CC410: System Programming

Dr. Manal Helal – Fall 2014 – Lecture 8 - Assembler 3

Learning Objectives

- Understand Assemblers machine independent features

Machine-Independent Assembler Features

- **Symbol Defining Statement**
- **Expressions**
- **Program Blocks (next lecture)**
- **Control Sections and Program Linking (next lecture)**

2.3.2 Symbol-Defining Statements

- Allow the programmer to define symbols and specify their values.
  - Assembler directive EQU.
  - Improved readability in place of numeric values.

```
          +LDT  #4096
MAXLEN EQU  4096
          +LDT  #MAXLEN
```

- Use EQU in defining mnemonic names for registers.
  - Registers A, X, L can be used by numbers 0, 1, 2.

```
          RMO  A, X
```

```
A         EQU       0
X         EQU       1
L         EQU       2
```

2.3.2 Symbol-Defining Statements

- The standard names reflect the usage of the registers.

```
BASE    EQU   R1
COUNT   EQU   R2
INDEX   EQU   R3
```

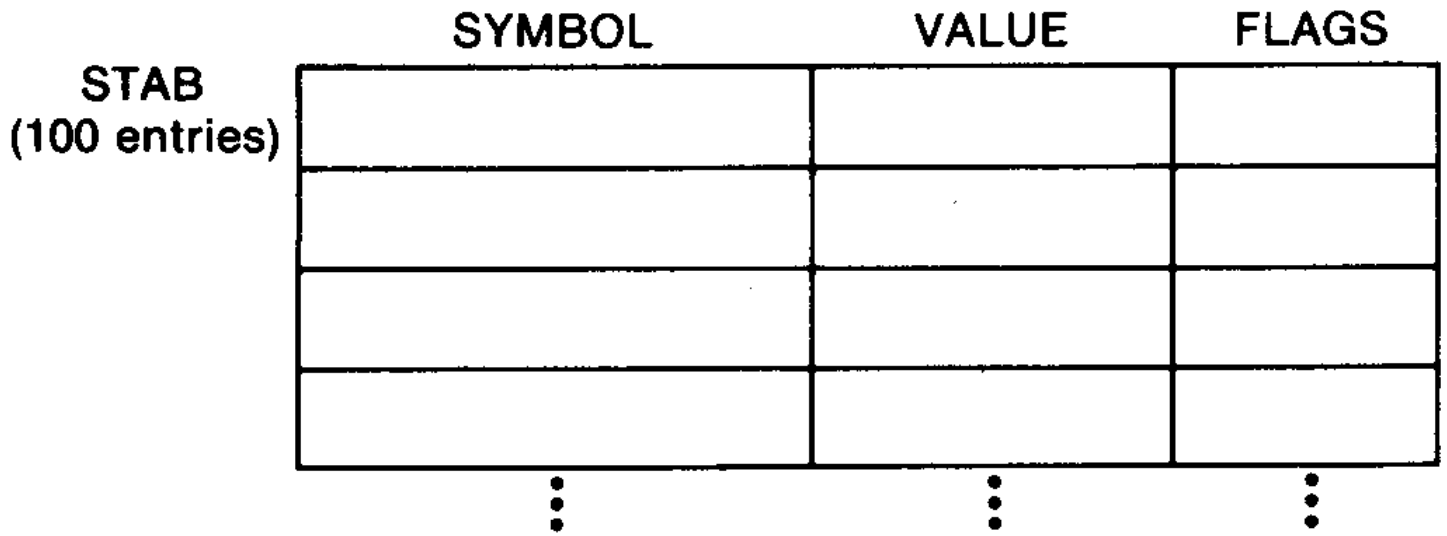
- Assembler directive ORG
  - Use to indirectly assign values to symbols.

```
ORG   value
```

  - The assembler resets its LOCCTR to the specified value.
  - ORG can be useful in label definition.

2.3.2 Symbol-Defining Statements

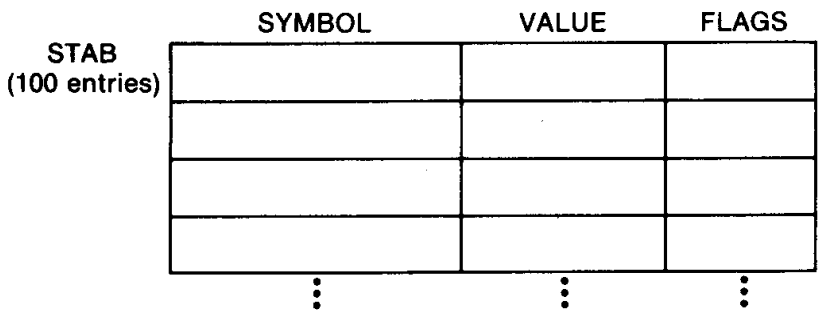
- The location counter is used to control assignment of storage in the object program
  - In most cases, altering its value would result in an incorrect assembly.
- ORG is used
  - SYMBOL is 6-byte, VALUE is 3-byte, and FLAGS is 2-byte.



2.3.2 Symbol-Defining Statements

STAB	SYMBOL	VALUE	FLAGS
(100 entries)	6	3	2

```
1000        STAB      RESB     1100
2100                  ORG      STAB
1000        SYMBOL    RESB     6
1006        VALUE     RESW     1
1009        FLAGS     RESB     2
1000                  ORG      STAB+1100
```



Equivalent to:

```
1000        STAB      RESB     1100
2100        SYMBOL    EQU      STAB
2100        VALUE     EQU      STAB +6
2100        FLAGS     EQU      STAB +9
```

» Use LDA VALUE,X to fetch the VALUE field form the table entry indicated by the contents of register X.

2.3.2 Symbol-Defining Statements

- All terms used to specify the value of the new symbol — must have been defined previously defined in the program. Forward reference will require more passes as follows:

```
BETA    EQU     ALPHA
ALPHA   RESW    1
```

Need 2 passes

- *Another Example:*

```
ALPHA   EQU     BETA
BETA    EQU     DELTA
DELTA   RESW1
```

Need 3 passes

2.3.2 Symbol-Defining Statements

- All symbols used to specify new location counter value must have been previously defined. The following code is incorrect:

```
        ORG         ALPHA
BYTE1   RESB 1
BYTE2   RESB 1
BYTE3   RESB 1
        ORG
ALPHA   RESW        1
```

2.3.3 Expressions

- Allow arithmetic expressions formed
  - Using the operators +, -, *, /.
  - Division is usually defined to produce an integer result.
  - Expression may be constants, user-defined symbols, or special terms.

```
106    1036        BUFEND    EQU    *
```

  - Gives BUFEND a value that is the address of the next byte after the buffer area.
- Absolute expressions or relative expressions
  - A relative term or expression represents some value (S +r), S: starting address, r: the relative value.

2.3.3 Expressions

```
107    1000      MAXLEN    EQU    BUFEND-BUFFER
```

- Both BUFEND and BUFFER are relative terms.
- The expression represents absolute value: the *difference* between the two addresses.
- Loc =1000 (Hex)
- The value that is associated with the symbol that appears in the source statement.
- BUFEND+BUFFER, 100-BUFFER, 3*BUFFER represent neither absolute values nor locations.

» Symbol tables entries

Symbol	Type	Value
RETADR	R	0030
BUFFER	R	0036
BUFEND	R	1036
MAXLEN	A	1000