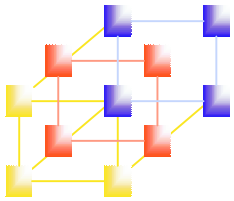
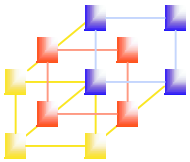


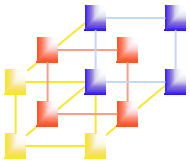
Chapter 2 Assemblers

-- 2.2 Machine-Dependent Assembler Features



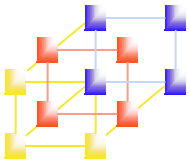
Outline

- Instruction format and addressing mode
- Program relocation



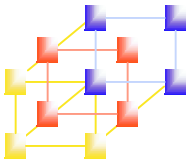
Instruction format and addressing mode

- PC-relative or Base-relative addressing
  - `op  m`
- Indirect addressing
  - `op  @m`
- Immediate addressing
  - `op  #c`
- Extended format
  - `+op  m`
- Index addressing
  - `op  m,x`
- Register-to-register instructions



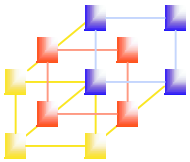
Example SIC/XE program (Figure 2.6, pp.58)

Loc		Source statement		Object code
0000	COPY	START	0	
0000	FIRST	STL	RETADR	17202D
0003		LDB	#LENGTH	69202D
		BASE	LENGTH	
0006	CLOOP	+JSUB	RDREC	4B101036
000A		LDA	LENGTH	032026
000D		COMP	#0	290000
0010		JEQ	ENDFIL	332007
0013		+JSUB	WRREC	4B10105D
0017		J	CLOOP	3F2FEC
001A	ENDFIL	LDA	EOF	032010
001D		STA	BUFFER	0F2016
0020		LDA	#3	010003
0023		STA	LENGTH	0F200D
0026		+JSUB	WRREC	4B10105D
002A		J	@RETADR	3E2003
002D	EOF	BYTE	C'EOF'	454F46
0030	RETADR	RESW	1	
0033	LENGTH	RESW	1	
0036	BUFFER	RESB	4096	



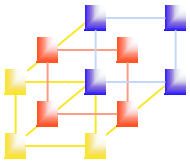
Example SIC/XE program (Figure 2.6, pp.58)

```
                .
                .       SUBROUTINE TO READ RECORD INTO BUFFER
1036    RDREC     CLEAR     X         B410
1038              CLEAR     A         B400
103A              CLEAR     S         B440
103C              +LDY      #4096     75101000
1040    RLOOP     TD        INPUT     E32019
1043              JEQ       RLOOP     332FFA
1046              TD        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
```



Example SIC/XE program (Figure 2.6, pp.58)

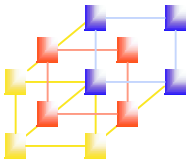
```
                .
                .       SUBROUTINE TO READ RECORD INTO BUFFER

105D        WRREC       CLEAR       X           B410
105F                    LDT         LENGTH      774000
1062        WLOOP       TD          OUTPUT      E32011
1065                    JEQ         WLOOP       332FFA
1068                    LDCH        BUFFER,X    53C003
106B                    WD          OUTPUT      DF2008
106E                    TIXR        T           B850
1070                    JLT         WLOOP       3B2FEF
1073                    RSUB                    4F0000
1076        OUTPUT      BYTE        X'05'       05
                        END         FIRST
```



START statement

```
5       COPY        START       0
```

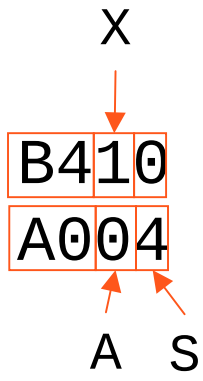
- COPY: program name
- START directive specifies a beginning program address
- 0: a relocatable program
  - Tread as if the program is loaded starting at address 0

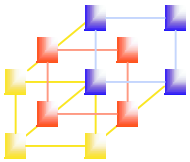


Register-register instruction

- Convert the mnemonic name to their number equivalents
  - Register name (A, X, L, B, S, T, F, PC, SW) and their values (0,1, 2, 3, 4, 5, 6, 8, 9)
  - May implement in a separate table or preload the register names and values to SYMTAB

```
125   CLEAR      X
150   COMPR      A, S
```

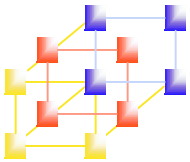




Address translation

- Most register-memory instructions use *program counter relative* or *base relative* addressing
  - Format 3: 12-bit address field
    - `base-relative: 0~4095`
    - `pc-relative: -2048~2047`
  - Format 4: 20-bit address field
- Addressing mode (Refer to Chapter 1)

Mode	Indication	Operand value
Immediate addressing	n=0, i=1,	**TA**
Indirect addressing	n=1, i=0	**((TA))**
Simple addressing	n=0, i=0	Standard SIC
	n=1, i=1	**(TA)**



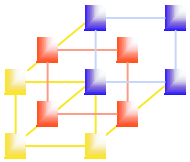
Program counter relative

- Calculate displacement
  - Displacement must be small enough to fit in a 12-bit field (-2048..2047)
  - In SIC, PC is advanced *after each instruction is fetched* and *before it is executed*; *i.e.*, PC contains the address of the next instruction.

```
10    0000      FIRST      STL        RETADR
```

RETADR is at address $(0030)_{16}$
After the SIC fetches this instruction, (PC) = $(0003)_{16}$
TA = (PC) + disp $\Rightarrow$ disp = TA – (PC) = 0030 – 0003 = $(02D)_{16}$

```
   op     n i x b p e      disp
000101   1 1 0 0 1 0      02D    ⇒ 17202D
```



Program counter relative

```
40    0017        J              CLOOP
```

CLOOP is at address $(0006)_{16}$
After the SIC fetches this instruction, (PC) = $(001A)_{16}$
TA = (PC) + disp $\Rightarrow$ disp = TA – (PC) = 0006 – 001A = $(FFEC)_{16}$

```
   op     n i x b p e      disp
001111    1 1 0 0 1 0      FEC     ⇒ 3F2FEC
```

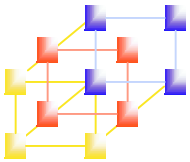
12-bits

```
70    002A        J              @RETADR
```

Indirect addressing

CLOOP is at address $(0030)_{16}$
After the SIC fetches this instruction, (PC) = $(002D)_{16}$
TA = (PC) + disp $\Rightarrow$ disp = TA – (PC) = 0030 – 002D = $(0003)_{16}$

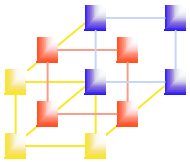
```
   op     n i x b p e      disp
001111    1 0 0 0 1 0      003     ⇒ 3E2003
```



Base relative

- 12 bits displacement (0 ~ 4095)
- Base register is under the control of the programmer.
  - The programmer must tell the assembler what the base register will contain during execution of program.
- Assembler directive
  - **BASE**: tell the assembler what the base register will contain
  - **NOBASE**: tell the assembler that the contents of the base register can no longer be used for addressing.
  - When based register can be relied upon, the assembler can use base relative, otherwise only the PC-relative can be used
  - The assembler first choose PC-relative;
    if displacement is not enough, choose base relative

```
LDB     #LENGTH    (instruction)
BASE    LENGTH     (directive)
        :
NOBASE
```



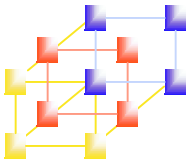
Base relative

```
12  0003          LDB      #LENGTH     69202D
13                BASE     LENGTH
                    :          :
100 0033 LENGTH   RESW     1
105 0036 BUFFER   RESB     4096
                    :          :
160 104E          STCH     BUFFER,X    57C003
165 1051          TIXR     T           B850
```

- PC-relative is no longer applicable
  - $(0036)_{16} - (1051)_{16} = (-1015)_{16} < (-0800)_{16} = (-2048)_{10}$
- LDB loads the address of LENGTH into base register during execution
- BASE directive explicitly informs the assembler that the base register will contain the address of LENGTH

BUFFER is at address $(0036)_{16}$
(B) = $(0033)_{16}$
disp = 0036 – 0033 = $(0003)_{16}$

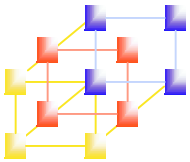
```
  op     n i x b p e      disp
010101   1 1 1 1 0 0      003    ⇒ 57C003
```



Base relative

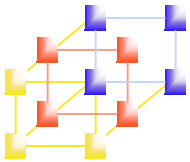
```
20   000A          LDA       LENGTH    032026
                     :           :
175  1056 EXIT     STX       LENGTH    134000
```

- Line 20, using PC-relative
- Consider Line 175
  - If we use PC-relative
    - `LENGTH at address 0033`
    - `Disp = TA – (PC) = 0033 –1059 = EFDA`
    - `PC relative is no longer applicable, try to use BASE relative addressing`



Choice of Addressing Modes

1. Programmer must specify the extended format (4-byte) by using the prefix +
2. If not, assembler first attempts PC-relative
3. If the required displacement is out of range, use base relative addressing can be use
4. Otherwise, generate an error message



Immediate addressing

- No memory reference is involved
- If immediate mode is specified, the target address becomes the operand

```
55    0020        LDA         #3
```

Immediate operand

$TA = (0003)_{16}$

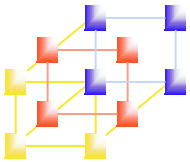
op	n	i	x	b	p	e	disp
000000	0	1	0	0	0	0	003 ⇒ 010003

```
133   103C        +LDT        #4096
```

Extended instruction format

$TA = (01000)_{16}$

op	n	i	x	b	p	e	disp(20 bits)
011101	0	1	0	0	0	1	01000 ⇒ 75101000



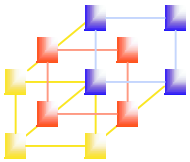
Immediate & PC-relative addressing

```
12    0003        LDB         #LENGTH
```

LENGTH is at address 0033

TA = (PC) + disp $\Rightarrow$ disp = 0033 – 0006 = $(002D)_{16}$

```
   op     n i x b p e      disp
011010   0 1 0 0 1 0      02D    ⇒ 69202D
```



Indirect & PC-relative addressing

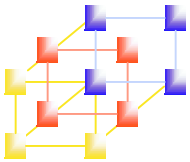
```
70    002A              J          @RETADR
                        :            :
95    0030 RETADR       RESW       1
```

RETADR is at address 0030

TA = (PC) + disp $\Rightarrow$ disp = 0030 – 002D = $(0003)_{16}$

op	n	i	x	b	p	e	disp
001111	1	0	0	0	1	0	003

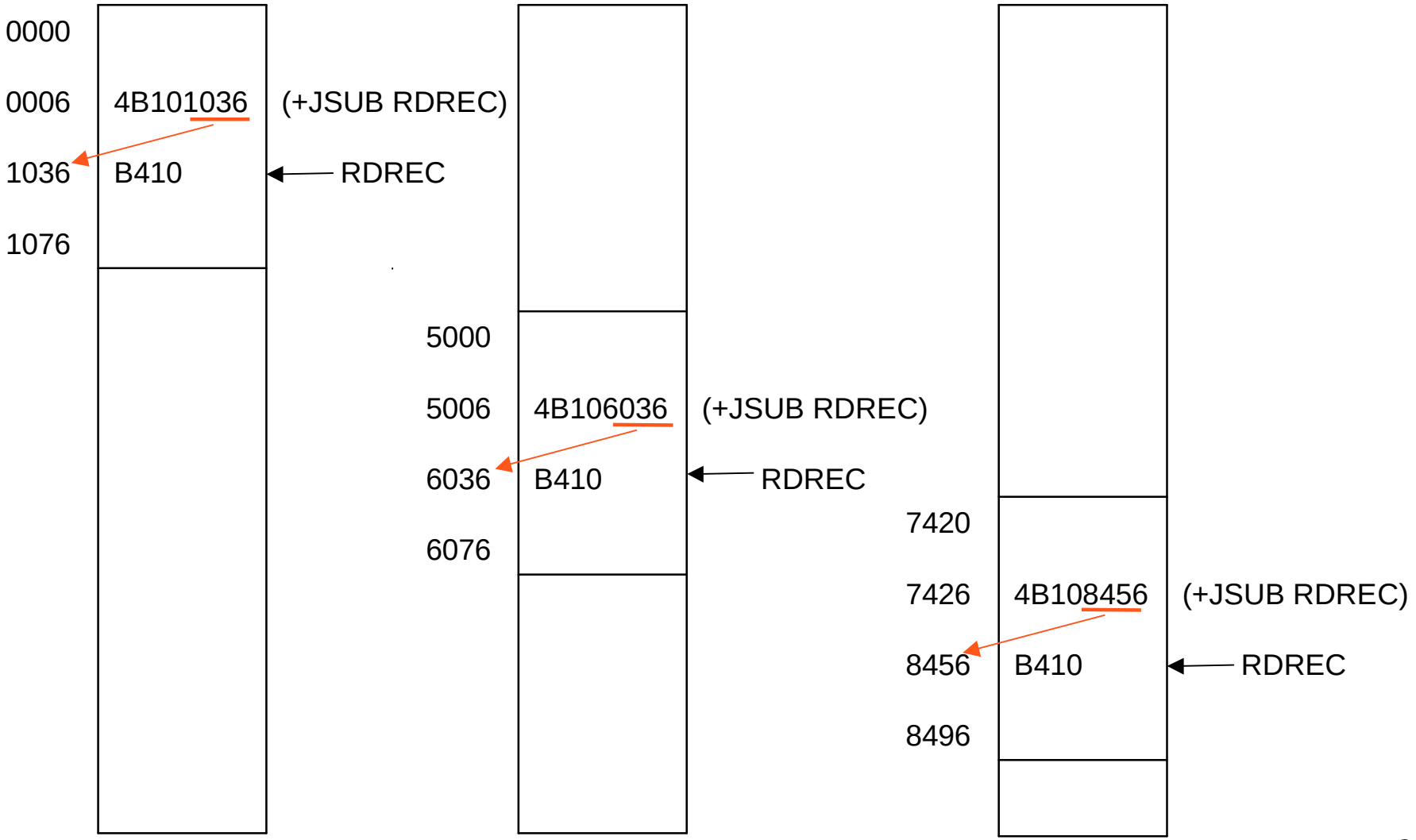
$\Rightarrow$ 3E2003

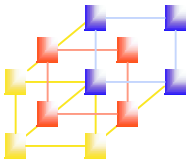


Program relocation

- Why
  - It is desirable to load and run several programs at the same time
  - The system must be able to load programs into memory wherever there is room
  - The exact starting address of the program is not known until load time

Example of program relocation (Figure 2.7, pp.63)





Program relocation

- Absolute Program
  - Program with starting address specified at assembly time
  - The address may be invalid if the program is loaded into somewhere else.
  - Example: (Figure 2.2, pp.47)

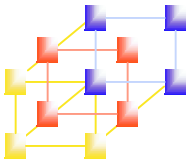
```
55      101B            LDA     THREE           00102D
```

Calculate based on the starting address 1000

Reload the program starting at 3000

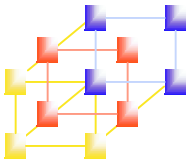
```
55      101B            LDA     THREE           00302D
```

The absolute address should be modified



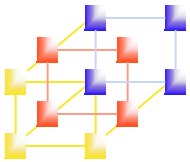
Program relocation

- The only parts of the program that require modification at load time are those that specify direct addresses
- The rest of the instructions need not be modified
  - Not a memory address (immediate addressing)
  - PC-relative, Base-relative
- From the object program, it is not possible to distinguish the address and constant
  - The assembler must keep some information to tell the loader
  - The object program that contains the modification record is called a relocatable program



The way to solve the relocation problem

- For an address label, its address is assigned relative to the start of the program (START 0)
- Produce a Modification record to store the starting location and the length of the address field to be modified.
- The command for the loader must also be a part of the object program



Modification record

Modification record

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

- One modification record for each address to be modified
- The length is stored in half-bytes (4 bits)
- The starting location is the location of the byte containing the leftmost bits of the address field to be modified.
- If the field contains an odd number of half-bytes, the starting location begins in the middle of the first byte.

Relocatable Object Program (Figure 2.8, pp.65)