11/9/17 monday

MODULE II

ASSEMBLER

Functions

- Converts operation code to machine code

OPTAB { LDA → 00 (m/c code) ALPHA

- Converts symbolic operand to machine address.
- Generation of object code for each assembly instruction.

* Two types of assembler

Single pass
→ Generates object code in 1 pass

Two pass
→ Generates object code in 2 pass

* Forward reference :- Reference to label defined later in program

eg:-
```
1003   CLOOP   LDA   RETADR
                .
                .
                .
1033   RETADR  RESB  1
```

This is the problem with single pass assembler resolved using two pass assembler.

⇒ Two pass assembler, SYMTAB. In first pass it stores all labels and its corresponding address in SYMTAB & in second pass generates op-code

Data structures

1) OPTAB (Operation code Table)

opcode	machine code	Format	length
LDA	00	3	3

→ While generating machine object code of opcode, its machine code is required, this is obtained through OPTAB.

→ construction during pass I

2) SYMTAB (Symbol Table)

→ Relative - R
Absolute - A

SYMBOL	TYPE	ADDRESS
RETADR	R	1033
CLOOP	A	2033

→ Used to store symbols and its corresponding address.

→ construction during pass I

3) LOCCTR (Location Counter)

→ single data structure used to store the address of each instruction.

8/9/11

HAND ASSEMBLY

LOC	LENGTH	LABEL	MNEMONIC	OPERAND	OBJECT CODE
0000		SUM	START	0	
0000	3		LDX	#0	050000
0003	3		LDA	#0	010000
0006	4		+LDB	#TABLE2	69100000
			BASE	TABLE2	
000A	3	LOOP	ADD	TABLE,X	1BA013
000D	3		ADD	TABLE2,X	1BC000
0010	3		TIX	COUNT	2F200A
0013	3		JLT	LOOP	3B2FF4
0016	4		+STA	TOTAL	0F102F00
001A	3		RSUB		4C0000
001D	3	COUNT	RESW	1	
0020	1770	TABLE	RESW	2000	
1790	1770	TABLE2	RESW	2000	
2FC0	3	TOTAL	RESW	1	
2FC3			END	FIRST	

★ LDX #0

n i x b p e

0000	01	0	1	0	0	0	0	0000	0000	0000
0		5		0		0		0		0

★ LDA #0

0000	00	0	1	0	0	0	0	0000	0000	0000
0		1		0		0		0		0

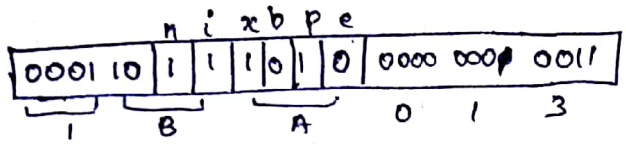
★ +LDB #TABLE2

n i x b p e

0110	10	0	1	0	0	0	1	0000	0000	0000	0000	0000
6		9		1		0		0		0		0

14/9/17

ADD TABLE, X

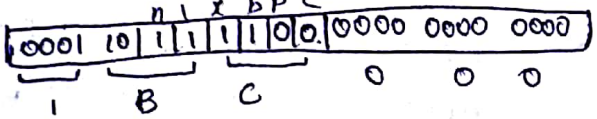


disp = TA − PC

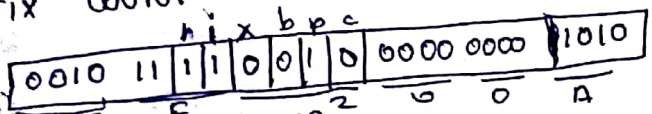
= 0020 − 0007

= 0013

⇒ ADD TABLE2, X



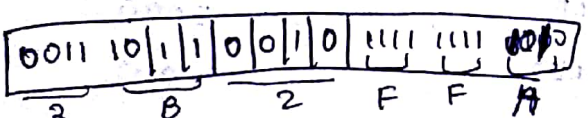
⇒ TIX COUNT



disp = 0010 − 0013

= 000A

⇒ JLT LOOP

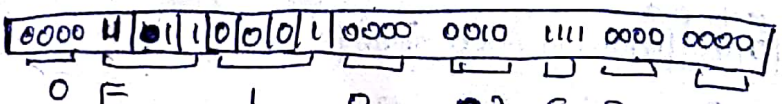


disp = 000A − 0016

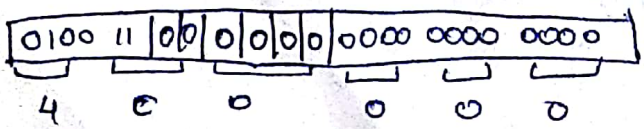
= −0012

2's complement of −0012 = FFEA

⇒ STA TOTAL



⇒ RSUB



0013 −
TA − B
TA − P.C
disp = 1790 − 0010
(1780) . 6

b=0 p=1

b=1 p=0

b=0 p=1 1790 − 1790

b=0 p=1

0010 −
0013
000A

0016 −
000A
0006

F

14/9/17

Assembler Output Format

Object program

1) Header (1)
2) Text Record (more than 1)
3) End Record (1)

a) Header Record

col 1 - H
col 2-7 - Program name
col 8-13 - Starting address (Hex)
col 14-19 - Length of object prgm

eg:-

1000	COPY	START	1000	
1000	FIRST	STL	RETADR	141033
1003	CLOOP	JSUB	RDREC	482039
		:		
		LDA	THREE	001020
101E		STA	LENGTH	0C1036
1021		JSUB	WRREC	482061
		:		
2079		END	FIRST	

H ^ COPY ^ 001000 ^ 000079

- COPY → pgm name
- 001000 → starting addr
- 000079 → length of pgm

(b) 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 (in hex)

For the above eg:-

T^001000^1E^141033, 482039^001036^281030^301015^

482061^3C1003^00102A, 0C1039^00102D

T^00101E^15^0C1036^482061^

⋮

(c) End Record

col 1 : E

col 2-17 Address of 1st executable instruction

eg:- E^001000