系統程式

SYSTEM PROGRAMMING

Speaker: Shu-Yu Kuo

Department of Computer Science

National Chung Hsing University

SYSTEM SOFTWARE: AN INTRODUCTION TO SYSTEMS PROGRAMMING

LELAND .L. BECK

Chapter 2 Assemblers

OUTLINE

- 2.1 Basic Assembler Functions
- **2.2 Machine-Dependent Assembler Features**
  - Instruction Formats and Addressing Modes
  - Program Relocation
- 2.3 Machine-Independent Assembler Features
- 2.4 Assembler Design Options
- 2.5 Implementation Examples

2.2 MACHINE DEPENDENT ASSEMBLER FEATURES

- Machine Dependent Assembler Features
  - Example: SIC/XE supports
    - More instruction formats and addressing modes
    - Program relocation

SIC/XE ASSEMBLER

- Previous, we know how to implement the 2-pass SIC assembler.

- What's new for SIC/XE?
  - More addressing modes and instruction formats.
  - Program Relocation.

SIC/XE ASSEMBLER (CONT.)

- SIC/XE
  - Immediate addressing: op #c
  - Indirect addressing: op @m
  - **PC-relative** or **Base-relative** addressing: op m
    - The assembler directive **BASE** is used with base-relative addressing
    - If displacements are too large to fit into a 3-byte instruction, then 4-byte extended format is used
  - Extended format: +op m
  - Indexed addressing: op m, **x**
  - Register-to-register instructions
  - Large memory
    - Support multiprogramming and need *program reallocation* capability

EXAMPLE OF A SIC/XE PROGRAM (FIG 2.5)

- Improve the execution speed of Fig. 2.2 (SIC version)
  - Register-to-register instructions
  - Immediate addressing: op #c
    - Operand is already present as part of the instruction
  - Indirect addressing: op @m
    - Often avoid the need of another instruction

EXAMPLE OF A SIC/XE PROGRAM (FIG 2.5,2.6)

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	

EXAMPLE OF A SIC/XE PROGRAM (FIG 2.5,2.6) (CONT.)

```
110                 .
115                 .          SUBROUTINE TO READ RECORD INTO BUFFER
120                 .
125     1036        RDREC      CLEAR     X                B410
130     1038                   CLEAR     A                B400
132     103A                   CLEAR     S                B440
133     103C                  +LDT      #4096             75101000
135     1040        RLOOP      TD        INPUT            E32019
140     1043                   JEQ       RLOOP            332FFA
145     1046                   RD        INPUT            DB2013
150     1049                   COMPR     A,S              A004
155     104B                   JEQ       EXIT             332008
160     104E                   STCH      BUFFER,X         57C003
165     1051                   TIXR      T                B850
170     1053                   JLT       RLOOP            3B2FEA
175     1056        EXIT       STX       LENGTH           134000
180     1059                   RSUB                       4F0000
185     105C        INPUT      BYTE      X'F1'            F1
```

EXAMPLE OF A SIC/XE PROGRAM (FIG 2.5,2.6) (CONT.)

```
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
```

Figure 2.6 Program from Fig. 2.5 with object code.

OUTLINE

- 2.1 Basic Assembler Functions
- **2.2 Machine-Dependent Assembler Features**
  - **Instruction Formats and Addressing Modes**
  - Program Relocation
- 2.3 Machine-Independent Assembler Features
- 2.4 Assembler Design Options
- 2.5 Implementation Examples

2.2.1 INSTRUCTION FORMATS AND ADDRESSING MODES

- START now specifies a beginning program address of 0
  - Indicate a *relocatable program*
- Register translation
  - For example: *COMPR A, S => A004*
  - Must keep the register name (A, X, L, B, S, T, F, PC, SW) and their values (0, 1, 2, 3, 4, 5, 6, 8, 9)
    - Keep in **SYMTAB**

ADDRESS TRANSLATION

- Most register-to-memory instructions are assembled using *PC relative* or *base relative* addressing
  - Assembler must calculate a *displacement* as part of the object instruction
  - If displacement can be fit into 12-bit field, format 3 is used.
  - Format 3: 12-bit address field
    - **Base-relative: 0~4095**
    - **PC-relative: -2048~2047**
  - *Assembler attempts to* ***translate using PC-relative first, then base-relative***
    - If displacement in PC-relative is out of range, then try base-relative

ADDRESS TRANSLATION (CONT.)

- If displacement can not be fit into 12-bit field in the object instruction, format 4 must be used.
  - Format 4: 20-bit address field
  - No displacement need to be calculated.
    - 20-bit is large enough to contain the full memory address
  - Programmer must specify extended format: +op m
  - For example: *+JSUB RDREC => 4B101036*
    - *LOC(RDREC) = 1036, get it from SYMTAB*

PC-RELATIVE ADDRESSING MODES

- *10 0000 FIRST STL RETADR 17202D*
  - Displacement= RETADR – (PC) = 30-**3** = 2D
  - opcode (6 bits) $=14_{16}=00010100_2$
  - nixbpe=110010
    - n=1, i = 1: indicate neither *indirect* nor *immediate* addressing
    - p = 1: indicate *PC-relative* addressing

OPCODE	n	i	x	b	p	e	Displacement
0001 01	1	1	0	0	1	0	$(02D)_{16}$

Object Code = 17202D

PC-RELATIVE ADDRESSING MODES (CONT.)

- 40 0017 J CLOOP 3F2FEC
  - Displacement= CLOOP - (PC) = 6 - **1A** = -14 = FEC (2's complement for negative number)
  - opcode=$3C_{16}$ = 00111100_2
  - nixbpe=110010

OPCODE	n	i	x	b	p	e	Displacement
0011 11	1	1	0	0	1	0	$(FEC)_{16}$

Object Code = 3F2FEC

BASE-RELATIVE ADDRESSING MODES

- Base register is under the control of the programmer
  - Programmer use **assembler directive *BASE*** to specify which value to be assigned to base register (B)
  - **Assembler directive *NOBASE***: inform the assembler that the contents of base register no longer be used for addressing
  - ***BASE*** and ***NOBASE*** produce no executable code

BASE-RELATIVE ADDRESSING MODES

- 175 1056 STX LENGTH 134000
  - Try PC-relative first
    - Displacement= LENGTH - (PC) = 0033 - 1059 = -1026 (hex)
  - Try base-relative next
    - displacement= LENGTH – (B) = 0033 – 0033 =0
    - Opcode=10_{16} = $(00010000)_2$
    - nixbpe=110100
      - n=1, i = 1: indicate neither *indirect* nor *immediate* addressing
      - b = 1: *base-relative* addressing

OPCODE	n	i	x	b	p	e	Displacement
000100	1	1	0	1	0	0	$(000)_{16}$

BASE-RELATIVE ADDRESSING MODES (CONT.)

- *160 104E STCH BUFFER, X 57C003*
- The displacement of PC-relative is out of range
- Displacement= BUFFER – (B) = 0036 – 0033(=LOC(LENGTH)) = 3
- opcode=54
- nixbpe=111100
  - n=1, i = 1: indicate neither *indirect* nor *immediate* addressing
  - x = 1: *indexed* addressing
  - b = 1: *base-relative* addressing

OPCODE	n	i	x	b	p	e	Displacement
0101 01	1	1	1	1	0	0	$(003)_{16}$

Object Code = 57C003

IMMEDIATE ADDRESS TRANSLATION

- Convert the *immediate* operand to its internal representation and insert it into the instruction
- 55 0020 LDA #3 010003
  - opcode=00
  - nixbpe=010000
    - i = 1: *immediate addressing*

OPCODE	n	i	x	b	p	e	Displacement
0000 00	0	1	0	0	0	0	$(003)_{16}$

Object Code = 010003

IMMEDIATE ADDRESS TRANSLATION (CONT.)

- 133 103C +LDT #4096 75101000
  - opcode=74
  - nixbpe=010001
    - i = 1: *immediate addressing*
    - e = 1: *extended instruction format* since 4096 is too large to fit into the 12-bit displacement field

OPCODE	n	i	x	b	p	e	Address
0111 01	0	1	0	0	0	1	$(01000)_{16}$

Object Code = 75101000

IMMEDIATE ADDRESS TRANSLATION (CONT.)

- 12 0003 LDB #LENGTH 69202D
  - The immediate operand is the symbol LENGTH
    - The address of LENGTH is loaded into register B
  - Displacement=LENGTH – (PC) = 0033 – 0006 = 02D
  - opcode=68_{16} = 01101000_2
  - nixbpe=010010
    - Combined *PC relative* (p=1) with *immediate addressing* (i=1)

OPCODE	n	i	x	b	p	e	Displacement
0110 10	0	1	0	0	1	0	$(02D)_{16}$

INDIRECT ADDRESS TRANSLATION

- Indirect addressing
  - The contents stored at the location represent the *address* of the operand, not the operand itself
  - Target addressing is computed as usual (PC-relative or BASE-relative)
  - *n* bit is set to 1

INDIRECT ADDRESS TRANSLATION (CONT.)

- 70 002A J @RETADR 3E2003
  - Displacement= RETADR- (PC) = 0030 – 002D =3
  - opcode= 3C
  - nixbpe=100010
    - n = 1: *indirect addressing*
    - p = 1: *PC-relative addressing*

OPCODE	n	i	x	b	p	e	Displacement
0011 11	1	0	0	0	1	0	$(003)_{16}$

OUTLINE

- 2.1 Basic Assembler Functions
- **2.2 Machine-Dependent Assembler Features**
  - Instruction Formats and Addressing Modes
  - **Program Relocation**
- 2.3 Machine-Independent Assembler Features
- 2.4 Assembler Design Options
- 2.5 Implementation Examples

2.2.2 PROGRAM RELOCATION

- ***Program relocation***

  - Load programs into memory wherever there is room

  - Not specifying a fixed address at assembly time

2.2.2 PROGRAM RELOCATION (CONT.)

- *Absolute program* (or *absolute assembly*)
  - Program must be loaded at the address specified *at assembly time.*
  - E.g. Fig. 2.1

```
COPY   START   1000
FIRST  STL     RETADR
                 :
                 :
```

program loading starting address 1000

  - e.g. 55 101B LDA THREE <u>00102D</u>

- What if the program is loaded to 2000

  e.g. 55 101B LDA THREE <u>00202D</u>

  - Each **absolute address** should be modified

EXAMPLE OF PROGRAM RELOCATION (FIG 2.7)

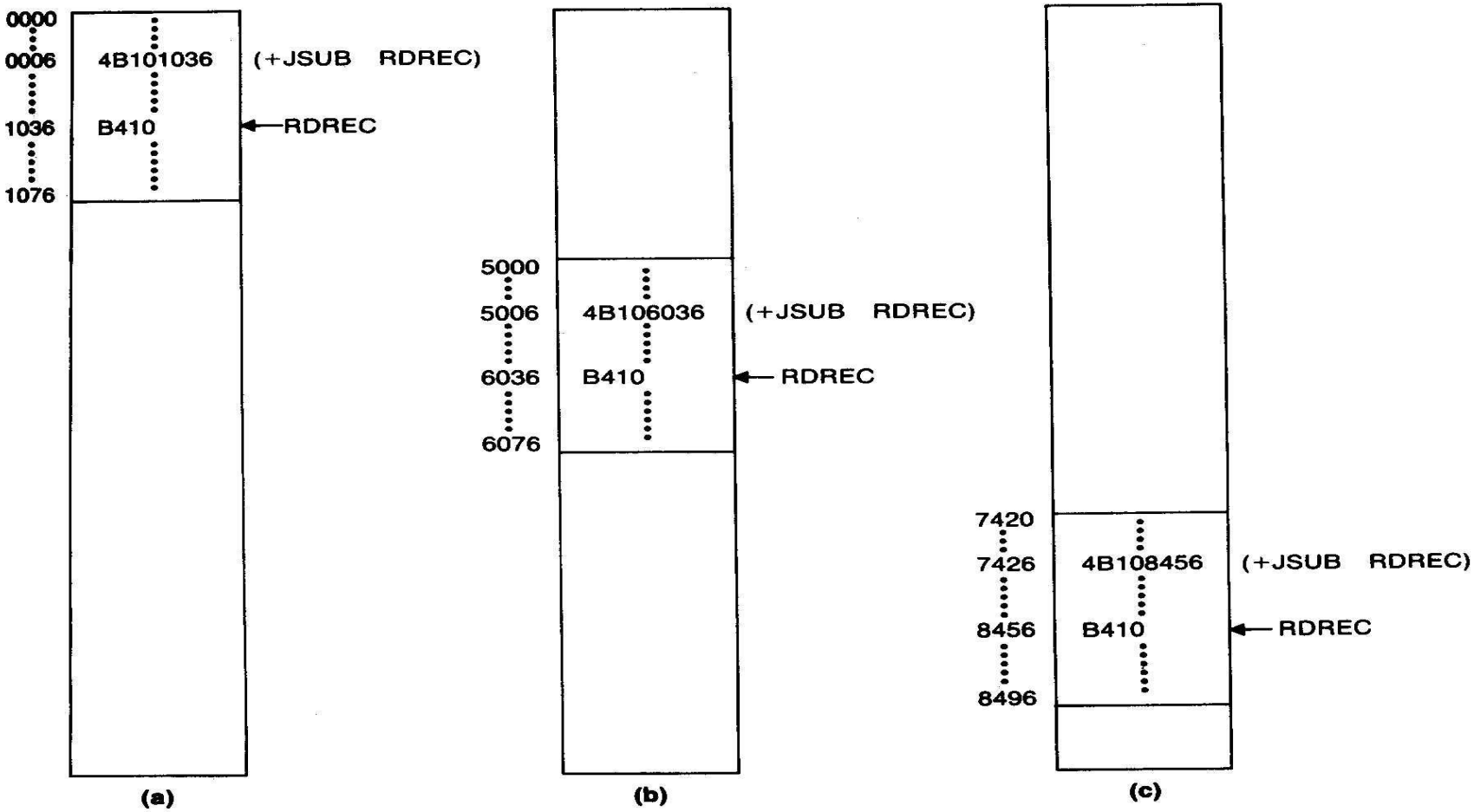

Figure 2.7 Examples of program relocation.

2.2.2 PROGRAM RELOCATION (CONT.)

- *Relocatable* program

```
COPY   START   0
FIRST  STL     RETADR
              :
              :
```

program loading starting address is determined *at load time*

- **No instruction modification is needed for**
  - **Immediate addressing (not a memory address)**
  - **PC-relative, Base-relative addressing**
- The only parts that require modification at load time are those that specify ***direct addresses***
  - In SIC/XE, only found in ***extended format instructions***
- The **assembler** must identify for the **loader** *those parts of object program that need modification*.
  - The object program must contain the information necessary to perform *address modification*

INSTRUCTION FORMAT VS. RELOCATABLE LOADER

- In SIC/XE
  - Format 1, 2, 3
    - Not affect
  - Format 4
    - Should be modified
- In SIC
  - Format 3 with address field
    - Should be modified
    - SIC does not support PC-relative and base-relative addressing

RELOCATABLE PROGRAM

- We use modification records that are added to the object files.

Pass the *address–modification* information to the relocatable loader

- *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
  - E.g M^000007^05

Beginning address of the program is to be added to a field that begins at addr ox000007 and is 2.5 bytes in length.

OBJECT PROGRAM FOR FIG 2.6 (FIG 2.8)

```
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
```

In fact, the starting address should be 7.5. However, the assembler avoids using float point.

Figure 2.8 Object program corresponding to Fig. 2.6.