Assemblers (continued)

Program Relocation:

Example 1: Suppose the START directive in a program for SIC specifies 100, the instruction

```
LDA    THREE
```

has LC value = 103, and the symbol THREE has LC value = 115. The assembled form of the instruction (in hex) is 000073.

- The above instruction uses *absolute* addresses.
- Works correctly as long as the program starts from location 100.
- Requirement too rigid in a *multiprogramming* environment.
- **Relocatable program:** A program that works correctly regardless of starting address.
- Assembler should produce relocatable object code.

- Assembler assumes the starting address to be zero.
- Assembler identifies parts of the object program that need to be modified when the program is relocated.

Modification Record:

- Each time the assembler produces an instruction with an address, a modification record is produced.
- Each modification record contains
  (a) Starting location of the address field to be modified.
  (b) Length of the address field (in say, bytes, half-bytes or bits).
- Modification records are appended to the object code.

Example 2: Relocatable translation for the instruction in Example 1.

- The object code produced is 00000F.
- Modification record: Starting location = 4 and Length = 15 (bits).

Relocation for SIC and SIC/XE:

- SIC: All instructions except RSUB and I/O instructions cause a modifier record to be written.
- SIC/XE: Only 4-byte instructions may cause a modifier record to be written.

Errors Detected by Assemblers:

- Undefined symbols. (These are usually indicated at the end of the program listing.)
- Multiply defined symbols.
- Illegal opcode.
- Missing or extra operands.
- Relative addressing infeasible (SIC/XE).

Literals:

- Literal: Constant operand written as part of the instruction.

Example:

```
        LDA     =C'PQR'
        TD      =X'05'
```

- Literals are different from immediate operands.

Example: The SIC/XE statement

```
        LDA     #112
```

will be assembled into the 3-byte instruction 010070 (hex). However, the statement

```
        LDA     =C'PQR'
```

is equivalent to

```
        LDA     LIT1
          .
          .
LIT1    BYTE    =C'PQR'
```

- Assembler may create new (special) labels for literals.
- Normally, a "literal pool" is created at the end of a program.
- Assembler may choose to have the literal pool at a different point (instead of at the end) to allow PC-relative addressing.
- A special directive LTORG used for this purpose.

Example:

```
          LDA       =C'PQR'
            .
            .
          J         NEXT
          LTORG
            .
            .
NEXT      LDB       #80
```

- Some assemblers use a Literal Table to save space for duplicate literals.

Symbol Defining Directives:

- EQU directive is useful in defining constants.

Example:

```
MAXADR    EQU    65535
```

- Allows instructions such as

```
          LDA    #MAXADR
```

- Symbol Table can be used to handle such constants.

Handling Expressions:

- Expressions may involve constants.

Example:

```
NREC      EQU    200
RSIZE     EQU    15
            .
            .
LOC       RESW   NREC*SIZE
```

- Expressions may also involve addresses.

Example:

```
STA     START+2
```

- Expression evaluation needed in both passes.
  - Values of expressions used with RESW and RESB directives must be computed in Pass 1.
  - Values of expressions such as `START+2` may need to be computed in Pass 2.
- Expression evaluation algorithm:
  1. Convert expression to postfix form.
  2. Evaluate postfix expression using a stack.

One-Pass Assemblers:

- One-pass: Assembler makes only one physical pass over the source file.
- Main problem: Forward references.
- Two types: Load-and-go and Object file assemblers.

(a) Load-and-Go Assemblers:

- Intermediate version and final object code are kept in main memory.
- Program can begin execution right after assembly.
- To handle forward references, modify Symbol Table (ST).
- Each ST entry for a two-pass assembler has symbol and LC value.
- Each ST entry for a one-pass assembler has:
  - Symbol
  - Defined? (Boolean flag)
  - LC Value
  - Pointer to the list of locations where the LC value for the symbol is needed.

Outline of Algorithm: See Handout 3.1.

Example: To be discussed in class using the program segment in Handout 3.2.

(b) Object File Assemblers:

- Object code bytes written out to the file are "unavailable" for patching.
- Patching is done at run time (by the loader).
- As in load-and-go assemblers, use the modified symbol table and store forward references as linked lists.
- When a symbol gets defined, output a text record for each forward reference of the symbol using the list.

Example: To be discussed in class using the program segment in Handout 3.2.

Multi-pass Assemblers:

- Two passes may not be enough.

Example:

```
ALPHA     EQU    BETA
DELTA     EQU    ALPHA
            .
            .
ARRAY     RESW   ALPHA
            .
            .
DELTA     EQU    24
```

- Multi-pass assemblers are not common.
  - Assembly takes more time.
  - Handling ST requires additional overhead. (A symbol may appear in the label field many times.)
  - Such programs are more difficult to understand.