Assembler

Assembler

- Fundamental functions
  - Translating mnemonic operation codes to their machine language equivalents
  - Assigning machine addresses to symbolic labels

- Machine dependency
  - Different machine instruction formats and codes

Assembler Functions

- Convert *mnemonic operation codes* to their *machine language equivalents*
    - eg: translate STL to 14
- Convert *symbolic operands* to their *equivalent machine addresses*
    - eg: translate RETADDR to 1033
- Build the *machine instructions* in the *proper format*
- Convert the *data constants* to *internal machine representations*
    - eg: translate EOF to 454F46
- Write the *object program* and the *assembly listing*

Assembly language Program (Data Movement)

Label	**Instruction**	**Operand**
TEST	START	1003
FIRST	LDA	FIVE
	STA	ALPHA
FIVE	WORD	5
ALPHA	RESW	1
	END	FIRST

LDA	00
STA	0C

Assembly language Program (Data Movement)

Loc	**Label**	**Instruction**	**Operand**	**Object Code**
1003	TEST	START	1003	
1003	FIRST	LDA	FIVE	00100C
1006		STA	ALPHA	0C1009
1009	FIVE	WORD	5	000005
100C	ALPHA	RESW	1	******
100F		END	FIRST	

Assembler

- Forward reference: reference to a label that is defined later in the program.
- Most assemblers make two passes over the source program
- First pass: Scans the source program for label definitions
- Second pass: Performs most of the actual translation

Object Program

- 3 types of records
  - Header record, Text record, End record
- Header Record
  - Program name, starting address and length of the program
- Text Record
  - Translated (machine code) instructions and data of the program together with an indication of the address where they are to be loaded
- End Record
  - Marks the end of the program and specifies the address in the program where the execution is to begin
  - Taken from the operand of the program’s END statement
  - If no operand is specified, address of the first executable instruction is used

Object Program

- Header Record

Col. 1	H
Col. 2-7	Program name
Col. 8-13	Starting address of the object program (hexadecimal)
Col. 14-19	Length of object program in bytes (hexadecimal)

- Text Record

Col. 1	T
Col. 2-7	Starting address for object code in this record (hexadecimal)
Col. 8-9	Length of object code in this record in bytes (hexadecimal)
Col. 10-69	Object code, represented in hexadecimal (2 columns per byte of object code)

- End Record

Col. 1	E
Col. 2-7	Address of first executable instruction in the program (hexadecimal)

Object Program

- H^TEST ^001003^00000C
- T^001003^09^001009^0C100C^000005
- E^001003

Loc	Label	Instruction	Operand	Object Code
1003	TEST	START	1003	
1003	FIRST	LDA	FIVE	001009
1006		STA	ALPHA	0C100C
1009	FIVE	WORD	5	000005
100C	ALPHA	RESW	1	******
		END	FIRST	

EXAMPLE

```
SUM      START    4000
FIRST    LDA      ALPHA
         ADD      INCR
         SUB      ONE
         STA      BETA
         LDA      GAMMA
         ADD      INCR
         SUB      ONE
         STA      DELTA
ONE      WORD     1
ALPHA    RESW     1
BETA     RESW     1
GAMMA    RESW     1
DELTA    RESW     1
INCR     RESW     1
         END      FIRST
```

Mnemonic Operation Code	Machine Language Equivalent
ADD	18
LDA	00
SUB	1C
STA	0C

EXAMPLE

```
4000    SUM     START   4000
4000    FIRST   LDA     ALPHA   00401B
4003            ADD     INCR    184027
4006            SUB     ONE     1C4018
4009            STA     BETA    1C401E
400C            LDA     GAMMA   004021
400F            ADD     INCR    184027
4012            SUB     ONE     1C4018
4015            STA     DELTA   0C4024
4018    ONE     WORD    1       000001
401B    ALPHA   RESW    1
401E    BETA    RESW    1
4021    GAMMA   RESW    1
4024    DELTA   RESW    1
4027    INCR    RESW    1
                END     FIRST
```

OPTAB	
Mnemonic Operation Code	Machine Language Equivalent
ADD	18
LDA	00
SUB	1C
STA	0C

SYMTAB	
Symbol	Address
ONE	4018
ALPHA	401B
BETA	401E
GAMMA	4021
DELTA	4024
INCR	4027

Data Structures

- Two major internal data structures:
  - Operation Code Table (OPTAB)
  - Symbol Table (SYMTAB)
- Variable: Location Counter(LOCCTR)

Data Structures

- Operation Code Table (OPTAB)
  - Used to look up mnemonic operation codes and translate them into machine language equivalents
  - Contains the mnemonic operation code and its machine language equivalent
  - In more complex assemblers, contains information like instruction format and length
- Characteristic
  - Static table
  - Entries are not normally added or deleted

Mnemonic Operation Code	**Machine Language Equivalent**	**Length (bytes)**
ADD	18	3
LDA	00	3

Data Structures

- Pass 1
  - Used to look up and validate opcodes in the source program
- Pass 2
  - Used to translate the operation codes to machine language
- Implementation
  - Organized as a hash table with mnemonic operation code as the key
  - Provides retrieval with minimum of searching
  - Information in OPTAB is predefined when the assembler itself is written

Data Structures

- Symbol Table (SYMTAB)
  - Used to store values (addresses) assigned to labels
  - Includes the name and value for each label
  - Flags to indicate error conditions, e.g. duplicate definition of labels
  - May contain other information about the data area or instruction labeled like type or length
- Characteristic
  - dynamic table (insert, delete, search)
  - Deletion is performed rarely
- Implementation
  - hash table
  - For efficiency of insertion and retrieval

LABEL	**Address (LOCCTR value)**
FIRST	1003
FIVE	1009
ALPHA	100C

Data Structures

- Pass1
  - Labels are entered into the symbol table along with their assigned addresses (from LOCCTR)
- Pass2
  - Address of the symbols used as operands are looked up in SYMTAB to insert the address in the assembled instructions

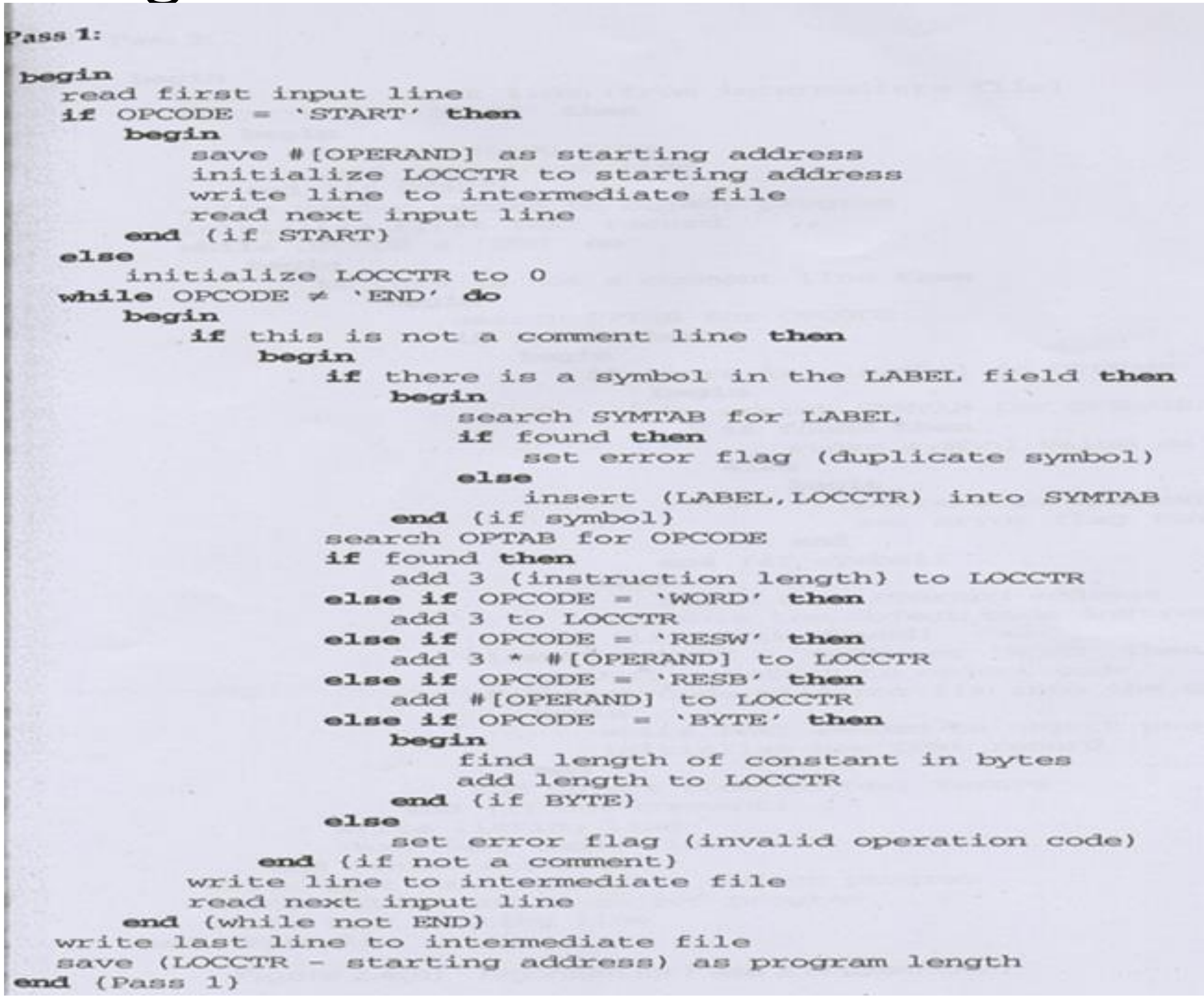
Data Structures

- LOCCTR
    - Used to help in the assignment of addresses
    - Initialized to the beginning address specified in the START statement
    - After each source statement is processed, the length of the assembled instruction or data area to be generated is added
    - Gives the address of a label
    - Counted in bytes

Two Pass Assembler

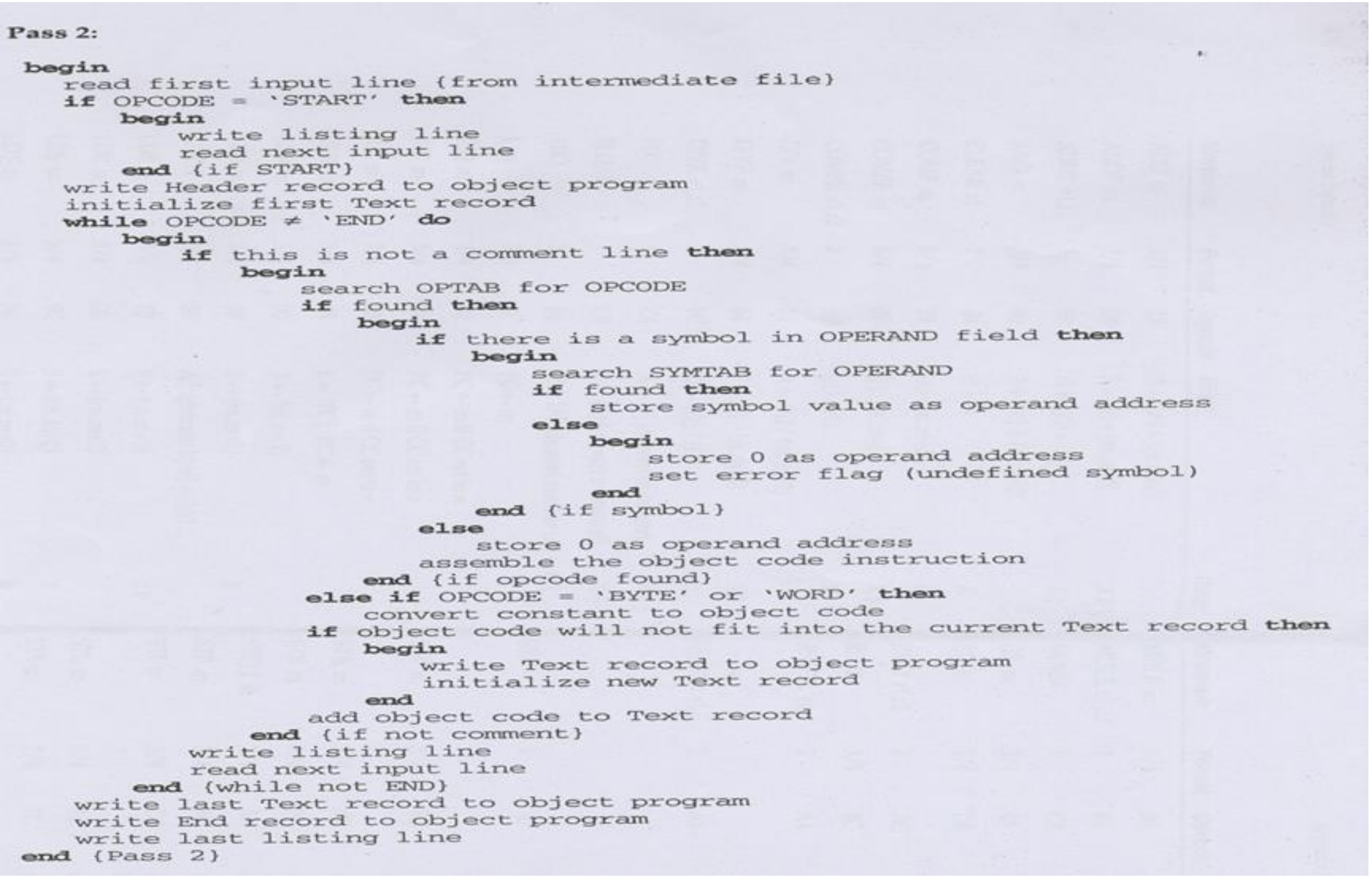
- Pass 1
  - Assign addresses to all statements in the program
  - Save the values (addresses) assigned to all labels for use in Pass 2
  - Perform some processing of assembler directives.
    - This includes processing that affects the address assignments such as determining the length of the data areas defined by BYTE, RESB etc.
- Pass 2
  - Assemble instructions
  - Generate data values defined by BYTE, WORD
  - Perform processing of assembler directives not done in Pass 1
  - Write the object program and the assembly listing

Pass 1 Algorithm



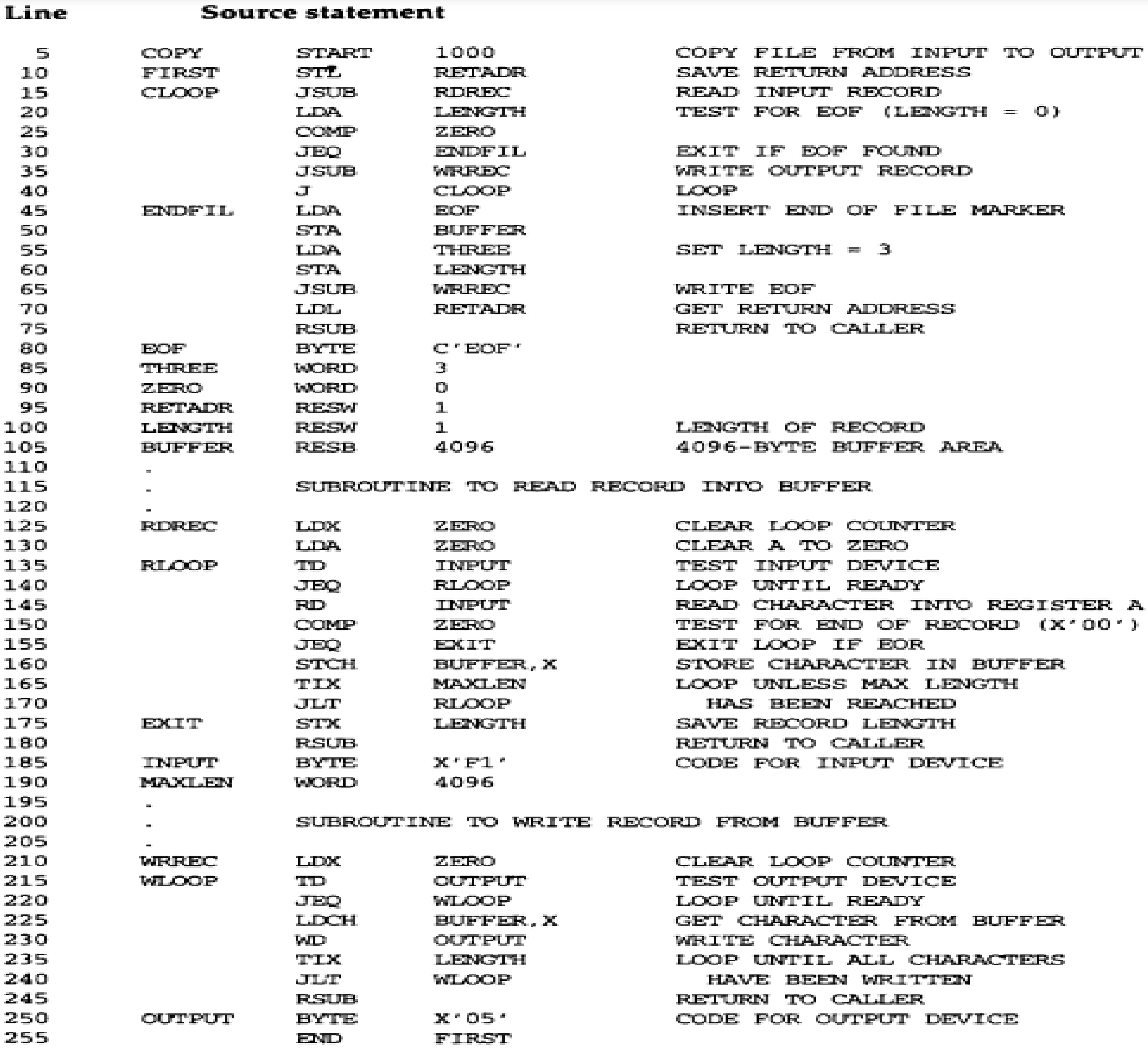
```
Pass 1:

begin
   read first input line
   if OPCODE = 'START' then
      begin
         save #[OPERAND] as starting address
         initialize LOCCTR to starting address
         write line to intermediate file
         read next input line
      end {if START}
   else
      initialize LOCCTR to 0
   while OPCODE ≠ 'END' do
      begin
         if this is not a comment line then
            begin
               if there is a symbol in the LABEL field then
                  begin
                     search SYMTAB for LABEL
                     if found then
                        set error flag (duplicate symbol)
                     else
                        insert (LABEL,LOCCTR) into SYMTAB
                  end {if symbol}
               search OPTAB for OPCODE
               if found then
                  add 3 {instruction length} to LOCCTR
               else if OPCODE = 'WORD' then
                  add 3 to LOCCTR
               else if OPCODE = 'RESW' then
                  add 3 * #[OPERAND] to LOCCTR
               else if OPCODE = 'RESB' then
                  add #[OPERAND] to LOCCTR
               else if OPCODE  = 'BYTE' then
                  begin
                     find length of constant in bytes
                     add length to LOCCTR
                  end {if BYTE}
               else
                  set error flag (invalid operation code)
            end {if not a comment}
         write line to intermediate file
         read next input line
      end {while not END}
   write last line to intermediate file
   save (LOCCTR - starting address) as program length
end {Pass 1}
```

Pass 2 Algorithm



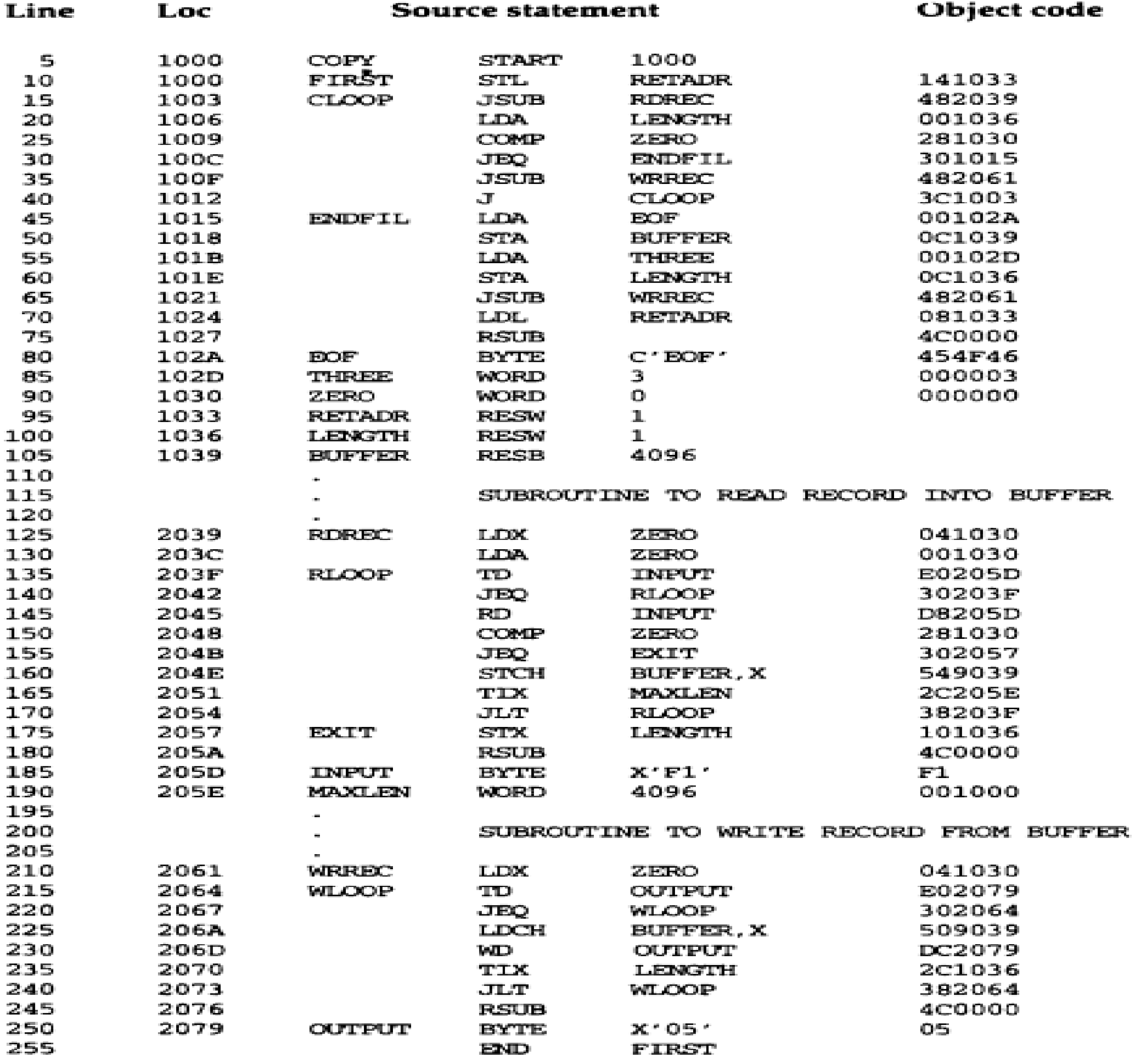
Pass 2:

```
begin
   read first input line {from intermediate file}
   if OPCODE = 'START' then
      begin
         write listing line
         read next input line
      end {if START}
   write Header record to object program
   initialize first Text record
   while OPCODE ≠ 'END' do
      begin
         if this is not a comment line then
            begin
               search OPTAB for OPCODE
               if found then
                  begin
                     if there is a symbol in OPERAND field then
                        begin
                           search SYMTAB for OPERAND
                           if found then
                              store symbol value as operand address
                           else
                              begin
                                 store 0 as operand address
                                 set error flag (undefined symbol)
                              end
                        end {if symbol}
                     else
                        store 0 as operand address
                     assemble the object code instruction
                  end {if opcode found}
               else if OPCODE = 'BYTE' or 'WORD' then
                  convert constant to object code
               if object code will not fit into the current Text record then
                  begin
                     write Text record to object program
                     initialize new Text record
                  end
               add object code to Text record
            end {if not comment}
         write listing line
         read next input line
      end {while not END}
   write last Text record to object program
   write End record to object program
   write last listing line
end {Pass 2}
```



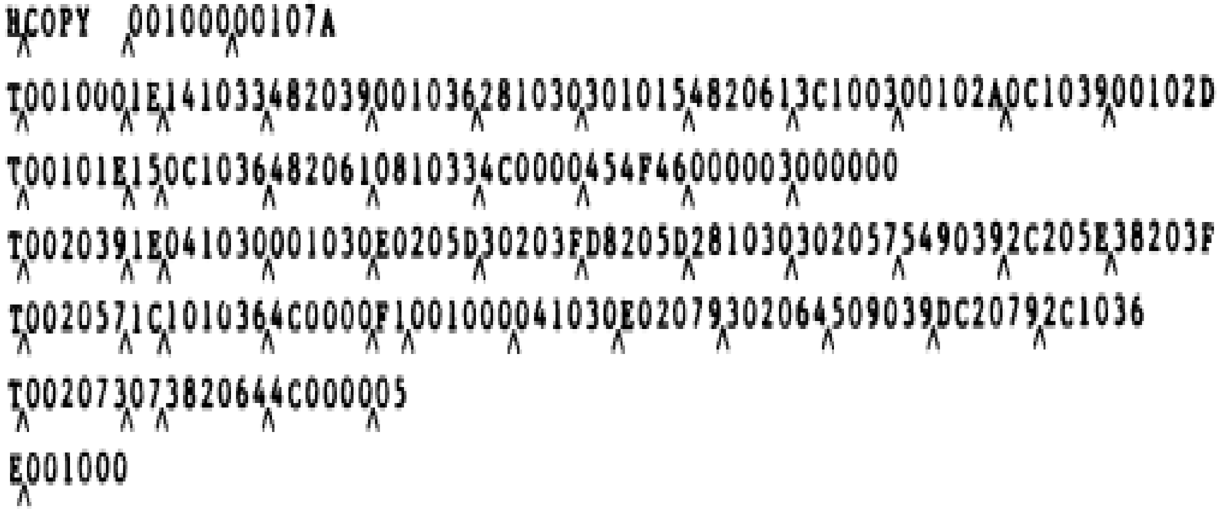
```
Line          Source statement

  5      COPY      START   1000            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    ZERO
 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     THREE           SET LENGTH = 3
 60                STA     LENGTH
 65                JSUB    WRREC           WRITE EOF
 70                LDL     RETADR          GET RETURN ADDRESS
 75                RSUB                    RETURN TO CALLER
 80      EOF       BYTE    C'EOF'
 85      THREE     WORD    3
 90      ZERO      WORD    0
 95      RETADR    RESW    1
100      LENGTH    RESW    1               LENGTH OF RECORD
105      BUFFER    RESB    4096            4096-BYTE BUFFER AREA
110      .
115      .         SUBROUTINE TO READ RECORD INTO BUFFER
120      .
125      RDREC     LDX     ZERO            CLEAR LOOP COUNTER
130                LDA     ZERO            CLEAR A TO ZERO
135      RLOOP     TD      INPUT           TEST INPUT DEVICE
140                JEQ     RLOOP           LOOP UNTIL READY
145                RD      INPUT           READ CHARACTER INTO REGISTER A
150                COMP    ZERO            TEST FOR END OF RECORD (X'00')
155                JEQ     EXIT            EXIT LOOP IF EOR
160                STCH    BUFFER,X        STORE CHARACTER IN BUFFER
165                TIX     MAXLEN          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    4096
195      .
200      .         SUBROUTINE TO WRITE RECORD FROM BUFFER
205      .
210      WRREC     LDX     ZERO            CLEAR LOOP COUNTER
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                TIX     LENGTH          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 2.1 Example of a SIC assembler language program.



```
Line     Loc         Source statement                         Object code

  5      1000    COPY      START    1000
 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     WRREC                     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     WRREC                     482061
 70      1024              LDL      RETADR                    081033
 75      1027              RSUB                               4C0000
 80      102A    EOF       BYTE     C'EOF'                    454F46
 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
110              .
115              .         SUBROUTINE TO READ RECORD INTO BUFFER
120              .
125      2039    RDREC     LDX      ZERO                      041030
130      203C              LDA      ZERO                      001030
135      203F    RLOOP     TD       INPUT                     E0205D
140      2042              JEQ      RLOOP                     30203F
145      2045              RD       INPUT                     D8205D
150      2048              COMP     ZERO                      281030
155      204B              JEQ      EXIT                      302057
160      204E              STCH     BUFFER,X                  549039
165      2051              TIX      MAXLEN                    2C205E
170      2054              JLT      RLOOP                     38203F
175      2057    EXIT      STX      LENGTH                    101036
180      205A              RSUB                               4C0000
185      205D    INPUT     BYTE     X'F1'                     F1
190      205E    MAXLEN    WORD     4096                      001000
195              .
200              .         SUBROUTINE TO WRITE RECORD FROM BUFFER
205              .
210      2061    WRREC     LDX      ZERO                      041030
215      2064    WLOOP     TD       OUTPUT                    E02079
220      2067              JEQ      WLOOP                     302064
225      206A              LDCH     BUFFER,X                  509039
230      206D              WD       OUTPUT                    DC2079
235      2070              TIX      LENGTH                    2C1036
240      2073              JLT      WLOOP                     382064
245      2076              RSUB                               4C0000
250      2079    OUTPUT    BYTE     X'05'                     05
255                        END      FIRST
```

Figure 2.2 Program from Fig. 2.1 with object code.



```
H^COPY  ^001000^00107A
T^001000^1E^141033^482039^001036^281030^301015^482061^3C1003^00102A^0C1039^00102D
T^00101E^15^0C1036^482061^081033^4C0000^454F46^000003^000000
T^002039^1E^041030^001030^E0205D^30203F^D8205D^281030^302057^549039^2C205E^38203F
T^002057^1C^101036^4C0000^F1^001000^041030^E02079^302064^509039^DC2079^2C1036
T^002073^07^382064^4C0000^05
E^001000
```

Figure 2.3 Object program corresponding to Fig. 2.2.

Header record:

Col. 1	H
Col. 2-7	Program name
Col. 8-13	Starting address of object program (hexadecimal)
Col. 14-19	Length of object program in bytes (hexadecimal)

Text record:

Col. 1	T
Col. 2-7	Starting address for object code in this record (hexadecimal)
Col. 8-9	Length of object code in this record in bytes (hexadecimal)
Col. 10 – 69	Object code, represented in hexadecimal

End record:

Col. 1	E
Col. 2-7	Address of first executable instruction in object program (hexadecimal)

Questions

1. Consider the statements in SIC program. Consider the program being assembled using a 2 pass assembler.

Line no	Location	Label	Opcode	Operand
10	1000	LENGTH	RESW	4
20	------------	NEW	WORD	3

What will be the address value assigned to the symbol NEW during pass 1? (5)

2. Write a sequence of instructions for SIC/ XE to find the average of three numbers, BETA, GAMMA and DELTA. (3)

3. Explain the format of the object program generated by a two-pass SIC Assembler, highlighting the contents of each record type. (3)

4. Explain the data structures used and their purposes in a two-pass assembler. (3)

5. What is the difference between the instructions LDA #5 and LDA FIVE? Explain how each instruction is executed. (3)

6. What are the uses of OPTAB and SYMTAB during the assembling process? Specify the uses of each during pass 1 and pass2 of a two pass assembler. (3)

Questions

7. What is meant by forward reference? How it is resolved by two pass assembler? (3)

8. Let A,B & C are arrays of 10 words each. Write a SIC/XE program to add the corresponding elements of A & B and store the result in C (6)

9. Write the sequence of instructions in SIC/XE to add two integer arrays S and T and store the contents to array Z. S and T each contains 10 integers (4)

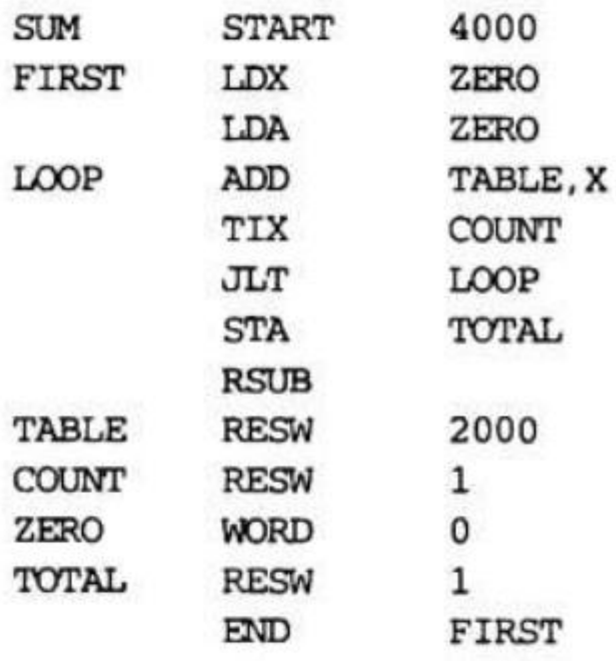
10. Write the sequence of instructions in SIC, to transfer the string ”UNIVERSITY” stored at location LOCA1 to LOCA2. (4)

11. Write a sequence of instructions for SIC/XE to set ALPHA equal to 4*BETA-9. Use immediate addressing modes for constants and assume ALPHA and BETA to be floating point numbers (4)

```
PGM     START   1000
        LDF     #4
        MULF    BETA
        SUBF    #9
        STF     ALPHA
BETA    BYTE    09 11 0A 23 24 56 ; 6 byte floating point number
ALPHA   RESB    6
        END     1000
```

Questions

12. Generate the assembled object program for the below SIC program. The machine code for the instructions used are: LDX – 04, LDA – 00, ADD – 18, TIX – 2C, STA – 0C, JLT – 38 and RSUB – 4C. Show the location counter value for each instruction (6)



```
SUM      START    4000
FIRST    LDX      ZERO
         LDA      ZERO
LOOP     ADD      TABLE,X
         TIX      COUNT
         JLT      LOOP
         STA      TOTAL
         RSUB
TABLE    RESW     2000
COUNT    RESW     1
ZERO     WORD     0
TOTAL    RESW     1
         END      FIRST
```

Loc	Label	Opcode	Operand	ObjectCode
	SUM	START	4000	
4000	FIRST	LDX	ZERO	045788
4003		LDA	ZERO	005788
4006	LOOP	ADD	TABLE,X	18C015
4009		TIX	COUNT	2C5785
400C		JLT	LOOP	384006
400F		STA	TOTAL	0C578B
4012		RSUB		4C0000
4015	TABLE	RESW	2000	
5785	COUNT	RESW	1	
5788	ZERO	WORD	0	000000
578B	TOTAL	RESW	1	
578E		END	FIRST	

Object Program

```
H^SUM^4000^78E
T^4000^15^045788^005788^18C015^2C5785^384006^0C578B^4C0000
T^5788^3^000000
E^4000
```