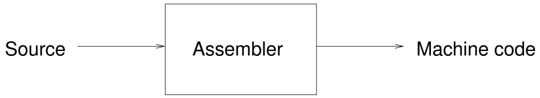
CSI 402 – Lecture 4

(Assemblers)

Assembler Basics

Ref: Chapter 2 of [Beck].

Basic Function:



- Precursor to compilers.

Conventions in SIC and SIC/XE:

- Lines starting with '.' are comments.
- Fields in a statement: Label (optional), Opcode, Operands (if needed) and Comment (optional).
- Assembler directives (or pseudo opcodes): `START`, `END`, `BYTE`, `WORD`, `RESB`, `RESW`, `BASE` and `NOBASE`.

Conventions in SIC and SIC/XE (continued)

- Indirect addressing indicated by '@':

```
JEQ     @RADDR
```

- Immediate operands are indicated by '#':

```
LDA     #50
```

- BASE directive used to indicate base + displacement mode.

```
LDB     #LEN
BASE    LEN
```

- NOBASE directive ends base + displacement mode; that is, the BASE directive is in effect for the segment between BASE and NOBASE directives.

- 4-byte instruction specified by preceding the opcode with '+'.

```
+LDA    MEM
```

Assembling a SIC Program

Note: Line numbers shown below are for convenience; they are not part of the program.

```
(00)      MCHRS     START    1000
(01)      FIRST     LDX      ZERO
(02)      MOVECH    LDCH     STR1,X
(03)                STCH     STR2,X
(04)                TIX      ELEVEN
(05)                JLT      MOVECH
(06)                RSUB
(07)      STR1      BYTE     C'TEST STRING'
(08)      STR2      RESB     11
(09)      ZERO      WORD     0
(10)      ELEVEN    WORD     11
(11)                END      FIRST
```

Assembling a SIC Program (continued)

Note: The SIC program on the previous slide is shown below with Location Counter (LC) values (in decimal) for each line.

```
LC Value

1000           MCHRS      START    1000
1000           FIRST      LDX      ZERO
1003           MOVECH     LDCH     STR1,X
1006                      STCH     STR2,X
1009                      TIX      ELEVEN
1012                      JLT      MOVECH
1015                      RSUB
1018           STR1       BYTE     C'TEST STRING'
1029           STR2       RESB     11
1040           ZERO       WORD     0
1043           ELEVEN     WORD     11
1046                      END      FIRST
```

Assembling a SIC Program (continued)

Note: The symbol table for the SIC program on the previous slide is shown below. (The LC values are shown in decimal.)

Symbol	**LC Value**
`MCHRS`	1000
`FIRST`	1000
`MOVECH`	1003
`STR1`	1018
`STR2`	1029
`ZERO`	1040
`ELEVEN`	1043

Assembling a SIC Program (continued)

Observations:

- Use of Location Counter (LC).
- Two pass assembly (to allow forward referencing).
- Symbol Table (ST) used for labels.

Summary of Actions:

Pass 1:

- Assign addresses to instructions using LC.
- Build Symbol Table.
- Process pseudo opcodes.
- Produce partial machine code. (This may also be done in Pass 2.)

Summary of Actions (continued)

Pass 2:

- Complete assembly of instructions (resolving forward references).
- Output object program and listing to appropriate files.
- Generate information for linker (to be seen later).

Pseudocode for Passes 1 and 2:

- Figures 2.4(a) and 2.4(b) of [Beck]: Reading assignment.

Tables Used by the Assembler

(A) Symbol Table:

- Each entry has a symbol and its LC value.
- Searched in both Pass 1 and Pass 2.
- Dynamic table (grows in Pass 1).
- Efficient implementation essential.

(B) Machine Opcode Table (MOT):

- Each entry has a mnemonic, binary opcode and instruction length.
- Must be searched in Pass 1.
- Static table (contents don't change).

Format of object Code

- **Header Record:** Contains program name, starting address and length (in bytes).
- **Text Record:** Contains starting address for the code in the record, length of the code and the code itself.
- **End Record:** Contains the address of the first executable instruction in the object program.

Note: Additional details regarding the structure of the above records are discussed in [Beck, pages 48–49].

Modifications for SIC/XE

Relative Addressing: Can be used only with 3-byte instructions. (Assembler must set the `b` and `p` bits appropriately.)

- Instructions may use base-relative or PC-relative modes.
- If BASE directive is in effect, assembler uses that mode; otherwise PC-relative mode is used.

Exception: PC-relative mode may be used even when BASE is in effect, if the required displacement is negative.

Modifications for SIC/XE (continued)

Example:

```
        LDB   #LEN
        BASE  LEN
          .      <---- SIC/XE code
          .
        JEQ   NEXT
        STA   SAVE
NEXT    LDA   VAL
          .      <---- SIC/XE code
          .
LEN     RESW  1
SAVE    RESW  1
```

Notes:

- For the JEQ instruction, PC-relative addressing is used even though BASE directive is in effect.
- For the STA instruction, base-relative addressing is used.

Modifications for SIC/XE (continued)

Note: Assembler produces an error message if neither base-relative nor PC-relative modes can be used. (How can this happen?)

Computing Displacement:

- **PC-relative:** Displacement may be positive or negative. (Examples to be discussed in class.)
- **Base-relative:** Idea similar to PC-relative mode; displacement cannot be negative.

Extended Addressing (3-byte Instructions):

- **Immediate mode:** Set the `i`-bit to 1, `n`-bit to 0; store the operand itself in the 12-bit displacement field.
- **Indirect mode:** Set the `i`-bit to 0, `n`-bit to 1; store the displacement of the operand in the 12-bit field.

Modifications for SIC/XE (continued)

Handling 4-byte Instructions:

- Opcode has the `'+'` prefix to indicate 4-byte format.
- The `e`-bit must be set to 1.
- The `b`-bit and the `p`-bit are both set to 0.
- The address (or the immediate operand) is stored in the least significant 20 bits.
- The `i`-bit and the `n`-bit are set appropriately.

Suggested Exercises

1. Consider SIC and SIC/XE program segments from [Beck] (e.g. program segments on pages 111 and 114). Compute the LC values for the instructions. Also construct the symbol table for those program segments.
2. Construct an example of a SIC/XE program segment where the assembler cannot use either base-relative or PC-relative addressing. (This question is mentioned on Slide 4-13.)
3. Understand the various functions of an assembler by constructing object code <u>manually</u> for various assembly language instructions of SIC and SIC/XE.