PAGE NO. .....

Unit 2 :- Programming the Basic Computer

1. Introduction
2. M/C Language
3. Assembly Language
4. Assembler
5. Program loops
6. Programing Arithmetic & logic Operations
7. Subroutines
8. I-O Programming
9. Micro Programmed Control : Control Memory, Address sequencing, Micro program Example, Design of control Unit.

Computer → hardware (physical components)
→ software (Program)

- Program written by users dependent or independent of hardware

Machine Language:→

- Program is a list of instructions or statement for directing the comp. to perform a required data-processing task.
- Programs are translated into binary representation.
  1. Binary code:-
  2. Octal or hexadecimal code
  3. Symbolic code (letter, numerals, special char)
  4. High-level programming language.
- M/C language program is a binary program.

Because 8086 (total 25 instruction
16-memory
16-register)

Assembly Language:→

- A programming language is defined by a set of rules.
- Basic unit of an assembly lang. program is a line of code.

Rules of Language:→

AL arranged in 3 columns called fields

1. Label field: empty or symbolic address
2. Instruction field: M/C instruction or pseudoinstruction
3. Comment field: empty or include comment.

Instruction field in AL

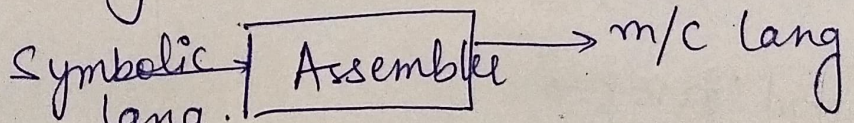
PAGE NO.

1. Memory-reference instr. (MRI) eg ADD OPR, ADD PTR 1
2. Reg-Reference or I/P-O/P Instruction (non-MRI) CLA
3. pseudo instruction with or without an operand. ORG 0

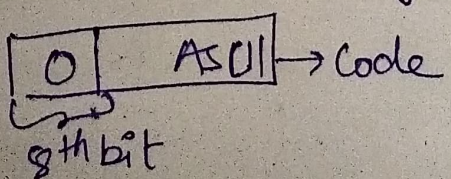
- first program is written on assembly language and converted into hexadecimal code

The Assembler:→

An assembler is a program that accepts a symbolic language program and produces its binary m/c lang equivalent.



- user type symbolic program on a system, each char is represented by 8-bit



- CR is used to produce the return key

eg:→ PL3, LDA SUB I → if I missing d. in-direct add.

- Line of code stored in consecutive memory location

This code stored at 7 location

1	P	L
2	3	,
3	L	D
4	A	Space
5	S	U
6	B	Space
7	I	CR

if comment is // then neglet the comment.

- Assembler have two-pass

o In first pass:—

- Read/scan symbolic program twice.

In 1st scan — convert symbols into binary value.

2nd scan convert into binary translation.

- Track memory with the help of LC (location counter)
- ORG pseudo-instruction intialize the location counter / if ORG missing LC=0

o Second Pass:→

- M/C instructions are translated during the second pass by means of table-lookup procedures.
- Assembler uses four table for this

① Pseudo-instruction table → used for subroutine ORG, END, DEC, HEX

② MRI table — Memory Ref. Inst.

③ Non-MRI table — Reg. Ref. & I/O instruction

④ Address symbol table

Program Loops :→

- A program loop is a sequence of instructions that are executed many times, each time with a different set of data.

eg:→

```
int i, sum = 0, a[100];
for (i=1 ; i<=100 ; i++)
{
    sum = sum + a[i];
}
```

So, this program is converted into symbolic-progra and assembler work on it.

Programming Arithmetic & Logic Operations :-

- System have hundreds of instruction to perform operations, but few systems have basic operation to perform.
- Operations that are implemented in a computer with one m/c instruction are said to be implemented by hardware.

- Operations implemented by a set of instruction that constitute a program are said to be implemen by S/W.
- H/W implementation more costly (due to additional circuits)
- S/W implementation result in long programs both in no. of instructions and in execution time.

X holds multiplicand
Y holds multiplies
P forms product

```
  0000 1111
  0000 1011
-----------
  0000 1111
  0001 1110
  0000 0000 0
  0111 1000 0
-----------
  1010 0101
```

Double-precision Addition:-

- When 2 16-bit unsigned no. are multiplied the result is 32-bit product that must be staed in 2 memory words.
- A no. stored in two memory words is said to have double precision.
- When a partial product is computed, it is necessary that a double-precision no. be added to the shifted multiplicand, which is also a double precision no.

- 2 16-bit no's are stoded as follows

1st no in AL (low) & AH (high)
2nd no in BL (low) & BH (high)

- 2 low (AL+BL) order portions are added and carry transfered into $\bar{C}$.
- AC is cleared and bit in $\bar{C}$ is circulated into LSB of AC
- Two high order portions are added and carry also and stored in CL and CH

Logic Operations :-

- 3 M/C instructios to perform logic operations

AND
CMA — complement of $\bar{A}$
CLA — carry Look Ahead Adder.

OR is not in logic m/c instruction

To perform OR do this

```
LDA  A      — Load first operand A
CMA         — complement of Ā
STA  TMP    — Store in Temp location
LDA  B      Load B
CMA         complement of B̄
AND  TMP    AND with Ā to get Ā∧B̄
```

Shift Operations:-

- Circular shift op. are basic m/c instructions
- Two more shift are there → Logic
  → Arithmetic

CLE, CIR
CLE
CIL

Clear E to 0

Subroutines:→

- Set of common instructions that can be used in a program many times is called a subroutine.
  - It is used in main progeam. A branch is executed to the beginning of subroutine. After the subroutine, branch is back to main progeam.
  - It's consist of self contained sequence of instructions that carries out a given task.
  - Branch can be made to the subroutine from any part of the main program.
- In Computer, BSA (branch & save return add) is the link b/w the main program & a subroutine
- The procedure for branching to a subroutine and returning to the main program is refersed to as a subroutine linkage.
- BUN instruction is used for branch unconditionally

Input-Output Programming:-

- The symbols in which program is defined as string of characters and each char. is assigned an 8-bit code, so that it can be stored in computer memory.
- Binary-coded character enters the computer when an INP instruction is executed.
- Binary-coded character is transferred to the o/p device when OUT instruction is executed.
- SKI instruction check input flag to see char. is available ~~for~~ transfer.

  if flag = 1, next instruction skipped
  else flag = 0, next instruction executed in sequence

eg: Input char:-

```
CIF   SKI      — check input flag
      BUN CIF  — flag=0 branch to check again
      INP      — flag=1, input char
      OUT      — Print char
      STA CHR  — store char
      HLT      — / store char here
CHR
```

Output char:-

```
      LDA  CHR   — Load char in AC
COF   SKO        / check O/P flag
      BUN COF    — / flag = 0, branch to check again
      OUT        — / flag = 1, O/P char
      HLT
CHR   HEX 0057   / Character is "W"
```

—X—

Assignment Questions:-

1. Describe the Input-Output Programming with example.
2. Explain Subroutines.
3. Explain Assembler
4. Explain direct & indirect register addressing mode with suitable example.
5. Explain arithmetic micro-operation in register transfer language.
6. Write the difference b/w Hardwired CU and Micro programmed CU.

eg: of Assembly program

Subtract Two numbers

```
        ORG   100    / Origin of prog. 100
        LDA   SUB    / Load Subtrahend to AC
        CMA          / Complement AC
        INC          / Increment AC
        ADD   MIN    / Add minuend to AC
        STA   DIF    / Store diff
        HLT          / Halt computer
MIN,    DEC   83     / Minuend
SUB     DEC   -23    / Subtrahend
DIF     HEX   0      / Diff stored
        END          / End of symbolic program
```

Program Loop Example:- (Add 100 no.s)

```
1.          ORG   100     / origin 100
2.          LDA   ADS     / load 1'st operands
3.          STA   PTR     / store in Pointer
4.          LDA   NBR     / load minus 100
5.          STA   CTR     / Clear AC Store in counter
6.          CLA           / Clear AC, operand
7.  LOP     ADD   PTR I   / Add I 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 add of operands
14. PTR,    HEX   0       / Loc for pointer
15. NBR,    DEC   -100    / counter value:
16. CTR,    HEX   0       / counter after value
17. SUM,    HEX   0       / Sum is stored
18.         ORG   150     / origin of operand
19.         DEC   75      / first operand
 :
 :
118         DEC   23      / Last operand
119         END           / End of symbolic progm.
```

Arithmetic & Logic Op:-

Multiply Two Positive No.

```
        ORG, 100
LOP,    CLE       / Clear E
        LDA Y     / Load multiplier in AC
        CIR       / Transfer multiplier bit to E
        STA Y     / Stored shifted multiplier
        SZE       / Check if bit is zero
        BUN ONE   / Bit is zero; go to ZRO
        BUN ZRO   / Load multiplicand
ONE,    LDA X     /
        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 loc — counter
X,      HEX 000F  — Multiplicand Stored here
Y,      HEX 000B  ← Multiplier stored here
P,      HEX 0     — product formed here
        END
```

bit is one: go to ONE

Use of Subroutines :-

```
        ORG 100
100     LDA X      — Load X
101     BSA SH4    — Branch to subroy
102     STA X      — Store shifted no.
103     LDA Y      — Load Y
104     BSA SH4    — Branch to subroutine
105     STA Y      / store shifted no.
106     HLT
107 X   HEX 1234
108 Y   HEX 4321
                   / subroutine to shift left 4 times
109 SH4, HEX 0     / Store return address
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
```

DC → Digital Comp

POORNIMA
COLLEGE OF ENGINEERING
DETAILED LECTURE NOTES

PAGE NO. 1

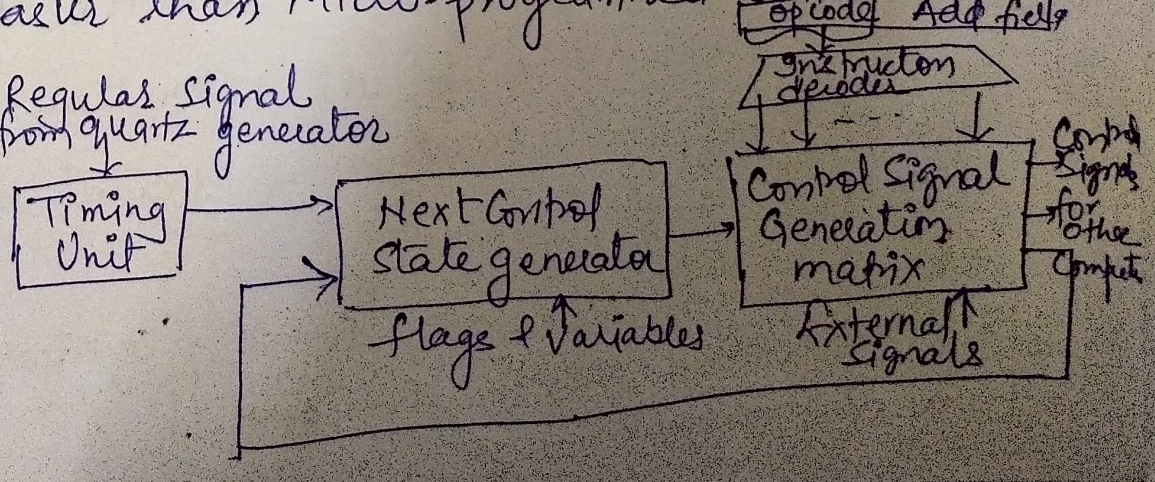
Microprogrammed Control

Control Memory :→ {Func of CU in DC is to initiate Seq. of micro-op

- To execute an instruction, CU generate the required control signal in the proper sequence.
- There are 2 approaches for generating control signals

① Hardwired control unit :- When signals are generated by H/W using conventional logic

- It is a structure of making control signals using finite state Machine (FSM) that changes from one state to another in every clock cycle depending on the contents of IR, condition codes & external inputs.
- The final circuit is generated by physically linking the components including gates, flip-flops and drums are known as Hardwired controller.
- It is faster than Micro-programmed control.



- RISC Architecture is based on the Hard. Control Unit.

② Micro progeammed Control Unit:→

- A control unit whose binary control values are saved as word in memory is called a microprogrammed control unit.
- Control signals are generated by a progeam that is similar to m/c lang progeams.
- ~~The micro pro~~ Memory add of the control unit denotes the add of micro-instruction.
- The microinstruction has a control word.

Important Term in micro-progeammed CU

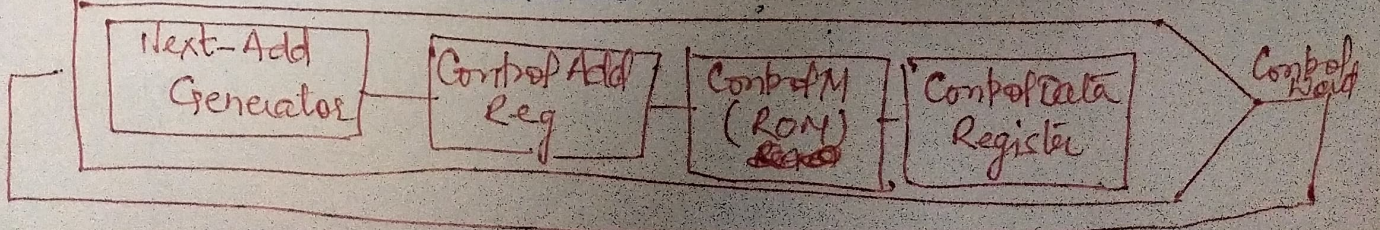
① Control Word (CW) :- Word individual bit represent various control signals

② Micro routine (MR) :- Seq. of CW to control seq of a m/c instruction constitutes MR.

③ Micro Instruction (MI) :- Individual CW in MR is MI

④ Micro-program (MP) Sequece of MI is called MP, Stored in ROM or RAM

⑤ Control store/M MR for all instructions in Inst Set of a comp are stored in a special memory called control store.



Hardwired Control Unit	Microprogrammed Unit
- Not applicable to change the structure and instruction set once it is developed.	- Applicable to make changes in microprogram saved in control memory.
- Design complex	- Design simplified.
- Architecture and inst. are not specified	- Archi. & Inst. are specified.
- Quick	- Moderate
- Processor to create signals executed in right seq.	- Microseq from which inst. are decoded and executed.
- Operate through flip-flop drums, seq. circuit.	- Control through ALU, registers, buses, IR.

Address Sequencing :→

- Micro inst. are stored in control memory in groups, Each group specifying a routine.
- So it follow following steps :-

① Initial add $\xrightarrow[\text{Loaded}]{}$ Control Add Reg (CAR) when power on.

② It is add. of first micro-inst. that activates the instruction fetch routine.

③ CAR Increment through micro inst.

④ CM go through routine and find the add of operand.

⑤ Now add is available in Memory Add Reg.

⑥ Generate micro-operations and execute inst fetched from M.

⑦ Mapped b/w inst code & control memory.

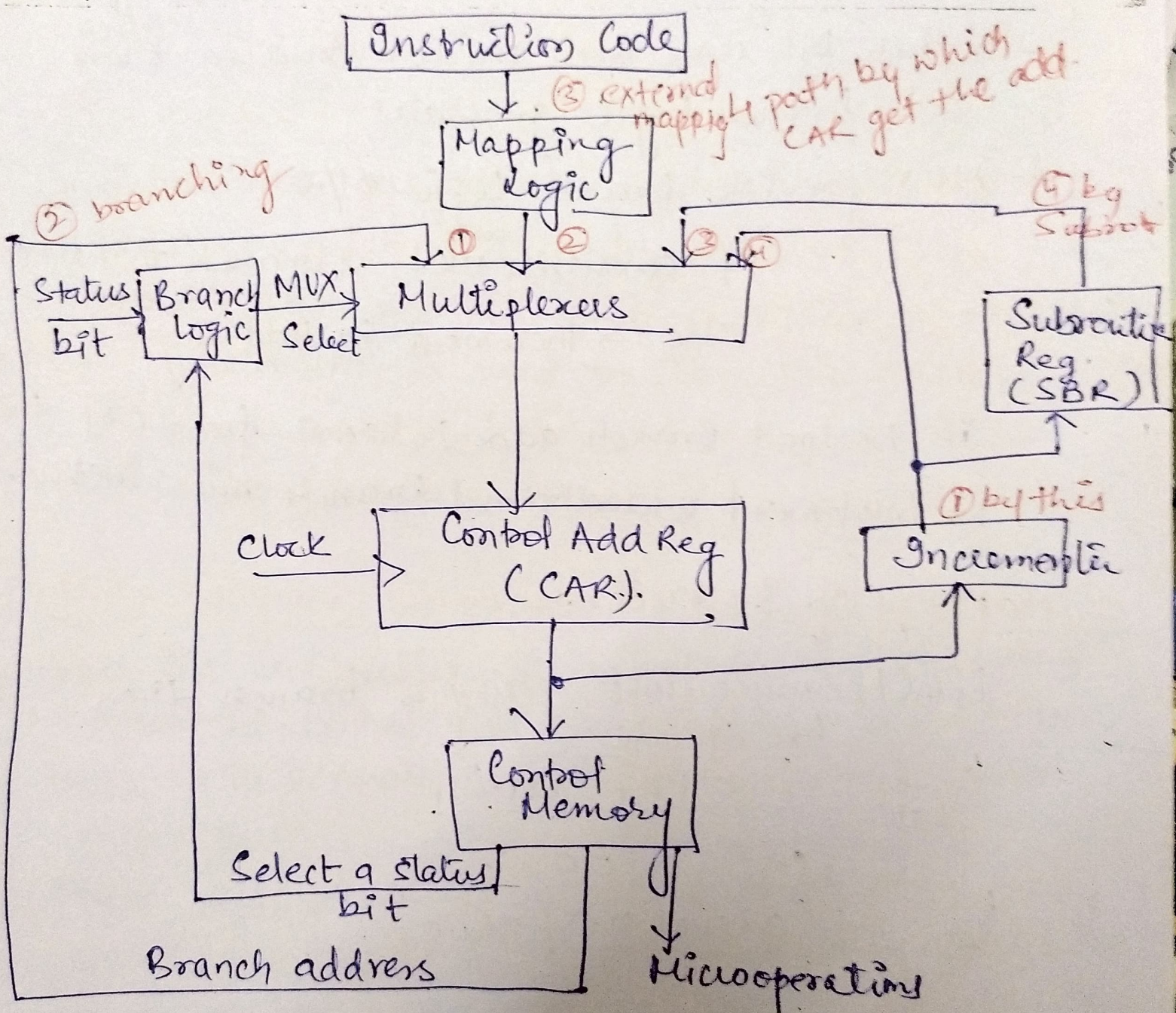
So we can say that basically these 4 steps take place.

① Increment of the control Add Reg.

② Unconditional branch or conditional branch, depending on status bit conditions.

③ A mapping procss from bits of inst to an add for CM.

④ A facility for subroutines call and return.



- In this diagram micro instructions is used to contain a set of bits in CM.
- With the help of Micro instr. we able to start micro-operations in comp. Reg.

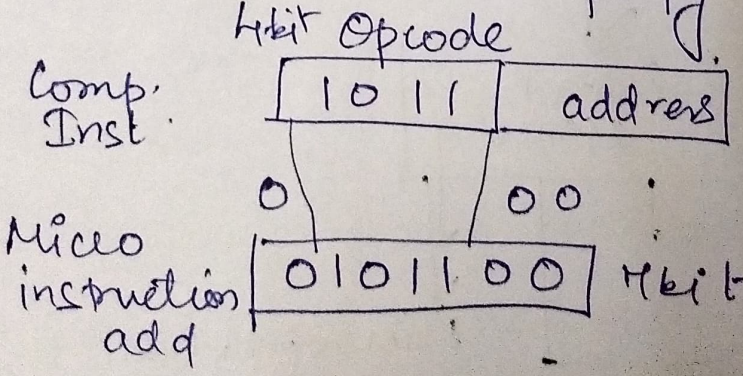
Control Unit Diagram

① Conditional Branching :-

- In diagram branch logic used to provide decision making capabilities in CU.
- Special bits (status condition) provide parameter info such as mode bit, sign bit, carry-out, input & o/p status
- If these bits come with micro instruction then branch logic generates.
- MUX provide branch logic H/W
  - if condition met → branch to initial add.
  - else → increment in Add. Reg.
- if We load Branch add in CAR from CM then we implement unconditional branch micro instruc.

② Mapping of Instructions :-

If in CM micro-inst specify a branch then



With the help of mapping 4-bit Opcode convert into 7 bit micro instruction add for CM.

- opcode - Mapped with memory using ROM or PLD (Programmable logic device) and

with the help of CAR come to CM.

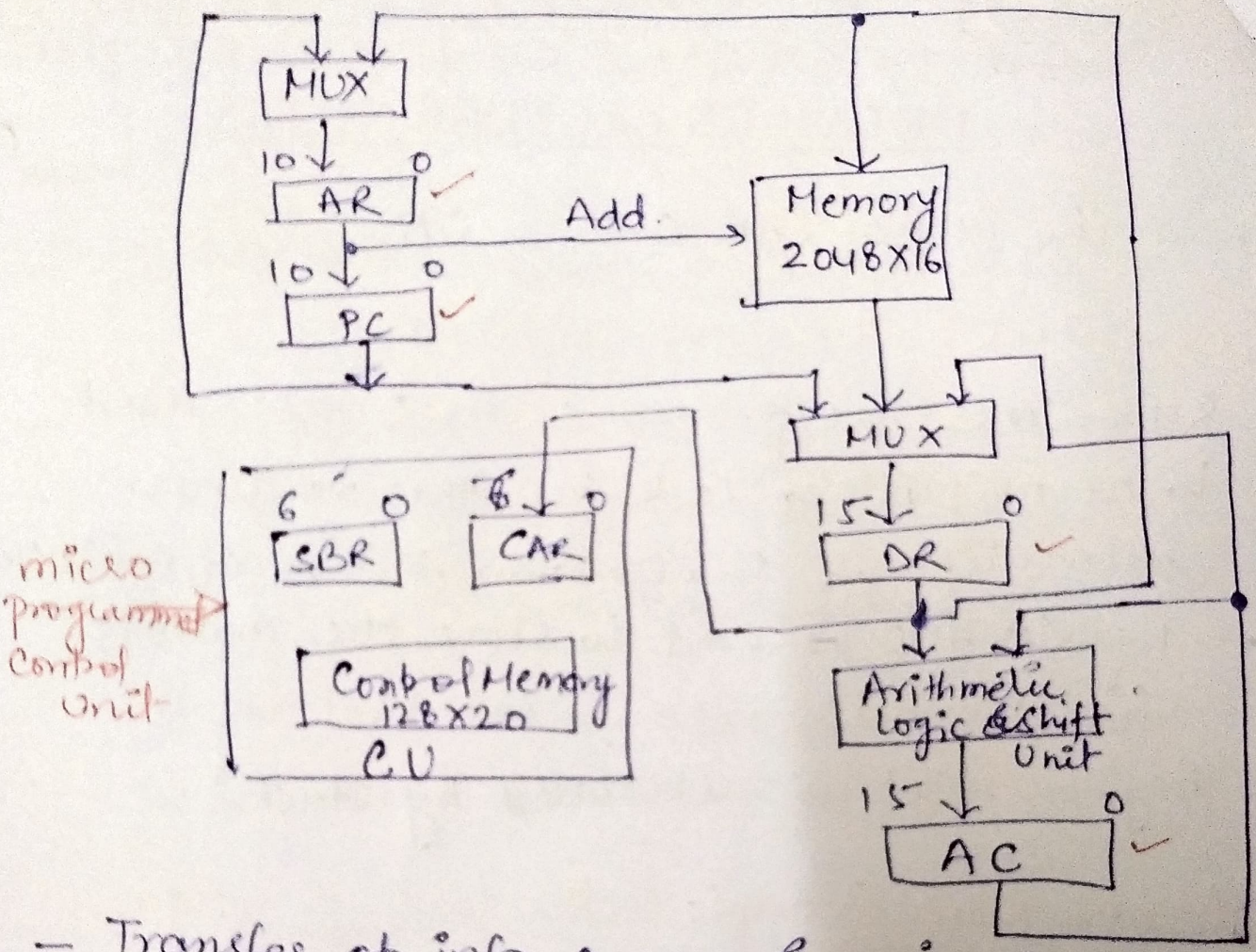
Subroutine: →

- Subroutines are programs that are used to accomplish a task by other routines.
- Subroutine use the common section of microcode.
- Register file is used to store the address for subroutines.
- Register files are structured by stack.

Microprogram Example: →

- Once the configuration of a computer and its micro programmed control unit is established, the designer task is to generate the microcode for the control memory.
- This code generation is called micro programming and is a process to similar to conventional m/c lang programming.
- It consist of 2 memry unit-
  - Main Memory ← storing instruction & data
  - Control memory → storing micro program

- 4 Reg with Processor unit & 2 with CU
- Processor Register are PC, MAR, MDR & AC
- CU has CAR & SBR (Subroutine register)

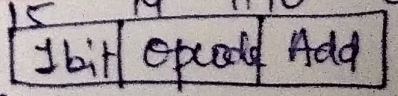


- Transfer of info among Reg. in the processor through MUX.

DR from AC, PC, Memory.
AR from PC or DR
PC from AR only

Instruction format:-

- 3 fields,



Indirect bit - Addressing

eg: ADD 0000 AC ← AC + M[EA]

opcode

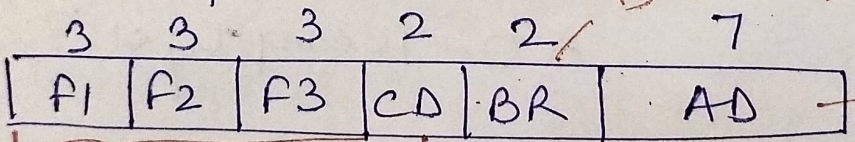
BRANCH :- branch of effective add if the operand in AC is negative.

BRANCH	0001	If (AC < 0) then (PC ← EA) (effective Add)
STORE	0010	M[EA] ← AC
EXCHAGE	0011	AC ← M[EA], M[EA] ← AC

~~Microcode~~

Micro-Instruction format :→

- Micro instruction (MI) format for the CM.
- 20 bits of the MI are divided into fours functional part

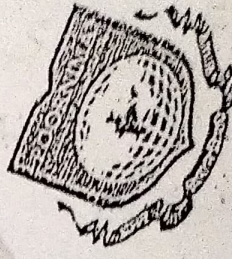


3 field (f1, f2, f3) → 7 diff micro operation

DR ← M[AR] & PC ← PC+1

with f2=100
with f3=101

Symbolic Micro-Instruction:-



- A microprogram is to define symbols for ea field of the microinstruction and to give user the capability for defining their own symbolic addresses.

1. Label field — empty or specify symbolic add.
   Label terminated with (:)
2. Micro-operation field consist of 1, 2 or 3 symbols Seprated by comma
   No more than one symbol in f field.
3. CD — letter U, I, S, Z
4. BR — four symbols.
5. AD field
   - ① Symbolic add
   - ② Symbol NEXT → next add in sequence
   - ③ if BR have — RET / MAP

   AD - is empty or 7 zeros.

ORG:- define ase first add or micro program routine

Fetch Routine:-

- CM has 128 words
- each word contain 20 bits
- When microprogram the CM it is necessary to determine bit value of all 128 words.

$AR \leftarrow PC$

$DR \leftarrow M[AR]$ $PC \leftarrow PC+1$

$AR \leftarrow DR(0\text{-}10)$, $CAR(2\text{-}5) \leftarrow DR(11\text{-}14)$, $CAR(0,1,6) \leftarrow 0$

- The fetch routine needs 3 micro-instructions which are placed in CM at add 64, 65 & 66.

Symbolic Micro-program:-

- The executed of the 3rd micro-instruction in the fetch routine result in a branch to add 0XXX00, where XXXX are 4-bit operation code.
- Each routine we must provide microinstructions for evaluating the effective add and for executing the instruction. Indirect add mode is associate with all memory-reference inst.