A S M 5 1 - Assemblerdirektiven

Grete PESCHEK, TGM, NT/EL

1. Allgemeines

Ein Assemblerprogramm kann aus drei verschiedenen Typen von Assembleranweisungen bestehen:

- **Maschinenbefehl**: wird direkt durch Maschinencode ersetzt, entspricht einem vom Prozessor direkt durchführbaren Befehl.
- **Assemblerdirektive**: wird verwendet um die Programmstruktur und Symbole zu definieren; erzeugen einen nicht ausführbaren Code.
- **Assembler controls**: Assemblierungsmode wird gesetzt.

Segmente, Module, Programme

Ein Segment ist ein Block von Befehlen oder Daten im Speicher. Ein Segment kann

- verschiebbar, **relocatable** oder
- nicht verschiebbar, **absolute**

sein. Ein **verschiebbares Segment** hat eine Namen, einen Typ und eventuell andere Attribute. Segmente mit gleichen Namen, aber von verschiedenen Modulen, sind Teile des selben Segments und werden **partielle Segmente** genannt. Partielle Segmente werden mit Segmenten durch den Binder(Linker) RL51 verbunden. Ein **absolutes Segment** hat keinen Namen und kann mit keinem anderen Segment kombiniert werden.

Ein **Modul** besteht aus einem oder mehreren Segmenten oder partiellen Segmenten. Dem Modul wird durch den User ein Name zugewiesen. Ein **Objektfile** enthält einen oder mehrere Module.

Ein **Programm** besteht aus einem einzigen absoluten Modul, der aus allen absoluten und verschiebbaren Segmenten zusammengemischt ist.

2. Operanden und Ausdrücke

Operanden:

Allgemein ist eine Befehlszeile wie folgt aufgebaut:

```
[label:] Mnemonic [operand][,operand][,operand]  [;comment]
```

Jeder Operand ist aus einer der folgenden Klasse:

- Spezielles Assemblersymbol, reserviertes Wort
- Indirekte Adresse, Inhalt eines Registers ist Datenadresse
- Immediate \
- Datenadresse \ Numerischer Ausdruck
- Bitadresse / Symbolisch oder
- Codeadresse /direkt als Wert berechenbar

Die fünf Adressbereiche haben folgende vordefinierte Namen:

- `DATA` Directly addressable data address space
- `BIT` Bit address space
- `XDATA` External data address space
- `CODE` Code address space
- `IDATA` Indirectly addressable data space

Berechnung von Ausdrücken zur Übersetzungszeit

Ein Ausdruck ist eine Kombination von Zahlen, Zeicheketten, Symbolen und Operatoren, dessen Wert letzlich in einer 16-Bit Zahl abgelegt werden kann.

Zahlen:		
	`B`	Binary
	`O,Q`	Octal
	`D` oder nichts	Decimal
	`H`	Hexadecimal

3. Assembler Direktiven

(1) Symboldefinitionen

```
SEGMENT, EQU, SET, DATA, IDATA, XDATA, BIT, CODE
```

Mit Hilfe von Symboldefinitionen können Namen für Segmente, Register, Zahlen und Adressen vereinbart werden. Keine dieser Direktiven hat ein Labelfeld. Außer der `SET`-Direktive kann keine, wenn einmal gesetzt, verändert werden.

SEGMENT-Direktive

```
relocatable_segment_name SEGMENT segment type [relocation
type]
```

Segmenttyp:

```
CODE/XDATA/DATA/IDATA/BIT
```

Relocationtyp:

`PAGE` Segment beginnt an einer 256-Byte Grenze, `INPAGE` muß innerhalb einer Seite liegen (nur für `CODE`, `XDATA`), `INBLOCK` muß innerhalb eines Block (2048-Byte) liegen, `BITADDRESSABLE` innerhalb vom bitadressierbaren Bereich auf Bytegrenze, `UNIT` Grenze auf Unit ausgerichtet, das ist beim Bit-Segment ein Bit, sonst Byte. Der `UNIT`-Typ wird als Defaultrelocationtyp verwendet. z.B.:

```
PART1   SEGMENT   CODE
```

EQU-Direktive

```
symbol_name EQU expression
```

oder

```
symbol_name EQU special_assembler symbol
```

Weist dem Symbolnamen einen Ausdruck zu. Der so vereinbarte Symbolname kann als symbolische Adresse in einem Daten-, Code-, Bit- oder externen Adreßsegment abhängig von Typ des Segmentes in dem es vereinbart wurde, verwendet werden. Ist der Wert des Ausdrucks eine Zahl, so kann dieses Symbol irgendwo verwendet werden.

Folgende speziellen Assemblersymbole können durch die EQU-Direktive neu vereinbart werden: A, R0, R1, R2, R3, R4, R5, R6, R7. z.B.:

```
REG1     EQU    R1
```

REG1 kann anstelle von R1 verwendet werden

```
ANZAHL   EQU    27
```

Statt der Zahl 27 kann das Symbol ANZAHL verwendet werden.

SET-Direktive

```
symbol_name SET expression

symbol_name SET special_assembler symbol
```

Ähnlich EQU, jedoch kann der Symbolname später neuerlich, duch eine weitere SET-Direktive definiert werden. z.B:

```
ZAEHLER   SET   0           ; Initialisierung des Zählers
ZAEHLER   SET   ZAEHLER+1   ; Inkrementieren des Zählers
```

BIT-Direktive

```
symbol_name BIT bit_address
```

Es wird einer Adresse im bitadressierbarem Speicher ein symbolischer Name zugeordnet. Zuordnung darf nicht durch eine weitere BIT-Direktive gändert werden. z.B.:

```
RSEG   DATA_SEG       ; relokatibles bitadressierbares
                      ; Segment
FLAGS:     DS      1
   ...
ALARM      BIT     FLAGS.0        ; Bit 0 im Byte FLAGS
LICHT_EIN  BIT     ALARM+1        ; Bit 1 im Byte FLAGS
DREHE_EIN  BIT     060H           ; absolutes Bit
```

DATA-Direktive

```
symbol_name  DATA  expression
```

Ordnet einer Adresse im internen Datenspeicher einen symbolischen Namen zu. Ergibt der Ausdruck eine Wert größer als 127, so wird ein symbolischer Name für ein Register vereinbart. Die Zuordnung darf nicht durch eine weitere DATA-Direktive im Programm verändert werden. z.B:

```
TABELLE   DATA   70H
TAB_END   DATA   7FH
```

XDATA-Direktive

```
symbol_name   XDATA   expression
```

Ordnet einer Adresse im externen Datenspeicher einen symbolischen Namen zu. Die Zuordnung darf nicht durch eine weitere XDATA-Direktive im Programm verändert werden.

IDATA-Direktive

```
symbol_name   IDATA   expression
```

Ordnet einer indirekten internen Adresse einen symbolischen Namen zu. Beim 8051 darf der Wert des Ausdrucks nicht größer als 127 sein. Die Zuordnung darf nicht durch eine weitere IDATA-Direktive im Programm verändert werden. z.B:

```
BUFFER      IDATA   60H
BUFFER_LEN  EQU     20H
BUFFER_END  IDATA   BUFFER+BUFFER_LEN-1
```

CODE

```
symbol_name   CODE   expression
```

Ordnet einer Adresse in Programmspeicher einen symbolischen Namen zu. Die Zuordnung darf nicht durch eine weitere CODE-Direktive im Programm aufgehoben werden. z.B.:

```
RESTART   CODE   00H
UPR1      CODE   70H
```

(2) Speicherinitialisierung und Speicherreservierung

```
DS, DBIT, DB, DW
```

DS-Direktive

```
[label:] DS   expression
```

Reservierung eines Bytebereiches. In jedem Segment, ausser in einem Bitsegment anwendbar.

DBIT-Direktive

```
[label:] DBIT expression
```

Vereinbarung von einem Bitbereich. Nur im bitadressierbaren Speicher möglich.

DB-Direktive

```
[label:] DB   expression list
```

Initialisierung eines Byte-Bereiches in einem Code-Segment. z.B.:

```
PRIM:  DB  1,2,3,5,7,11,13,17,19,23
ALTER: DB  'ERICH',22,'HANS',23
```

DW-Direktive

```
[label:] DW   expression
```

Initialisierung eines Wort-Bereiches in einem Code-Segment. z.B.:

```
ZWEIP:  DW  1024,2048,4096
SPRUNG: DW  UPR1,UPR2,UPR3
```

(3) Programm Binden (Program Linkage)

PUBLIC-Direktive

```
PUBLIC  list_of_names
```

Muß angegeben werden, wenn ein verwendetes Symbol außerhalb des zu assemblierenden Moduls definiert ist. z.B.:

```
PUBLIC Upr1_Name,Upr2_name
PUBLIC Anzahl1,Anzahl2
```

```
EXTRN-Direktive

EXTRN [segment_type(list_of_symbol_names)], ...
```

Mit der EXTRN-Direktive wird ein Bezug zwischen Symbolen, die in anderen Modulen definiert wurden und in diesem Modul verwendet werden, hergestellt. z.B.:

```
EXTRN  CODE(Mod_A,Mod_B)
```

NAME-Direktive

```
NAME module_name
```

Gibt dem aktuellen Modul eine Bezeichnung.

(4) Assembler Status Steuerung

END-Direktive

```
END
```

Darf keinen Label haben. Ist die letzte Anweisung in einem Programm.

ORG-Direktive

```
ORG expression
```

Durch die ORG-Direktive wird der Adreßzähler des Assemblers neu gesetzt. Nachfolgende Befehle beginnen mit der angegebenen Adresse. Direktive darf keinen Label haben.

(5) Segment Selection Directive

```
RSEG segment_name
```

Gibt an, daß nachfolgende Befehle im verschiebbaren Segment `segment_name` liegen.

```
CSEG/XSEG/DSEG/ISEG/BSEG   AT absolute_address
```

Gibt an, daß nachfolgende Befehle im absoluten Segment ab der Adresse `absolute_address` liegen. z.B.:

```
DATA_SEG1   SEGMENT  DATA     ; verschiebbares Datensegment

CODE_SEG1   SEGMENT  CODE     ; verschiebbares Codesegment

            BSEG  AT 70H      ; absolutes Bitsegment

DECIMAL_MODE: DBIT      1     ; absolutes Bit

CHAR_MODE:    DBIT      1

           RSEG     DATA_SEG1 ; selektiert das verschiebbare
                              ; Datensegment

TOTAL1:      DS        1
COUNT1:      DS        1

           RSEG     CODE_SEG1 ; selektiert das verschiebbare
                              ; Codesegment

BEGIN_CODE:
```

COMPUTER-BREVIER

Computer unser, der du bist im Rechenzentrum,
gereinigt werde dein Bildschirm.
Deine Vernetzung komme, dein COMMAND.COM geschehe,
wie im Hauptspeicher, so auf der Festplatte.
Unser tägliches Reset gib uns heute,
und vergib uns unsere Eingabefehler,
wie auch wir vergeben unseren schlampigen Programmierern.
Und führe uns nicht in die Endlosschleife,
sondern erlöse uns von allen Viren.
Denn dein ist die Floppy und der Drucker und der Joystick.
In Ewigkeit. ENTER.

(Gesehen in einem Klassenraum von "Arthur F.", modifiziert von Sepp Melchart)