SWE4001 – System Programming

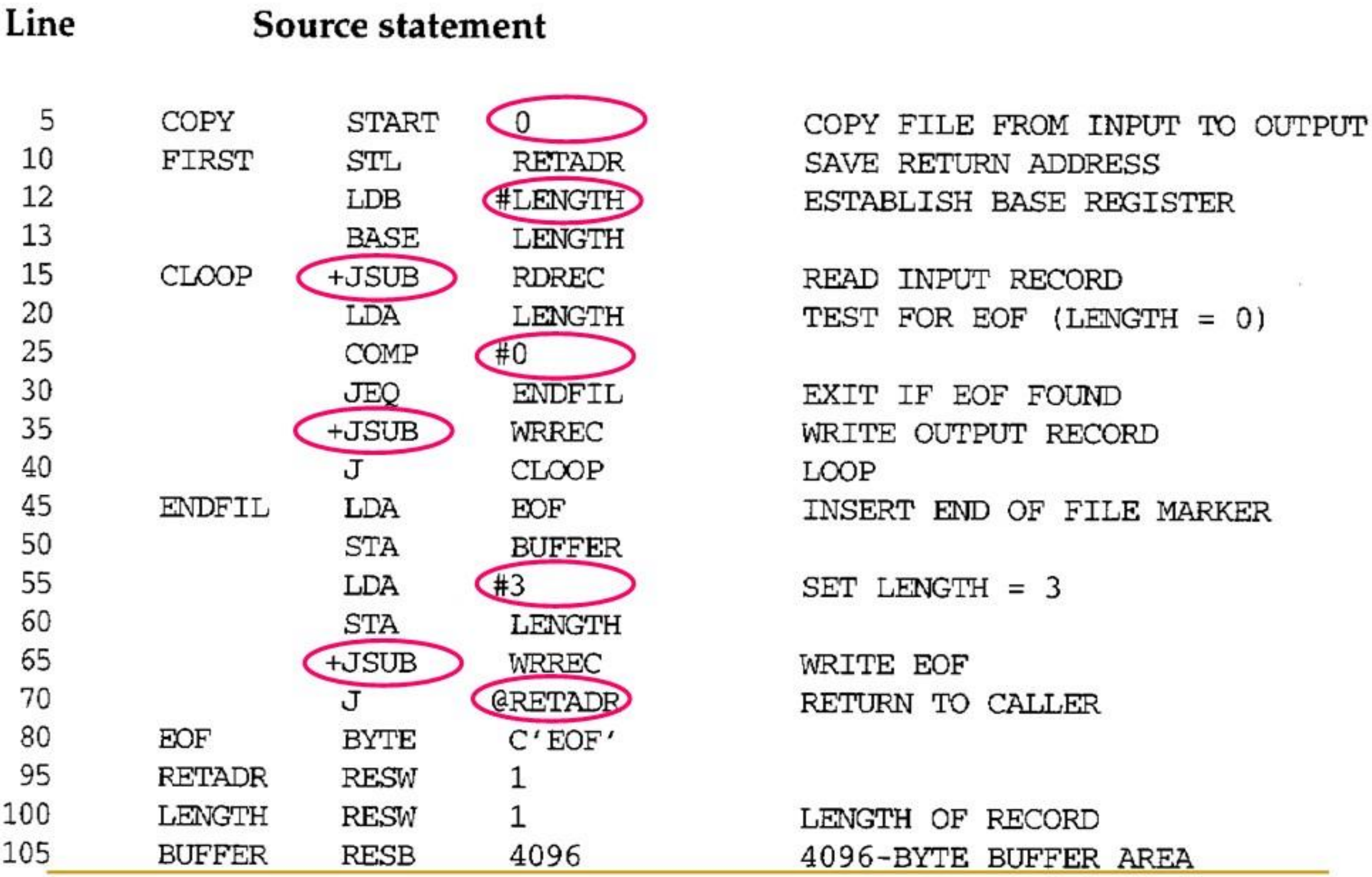
Module 3: Assembler

Lesson 5 of 7: Machine dependent features of Assembler

2.2 Machine-Dependent Assembler Features

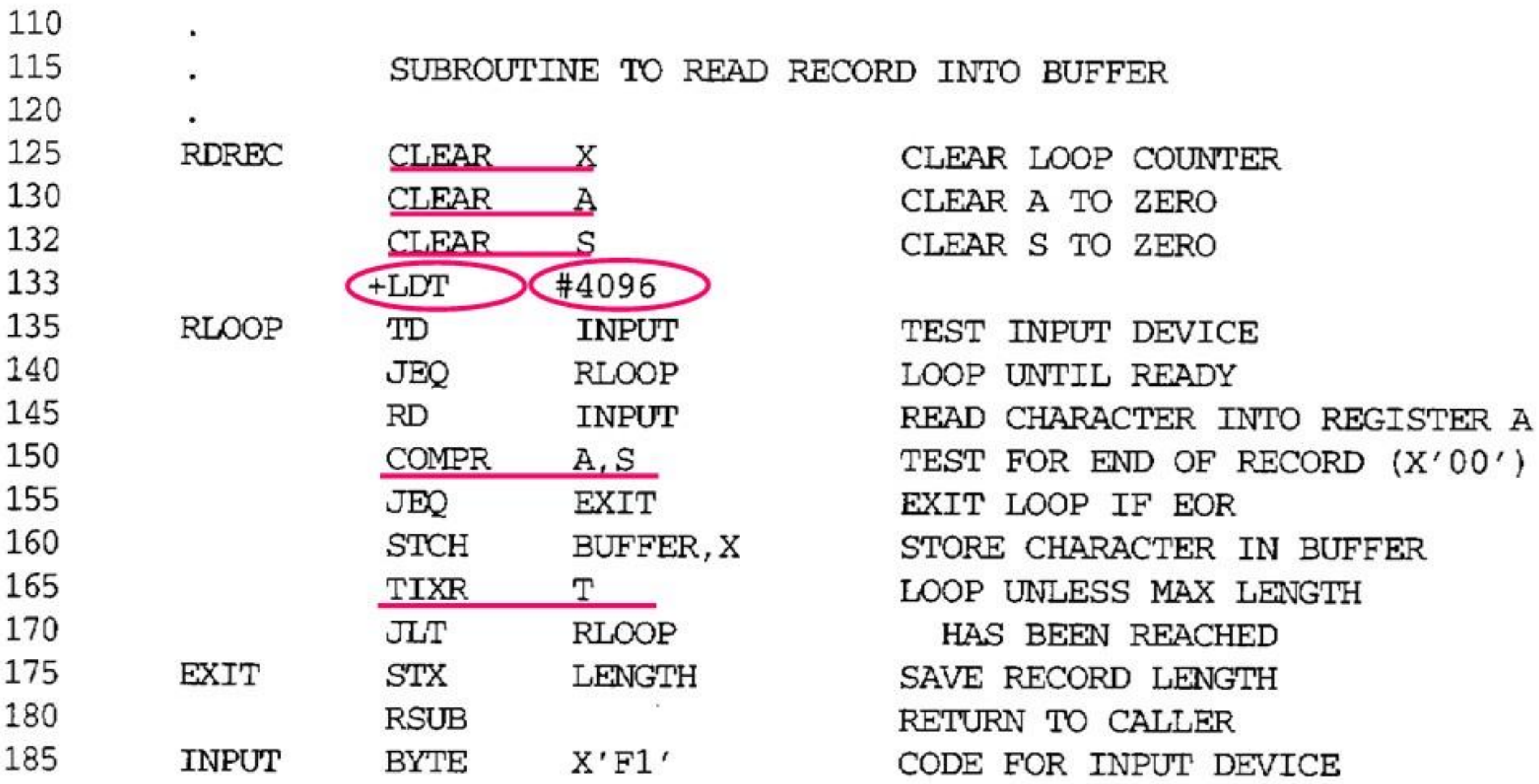
- ④ Indirect addressing
  - ⑨ Adding the prefix @ to operand (line 70).
  - ④ Immediate operands
    - ⑨ Adding the prefix # to operand (lines 12, 25, 55, 133).
- ④ Base relative addressing
  - ⑨ Assembler directive BASE (lines 12 and 13).
  - ④ Extended format
    - ⑨ Adding the prefix + to OP code (lines 15, 35, 65).
- ④ The use of register-register instructions.
    - ⑨ Faster and don’t require another memory reference.

Figure 2.5: First



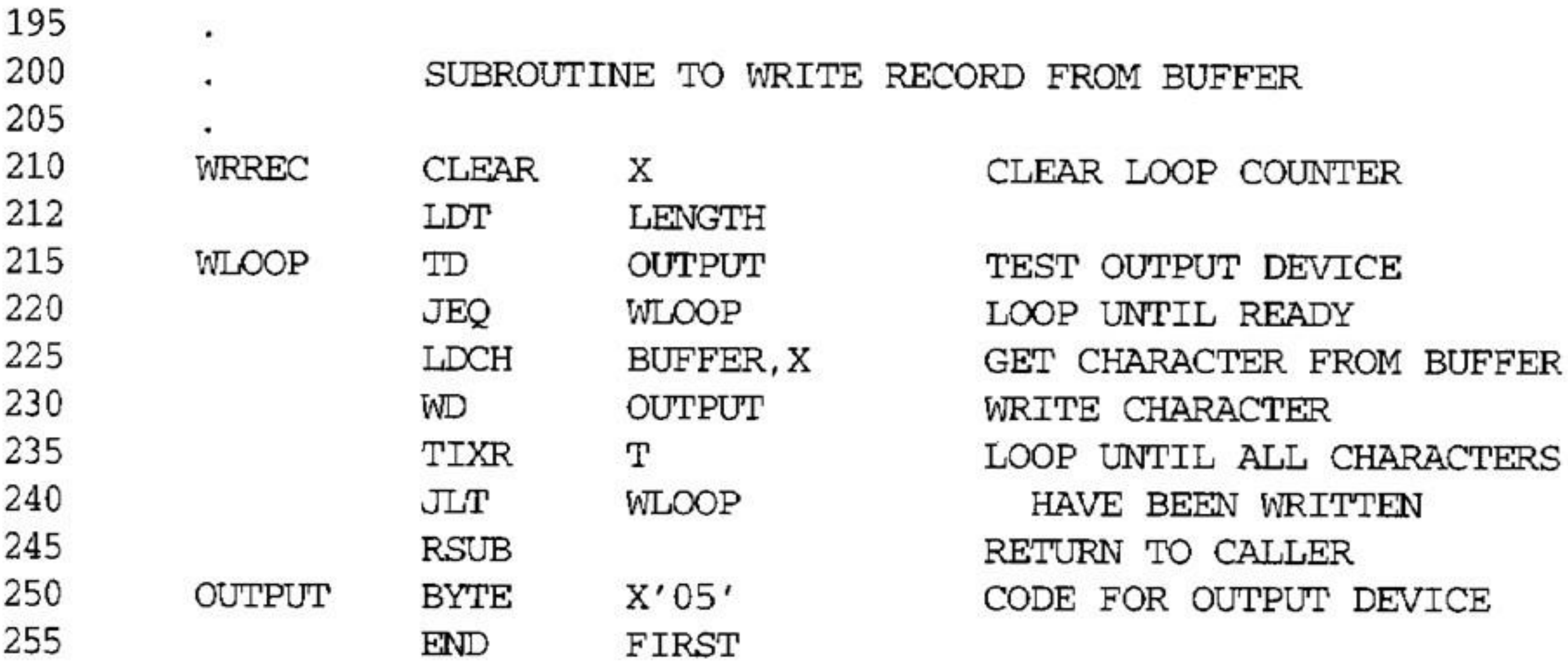
```
Line           Source statement

  5     COPY      START    0            COPY FILE FROM INPUT TO OUTPUT
 10     FIRST     STL      RETADR       SAVE RETURN ADDRESS
 12               LDB      #LENGTH      ESTABLISH BASE REGISTER
 13               BASE     LENGTH
 15     CLOOP    +JSUB     RDREC        READ INPUT RECORD
 20               LDA      LENGTH       TEST FOR EOF (LENGTH = 0)
 25               COMP     #0
 30               JEQ      ENDFIL       EXIT IF EOF FOUND
 35              +JSUB     WRREC        WRITE OUTPUT RECORD
 40               J        CLOOP        LOOP
 45     ENDFIL    LDA      EOF          INSERT END OF FILE MARKER
 50               STA      BUFFER
 55               LDA      #3           SET LENGTH = 3
 60               STA      LENGTH
 65              +JSUB     WRREC        WRITE EOF
 70               J        @RETADR      RETURN TO CALLER
 80     EOF       BYTE     C'EOF'
 95     RETADR    RESW     1
100     LENGTH    RESW     1            LENGTH OF RECORD
105     BUFFER    RESB     4096         4096-BYTE BUFFER AREA
```

Figure 2.5: RDREC



```
.
.        SUBROUTINE TO READ RECORD INTO BUFFER
.
RDREC    CLEAR    X                 CLEAR LOOP COUNTER
         CLEAR    A                 CLEAR A TO ZERO
         CLEAR    S                 CLEAR S TO ZERO
         +LDT     #4096
RLOOP    TD       INPUT             TEST INPUT DEVICE
         JEQ      RLOOP             LOOP UNTIL READY
         RD       INPUT             READ CHARACTER INTO REGISTER A
         COMPR    A,S               TEST FOR END OF RECORD (X'00')
         JEQ      EXIT              EXIT LOOP IF EOR
         STCH     BUFFER,X          STORE CHARACTER IN BUFFER
         TIXR     T                 LOOP UNLESS MAX LENGTH
         JLT      RLOOP               HAS BEEN REACHED
EXIT     STX      LENGTH            SAVE RECORD LENGTH
         RSUB                       RETURN TO CALLER
INPUT    BYTE     X'F1'             CODE FOR INPUT DEVICE
```

Figure 2.5: WRREC



```
195      .
200      .          SUBROUTINE TO WRITE RECORD FROM BUFFER
205      .
210      WRREC      CLEAR     X                CLEAR LOOP COUNTER
212                 LDT       LENGTH
215      WLOOP      TD        OUTPUT           TEST OUTPUT DEVICE
220                 JEQ       WLOOP            LOOP UNTIL READY
225                 LDCH      BUFFER,X         GET CHARACTER FROM BUFFER
230                 WD        OUTPUT           WRITE CHARACTER
235                 TIXR      T                LOOP UNTIL ALL CHARACTERS
240                 JLT       WLOOP              HAVE BEEN WRITTEN
245                 RSUB                       RETURN TO CALLER
250      OUTPUT     BYTE      X'05'            CODE FOR OUTPUT DEVICE
255                 END       FIRST
```

Figure 2.5 Example of a SIC/XE program.

2.2 Machine-Dependent Assembler Features

- SIC/XE

PC-relative/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	COMPR

- larger memory → multi-programming (program allocation)

2.2 Machine-Dependent Assembler Features

- Register translation
  - register name (A, X, L, B, S, T, F, PC, SW) and their values (0, 1, 2, 3, 4, 5, 6, 8, 9)
  - preloaded in SYMTAB
- Address translation
  - Most register-memory instructions use program counter relative or base relative addressing
  - Format 3: 12-bit disp (address) field
    - PC-relative: -2048~2047
    - Base-relative: 0~4095
  - Format 4: 20-bit address field (absolute addressing)

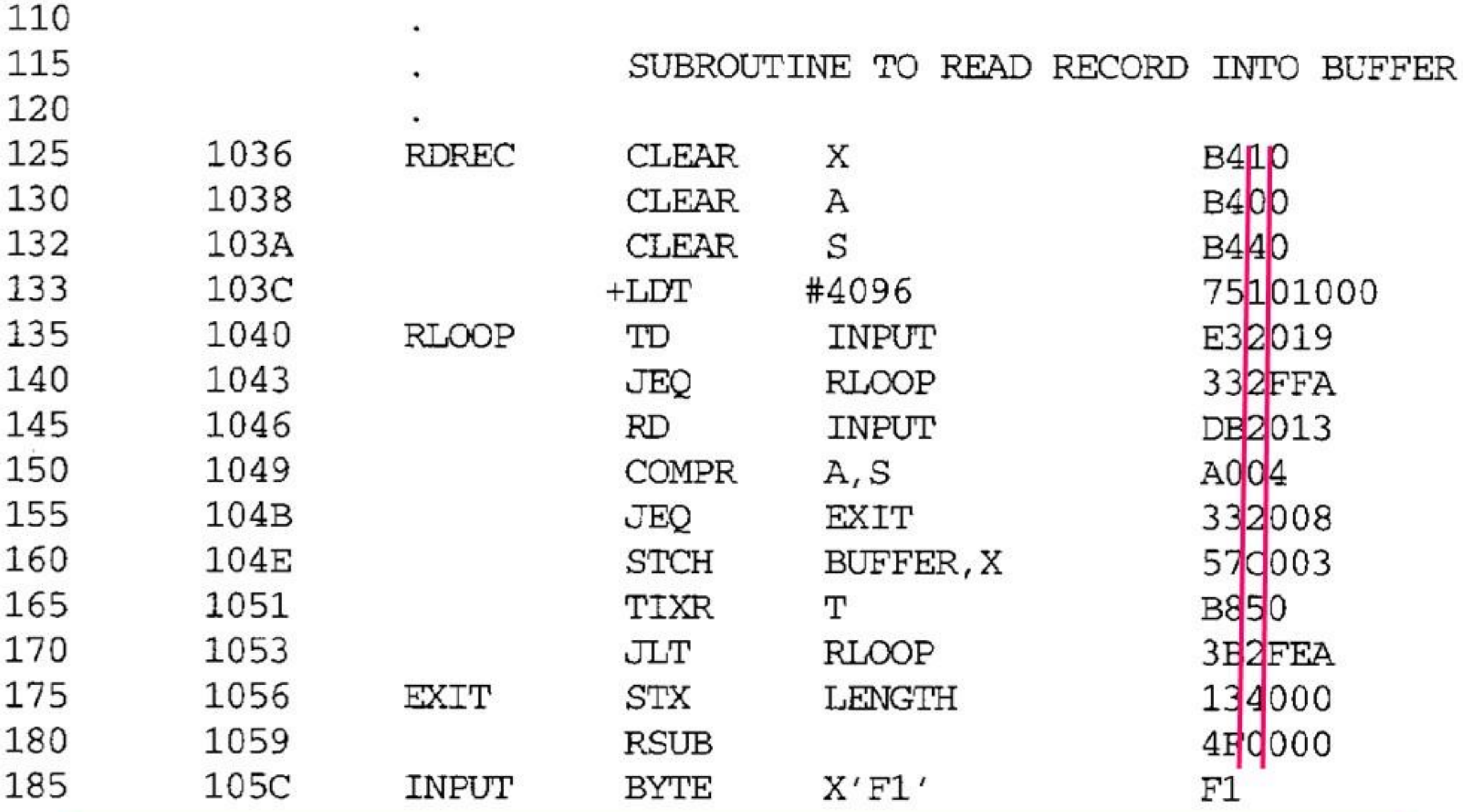
2.2.1 Instruction Formats & Addressing Modes

- The START statement
  - Specifies a beginning address of 0.
  - Register-register instructions
    - CLEAR & TIXR, COMPR
- Register-memory instructions are using
  - Program-counter (PC) relative addressing
  - The program counter is advanced *after* each instruction is fetched and *before* it is executed.
  - PC will contain the address of the *next* instruction.

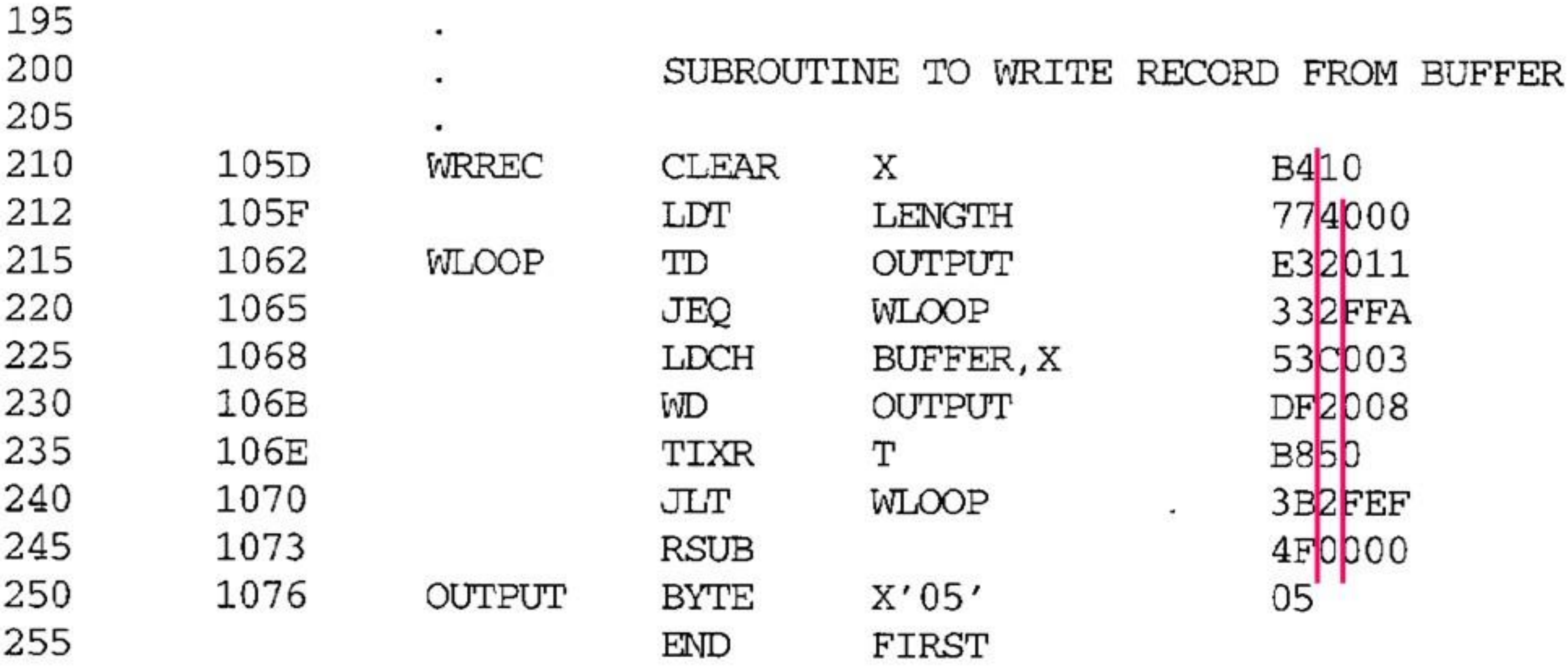
```
10    0000        FIRST  STL    RETADR        17202D
```

TA - (PC) = disp = 30H - 3H= 2D

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	



```
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
```



```
                .       SUBROUTINE TO WRITE RECORD FROM 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
```

Figure 2.6 Program from Fig. 2.5 with object code.

PC-Relative Addressing Mode

10 0000 FIRST STL RETADR 17202D

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

– Displacement= RETADR–PC = 0030–0003 = 02D

40 0017 J CLOOP 3F2FEC

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

– Displacement= CLOOP–PC= 0006–001A= –14= FEC

Base-Relative Addressing Mode

- BASE register and directive:

```
12                LDB    #LENGTH
13                BASE LENGTH
```

  - Base register is under the control of programmer
  - BASE directive tells assembler that LENGHTH is base address; NOBASE releases the binding

```
160   104E        STCH BUFFER, X           57C003
```

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

– Displacement = BUFFER – B = 0036 – 0033 = 3

Immediate Address Translation

- Immediate addressing

55 0020 LDA #3 010003

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

133 103C +LDT #4096 75101000

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

Immediate Address Translation

12 0003 LDB #LENGTH 69202D

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

12 0003 LDB #LENGTH 690033

OPCODE	n	i	x	b	p	e	Address
0110 10	0	1	0	0	0	0	$(033)_{16}$

- The immediate operand is the value of the symbol LENGTH, which is the address assigned to LENGTH
- LENGTH = 0033 = PC+ displacement =0006 + 02D

Indirect Address Translation

- Indirect addressing
  - Target addressing is computed as usual (PC-relative or BASE-relative)
  - Only the n bit is set to 1

```
70        002A          J        @RETADR   3E2003
```

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

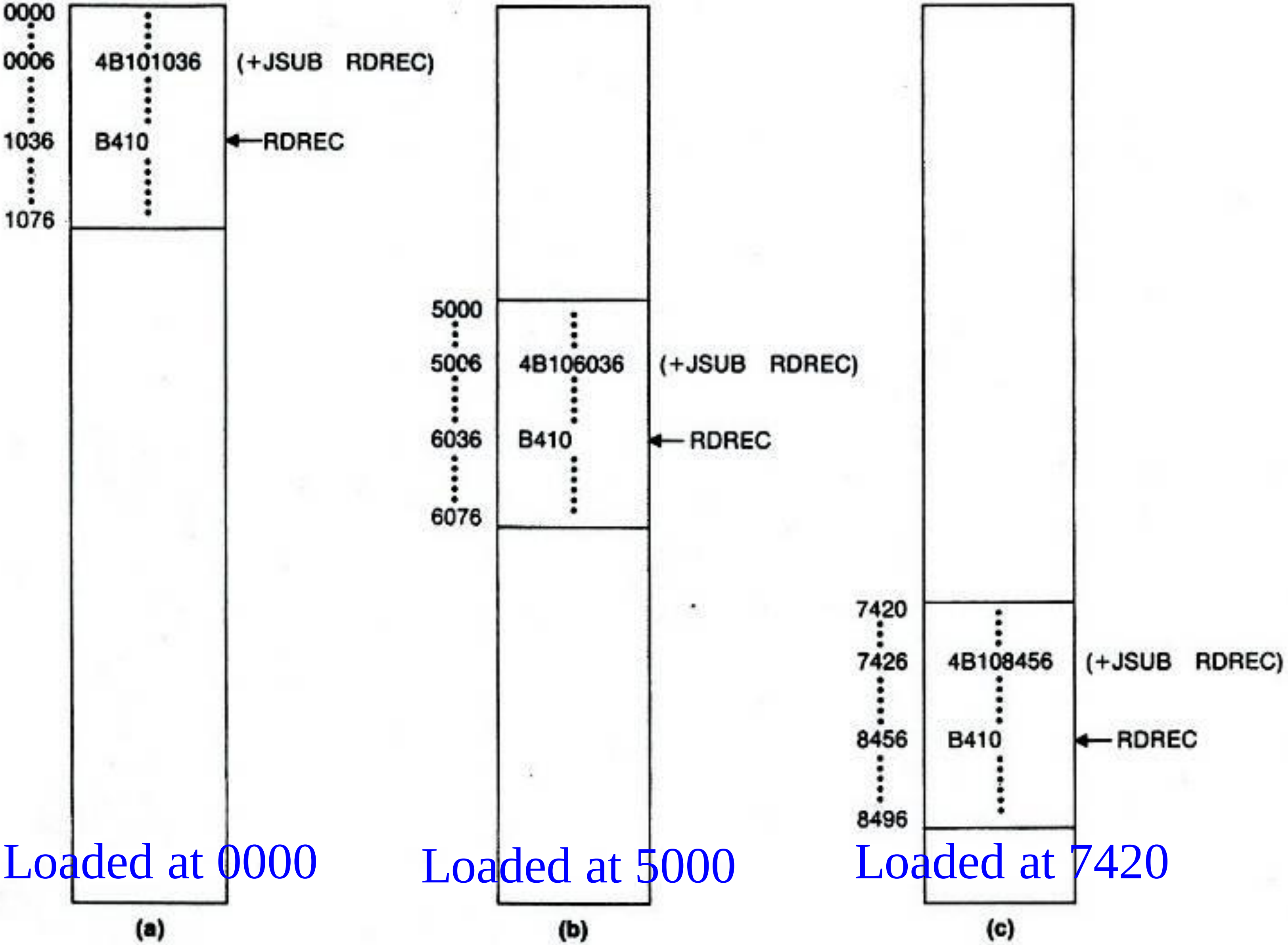
– TA=RETADR=0030

– TA=(PC) + displacement = 002D + 0003

+OP, e=1		Extended
n=1, i=1,	OPcode+3,	Simple
@m, n=1, i=0,	OPcode+2,	Indirect
#C, n=0, i=1,	OPcode+1,	Immediate

xbpe	2: PC-relative
	4: base-relative
	8: index (m,X)
	1: extended

2.2.2 Program Relocation



Loaded at 0000

Loaded at 5000

Loaded at 7420

Example of Program Relocation (1/2)

- Example Fig. 2.2
  - *Absolute program*, starting address ~~1000~~ → 2000

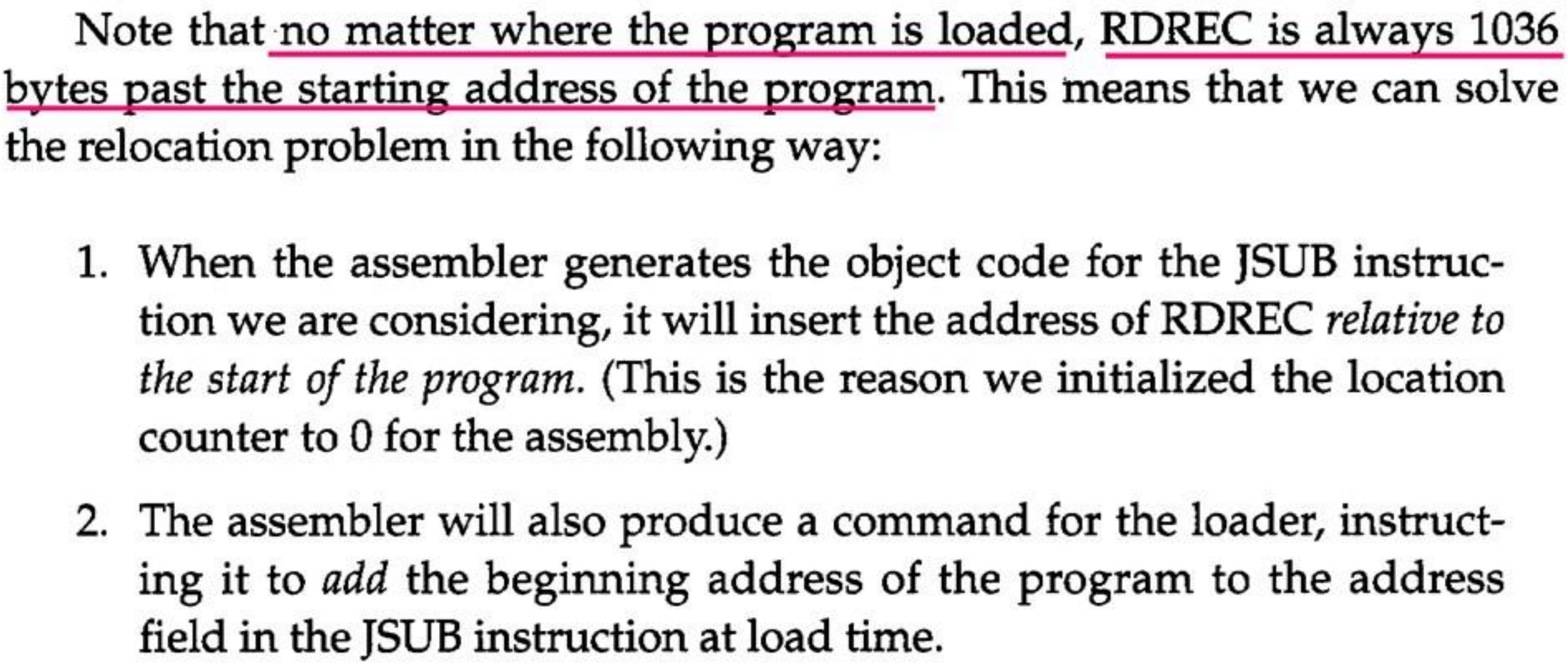
```
5      1000   COPY     START   ~~1000~~ → 2000
10     1000   FIRST    STL     RETADR             141033
15     1003   CLOOP    JSUB    RDREC              482039
20     1006            LDA     LENGTH             001036
25     1009            COMP    ZERO               281030
30     100C            JEQ     ENDFIL             301015
35     100F            JSUB    WREC               482061
40     1012            J       CLOOP              3C1003
45     1015   ENDFIL   LDA     EOF                00102A
50     1018            STA     BUFFER             0C1039
55     101B            LDA     THREE              00102D
60     101E            STA     LENGTH             0C1036
65     1021            JSUB    WREC               482061
70     1024            LDL     RETADR             081033
75     1027            RSUB                       4C0000
80     102A   EOF      BYTE    C'EOF'             454E46
85     102D   THREE    WORD    3                  000003
90     1030   ZERO     WORD    0                  000000
95     1033   RETADR   RESW    1
100    1036   LENGTH   RESW    1
105    1039   BUFFER   RESB    4096
```

Example of Program Relocation (2/2)

- Example Fig. 2.6:
  - Except for absolute address, rest of the instructions need not be modified
    - – not a memory address (immediate addressing)
    - – PC-relative, Base-relative
  - Parts requiring modification at load time are those with absolute addresses

```
5     0000   COPY     START   ~~0~~ → 1000
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   0036   BUFFER   RESB     4096
```

2.2.2 Program Relocation



Note that no matter where the program is loaded, RDREC is always 1036 bytes past the starting address of the program. This means that we can solve the relocation problem in the following way:

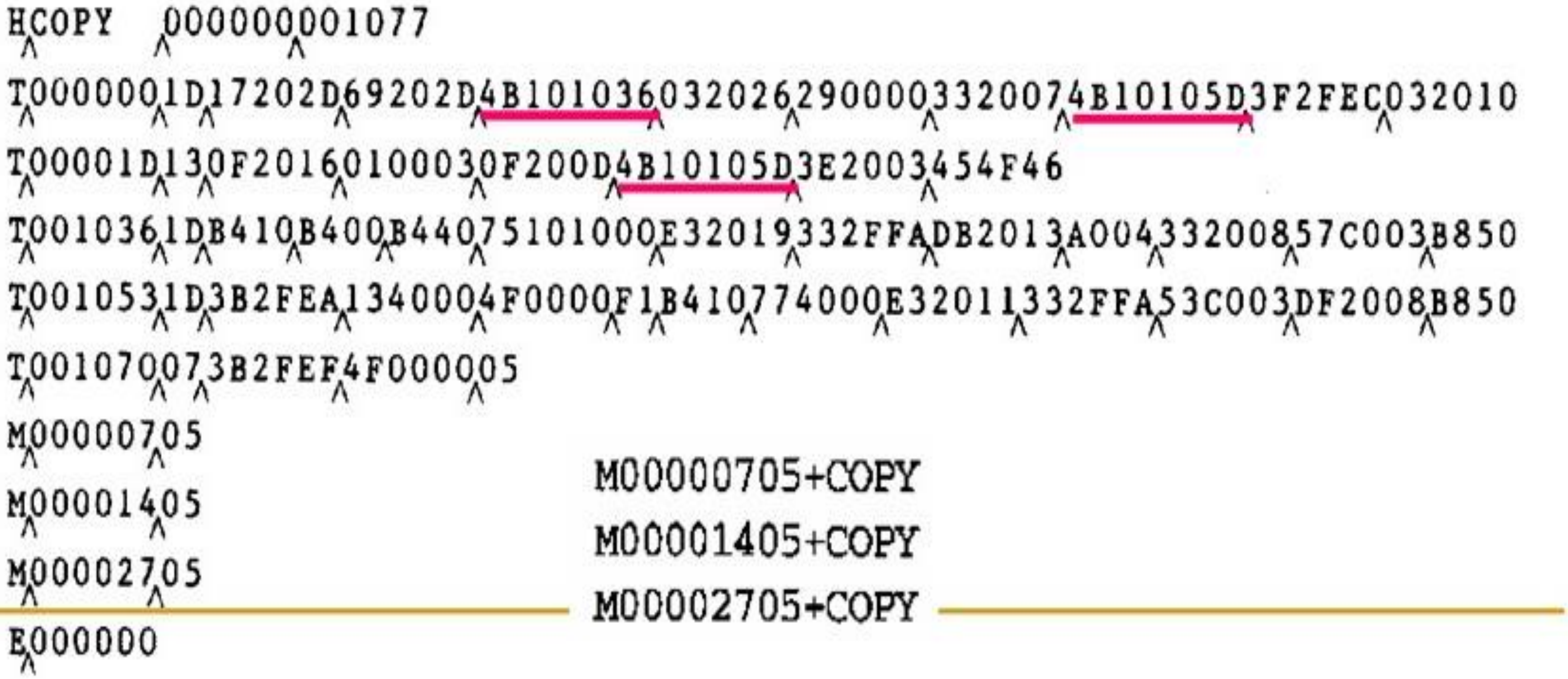
1. When the assembler generates the object code for the JSUB instruction we are considering, it will insert the address of RDREC *relative to the start of the program.* (This is the reason we initialized the location counter to 0 for the assembly.)
2. The assembler will also produce a command for the loader, instructing it to *add* the beginning address of the program to the address field in the JSUB instruction at load time.

Relocatable Program

- An object program that contains information needed for address modification for loading
- Modification record
  - Col 1 M
  - Col 2-7 Starting location of the address field to be modified, relative to the beginning of the program
  - Col 8-9 length of the address field to be modified.

Program Relocation

M^000007^05



```
HCOPY  000000001077
T0000001D17202D69202D4B1010360320262900003320074B10105D3F2FEC032010
T00001D130F20160100030F200D4B10105D3E2003454F46
T0010361DB410B400B44075101000E32019332FFADB2013A00433200857C003B850
T0010531D3B2FEA1340004F0000F1B410774000E32011332FFA53C003DF2008B850
T001070073B2FEF4F000005
M00000705
M00001405
M00002705
E000000
```

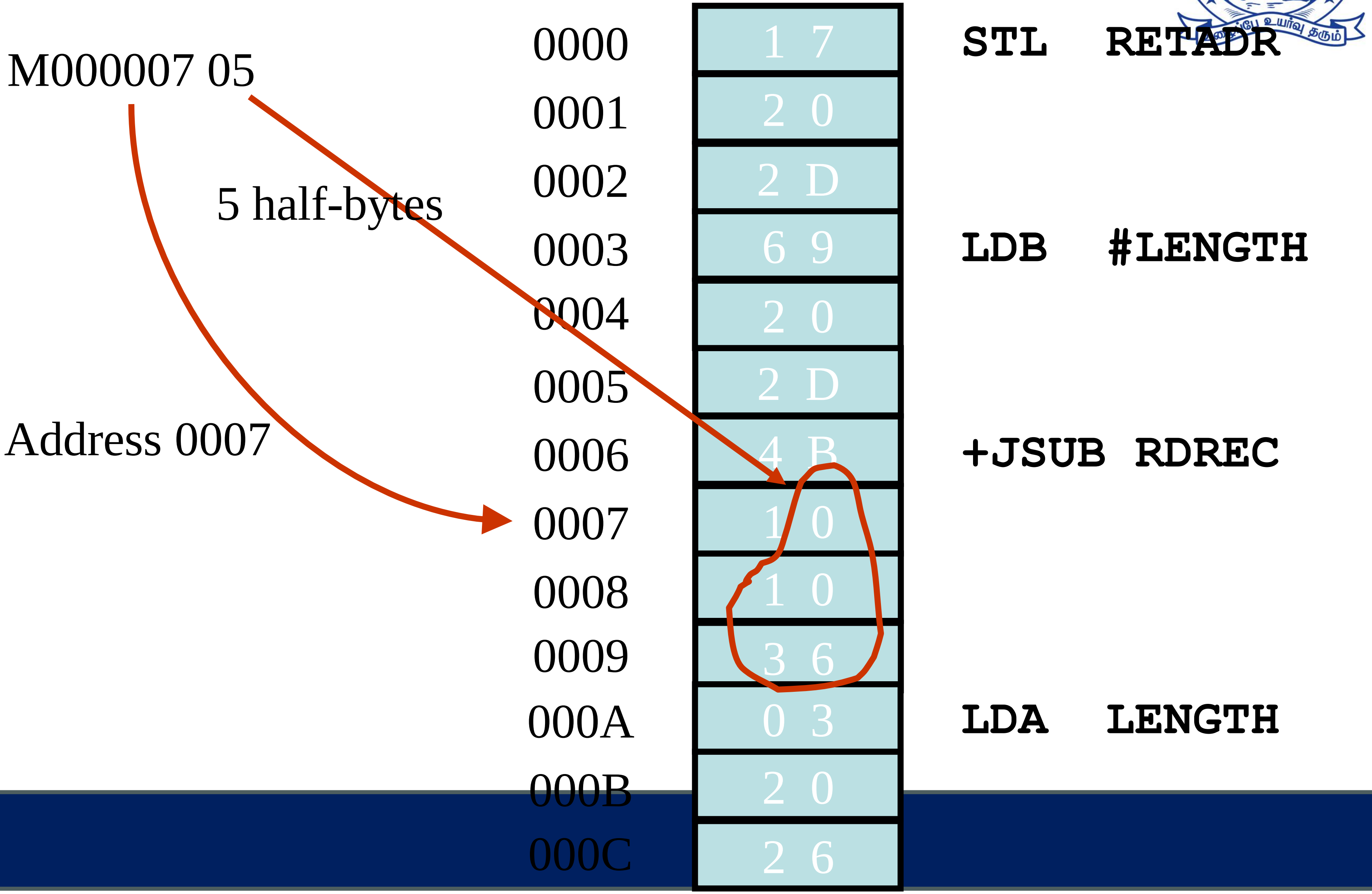
```
M00000705+COPY
M00001405+COPY
M00002705+COPY
```

Object File with M-Records

- Modification records are added to the object files. (See pp.64-65 and Figure 2.8.)
- Example:

```
HCOPY 001000 001077
T000000 1D 17202D…4B101036…
T00001D ……
…
M000007 05    ← Modification Record
……
E000000
```

Modification Record



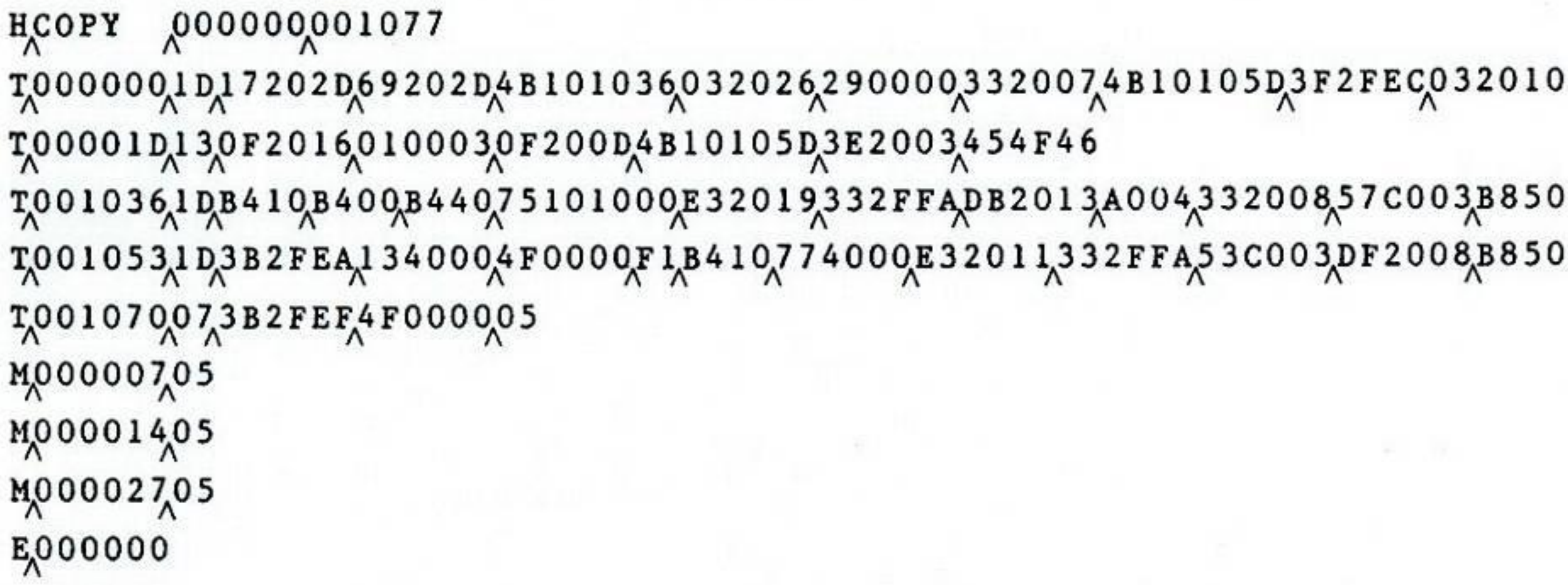
M000007 05

5 half-bytes

Address 0007

0000	1 7	STL RETADR
0001	2 0	
0002	2 D	
0003	6 9	LDB #LENGTH
0004	2 0	
0005	2 D	
0006	4 B	+JSUB RDREC
0007	1 0	
0008	1 0	
0009	3 6	
000A	0 3	LDA LENGTH
000B	2 0	
000C	2 6	

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
```

Figure 2.8 Object program corresponding to Fig. 2.6.