet tecken som ska skrivas.

  5. 10H : Error.Ska anropas med CALL eller RST, Byten efter anropet är felnumret.

  6. 125H : Start av BASIC, dock utan promptning.

- Rutiner i RAM:

  OFF90H : Inchar - läs ett tecken från tangentbordet. Tecknet hamnar i A. Denna adress används även när man gör en CALL 2 (vilket t ex BASIC gör när den läser ett nytt tecken).

- System-variabel, pekar.

  1. OFFOAH : Heap.

  2. OFFOCH : Lägsta minnseadress för BASIC-program.

  3. OFFOEH : Högsta minnesadress för BASIC-program.

  4. OFF2CH : Variabelrot.

  5. OFF7BH : Enhetslista.

6. OFF85H : Kontroll-C flaggbyte. Om denna adress sätts till 0, kommer kontroll-C att helt ignoreras. När kontroll-C ges, sätts den minst signifikanta biten i den cell som utpekas.

- System-variabler, värden.

1. OFF52H : Kolumn för Cursorn.

2. OFF53H : Rad för Cursorn.

3. OFF54H : Antal tecken per rad (40/80).

4. OFFE2H : Tecken finns i bufferten om värdet = 128 (mest signifikant bit satt).

5. OFFE3H : Tangentsbordsbuffert (1 tecken).

- I/O-portar.

1. 5H : Inmatning genererar en puls till högtalaren.

2. 20H : Dart - CH.A (skrivare) DATA.

3. 21H : Kontroll.

4. 22H : Tangentbord (Dart), DATA.

5. 23H : Kontroll

6. 40H : SIO - CH.B (V24) DATA.

7. 41H : Kontroll.

8. 42H : Tape, DATA.

9. 43 H : Kontroll

10. 60H : CTC(timer), Kontroll för kanal 0.

11. 61H : CTC, Kontroll för kanal 1 (V24 Baudrate).

12. 62H : CTC, Kontroll för kanal 2 (Baudrate Skrivare).

13 63H : CTC, Kontroll för kanal 3.

Appendix J

ABSBAS Program för skapande av listfiler

Programmet ABSBAS används då en fil, som har skapats med hjälp av ESTAB - Alltså en ABS-fil -, skall överföras till ett Basicprogram.

Programmet ABSBAS startas med RUN ABSBAS, varvid en meny visas, som ger möjlighet att välja mellan ett Basicprogram i listform (.BAS) bestående av POKE-satser eller DATA-satser.

Alternativet POKE-area väljs då man har valt att lägga programmet på en bestämd adress, t ex i "ledigt för POKE". Alternativet DATA-area väljs då assemblerprogramet skall placeras i en variabel. För att detta skall vara möjligt får assemblerprogrammet endast innehålla relativa hopp.

Exempel på hur DATA-area alternativet kan användas.

```
10 FOR I=1 TO D ! D=Antalet data
20 READ A¤
30 P¤=P¤+A¤!Summera strängen
40 NEXT I
50 Z=CALL (VARPTR(P¤),D) !ev. D-värde läggs i reg DE
60 !assemblerprogrammet startar
100 DATA -----
110 DATA ----
-
-
500 END
```

Då Area-typ har valts frågas efter ABS-fil, alltså assemblerfilen. Anges ingen extention antas .ABS. På frågan LIST-fil, anges önskat namn på Basicfilen. Anges ingen extention antas .BAS. Om endast RETURN svaras vid frågan, erhåller listfilen samma namn som assemblerfilen men med extention .BAS.

Basicfilen erhåller radnummer från 30000 och uppåt.

Den nu skapade listfilen länkas till aktuellt Basicprogram med hjälp av kommandot MERGE filnamn.ext.

```
ASMI-4.41 PROGRAMEXEMPEL                                ERRORS:   0   00:00:00   00-00-00   PAGE   1

   LOCATION PLC CODE      ARG     LC   SOURCE STATEMENT ------------------------------------------------------------

    0000                           1  MAIN     ZPROG                  PROGRAMEXEMPEL
    0000                           2           $PAGE 72
                                   3  ;
                                   4  ;        PROGRAMMET LÄSER ETT TECKEN FRÅN TANGENT-
                                   5  ;        BORDET, KONTROLLERAR OM DET ÄR ETT
                                   6  ;        KONTROLLTECKEN OCH OM DET ÄR DET, MARKERAS
                                   7  ;        DETTA MED ü. EX. KONTROLL A SKRIVS üA.
                                   8  ;
    0000                C700       9           ORG   0C700H
                                  10  ;
    C700                0002      11  INCHR    EQU   02H              ADRESS TILL RUTIN INCHR I ROM
    C700                000B      12  OUTCHR   EQU   0BH              ADRESS TILL RUTIN OUTCHR I ROM
                                  13  ;
                                  14  ;        MAIN PROGRAM
                                  15  ;
    C700        011000            16  BEGIN    LD    BC,TEXTLEN.
    C703        2112C7            17           LD    HL,TEXT
    C706        CD0B00            18           CALL  OUTCHR           SKRIV UT TEXT
                                  19  ;
                                  20  ;        LÄS ETT TECKEN ANROPA SUBRUTINEN SOUTCH
                                  21  ;        SOM TESTAR TECKNET OCH SKRIVER UT DET.
                                  22  ;
    C709        CD0200            23  LOOP     CALL  INCHR            LÄS ETT TECKEN
    C70C        CD22C7            24           CALL  SOUTCH           TESTA,SKRIV
    C70F        18F8              25           JR    LOOP             OÄNDLIG LOOP
                                  26  ;
    C711                          27  BUFFER   DEFS  1                VAR BUFFER
    C712                C712      28  A        DEFL  $
    C712        0C                29  TEXT     DEFB  12               FF TÖMMER SKÄRMEN
    C713        534B5249          30           DEFM  "SKRIV TECKEN:"
    C720        0D0A              31           DEFW  0A0DH            CR+LF
    C722                0010      32  TEXTLEN  EQU   $-A
                                  33  ;
                                  34  ;
                                  35  ;        SUBRUTIN SOM TESTAR VÄRDE I ACK.
                                  36  ;        OM DET ÄR ETT KONTROLLTECKEN SÅ MARKERA
                                  37  ;        MED ETT ü, SKRIV ANNARS UT TECKNET
                                  38  ;        SOM VANLIGT.
                                  39  ;
                                  40  ;
    C722        E67F              41  SOUTCH   AND   7FH              TA BORT BIT 8
    C724        FE20              42           CP    20H              >= SPACE
    C726        D236C7            43           JP    NC,SOUT1
    C729        C640              44           ADD   A,40H            KONTROLLTECKEN, GÖR DET SYNLIGT
    C72B        F5                45           PUSH  AF
    C72C        2141C7            46           LD    HL,MARK
    C72F        010100            47           LD    BC,1
    C732        CD0B00            48           CALL  OUTCHR           SKRIV UT +
    C735        F1                49           POP   AF
    C736        2111C7            50  SOUT1    LD    HL,BUFFER
    C739        77                51           LD    (HL),A
    C73A        010100            52           LD    BC,1
    C73D        CD0B00            53           CALL  OUTCHR           SKRIV UT TECKEN
    C740        C9                54           RET
                                  55  ;
    C741        5E                56  MARK     DEFB  "ü"
                                  57  ;
    C742        C700              58           END   BEGIN

      ERRORS :    0   WARNINGS :    0
```

```
ASMZ-4.41 PROGRAMEXEMPEL                                ERRORS:  0   00:00:00   00-00-00   PAGE  2

    CROSS REFERENCE LISTING

    SYMBOL      LOCATION   DECL   REFERENCES ----------------------------------------------------------

     A          C712         28   32
     BEGIN      C700         16   58
     BUFFER     C711         27   50
     INCHR      0002         11   23
     LOOP       C709         23   25
     MARK       C741         56   46
     OUTCHR     000B         12   18   48   53
     SOUT1      C736         50   43
     SOUTCH     C722         41   24
     TEXT       C712         29   17
     TEXTLEN    0010         32   16
```

LUXOR
Datorer

Luxor Datorer AB · Box 923 · 591 29 Motala · Tel. 0141-162 00

REGIONKONTOR:

Stockholm, tel. 08-84 04 90 · Göteborg, tel. 031-42 07 20 · Malmö, tel. 040-18 10 20 · Örnsköldsvik, tel. 0660-150 44