Machine Language

TABLE 6-2 Binary Program to Add Two Numbers

Location	Instruction code
0	0010 0000 0000 0100
1	0001 0000 0000 0101
10	0011 0000 0000 0110
11	0111 0000 0000 0001
100	0000 0000 0101 0011
101	1111 1111 1110 1001
110	0000 0000 0000 0000

Very difficult to understand !

Machine Language

TABLE 6-3 Hexadecimal Program to Add Two Numbers

Location	Instruction
000	2004
001	1005
002	3006
003	7001
004	0053
005	FFE9
006	0000

Still very difficult to understand !

Symbolic Language (Mnemonic)

TABLE 6-4 Program with Symbolic Operation Codes

Location	Instruction	Comments
000	LDA 004	Load first operand into *AC*
001	ADD 005	Add second operand to *AC*
002	STA 006	Store sum in location 006
003	HLT	Halt computer
004	0053	First operand
005	FFE9	Second operand (negative)
006	0000	Store sum here

Assembly Language

TABLE 6-5 Assembly Language Program to Add Two Numbers

```
        ORG 0       /Origin of program is location 0
        LDA A       /Load operand from location A
        ADD B       /Add operand from location B
        STA C       /Store sum in location C
        HLT         /Halt computer
A,      DEC 83      /Decimal operand
B,      DEC -23     /Decimal operand
C,      DEC 0       /Sum stored in location C
        END         /End of symbolic program
```


Simple Example

TABLE 6-8 Assembly Language Program to Subtract Two Numbers

```
        ORG 100     /Origin of program is location 100
        LDA SUB     /Load subtrahend to AC
        CMA         /Complement AC
        INC         /Increment AC
        ADD MIN     /Add minuend to AC
        STA DIF     /Store difference
        HLT         /Halt computer
MIN,    DEC 83      /Minuend
SUB,    DEC -23     /Subtrahend
DIF,    HEX 0       /Difference stored here
        END         /End of symbolic program
```


Translating to Binary

TABLE 6-9 Listing of Translated Program of Table 6-8

Hexadecimal code			
Location	Content	Symbolic program	
			ORG 100
100	2107		LDA SUB
101	7200		CMA
102	7020		INC
103	1106		ADD MIN
104	3108		STA DIF
105	7001		HLT
106	0053	MIN,	DEC 83
107	FFE9	SUB,	DEC -23
108	0000	DIF,	HEX 0
			END

Representation of Sybolic Program in Memory

TABLE 6-11 Computer Representation of the Line of Code: PL3, LDA SUB I

Memory word	Symbol	Hexadecimal code	Binary representation
1	P L	50 4C	0101 0000 0100 1100
2	3 ,	33 2C	0011 0011 0010 1100
3	L D	4C 44	0100 1100 0100 0100
4	A	41 20	0100 0001 0010 0000
5	S U	53 55	0101 0011 0101 0101
6	B	42 20	0100 0010 0010 0000
7	I CR	49 0D	0100 1001 0000 1101

First Pass Flowchart

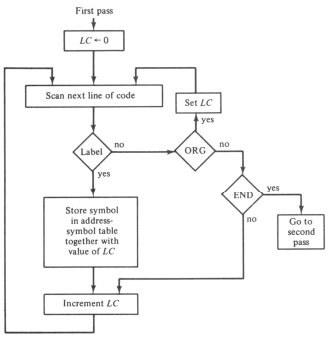


Figure 6-1 Flowchart for first pass of assembler.

Example of Symbol Table

TABLE 6-12 Address Symbol Table for Program in Table 6-8

Memory word	Symbol or (LC)*	Hexadecimal code	Binary representation
1	M I	4D 49	0100 1101 0100 1001
2	N ,	4E 2C	0100 1110 0010 1100
3	(LC)	01 06	0000 0001 0000 0110
4	S U	53 55	0101 0011 0101 0101
5	B ,	42 2C	0100 0010 0010 1100
6	(LC)	01 07	0000 0001 0000 0111
7	D I	44 49	0100 0100 0100 1001
8	F ,	46 2C	0100 0110 0010 1100
9	(LC)	01 08	0000 0001 0000 1000

* (LC) designates content of location counter.

Second Pass Flowchart

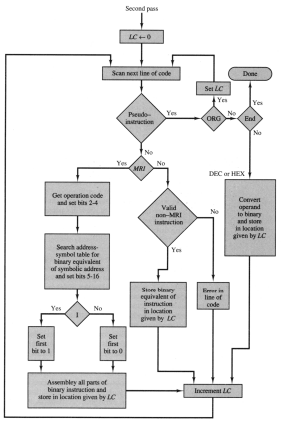


Figure 6-2 Flowchart for second pass of assembler.

Program Loops

TABLE 6-13 Symbolic Program to Add 100 Numbers

Line			
1		ORG 100	/Origin of program is HEX 100
2		LDA ADS	/Load first address of operands
3		STA PTR	/Store in pointer
4		LDA NBR	/Load minus 100
5		STA CTR	/Store in counter
6		CLA	/Clear accumulator
7	LOP,	ADD PTR I	/Add an operand to *AC*
8		ISZ PTR	/Increment pointer
9		ISZ CTR	/Increment counter
10		BUN LOP	/Repeat loop again
11		STA SUM	/Store sum
12		HLT	/Halt
13	ADS,	HEX 150	/First address of operands
14	PTR,	HEX 0	/This location reserved for a pointer
15	NBR,	DEC −100	/Constant to initialized counter
16	CTR,	HEX 0	/This location reserved for a counter
17	SUM,	HEX 0	/Sum is stored here
18		ORG 150	/Origin of operands is HEX 150
19		DEC 75	/First operand
•			
•			
•			
118		DEC 23	/Last operand
119		END	/End of symbolic program

Multiplication Algorithm

X **holds the multiplicand**
Y **holds the multiplier**
P **forms the product**

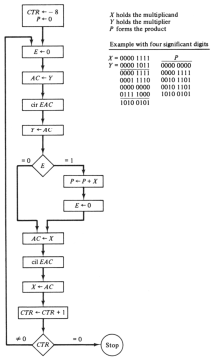
Example with four significant digits

	P
X = 0000 1111	0000 0000
Y = 0000 1011	0000 1111
0000 1111	0010 1101
0001 1110	0010 1101
0000 0000	1010 0101
0111 1000	
1010 0101	

Multiplication Program for 8 bit

TABLE 6-14 Program to Multiply Two Positive Numbers

```
        ORG 100
LOP,    CLE         /Clear E
        LDA Y       /Load multiplier
        CIR         /Transfer multiplier bit to E
        STA Y       /Store shifted multiplier
        SZE         /Check if bit is zero
        BUN ONE     /Bit is one; go to ONE
        BUN ZRO     /Bit is zero; go to ZRO
ONE,    LDA X       /Load multiplicand
        ADD P       /Add to partial product
        STA P       /Store partial product
        CLE         /Clear E
ZRO,    LDA X       /Load multiplicand
        CIL         /Shift left
        STA X       /Store shifted multiplicand
        ISZ CTR     /Increment counter
        BUN LOP     /Counter not zero; repeat loop
        HLT         /Counter is zero; halt
CTR,    DEC -8      /This location serves as a counter
X,      HEX 000F    /Multiplicand stored here
Y,      HEX 000B    /Multiplier stored here
P,      HEX 0       /Product formed here
        END
```



X holds the multiplicand
Y holds the multiplier
P forms the product

Example with four significant digits

	P
X = 0000 1111	0000 0000
Y = 0000 1011	0000 1111
0000 1111	0001 1110
0001 1110	0010 1101
0000 0000	0010 1101
0111 1000	1010 0101
1010 0101	

Figure 6-3 Flowchart for multiplication program.

Long Addition Program

TABLE 6-15 Program to Add Two Double-Precision Numbers

```
        LDA AL      /Load A low
        ADD BL      /Add B low, carry in E
        STA CL      /Store in C low
        CLA         /Clear AC
        CIL         /Circulate to bring carry into AC(16)
        ADD AH      /Add A high and carry
        ADD BH      /Add B high
        STA CH      /Store in C high
        HLT
AL,     —           /Location of operands
AH,     —
BL,     —
BH,     —
CL,     —
CH,     —
```

0

Subroutine's Example: 4 x CIL

TABLE 6-16 Program to Demonstrate the Use of Subroutines

```
Location
                ORG 100     /Main program
100             LDA X       /Load X
101             BSA SH4     /Branch to subroutine
102             STA X       /Store shifted number
103             LDA Y       /Load Y
104             BSA SH4     /Branch to subroutine again
105             STA Y       /Store shifted number
106             HLT
107     X,      HEX 1234
108     Y,      HEX 4321
                            /Subroutine to shift left 4 times
109     SH4,    HEX 0       /Store return address here
10A             CIL         /Circulate left once
10B             CIL
10C             CIL
10D             CIL         /Circulate left fourth time
10E             AND MSK     /Set AC(13–16) to zero
10F             BUN SH4 I   /Return to main program
110     MSK,    HEX FFF0    /Mask operand
                END
```

Parameter Example: OR

TABLE 6-17 Program to Demonstrate Parameter Linkage

Location			
		ORG 200	
200		LDA X	/Load first operand into *AC*
201		BSA OR	/Branch to subroutine OR
202		HEX 3AF6	/Second operand stored here
203		STA Y	/Subroutine returns here
204		HLT	
205	X,	HEX 7B95	/First operand stored here
206	Y,	HEX 0	/Result stored here
207	OR,	HEX 0	/Subroutine OR
208		CMA	/Complement first operand
209		STA TMP	/Store in temporary location
20A		LDA OR I	/Load second operand
20B		CMA	/Complement second operand
20C		AND TMP	/AND complemented first operand
20D		CMA	/Complement again to get OR
20E		ISZ OR	/Increment return address
20F		BUN OR I	/Return to main program
210	TMP,	HEX 0	/Temporary storage
		END	

Pointer Parameter Example: Move Block

TABLE 6-18 Subroutine to Move a Block of Data

		/Main program
	BSA MVE	/Branch to subroutine
	HEX 100	/First address of source data
	HEX 200	/First address of destination data
	DEC −16	/Number of items to move
	HLT	
MVE,	HEX 0	/Subroutine MVE
	LDA MVE I	/Bring address of source
	STA PT1	/Store in first pointer
	ISZ MVE	/Increment return address
	LDA MVE I	/Bring address of destination
	STA PT2	/Store in second pointer
	ISZ MVE	/Increment return address
	LDA MVE I	/Bring number of items
	STA CTR	/Store in counter
	ISZ MVE	/Increment return address
LOP,	LDA PT1 I	/Load source item
	STA PT2 I	/Store in destination
	ISZ PT1	/Increment source pointer
	ISZ PT2	/Increment destination pointer
	ISZ CTR	/Increment counter
	BUN LOP	/Repeat 16 times
	BUN MVE I	/Return to main program
PT1,	—	
PT2,	—	
CTR,	—	

Example: Read Char

TABLE 6-23 Program to Service an Interrupt

Location			
0	ZRO,	—	/Return address stored here
1		BUN SRV	/Branch to service routine
100		CLA	/Portion of running program
101		ION	/Turn on interrupt facility
102		LDA X	
103		ADD Y	/Interrupt occurs here
104		STA Z	/Program returns here after interrupt
•		•	
•		•	
•			/Interrupt service routine
200	SRV,	STA SAC	/Store content of *AC*
		CIR	/Move *E* into *AC*(1)
		STA SE	/Store content of *E*
		SKI	/Check input flag
		BUN NXT	/Flag is off, check next flag
		INP	/Flag is on, input character
		OUT	/Print character
		STA PT1 I	/Store it in input buffer
		ISZ PT1	/Increment input pointer
	NXT,	SKO	/Check output flag
		BUN EXT	/Flag is off, exit
		LDA PT2 I	/Load character from output buffer
		OUT	/Output character
		ISZ PT2	/Increment output pointer
	EXT,	LDA SE	/Restore value of *AC*(1)
		CIL	/Shift it to *E*
		LDA SAC	/Restore content of *AC*
		ION	/Turn interrupt on
		BUN ZRO I	/Return to running program
	SAC,	—	/*AC* is stored here
	SE,	—	/*E* is stored here
	PT1,	—	/Pointer of input buffer
	PT2,	—	/Pointer of output buffer