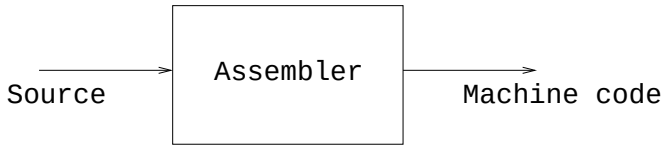
Assemblers

Ref: Chapter 2 of [Beck].

Basic function:



- Precursor to compilers.

Conventions in SIC:

- 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, NOBASE.

- 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. (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
```

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.)

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 text: Reading assignment.

Tables Used:

(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.

Modifications for SIC/XE:

(1) Relative Addressing: Can be used only with 3-byte instructions. (Assembler must set `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.

Example:

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

- 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.

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 and 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.

Handling 4-byte Instructions:

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