Systems Programming
Assemblers – Part 3-3
Program Blocks

Prof. Dr. Hani Mahdi
Department of Computer Science
Al-Isra University, Amman, Jordan

1

Assembler Design

2.1 Basic Assembler Functions
2.2 Machine Dependent Assembler Features
- instruction formats and addressing modes
- program relocation

2.3. Machine Independent Assembler Features
1. literals
2. symbol-defining statements
3. expressions
4. program blocks
5. control sections and program linking

- These are related to:
  - Programmer convenience
  - Software environment
- Assembler directives are widely used to support these features

Program Blocks and Control Sections

- Although the source program logically contains subroutines, data area, etc, they were assembled into a single block of object code in which the machine instructions and data appeared in the same order as they were in the source program.
- To provide flexibility:
  - Program blocks
    - Segments of code that are rearranged within a single object program unit
  - Control sections
    - Segments of code that are translated into independent object program units

Program Blocks

- Program blocks are the Segments of code that are rearranged within a single object program unit
- As an *example*, three blocks are used:
  - default: executable instructions
  - CDATA: all data areas that are less in length
  - CBLKS: all data areas that consists of larger blocks of memory
- The assembler directive USE indicates which portions of the source program belong to the various blocks.

Using USE

- Assembler directive USE indicates which portions (Block) of the source program belong to the various blocks

  `USE [blockname]`
- At the beginning, the default block is assumed.
- If no USE statements are included,
  the entire program belongs to this single block
- Example:

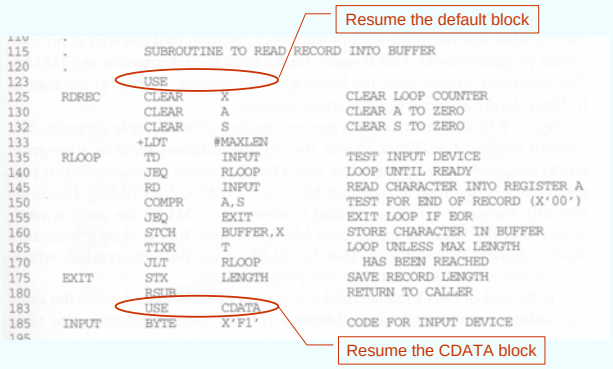
```
        .
        .
        .
92    USE   CDATA   ;begin block named CDATA
103   USE   CBLKS   ;begin block named CBLKS
183   USE           ;resume the default block
```


Program with Multiple Program Blocks

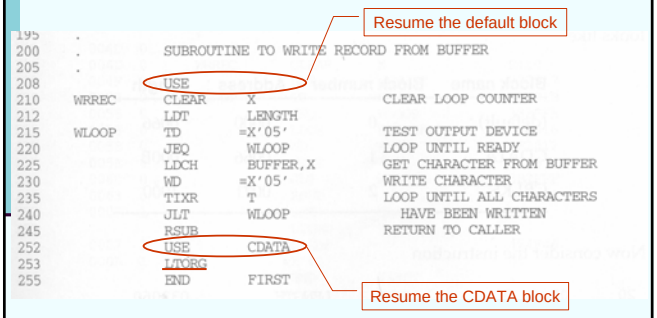
At the beginning, the default block is assumed.

```
 5    COPY      START   0               COPY FILE FROM INPUT TO OUTPUT
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
92              USE     CDATA
95    RETADR    RESW    1
100   LENGTH    RESW    1               LENGTH OF RECORD
103             USE     CBLKS
105   BUFFER    RESB    4096            4096-BYTE BUFFER AREA
106   BUFEND    EQU     *               FIRST LOCATION AFTER BUFFER
107   MAXLEN    EQU     BUFEND-BUFFER   MAXIMUM RECORD LENGTH
```

Program with Multiple Program Blocks



Program with Multiple Program Blocks



Program Blocks

- Each program block may actually contain several separate segments of the source program.
- The assembler will logically rearrange these segments to gather together the pieces of each block.
- The result is the same as if the programmer had physically rearranged the source statements to group together all the source lines belonging to each block.

Program Blocks Advantages

- To satisfy the contradictive goals:
  - Program readability is better if data areas are placed in the source program close to the statements that reference them.
  - Large buffer area is moved to the end of the object program
- Using the extended format instructions or base relative mode may be reduced. (lines 15, 35, and 65)
- Placement of literal pool is easier
  LTORG is used to make sure the literals are placed ahead of any large data areas (line 253)

How to Rearrange Codes into Program Blocks

- Pass 1
  - Maintain a separate LOCCTR for each program block
    - initialized to 0 when the block is first begun
    - saved when switching to another block
    - restored when resuming a previous block
  - Assign to each label an address relative to the start of the block that contains it
  - Store the block name or number in the SYMTAB along with the assigned relative address of the label
  - LOCCTR for each block at the end of Pass1 indicates the block length as the latest value of
  - Assign to each block a starting address in the object program by concatenating the program blocks in a particular order

How to Rearrange Codes into Program Blocks

- Pass 2
  - Calculate the address for each symbol relative to the start of the object program by adding
    - the location of the symbol relative to the start of its block
    - the assigned starting address of this block

Object Program with Multiple Program Blocks

Loc/Block

```
 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
```

0: default
1: CDATA
2: CBLKS

No block number because MAXLEN is an absolute symbol

Object Program with Multiple Program Blocks

```
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
```


Object Program with Multiple Program Blocks

```
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
```

Summary of Assembler Operations

Pass 1

Separate LOCCTR for each block, initialized to 0

Save and restore LOCCTR values when switching between two blocks

Each label is assigned an address relative to the start of the block that contains it, and label address is stored with block number in SYMTAB

Constructs a table that contains the starting addresses and lengths for all blocks

Pass 2

Generate address for each symbol relative to the start of the object program

by access SYMTAB, and add the location of the symbol to the block starting address

Table Example for Program Blocks

- At the end of Pass 1:

Block name	Block number	address	Length
(default)	0	0000	0066
CDATA	1	0066	000B
CBLKS	2	0071	1000

Symbol	Address	Bock number
LENGTH	0003	1

Example of Address Calculation

- In Pass 2

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

- The value of the operand (LENGTH)
  - Address 0003 relative to Block 1 (CDATA)

  →address 0003+0066=0069 relative to program

  →address 0069-0009=0060 relative to PC,
  in which the address of PC relative to program is 0009+0000=0009

Object Program

- It is not necessary to physically rearrange the generated code in the object program to place the pieces of each program block together.
- The assembler just simply insert the proper load address in each Text record.

```
HCOPY  000000001071
T0000001E172063482021032060290000332006482038F2FEE032055F2056010003
T00001E090F20484B20293E203F
T0000271DB410B400B44075101000E32038332FFADB2032A00433200857A02FB850
T000044093B2FEA13201F4F0000
T000006C01F1
T00004D19B410772017E3201B332FFA53A016DF2012B8503B2FEF4F0000
T000006D04454F4605
E000000
```


Program Blocks Loaded in Memory