MODULE 3

MICROPROGRAMMED CONTROL

Dr. Chhaya Suryabhan Dule
Associate Professor
School of CSE
Sapthagiri NPS University ,
BANGALORE

SYLLABUS

Microprogrammed Control : Microprogrammed Control Unit Definition ,Control Memory ,Address Sequencing ,Conditional Branching ,Mapping of Instruction ,Subroutine , Microprogram example , Computer Configuration ,Microinstruction format, Symbolic microinstruction ,The fetch routine, Symbolic microprogram .

Micro-operations : A basic elementary operation performed on data stored in registers or memory within computers central processing Unit (CPU) during a single clock cycle.

Load : Transferring data from memory to a register

Store : Transferring data from a register to memory

Arithmetic : Adding ,Subtracting ,incrementing or decrementing values in registers

Logical : Performing AND ,OR ,NOT ,XOR operations on bits in registers

Shift : Moving bits left or right within a register

Count : Incrementing or decrementing a register value

Clear : Setting a register value to zero

Rotate : Moving bits within a register circularizing the bits

These Micro-operations are the building blocks for more complex instructions executed by a computer

Micro-operations : In executing program , operations of a computer consists of a sequence of instruction cycles ,with one machine instruction per cycle .

Each Instruction cycle is made up of a number of smaller units

1. **Fetch**
2. **Decode**
3. **Execute**

Each of these cycles involves series of steps, each of which involves the processor registers. These steps are referred as micro-operations. The prefix micro refers to the fact that each of the step is very simple.

Control Unit - The function of the control unit in a digital computer is to **initiate sequences of microoperations.**

Finite number of different types of microperations are available in a system

Complexity of Digital system is derived from number of sequences of micro operations that are performed.

Two Major types of Control Unit

1. **Hardwired Control Unit**
2. **Microprogrammed Control Unit**

I. Hardwired Control Unit:

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

- High speed of operation
- Expensive
- Relatively complex
- No flexibility of adding new instructions
- Examples of CPU with hardwired control unit are Intel 8085, Motorola 6802, Zilog 80, and any RISC CPUs.

II. Micro Programmed Control Unit:

Microprogramming is a alternative for designing control unit of a digital computer based on the principle of controlling micro operations sequences in a digital computer

- Control function specifies a micro-operation as a binary variable
- When binary variable is in one binary state corresponding microperation is executed
- Control variable in the opposite binary state of a control variable doesnot change the state of the register in the system
- Active state of control variable may be either 1 state or 0 state depending on applications

II. Micro Programmed Control Unit:

A control unit whose binary control variables are stored in memory is called a micro programmed control unit.

- Control information is stored in control memory.
- Control memory is programmed to initiate the required sequence of micro-operations.

The key characteristics are

- Speed of operation is low when compared with hardwired
- Less complex and Less Expensive
- Flexibility to add new instructions
- Examples of CPU with microprogrammed control unit are Intel 8080, Motorola 68000 and any CISC CPUs.

Microoperation, Microinstruction, Micro program, Microcode

Microoperations:

• In computer central processing units, micro-operations (also known as a micro-ops or μops) are detailed low-level instructions used in some designs to implement complex machine instructions (sometimes termed macro-instructions in this context).

Micro instruction:

• **Each word** in control memory contain within its microinstruction

• Microinstruction specifies one or more microperation for the system

• A sequence of microinstruction constitute microprogram

• Alteration of microprogram are not needed once the control unit is in operation, the control can be read only memory(ROM)

Microoperation, Microinstruction, Micro program, Microcode

Micro instruction:

- Contents of words in ROM are fixed and made permanent during hardware production of unit
- Use of microprogram involves placing all control variables in words of ROM for use by control unit through successive read operations.
- Content of word in ROM at a given address specifies a microinstruction

Dynamic microprogramming:

A more advanced development known as dynamic microprogramming permits a microprogram to be loaded initially from an auxiliary memory such as a magnetic disk. Control units that use dynamic microprogramming employ a writable control memory. This type of memory can be used for writing. A memory that is part of control unit is referred as Control Memory

Control Memory:

Control Memory is the storage in the microprogrammed control unit to store the microprogram which can not be altered by user.

Writeable Control Memory:

Control Storage whose contents can be modified, allow the change in microprogram and Instruction set can be changed or modified is referred as Writeable Control Memory

Control Word:

The control variables at any given time can be represented by a control word string of 1 's and 0's called a control word.

Micro program:

• A sequence of microinstructions constitutes a microprogram.

• Since alterations of the microprogram are not needed once the control unit is in operation, the control memory can be a read-only memory (ROM).

• ROM words are made permanent during the hardware production of the unit.

• The use of a micro program involves placing all control variables in words of ROM for use by the control unit through successive read operations.

• The content of the word in ROM at a given address specifies a microinstruction.

Dynamic microprogramming:

Control Memory = RAM

• RAM can be used for writing (to change a writable control memory)

• Microprogram is loaded initially from an auxiliary memory such as a magnetic disk

Static microprogramming:

Control Memory = ROM

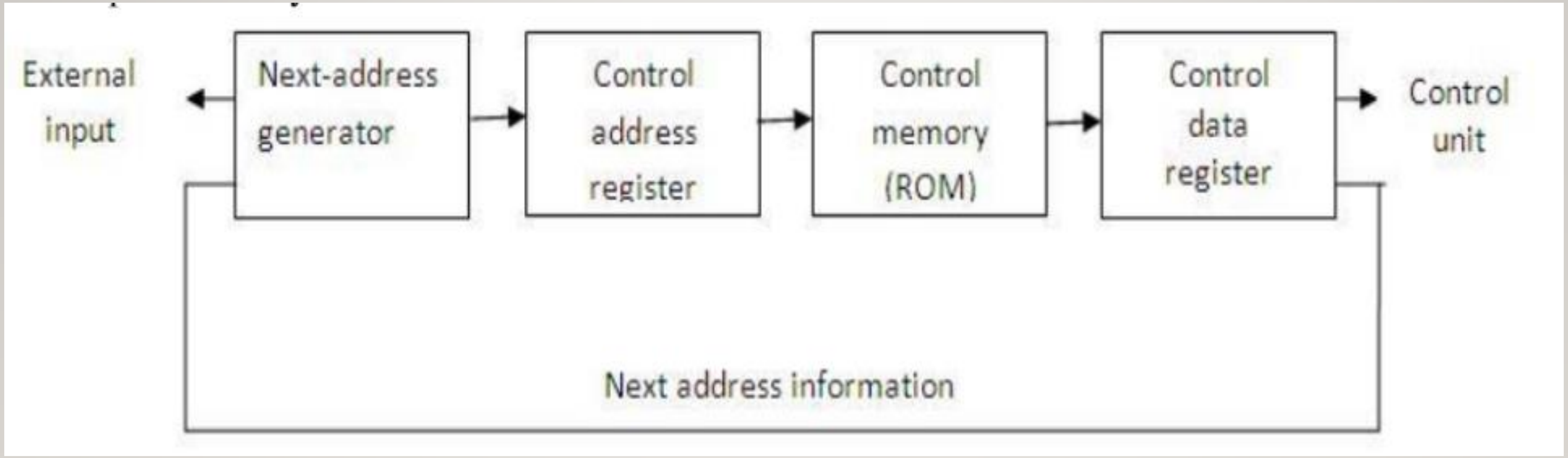
• Control words in ROM are made permanent during the hardware production.

Microprogram:

• Microprogram consists of micro instruction that specify various internal control signals for execution of register microoperations.

1. To fetch the instruction from main memory
2. To evaluate the effective address
3. To execute the operation specified by instruction
4. Return control to fetch phase in order to repeat cycle for next instruction

Microprogrammed control Organization



1. Control Memory

- Control memory is ROM ,within which all control information is permanently stored as Microinstruction.
- Allows the execution of the microoperations specified by the control word simultaneously with the generation of the next microinstruction.
- This requires two phase clock, with one clock applied to address Register and other to data register.

2. Control Memory Address Register :

- Specifies address of micro instruction

3. Control Data Register (Pipeline Register)

- Control data register contain control word that specifies one or more micro-operations for data processor.
- Once these operations are executed ,the control must determine next address

3. Control Data Register (Pipeline Register)

- It allows execution of the micro operation specified by the control word simultaneously with generation of next micro instruction
- It requires two phase clock with one clock applied to address register and other to data register

4. Sequencer (Next Address Generator)

- Location of next instruction may be present in some bits of microinstruction.
- The next instruction address may be function of external input conditions.
- While micro operations are being executed ,the next address is computed in the address generator circuit and then transferred to control address register to read next instruction.

4. Sequencer (Next Address Generator)

- Next address of the next microinstruction can be specified several way depending on the sequencer input.
  1. Incrementing control register by one
  2. Loading into the control address register ,an address from control memory transferring an external address

- **Main Advantage of Microprogrammed Control**

  1. Once the hardware configuration is established ,there should be no need for further hardware or writing changes
  2. To establish different control sequence for the system, we need to specify different set of microinstructions
  3. The hardware configuration should not be changed for different operations ,only microprogram in control memory will be changed.

Address Sequencing:

- Microinstructions are stored in control memory in group with each group specifying a Routine.
- Each computer instruction has its own microprogram routine in control memory to generate micro operation that execute the instruction
- Hardware that control address sequencing of control memory must be capable of sequencing micro instruction within routine and able to branch from one routine to another

Address Sequencing:

- Initial address is located in control address register when power is turned on the computer
- Address is usually the address of first microinstruction that activates instruction fetch routine
- The fetch routine may be sequenced by incrementing control address register through rest of its microinstructions
- At the end of fetch routine ,the instruction is in instruction register of the processor
- Control memory next must go through routine that determine effective address of operand

Address Sequencing:

- Machine instruction may have bits that specify various addressing modes such as indirect address and index register
- When the effective address computation routine is completed ,address of operand is available in MAR
- Effective address computation routine in control memory can be reached through branch microinstruction which is conditioned on status of mode bits of instruction
- Next step is to generate microoperations that execute instruction fetched from memory
- The microoperation steps to be generated in processor register depend upon opcode of instruction

Address Sequencing:

- Each instruction has its own microprogram routine stored in a given location of control memory
- Transformation from instruction code bits to an address in control memory where the routine is located is referred as **Mapping Process**
- Once required routine is reached ,the microinstruction that execute instruction may be sequenced by incrementing control address register but sometimes sequence of microoperation depend on value of status bits in processor register
- Microprogram that employ subroutine will require an external register for storing return address

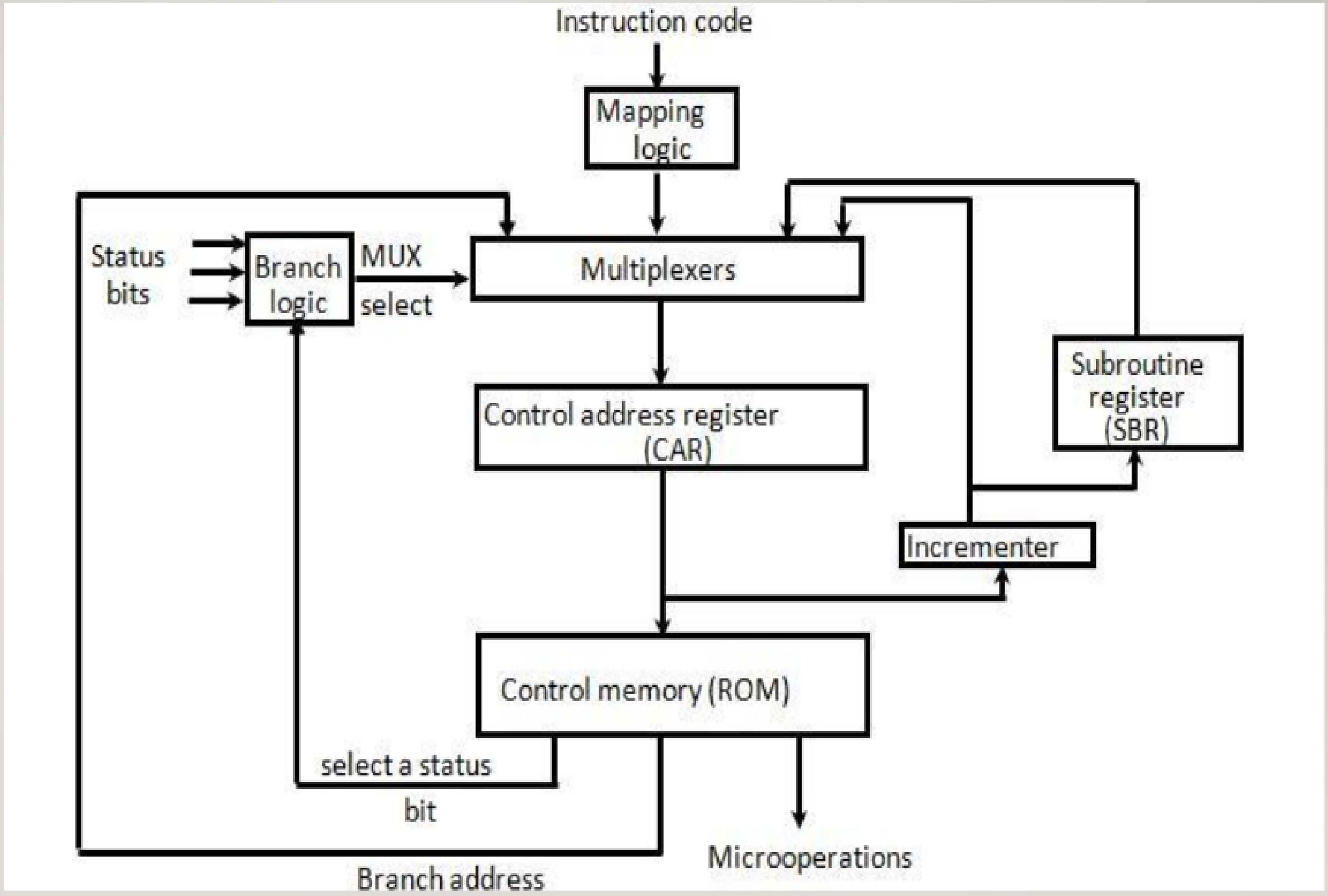
Address Sequencing:

- Return address can not be stored in ROM because it has no writing capability
- When execution of instruction is completed control must return to fetch routine accomplished by executing an unconditional branch microinstruction to the first address of fetch routine

The address sequencing capabilities required in a control memory are:

1. Incrementing of the control address register.
2. Unconditional branch or conditional branch, depending on status bit conditions.
3. A mapping process from the bits of the instruction to an address for control memory.
4. A facility for subroutine call and return.

Address Sequencing



Selection of Address for Control Memory

Conditional Branching:

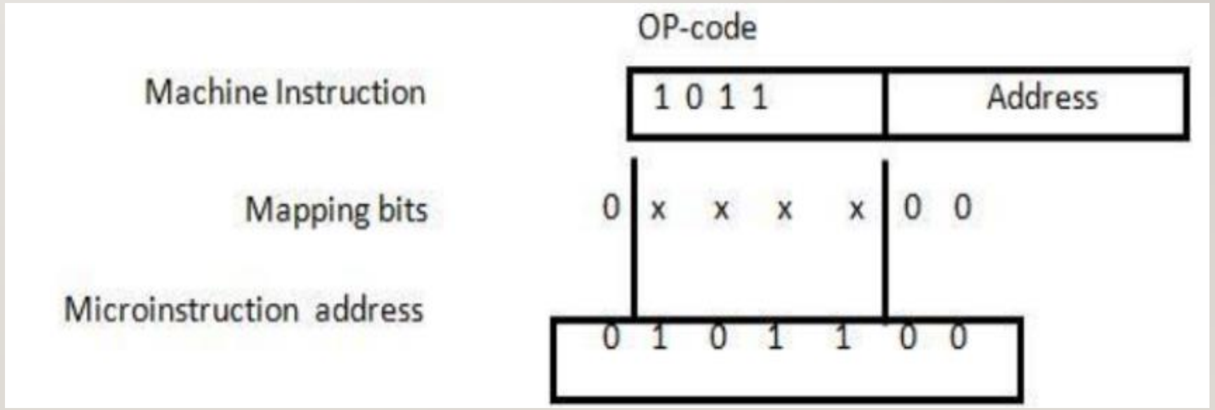
- The branch logic provides decision-making capabilities in the control unit.
- The status conditions are special bits in the system that provide parameter information such as the carry-out of an adder, the sign bit of a number, the mode bits of an instruction, and input or output status conditions.
- The status bits, together with the field in the microinstruction that specifies a branch address, control the conditional branch decisions generated in the branch logic.
- A 1 output in the multiplexer generates a control signal to transfer the branch address from the microinstruction into the control address register.
- A 0 output in the multiplexer causes the address register to be incremented.
- In this configuration, the microprogram follows one of two possible paths, depending on the value of the selected status bit.

Unconditional Branch:

- An Unconditional Branch microinstruction can be implemented by loading the branch address from control memory into the control address register.
- This can be accomplished by fixing the value of one status bit at the input of the multiplexer, so it is always equal to 1.

Mapping of an Instruction:

- A special type of branch exists when a microinstruction specifies a branch to the first word in control memory where a microprogram routine for an instruction is located.
- The status bits for this type of branch are the bits in the operation code part of the instruction. For example, a computer with a simple instruction format an operation code of four bits which can specify up to 16 distinct instructions.
- Assume that the control memory has 128 words, requiring an address of seven bits.
- One simple mapping process that converts the 4-bit operation code to a 7-bit address for control memory

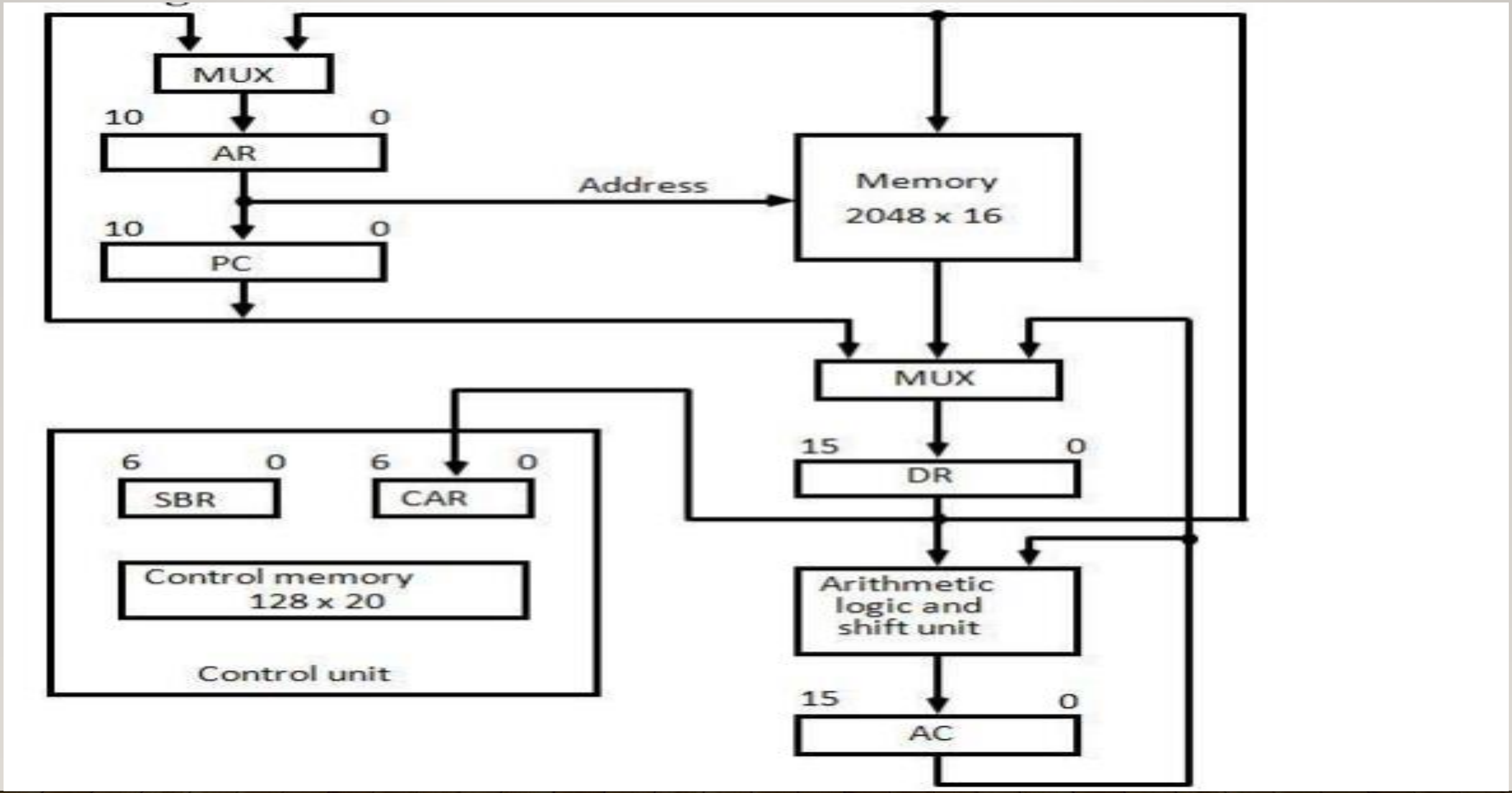


Mapping of an Instruction:

- This 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.
- If the routine needs more than four microinstructions, it can use addresses 1000000 through 1111111. If it uses fewer than four microinstructions, the unused memory locations would be available for other routines.
- In this way the microprogram routine that executes the instruction can be placed in any desired location in control memory.
- The mapping concept provides flexibility for adding instructions for control memory as the need arises.

MICROPROGRAM EXAMPLE

Computer Hardware Configuration



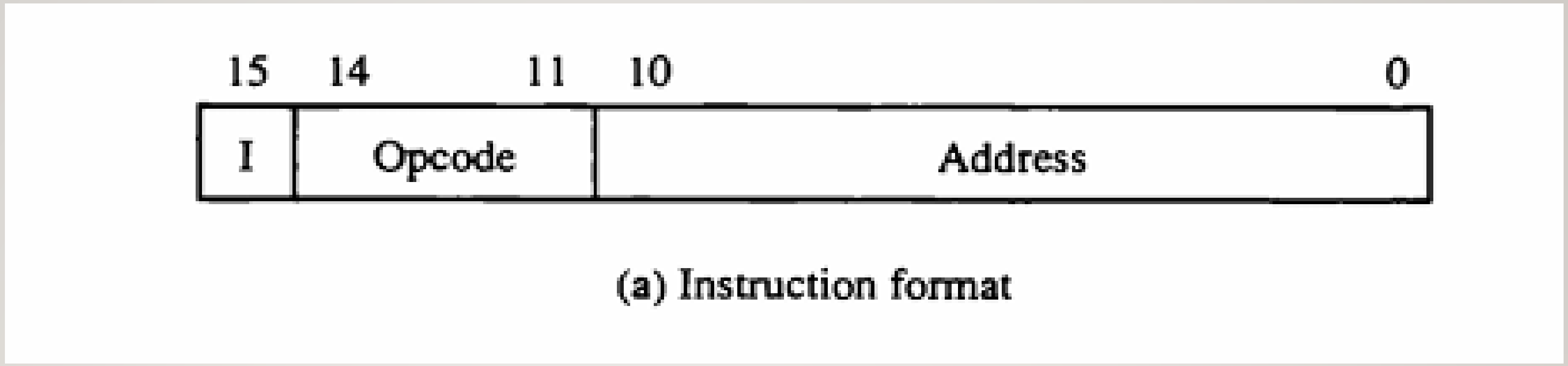
- It consists of two memory units:

  Main memory - for storing instructions and data

  Control memory - for storing the microprogram.

- Six Registers:
  - Processor unit register: AC (accumulator),PC (Program Counter), AR (Address Register), DR (Data Register)
  - Control unit register: CAR (Control Address Register), SBR (Subroutine Register)
- Multiplexers:
  - The transfer of information among the registers in the processor is done through multiplexers rather than a common bus.
- The arithmetic, logic, and shift unit perform microoperations with data from AC and DR and places the result in AC.
- DR can receive information from AC, PC, or memory.
- AR can receive information from PC or DR.
- PC can receive information only from AR.
- Input data written to memory come from DR, and data read from memory can go only to DR.

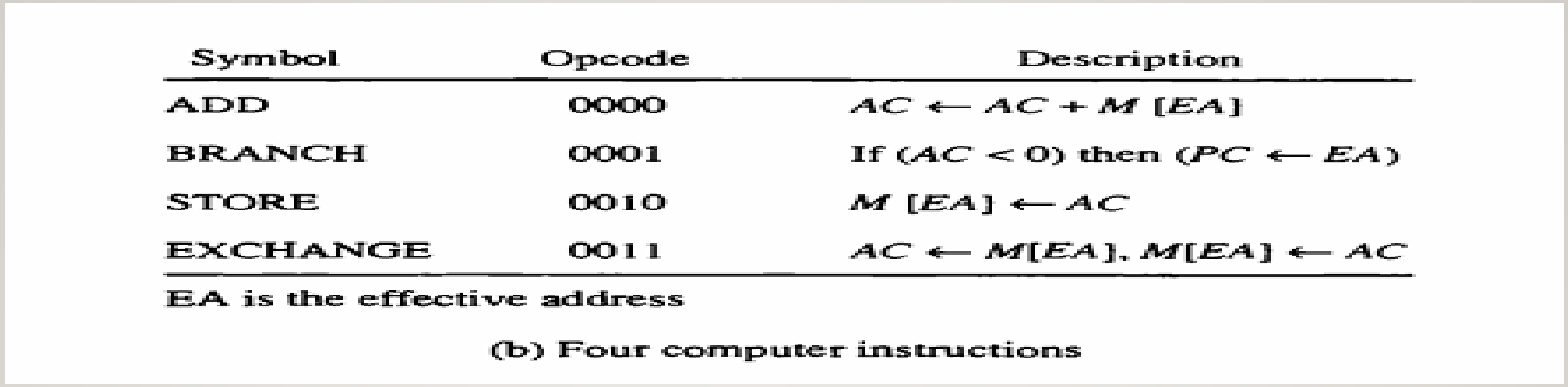
- **Instruction Format**



(a) Instruction format

- **It consist of 1 bit field for Indirect Addressing.**
- **4 bit operation code**
- **11 Bit address field**

- **Memory –Reference Instruction**



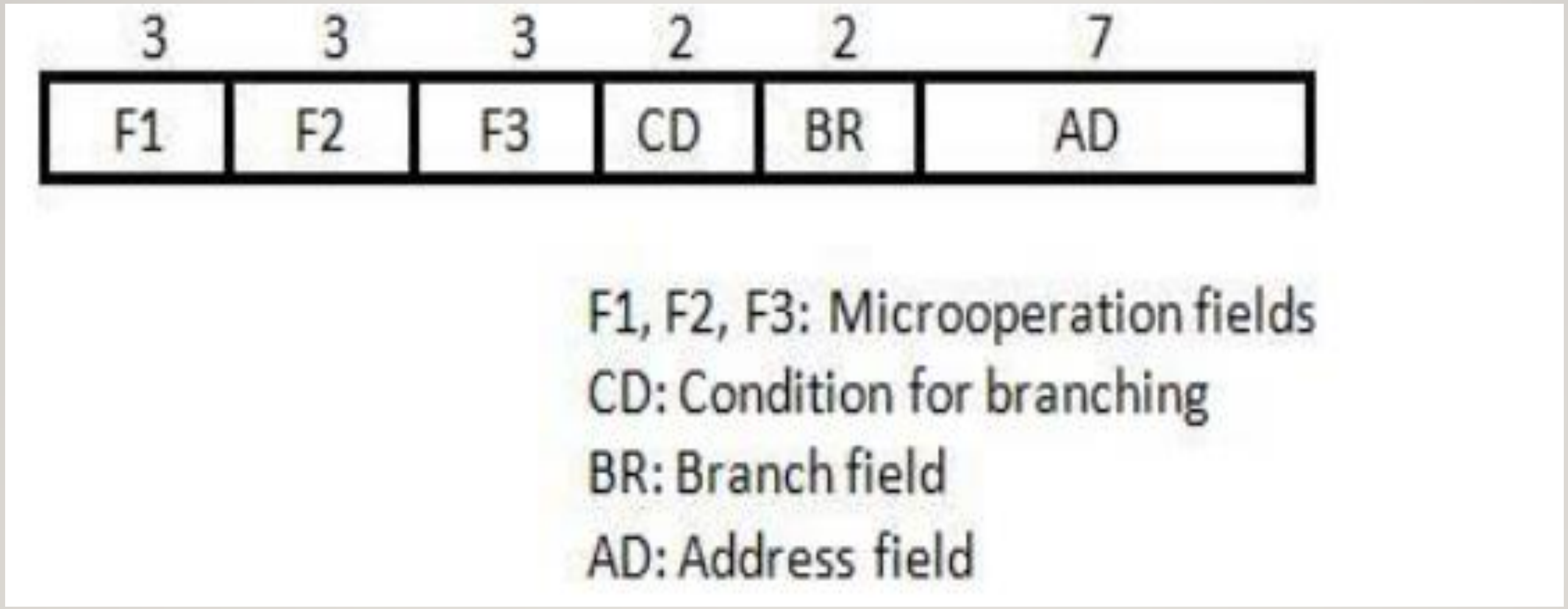
Symbol	Opcode	Description
ADD	0000	$AC \leftarrow AC + M[EA]$
BRANCH	0001	If $(AC < 0)$ then $(PC \leftarrow EA)$
STORE	0010	$M[EA] \leftarrow AC$
EXCHANGE	0011	$AC \leftarrow M[EA]$, $M[EA] \leftarrow AC$

EA is the effective address

(b) Four computer instructions

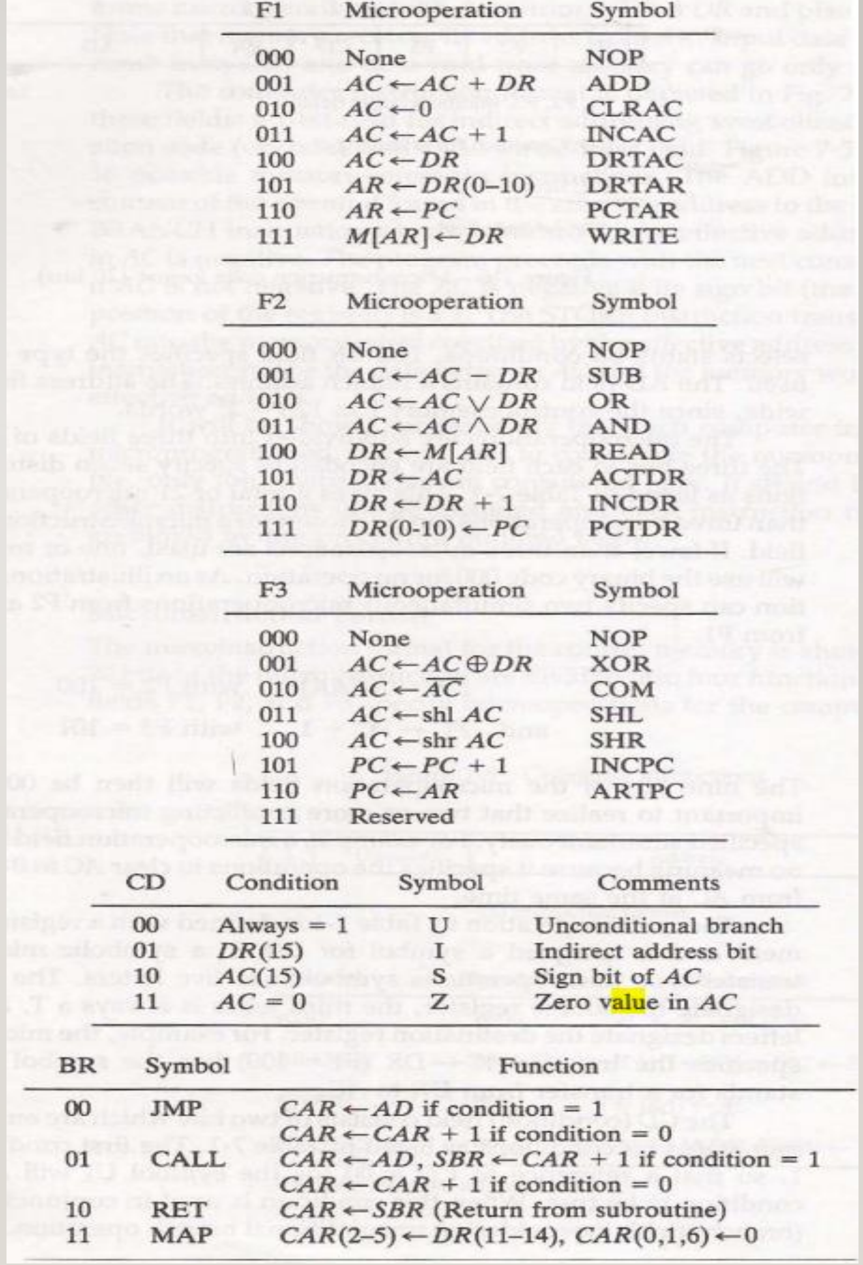
- **Add instruction adds the content of the operand found in the effective address of memory to the content of AC.**
- **Branch instruction causes branch to the effective address if the operand in AC is negative,the program proceeds with the next consecutive instruction if AC is not negative, The AC is negative if its sign bit is 1**
- **The Store instruction transfers content of AC into the memory word specified by the effective address**
- **The EXCHANGE instruction swaps the data between AC and the memory word specified by effective address**

Microinstruction Format



F1, F2, F3: Microoperation fields
CD: Condition for branching
BR: Branch field
AD: Address field

- **The 20 bits of the microinstruction are divided into four functional parts.**
- **The three fields Fl, F2, and F3 specify microoperations for the computer.**
- **The CD field selects status bit conditions - consists of 2 bits needed to specify 4 status bit conditions. Symbols U,I,S and Z for the 4 status bits when we write microprograms in symbolic form**
- **The BR field specifies the type of branch to be used - to choose the address of the next microinstruction.**
- **The AD Field contains a branch address. The address field is seven bits wide, since the control memory has 128 = 2^7 words.**



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

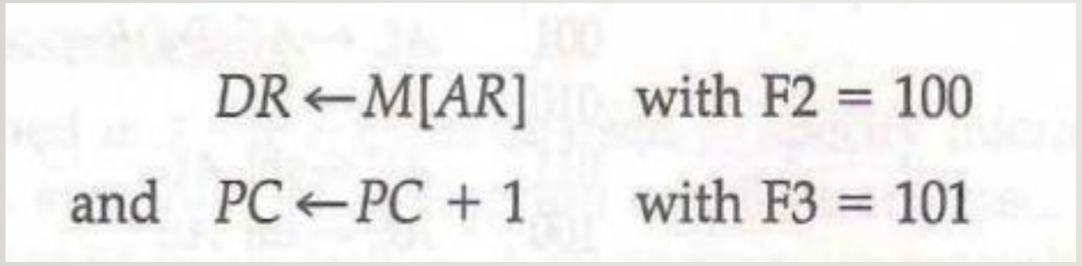
F2	Microoperation	Symbol
000	None	NOP
001	$AC \leftarrow AC - DR$	SUB
010	$AC \leftarrow AC \vee DR$	OR
011	$AC \leftarrow AC \wedge DR$	AND
100	$DR \leftarrow M[AR]$	READ
101	$DR \leftarrow AC$	ACTDR
110	$DR \leftarrow DR + 1$	INCDR
111	$DR(0\text{-}10) \leftarrow PC$	PCTDR

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

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

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

- **The microoperations are subdivided into three fields of three bits each the nine bits of the microoperation fields will then be 000 100 101.**
- **The three bits in each field are encoded to specify seven distinct micro-operations.**
- **As an example, a microinstruction can specify two simultaneous micro-operations from F2 and F3 and none from F1.**

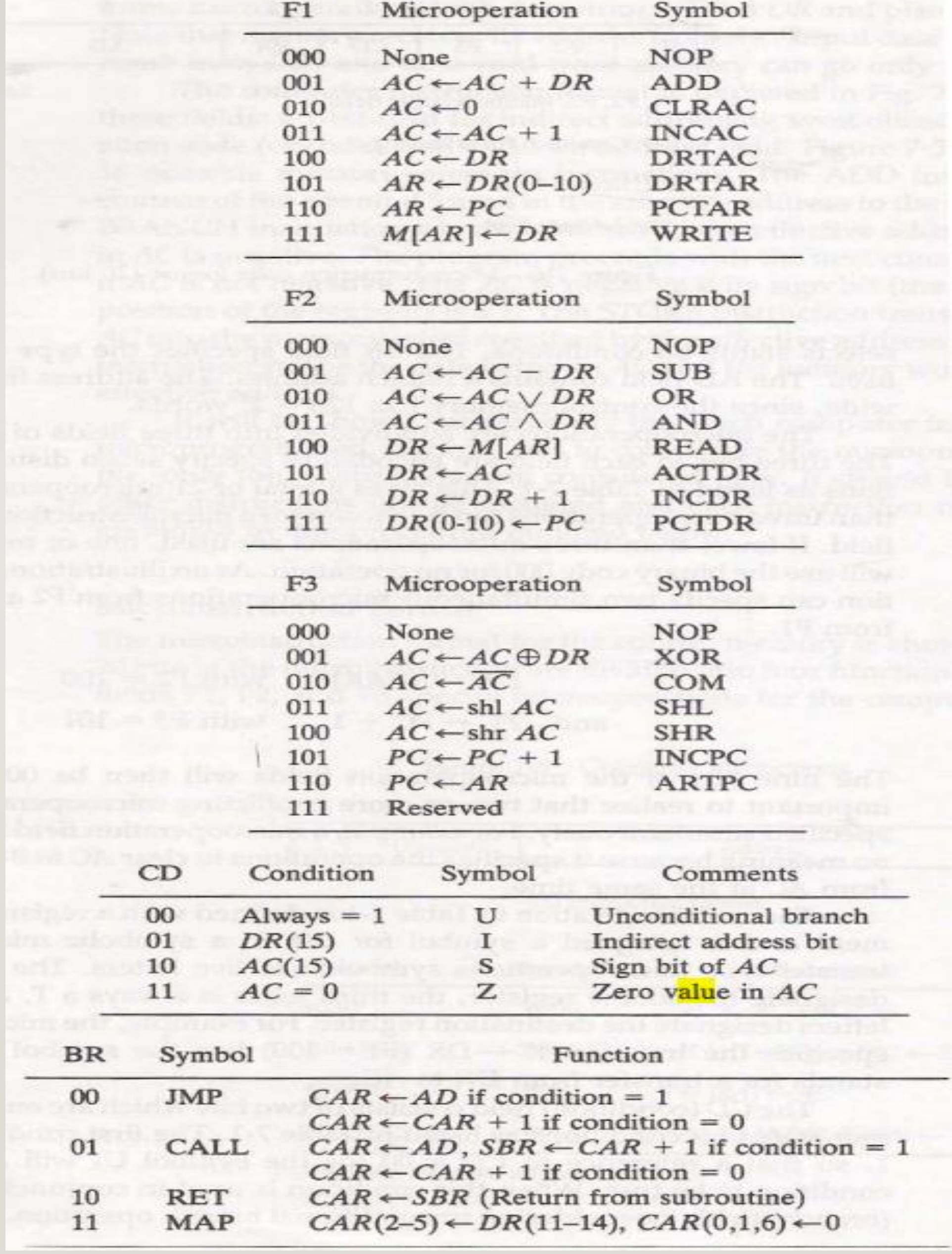


$$DR \leftarrow M[AR] \quad \text{with F2} = 100$$

$$\text{and} \quad PC \leftarrow PC + 1 \quad \text{with F3} = 101$$

- **All transfer-type micro-operations symbols use five letters. The first two letters designate the source register, the third letter is always a T, and the last two letters designate the destination register.**

  Example: DRTAC means DR to AC



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	

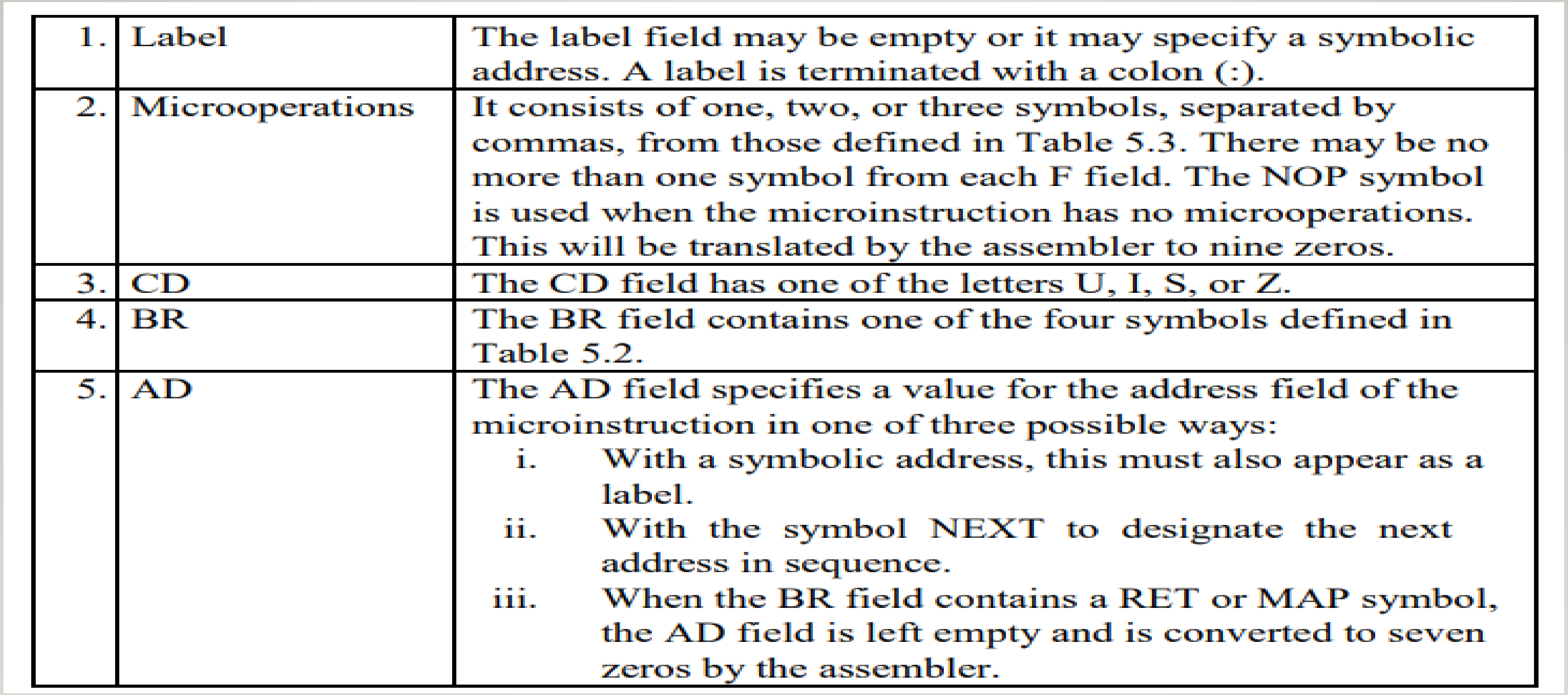
CD	Condition	Symbol	Comments
00	Always = 1	U	Unconditional branch
01	$DR(15)$	I	Indirect address bit
10	$AC(15)$	S	Sign bit of AC
11	$AC = 0$	Z	Zero value in AC

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

Symbolic Microinstruction

- A symbolic microprogram can be translated into its binary equivalent by means of an assembler .
- The Simplest and most straightforward way to formulate an assembly language for a microprogram is to define symbols for each field of the microinstruction and to give users the capability for defining their own symbolic addresses.
- Each line of the assembly language microprogram defines a symbolic microinstruction
- Each symbolic microinstruction is divided into five fields- Label, microoperation ,CD,BR and AD
- We will use also the pseudo-instruction ORG to define the origin, or first address, of a microprogram routine. Thus, the symbol ORG 64 informs the assembler to place the next microinstruction in control memory at decimal address 64, which is equivalent to the binary address 1000000.

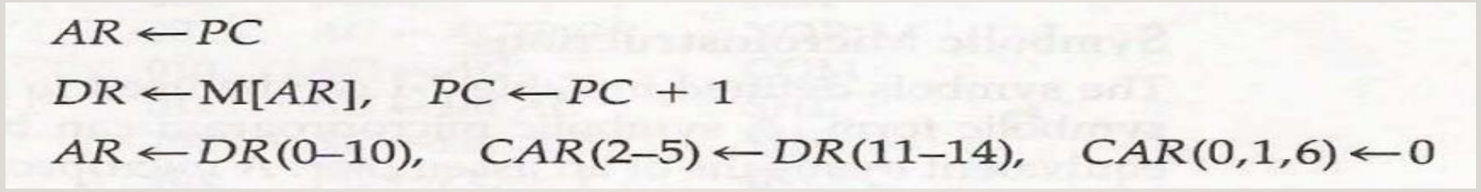
Symbolic Microinstruction



1.	Label	The label field may be empty or it may specify a symbolic address. A label is terminated with a colon (:).
2.	Microoperations	It consists of one, two, or three symbols, separated by commas, from those defined in Table 5.3. There may be no more than one symbol from each F field. The NOP symbol is used when the microinstruction has no microoperations. This will be translated by the assembler to nine zeros.
3.	CD	The CD field has one of the letters U, I, S, or Z.
4.	BR	The BR field contains one of the four symbols defined in Table 5.2.
5.	AD	The AD field specifies a value for the address field of the microinstruction in one of three possible ways: i. With a symbolic address, this must also appear as a label. ii. With the symbol NEXT to designate the next address in sequence. iii. When the BR field contains a RET or MAP symbol, the AD field is left empty and is converted to seven zeros by the assembler.

Fetch Routine

- The control memory has 128 words, and each word contains 20 bits. To microprogram the control memory, it is necessary to determine the bit values of each of the 128 words. The first 64 words (addresses 0 to 63) are to be occupied by the routines for the 16 instructions. The last 64 words may be used for any other purpose.
- The microinstructions needed for the fetch routine are

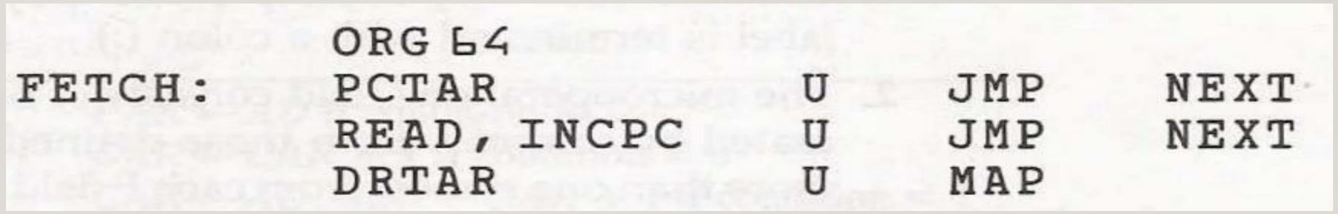


$AR \leftarrow PC$

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

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

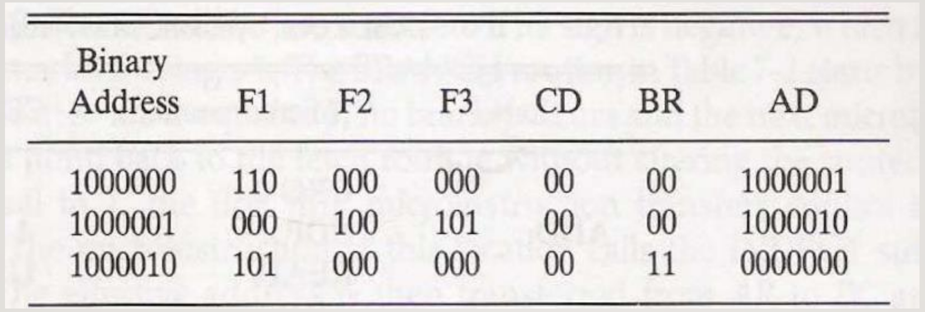
- The address part is transferred to AR and then control is transferred to one of 16 routines by mapping the operation code part of the instruction from DR into CAR.
- The fetch routine needs three microinstructions which are placed in control memory at addresses 64, 65, and 66.



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

Fetch Routine

- The translation of the symbolic microprogram to binary produces the following binary microprogram. The bit values are obtained from Table 3.3.

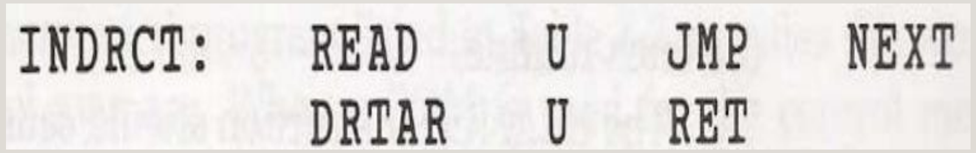


Binary Address	F1	F2	F3	CD	BR	AD
1000000	110	000	000	00	00	1000001
1000001	000	100	101	00	00	1000010
1000010	101	000	000	00	11	0000000

- The three microinstructions that constitute the fetch routine have been listed in three different representations.
- The symbolic representation is useful for writing microprograms in an assembly language format.
- The binary representation is the actual internal content that must be stored in control memory.

Symbolic Microprogram

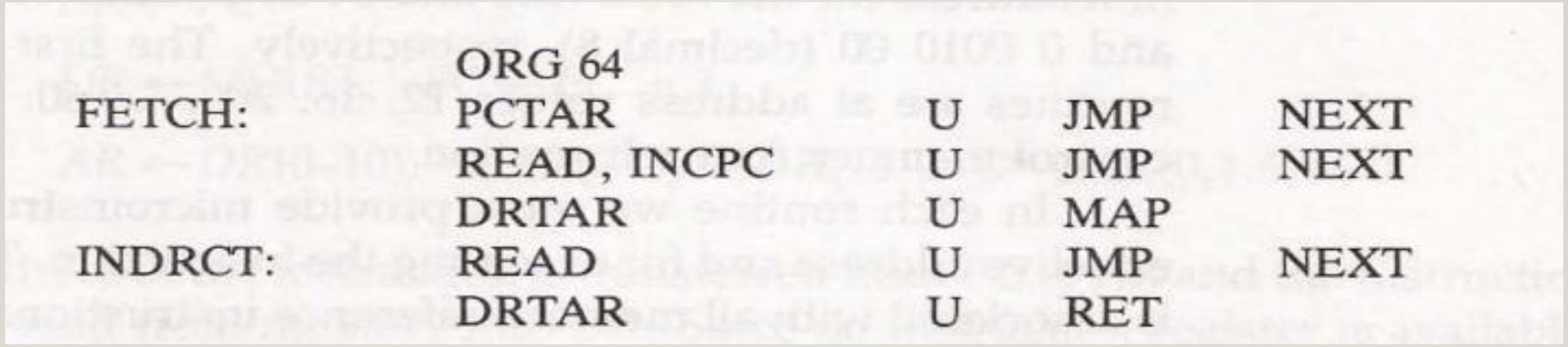
- The execution of the third (MAP) microinstruction in the fetch routine results in a branch to address 0xxxx00 where xxxx are four bits of the operation code
- The first address for the BRANCH and STORE routines are 0 0001 00 (decimal 4) and 0 0010 00 (decimal 8), respectively. The first adchess for the other 13 routines are at address values 12, 16, 20, ... , 60.
- The indirect address mode is associated with all memory-reference instructions.
- The subroutine, symbolized by INDRCT, is located right after the fetch routine, as shown below



```
INDRCT:   READ    U   JMP   NEXT
          DRTAR   U   RET
```

Symbolic Microprogram

- The transfer and return from the indirect subroutine occurs, assume that the MAP microinstruction at the end of the fetch routine caused a branch to address 0, where the ADD routine is stored.
- The first microinstruction in the ADD routine calls subroutine INDRCT, conditioned on status bit I.
- The INDRCT subroutine has two microinstructions.



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

Binary Microprogram

- The symbolic microprogram is useful format for expressing microprograms in a comprehensible manner. However, this is not how the microprogram is kept in memory. If the microprogram is basic enough, as it is in this case, the symbolic code must be translated to binary using either an assembler programme or by the user.
- In the symbolic microprogram address 3 has no equivalent in the symbolic microprogram since the ADD routine has only three microinstructions at addresses 0, 1, and 2.
- The next routine starts at address 4. Even though address 3 is not used, some binary value must be specified for each word in control memory.
- Since the position would never be utilized, we might have defined a word with only zeroes in it. However, it would be good to jump to address 64, which is the start of the fetch function, if an unforeseen problem occurs or if a noise signal sets CAR to the value of 3.