Unit-III

Part-1: MICROPROGRAMMED CONTROL

Contents:

- ✓ Control memory
- ✓ Address Sequencing
- ✓ Microprogram Example
- ✓ Design of Control Unit

Introduction:

- ➢ The function of the control unit in a digital computer is to initiate sequence of microoperations.
- ➢ Control unit can be implemented in two ways
  - o Hardwired control
  - o Microprogrammed control

Hardwired Control:

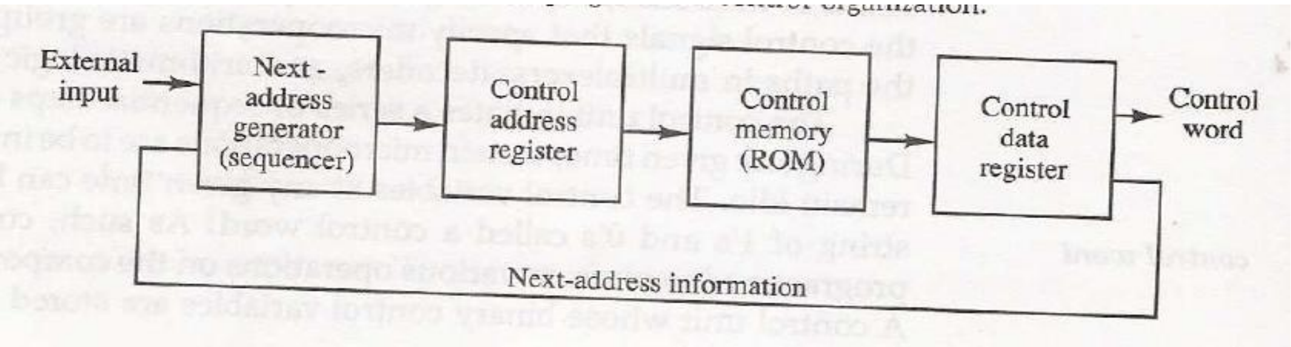
- ✓ When the control signals are generated by hardware using conventional logic design techniques, the control unit is said to be *hardwired.*
- ✓ The key characteristics are
  - o High speed of operation
  - o Expensive
  - o Relatively complex
  - o No flexibility of adding new instructions
- ✓ Examples of CPU with hardwired control unit are Intel 8085, Motorola 6802, Zilog 80, and any RISC CPUs.

Microprogrammed Control:

- ✓ Control information is stored in control memory.
- ✓ Control memory is programmed to initiate the required sequence of micro-operations.
- ✓ The key characteristics are
  - o Speed of operation is low when compared with hardwired
  - o Less complex
  - o Less expensive
  - o Flexibility to add new instructions
- ✓ Examples of CPU with microprogrammed control unit are Intel 8080, Motorola 68000 and any CISC CPUs.

1. Control Memory:

- ➢ The control function that specifies a microoperation is called as ***control variable.***
- ➢ When control variable is in one binary state, the corresponding microoperation is executed. For the other binary state the state of registers does not change.
- ➢ The active state of a control variable may be either 1 state or the 0 state, depending on the application.
- ➢ For bus-organized systems the control signals that specify microoperations are groups of bits that select the paths in multiplexers, decoders, and arithmetic logic units.
- ➢ ***Control Word:*** The control variables at any given time can be represented by a string of 1's and 0's called a control word.
- ➢ All control words can be programmed to perform various operations on the components of the system.
- ➢ ***Microprogram control unit:*** A control unit whose binary control variables are stored in memory is called a microprogram control unit.
- ➢ The control word in control memory contains within it a *microinstruction.*
- ➢ The microinstruction specifies one or more micro-operations for the system.
- ➢ A sequence of microinstructions constitutes a *microprogram*.
- ➢ The control unit consists of control memory used to store the microprogram.
- ➢ Control memory is a permanent i.e., read only memory (ROM).
- ➢ The general configuration of a micro-programmed control unit organization is shown as block diagram below.



- The control memory is ROM so all control information is permanently stored.
- The control memory address register (CAR) specifies the address of the microinstruction and the control data register (CDR) holds the microinstruction read from memory.
- The next address generator is sometimes called a microprogram sequencer. It is used to generate the next micro instruction address.
- The location of the next microinstruction may be the one next in sequence or it may be located somewhere else in the control memory.
- So it is necessary to use some bits of the present microinstruction to control the generation of the address of the microinstruction.
- Sometimes the next address may also be a function of external input conditions.
- The control data register holds the present microinstruction while next address is computed and read from memory. The data register is times called a *pipeline register.*

- A computer with a microprogrammed control unit will have two separate memories: a main memory and a control memory
- The microprogram consists of microinstructions that specify various internal control signals for execution of register microoperations
- These microinstructions generate the microoperations to:
    - fetch the instruction from main memory
    - evaluate the effective address
    - execute the operation
    - return control to the fetch phase for the next instruction

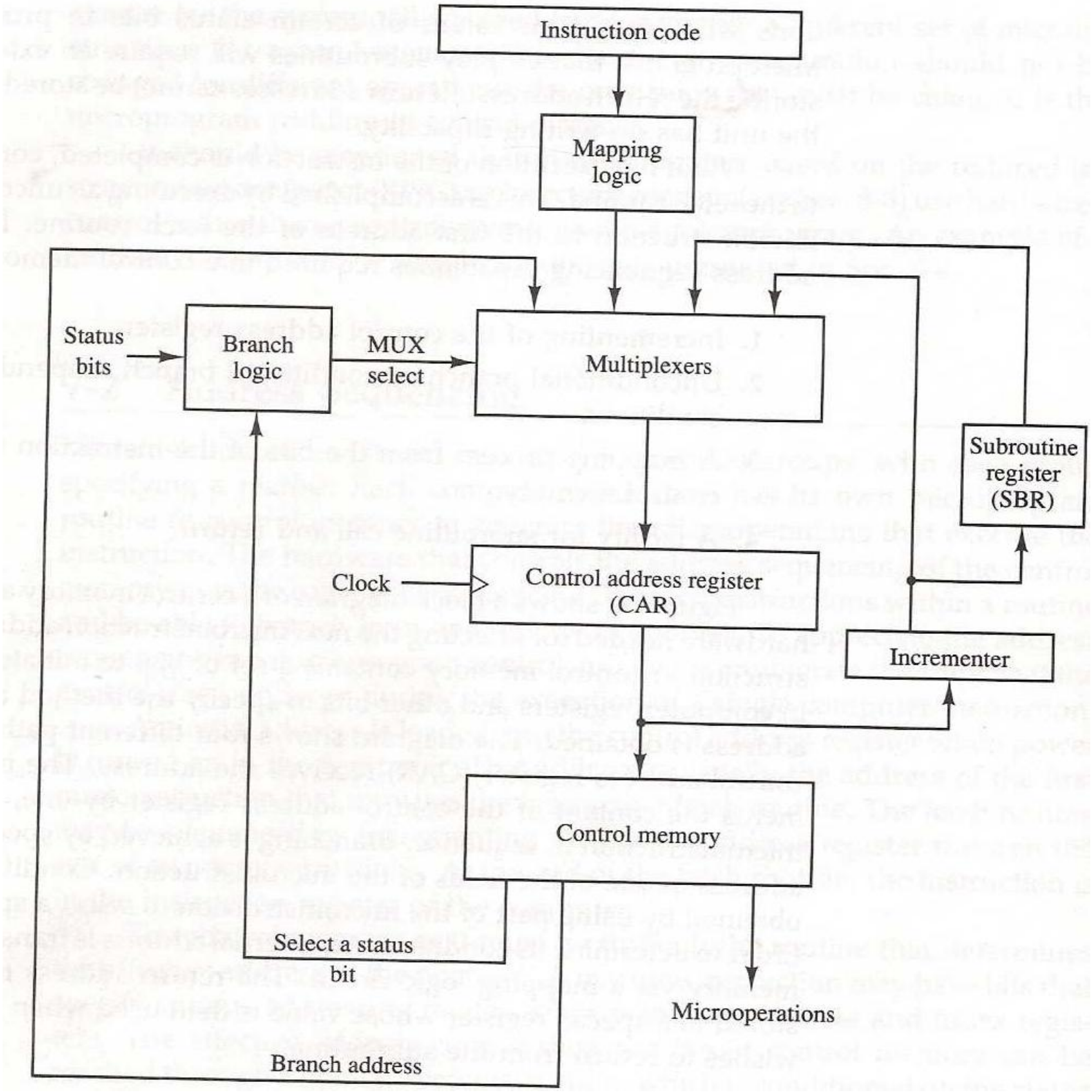
<u>2. Address Sequencing:</u>

- Microinstructions are stored in control memory in groups, with each group specifying a *routine.*
- Each computer instruction has its own microprogram routine to generate the microoperations.
- The hardware that controls the address sequencing of the control memory must be capable of sequencing the microinstructions within a routine and be able to branch from one routine to another
- Steps the control must undergo during the execution of a single computer instruction:
    - Load an initial address into the CAR when power is turned on in the computer. This address is usually the address of the first microinstruction that activates the instruction fetch routine – IR holds instruction
    - The control memory then goes through the routine to determine the effective address of the operand – AR holds operand address
    - The next step is to generate the microoperations that execute the instruction by considering the opcode and applying a *mapping process.*
        - *The transformation of the instruction code bits to an address in control memory where the routine of instruction located is referred to as mapping process.*

    - After execution, control must return to the fetch routine by executing an unconditional branch
- In brief the address sequencing capabilities required in a control memory are:
    - Incrementing of the control address register.
    - Unconditional branch or conditional branch, depending on status bit conditions.
    - A mapping process from the bits of the instruction to an address for control memory.

- A facility for subroutine call and return.

- The below figure shows a block diagram of a control memory and the associated hardware needed for selecting the next microinstruction address.



- The microinstruction in control memory contains a set of bits to initiate microoperations in computer registers and other bits to specify the method by which the next address is obtained.
- In the figure four different paths form which the control address register (CAR) receives the address.
  - The incrementer increments the content of the control register address register by one, to select the next microinstruction in sequence.
  - Branching is achieved by specifying the branch address in one of the fields of the microinstruction.
  - Conditional branching is obtained by using part of the microinstruction to select a specific status bit in order to determine its condition.
  - An external address is transferred into control memory via a mapping logic circuit.
  - The return address for a subroutine is stored in a special register, that value is used when the micoprogram wishes to return from the subroutine.

Conditional Branching:

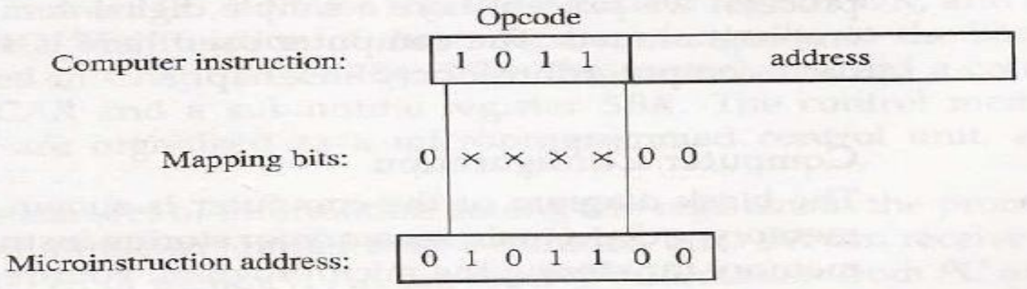
- Conditional branching is obtained by using part of the microinstruction to select a specific status bit in order to determine its condition.
- The status conditions are special bits in the system that provide parameter information such as the carry-out of an adder, the sign bit of a number, the mode bits of an instruction, and i/o status conditions.
- The status bits, together with the field in the microinstruction that specifies a branch address, control the branch logic.

- The branch logic tests the condition, if met then branches, otherwise, increments the CAR.
- If there are 8 status bit conditions, then 3 bits in the microinstruction are used to specify the condition and provide the selection variables for the multiplexer.
- For unconditional branching, fix the value of one status bit to be one load the branch address from control memory into the CAR.

Mapping of Instruction:

- A special type of branch exists when a microinstruction specifies a branch to the first word in control memory where a microprogram routine is located.
- The status bits for this type of branch are the bits in the opcode.
- Assume an opcode of four bits and a control memory of 128 locations. The mapping process converts the 4-bit opcode to a 7-bit address for control memory shown in below figure.

Figure 7-3 Mapping from instruction code to microinstruction address.



- Mapping consists of placing a 0 in the most significant bit of the address, transferring the four operation code bits, and clearing the two least significant bits of the control address register.
- This provides for each computer instruction a microprogram routine with a capacity of four microinstructions.

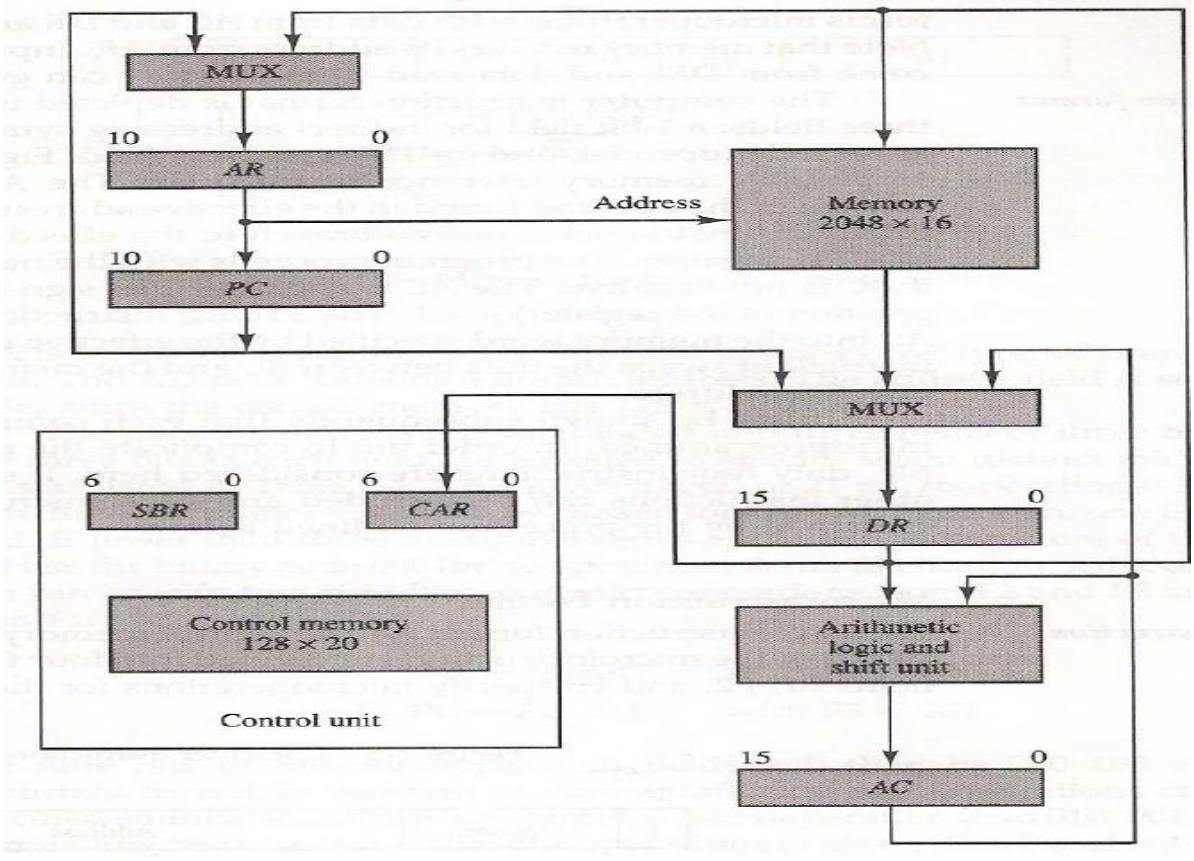
Subroutines:

- Subroutines are programs that are used by other routines to accomplish a particular task and can be called from any point within the main body of the microprogram.
- Frequently many microprograms contain identical section of code.
- Microinstructions can be saved by employing subroutines that use common sections of microcode.
- Microprograms that use subroutines must have a provision for storing the return address during a subroutine call and restoring the address during a subroutine return.
- A subroutine register is used as the source and destination for the addresses

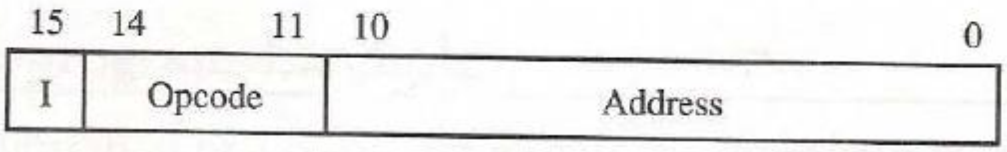
<u>3. Microprogram Example:</u>

- The process of code generation for the control memory is called *microprogramming*.
- The block diagram of the computer configuration is shown in below figure.
- Two memory units:
    - Main memory – stores instructions and data
    - Control memory – stores microprogram
- Four processor registers
    - Program counter – PC
    - Address register – AR
    - Data register – DR
    - Accumulator register - AC
- Two control unit registers
    - Control address register – CAR
    - Subroutine register – SBR

- Transfer of information among registers in the processor is through MUXs rather than a bus.



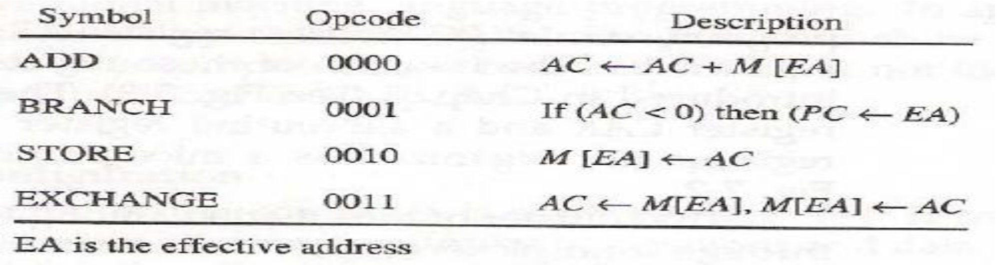
- The computer instruction format is shown in below figure.



(a) Instruction format

- Three fields for an instruction:
  - 1-bit field for indirect addressing
  - 4-bit opcode
  - 11-bit address field

- The example will only consider the following 4 of the possible 16 memory instructions

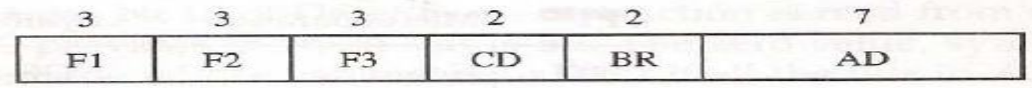


Symbol	Opcode	Description
ADD	0000	$AC \leftarrow AC + M[EA]$
BRANCH	0001	If $(AC < 0)$ then $(PC \leftarrow EA)$
STORE	0010	$M[EA] \leftarrow AC$
EXCHANGE	0011	$AC \leftarrow M[EA], M[EA] \leftarrow AC$

EA is the effective address

(b) Four computer instructions

- The microinstruction format for the control memory is shown in below figure.



F1, F2, F3: Microoperation fields

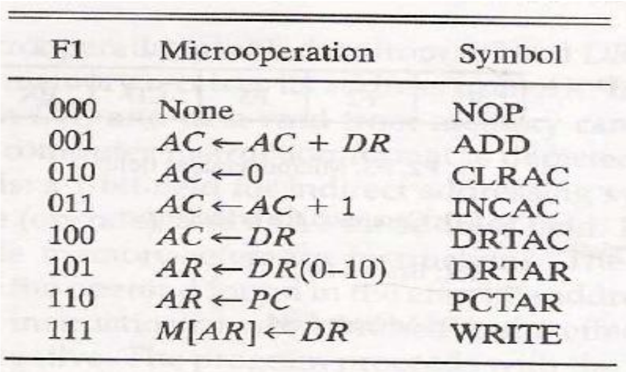
CD: Condition for branching

BR: Branch field

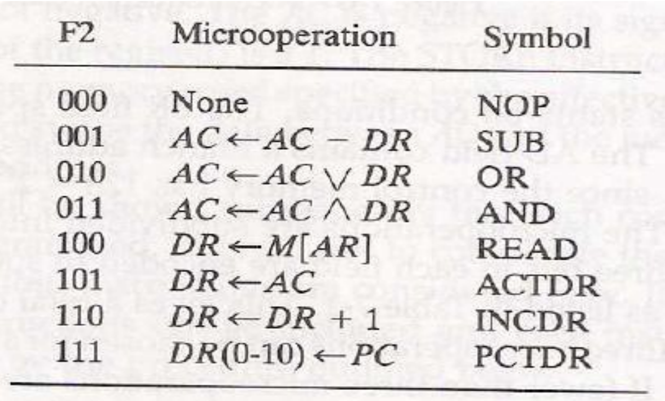
AD: Address field

Figure 7-6 Microinstruction code format (20 bits).

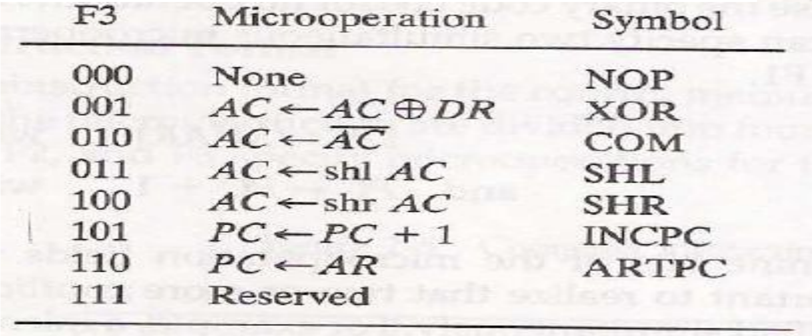
- ➢ The microinstruction format is composed of 20 bits with four parts to it
  - Three fields F1, F2, and F3 specify microoperations for the computer [3 bits each]
  - The CD field selects status bit conditions [2 bits]
  - The BR field specifies the type of branch to be used [2 bits]
  - The AD field contains a branch address [7 bits]
- ➢ Each of the three microoperation fields can specify one of seven possibilities.
- ➢ No more than three microoperations can be chosen for a microinstruction.
- ➢ If fewer than three are needed, the code 000 = NOP.
- ➢ The three bits in each field are encoded to specify seven distinct microoperations listed in below table.



F1	Microoperation	Symbol
000	None	NOP
001	$AC \leftarrow AC + DR$	ADD
010	$AC \leftarrow 0$	CLRAC
011	$AC \leftarrow AC + 1$	INCAC
100	$AC \leftarrow DR$	DRTAC
101	$AR \leftarrow DR(0\text{–}10)$	DRTAR
110	$AR \leftarrow PC$	PCTAR
111	$M[AR] \leftarrow DR$	WRITE



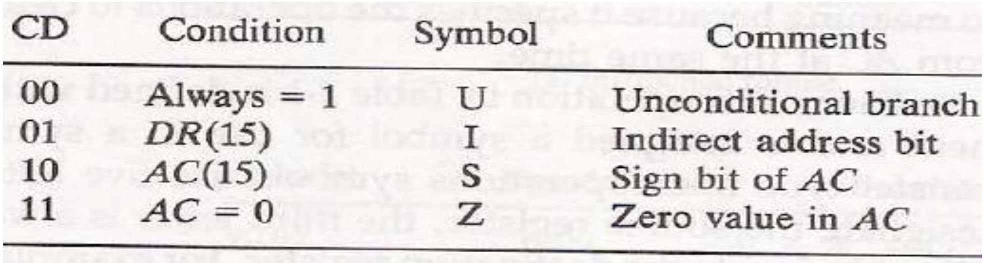
F2	Microoperation	Symbol
000	None	NOP
001	$AC \leftarrow AC - DR$	SUB
010	$AC \leftarrow AC \vee DR$	OR
011	$AC \leftarrow AC \wedge DR$	AND
100	$DR \leftarrow M[AR]$	READ
101	$DR \leftarrow AC$	ACTDR
110	$DR \leftarrow DR + 1$	INCDR
111	$DR(0\text{-}10) \leftarrow PC$	PCTDR



F3	Microoperation	Symbol
000	None	NOP
001	$AC \leftarrow AC \oplus DR$	XOR
010	$AC \leftarrow \overline{AC}$	COM
011	$AC \leftarrow \text{shl } AC$	SHL
100	$AC \leftarrow \text{shr } AC$	SHR
101	$PC \leftarrow PC + 1$	INCPC
110	$PC \leftarrow AR$	ARTPC
111	Reserved	

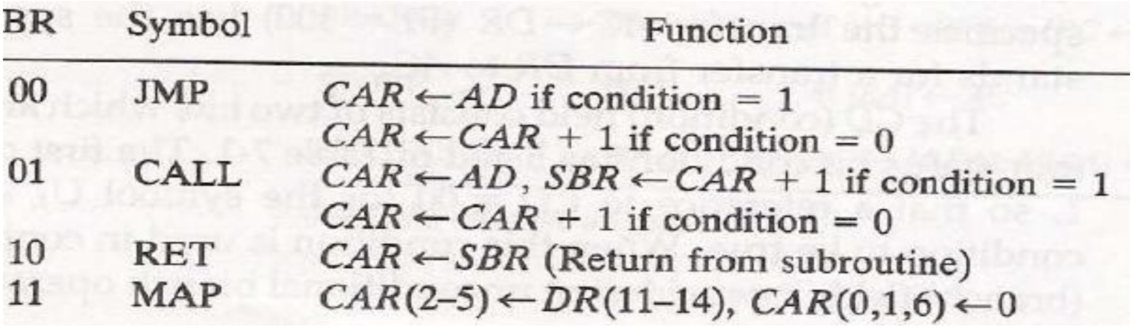
- ➢ Five letters to specify a transfer-type microoperation
  - First two designate the source register
  - Third is a ‘T’
  - Last two designate the destination register

  AC ← DR F1 = 100   = DRTAC
- ➢ The condition field (CD) is two bits to specify four status bit conditions shown below



CD	Condition	Symbol	Comments
00	Always = 1	U	Unconditional branch
01	$DR(15)$	I	Indirect address bit
10	$AC(15)$	S	Sign bit of AC
11	$AC = 0$	Z	Zero value in AC

- ➢ The branch field (BR) consists of two bits and is used with the address field to choose the address of the next microinstruction.

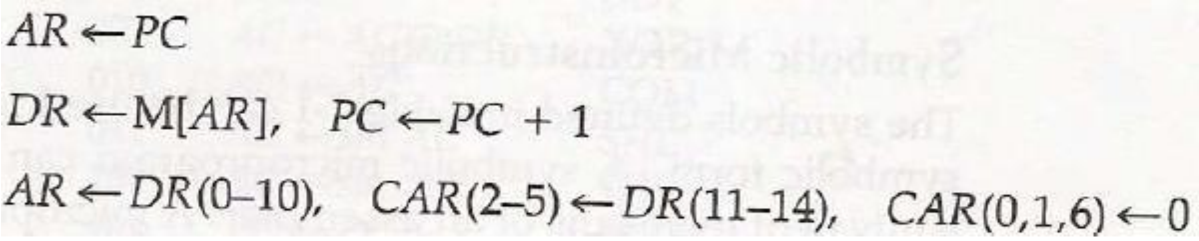


BR	Symbol	Function
00	JMP	$CAR \leftarrow AD$ if condition = 1 $CAR \leftarrow CAR + 1$ if condition = 0
01	CALL	$CAR \leftarrow AD, SBR \leftarrow CAR + 1$ if condition = 1 $CAR \leftarrow CAR + 1$ if condition = 0
10	RET	$CAR \leftarrow SBR$ (Return from subroutine)
11	MAP	$CAR(2\text{–}5) \leftarrow DR(11\text{–}14),\ CAR(0,1,6) \leftarrow 0$

- Each line of an assembly language microprogram defines a symbolic microinstruction and is divided into five parts
  1. The label field may be empty or it may specify a symbolic address. Terminate with a colon (:).
  2. The microoperations field consists of 1-3 symbols, separated by commas. Only one symbol from each field. If NOP, then translated to 9 zeros
  3. The condition field specifies one of the four conditions
  4. The branch field has one of the four branch symbols
  5. The address field has three formats
     a. A symbolic address – must also be a label
     b. The symbol NEXT to designate the next address in sequence
     c. Empty if the branch field is RET or MAP and is converted to 7 zeros
- The symbol ORG defines the first address of a microprogram routine.
- ORG 64 – places first microinstruction at control memory 1000000.

Fetch Routine:

- The control memory has 128 locations, each one is 20 bits.
- The first 64 locations are occupied by the routines for the 16 instructions, addresses 0-63.
- Can start the fetch routine at address 64.
- The fetch routine requires the following three microinstructions (locations 64-66).
- The microinstructions needed for fetch routine are:

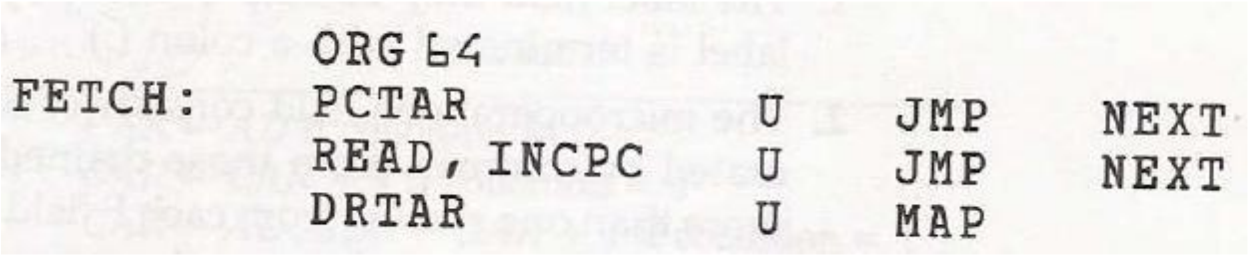


$$AR \leftarrow PC$$

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

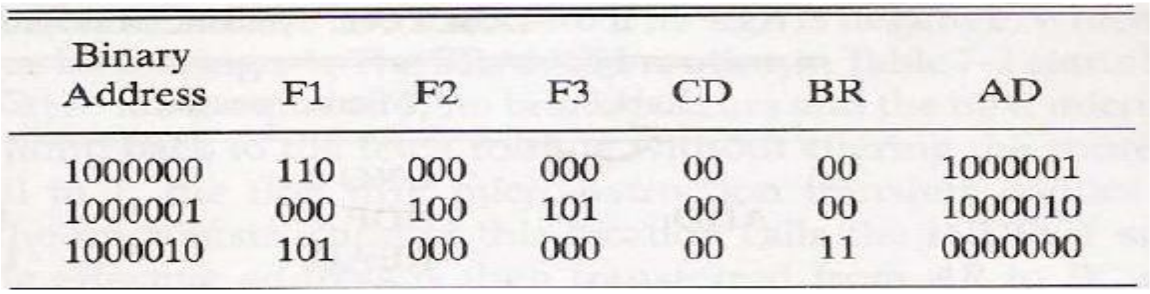
$$AR \leftarrow DR(0\text{–}10),\quad CAR(2\text{–}5) \leftarrow DR(11\text{–}14),\quad CAR(0,1,6) \leftarrow 0$$

- It's Symbolic microprogram:



```
         ORG 64
FETCH:   PCTAR            U    JMP    NEXT
         READ, INCPC      U    JMP    NEXT
         DRTAR            U    MAP
```

- It's Binary microprogram:



Binary Address	F1	F2	F3	CD	BR	AD
1000000	110	000	000	00	00	1000001
1000001	000	100	101	00	00	1000010
1000010	101	000	000	00	11	0000000