Computer System Architecture

DR. Howida Youssry

INTRODUCTION

Those concerned with computer architecture should have a knowledge of both hardware and software because the two branches influence each other.

Instruction Set of the *Basic Computer*

Symbol	*Hexa code*	*Description*
AND	0 or 8	AND M to AC
ADD	1 or 9	Add M to AC, carry to E
LDA	2 or A	Load AC from M
STA	3 or B	Store AC in M
BUN	4 or C	Branch unconditionally to m
BSA	5 or D	Save return address in m and branch to m+1
ISZ	6 or E	Increment M and skip if zero
CLA	7800	Clear AC
CLE	7400	Clear E
CMA	7200	Complement AC
CME	7100	Complement E
CIR	7080	Circulate right E and AC
CIL	7040	Circulate left E and AC
INC	7020	Increment AC, carry to E
SPA	7010	Skip if AC is positive
SNA	7008	Skip if AC is negative
SZA	7004	Skip if AC is zero
SZE	7002	Skip if E is zero
HLT	7001	Halt computer
INP	F800	Input information and clear flag
OUT	F400	Output information and clear flag
SKI	F200	Skip if input flag is on
SKO	F100	Skip if output flag is on
ION	F080	Turn interrupt on
IOF	F040	Turn interrupt off

m: effective address
M: memory word (operand)
found at m

MACHINE LANGUAGE

Program

A list of instructions or statements for directing the computer to perform a required data processing task

Various types of programming languages

- Hierarchy of programming languages

- **Machine-language**
  - **Binary code**
  - **Octal or hexadecimal code**
- **Assembly-language (Assembler)**
  - **Symbolic code**
- **High-level language (Compiler)**

COMPARISON OF PROGRAMMING LANGUAGES

- **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

- **Hexa program**

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

- **Program with Symbolic OP-Code**

Location	Instruction		Comments
000	LDA	004	Load 1st operand into AC
001	ADD	005	Add 2nd operand to AC
002	STA	006	Store sum in location 006
003	HLT		Halt computer
004	0053		1st operand
005	FFE9		2nd operand (negative)
006	0000		Store sum here

- **Assembly-Language Program**

```
     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
```

- **Fortran Program**

```
INTEGER  A, B, C
DATA  A,83 / B,-23
C = A + B
END
```

ASSEMBLY LANGUAGE

Syntax of the BC assembly language

Each line is arranged in three columns called fields

***Label* field**

- **- May be empty or may specify a symbolic address consists of up to 3 characters**
- **- Terminated by a comma**

***Instruction* field**

- **- Specifies a machine or a pseudo instruction**
- **- May specify one of**
  - *** Memory reference instr. (MRI)**

    MRI consists of two or three symbols separated by spaces.

    ADD OPR (direct address MRI)

    ADD PTR I (indirect address MRI)
  - *** Register reference or input-output instr.**

    Non-MRI does not have an address part
  - *** Pseudo instr. with or without an operand**

    Symbolic address used in the instruction field must be defined somewhere as a label

***Comment* field**

- **- May be empty or may include a comment**

PSEUDO-INSTRUCTIONS

ORG N

Hexadecimal number N is the memory loc. for the instruction or operand listed in the following line

END

Denotes the end of symbolic program

DEC N

Signed decimal number N to be converted to the binary

HEX N

Hexadecimal number N to be converted to the binary

Example: 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
```

TRANSLATION TO BINARY

Hexadecimal Code		*Symbolic Program*	
Location	*Content*		
			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

ASSEMBLER - FIRST PASS -

Assembler

Source Program - Symbolic Assembly Language Program

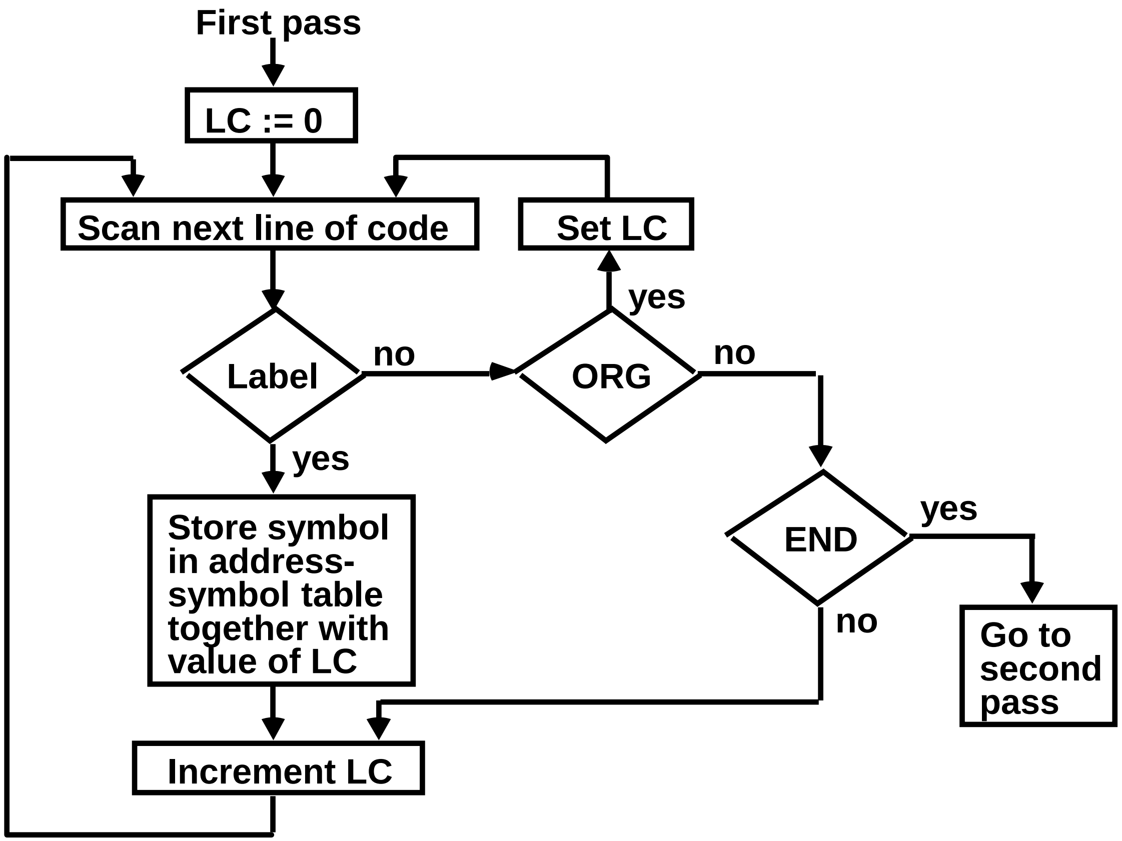
Object Program - Binary Machine Language Program

Two pass assembler

1st pass: generates a table that correlates all user defined (address) symbols with their binary equivalent value

2nd pass: binary translation

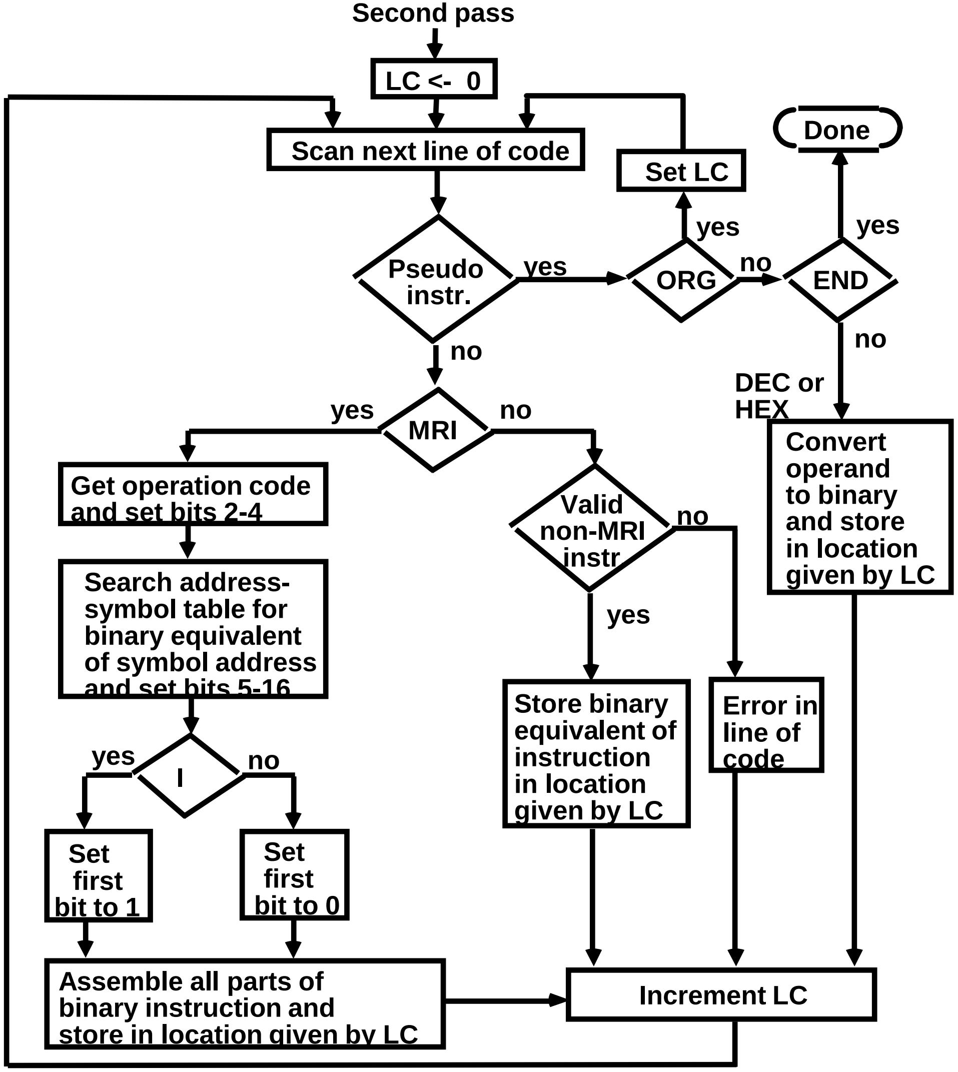
First pass



ASSEMBLER - SECOND PASS -

Second Pass

Machine instructions are translated by means of table-lookup procedures;
(1. Pseudo-Instruction Table, 2. MRI Table, 3. Non-MRI Table
4. Address Symbol Table)


Second pass
LC <- 0
Scan next line of code
Set LC
Done
yes
yes
Pseudo instr.
yes
ORG
no
END
no
no
DEC or HEX
yes
MRI
no
Convert operand to binary and store in location given by LC
Get operation code and set bits 2-4
Valid non-MRI instr
no
Search address-symbol table for binary equivalent of symbol address and set bits 5-16
yes
Store binary equivalent of instruction in location given by LC
Error in line of code
yes
I
no
Set first bit to 1
Set first bit to 0
Assemble all parts of binary instruction and store in location given by LC
Increment LC