SWE4001 – System Programming
Module 3: Assembler
Lesson 6 of 9: Machine independent features of Assembler

Outline

- Machine Independent Assembler Features
  - Literals
  - Symbol Defining Statements
  - Expressions
  - Program Blocks
  - Control Sections and Program Linking

Literals

- Design idea
  - Let programmers write the value of a constant operand as a part of the instruction that uses it
  - Avoid having to define the constant elsewhere in the program and make up a label for it

- Example (Fig. 2.10)

```
 45    001A  ENDFIL   LDA    =C'EOF'
215    1062  WLOOP    TD     =X'05'
```

- A literal is identified with the prefix =

```
45    001A        ENDFIL        LDA    =C'EOF'        032010
```

  - Specifies a 3-byte operand whose value is the character string EOF.

```
215   1062        WLOOP         TD     =X'05'         E32011
```

  - Specifies a 1-byte literal with the hexadecimal value 05

```
5          COPY     START   0
10         FIRST    STL     RETADR
13                  LDB     #LENGTH
14                  BASE    LENGTH
15         CLOOP    +JSUB   RDREC
20                  LDA     LENGTH
25                  COMP    #0
30                  JEQ     ENDFIL
35                  +JSUB   WRREC
40                  J       CLOOP
45         ENDFIL   LDA     =C'EOF'
50                  STA     BUFFER
55                  LDA     #3
60                  STA     LENGTH
65                  +JSUB   WRREC
70                  J       @RETADR
93                  LTORG
95         RETADR   RESW    1
100        LENGTH   RESW    1
105        BUFFER   RESB    4096
106        BUFEND   EQU     *
107        MAXLEN   EQU     BUFEND - BUFFER
```

```
115         .     READ RECORD INTO BUFFER
120         .
125         RDREC  CLEAR    X
130                CLEAR    A
132                CLEAR    S
133                +LDT     #MAXLEN
135         RLOOP  TD       INPUT
140                JEQ      RLOOP
145                RD       INPUT
150                COMPR    A,S
155                JEQ      EXIT
160                STCH     BUFFER,X
165                TIXR     T
170                JLT      RLOOP
175         EXIT   STX      LENGTH
180                RSUB
185         INPUT  BYTE     X'F1'
```

```
195         .
200         .     WRITE RECORD FROM BUFFER
205         .
210         WRREC   CLEAR   X
212                 LDT     LENGTH
215         WLOOP   TD      =X'05'
220                 JEQ     WLOOP
225                 LDCH    BUFFER,X
230                 WD      =X'05'
235                 TIXR    T
240                 JLT     WLOOP
245                 RSUB
255                 END     FIRST
```

2.3.1 Literals

④ The difference between literal operands and immediate operands

⑨ =, #

⑨ Immediate addressing, the operand value is assembled as part of the machine instruction, no memory reference.

⑨ With a literal, the assembler generates the specified value as a constant at some other memory location. The *address* of this generated constant is used as the TA for the machine instruction, using PC-relative or base-relative addressing with memory reference.

Literal - Implementation

- *Literal pools*
  - Normally literals are placed into a pool at the end of the program
    - See Fig. 2.10 (END statement)
  - In some cases, it is desirable to place literals into a pool at some other location in object program
    - Use assembler directive LTORG: create a literal pool that contains all of the literal operands used since the previous LTROG, and place it where LTORG was encountered
    - Reason: keep literal operand close to the instruction

```
5    0000   COPY     START   0
10   0000   FIRST    STL     RETADR   17202D
13   0003            LDB     #LENGTH  69202D
14                   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     =C'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
93                   LTORG
     002D   *        =C'EOF'          454F46
95   0030   RETADR   RESW    1
100  0033   LENGTH   RESW    1
105  0036   BUFFER   RESB    4096
106  1036   BUFEND   EQU     *
107  1000   MAXLEN   EQU     BUFEND - BUFFER
```

```
115         .     READ RECORD INTO BUFFER
120         .
125  1036 RDREC  CLEAR    X              B410
130  1038        CLEAR    A              B400
132  103A        CLEAR    S              B440
133  103C        +LDT     #MAXLEN        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
```

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

2.3.1 Literals

- When to use LTORG
  - The literal operand would be placed too far away from the instruction referencing.
  - Cannot use PC-relative addressing or Base-relative addressing to generate Object Program.
- Most assemblers recognize duplicate literals.
  - By comparison of the character strings defining them.
  - =C'EOF' and =X'454F46'

2.3.1 Literals

- Allow literals that refer to the current value of the location counter.
  - Such literals are sometimes useful for loading base registers.

```
LDB     =*
```

; register B=beginning address of statement=current LOC

```
BASE    *
```

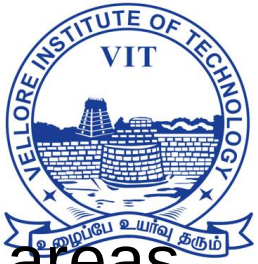
; for base relative addressing

- If a literal **=*** appeared on line 13 or 55
  - Specify an operand with value 0003 (Loc) or 0020 (Loc).

2.3.1 Literals

- Literal table (LITTAB)
  - Contains the literal name (=C'EOF'), the operand value (454F46) and length (3), and the address (002D).
  - Organized as a hash table.
  - Pass 1, the assembler creates or searches LITTAB for the specified literal name.
  - Pass 1 encounters a LTORG statement or the end of the program, the assembler makes a scan of the literal table.
  - Pass 2, the operand address for use in generating OC is obtained by searching LITTAB.

Symbol-Defining Statements



- Users can define *labels* on instructions or data areas
  - The value of a label is the address assigned to the statement
- Users can also define *symbols* with values

  `symbol  EQU  value`
  - **`value`** can be constants, other symbols, expressions
  - Making source program easier to understand
  - No forward reference

Symbol-Defining Statements

- Example 1:

```
MAXLEN    EQU     4096
133       +LDT    #MAXLEN
```

```
          +LDT    #4096
```

- Example 2:

```
BASE      EQU     R1
COUNT     EQU     R2
INDEX     EQU     R3
```

- Example 3:

```
MAXLEN    EQU     BUFEND-BUFFER
```

2.3.2 Symbol-Defining Statements

- Allow the programmer to define symbols and specify their values.
  - Assembler directive EQU.
  - Improved readability in place of numeric values.

```
        +LDT  #4096
MAXLEN  EQU   BUFEND-BUFFER (4096)
        +LDT  #MAXLEN
```

- Use EQU in defining mnemonic names for registers.
  - Registers A, X, L can be used by numbers 0, 1, 2.

```
RMO   0, 1
RMO   A, X
```

```
A       EQU     0
X       EQU     1
L       EQU     2
```

2.3.2 Symbol-Defining Statements

- The standard names reflect the usage of the registers.

```
BASE    EQU   R1
COUNT   EQU   R2
INDEX   EQU   R3
```

- Assembler directive ORG
  - Use to indirectly assign values to symbols.

    ORG value
  - The assembler resets its LOCCTR to the specified value.
  - ORG can be useful in label definition.

2.3.2 Symbol-Defining Statements

④ All terms used to specify the value of the new symbol --- must have been defined previously in the program.

```
BETA     EQU   ALPHA
ALPHA    RESW  1
```

Need 2 passes

2.3.2 Symbol-Defining Statements

- All symbols used to specify new location counter value must have been previously defined.

- Forward reference

```
ALPHA    EQU   BETA
BETA     EQU   DELTA
DELTA    RESW  1
```

  Need 3 passes

2.3.3 Expressions

- Allow arithmetic expressions formed
  - Using the operators +, -, ×, /.
  - Division is usually defined to produce an integer result.
  - Expression may be constants, user-defined symbols, or special terms.

```
106     1036        BUFEND    EQU  *
```

  - Gives BUFEND a value that is the address of the next byte after the buffer area.
- Absolute expressions or relative expressions
  - A relative term or expression represents some value (S+r), S: starting address, r: the relative value.

2.3.3 Expressions

107 1000 MAXLEN EQU BUFEND-BUFFER

- Both BUFEND and BUFFER are relative terms.
- The expression represents absolute value: the *difference* between the two addresses.
- Loc =1000 (Hex)
- The value that is associated with the symbol that appears in the source statement.
- BUFEND+BUFFER, 100-BUFFER, 3*BUFFER represent neither absolute values nor locations.

④ Symbol tables entries

Symbol	Type	Value
RETADR	R	0030
BUFFER	R	0036
BUFEND	R	1036
MAXLEN	A	1000

SYMTAB

Name	Value
COPY	0
FIRST	0
CLOOP	6
ENDFIL	1A
RETADR	30
LENGTH	33
BUFFER	36
BUFEND	1036
MAXLEN	1000
RDREC	1036
RLOOP	1040
EXIT	1056
INPUT	105C
WREC	105D
WLOOP	1062

LITTAB

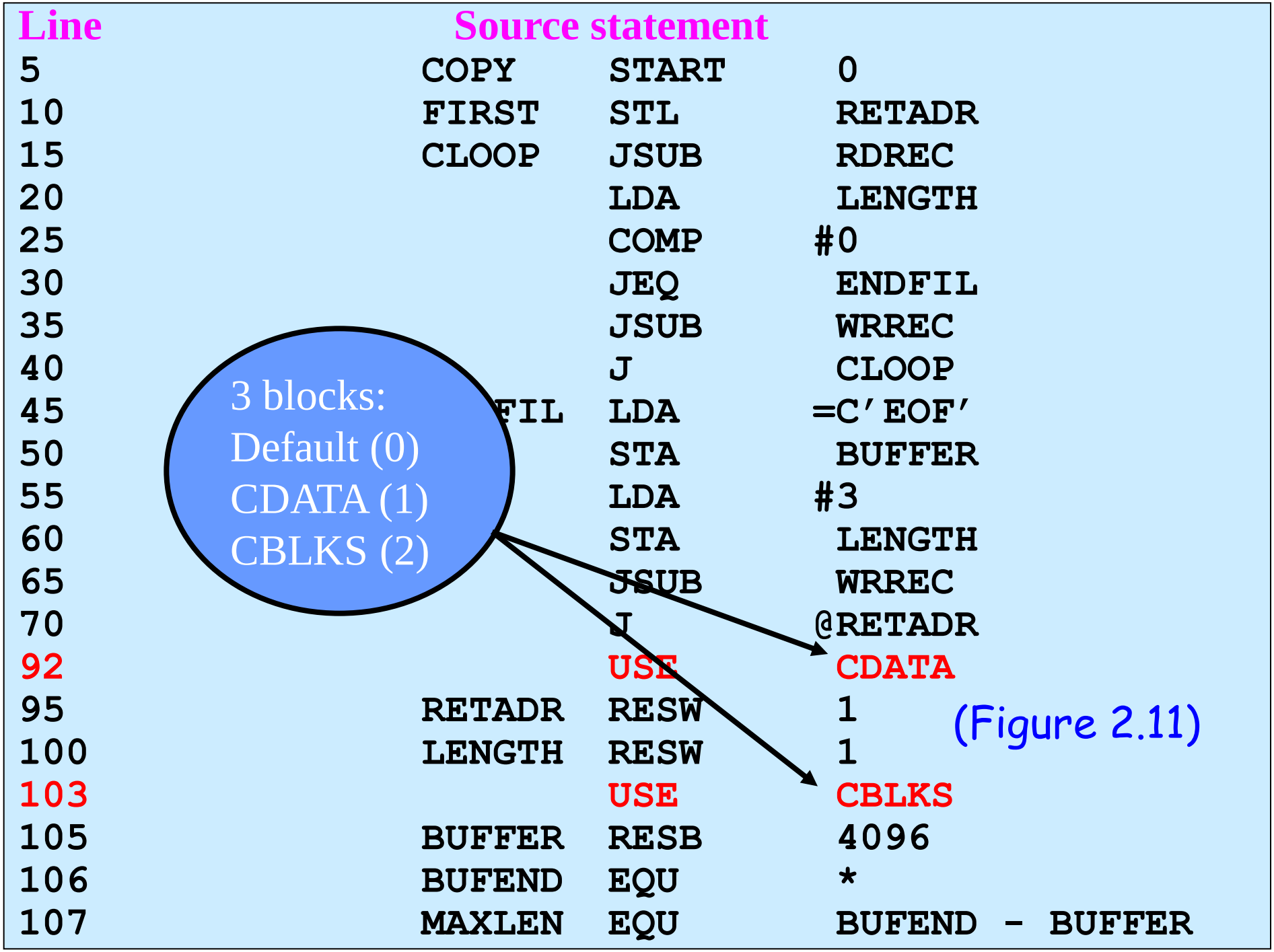
C'EOF'	454F46	3	002D
X'05'	05	1	1076

2.3.4 Program Blocks

- The source program logically contained main, subroutines, data areas.
  - In a single block of object code.
- More flexible (Different blocks)
  - Generate machine instructions (codes) and data in a different order from the corresponding source statements.
- Program blocks
  - Refer to segments of code that are rearranged within a single object program unit.
- Control sections
  - Refer to segments of code that are translated into independent object program units.

2.3.4 Program Blocks

- Three blocks, Figure 2.11
  - Default (USE), CDATA (USE CDATA), CBLKS (USE CBLKS).
  - Assembler directive USE
    - Indicates which portions of the source program blocks.
    - At the beginning of the program, statements are assumed to be part of the default block.
    - Lines 92, 103, 123, 183, 208, 252.
- Each program block may contain several separate segments.
    - The assembler will rearrange these segments to gather together the pieces of each block.


Line
Source statement
5 COPY START 0
10 FIRST STL RETADR
15 CLOOP JSUB RDREC
20 LDA LENGTH
25 COMP #0
30 JEQ ENDFIL
35 JSUB WRREC
40 J CLOOP
45 FIL LDA =C'EOF'
50 STA BUFFER
55 LDA #3
60 STA LENGTH
65 JSUB WRREC
70 J @RETADR
92 USE CDATA
95 RETADR RESW 1
100 LENGTH RESW 1
103 USE CBLKS
105 BUFFER RESB 4096
106 BUFEND EQU *
107 MAXLEN EQU BUFEND - BUFFER
3 blocks:
Default (0)
CDATA (1)
CBLKS (2)
(Figure 2.11)

```
115         .     READ RECORD INTO BUFFER
120         .
123                     USE
125               RDREC  CLEAR  X
130                      CLEAR  A
132                      CLEAR  S
133                     +LDT    #MAXLEN
135               RLOOP  TD      INPUT
140                      JEQ    RLOOP
145                      RD     INPUT
150                      COMPR   A,S
155                      JEQ    EXIT
160                      STCH   BUFFER,X
165                      TIXR   T
170                      JLT    RLOOP
175               EXIT   STX    LENGTH
180                      RSUB
183                      USE    CDATA
185               INPUT  BYTE   X'F1'
```

```
195         .
200         .     WRITE RECORD FROM BUFFER
205         .
208                       USE
210               WRREC   CLEAR   X
212                       LDT     LENGTH
215               WLOOP   TD      =X'05'
220                       JEQ     WLOOP
225                       LDCH    BUFFER,X
230                       WD      =X'05'
235                       TIXR    T
240                       JLT     WLOOP
245                       RSUB
252                       USE     CDATA
253                       LTORG


255                 END     FIRST
```

2.3.4 Program Blocks

- Pass 1, Figure 2.12
  - The block number is started form 0.
  - A separate location counter for each program block.
  - The location counter for a block is initialized to 0 when the block is first begun.
  - Assign each block a starting address in the object program (location 0).
  - Labels, block name or block number, relative addr.
  - Working table is generated

Block name	Block number	Address	End	Length
default	**0**	**0000**	**0065**	**0066 (0~0065)**
CDATA	**1**	**0066**	**0070**	**000B (0~000A)**
CBLKS	**2**	**0071**	**1070**	**1000 (0~0FFF)**

Line	Loc	Block	Source statement			Object code
5	0000	0	COPY	START	0	
10	0000	0	FIRST	STL	RETADR	172063
15	0003	0	CLOOP	JSUB	RDREC	4B2021
20	0006	0		LDA	LENGTH	032060
25	0009	0		COMP	#0	290000
30	000C	0		JEQ	ENDFIL	332006
35	000F	0		JSUB	WRREC	4B203B
40	0012	0		J	CLOOP	3F2FEE
45	0015	0	ENDFIL	LDA	=C'EOF'	032055
50	0018	0		STA	BUFFER	0F2056
55	001B	0		LDA	#3	010003
60	001E	0		STA	LENGTH	0F2048
65	0021	0		JSUB	WRREC	4B2029
70	0024	0		J	@RETADR	3E203F
92	0000	1		USE	CDATA	
95	0000	1	RETADR	RESW	1	
100	0003	1	LENGTH	RESW	1	
103	0000	2		USE	CBLKS	
105	0000	2	BUFFER	RESB	4096	
106	1000	2	BUFEND	EQU	*	
107	1000		MAXLEN	EQU	BUFEND-BUFFER	

```
110                                 .
115                                 .     SUBROUTINE TO READ RECORD INTO BUFFER
120                                 .
123     0027  0                     USE
125     0027  0     RDREC           CLEAR   X           B410
130     0029  0                     CLEAR   A           B400
132     002B  0                     CLEAR   S           B440
133     002D  0                    +LDT     #MAXLEN     75101000
135     0031  0     RLOOP           TD      INPUT       E32038
140     0034  0                     JEQ     RLOOP       332FFA
145     0037  0                     RD      INPUT       DB2032
150     003A  0                     COMPR   A,S         A004
155     003C  0                     JEQ     EXIT        332008
160     003F  0                     STCH    BUFFER,X    57A02F
165     0042  0                     TIXR    T           B850
170     0044  0                     JLT     RLOOP       3B2FEA
175     0047  0     EXIT            STX     LENGTH      13201F
180     004A  0                     RSUB                4F0000
183     0006  1                     USE     CDATA
185     0006  1     INPUT           BYTE    X'F1'       F1
```

```
195                     .
200                     .       SUBROUTINE TO WRITE RECORD FROM BUFFER
205                     .
208     004D    0               USE
210     004D    0       WRREC   CLEAR   X               B410
212     004F    0               LDT     LENGTH          772017
215     0052    0       WLOOP   TD      =X'05'          E3201B
220     0055    0               JEQ     WLOOP           332FFA
225     0058    0               LDCH    BUFFER,X        53A016
230     005B    0               WD      =X'05'          DF2012
235     005E    0               TIXR    T               B850
240     0060    0               JLT     WLOOP           3B2FEF
245     0063    0               RSUB                    4F0000
252     0007    1               USE     CDATA
253                             LTORG
        0007    1       *       =C'EOF'                 454F46
        000A    1       *       =X'05'                  05
255                             END     FIRST
```

Figure 2.12 Program from Fig. 2.11 with object code.

2.3.4 Program Blocks

- Pass 2, Figure 2.12
  - The assembler needs the address for each symbol relative to the start of the object program.
  - Loc shows the relative address and block number.
  - Notice that the value of the symbol MAXLEN (line 70) is shown without a block number.

```
20      0006 0    LDA      LENGTH 032060
```

0003(CDATA) +0066 =0069 =TA

using program-counter relative addressing

TA - (PC) =0069-0009 =0060 =disp

2.3.4 Program Blocks

- Separation of the program into blocks.
  - Because the large buffer (CBLKS) is moved to the end of the object program.
  - No longer need extended format, base register, simply a LTORG statement.
- No need Modification records.
  - Improve program readability.
  - Figure 2.13
  - Reflect the starting address of the block as well as the relative location of the code within the block.
- Figure 2.14
  - Loader simply loads the object code from each record at the dictated.
  - CDATA(1) & CBLKS(1) are not actually present in OP.

2.3.4 Program Blocks

```
HCOPY  000000001071
T0000001E1720634B20210320602900003320064B203B3F2FEE0320550F2056010003
T00001E090F20484B20293E203F                                          Default 1
T0000271DB410B400B44075101000E32038332FFADB2032A00433200857A02FB850
T000044093B2FEA13201F4F0000                                          Default 2
T00006C01F1                                                          CDATA 2
T00004D19B410772017E3201B332FFA53A016DF2012B8503B2FEF4F0000          Default 3
T00006D04454F4605                                                    CDATA 3
E000000
```

Figure 2.13 Object program corresponding to Fig. 2.11.

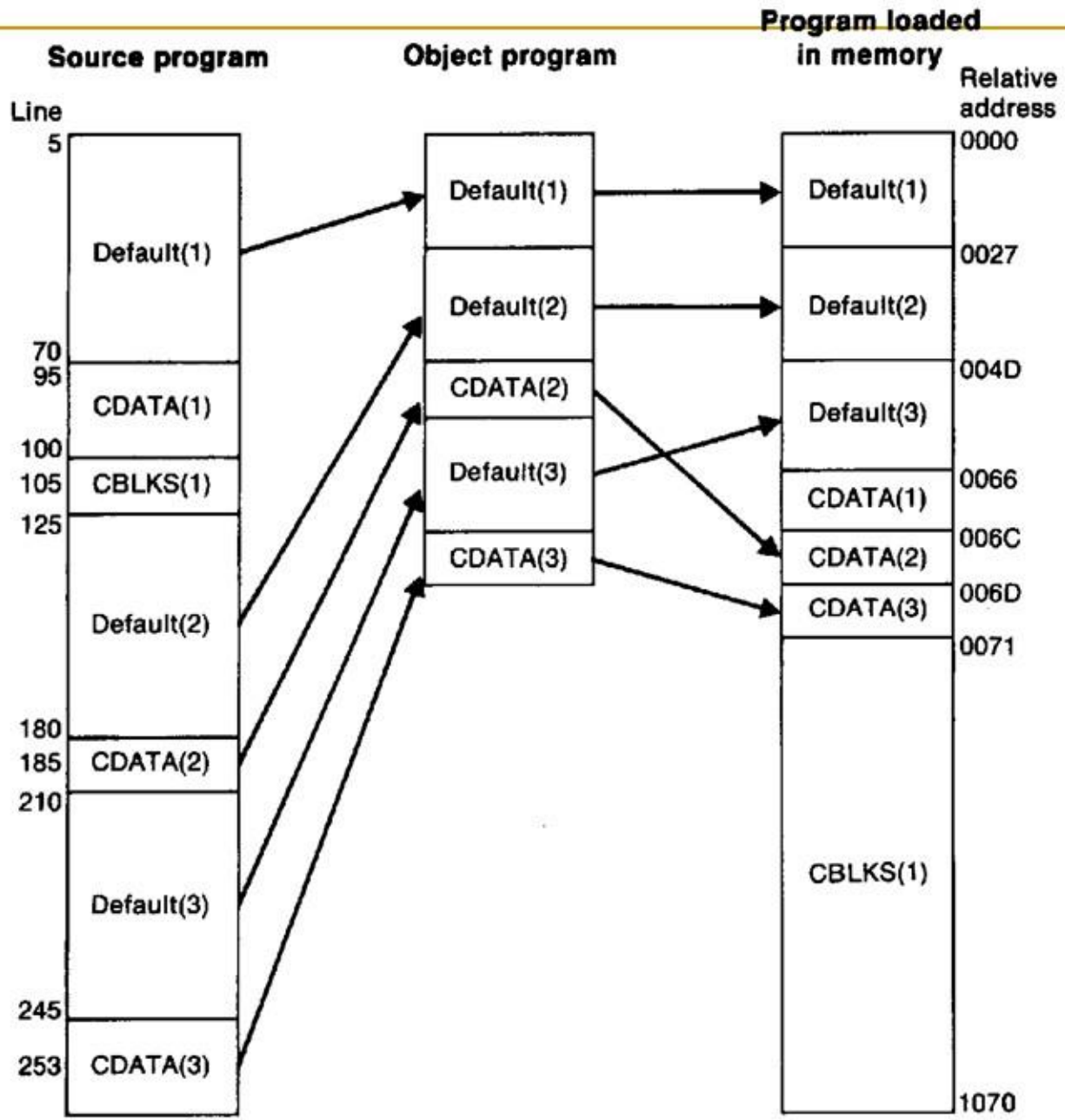


Figure 2.14 Program blocks from Fig. 2.11 traced through the assembly and loading processes.

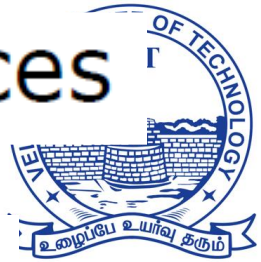
2.3.5 Control Sections & Program Linking

- Control section
  - Handling of programs that consist of multiple control sections.
  - Each control section is a part of the program.
  - Can be assembled, loaded and relocated independently.
  - Different control sections are most often used for subroutines or other logical subdivisions of a program.
  - The programmer can assemble, load, and manipulate each of these control sections separately.
  - More Flexibility then the previous.
  - Linking control sections together.

2.3.5 Control Sections & Program Linking

- External references (external symbol references)
  - Instructions in one control section might need to refer to instructions or data located in another section.
- Figure 2.15, multiple control sections.
  - Three sections, main COPY, RDREC, WRREC.
  - Assembler directive CSECT.
  - Assembler directives EXTDEF and EXTREF for external symbols.
  - The order of symbols is not significant.

External Definition and References



External definition

- **EXTDEF name [, name]**
- EXTDEF names symbols that are defined in this control section and may be used by other sections

External reference

- **EXTREF name [,name]**
- EXTREF names symbols that are used in this control section and are defined elsewhere

```
COPY    START   0
        EXTDEF  BUFFER, BUFEND, LENGTH
        EXTREF  RDREC, WRREC (symbol name)
```

Line **Source statement**

```
  5     COPY      START   0                   COPY FILE FROM INPUT TO OUTPUT
  6               EXTDEF  BUFFER,BUFEND,LENGTH
  7               EXTREF  RDREC,WRREC
 10     FIRST     STL     RETADR              SAVE RETURN ADDRESS
 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     =C'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
 95     RETADR    RESW    1
100     LENGTH    RESW    1                   LENGTH OF RECORD
103               LTORG
105     BUFFER    RESB    4096                4096-BYTE BUFFER AREA
106     BUFEND    EQU     *
107     MAXLEN    EQU     BUFEND-BUFFER
```

```
109     RDREC     CSECT
110     .
115     .         SUBROUTINE TO READ RECORD INTO BUFFER
120     .
122               EXTREF    BUFFER,LENGTH,BUFEND
125               CLEAR     X                  CLEAR LOOP COUNTER
130               CLEAR     A                  CLEAR A TO ZERO
132               CLEAR     S                  CLEAR S TO ZERO
133               LDT       MAXLEN
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
190     MAXLEN    WORD      BUFEND-BUFFER
```

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

Figure 2.15 Illustration of control sections and program linking.

2.3.5 Control Sections & Program Linking

- Figure 2.16, the generated object code.

```
15      0003   CLOOP +JSUB RDREC        4B100000
160     0017         +STCH BUFFER,X     57900000
```

  - The LOC of all control section is started form 0
  - RDREC is an external reference.
  - The assembler has no idea where the control section containing RDREC will be loaded, so it cannot assemble the address.
  - The proper address to be inserted at load time.
  - Must use extended format instruction for external reference (M records are needed).

```
190     0028   MAXLEN         WORD    BUFEND-BUFFER
```

  - An expression involving two external references.

Line	Loc	Source statement			Object code
5	0000	COPY	START	0	
6			EXTDEF	BUFFER, BUFEND, LENGTH	
7			EXTREF	RDREC, WRREC	
10	0000	FIRST	STL	RETADR	172027
15	0003	CLOOP	+JSUB	RDREC	4B100000
20	0007		LDA	LENGTH	032023
25	000A		COMP	#0	290000
30	000D		JEQ	ENDFIL	332007
35	0010		+JSUB	WRREC	4B100000
40	0014		J	CLOOP	3F2FEC
45	0017	ENDFIL	LDA	=C'EOF'	032016
50	001A		STA	BUFFER	0F2016
55	001D		LDA	#3	010003
60	0020		STA	LENGTH	0F200A
65	0023		+JSUB	WRREC	4B100000
70	0027		J	@RETADR	3E2000
95	002A	RETADR	RESW	1	
100	002D	LENGTH	RESW	1	
103			LTORG		
	0030	*	=C'EOF'		454F46
105	0033	BUFFER	RESB	4096	
106	1033	BUFEND	EQU	*	
107	1000	MAXLEN	EQU	BUFEND-BUFFER	

```
109     0000    RDREC     CSECT
110             .
115             .         SUBROUTINE TO READ RECORD INTO BUFFER
120             .
122                       EXTREF    BUFFER,LENGTH,BUFEND
125     0000              CLEAR     X          B410
130     0002              CLEAR     A          B400
132     0004              CLEAR     S          B440
133     0006              LDT       MAXLEN     77201F
135     0009    RLOOP     TD        INPUT      E3201B
140     000C              JEQ       RLOOP      332FFA
145     000F              RD        INPUT      DB2015
150     0012              COMPR     A,S        A004
155     0014              JEQ       EXIT       332009
160     0017              +STCH     BUFFER,X   57900000
165     001B              TIXR      T          B850
170     001D              JLT       RLOOP      3B2FE9
175     0020    EXIT      +STX      LENGTH     13100000
180     0024              RSUB                 4F0000
185     0027    INPUT     BYTE      X'F1'      F1
190     0028    MAXLEN    WORD      BUFEND-BUFFER  000000
```

```
193       0000     WRREC     CSECT
195                  .
200                  .        SUBROUTINE TO WRITE RECORD FROM BUFFER
205                  .
207                           EXTREF   LENGTH,BUFFER
210       0000                CLEAR    X                    B410
212       0002               +LDT      LENGTH               77100000
215       0006     WLOOP      TD       =X'05'               E32012
220       0009                JEQ      WLOOP               332FFA
225       000C               +LDCH     BUFFER,X             53900000
230       0010                WD       =X'05'               DF2008
235       0013                TIXR     T                    B850
240       0015                JLT      WLOOP               3B2FEE
245       0018                RSUB                          4F0000
255                           END      FIRST
          001B     *         =X'05'                        05
```

Figure 2.16 Program from Fig. 2.15 with object code.

2.3.5 Control Sections & Program Linking

- The loader will add to this data area with the address of BUFEND and subtract from it the address of BUFFER. (COPY and RDREC for MAXLEN)
- Line 190 and 107, in 107, the symbols BUFEND and BUFFER are defined in the same section.
- The assembler must remember in which control section a symbol is defined.
- The assembler allows the same symbol to be used in different control sections, lines 107 and 190.

④ Figure 2.17, two new records.

- Defined record for EXTDEF, relative address.
- Refer record for EXTREF.

Define record:

Col. 1	D
Col. 2–7	Name of external symbol defined in this control section
Col. 8–13	Relative address of symbol within this control section (hexadecimal)
Col. 14–73	Repeat information in Col. 2–13 for other external symbols

Refer record:

Col. 1	R
Col. 2–7	Name of external symbol referred to in this control section
Col. 8–73	Names of other external reference symbols

The other information needed for program linking is added to the Modification record type. The new format is as follows.

Modification record (revised):

Col. 1	M
Col. 2–7	Starting address of the field to be modified, relative to the beginning of the control section (hexadecimal)
Col. 8–9	Length of the field to be modified, in half-bytes (hexadecimal)
Col. 10	Modification flag (+ or –)
Col. 11–16	External symbol whose value is to be added to or subtracted from the indicated field

2.3.5 Control Sections & Program Linking

- Modification record
  - M
  - Starting address of the field to be modified, relative to the beginning of the control section (Hex).
  - Length of the field to be modified, in *half-bytes*.
  - Modification flag (+ or -).
  - External symbol.

M^000004^05+RDREC

M^000028^06+BUFEND

M^000028^06-BUFFER

to

```
M00000705
M00001405
M00002705
```

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

- Use Figure 2.8 for program relocation.

```
HCOPY  000000001033
DBUFFER000033BUFEND001033LENGTH00002D
RRDREC WRREC
T00000001D1720274B1000000320232900003320074B1000003F2FEC0320160F2016
T00001D0D0100030F200A4B1000003E2000
T00003003454F46
M00000405+RDREC
M00001105+WRREC
M00002405+WRREC
E000000

HRDREC 000000000002B
RBUFFERLENGTHBUFEND
T00000001DB410B400B44077201FE3201B332FFADB2015A00433200957900000B850
T00001D0E3B2FE9131000004F0000F1000000
M00001805+BUFFER
M00002105+LENGTH
M00002806+BUFEND
M00002806-BUFFER
E
```

```
HWRREC 000000000001C
RLENGTHBUFFER
T0000001CB41077100000E32012332FFA53900000DF2008B8503B2FEE4F000005
M00000305+LENGTH
M00000D05+BUFFER
E
```

Figure 2.17 Object program corresponding to Fig. 2.15.