COMPUTER ORGANIZATION & ARCHITECTURE

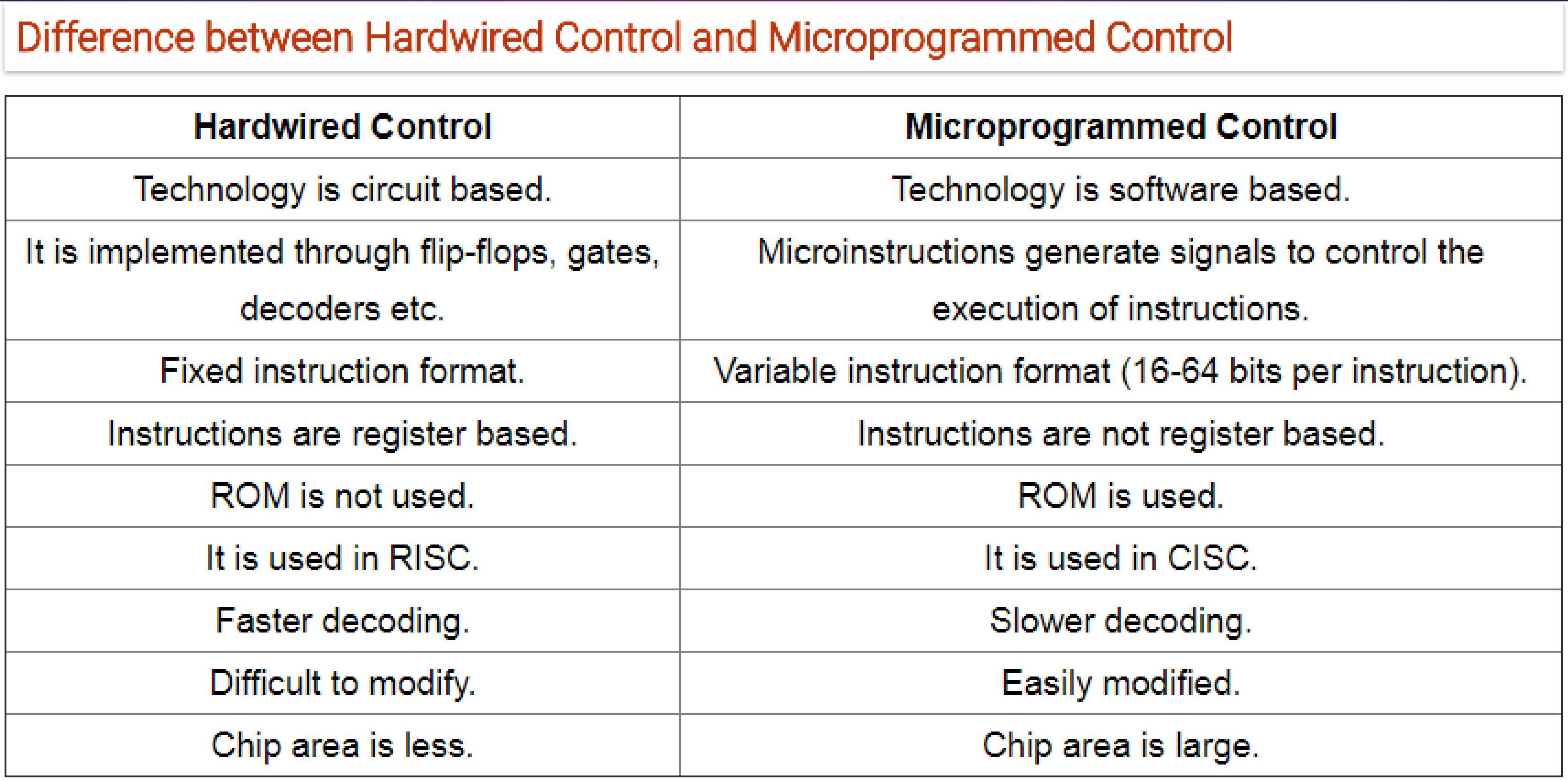
CONTROL UNIT

- Control unit generates timing and control signals for the operations of the computer.
- It’s the part of the CPU that initiates sequences of microoperations.
- It tells the computer’s memory, arithmetic & logic unit and input/output devices how to respond to a program’s instructions.

TYPES :

There are two methods to implement the control unit:

- **Hardwired Control Unit (**uses fixes instructions, combinational logic units of AND/OR (logic gates), encoders, decoders, etc.)
- **Microprogrammed Control Unit (t**he logic of the control unit is specified by microprograms (consists of a sequence of instructions that specify microoperations).



Difference between Hardwired Control and Microprogrammed Control

Hardwired Control	Microprogrammed Control
Technology is circuit based.	Technology is software based.
It is implemented through flip-flops, gates, decoders etc.	Microinstructions generate signals to control the execution of instructions.
Fixed instruction format.	Variable instruction format (16-64 bits per instruction).
Instructions are register based.	Instructions are not register based.
ROM is not used.	ROM is used.
It is used in RISC.	It is used in CISC.
Faster decoding.	Slower decoding.
Difficult to modify.	Easily modified.
Chip area is less.	Chip area is large.

IMPORTANT TERMS

Control unit initiates a series of sequential steps of microoperations.

- **CONTROL WORD :**

- The control variables at any time are represented by 1’s and 0’s, known as Control Word.

- Control words can be programmed to perform various operations.

- **MICROPROGRAMMED CONTROL UNIT** :

- A control unit whose binary control variables are stored in the memory is called MICROPROGRAMMED CONTROL UNIT.

- **MICROINSTRUCTIONS** :

- Each word in the control memory contains microinstructions .

- It specifies one or more microoperations for the system.

MICROPROGRAM

- **MICROPROGRAM** :
  - Sequence of microinstructions constitutes a microprogram.
  - The microprogram controls the function of the CPU.
  - Alterations in microprograms are not needed once the control unit is in operation. The control memory can be read-only memory (ROM).
  - The Contents of words in **ROM** are fixed and cannot be altered, since there is no writing capacity in ROM.
- **DYNAMIC MICROPROGRAMMING :**
  - microprogram is loaded initially from an auxiliary memory (secondary memory), such as magnetic disk.
  - employs a writable control memory which allows the user to change the microprogram, though it is mostly used for reading.

TYPES OF MEMORY IN MPC

1. MAIN MEMORY

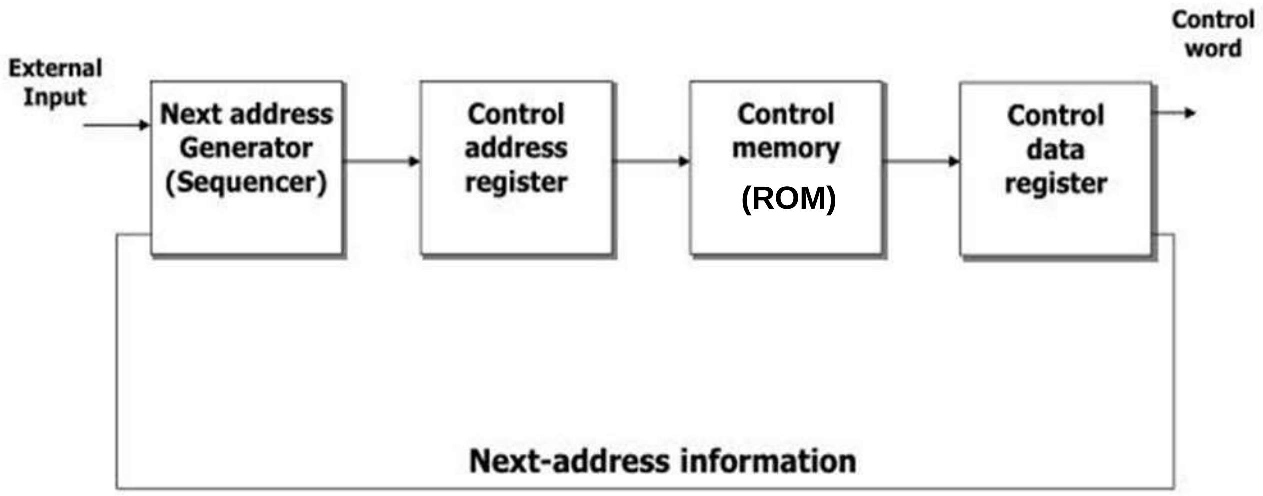
- The main memory is used for storing programs. The content of the main memory can be altered when data is manipulated and each time the program is changed.
- Program in main memory contains machine instructions and data.

2. CONTROL MEMORY

- Memory that is a part of the control unit is called Control memory.
- Control memory holds microprograms that cannot be altered by the user. The microprogram consists of microinstructions to execute register microoperations.
- Machine instruction initiates a series of microinstructions in the control memory.
- The microinstruction generates microoperations to fetch instructions from main memory, to evaluate effective address, to execute the operations specified by instructions or to repeat the cycle for the next instruction.

CONFIGURATION

The general configuration of microprogrammed control unit is demonstrated as follows:



SEQUENCER AND PIPELINE REGISTER

- **SEQUENCER**:

  The next address generator is called a sequencer, as it determines the address sequence that is read from the control memory.

- FUNCTIONS:

1. Incrementing the control register
2. Loading an address from control memory to CAR.
3. Loading an initial address

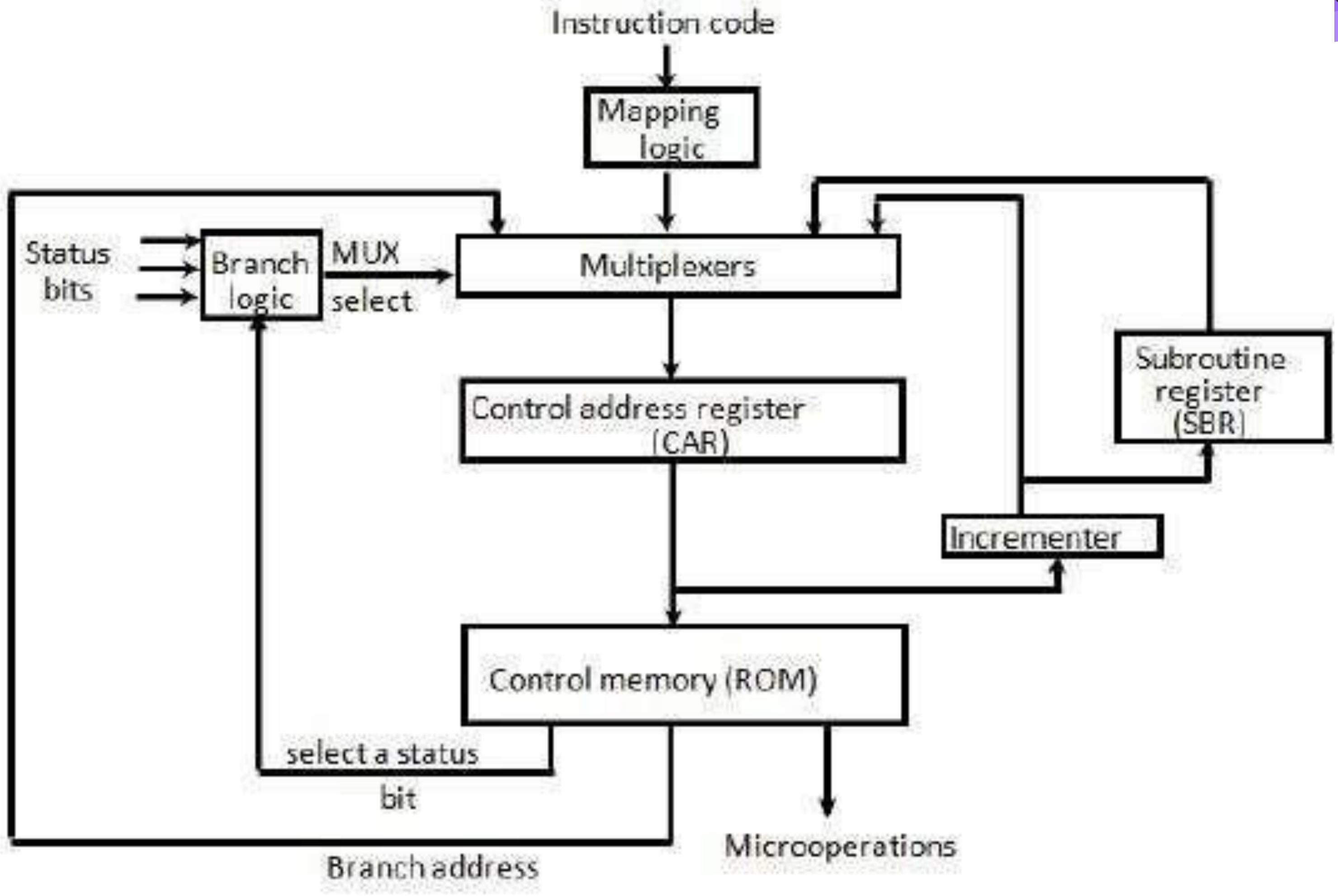
- **PIPELINE REGISTER** :

  The control data register holds the present microinstruction while the next address is computed and read from the memory. This data register is called pipeline register.

- IMPORTANT ADVANTAGE OF MICROPROGRAMMED CONTROL :

  No hardware or wiring changes is required.

SELECTION OF ADDRESS FOR CONTROL MEMORY



ADDRESS SEQUENCING

- Micro instructions are stored in control memory in group with each group specifying a routine.
- An initial address is loaded into the control address register when power is turned on. The fetch routine may be sqenced by incrementing CAR. At end of fetch routine instructions is in IR .
- The effective address computation routine in control memory can be reached through branch micro instructions which is conditioned on status mode bits of instructions.
- The micro operations steps to be generated in processor register depend on opcode of instruction.
- Subroutine will require an external register for storing return address. Return address cannot be stored in ROM because it has no writing capability.

ADDRESS SEQUENCING

- The micro instructions in control memory contains a set of bits to initiate micro operations in registers and other bits to specify method by which next address is obtained.
- The diagram shows four different paths from which the CAR receives the address.
- The incrementer increments the CAR content by one.
- Branching is achieved by specifying branch address in one of the fields of micro instructions.
- Conditional branching is obtained by status bit in order to determine it's condition.
- The return address for a sub routine is stored in special register (SBR) whose value is then used when microprogram wishes to return from the subroutine.

STEPS IN SELECTION OF ADDRESS

- **CONDITIONAL BRANCHING**

 - branching from one routine to another depending on status bit conditions

 - status bits provide parameter information.

 - information in status bits are tested and actions are initiated based on their conditions:1or0

- **UNCONDITIONAL BRANCH**

 - fix value of status bit to 1

- **SUBROUTINES**

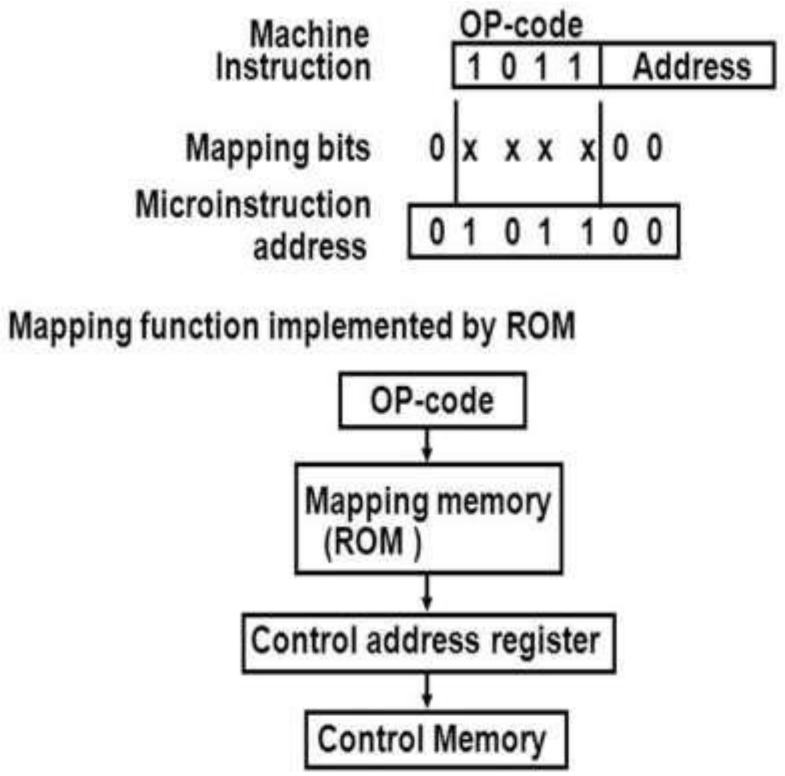
- A set of common instructions that can be used in a program many times

- Each time a subroutine is used in main program, a branch is made to the beginning of subroutine

MAPPING OF INSTRUCTION

- Each computer instruction has its own microprogram
- Routine stored in a given location of the control memory
- **MAPPING :**

- transformation from instruction code bits to microinstruction address in control memory where routine is located.



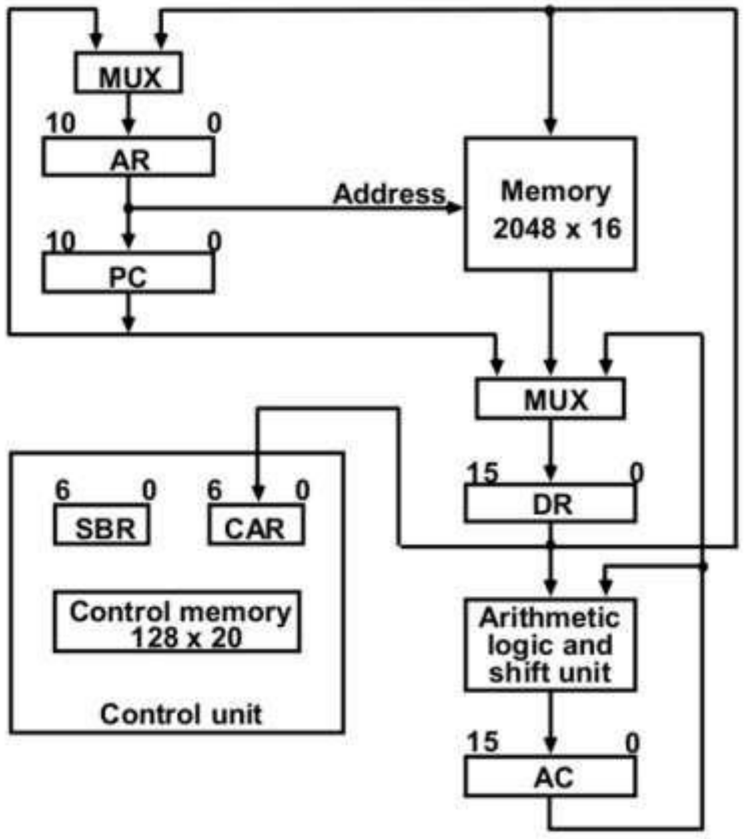
COMPUTER HARDWARE CONFIGURATION

MICROPROGRAM EXAMPLE:

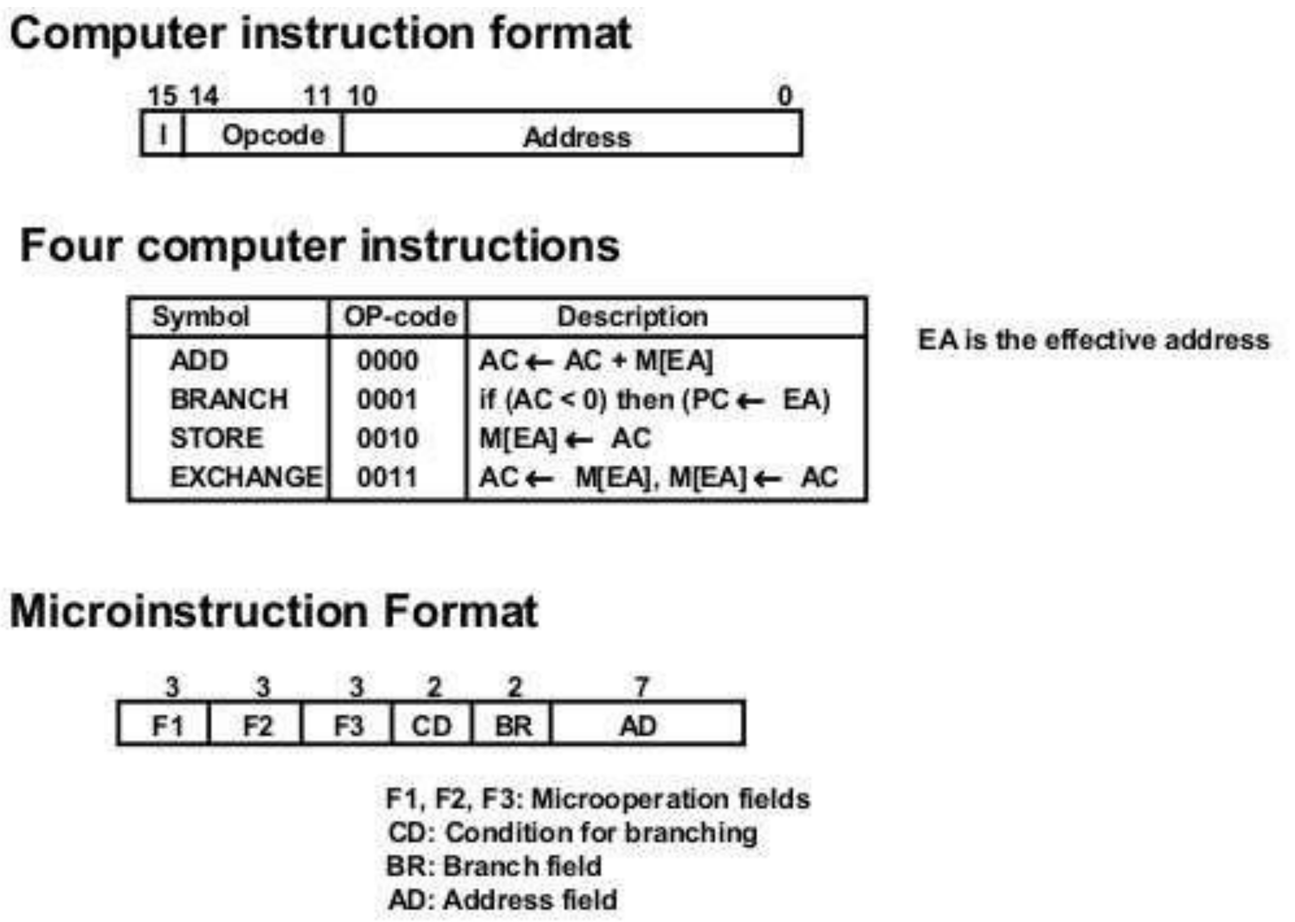
- Two memory units: Main memory, control memory.

- CONTROL MEMORY :

- 4 registers are associated with processor unit (PC,AR,DR,AC)
- 2 registers are associated with the control unit (CAR,SBR)



MICROINSTRUCTION CODE FORMAT – 20 BITS



Computer instruction format

15	14 – 11	10 – 0
I	Opcode	Address

Four computer instructions

Symbol	OP-code	Description
ADD	0000	AC ← AC + M[EA]
BRANCH	0001	if (AC < 0) then (PC ← EA)
STORE	0010	M[EA] ← AC
EXCHANGE	0011	AC ← M[EA], M[EA] ← AC

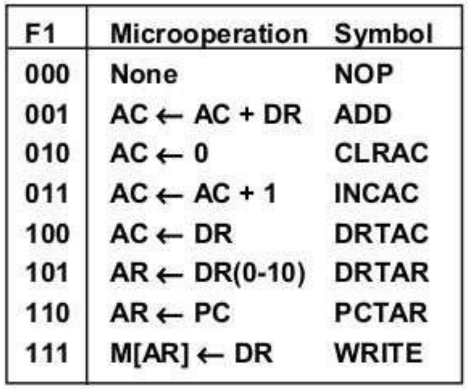
EA is the effective address

Microinstruction Format

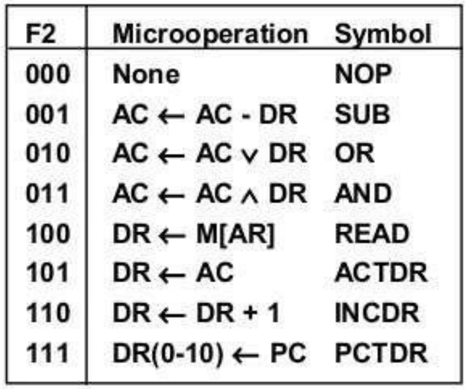
3	3	3	2	2	7
F1	F2	F3	CD	BR	AD

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

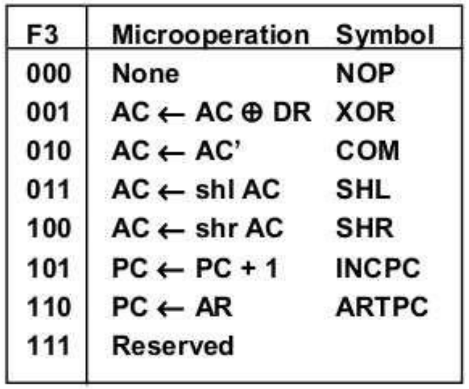
SYMBOLS & BINARY CODE – I F1,F2,F3



F1	Microoperation	Symbol
000	None	NOP
001	AC ← AC + DR	ADD
010	AC ← 0	CLRAC
011	AC ← AC + 1	INCAC
100	AC ← DR	DRTAC
101	AR ← DR(0-10)	DRTAR
110	AR ← PC	PCTAR
111	M[AR] ← DR	WRITE

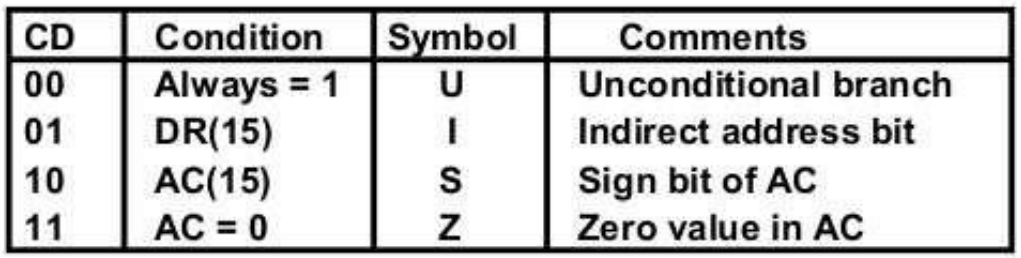


F2	Microoperation	Symbol
000	None	NOP
001	AC ← AC - DR	SUB
010	AC ← AC ∨ DR	OR
011	AC ← AC ∧ DR	AND
100	DR ← M[AR]	READ
101	DR ← AC	ACTDR
110	DR ← DR + 1	INCDR
111	DR(0-10) ← PC	PCTDR

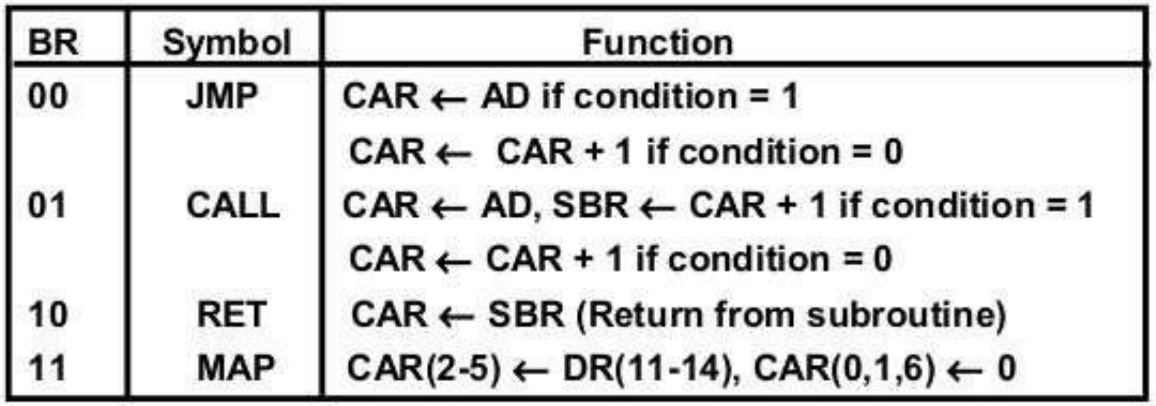


F3	Microoperation	Symbol
000	None	NOP
001	AC ← AC ⊕ DR	XOR
010	AC ← AC'	COM
011	AC ← shl AC	SHL
100	AC ← shr AC	SHR
101	PC ← PC + 1	INCPC
110	PC ← AR	ARTPC
111	Reserved	

SYMBOLS & BINARY CODE – II CD,BR

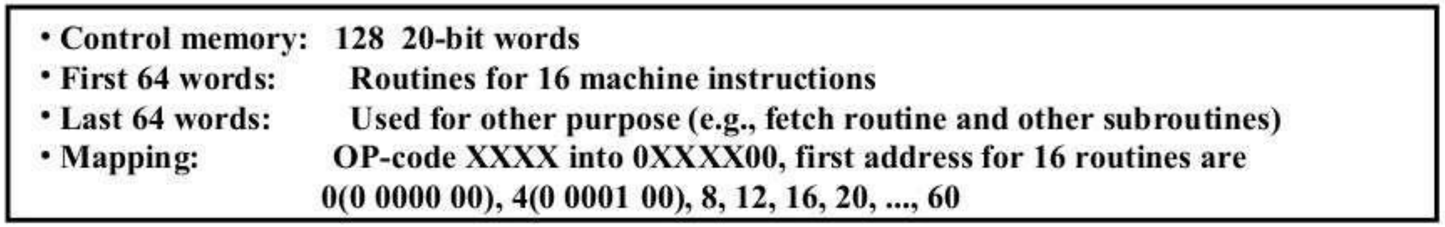


CD	Condition	Symbol	Comments
00	Always = 1	U	Unconditional branch
01	DR(15)	I	Indirect address bit
10	AC(15)	S	Sign bit of AC
11	AC = 0	Z	Zero value in AC

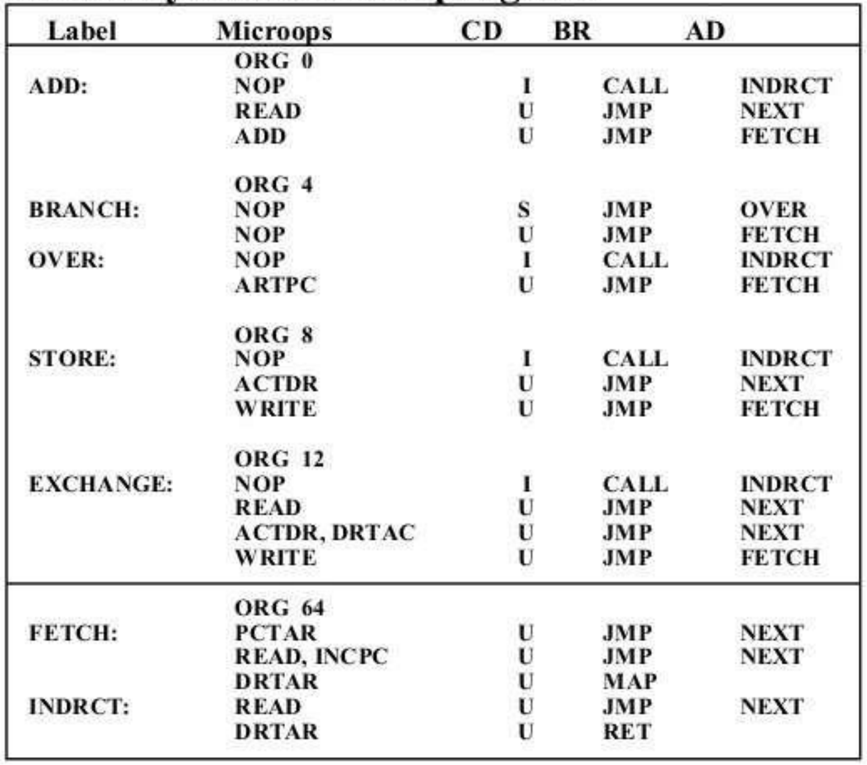


BR	Symbol	Function
00	JMP	CAR ← AD if condition = 1
		CAR ← CAR + 1 if condition = 0
01	CALL	CAR ← AD, SBR ← CAR + 1 if condition = 1
		CAR ← CAR + 1 if condition = 0
10	RET	CAR ← SBR (Return from subroutine)
11	MAP	CAR(2-5) ← DR(11-14), CAR(0,1,6) ← 0

SYMBOLIC MICROPROGRAM



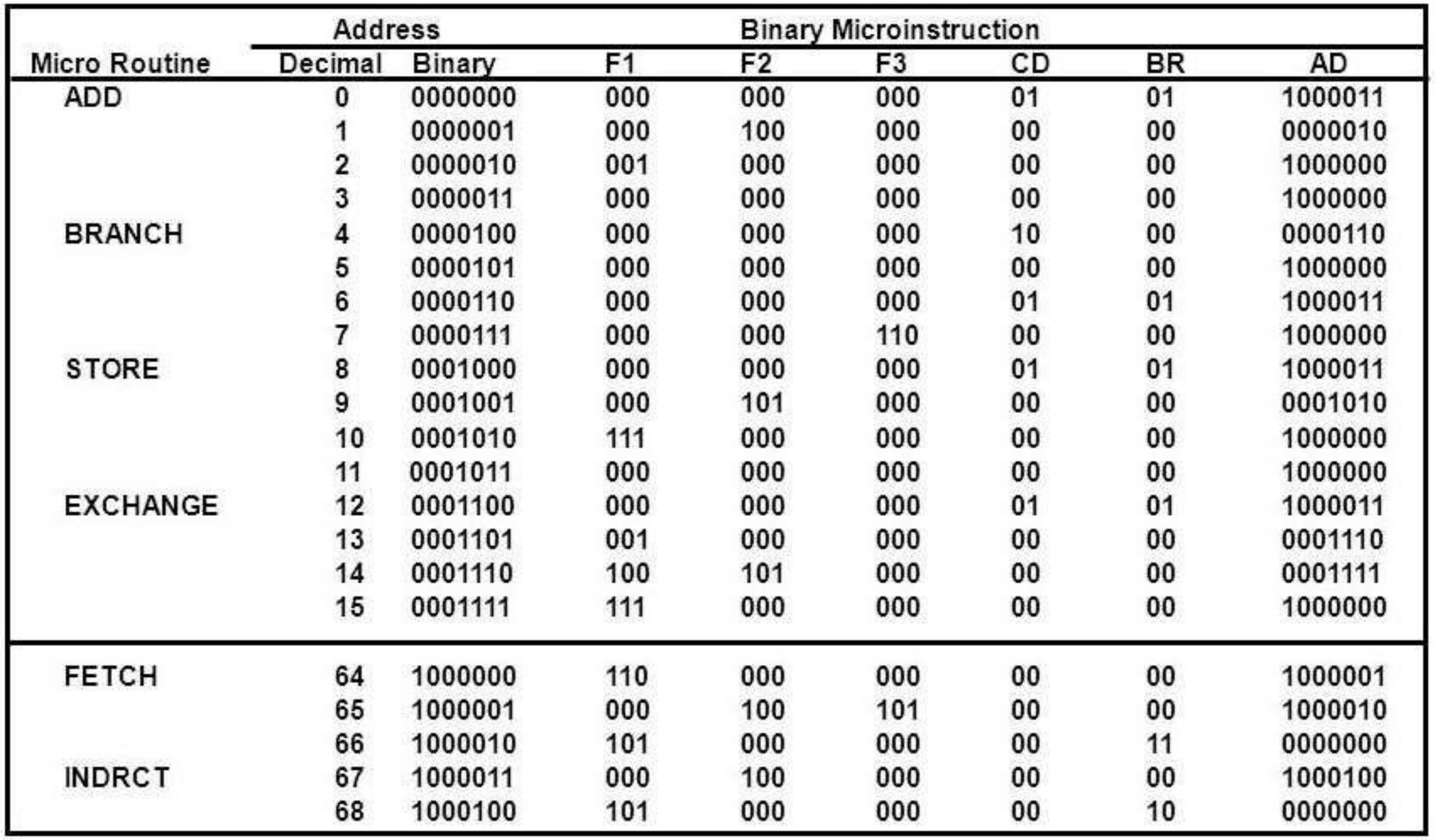
- Control memory: 128 20-bit words
- First 64 words: Routines for 16 machine instructions
- Last 64 words: Used for other purpose (e.g., fetch routine and other subroutines)
- Mapping: OP-code XXXX into 0XXXX00, first address for 16 routines are 0(0 0000 00), 4(0 0001 00), 8, 12, 16, 20, ..., 60



Partial Symbolic Microprogram

Label	Microops	CD	BR	AD
	ORG 0			
ADD:	NOP	I	CALL	INDRCT
	READ	U	JMP	NEXT
	ADD	U	JMP	FETCH
	ORG 4			
BRANCH:	NOP	S	JMP	OVER
	NOP	U	JMP	FETCH
OVER:	NOP	I	CALL	INDRCT
	ARTPC	U	JMP	FETCH
	ORG 8			
STORE:	NOP	I	CALL	INDRCT
	ACTDR	U	JMP	NEXT
	WRITE	U	JMP	FETCH
	ORG 12			
EXCHANGE:	NOP	I	CALL	INDRCT
	READ	U	JMP	NEXT
	ACTDR, DRTAC	U	JMP	NEXT
	WRITE	U	JMP	FETCH
	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



	Address		Binary Microinstruction					
Micro Routine	Decimal	Binary	F1	F2	F3	CD	BR	AD
ADD	0	0000000	000	000	000	01	01	1000011
	1	0000001	000	100	000	00	00	0000010
	2	0000010	001	000	000	00	00	1000000
	3	0000011	000	000	000	00	00	1000000
BRANCH	4	0000100	000	000	000	10	00	0000110
	5	0000101	000	000	000	00	00	1000000
	6	0000110	000	000	000	01	01	1000011
	7	0000111	000	000	110	00	00	1000000
STORE	8	0001000	000	000	000	01	01	1000011
	9	0001001	000	101	000	00	00	0001010
	10	0001010	111	000	000	00	00	1000000
	11	0001011	000	000	000	00	00	1000000
EXCHANGE	12	0001100	000	000	000	01	01	1000011
	13	0001101	001	000	000	00	00	0001110
	14	0001110	100	101	000	00	00	0001111
	15	0001111	111	000	000	00	00	1000000
FETCH	64	1000000	110	000	000	00	00	1000001
	65	1000001	000	100	101	00	00	1000010
	66	1000010	101	000	000	00	11	0000000
INDRCT	67	1000011	000	100	000	00	00	1000100
	68	1000100	101	000	000	00	10	0000000

THANK YOU!