Control Unit

Unit: 3

Computer Organization & Architecture(ACSE0305)

B Tech 3rd Sem

Pradeep Kumar

Assistant Professor CSE

Department

CSE 3rd Semester Evaluation Scheme

EVALUATION SCHEME
SEMESTER-III

Sl. No.	Subject Codes	Subject Name	Periods			Evaluation Schemes				End Semester		Total	Credit
			L	T	P	CT	TA	TOTAL	PS	TE	PE		
WEEKS COMPULSORY INDUCTION PROGRAM													
1	AAS0301A	Engineering Mathematics III	3	1	0	30	20	50		100		150	4
2	ACSE0306	Discrete Structures	3	0	0	30	20	50		100		150	3
3	ACSE0304	Digital Logic & Circuit Design	3	0	0	30	20	50		100		150	3
4	ACSE0301	Data Structures	3	1	0	30	20	50		100		150	4
5	ACSE0302	Object Oriented Techniques using Java	3	0	0	30	20	50		100		150	3
6	ACSE0305	Computer Organization & Architecture	3	0	0	30	20	50		100		150	3
7	ACSE0354	Digital Logic & Circuit Design Lab	0	0	2				25		25	50	1
8	ACSE0351	Data Structures Lab	0	0	2				25		25	50	1
9	ACSE0352	Object Oriented Techniques using Java Lab	0	0	2				25		25	50	1
10	ACSE0359	Internship Assessment-I	0	0	2				50			50	1
11	ANC0301/ ANC0302	Cyber Security*/ Environmental Science*(Non	2	0	0	30	20	50		50		100	0

Syllabus

UNIT-I	Introduction	8 Hours
Computer Organization and Architecture, Functional units of digital system and their interconnections, buses, bus architecture, types of buses and bus arbitration and it's types. Register, bus and memory transfer. Process or organization, general registers organization, stack organization and address in g modes.		
UNIT-II	ALU Unit	8 Hours
Arithmetic and logic unit: Lookahead carries adders. Multiplication: Signed operand multiplication, Booth's algorithm and array multiplier. Division and logic operations. Floating point arithmetic operation, Arithmetic & logic unit design. IEEE Standard for Floating Point Numbers.		
UNIT-III	Control Unit	8 Hours
Control Unit: Instruction types, formats, instruction cycles and sub cycles (fetch and execute etc.), micro-operations, execution of a complete instruction. Program Control, Reduced Instruction Set Computer, Complex Instruction Set Computer, Pipelining. Hardwire and microprogrammed control, Concept of horizontal and vertical microprogramming, Flynn's classification.		
UNIT-IV	Memory Unit	8 Hours
Memory: Basic concept and hierarchy, semiconductor RAM memories, 2D & 2 1/2D memory organization. ROM memories. Cache memories: concept and design issues & performance, address mapping and replacement Auxiliary memories: magnetic disk, magnetic tape and optical disks Virtual memory: concept implementation, Memory Latency, Memory Bandwidth, Memory Seek Time.		
UNIT-V	Input/Output	8 Hours
Peripheral devices, I/O interface, I/O ports, Interrupts: interrupt hardware, types of interrupts and exceptions. Modes of Data Transfer: Programmed I/O, interrupt initiated I/O and Direct Memory Access. ,I/O channels and processors. Serial Communication: Synchronous & asynchronous communication.		

Course Objective

- The objective of this course is to understand the types of organizations, structures and functions of computer, design of arithmetic and logic unit and float point arithmetic as well as to understand the concepts of memory system, communication with I/O devices and interfaces

Course Outcome

- Understand the basic structure and operation of a digital computer system.

Analyze the design of arithmetic & logic unit and understand the fixed point and floating-point arithmetic operations.

Implement control unit techniques and the concept of Pipelining

Understand the hierarchical memory system, cache memories and virtual memory.

Understand different ways of communicating with I/O devices and standard I/O interfaces.

Program Outcome

1.Engineering knowledge

2. Problem analysis

3. Design/development of solutions

4. Conduct investigations of complex problems

5. Modern tool usage

6. The engineer and society

7. Environment and sustainability

8. Ethics

9. Individual and team work

10. Communication

11. Project management and finance

12. Life-long learning

CO-PO Mapping

COMPUTER ORGANIZATION AND ARCHITECTURE (ACSE0305)												
CO.K	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
ACSE0305.3	3	2	2	1	2	2	1	1	2	2	1	2

Program Specific Outcome

On successful completion of graduation degree, The computer Science & Engineering graduates will be able to:

PSO1: identify, analyze real world problems and design their ethical solutions using artificial intelligence, robotics, virtual/augmented reality, data analytics, block chain technology, and cloud computing.

PSO2: design and develop the hardware sensor devices and related interfacing software systems for solving complex engineering problems.

PSO 3: understand inter-disciplinary computing techniques and to apply them in the design of advanced computing.

PSO 4: conduct investigation of complex problem with the help of technical, managerial, leadership qualities, and modern engineering tools provided by industry sponsored laboratories.

CO- PSO Mapping

COMPUTER ORGANIZATION AND ARCHITECTURE (ACSE0305)				
CO.K	**PSO1**	**PSO2**	**PSO3**	**PSO4**
ACSE0305.3	3	3	3	2

Program Educational Objectives

PEO 1: To have an excellent scientific and engineering breadth so as to comprehend, analyze, design and provide sustainable solutions for real-life problems using state-of-the-art technologies.

PEO 2: To have a successful career in industries, to pursue higher studies or to support entrepreneurial endeavors and to face the global challenges.

PEO 3: To have an effective communication skills, professional attitude, ethical values and a desire to learn specific knowledge in emerging trends, technologies for research, innovation and product development and contribution to society.

PEO 4: To have life-long learning for up-skilling and re-skilling for successful professional career as engineer, scientist, entrepreneur and bureaucrat for betterment of society.

Prerequisite and Recap

- **Arithmetic Logic Unit**

Content

Control Unit:

- Instruction types, formats
- Instruction cycles and sub cycles (fetch and execute etc)
- Micro operations
- Execution of a complete instruction.
- Program Control
- Reduced Instruction Set Computer
- Pipelining
- Hardwire and micro programmed control
- Micro programmed sequencing
- Concept of horizontal and vertical microprogramming.

Unit Objective

- Implementation of control unit techniques and the concept of Pipelining.
- Study of Instruction, types of instruction, format of instuction,Cycle and sub cycle, micro operation.

Control Unit (CU)

Control unit (CU) of a processor translates from machine instructions to the control signals for the microoperations that implement them

Control units are implemented in one of two ways

Hardwired Control

CU is made up of sequential and combinational circuits to generate the control signals

Microprogrammed Control

A control memory on the processor contains microprograms that activate the necessary control signals

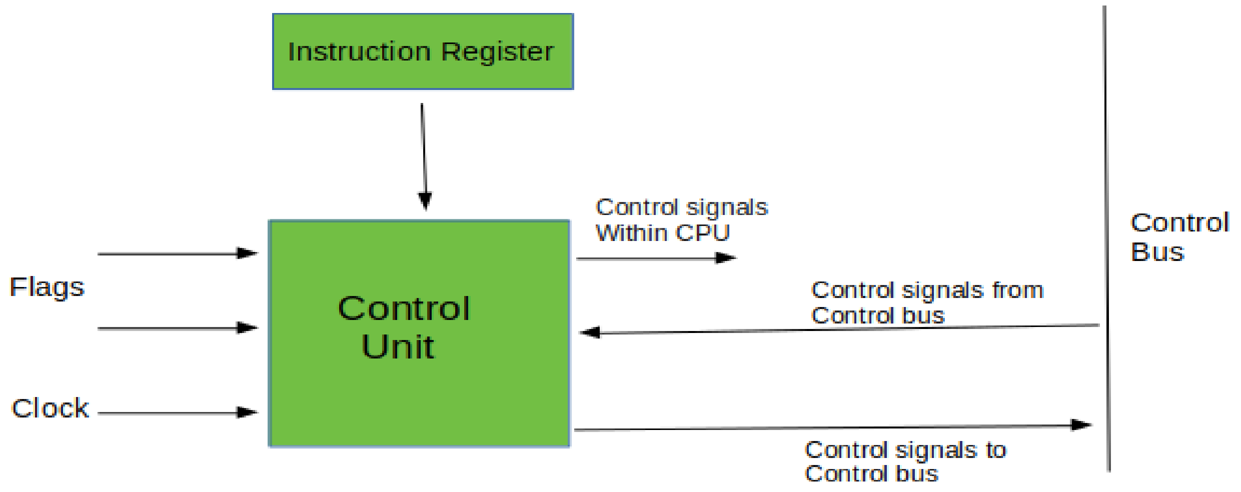
We will consider a hardwired implementation of the control unit for the Basic Computer

Control Unit (CU)

•**Control Unit** is the part of the computer's central processing unit (CPU), which directs the operation of the processor.

•It is the responsibility of the Control Unit to tell the computer's memory, arithmetic/logic unit and input and output devices how to respond to the instructions that have been sent to the processor.

•A control unit works by receiving input information to which it converts into control signals, which are then sent to the central processor.


Instruction Register
Control signals Within CPU
Control Bus
Flags
Control Unit
Control signals from Control bus
Clock
Control signals to Control bus

Block Diagram of the Control Unit

Functions of the Control Unit

- ➢It coordinates the sequence of data movements into, out of, and between a processor's many sub-units.
- ➢It interprets instructions.
- ➢It controls data flow inside the processor.
- ➢It receives external instructions or commands to which it converts to sequence of control signals.
- ➢It controls many execution units(i.e. ALU, data buffers and registers) contained within a CPU.
- ➢It also handles multiple tasks, such as fetching, decoding, execution handling and storing results.

Types of the Control Unit

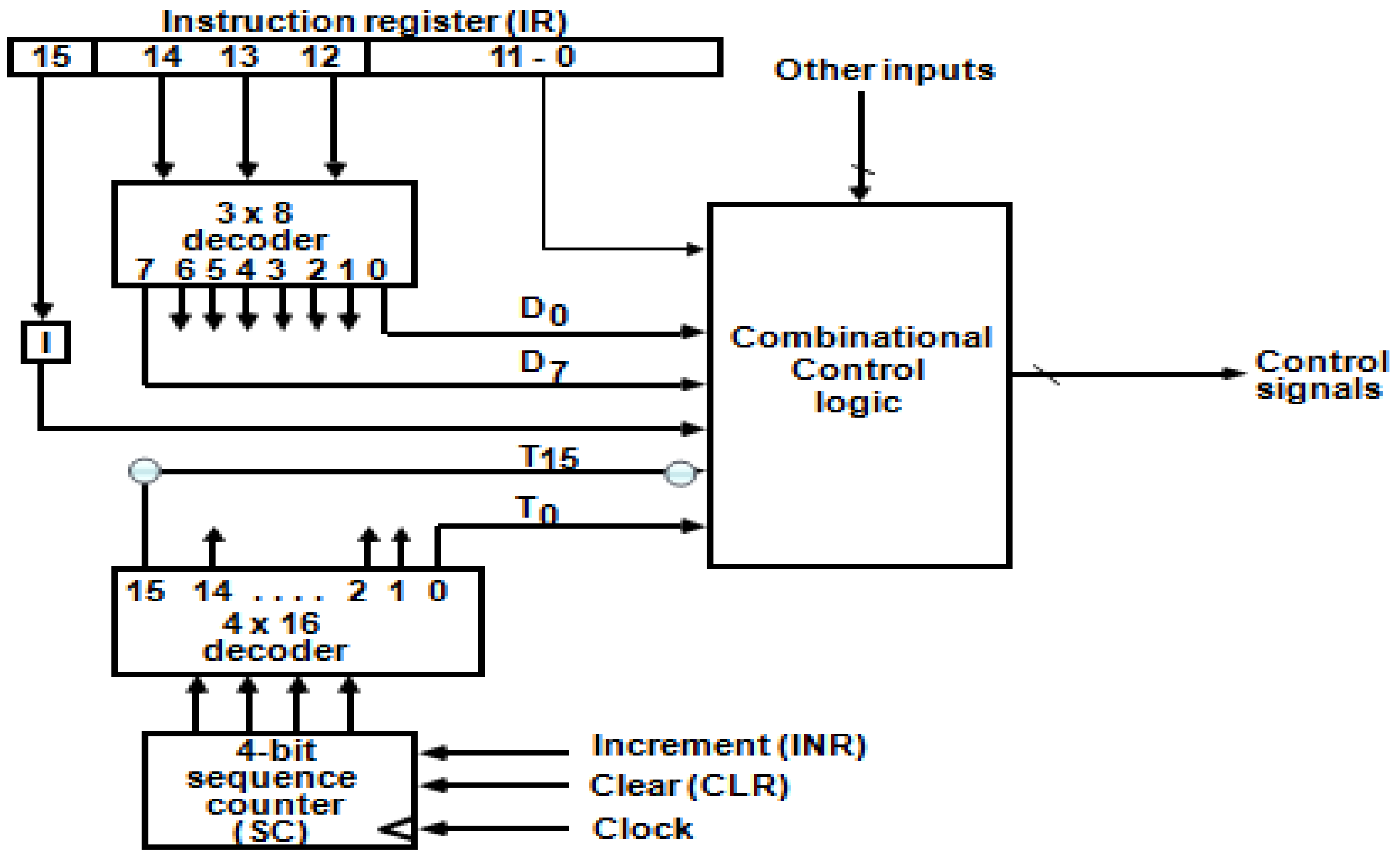
There are two types of control units:

Hardwired control unit and

Micro programmable control unit.

Control Unit (CU)

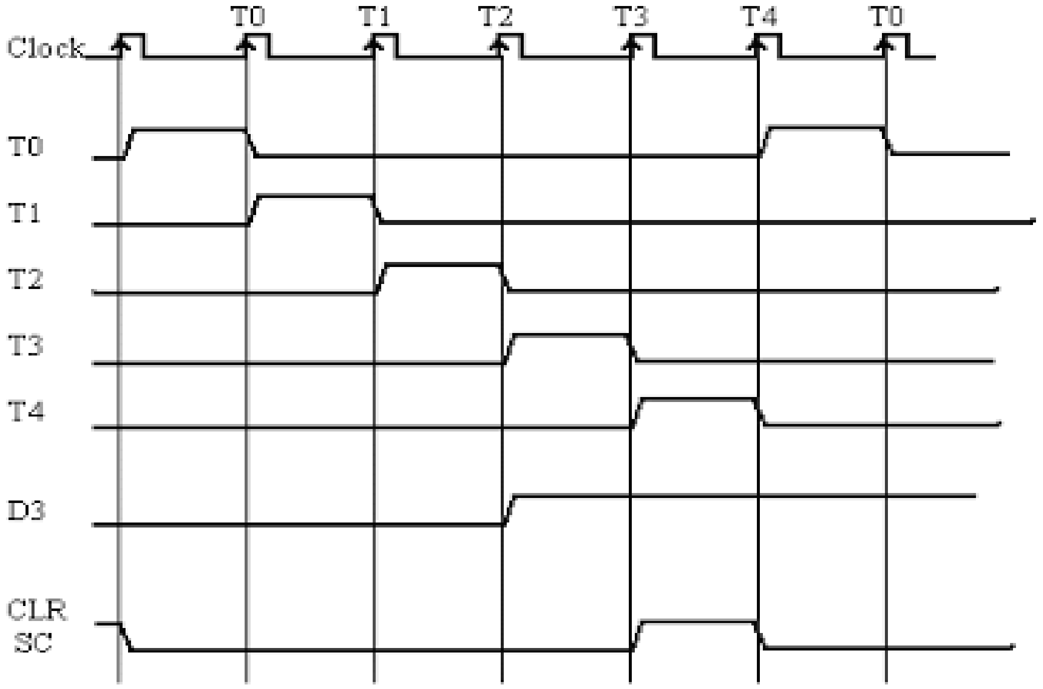
•Control unit of Basic Computer



Control Unit (CU)

Example:

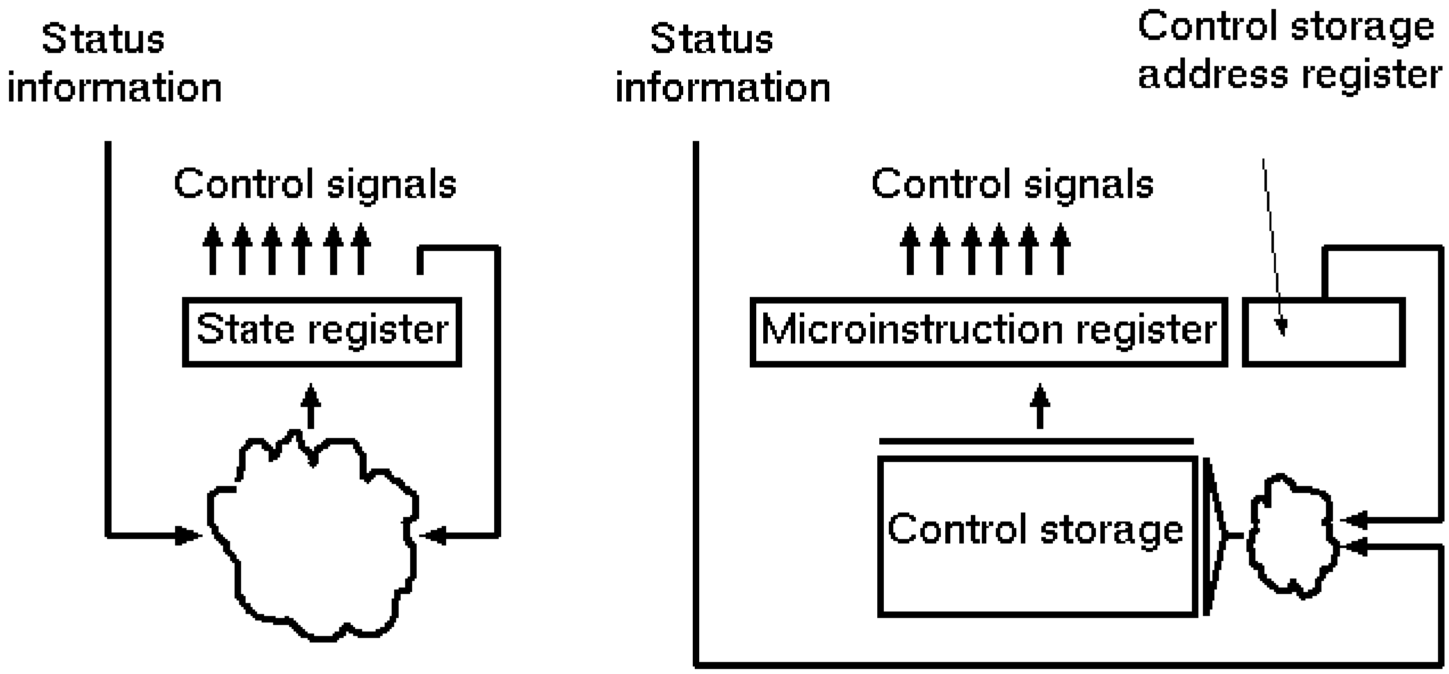
$$D_3T_4: SC \leftarrow 0$$



Example of control timing signals

Control Unit (CU)

- **HARDWIRED/MICROPROGRAMMED**

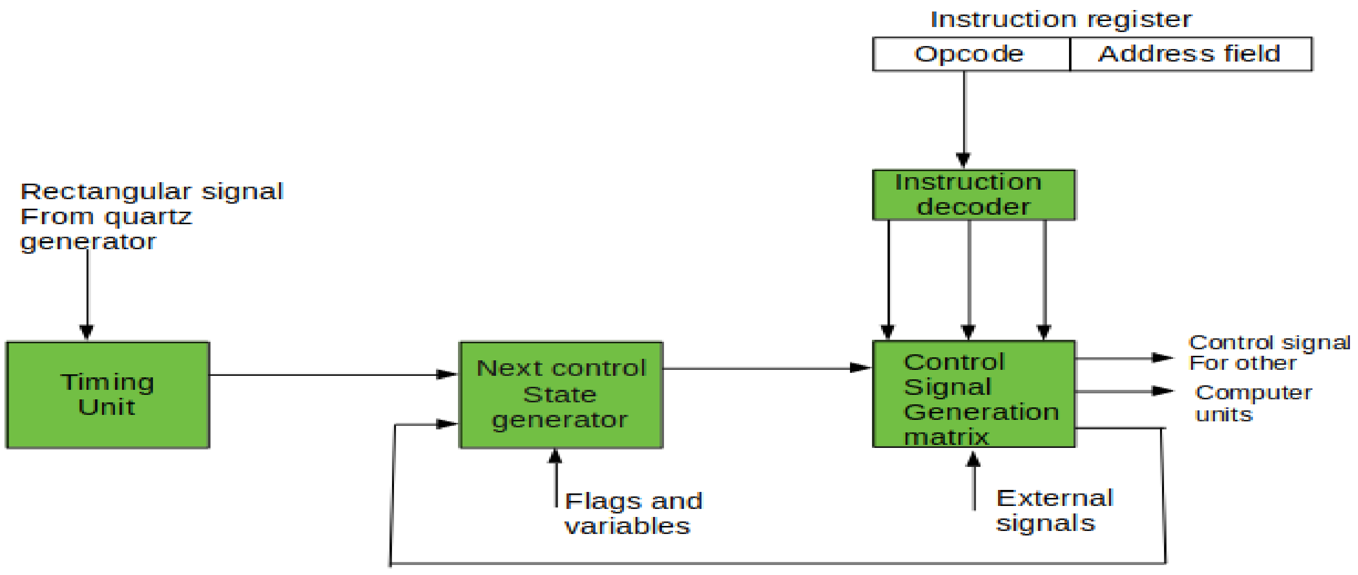


(a) Hardwired control (b) Microprogrammed control

Hardwired control unit

➢It In the hardwired organization, the control logic is implemented with gates, flip-flops, decoders, and other digital circuits. It has the advantage that it can be optimized to produce a fast mode of operation.

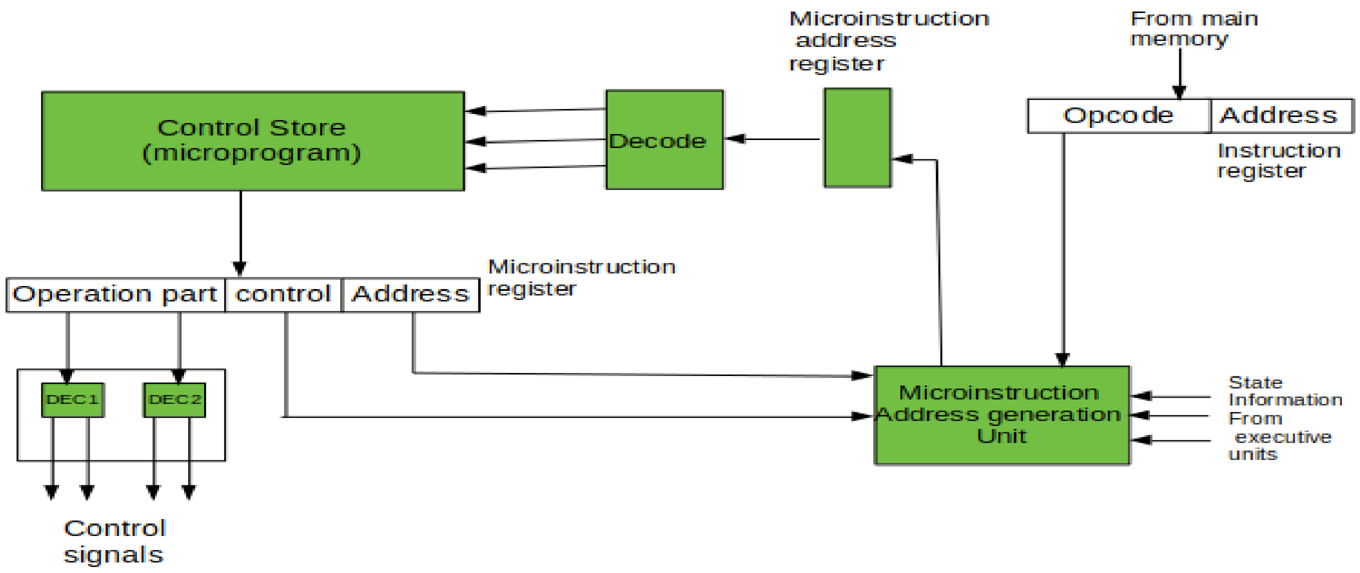
➢The hardwired control, as the name implies, requires changes in the wiring among the various components if the design has to be modified or changed.


Instruction register
Opcode
Address field
Instruction decoder
Rectangular signal From quartz generator
Timing Unit
Next control State generator
Control Signal Generation matrix
Control signal For other Computer units
Flags and variables
External signals

Block diagram of a hardwired control unit of a computer

Microprogrammed control unit

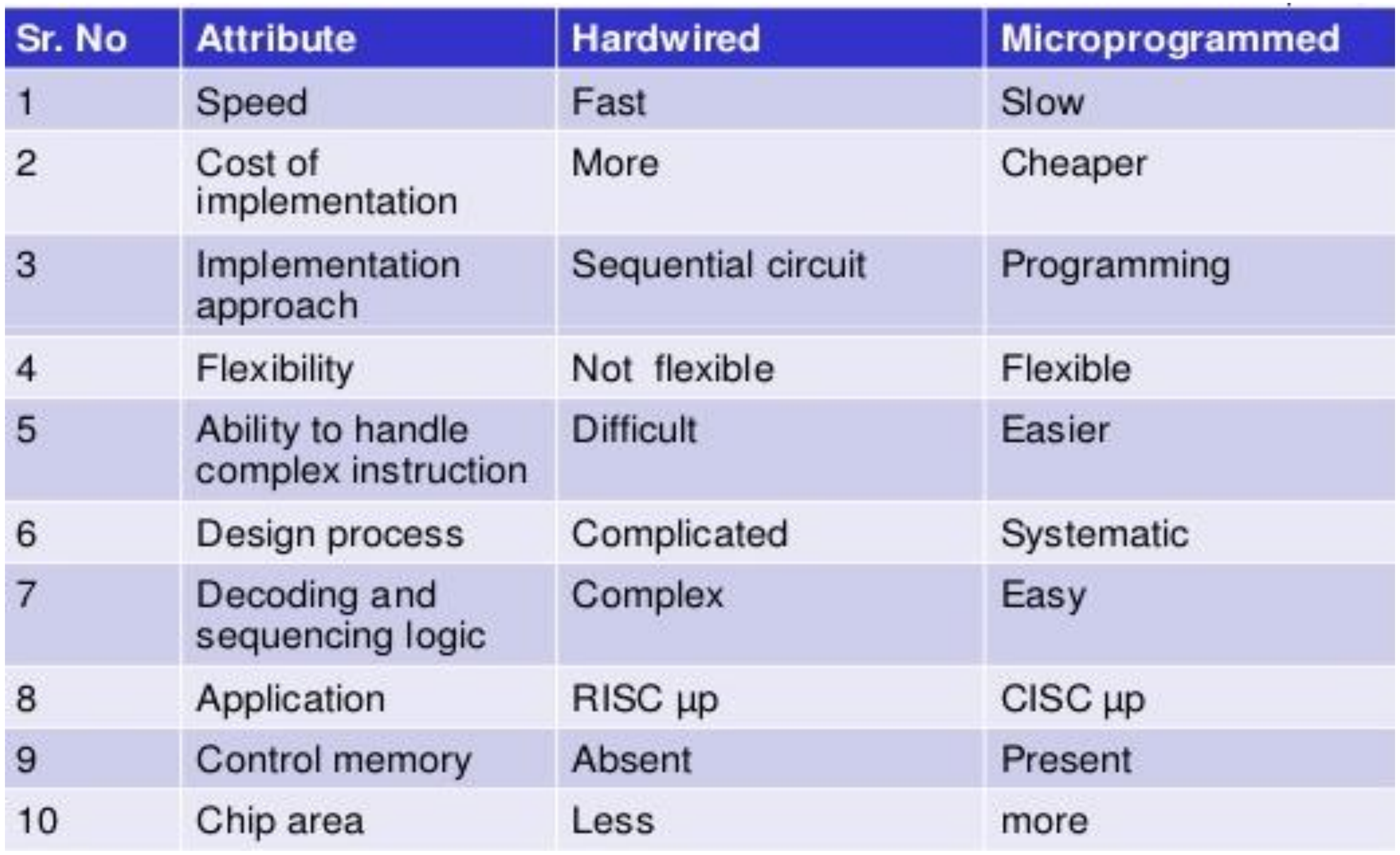
- It In the microprogrammed organization, the control information is stored in a control memory.
- The control memory is programmed to initiate the required sequence of microoperations.
- In the microprogrammed control, any required changes or modifications can be done by updating the microprogram in control memory.



Microinstruction address register
From main memory
Control Store (microprogram)
Decode
Opcode
Address
Instruction register
Operation part
control
Address
Microinstruction register
DEC1
DEC2
Microinstruction Address generation Unit
State Information From executive units
Control signals

Microprogrammed control unit with a single level control store

Hardwired/Microprogrammed



Sr. No	Attribute	Hardwired	Microprogrammed
1	Speed	Fast	Slow
2	Cost of implementation	More	Cheaper
3	Implementation approach	Sequential circuit	Programming
4	Flexibility	Not flexible	Flexible
5	Ability to handle complex instruction	Difficult	Easier
6	Design process	Complicated	Systematic
7	Decoding and sequencing logic	Complex	Easy
8	Application	RISC µp	CISC µp
9	Control memory	Absent	Present
10	Chip area	Less	more

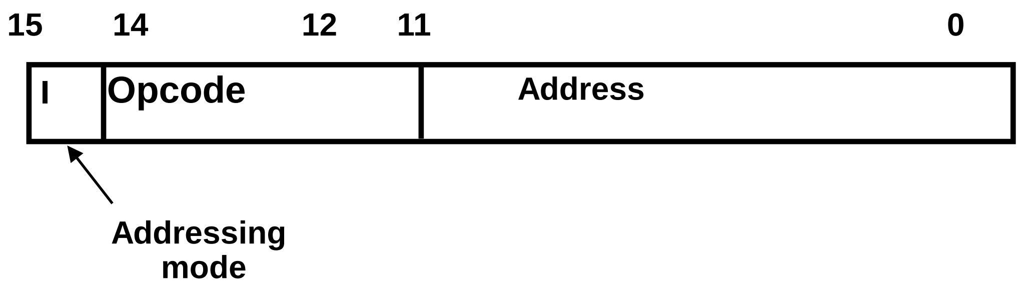
Instruction

- **Program**
  - A sequence of (machine) instructions
- **(Machine) Instruction**
  - A group of bits that tell the computer to *perform a specific operation* (a sequence of micro-operation)
- The instructions of a program, along with any needed data are stored in memory
- The CPU reads the next instruction from memory
- It is placed in an *Instruction Register* (IR)
- Control circuitry in control unit then translates the instruction into the sequence of micro operations necessary to implement it

Instruction Format

- In the Basic Computer, bit 15 of the instruction specifies the *addressing mode* (0: direct addressing, 1: indirect addressing)
- Since the memory words, and hence the instructions, are 16 bits long, that leaves 3 bits for the instruction's opcode

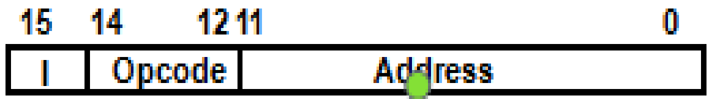
Instruction Format



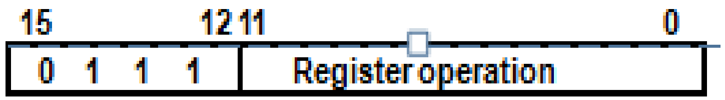
Instruction Format

- Basic Computer Instruction Format

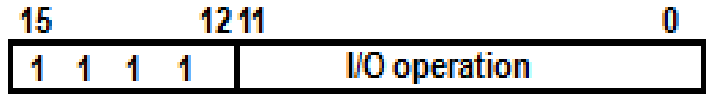
Memory-Reference Instructions (OP-code = 000 ~ 110)



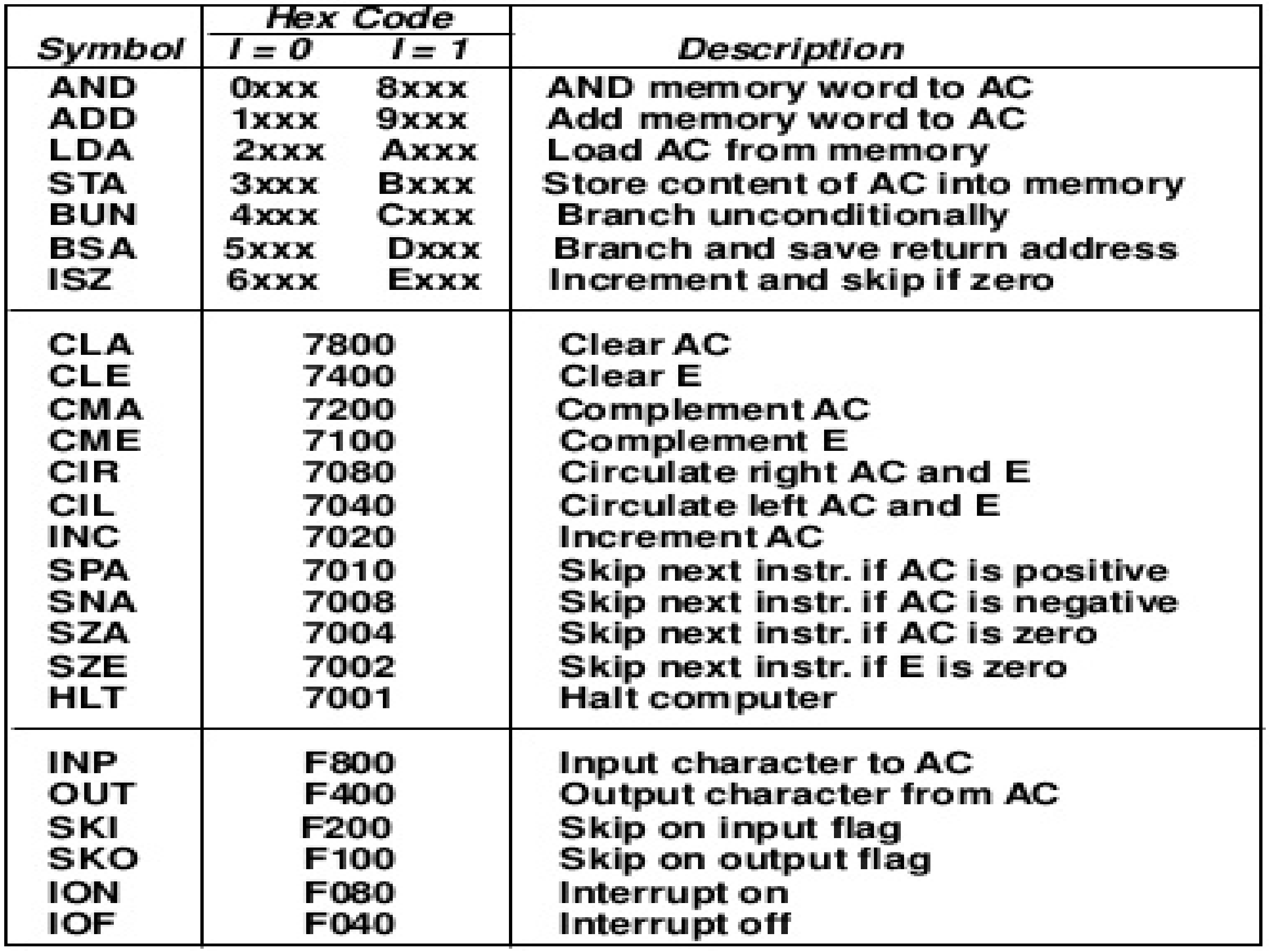
Register-Reference Instructions (OP-code = 111, I = 0)



Input-Output Instructions (OP-code =111, I = 1)



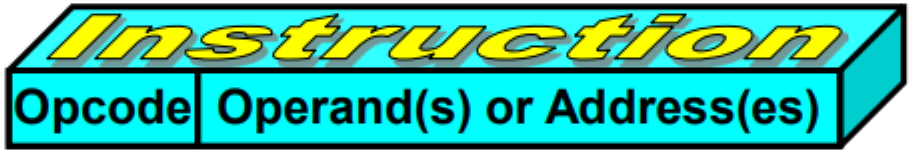
Instruction Format



Symbol	Hex Code I = 0	Hex Code I = 1	Description
AND	0xxx	8xxx	AND memory word to AC
ADD	1xxx	9xxx	Add memory word to AC
LDA	2xxx	Axxx	Load AC from memory
STA	3xxx	Bxxx	Store content of AC into memory
BUN	4xxx	Cxxx	Branch unconditionally
BSA	5xxx	Dxxx	Branch and save return address
ISZ	6xxx	Exxx	Increment and skip if zero
CLA	7800		Clear AC
CLE	7400		Clear E
CMA	7200		Complement AC
CME	7100		Complement E
CIR	7080		Circulate right AC and E
CIL	7040		Circulate left AC and E
INC	7020		Increment AC
SPA	7010		Skip next instr. if AC is positive
SNA	7008		Skip next instr. if AC is negative
SZA	7004		Skip next instr. if AC is zero
SZE	7002		Skip next instr. if E is zero
HLT	7001		Halt computer
INP	F800		Input character to AC
OUT	F400		Output character from AC
SKI	F200		Skip on input flag
SKO	F100		Skip on output flag
ION	F080		Interrupt on
IOF	F040		Interrupt off

Instruction Format

- Three-Address Instructions
  - ADD R1, R2, R3 $R3 \leftarrow [R1] + [R2]$
- Two-Address Instructions
  - ADD R1, R2 $R2 \leftarrow [R1] + [R2]$
- One-Address Instructions
  - ADD M $AC \leftarrow AC + [M]$
- Zero-Address Instructions
  - ADD $TOS \leftarrow [TOS] + [(TOS - 1)]$
- RISC Instructions
  - Lots of registers. Memory is restricted to Load & Store



Instruction Format

Example: Evaluate X = (A+B) ∗ (C+D)

- Three-Address

1.	ADD	A, B, R1	; R1 ← [A] + [B]
2.	ADD	C, D, R2	; R2 ← [C] + [D]
3.	MUL	R1, R2, X	; X ← [R1] ∗ [R2]

Example: Evaluate X = (A+B) ∗ (C+D)

- Two-Address

1.	MOV	A, R1	; R1 ← [A]
2.	ADD	B, R1	; R1 ← [R1] + [B]
3.	MOV	C, R2	; R2 ← [C]
4.	ADD	D, R2	; R2 ← [R2] + [D]
5.	MUL	R2, R1	; R1 ← [R1] ∗ [R2]
6.	MOV	R1, X	; X ← [R1]

Instruction Format

Example: Evaluate X = (A+B) * (C+D)

- One-Address

```
1.  LOAD   A          ; AC ← [A]
2.  ADD    B          ; AC ← [AC] + [B]
3.  STORE  T          ; T ← [AC]
4.  LOAD   C          ; AC ← [C]
5.  ADD    D          ; AC ← [AC] + [D]
6.  MUL    T          ; AC ← [AC] * [T]
7.  STORE  X          ; X ← [AC]
```

Example: Evaluate X = (A+B) * (C+D)

- Zero-Address

```
1.  PUSH   A          ; TOS ← [A]
2.  PUSH   B          ; TOS ← [B]
3.  ADD               ; TOS ← [A] + [B]
4.  PUSH   C          ; TOS ← [C]
5.  PUSH   D          ; TOS ← [D]
6.  ADD               ; TOS ← [C] + [D]
7.  MUL               ; TOS ← (C+D)*(A+B)
8.  POP    X          ; X ← [TOS]
```

Instruction Types

Functional Instructions

- Arithmetic, logic, and shift instructions
- ADD, CMA, INC, CIR, CIL, AND, CLA

Transfer Instructions

- Data transfers between the main memory and the processor registers
- LDA, STA

Control Instructions

- Program sequencing and control
- BUN, BSA, ISZ

Input/Output Instructions

- Input and output
- INP, OUT

Instruction Cycle & Sub Cycle

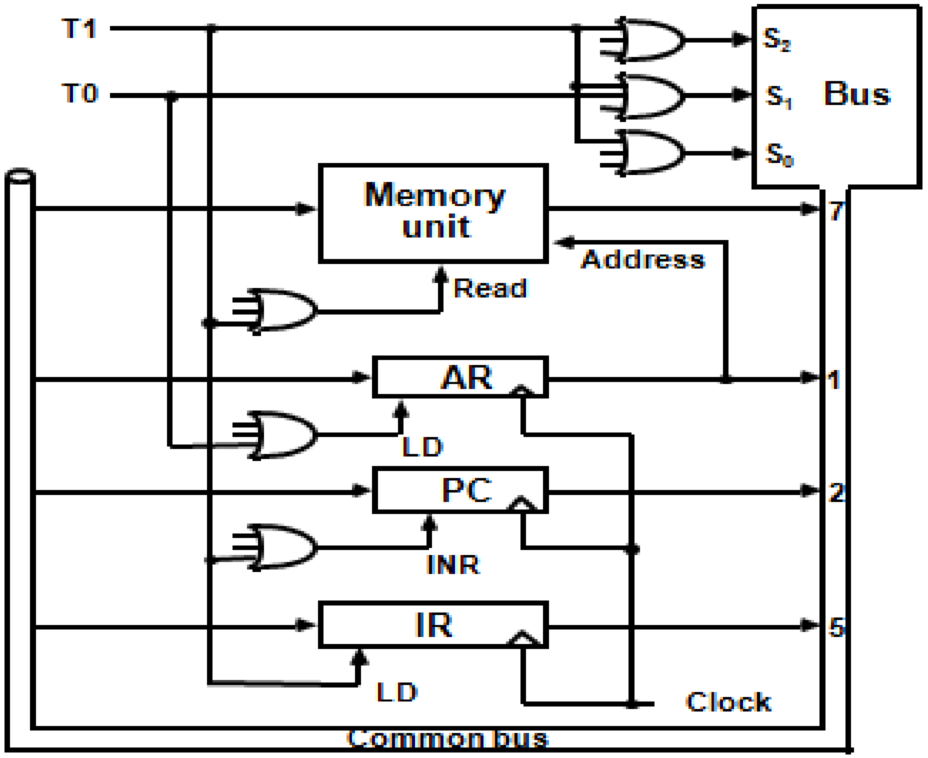
- In Basic Computer, a machine instruction is executed in the following cycle:
  - Fetch an instruction from memory
  - Decode the instruction
  - Read the effective address from memory if the instruction has an indirect address
  - Execute the instruction

- After an instruction is executed, the cycle starts again at step 1, for the next instruction
- *Note*: Every different processor has its own (different) instruction cycle

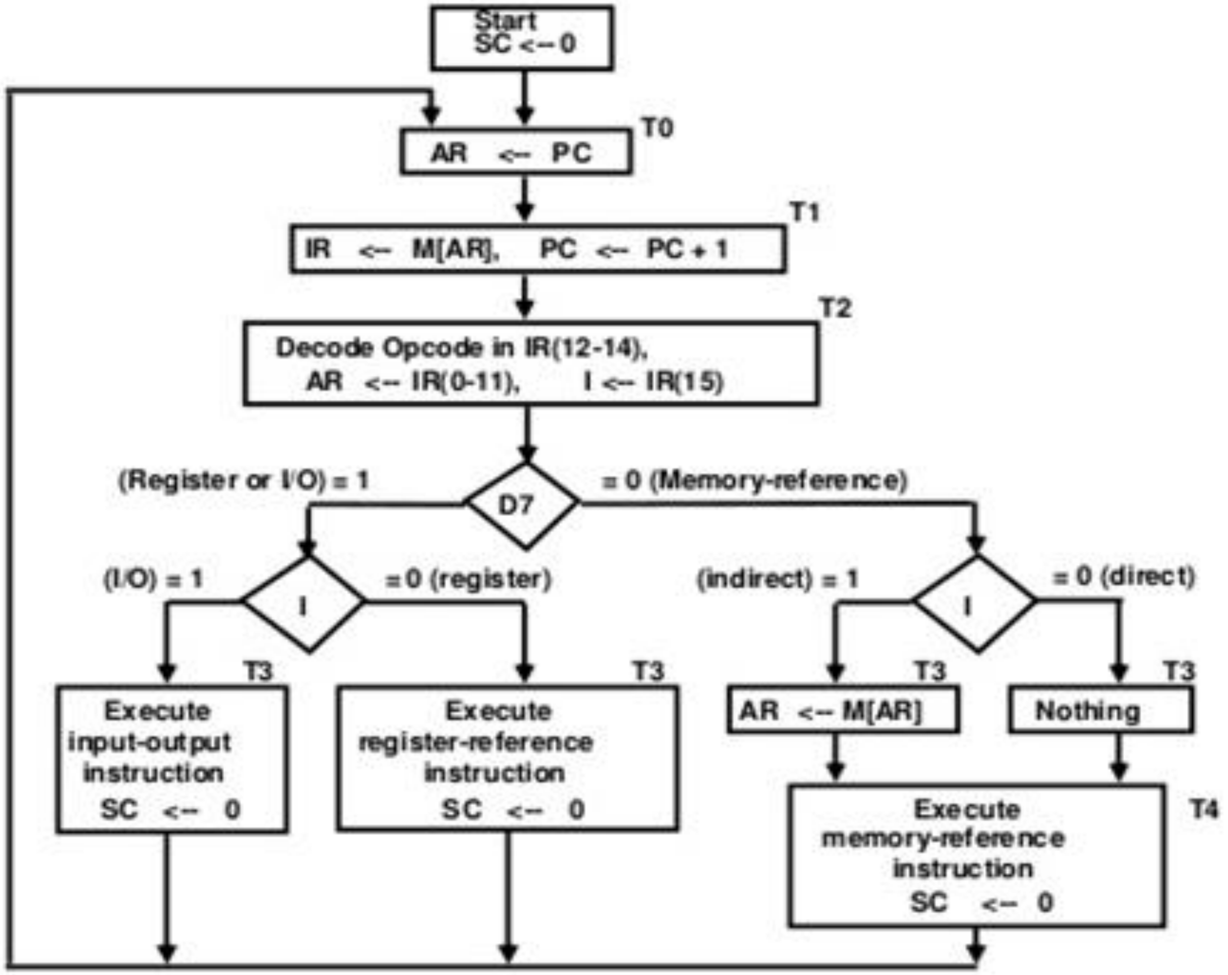
Fetch and Decode

- **Fetch and Decode**

T0: AR ← PC ($S_0S_1S_2$=010, T0=1)
T1: IR ← M [AR], PC ← PC + 1 (S0S1S2=111, T1=1)
T2: D0, . . . , D7 ← Decode IR(12-14), AR ← IR(0-11), I ← IR(15)

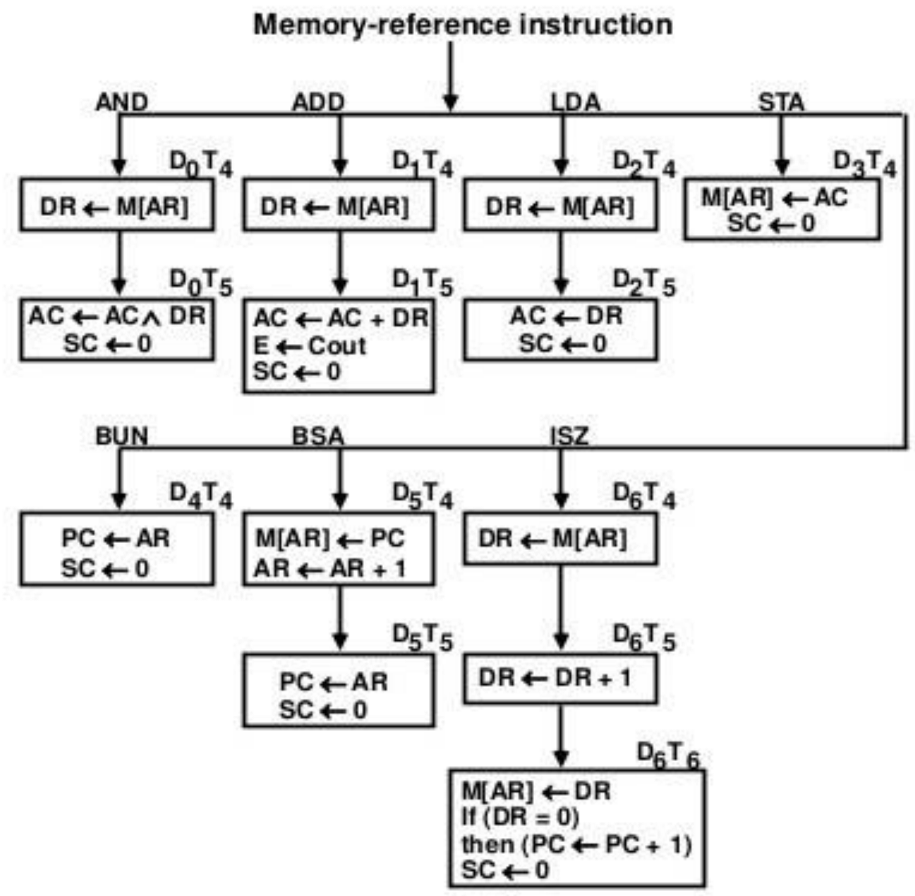


Instruction Cycle

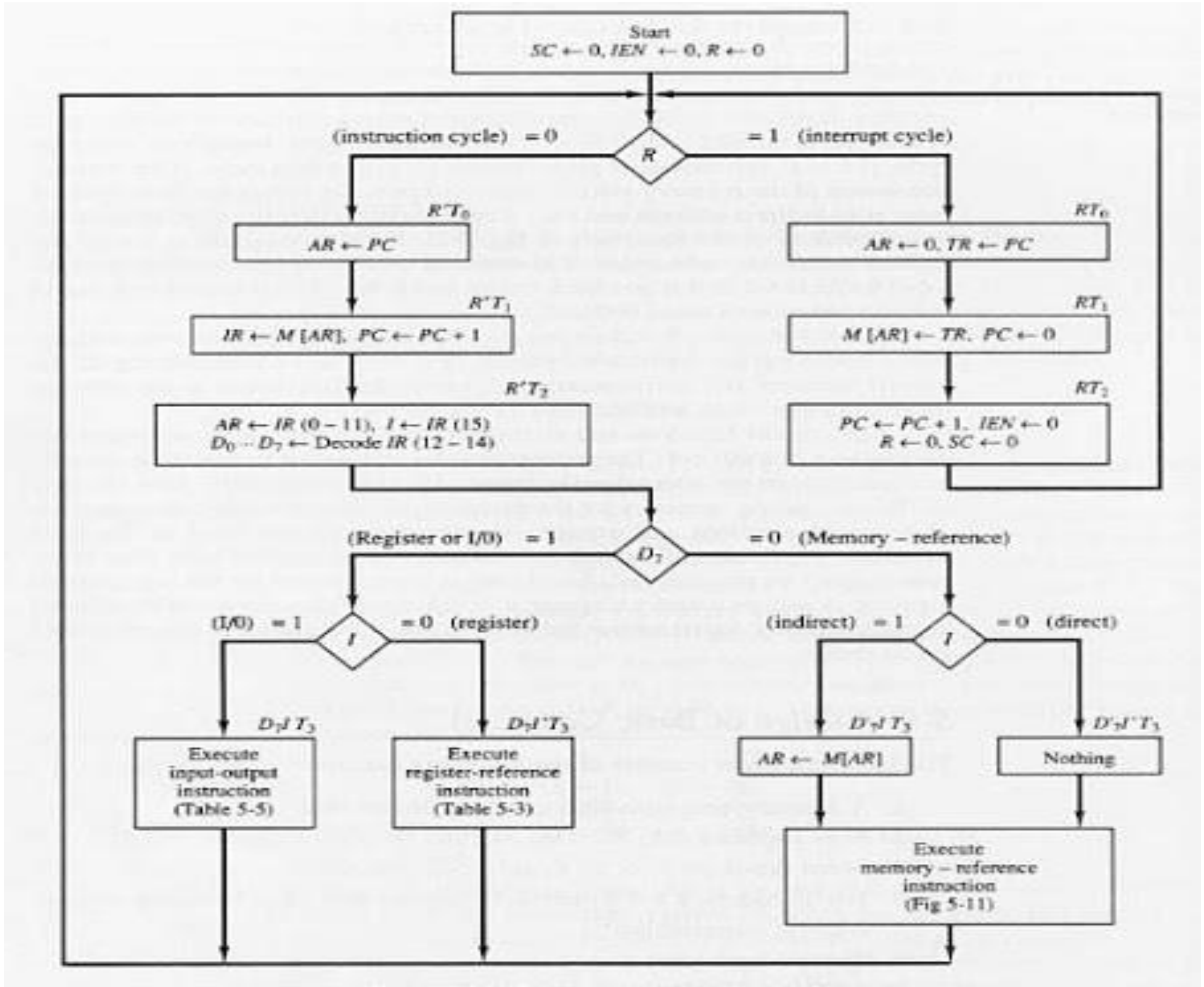

Start
SC <-- 0
T0
AR <-- PC
T1
IR <-- M[AR], PC <-- PC + 1
T2
Decode Opcode in IR(12-14),
AR <-- IR(0-11), I <-- IR(15)
(Register or I/O) = 1
D7
= 0 (Memory-reference)
(I/O) = 1
I
= 0 (register)
(indirect) = 1
I
= 0 (direct)
T3
Execute input-output instruction SC <-- 0
T3
Execute register-reference instruction SC <-- 0
T3
AR <-- M[AR]
T3
Nothing
T4
Execute memory-reference instruction SC <-- 0

Instruction Cycle

FLOWCHART FOR MEMORY REFERENCE INSTRUCTIONS



Instruction Cycle with Interrupt



Start
$SC \leftarrow 0, IEN \leftarrow 0, R \leftarrow 0$
(instruction cycle) = 0
R
= 1 (interrupt cycle)
$R'T_0$
$AR \leftarrow PC$
$R'T_1$
$IR \leftarrow M[AR], PC \leftarrow PC + 1$
$R'T_2$
$AR \leftarrow IR(0-11), I \leftarrow IR(15)$
$D_0 \cdots D_7 \leftarrow$ Decode $IR(12-14)$
RT_0
$AR \leftarrow 0, TR \leftarrow PC$
RT_1
$M[AR] \leftarrow TR, PC \leftarrow 0$
RT_2
$PC \leftarrow PC + 1, IEN \leftarrow 0$
$R \leftarrow 0, SC \leftarrow 0$
(Register or I/O) = 1
D_7
= 0 (Memory – reference)
(I/O) = 1
I
= 0 (register)
(indirect) = 1
I
= 0 (direct)
D_7IT_3
Execute input-output instruction (Table 5-5)
$D_7I'T_3$
Execute register-reference instruction (Table 5-3)
D'_7IT_3
$AR \leftarrow M[AR]$
$D'_7I'T_3$
Nothing
Execute memory – reference instruction (Fig 5-11)

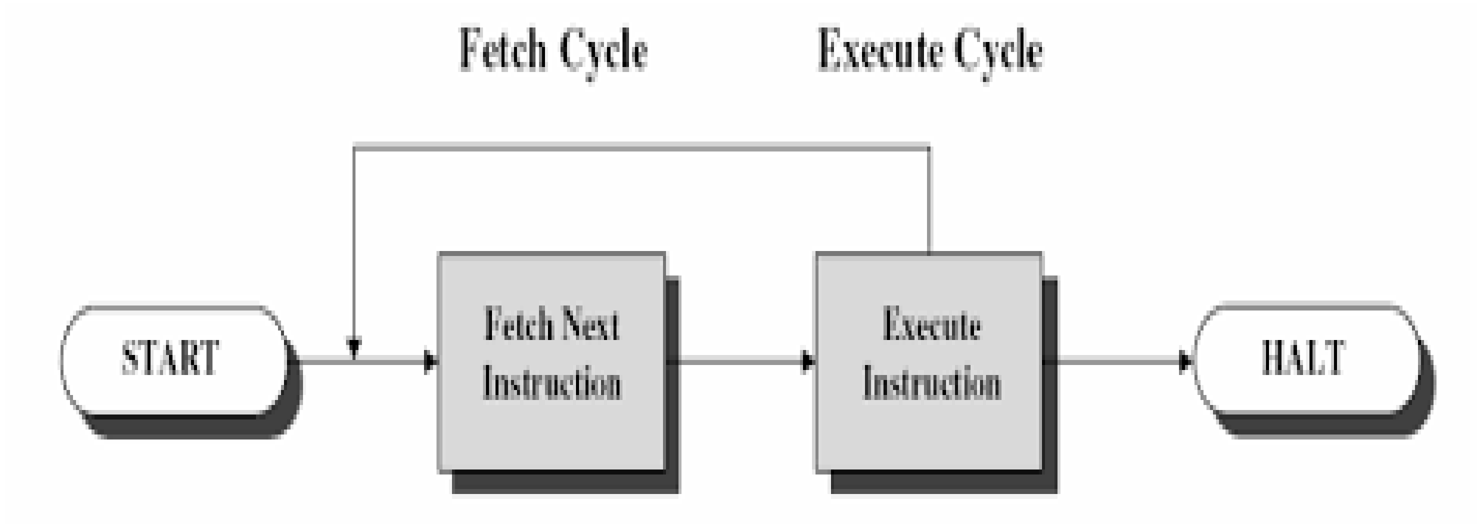
Figure 5-15 Flowchart for computer operation.

Instruction Cycle & Sub Cycle

Instruction Cycle

•The time period during which one instruction is fetched from memory and executed when a computer is given an instruction in machine language.

•There are typically four stages of an instruction cycle that the CPU carries out:



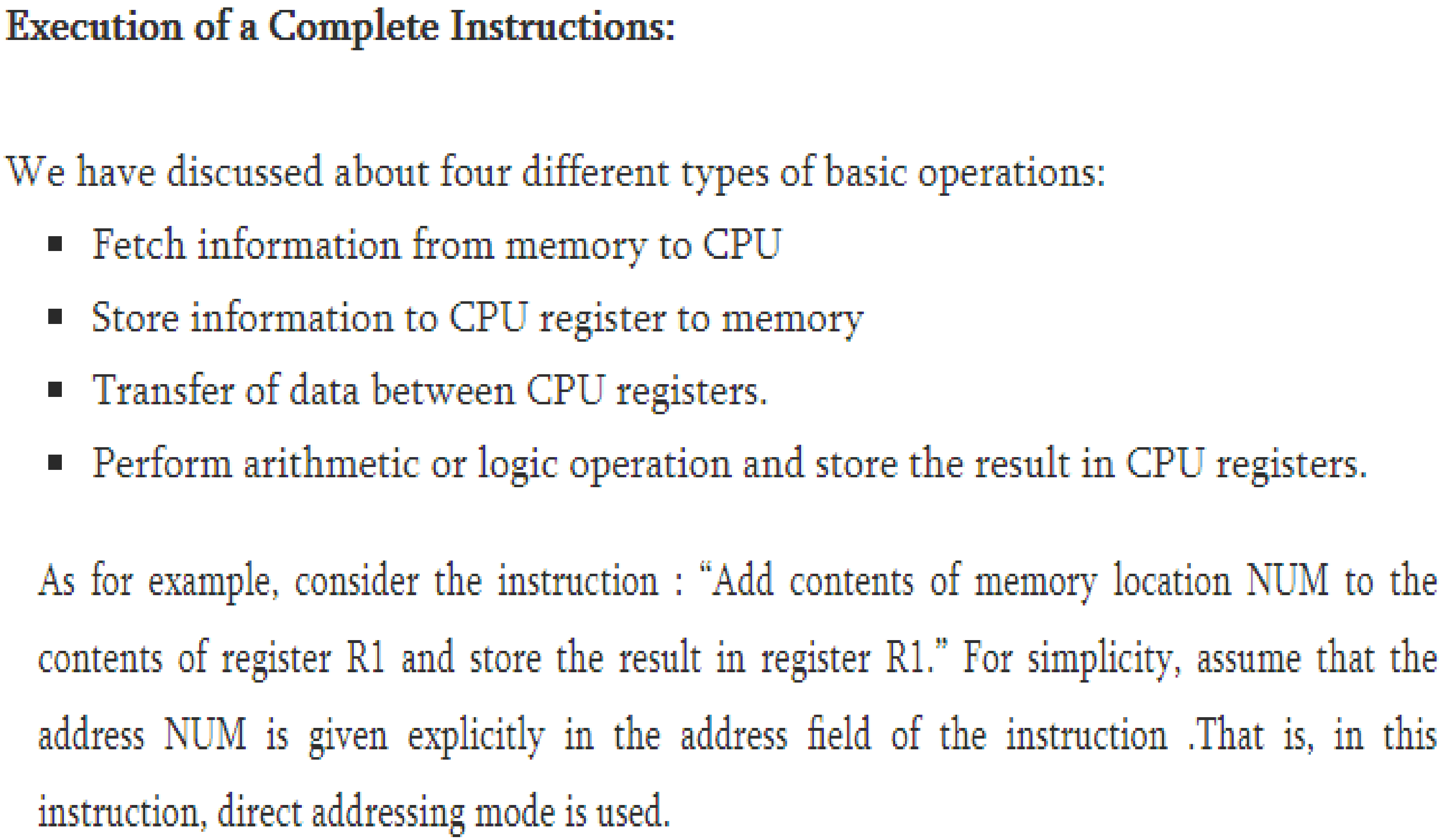
Instruction Cycle & Sub Cycle

Instruction Cycle

There are typically four stages of an instruction cycle that the CPU carries out:

- **Fetch** the instruction from memory. This step brings the instruction into the *instruction register*, a circuit that holds the instruction so that it can be decoded and executed.
- Decode the instruction.
- Read the effective address from memory if the instruction has an indirect address.
- **Execute** the instruction.

Execution of Complete Instruction



Execution of a Complete Instructions:

We have discussed about four different types of basic operations:

- Fetch information from memory to CPU
- Store information to CPU register to memory
- Transfer of data between CPU registers.
- Perform arithmetic or logic operation and store the result in CPU registers.

As for example, consider the instruction : "Add contents of memory location NUM to the contents of register R1 and store the result in register R1." For simplicity, assume that the address NUM is given explicitly in the address field of the instruction .That is, in this instruction, direct addressing mode is used.

Execution of Complete Instruction

Execution of this instruction requires the following action :

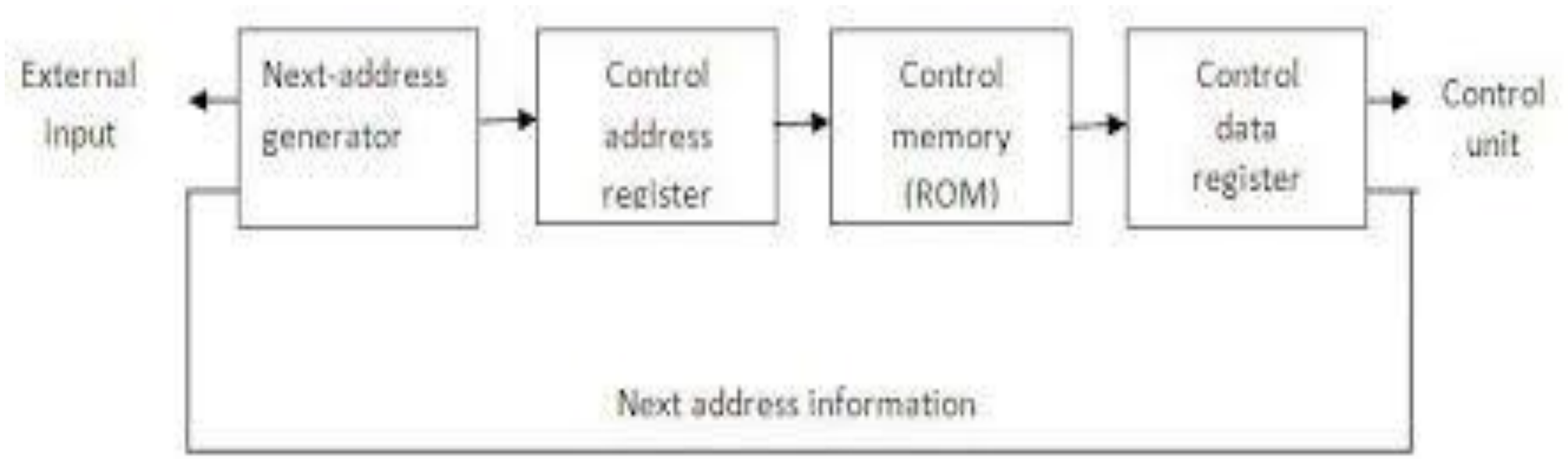
1. Fetch instruction
2. Fetch first operand (Contents of memory location pointed at by the address field of the instruction)
3. Perform addition
4. Load the result into R1.

Following sequence of control steps are required to implement the above operation for the single-bus architecture that we have discussed in earlier section.

Execution of Complete Instruction

Steps	Actions
1.	PC_{out}, MAR_{in}, Read, Clear Y, Set carry -in to ALU, Add, Z_{in}
2.	Z_{out}, PC_{in}, Wait For MFC
3.	MDR_{out}, Ir_{in}
4.	Address-field- of-IR_{out}, MAR_{in}, Read
5.	$R1_{out}$, Y_{in}, Wait for MFC
6.	MDR_{out}, Add, Z_{in}
7.	Z_{out}, $R1_{in}$
8.	END

Micro Programmed Control Organization



Micro programmed Control

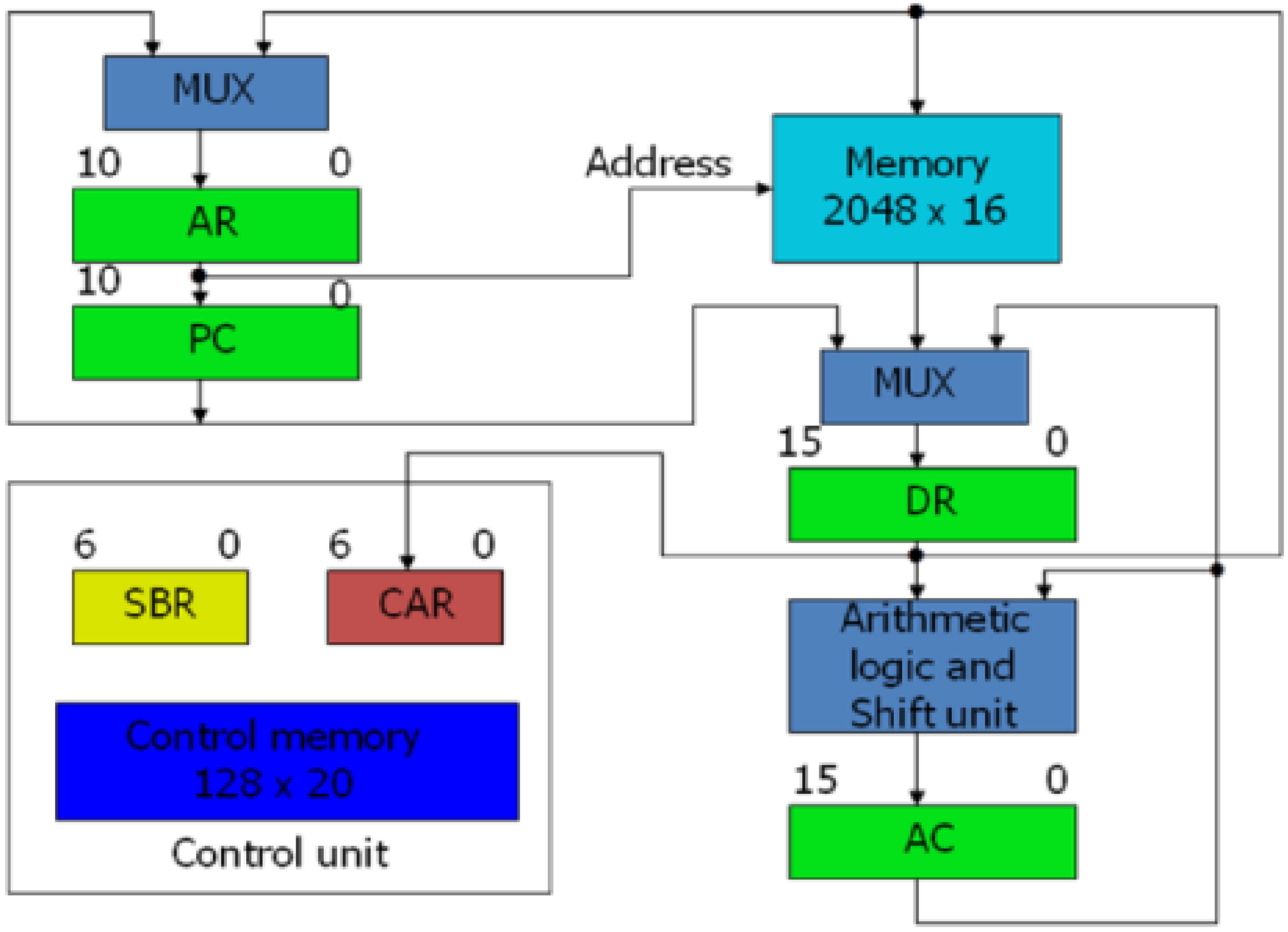
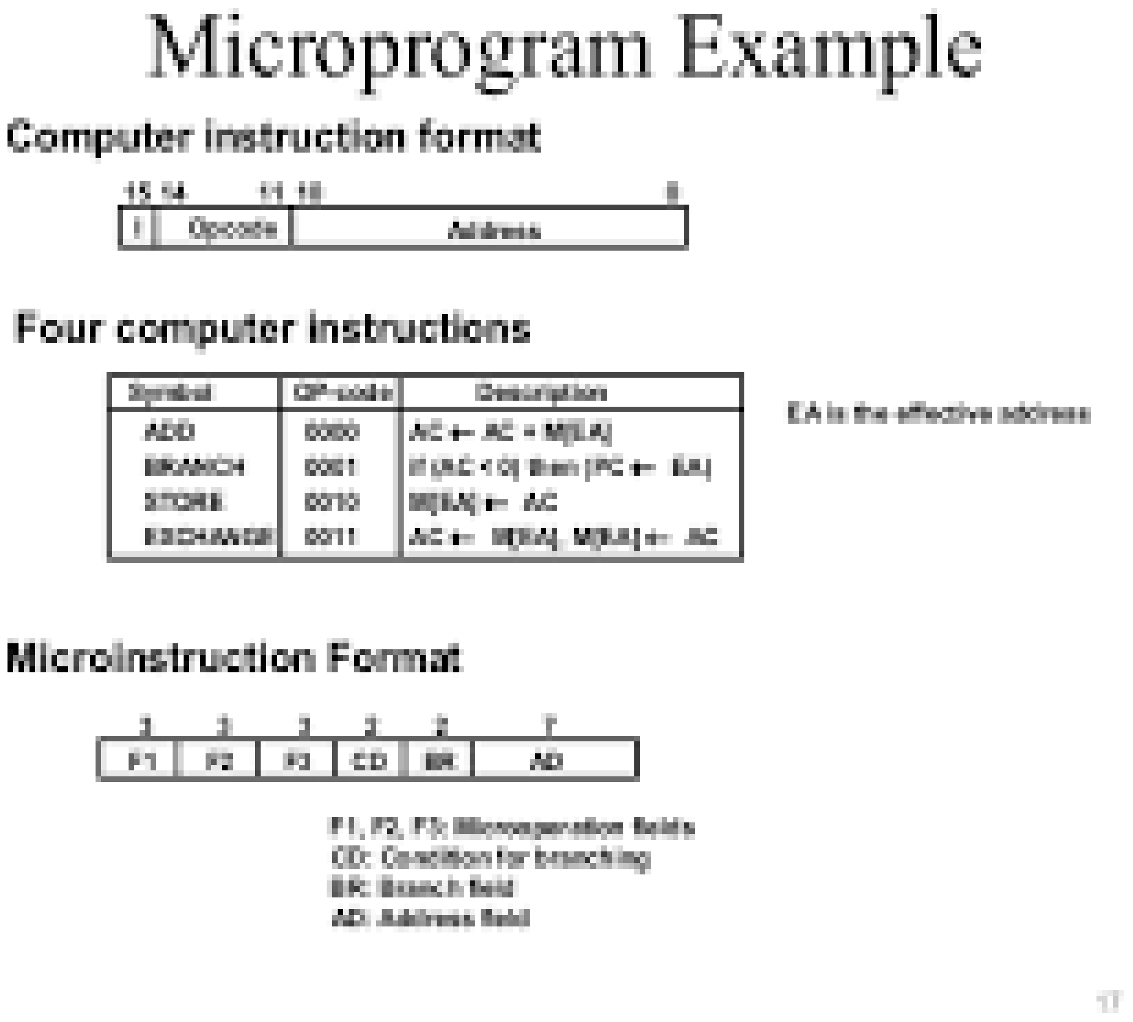


Fig 3-4: Computer hardware configuration

Micro programmed Control



Microprogram Example

Computer instruction format

15	14 11	10 0
I	Opcode	Address

Four computer instructions

Symbol	OP-code	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

Microinstruction Format

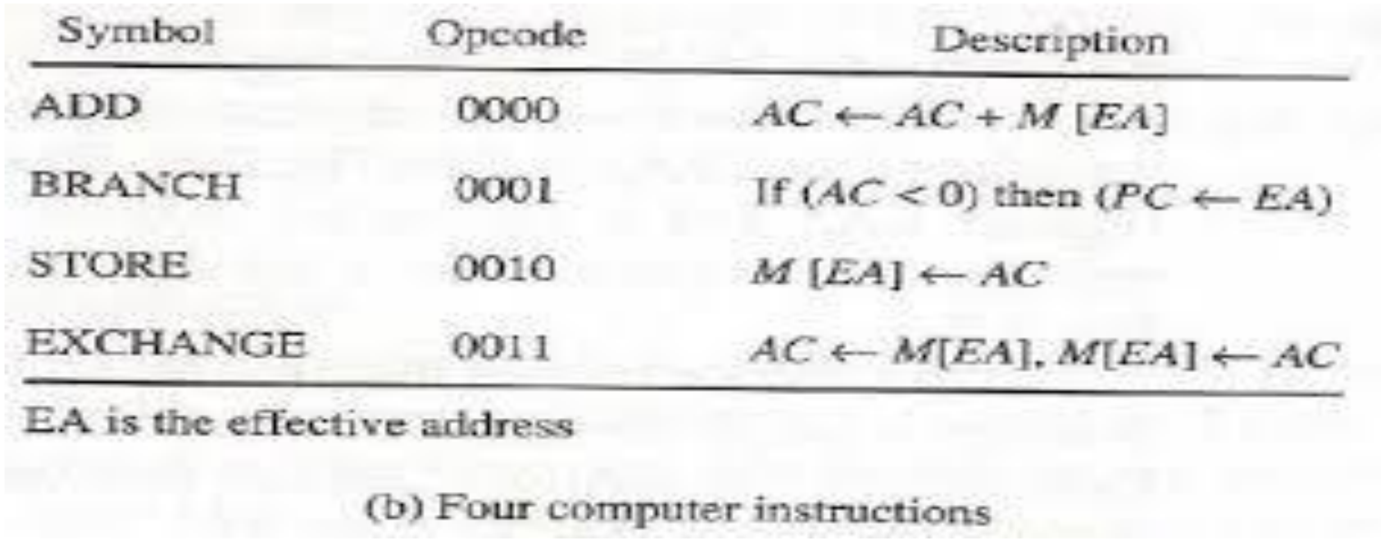
3	3	3	2	2	7
F1	F2	F3	CD	BR	AD

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

17

Micro programmed Control

Instruction Format:

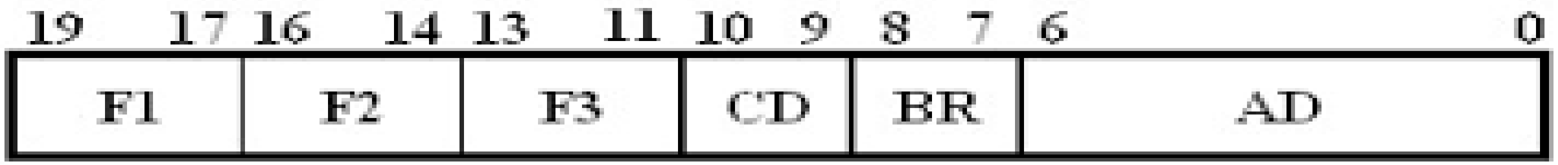


Symbol	Opcode	Description
ADD	0000	$AC \leftarrow AC + M[EA]$
BRANCH	0001	If $(AC < 0)$ then $(PC \leftarrow EA)$
STORE	0010	$M[EA] \leftarrow AC$
EXCHANGE	0011	$AC \leftarrow M[EA], M[EA] \leftarrow AC$

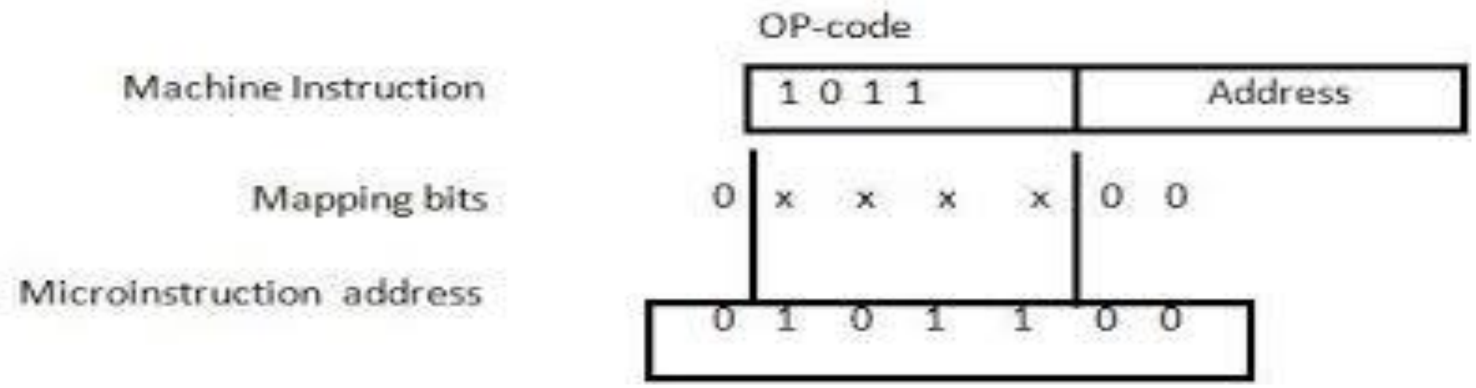
EA is the effective address

(b) Four computer instructions

Micro programmed Control



F1, F2, F3:		Microoperation Field, each field 3-bits
CD	:	Conditional for Branching 2-bits Field
BR	:	Branch 2-bits Field
AD	:	Address 7-bits Field



Control

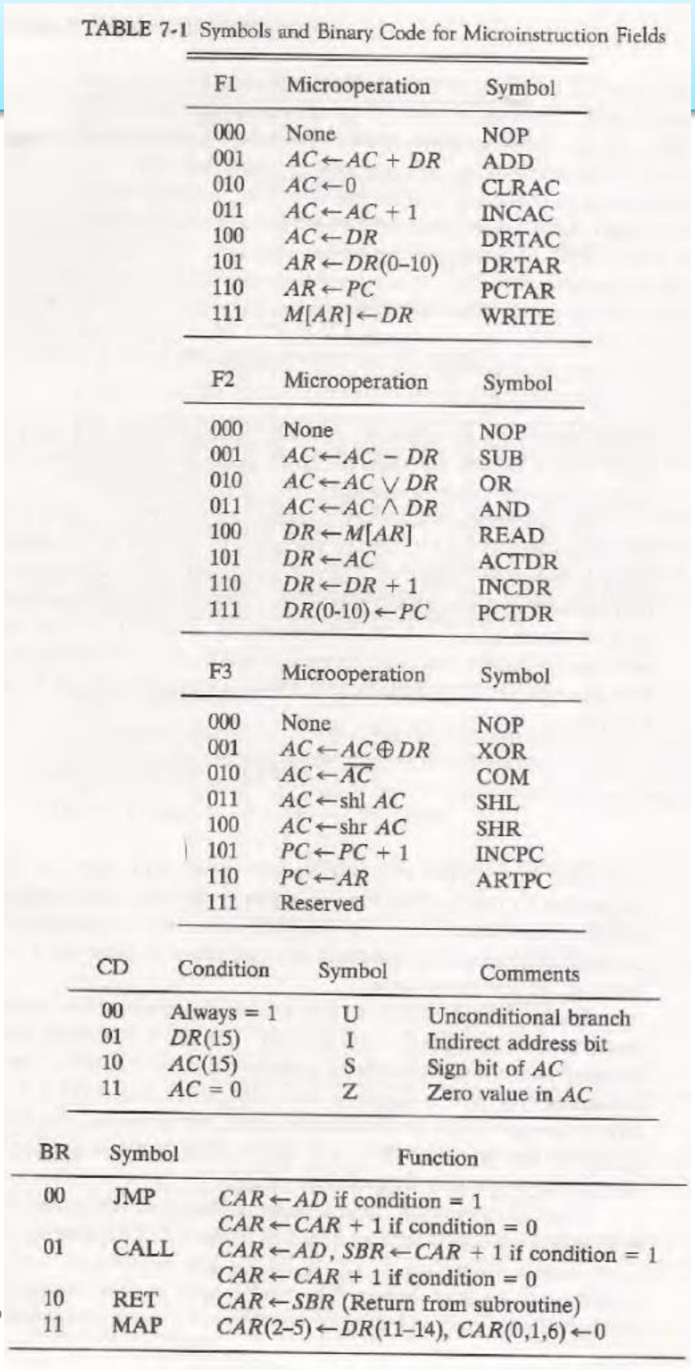


TABLE 7-1 Symbols and Binary Code for Microinstruction Fields

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	$AR \leftarrow DR(0–10)$	DRTAR
110	$AR \leftarrow PC$	PCTAR
111	$M[AR] \leftarrow DR$	WRITE

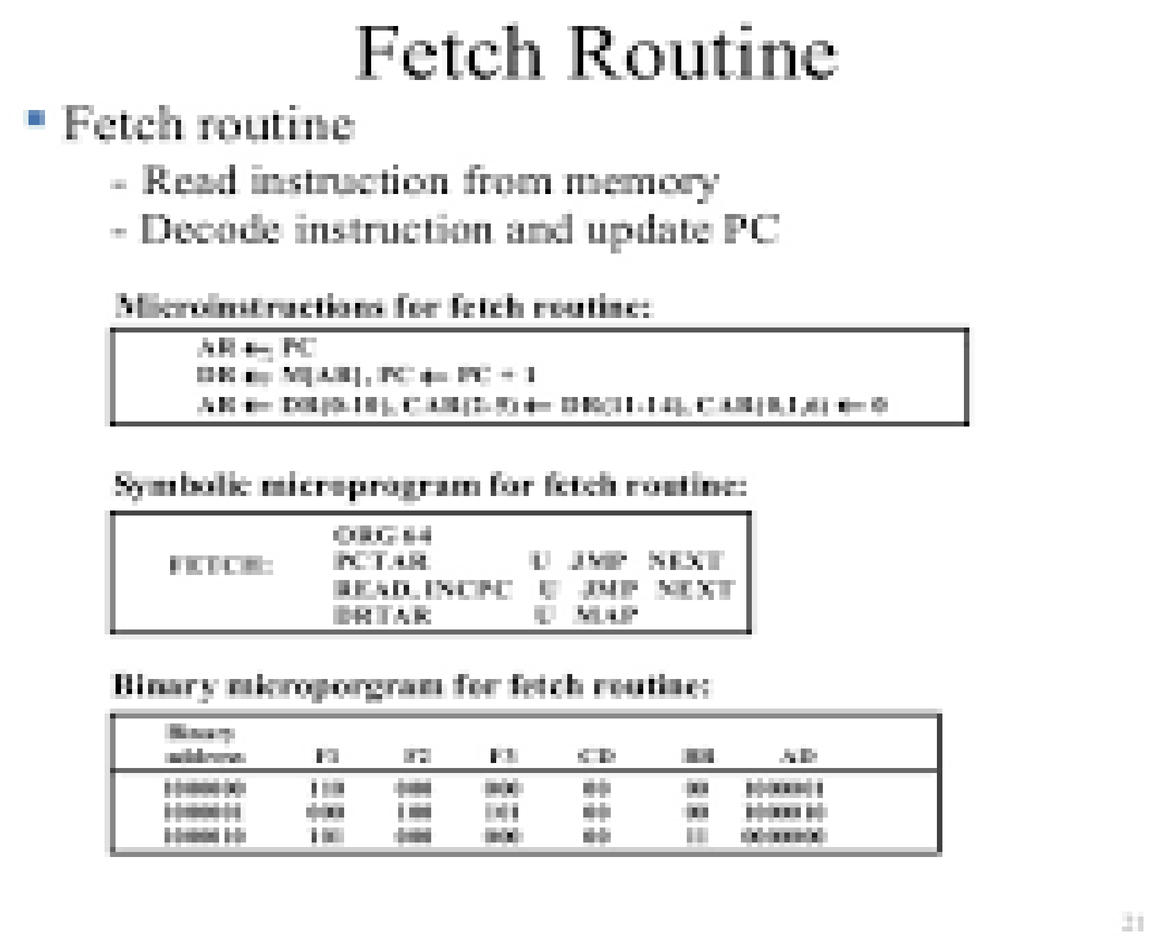
F2	Microoperation	Symbol
000	None	NOP
001	$AC \leftarrow AC - DR$	SUB
010	$AC \leftarrow AC \vee DR$	OR
011	$AC \leftarrow AC \wedge DR$	AND
100	$DR \leftarrow M[AR]$	READ
101	$DR \leftarrow AC$	ACTDR
110	$DR \leftarrow DR + 1$	INCDR
111	$DR(0\text{-}10) \leftarrow PC$	PCTDR

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

Micro programmed Control



Fetch Routine

- Fetch routine
  - Read instruction from memory
  - Decode instruction and update PC

Microinstructions for fetch routine:

```
AR ← PC
DR ← M[AR], PC ← PC + 1
AR ← DR(0-10), CAR(2-5) ← DR(11-14), CAR(0,1,6) ← 0
```

Symbolic microprogram for fetch routine:

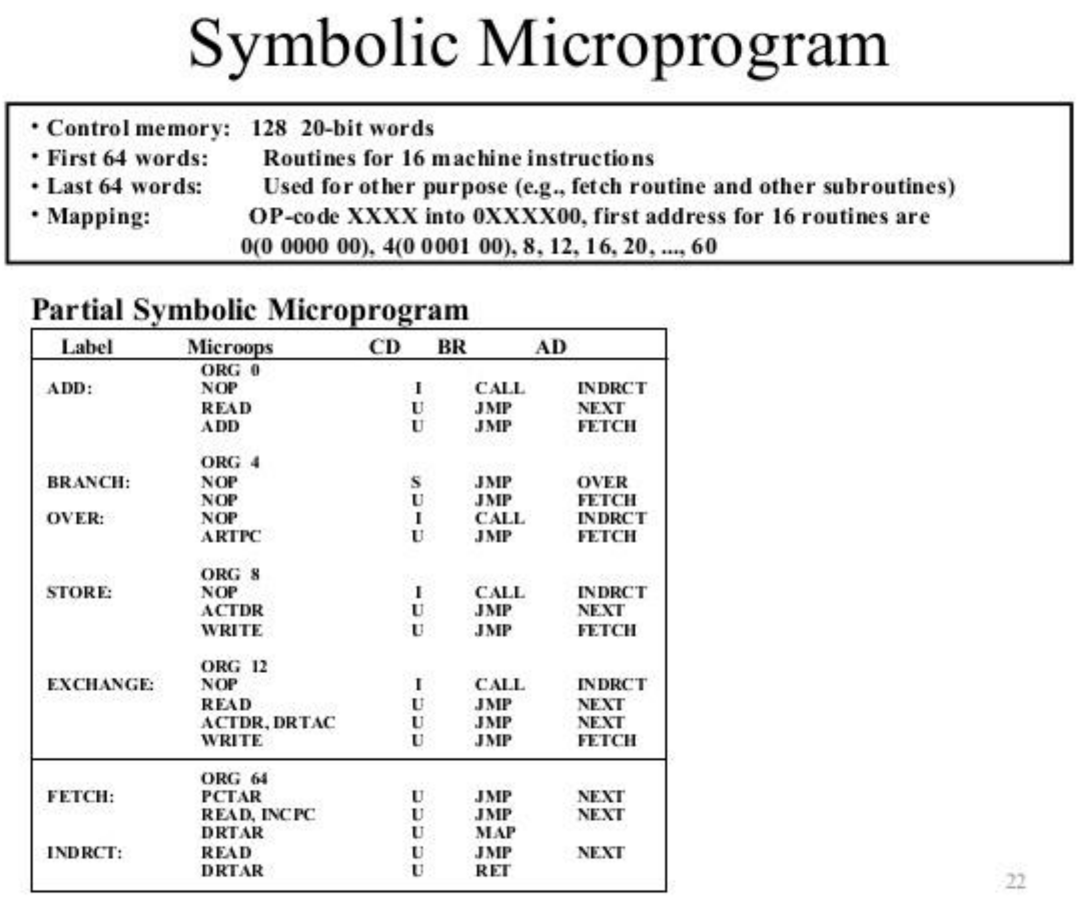
```
          ORG 64
FETCH:    PCTAR          U  JMP  NEXT
          READ, INCPC    U  JMP  NEXT
          DRTAR          U  MAP
```

Binary microporgram for fetch routine:

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

21

Micro programmed Control



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

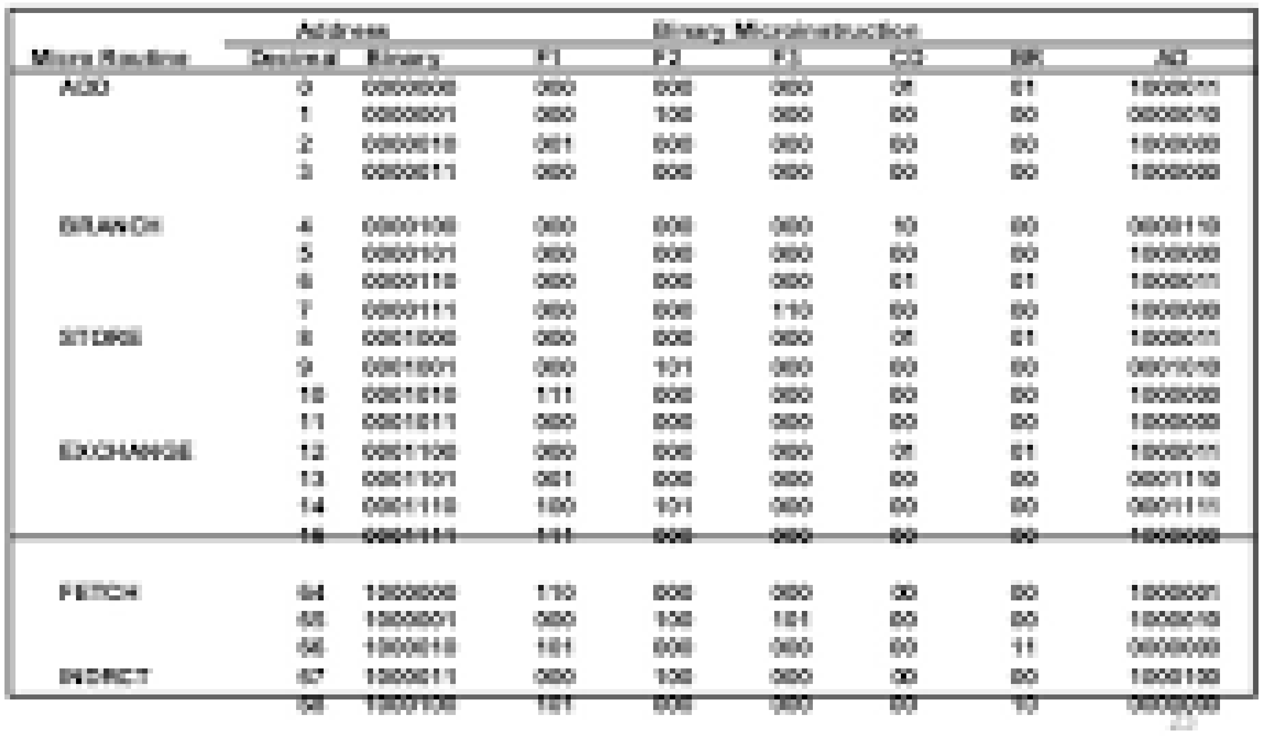
Partial Symbolic Microprogram

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	

22

Micro programmed Control

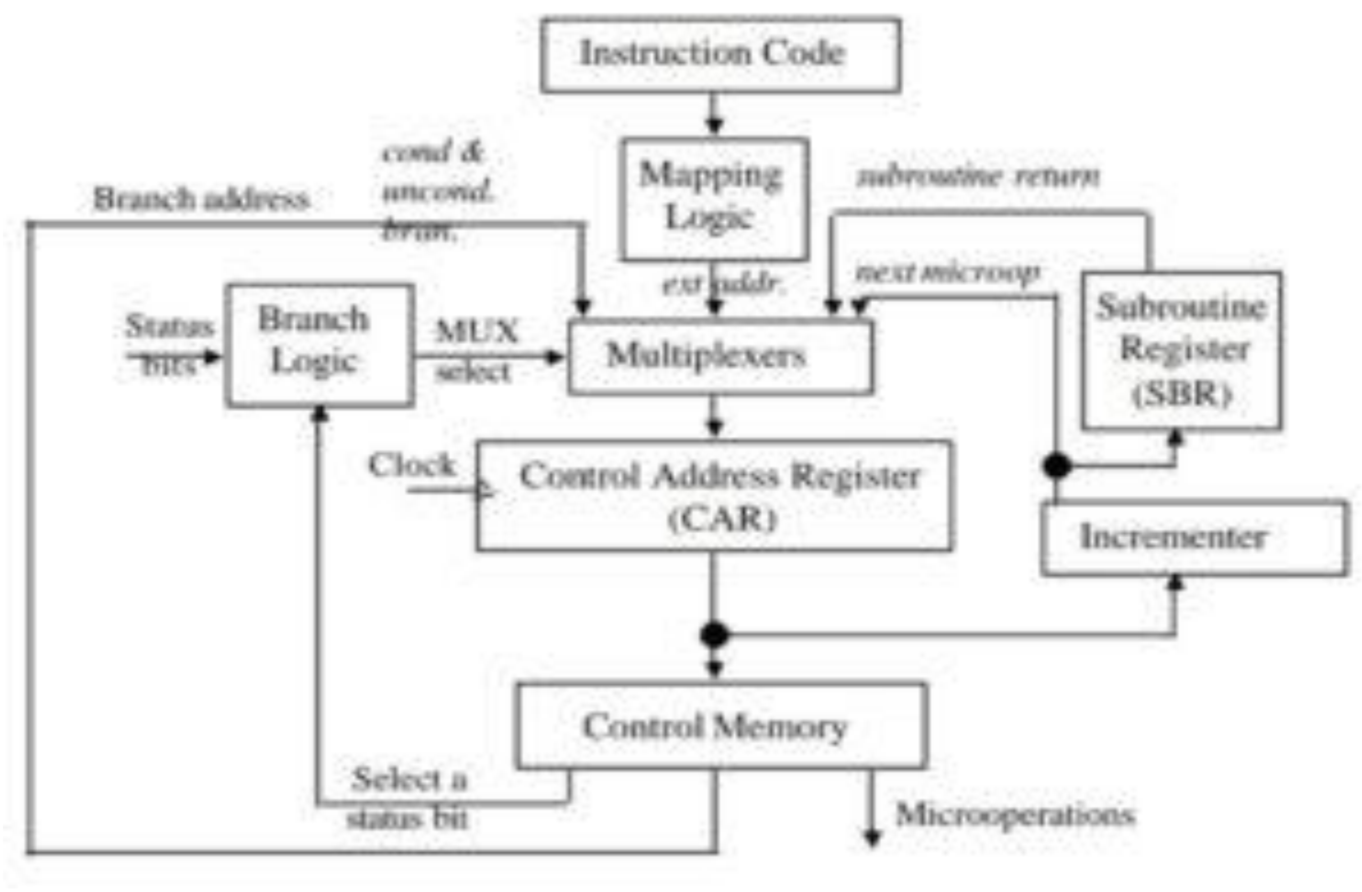
Binary Microprogram



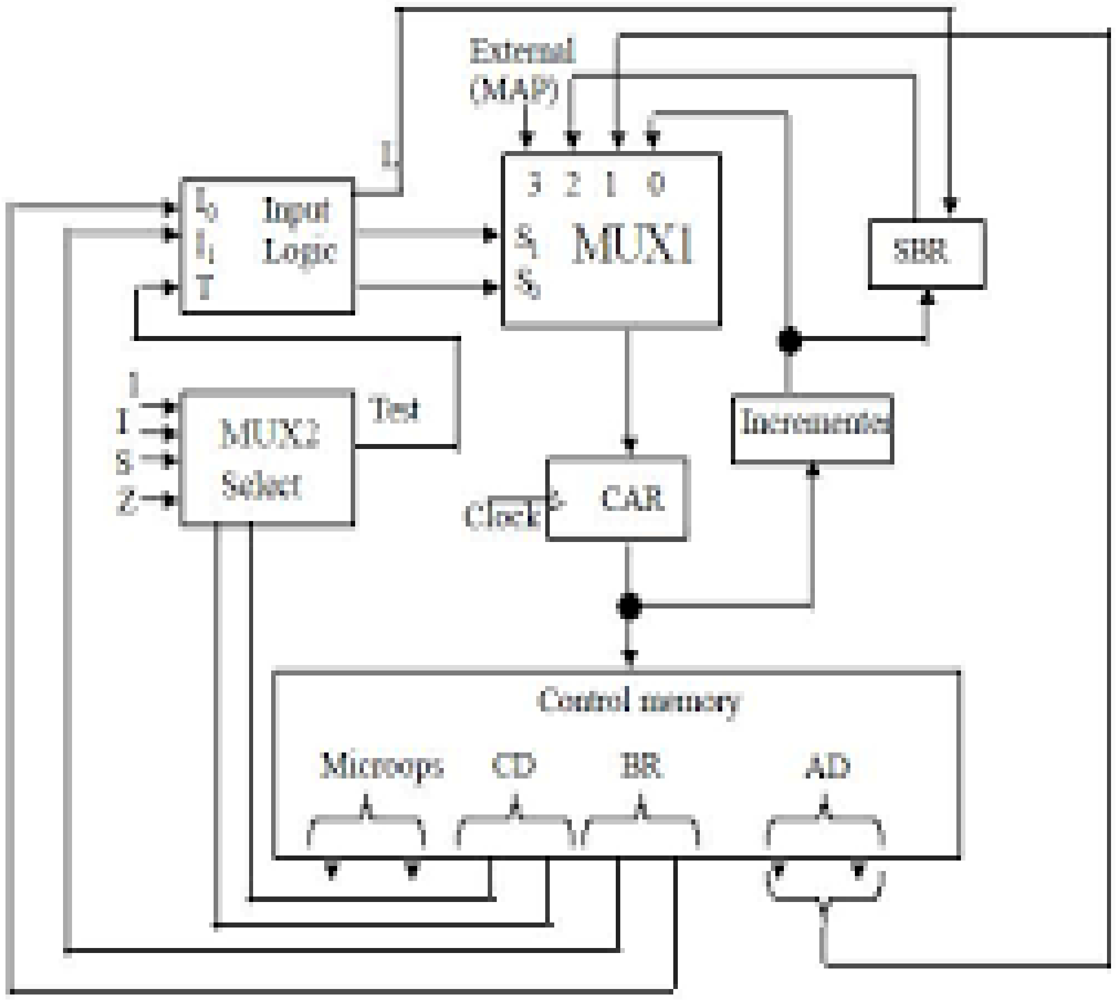
	Address		Binary Microinstruction					
Micro Routine	Decimal	Binary	F1	F2	F3	CD	BR	AD
ADD	0	0000000	000	000	000	01	01	1000011
	1	0000001	000	100	000	00	00	0000010
	2	0000010	001	000	000	00	00	1000000
	3	0000011	000	000	000	00	00	1000000
BRANCH	4	0000100	000	000	000	10	00	0000110
	5	0000101	000	000	000	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

Micro programmed Control

Selection Of Address For Control Memory

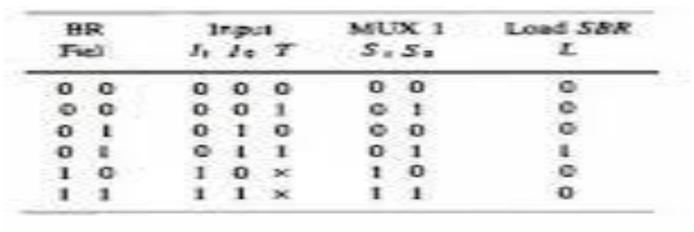


Micro programmed Control



Micro programmed Control

Table: Input Logic Truth Table for micro-program sequencer



BR Field		Input			MUX 1		Load SBR
		I_1	I_0	T	S_1	S_0	L
0	0	0	0	0	0	0	0
0	0	0	0	1	0	1	0
0	1	0	1	0	0	0	0
0	1	0	1	1	0	1	1
1	0	1	0	×	1	0	0
1	1	1	1	×	1	1	0

Micro operations

The operations performed on the data stored in registers known as micro-operations. Example: Shift, Count Clear and Load. The micro-operations are classified as follows.

1.Register transfer micro-operations: These type of micro operations are used to transfer from one register to another binary information.

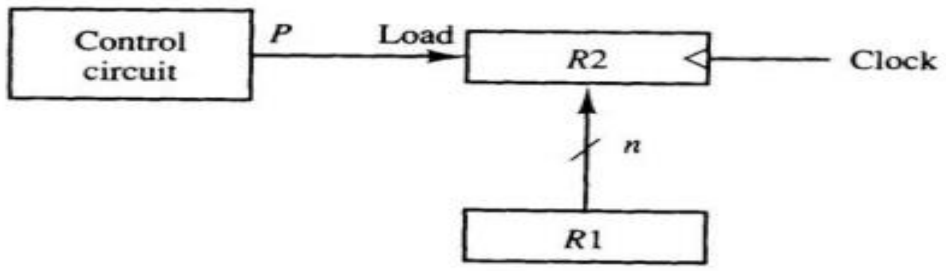
2.Arithmetic micro-operations : These micro-operations are used to perform on numeric data stored in the registers some arithmetic operations.

3.Logic micro-operations: These micro operations are used to perform bit style operations / manipulations on non numeric data.

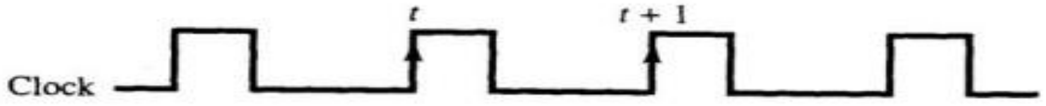
4.Shift micro operations: As their name suggests they are used to perform shift operations in data store in registers.

Micro operations

1. Register transfer micro-operations: These type of micro operations are used to transfer from one register to another binary information.
2. Designate information transfer from one register to another by
   R2 ← R1
   If the transfer is to occur only under a predetermined control condition, designate it by If (P = 1) then (R2 ← R1) or, P: R2 ← R1, where P is a control function that can be either 0 or 1



(a) Block diagram



Micro operations

Arithmetic Micro operations

Example	Description
$R3 \leftarrow R1 + R2$	Addition
$R3 \leftarrow R1 - R2$ $(R1 + R2' + 1)$	Subtraction
$R2 \leftarrow R2'$	Complement (really a logic operation)
$R2 \leftarrow -R2$ $(R2' + 1)$	Negation
$R1 \leftarrow R1 + 1$	Increment
$R1 \leftarrow R1 - 1$	Decrement

Micro operations

Logic Micro operations

Logic Micro-Operations: individual bits of registers are operated with other corresponding register bits. Example: the XOR of R2 and R1 is symbolized by

P: $R1 \leftarrow R1 \oplus R2$

Example: R1 = 1010 and R2 = 1100

1010

Content of R1

1100

Content of R2

0110

Content of R1 after P = 1

Micro operations

Shift Micro operations

Shift Micro-Operations: – these operations are used for serial transfer of data. They are also used in conjunction with arithmetic, logic, and other data-processing operation.
The content of register can be shifted to the left or to the right. At the same time the bits are shifted, the flip flop receives the binary information from the serial input.
There are three types of shift micro operation-
1. **Logic shift** 2. **Circular shift** 3. **Arithmetic shift**

Logical Shift:- The symbol **"shl"** is used for **logical shift left** and **"shr"** is used for **logical shift right**.

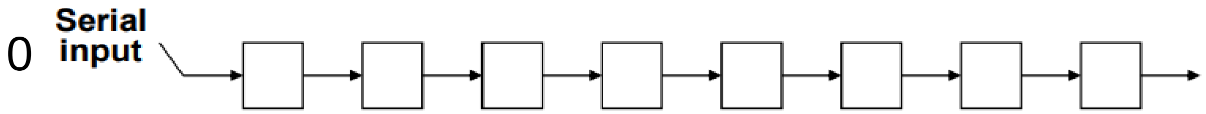
Micro operations

A *logical* shift is one that transfers 0 through the serial input. We will adopt the symbols *shl* and *shr* for logical shift-left and shift-right microoperations. For example:

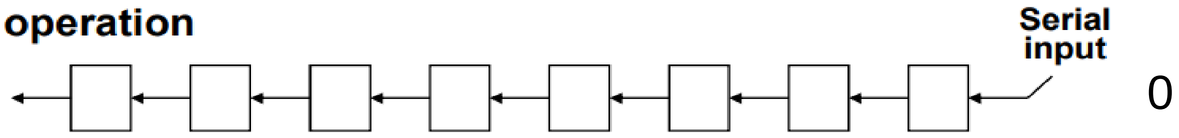
$$R1 \leftarrow \text{shl } R1$$

$$R2 \leftarrow \text{shr } R2$$

- **A right shift operation**



- **A left shift operation**

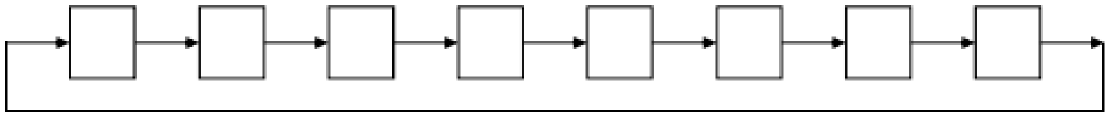


Micro operations

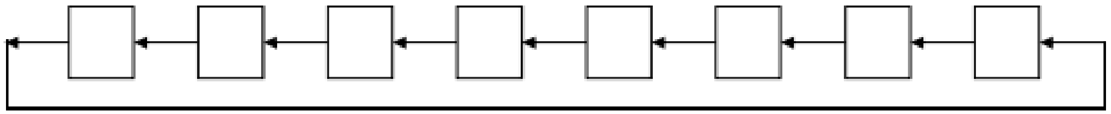
CIRCULAR SHIFT

- In a circular shift the serial input is the bit that is shifted out of the other end of the register.

- A right circular shift operation:



- A left circular shift operation:

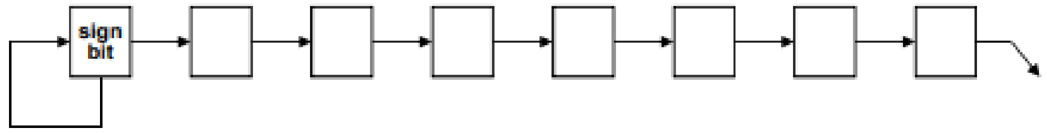


- In a RTL, the following notation is used
  - *cil* for a circular shift left
  - *cir* for a circular shift right
  - Examples:
    - R2 ← *cir* R2
    - R3 ← *cil* R3

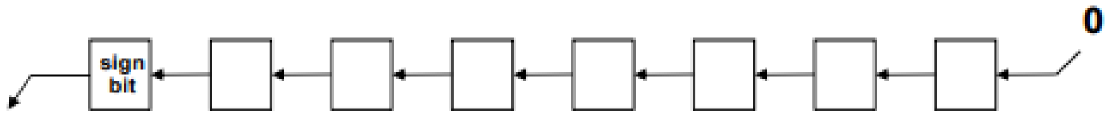
Micro operations

ARITHMETIC SHIFT

- An arithmetic shift is meant for signed binary numbers (integer)
- An arithmetic left shift multiplies a signed number by two
- An arithmetic right shift divides a signed number by two
- The main distinction of an arithmetic shift is that it must keep the sign of the number the same as it performs the multiplication or division



- A right arithmetic shift operation:

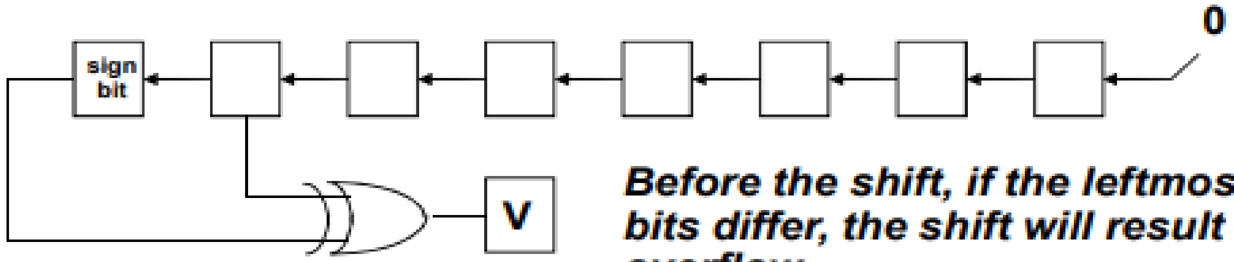


- A left arithmetic shift operation:

Micro operations

ARITHMETIC SHIFT

- An left arithmetic shift operation must be checked for the overflow



Before the shift, if the leftmost two bits differ, the shift will result in an overflow

- In a RTL, the following notation is used
  - *ashl* for an arithmetic shift left
  - *ashr* for an arithmetic shift right
  - Examples:
    - R2 $\leftarrow$ *ashr* R2
    - R3 $\leftarrow$ *ashl* R3

Concept of horizontal and vertical microprogramming.

Micro-programmed control unit

Micro-programmed control unit can be classified into two types based on the type of Control Word stored in the Control Memory.

- Horizontal micro-programmed control unit
- Vertical micro-programmed control unit.

Horizontal micro-programmed control unit, the control signals are represented in the decoded binary format, i.e., 1 bit/CS. Here 'n' control signals require n bit encoding.

- In horizontal organization, as mentioned above, you can assume that every bit in the control word corresponds to a control signal.
- Horizontal organization has more control over the potential parallelism of operations in the data path; however, it uses up lots of control store.

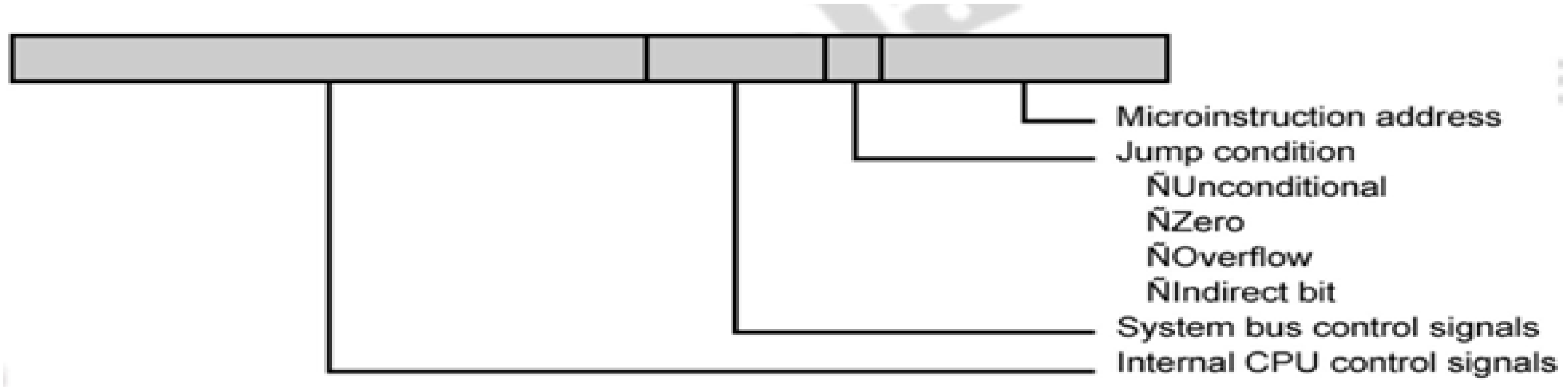
Concept of horizontal and vertical microprogramming.

Vertical micro-programmed control unit, the control signals are represented in the encoded binary format. Here ‘n’ control signals require $\log_2 n$ bit encoding.

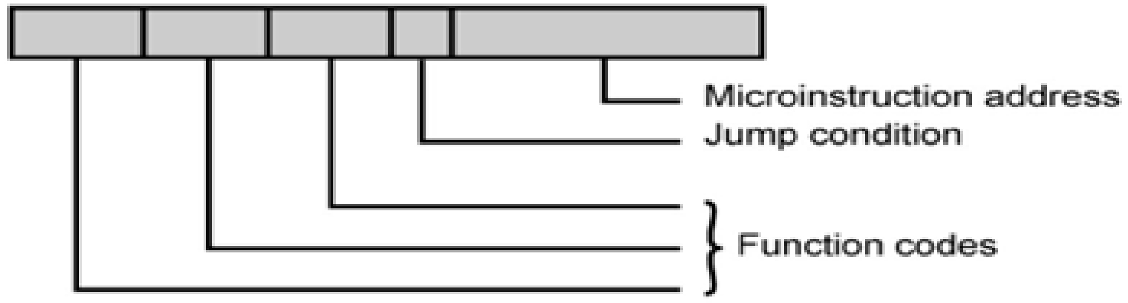
•In the case of a vertical organization, the signals are grouped and encoded in order to reduce the size of the control word.

•Vertical organization, on the other hand, is easier to program, not very different from programming a RISC machine in assembly language, but needs extra level of decoding and may slow the machine down.

Concept of horizontal and vertical microprogramming.



(a) Horizontal microinstruction



(b) Vertical microinstruction

Program Control

•Program control is how a program makes decisions or organizes its activities. Program control typically involves executing particular code based on the outcome of a prior operation or a user input.

•**Program control** is how a **program** makes decisions or organizes its activities. **Program control** typically involves executing particular code based on the outcome of a prior operation or a user input.

•A **program control** instruction changes address value in the **PC** and hence the normal flow of execution.

•Change in **PC** causes a break in the execution of instructions. capability to branch to different **program** segments. Branch (BR) and Jump (JMP) instructions are used sometimes interchangeably but, they are different.

Program Control

Types of Program Control Instructions:

There are different types of Program Control Instructions:

1. **Compare Instruction:**

   Compare instruction is specifically provided, which is similar t a subtract instruction except the result is not stored anywhere, but flags are set according to the result.

Example: CMP R1, R2 ;

2. **Unconditional Branch Instruction:**

It causes an unconditional change of execution sequence to a new location.

Example: JUMP L2 Mov R3, R1 goto L2

3. **Conditional Branch Instruction:**

A conditional branch instruction is used to examine the values stored in the condition code register to determine whether the specific condition exists and to branch if it does.

Program Control

Example: Assembly Code : BE R1, R2, L1
Compiler allocates R1 for x and R2 for y
High Level Code: if (x==y) goto L1;

4. Subroutines:

A subroutine is a program fragment that lives in user space, performs a well-defined task. It is invoked by another user program and returns control to the calling program when finished.

Example: CALL and RET

5. Halting Instructions:

NOP Instruction – NOP is no operation. It cause no change in the processor state other than an advancement of the program counter. It can be used to synchronize timing.

HALT – It brings the processor to an orderly halt, remaining in an idle state until restarted by interrupt, trace, reset or external action.

Program Control

6. Interrupt Instructions:

Interrupt is a mechanism by which an I/O or an instruction can suspend the normal execution of processor and get itself serviced.

RESET – It reset the processor. This may include any or all setting registers to an initial value or setting program counter to standard starting location.

TRAP – It is non-maskable edge and level triggered interrupt. TRAP has the highest priority and vectored interrupt.

INTR – It is level triggered and maskable interrupt. It has the lowest priority. It can be disabled by resetting the processor.

➢Reduced Instruction Set Computer

➢RISC is a microprocessor that is designed to perform a smaller number of types of computer instructions so that it can operate at a higher speed (MIPS).

➢Since each instruction type that a computer must perform requires additional transistors and circuitry.

➢A larger list or set of computer instructions tends to make the microprocessor more complicated and slower in operation.

➢Reduced Instruction Set Computer

Characteristic of RISC –

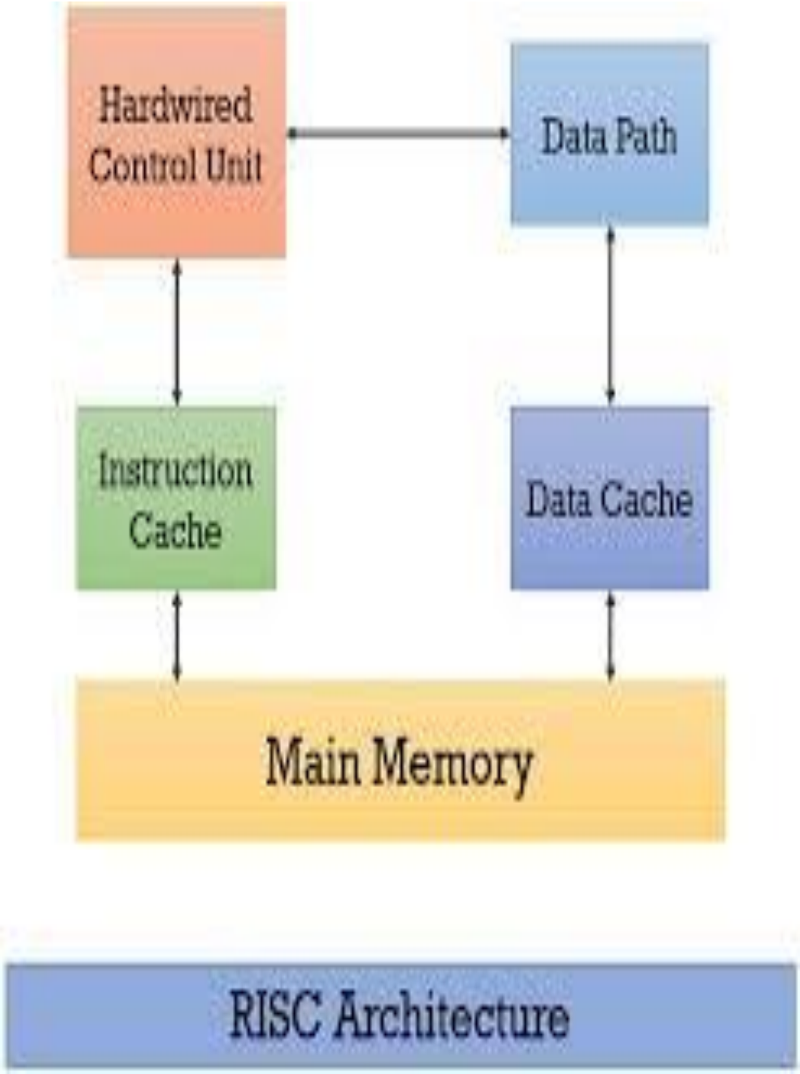
•Simpler instruction, hence simple instruction decoding.

•Instruction come under size of one word.

•Instruction take single clock cycle to get executed.

•More number of general purpose register.

•Simple Addressing Modes.
•Less Data types.
•Pipeline can be achieved.



➢Complex Instruction Set Computing

➢ It is a CPU design plan based on single commands, which are skilled in executing multi-step operations.

➢A complex instruction set computer is a computer where single instructions can perform numerous low-level operations like a load from memory, an arithmetic operation, and a memory store".

➢CISC has the capacity to perform multi-step operations or addressing modes within one instruction set. It is the CPU design where one instruction works several low-level acts.

➢The CISC approach attempts to minimize the number of instructions per program but at the cost of increase in number of cycles per instruction.

$$CPU\ Time = \frac{Seconds}{Program} = \frac{Instructions}{Program} X \frac{Cycles}{Instructions} X \frac{Seconds}{Cycle}$$

Reduced Instruction Set Computer

Characteristic of CISC

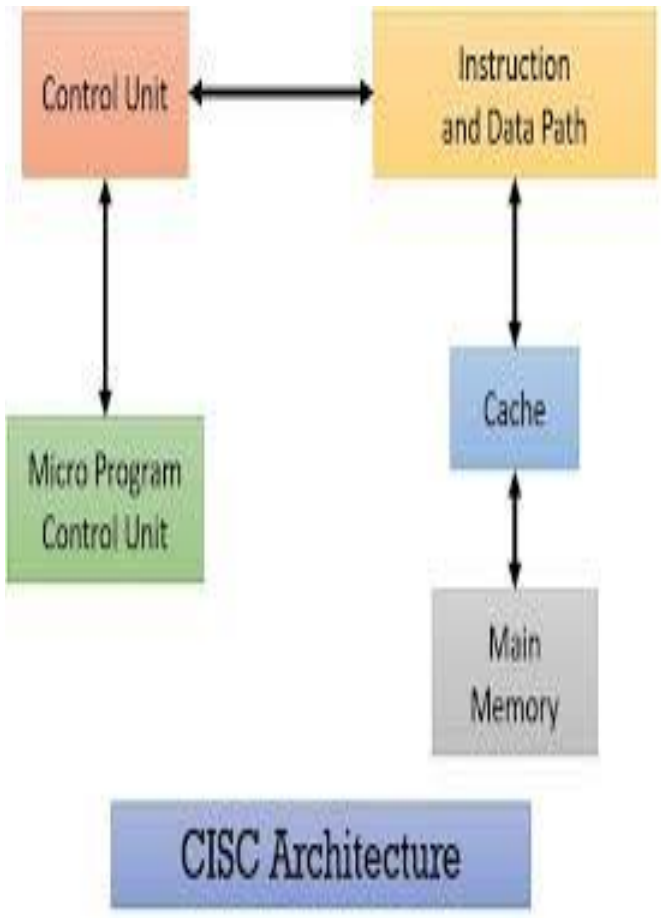
•Complex instruction, hence complex instruction decoding.

•Instruction are larger than one word size.

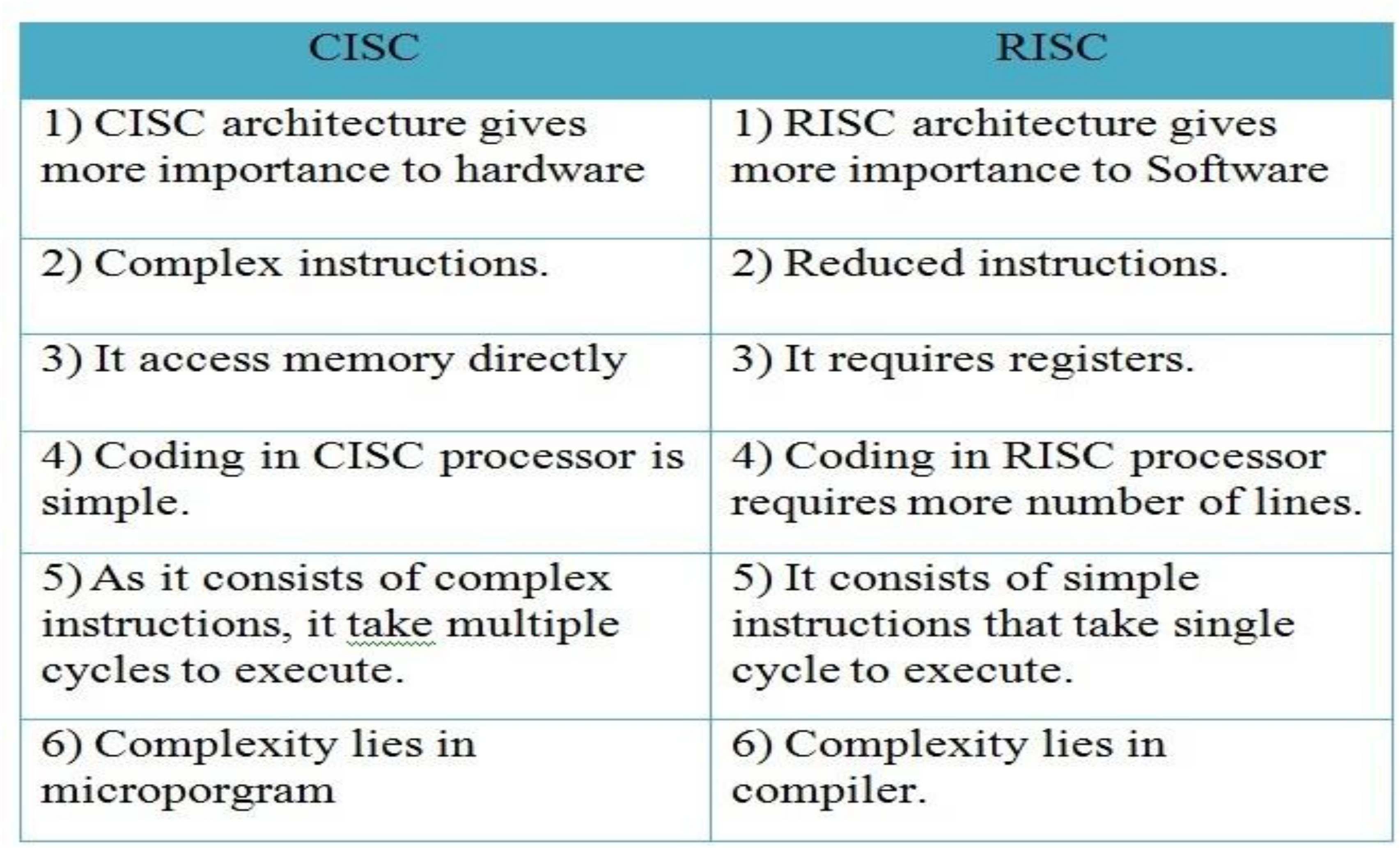
•Instruction may take more than single clock cycle to get executed.

•Less number of general purpose register as operation get performed in memory itself.

•Complex Addressing Modes.
•More Data types.



CISC Vs. RISC



CISC	RISC
1) CISC architecture gives more importance to hardware	1) RISC architecture gives more importance to Software
2) Complex instructions.	2) Reduced instructions.
3) It access memory directly	3) It requires registers.
4) Coding in CISC processor is simple.	4) Coding in RISC processor requires more number of lines.
5) As it consists of complex instructions, it take multiple cycles to execute.	5) It consists of simple instructions that take single cycle to execute.
6) Complexity lies in microporgram	6) Complexity lies in compiler.

Pipelining

- **Pipelining** is a technique where multiple instructions are overlapped during execution.

- **Pipelining** is the process of accumulating instruction from the processor through a **pipeline**.

- It allows storing and executing instructions in an orderly process. It is also known as **pipeline** processing.

- Pipelining is a process of arrangement of hardware elements of the CPU such that its overall performance is increased.

- Simultaneous execution of more than one instruction takes place in a pipelined processor.

Pipelining

Pipeline Stages

RISC processor has 5 stage instruction pipeline to execute all the instructions in the RISC instruction set.

➢**Stage 1 (Instruction Fetch)**

In this stage the CPU reads instructions from the address in the memory whose value is present in the program counter.

➢**Stage 2 (Instruction Decode)**

In this stage, instruction is decoded and the register file is accessed to get the values from the registers used in the instruction.

➢**Stage 3 (Instruction Execute)**

In this stage, ALU operations are performed.

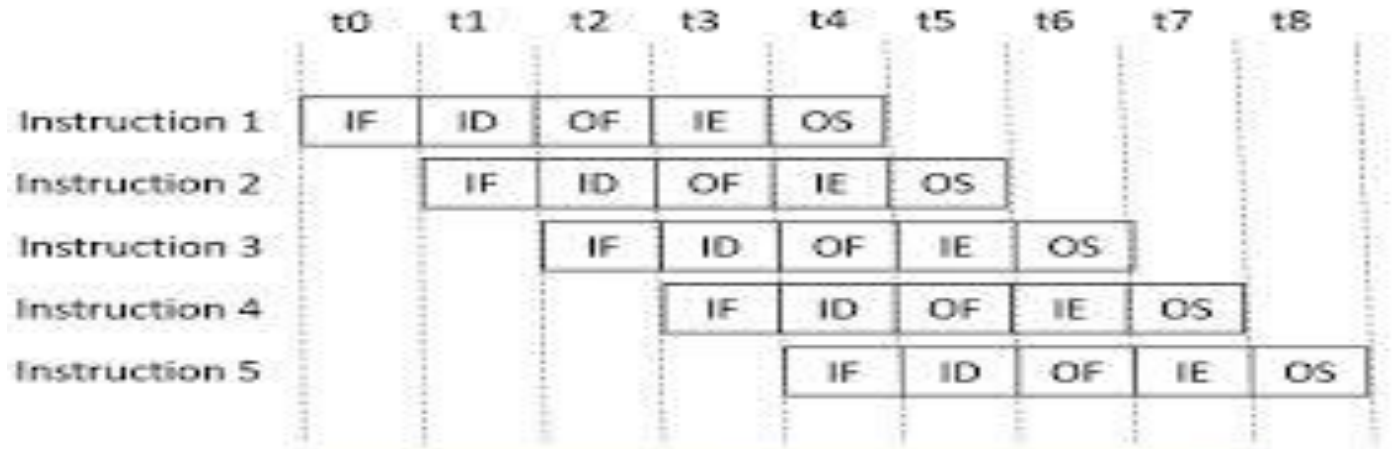
Pipelining

➢**Stage 4 (Memory Access)**

In this stage, memory operands are read and written from/to the memory that is present in the instruction.

➢**Stage 5 (Write Back)**

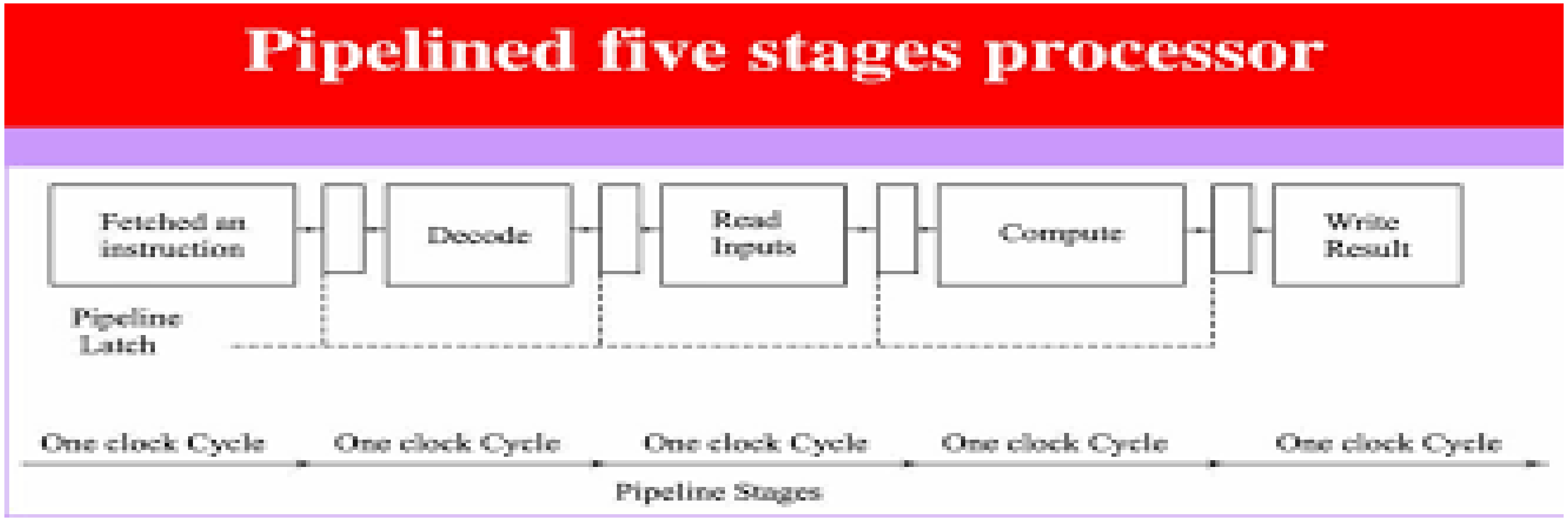
In this stage, computed/fetched value is written back to the register present in the instructions

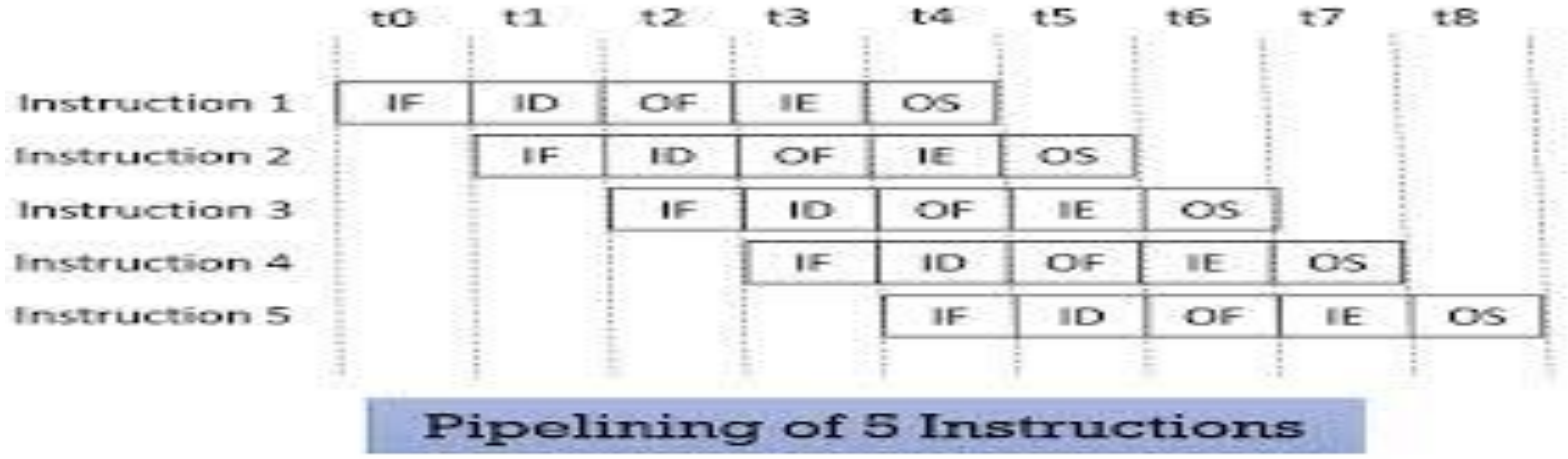


	t0	t1	t2	t3	t4	t5	t6	t7	t8
Instruction 1	IF	ID	OF	IE	OS				
Instruction 2		IF	ID	OF	IE	OS			
Instruction 3			IF	ID	OF	IE	OS		
Instruction 4				IF	ID	OF	IE	OS	
Instruction 5					IF	ID	OF	IE	OS

Pipelining of 5 Instructions

Pipelining





	t0	t1	t2	t3	t4	t5	t6	t7	t8
Instruction 1	IF	ID	OF	IE	OS				
Instruction 2		IF	ID	OF	IE	OS			
Instruction 3			IF	ID	OF	IE	OS		
Instruction 4				IF	ID	OF	IE	OS	
Instruction 5					IF	ID	OF	IE	OS

Pipelining of 5 Instructions

Daily Quiz

- Sketch Instruction cycle and sub cycle
- Write down the main function of control unit
- Perform ashl and ashr for given CSE code 10.
- Define pipelining with example.
- Perform shift micro operation on 1100101 .

Weekly Assignment

- Define micro operation, micro instruction, micro program, microcode.
- List the characteristics of RISC and CISC.
- Differentiate between horizontal and vertical microprogramming.
- Differentiate between hardwired control and micro programmed control. Explain each component of hardwired control unit organization.
- Explain phases of Instruction cycle.
- What is micro programmed control unit? Explain the basic structure of micro programmed control unit
- Explain the working of microprogram sequencer with neat diagram.

Faculty Video Links, You tube Courses Details

You tube/other Video Links

- https://www.youtube.com/watch?v=vcvgvqnH7GA
- https://www.youtube.com/watch?v=U62iP8RkZIk
- https://www.youtube.com/watch?v=8b1Cs1Uf6hI
- https://www.youtube.com/watch?v=MSac_s-W0pc
- https://www.youtube.com/watch?v=sJdCD_APVq8
- https://www.youtube.com/watch?v=_EKgwOAAWZA

MCQ

- 1. 1. The decoded instruction is stored in ______

  a) IR   b) PC   c) Registers   d) MDR

  2. Which registers can interact with the secondary storage?

  a) MAR   b) PC   c) IR   d) R0

  3. During the execution of a program which gets initialized first ?

  a) MDR   b) IR   c) PC   d) MAR

  4. _______ is used to store data in registers.

  a) D flip flop   b) JK flip flop   c) RS flip flop d) None of the mentioned

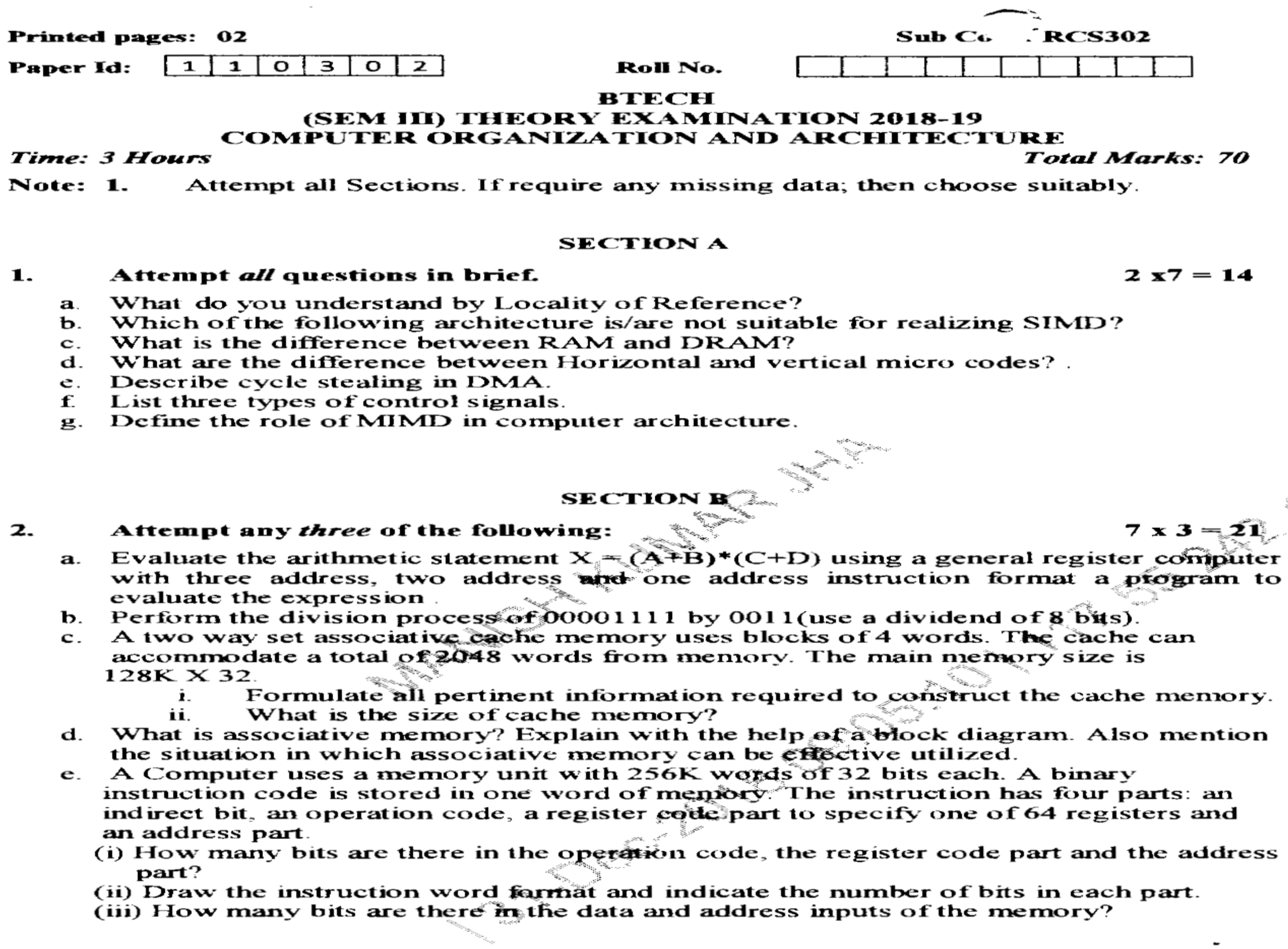
  5. Write the phases of Instruction------------------------------

  6 Perform cil and cir on given data 1001.

  7. Write full form of RISC and CISC--

Solution **1 a. 2a. 3 c. 4** a

Old Question Papers



Printed pages: 02 Sub Code: RCS302

Paper Id: | 1 | 1 | 0 | 3 | 0 | 2 | Roll No. | | | | | | | | | | |

BTECH

(SEM III) THEORY EXAMINATION 2018-19

COMPUTER ORGANIZATION AND ARCHITECTURE

Time: 3 Hours *Total Marks: 70*

Note: 1. Attempt all Sections. If require any missing data; then choose suitably.

SECTION A

1. Attempt *all* questions in brief. **2 x7 = 14**

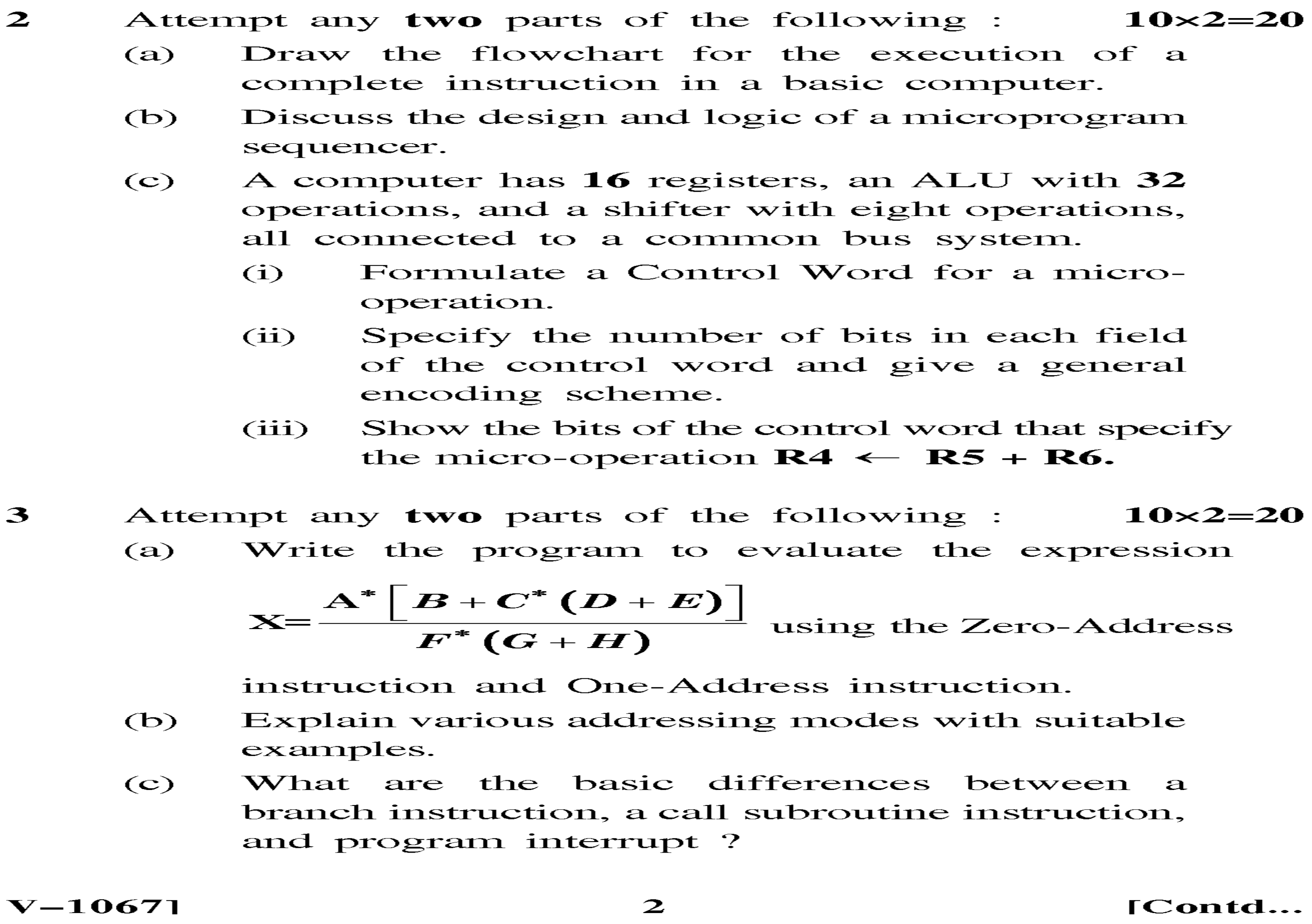
a. What do you understand by Locality of Reference?
b. Which of the following architecture is/are not suitable for realizing SIMD?
c. What is the difference between RAM and DRAM?
d. What are the difference between Horizontal and vertical micro codes? .
e. Describe cycle stealing in DMA.
f. List three types of control signals.
g. Define the role of MIMD in computer architecture.

SECTION B

2. Attempt any *three* of the following: **7 x 3 = 21**

a. Evaluate the arithmetic statement X = (A+B)*(C+D) using a general register computer with three address, two address and one address instruction format a program to evaluate the expression .
b. Perform the division process of 00001111 by 0011(use a dividend of 8 bits).
c. A two way set associative cache memory uses blocks of 4 words. The cache can accommodate a total of 2048 words from memory. The main memory size is 128K X 32.
   i. Formulate all pertinent information required to construct the cache memory.
   ii. What is the size of cache memory?
d. What is associative memory? Explain with the help of a block diagram. Also mention the situation in which associative memory can be effective utilized.
e. A Computer uses a memory unit with 256K words of 32 bits each. A binary instruction code is stored in one word of memory. The instruction has four parts: an indirect bit, an operation code, a register code part to specify one of 64 registers and an address part.
   (i) How many bits are there in the operation code, the register code part and the address part?
   (ii) Draw the instruction word format and indicate the number of bits in each part.
   (iii) How many bits are there in the data and address inputs of the memory?

Old Question Papers



2 Attempt any **two** parts of the following : **10×2=20**

(a) Draw the flowchart for the execution of a complete instruction in a basic computer.

(b) Discuss the design and logic of a microprogram sequencer.

(c) A computer has **16** registers, an ALU with **32** operations, and a shifter with eight operations, all connected to a common bus system.

(i) Formulate a Control Word for a micro-operation.

(ii) Specify the number of bits in each field of the control word and give a general encoding scheme.

(iii) Show the bits of the control word that specify the micro-operation **R4 ← R5 + R6.**

3 Attempt any **two** parts of the following : **10×2=20**

(a) Write the program to evaluate the expression

$$X=\frac{A^*\left[B+C^*(D+E)\right]}{F^*(G+H)}$$ using the Zero-Address instruction and One-Address instruction.

(b) Explain various addressing modes with suitable examples.

(c) What are the basic differences between a branch instruction, a call subroutine instruction, and program interrupt ?

V–10671] **2** **[Contd...**

Expected Questions for University Exam

- Write a program to evaluate the arithmetic expression by using Three, Two, One and Zero address instruction. **X = (A+B*C) / (D+E*F/G+H).**
- Differentiate between RISC & CISC based microprocessor.
- What is micro programmed control unit? Give the basic structure of micro programmed control unit.
- How pipeline performance can be measured? Discuss. Give a space time diagram for visualizing the pipeline behavior for a four stage pipeline.
- Perform Shift micro operation for given data 11001010.

Summary

In previous slides we discuss in details

Control Unit:

- ➢Instruction types, formats
- ➢Instruction cycles and sub cycles (fetch and execute etc)
- ➢Micro operations
- ➢Execution of a complete instruction.
- ➢Program Control
- ➢Reduced Instruction Set Computer
- ➢Pipelining
- ➢Hardwire and micro programmed control
- ➢Micro programmed sequencing
- ➢Concept of horizontal and vertical microprogramming.