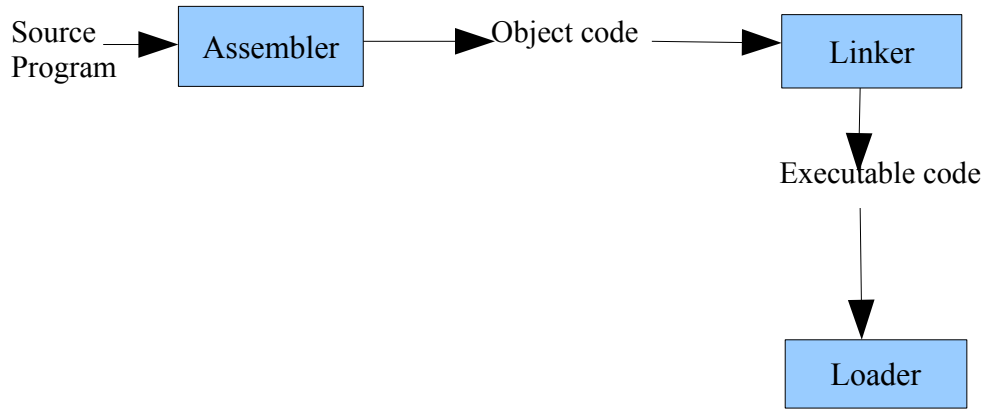
Unit 2
Assemblers - 1

1. Introduction:

The assembler is responsible for translating the assembly language into machine code. When the source language I s a symbolic representation for a numerical machine language, the translator is called as assembler & the source language is called as an assembly language.

The Role of Assembler:



Statement Format:

- Assembly language statements are written one per line.
- Each line consists of an assembly language program is split into four fields as follows:
  [LABEL] <OPCODE> <OPERAND> COMMENTS

Label: It is an identifier & optional field. Labels are used extensively in programs to reduce reliance upon programmers remembering where data or code is located.

Opcode: This field contains a mnemonic. Opcode stands for operation code which may also require additional information i.e. operands.

Operand: This field consists of additional information or data that the opcode requires. The operand is used to specify constants or labels, immediate data, Data contained in another accumulator or register, an address.

Assembler Directives:

- START : Specify name and starting address for the program.
- END : Indicate the end of the source program and specify the first executable instruction in the program.
- BYTE : Generate character or hexadecimal constant, occupying as many bytes as needed to represent the constant.
- WORD : Generate one-word integer constant.
- RESB : Reserve the indicated number of bytes for a data area.
- RESW : Reserve the indicated number of words for a data area.

2.1 Basic Assembler Function :

1. A Simple SIC Assembler.
2. Assembler Algorithm & Data Structures.

- Figure 2.1 shows an assembler language program for the basic version of SIC. The program contains a main routine that reads records from an input device and copies them to an output device.
- This main routine calls subroutine RDREC to read a record into a buffer and subroutine WRREC to write the record from the buffer to the output device.
- Each subroutine must transfer the record one character at a time because the only I/O instructions available are RD and WD.
- If a record is longer than the length of the buffer, only the first 4096 bytes are copied.
- The end of the file to be copied is indicated by a zero –length record.
- When the end of the file is detected, the program writes EOF on the output device and terminates by executing an RSUB instruction.

Line	Source Statement		
5	COPY	START	1000
10	FIRST	STL	RETADR
15	CLOOP	JSUB	RDREC
20		LDA	LENGTH
25		COMP	ZERO
30		JEQ	ENDFIL
35		JSUB	WRREC
40		J	CLOOP
45	ENDFIL	LDA	EOF
50		STA	BUFFER
55		LDA	THREE
60		STA	LENGTH
65		JSUB	WRREC
70		LDL	RETADR
75		RSUB	
80	EOF	BYTE	C'EOF'
85	THREE	WORD	3
90	ZERO	WORD	0
95	RETADR	RESW	1
100	LENGTH	RESW	1
105	BUFFER	RESB	4096
110 115 120	. . Subroutine to read record into BUFFER .		
125	RDREC	LDX	ZERO
130		LDA	ZERO
135	RLOOP	TD	INPUT
140		JEQ	RLOOP
145		RD	INPUT
150		COMP	ZERO
155		JEQ	EXIT

160		STCH	BUFFER,X
165		TIX	MAXLEN
170		JLT	RLOOP
175	EXIT	STX	LENGTH
180		RSUB	
185	INPUT	BYTE	X'F1'
190	MAXLEN	WORD	4096
195 200 205	. . Subroutine to write record from BUFFER .		
210	WRREC	LDX	ZERO
215	WLOOP	TD	OUTPUT
220		JEQ	WLOOP
225		LDCH	BUFFER,X
230		WD	OUTPUT
235		TIX	LENGTH
240		JLT	WLOOP
245		RSUB	
250	OUTPUT	BYTE	X'05'
255		END	FIRST

Figure 2.1

2.1.1 A Simple SIC Assembler:

Figure 2.2 shows the same program as in Figure 2.1, with the generated object code for each statement. The column headed Loc gives the machine address (Hexadecimal) for each part of the assembled program. Here the starting address of program is 1000. Each instruction in SIC takes 3 Bytes of memory location.

Assembler Functions:

The translation of source program to object code requires us to accomplish the following instructions.

- Convert mnemonic operation codes to their machine language equivalents.
  Ex: translate STL to 14 (Line 10).
- Convert symbolic operands to their equivalent machine addresses.
  Ex: translate RETADR to 1033(Line 10).
- Build the machine instruction in the proper format.
- Convert the data constants to internal machine representations.
- Write the object program and the assembly listing.

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	00 1036
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	0 81033
75	1027		RSUB		4C0000
80	102A	EOF	BYTE	C'EOF'	454F46
85	102D	THREE	WORD	3	0 00003
90	1030	ZERO	WORD	0	0 00000
95	1033	RETADR	RESW	1	----
100	1036	LENGTH	RESW	1	----
105	1039	BUFFER	RESB	4096	-----
110 115 120		. . Subroutine to read record into BUFFER .			
125	2039	RDREC	LDX	ZERO	0 41030
130	203C		LDA	ZERO	00 1030
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	00 1000
195 200 205		. . Subroutine to write record from BUFFER .			
210	2061	WRREC	LDX	ZERO	0 41030
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'	0 5
255			END	FIRST	

Figure 2.2

- The translation of addresses presents a problem. Consider the statement:
  10 1000 FIRST STL RETADR 141033
- This instruction contains a forward reference i.e. a reference to a label(RETADR) that is defined later in the program.
- If we attempt to translate the program line by line, we will be unable to process such instruction because we don't know the address that will be assigned to this label.
- Because of this most assemblers make two passes over the source program.
- If the statement contain the assembler directives then these are not translated into machine instruction such START, END, RESB & RESW.
- The assembler must write the generated object code into the object program.

Structure of Object Program:

The object program format contains 3 types of records.

1. **Header Record :** It contains the program name, starting address & length.
   Syntax:
   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(Hexadecimal)

2. **Text Record:** It contains the translated instructions & data of the program, together with an indication of the addresses where these are to be loaded.
   Text record:
   Col. 1 : T
   Col. 2-7 : Starting address for object code in this record
   Col. 8-9 : Length of object code in this record in bytes
   Col. 10-69 : Object code, represented in hexadecimal(2 columns per byte)

3. **End Record:** It marks the end of the object program & specifies the address in the program where execution is to begin.
   Col.1 : E
   Col. 2-7 : Address of first executable instruction in object program.

Figure 2.3 shows the object program corresponding to Figure 2.2. In this figure ^ is used to separate the fields.

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 ^302064^509036^DC2079^2C1036

T^002073^07^382064^4C0000^05

E^001000

Functions of the Two Passes of Simple Assembler :

Pass 1

1. Assign addresses to all statements in the program.
2. Save the values assigned to all labels for use in pass 2.
3. Perform some processing of assembler directives. (This includes determining length of the data areas defined by BYTE, RESW etc.).

Pass 2

1. Assemble instructions (translating operation codes & looking up addresses).
2. Generate the data values defined by BYTE, WORD, etc.
3. Perform processing of assembler directives not done during pass 1.
4. Write the object program and the assembly listing.

2.1.2 Assembler Algorithm & Data Structures:

Assembler uses two major internal data structures: the Operation Code Table(OPTAB) and the Symbol Table(SYMTAB).

- **OPTAB** is used to look up mnemonic operation codes and translate them to their machine language equivalents. OPTAB must contain at least mnemonic operation codes & its machine language equivalent. In more complex assemblers, it also contains information about instruction format & length. During pass 1, OPTAB is used to look up and validate operation codes in the source program. In pass 2, it is used to translate the operation codes to machine language. OPTAB is usually organized as a hash table, with mnemonic operation code as a key. The hash table organization is particularly appropriate, since it provides fast retrieval with a minimum of searching.

- **SYMTAB** is used to store values(addresses) assigned to labels. It includes the name & address for each label in the source program, together with flags to indicate error conditions. It may also contain other info' about data area or instruction labeled. During Pass 1, labels are entered into SYMTAB along with their assigned addresses. During Pass 2, symbols used as operands are looked up in SYMTAB to obtain the addresses to be inserted in the assembler instructions. SYMTAB is usually organized as a hash table for efficiency of insertion & retrieval.

- **Location Counter LOCCTR** is a variable that is used to help in the assignment of addresses. LOCCTR is 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 to LOCCTR.

Algorithm of PASS 1 of TWO-PASS ASSEMBLER :

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 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 SYSTAB
        end (if symbol)
        search OPTAB for OPCODE
        if found then
        add 3 (instr 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
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}

- The algorithm scans the first statement START and saves the operand field (the address) as the starting address of the program. Initializes the LOCCTR value to this address. This line is written to the intermediate line.
- If no operand is mentioned the LOCCTR is initialized to zero. If a label is encountered, the symbol has to be entered in the symbol table along with its associated address value.
- If the symbol already exists that indicates an entry of the same symbol already exists. So an error flag is set indicating a duplication of the symbol.
- It next checks for the mnemonic code, it searches for this code in the OPTAB. If found then the length of the instruction is added to the LOCCTR to make it point to the next instruction.
- If the opcode is the directive WORD it adds a value 3 to the LOCCTR. If it is RESW, it needs to add the number of data word to the LOCCTR. If it is BYTE it adds a value one to the LOCCTR, if RESB it adds number of bytes.
- If it is END directive then it is the end of the program it finds the length of the program by evaluating current LOCCTR – the starting address mentioned in the operand field of the END directive. Each processed line is written to the intermediate file.

Algorithm of PASS 2 of TWO-PASS ASSEMBLER :

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 OPERAND

if found then

begin

if there is a symbol in OPERAND field then

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}
read last Text record to object program
write End record to object program
write last listing line
End {pass 2}
```

- The algorithm scans the first input line from intermediate file & if it is START then as it is written & reads the next input line.
- First line will be written to object program in form of Header record format.
- Initializes the Text record. Until the OPCODE != END all the statements object code will be written to Text Record by searching OPTAB & SYMTAB for operand.
- If it is not possible to fit the object code into the current Text record then it initializes another Text record to which remaining object code will be written.
- After writing the last Text record to the object program, then write the End record to the object program.

2.2 Machine-Dependent Assembler Features:

- Instruction Formats & Addressing Modes
- Program Relocation

To study the machine-Dependent features we must consider the example of SIC/XE.

Line	Source Statement		
5	COPY	START	0
10	FIRST	STL	RETADR
12		LDB	#LENGTH
13		BASE	LENGTH
15	CLOOP	+JSUB	RDREC
20		LDA	LENGTH
25		COMP	#0
30		JEQ	ENDFIL
35		+JSUB	WRREC
40		J	CLOOP
45	ENDFIL	LDA	EOF
50		STA	BUFFER
55		LDA	#3
60		STA	LENGTH
65		+JSUB	WRREC
70		J	@RETADR
80	EOF	BYTE	C'EOF'
95	RETADR	RESW	1
100	LENGTH	RESW	1
105	BUFFER	RESB	4096
110 115 120	. . Subroutine to read record into BUFFER .		
125	RDREC	CLEAR	X
130		CLEAR	A
132		CLEAR	S
133		+LDT	#4096
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'
195 200 205	. . Subroutine to write record from BUFFER .		
210	WRREC	CLEAR	X
2012		LDT	LENGTH
215	WLOOP	TD	OUTPUT
220		JEQ	WLOOP
225		LDCH	BUFFER,X
230		WD	OUTPUT
235		TIXR	T
240		JLT	WLOOP
245		RSUB	
250	OUTPUT	BYTE	X'05'
255		END	FIRST

2.2.1. Instruction Formats & Addressing Modes :

- Translation of register-to-register instructions such as CLEAR,COMPR, TIXR supports Instruction Format 2 (2 bytes). The assembler must simply convert the mnemonic operation code to machine language using OPTAB & change each register mnemonic with its numeric equivalent.
  Ex: CLEAR X → B410
- Most of the register-to-memory instructions are assembled using either program-counter relative or base relative addressing. The assembler must in either case, calculate a displacement to be assembled as part of the object instruction. Then correct target address is computed by adding the displacement value with the contents of PC or B register.
- If neither program-counter nor base relative addressing can be used then the 4-byte extended instruction format(Format 4) must be used. It contains a 20 bit address field, which is large enough to contain the full memory address. In this case, no displacement to be computed. In the program extended format is specified by using the prefix +.
- If the program contains the backword jump instruction, then displacement will be -ve value which is to be represented in 2's complement format.
- If the program contains BASE assembler directive, then it informs to the assembler that the operand value is available in Base register (Base relative addressing). If the value of operand is unable to fit into the PC register (-2048 to +2047), then also Base relative addressing must be used in the program.
- If immediate addressing is used in the program, then the operand value is directly converted to its internal representation & insert it into the object code.

Line	Loc	Source Statement			Object Code
5	0 000	COPY	START	0	
10	0 000	FIRST	STL	RETADR	17202D
12	0 003		LDB	#LENGTH	69202D
13			BASE	LENGTH	
15	0 006	CLOOP	+JSUB	RDREC	4B101036
20	0 00A		LDA	LENGTH	0 32026
25	0 00D		COMP	#0	290000
30	0 010		JEQ	ENDFIL	332007
35	0 013		+JSUB	WRREC	4B10105D
40	0 017		J	CLOOP	3F2FEC
45	0 01A	ENDFIL	LDA	EOF	0 32010
50	0 01D		STA	BUFFER	0F2016
55	0 020		LDA	#3	0 10003
60	0 023		STA	LENGTH	0F200D
65	0 026		+JSUB	WRREC	4B10105D
70	0 02A		J	@RETADR	3E2003
80	0 02D	EOF	BYTE	C'EOF'	454F46
95	0 030	RETADR	RESW	1	
100	0 033	LENGTH	RESW	1	
105	0 036	BUFFER	RESB	4096	
110 115 120		. . Subroutine to read record into BUFFER .			
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
195 200 205		. . Subroutine to write record from BUFFER .			

210	105D	WRREC	CLEAR	X	B410
2012	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'	0 5
255			END	FIRST	

2.2.2 Program Relocation:

Definition: The program relocation is one which modifies the object program so that it can be loaded at an address different from the location originally specified.

- It is often desirable to have more than one program at a time sharing the memory & other resources of the machine. It is desirable to be able to load a program into memory wherever there is a room for it. In such a situation the actual starting address of the program is not known until load time.
- If the program of SIC is considered where it makes use of absolute programming, then it must be loaded at address specified in the program in order to execute it properly. But there are parts of the program that remain same(constant 3) regardless of where the program is loaded.
- There are some instructions in the program that need modification in the address based on the starting address of the program.
- Figure 2.7(a) shows that the program is loaded at starting address 0000. The JSUB instruction is loaded at 0006 & the address field of this instruction labeled RDREC contains 01036. If we load the same program at starting address 5000. The address of the instruction labeled RDREC is at 6036 shown in Figure 2.7(b). Thus the JSUB instruction must be modified to contain this new address. If we load the same program at starting address 7420. The address of the instruction labeled RDREC is at 8456 shown in Figure 2.7(c). Thus the JSUB instruction must be modified to contain this new address.

0 000	1					
0 006	4B101036	+JSUB RDREC				
1036	B410	CLEAR X				
1076						
		5000				
		5006	4B106036	+JSUB RDREC		
		6036	B410	CLEAR X		
		6076				
				7420		
				7426	4B108456	+JSUB RDREC
				8456	B410	CLEAR X
				8496		

(a) (b) (c)

Figure 2.7

The relocation program can be solved in the following way:

1. When the assembler generates the object code for the JSUB instruction, it will insert the address of RDREC relative to the start of the program.
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.

The command for the loader must also be a part of the object program. We can accomplish this with a Modification record having the following format :

Modification Record :

Col. 1 : M

Col. 2-7 : Starting location of the address field to be modified, relative to the beginning of the program(hexadecimal).

Col. 8-9 : Length of the address field to be modified, in half-bytes(Hexadecimal).

For JSUB instruction we are using an example, the Modification record would be

M^000007^05

Examples:

1. Generate the object code for each statement and write the object programs for the following SIC/XE program.

Given that CLEAR=B4, LDA=00, LDB=68, ADD=18, TIX=2C, JLT=38, STA=0C

SUM	START	0
FIRST	CLEAR	X

	LDA	#0
	+LDB	#TOTAL
	BASE	TOTAL
LOOP	ADD	TABLE, X
	TIX	COUNT
	JLT	LOOP
	STA	TOTAL
COUNT	RESW	1
TABLE	RESW	2000
TOTAL	RESW	1
	END	START

Ans:

Loc	Source Statements			Object Code
0 000	SUM	START	0	
0 000	FIRST	CLEAR	X	B410
0 002		LDA	#0	0 10000
0 005		+LDB	#TOTAL	69101788
		BASE	TOTAL	
0 009	LOOP	ADD	TABLE, X	1BA00C
0 00C		TIX	COUNT	2F2006
0 00F		JLT	LOOP	3B2FF7
0 012		STA	TOTAL	F40000
0 015	COUNT	RESW	1	
0 018	TABLE	RESW	2000	
1788	TOTAL	RESW	1	
		END	START	

Length of the program = 178B-0000=178B

1. FIRST CLEAR X

8	4	4
Op	r1	r2
B4	1	0

Format 2

2. Format 3(3 bytes) LDA #0

6	1	1	1	1	1	1	12
op	n	I	X	b	P	e	Disp
0000 00	0	1	0	0	0	0	0 0 0

0 1 0 0 0 0

Immediate addressing

3. Format 4(4 bytes) +LDB #TOTAL

6	1	1	1	1	1	1	20
op	n	i	X	b	P	e	Disp
0110 10	0	1	0	0	0	1	0 1788

6 9 1 01788

Format 4 with immediate addressing hence no program-counter or base-relative addressing. TOTAL is available at 1788.

4. Format 3(3 bytes) LOOP ADD TABLE, X

6	1	1	1	1	1	1	12
op	n	I	X	b	P	e	Disp
0001 10	1	1	1	0	1	0	00C

1 B A 00C

This instruction is Program-counter with Indexed relative addressing. PC contains the address 000C. TABLE is at address 0018. Address : (0018-000C=000C).

5. Format 3(3 bytes) TIX COUNT

6	1	1	1	1	1	1	12
op	n	I	X	b	P	e	Disp
0010 11	1	1	0	0	1	0	0 0 6

2 F 2 0 0 6

This instruction is Program-counter relative addressing. PC contains the address 000F. COUNT is at address 0015.

Address : 0015 – 000F= 0006

6. Format 3(3 bytes) JLT LOOP

6	1	1	1	1	1	1	12
op	n	I	X	b	P	e	Disp
0011 10	1	1	0	0	1	0	FF7

3 B 2 FF7

(PC)=0012(in hex) = 18

LOOP=0009(in hex)=9 (9-18= -9) hexadecimal subtraction.

7. Format 3(3 bytes) STA TOTAL

6	1	1	1	1	1	1	12
op	n	I	X	b	P	e	Disp
0000 11	1	1	0	1	0	0	0000 0000 0000

As TOTAL is declared as BASE. It is Base-relative addressing. For Store instruction in base-relative addressing the content of BASE of register 1788. STA TOTAL is also contain 1788. 1788-1788=0.

8. RSUB

6	1	1	1	1	1	1	12
op	n	I	X	b	P	e	disp
0100 11	1	1	0	0	0	0	0 00

4 F 0 0 0 0

RSUB is format 3 instruction which do not take any operand value.

Object Program :

H^SUM ^000000^00178B

T^000000^15^B410^010000^690101788^1BA00C^2F2006^2B2FF7^4F0000

M^000006^05

E^000000

2. Generate the object code for each statement in the following SIC/XE program.
Assume LDX =04, LDA=00,LDB=68, ADD=18, TIX=2C, JLT=38, STA=0C,RSUB=4C

SUM	START	0
FIRST	LDX	#0
	LDA	#0
	+LDB	#TABLE2
	BASE	TABLE2
LOOP	ADD	TABLE, X
	ADD	TABLE2, X

	TIX	COUNT
	JLT	LOOP
	+STA	TOTAL
	RSUB	
COUNT	RESW	1
TABLE	RESW	2000
TABLE2	RESW	2000
TOTAL	RESW	1
	END	START

Solution :

Loc	Source Statements			Object Code
0 000	SUM	START	0	
0 000	FIRST	LDX	#0	0 50000
0 003		LDA	#0	0 10000
0 006		+LDB	#TABLE2	69101790
		BASE	TABLE2	
0 00A	LOOP	ADD	TABLE, X	1BA013
0 00D		ADD	TABLE2, X	1BC000
0 010		TIX	COUNT	2F200A
0 013		JLT	LOOP	3B2FF4
0 016		+STA	TOTAL	0F102F00
0 01A		RSUB		4F0000
0 01D	COUNT	RESW	1	
0 020	TABLE	RESW	2000	
1790	TABLE2	RESW	2000	
2F00	TOTAL	RESW	1	
		END	START	

1. Format 3(3 bytes) FIRST LDX #0

6	1	1	1	1	1	1	12
op	n	I	X	b	P	e	Disp
0000 01	0	1	0	0	0	0	0000 0000 0000

0 5 0 0 0 0

Immediate Addressing

2. Format 3(3 bytes) LDA #0

6	1	1	1	1	1	1	12
op	n	I	X	b	P	e	Disp
0000 00	0	1	0	0	0	0	0 0 0

0 1 0 0 0 0

Immediate Addressing

3. Format 4(4 bytes) +LDB #TABLE2

6	1	1	1	1	1	1	20
op	n	i	X	b	P	e	Disp
0110 10	0	1	0	0	0	1	0 1790

6 9 1 0 1790

Format 4 with immediate addressing hence no program-counter or base-relative addressing.

4. Format 3(3 bytes) LOOP ADD TABLE, X

6	1	1	1	1	1	1	12
op	n	I	X	b	P	e	Disp
0001 10	1	1	1	0	1	0	00D

1 B A 00D

This instruction is Program-counter with Indexed relative addressing. PC contains the address 000D. TABLE is at address 0020. (0020-000D=000D).

5. Format 3(3 bytes) LOOP ADD TABLE2, X

6	1	1	1	1	1	1	12
op	n	I	X	b	P	e	Disp
0001 10	1	1	1	0	1	0	0 00

1 B A 0 0 0

As TABLE2 is declared as BASE. It is Base-relative addressing. For ADD instruction in base-relative addressing the content of BASE of register 1790. ADD TABLE2,X is also contain 1790. 1790-1790=0.

6. Format 3(3 bytes) TIX COUNT

6	1	1	1	1	1	1	12
op	n	I	X	b	P	e	Disp

0010 11	1	1	0	0	1	0	0 0 A

2 F 2 0 0 A

This instruction is Program-counter relative addressing. PC contains the address 0013. COUNT is at address 001D. Address : 001D – 0013= 000A

7. Format 3(3 bytes) JLT LOOP

6	1	1	1	1	1	1	12
op	n	I	X	b	P	e	Disp
0011 10	1	1	0	0	1	0	FF4

3 B 2 FF4

PC = 0016(hex)

LOOP = 000A(hex) Hexadecimal subtraction = FF4

8. Format 4(4 bytes) +STA TOTAL

6	1	1	1	1	1	1	20
op	n	I	X	b	P	e	Address
0000 11	1	1	0	0	0	1	0 2F00

0 F 1 02F00

This instruction is format 4. Hence no program-counter nor base-relative addressing.

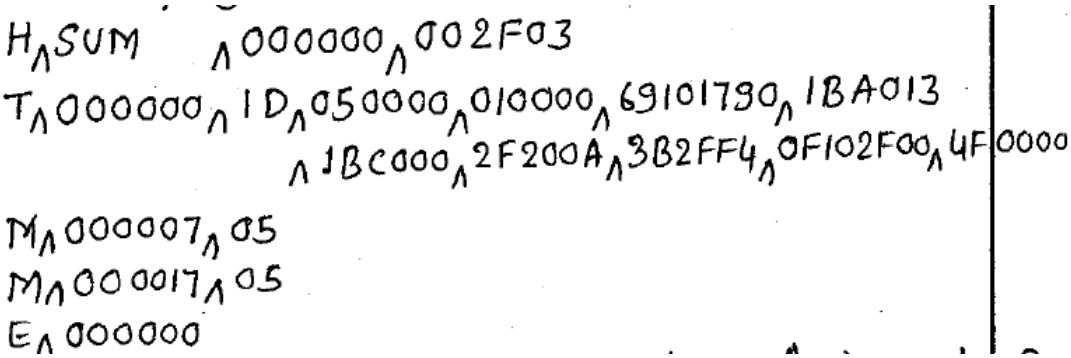
9. RSUB

6	1	1	1	1	1	1	12
op	n	I	X	b	P	e	disp
0100 11	1	1	0	0	0	0	0 00

4 F 0 0 0 0

RSUB is format 3 instruction which do not take any operand value.

Object Program :



3.

Generate the complete object program for the following assembly level program with the symbol table. Assume : **(14 Marks)**

CLEAR = B4 LDT = 74 TD = EO JEQ = 30
TIXR = B8 JLT = 38 RSUB = 4C LDCH = 50
WD = DC X = 1 T = 5

```
WRREC   START   105D
        CLEAR   X
        LDT     LENGTH
WLOOP   TD      OUTPUT
        JEQ     WLOOP
        LDCH    BUFFER, X
        WD      OUTPUT
        TIXR    T
        JLT     WLOOP
        RSUB
OUTPUT  BYTE    X '05'
BUFFER  RESB    400
LENGTH  RESB    2
        END     WRREC
```

Solution:

Loc	Source Statements			Object Code
105D	WRREC	START	105D	
105D		CLEAR	X	B410
105F		LDT	LENGTH	7321A5
1062	WLOOP	TD	OUTPUT	E32011
1065		JEQ	WLOOP	332FFA
1068		LDCH	BUFFER,X	53A00C
106B		WD	OUTPUT	DF2008
106E		TIXR	T	B850
1070		JLT	WLOOP	3B2FEF
1073		RSUB		4F0000
1076	OUTPUT	BYTE	X'05'	0 5
1077	BUFFER	RESB	400	
1207	LENGTH	RESB	2	
		END	WRREC	

1. Format 2(2 bytes) CLEAR X

8	4	4
Op	r1	r2
B4	1	0

2. Format 3(3 bytes) LDT LENGTH

6	1	1	1	1	1	1	12
op	n	I	X	b	P	e	Disp
0111 01	1	1	0	0	1	0	1A5

7 3 2 1A5

This instruction is Program-counter relative addressing. PC contains the address 1062. LENGTH is at address 1207. Address : 1207 – 1062= 1A5.

3. Format 3(3 bytes) WLOOP TD OUTPUT

6	1	1	1	1	1	1	12
op	n	I	X	b	P	e	Disp
1110 00	1	1	0	0	1	0	0 11

E 3 2 011

This instruction is Program-counter relative addressing. PC contains the address 1065. OUTPUT is at address 1076. Address : 1076 – 1065= 0011.

4. Format 3(3 bytes) JEQ WLOOP

6	1	1	1	1	1	1	12
op	n	I	X	b	P	e	Disp
0011 00	1	1	0	0	1	0	FFA

3 3 2 FFA

(PC)=1068(in hex)

WLOOP=1062(in hex) (1062-1068=-6) -ve number is implemented in 2's complement

2's complement= 0000 0000 0110

1's complement= 1111 1111 1001

+ 1

1111 1111 1010

F F A

5. Format 3(3 bytes) LDCH BUFFER, X

6	1	1	1	1	1	1	12
op	n	I	X	b	P	e	Disp
0101 00	1	1	1	0	1	0	00C

5 3 A 00C

This instruction is Program-counter with Indexed relative addressing. PC contains the address 106B. BUFFER is at address 1077. (1077-106B=00C).

6. Format 3(3 bytes) WD OUTPUT

6	1	1	1	1	1	1	12
op	n	I	X	b	P	e	Disp
1101 11	1	1	0	0	1	0	0 08
D	F			2			008

This instruction is Program-counter relative addressing. PC contains the address 106E. OUTPUT is at address 1076. Address : 1076 – 106E= 0008.

7. Format 2(2 bytes) TIXR T

8	4	4
Op	r1	r2
B8	5	0

8. Format 3(3 bytes) JLT WLOOP

6	1	1	1	1	1	1	12
op	n	I	X	b	P	e	Disp
0011 10	1	1	0	0	1	0	FEF
3	B			2			FEF

(PC)=1073(in hex)

WLOOP=1062(in hex) (1062-1073=-11) -ve number is implemented in 2's complement

2's complement= 0000 0001 0001

1's complement= 1111 1110 1110

+ 1

1111 1110 1111

F E F