Assembler Features

- Machine Dependent Assembler Features
  - Instruction formats and addressing modes (SIC/XE)
  - Program relocation
- Machine Independent Assembler Features
  - Literals
  - Symbol-defining statements
  - Expressions
  - Program blocks
  - Control sections and program linking

Instruction Formats and Addressing Modes

- Instruction formats depends on memory organization and the size of memory.
- In SIC:
  - only one instruction format.
  - only two registers : register A and Indexed register
  - support three addressing modes: Direct, Indirect and Indexed.
- In SIC/XE:
  - four different types of instruction

A SIC/XE Program

```
COPY    START   0           COPY FILE FROM INPUT TO OUTPUT
FIRST   STL     RETADR      SAVE RETURN ADDRESS
        LDB     #LENGTH     ESTABLISH BASE REGISTER
        BASE    LENGTH
CLOOP  +JSUB    RDREC       READ INPUT RECORD
        LDA     LENGTH      TEST FOR EOF (LENGTH = 0)
        COMP    #0
        JEQ     ENDFIL      EXIT IF EOF FOUND
       +JSUB    WRREC       WRITE OUTPUT RECORD
        J       CLOOP       LOOP
ENDFIL  LDA     EOF         INSERT END OF FILE MARKER
        STA     BUFFER
        LDA     #3          SET LENGTH = 3
        STA     LENGTH
       +JSUB    WRREC       WRITE EOF
        J       @RETADR     RETURN TO CALLER
EOF     BYTE    C'EOF'
RETADR  RESW    1
LENGTH  RESW    1           LENGTH OF RECORD
BUFFER  RESB    4096        4096-BYTE BUFFER AREA
.
```

```
115   .        SUBROUTINE TO READ RECORD INTO BUFFER
120   .
125   RDREC    CLEAR   X           CLEAR LOOP COUNTER
130            CLEAR   A           CLEAR A TO ZERO
132            CLEAR   S           CLEAR S TO ZERO
133           +LDT    #4096
135   RLOOP    TD      INPUT       TEST INPUT DEVICE
140            JEQ     RLOOP       LOOP UNTIL READY
145            RD      INPUT       READ CHARACTER INTO REGISTER A
150            COMPR   A,S         TEST FOR END OF RECORD (X'00')
155            JEQ     EXIT        EXIT LOOP IF EOR
160            STCH    BUFFER,X    STORE CHARACTER IN BUFFER
165            TIXR    T           LOOP UNLESS MAX LENGTH
170            JLT     RLOOP         HAS BEEN REACHED
175   EXIT     STX     LENGTH      SAVE RECORD LENGTH
180            RSUB                RETURN TO CALLER
185   INPUT    BYTE    X'F1'       CODE FOR INPUT DEVICE
195   .
```

```
.
.        SUBROUTINE TO WRITE RECORD FROM BUFFER
.
WRREC    CLEAR   X            CLEAR LOOP COUNTER
         LDT     LENGTH
WLOOP    TD      OUTPUT       TEST OUTPUT DEVICE
         JEQ     WLOOP        LOOP UNTIL READY
         LDCH    BUFFER,X     GET CHARACTER FROM BUFFER
         WD      OUTPUT       WRITE CHARACTER
         TIXR    T            LOOP UNTIL ALL CHARACTERS
         JLT     WLOOP          HAVE BEEN WRITTEN
         RSUB                 RETURN TO CALLER
OUTPUT   BYTE    X'05'        CODE FOR OUTPUT DEVICE
         END     FIRST
```

SIC/XE Instruction Formats and Addressing Modes

• PC-relative or Base-relative (BASE directive needs to be used) addressing:	op m
• Indirect addressing:	op @m
• Immediate addressing:	op #c
• Extended format (4 bytes):	+op m
• Index addressing:	op m,X
• Register-to-register instructions	COMPR

Relative Addressing Modes

- PC-relative or base-relative addressing mode is preferred over direct addressing mode.
  - Can save one byte from using format 3 rather than format 4.
    - Reduce program storage space
    - Reduce program instruction fetch time
  - Relocation will be easier.

The Differences Between the SIC and SIC/XE Programs

- Register-to-register instructions are used whenever possible to improve execution speed.
  - Fetch a value stored in a register is much faster than fetch it from the memory.
- Immediate addressing mode is used whenever possible.
  - Operand is already included in the fetched instruction. There is no need to fetch the operand from the memory.
- Indirect addressing mode is used whenever possible.
  - Just one instruction rather than two is enough.

The Object Code for Fig. 2.5

Line	Loc	Source statement			Object code
5	0000	COPY	START	0	
10	0000	FIRST	STL	RETADR	17202D
12	0003		LDB	#LENGTH	69202D
13			BASE	LENGTH	
15	0006	CLOOP	+JSUB	RDREC	4B101036
20	000A		LDA	LENGTH	032026
25	000D		COMP	#0	290000
30	0010		JEQ	ENDFIL	332007
35	0013		+JSUB	WRREC	4B10105D
40	0017		J	CLOOP	3F2FEC
45	001A	ENDFIL	LDA	EOF	032010
50	001D		STA	BUFFER	0F2016
55	0020		LDA	#3	010003
60	0023		STA	LENGTH	0F200D
65	0026		+JSUB	WRREC	4B10105D
70	002A		J	@RETADR	3E2003
80	002D	EOF	BYTE	C'EOF'	454F46
95	0030	RETADR	RESW	1	
100	0033	LENGTH	RESW	1	
105	0036	BUFFER	RESB	4096	

Relocatable program

```
                        .
                        .        SUBROUTINE TO READ RECORD INTO BUFFER
                        .
        1036    RDREC   CLEAR    X                      B410
        1038            CLEAR    A                      B400
        103A            CLEAR    S                      B440
        103C           +LDT      #4096                  75101000
        1040    RLOOP   TD       INPUT                  E32019
        1043            JEQ      RLOOP                  332FFA
        1046            RD       INPUT                  DB2013
        1049            COMPR    A,S                    A004
        104B            JEQ      EXIT                   332008
        104E            STCH     BUFFER,X               57C003
        1051            TIXR     T                      B850
        1053            JLT      RLOOP                  3B2FEA
        1056    EXIT    STX      LENGTH                 134000
        1059            RSUB                            4F0000
        105C    INPUT   BYTE     X'F1'                  F1
```

```
195                 .
200                 .          SUBROUTINE TO WRITE RECORD FROM BUFFER
205                 .
210     105D    WRREC      CLEAR      X                 B410
212     105F               LDT        LENGTH            774000
215     1062    WLOOP      TD         OUTPUT            E32011
220     1065               JEQ        WLOOP             332FFA
225     1068               LDCH       BUFFER,X          53C003
230     106B               WD         OUTPUT            DF2008
235     106E               TIXR       T                 B850
240     1070               JLT        WLOOP             3B2FEF
245     1073               RSUB                         4F0000
250     1076    OUTPUT     BYTE       X'05'             05
255                        END        FIRST
```

Generate Relocatable Programs

- Let the assembled program starts at address 0 so that later it can be easily moved to any place in the physical memory.
  - Actually, as we have learned from virtual memory, now every process (executed program) has a separate address space starting from 0.
- Assembling register-to-register instructions presents no problems. (e.g., line 125 and 150)
  - Register mnemonic names need to be converted to their corresponding register numbers.
  - This can be easily done by looking up a name table.

PC or Base-Relative Modes

- Format 3: 12-bit displacement field (in total 3 bytes)
  - Base-relative: 0~4095
  - PC-relative: -2048~2047
- Format 4: 20-bit address field (in total 4 bytes)
- The displacement needs to be calculated so that when the displacement is added to PC (which points to the following instruction after the current instruction is fetched) or the base register (B), the resulting value is the target address.
- If the displacement cannot fit into 12 bits, then format 4 needs to be used. (E.g., line 15 and 125)
  - Bit e needs to be set to indicate format 4.
  - A programmer must specify the use of format 4 by putting a + before the instruction. Otherwise, it will be treated as an error.

PC-Relative Example (1)

10	0000	FIRST STL	RETADR	17202D
12	0003			

op(6)	n	i	x	b	p	e	disp(12)
$(14)_{16}$	1	1	0	0	1	0	$(02D)_{16}$

(0001 0111) (0010 0000) $(2D)_{16}$

$(17)_{16}$ $(20)_{16}$ $(2D)_{16}$

displacement= RETADR - PC = 30 - 3 = 2D

After fetching this instruction and before executing it, the PC will be 0003.

PC-Relative Example (2)

40	0017	J	CLOOP	3F2FEC
45	001A	…….		

op(6)	n	i	x	b	p	e	disp(12)

$(3C)_{16}$ 1 1 0 0 1 0 $(FEC)_{16}$

(0011 1111) (0010 1111) $(FC)_{16}$

$(3F)_{16}$ $(2F)_{16}$ $(FC)_{16}$

Displacement = CLOOP - PC= 6 - 1A= -14 = FEC

Base-Relative v.s. PC-Relative

- The difference between PC and base relative addressing modes is that the assembler knows the value of PC when it tries to use PC-relative mode to assembles an instruction. However, when trying to use base-relative mode to assemble an instruction, the assembler does not know the value of the base register.
  - Therefore, the programmer must tell the assembler the value of register B.
  - This is done through the use of the BASE directive. (line 13)
  - Also, the programmer must load the appropriate value into register B by himself.
  - Another BASE directive can appear later, this will tell the assembler to change its notion of the current value of B.
  - NOBASE can also be used to tell the assembler that no more base-relative addressing mode should be used.

Base-Relative Example

The address of LENGTH,
not the content of LENGTH
(69202D)

12		LDB	#LENGTH	
13		BASE	LENGTH	
160	104E	STCH	BUFFER, X	57C003

op(6)	n	i	x	b	p	e	disp(12)
$(54)_{16}$	1	1	1	1	0	0	$(003)_{16}$
(01010111)	(1100)						
(57)	(C0)						(03)

Displacement = BUFFER - B = 0036 - 0033 = 3

Immediate Addressing Example (1)

12   0003   LDB   #LENGTH   69202D

op(6)	n	i	x	b	p	e	disp(12)
$(68)_{16}$	0	1	0	0	1	0	$(02D)_{16}$

(0110) (10 0 1) (0 0 1 0) $(02D)_{16}$

(69)   (20)   (2D)

- The immediate operand is the value of the symbol LENGTH, which is its address (not its content)
- Displacement = 0033 – 0006 = 002D
- If immediate mode is specified, the target address becomes the operand.

Immediate Addressing Example (2)

55 0020 LDA #3 010003

op(6)	n	i	x	b	p	e	disp(12)
$(00)_{16}$	0	1	0	0	0	0	$(003)_{16}$

133 103C +LDT #4096 75101000

op(6)	n	i	x	b	p	e	disp(12)
$(74)_{16}$	0	1	0	0	0	1	$(01000)_{16}$

Indirect Addressing Example

- The target address is computed as usual (either PC-relative or BASE-relative)
- We only need to set the n bit to 1 to indicate that the content stored at this location represents the address of the operand, not the operand itself.

70 002A J @RETADR 3E2003

op(6)	n	i	x	b	p	e	disp(12)

$(3C)_{16}$ 1 0 0 0 1 0 $(003)_{16}$

(0011) (1110) (0010) $(003)_{16}$

(3E) (20) (03)

Displacement = 0030 – 002D = 0003

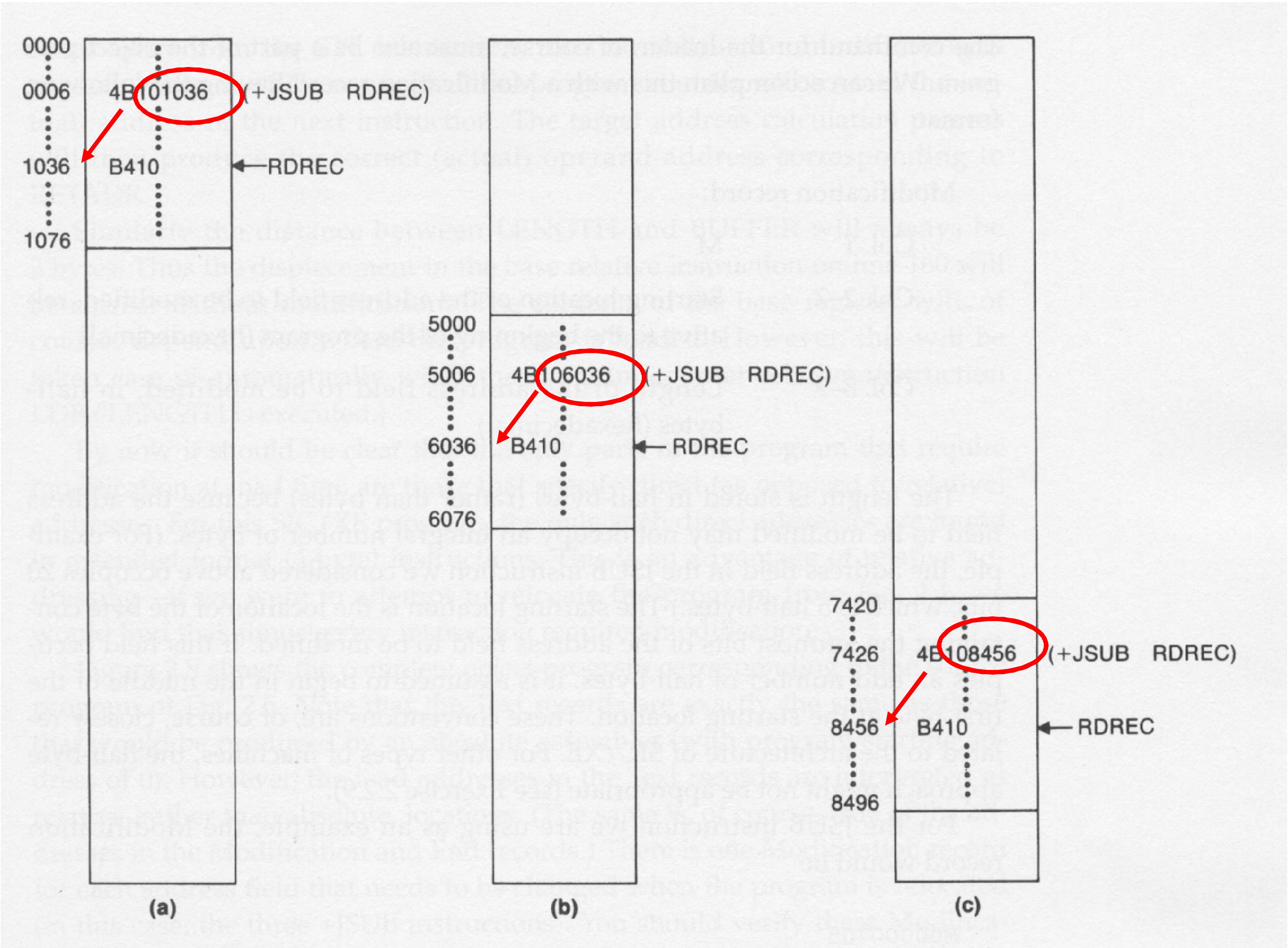
Relocatable Is Desired

- The program in Fig. 2.1 specifies that it must be loaded at address 1000 for correct execution. This restriction is too inflexible for the loader.
- If the program is loaded at a different address, say 2000, its memory references will access wrong data! For example:
  - 55 101B LDA THREE 00102D
- Thus, we want to make programs relocatable so that they can be loaded and execute correctly at any place in the memory.

Program Relocation

- The need for program relocation:
  - It is desirable to load and run several programs at same time.
  - The system must be able to load programs into memory wherever there is room.
  - The exact starting address of program is not known until load time.
- Absolute Program:
  - Program with starting address specified at assembly time.
  - The address may be invalid if the program is loaded into somewhere else.

Address Modification Is Required



What Instructions Needs to be Modified?

- Only those instructions that use absolute (direct) addresses to reference symbols.
- The following need not be modified:
  - Immediate addressing (no memory references)
  - PC or Base-relative addressing (Relocatable is one advantage of relative addressing, among others.)
  - Register-to-register instructions (no memory references)

Relocatable Program

- The object program that contains modification record is called relocatable program.
- To solve relocation program:
  - address is assigned relative to the start of the program (START 0)
  - produce a modification record to store starting location and length of address field to be modified.
  - the command for loader i.e. instructing to add beginning address of program to address field, must be part of object program

The Modification Record

- When the assembler generate an address for a symbol, the address to be inserted into the instruction is relative to the start of the program.
- The assembler also produces a modification record, in which the address and length of the need-to-be-modified address field are stored.
- The loader, when seeing the record, will then add the beginning address of the loaded program to the address field stored in the record.

The Modification Record Format

Example: M00000705 for +JSUB RDREC

Modification record:

Col. 1	M
Col. 2–7	Starting location of the address field to be modified, relative to the beginning of the program (hexadecimal)
Col. 8–9	Length of the address field to be modified, in half-bytes (hexadecimal)

At half-byte boundary. The address field in the format 4 instructions has 20 bits – 5 half-bytes.

If the field contains an odd number of half-bytes, the starting location begins in the middle of the first byte.

The Relocatable Object Code

```
H^COPY  ^000000^001077
T^000000^1D^17202D^69202D^4B101036^032026^290000^332007^4B10105D^3F2FEC^032010
T^00001D^13^0F2016^010003^0F200D^4B10105D^3E2003^454F46
T^001036^1D^B410^B400^B440^75101000^E32019^332FFA^DB2013^A004^332008^57C003^B850
T^001053^1D^3B2FEA^134000^4F0000^F1^B410^774000^E32011^332FFA^53C003^DF2008^B850
T^001070^07^3B2FEF^4F0000^05
M^000007^05
M^000014^05
M^000027^05
E^000000
```