UNIT-6: Processing Unit & Micro-programmed control

JNTUK R16 2-2 CSE

Syllabus:

→ Processing Unit
  → Fundamental Concepts
    → Register Transfers ✓
    → Performing an Arithmetic (or) Logic Operation
    → Fetching a word from memory
  → Execution of a Complete Instruction ✓
  → Hardwired Control ✓
→ Micro-programmed Control
  → Micro Instructions
  → Microprogram Sequencing ✓
  → Wide-branch Addressing
  → MicroInstructions with Next-Address Field ✓

① Fundamental Concepts:

→ To execute a program, the processor fetches one instruction at a time and performs the operations specified.

→ Instructions are fetched from successive memory locations until a branch (or) Jump instruction is encountered.

→ The processor keeps track of the address of the memory locations containing the next instruction to be fetched using the program counter, PC.

→ Another key register in the processor is the Instruction Register, IR.

→ Suppose that each instruction comprises 4 bytes, and that it is stored in one memory word.

→ To execute an instruction, the processor has to perform the following three steps.

* Fetch the contents of the memory location pointed to by the PC. They are loaded into the IR.

$$IR \leftarrow [[PC]]$$

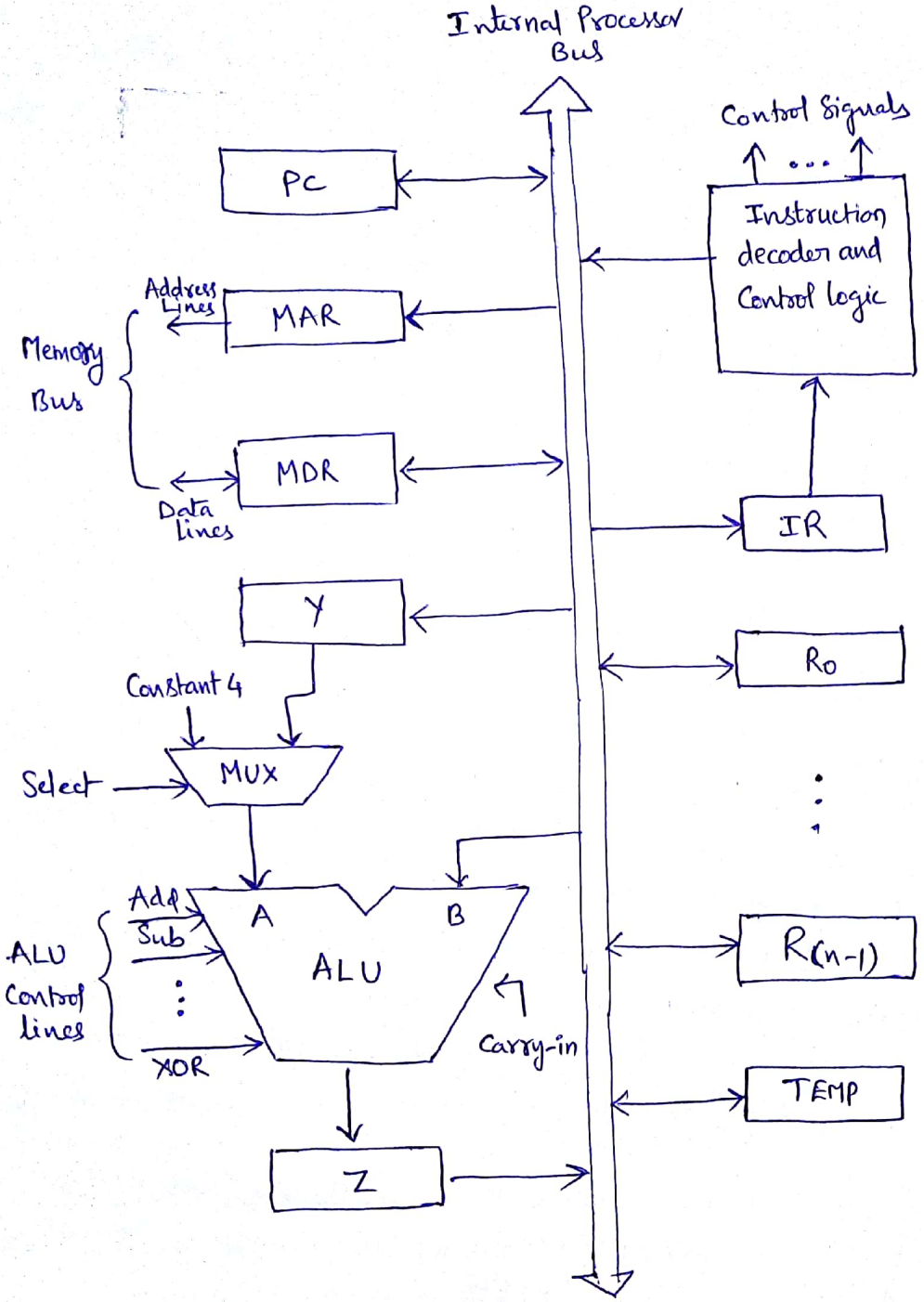
* Assuming that the memory is byte addressable, increment the contents of the PC by 4

$$PC \leftarrow [PC] + 4$$

* Carry out the actions specified by the instruction in the IR.

→ Here, First two steps represents fetch phase, Third step represents execution phase.

→ Single-bus organization of the datapath inside a processor is depicted as,



→ Data may be loaded into MDR either from the memory bus (or) from the internal processor bus.

→ The input of MAR is connected to the internal bus and its output is connected to the external bus.

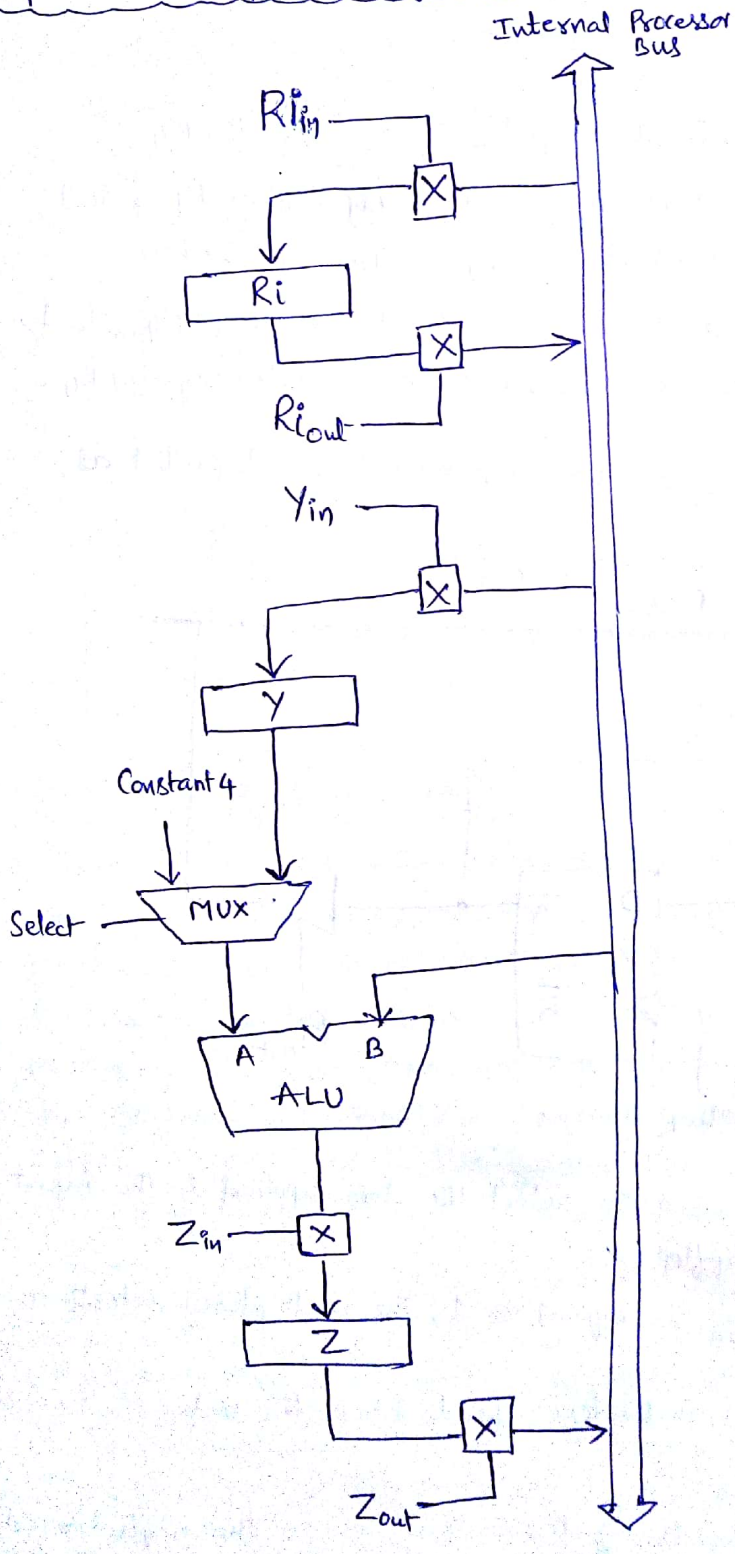
→ The multiplexer MUX selects either the output of register Y (or) a constant value 4 to be provided as input A of the ALU.

(a) Register Transfer :

→ Instruction execution involves a sequence of steps in which data are transferred from one register to another.

→ For each register, two control signals are used to place the contents of that register on the bus (or) load the data on the bus into the register.

→ Input and output gating for the registers is depicted as,



→ The input and output of register R_i are connected to the bus via switches controlled by the signals Ri_{in} and Ri_{out}, respectively.

→ When Ri_{in} is set to 1, the data on the bus are loaded into R_i

→ Similarly, when Ri_{out} is set to 1, the contents of Register R_i are placed on the bus.

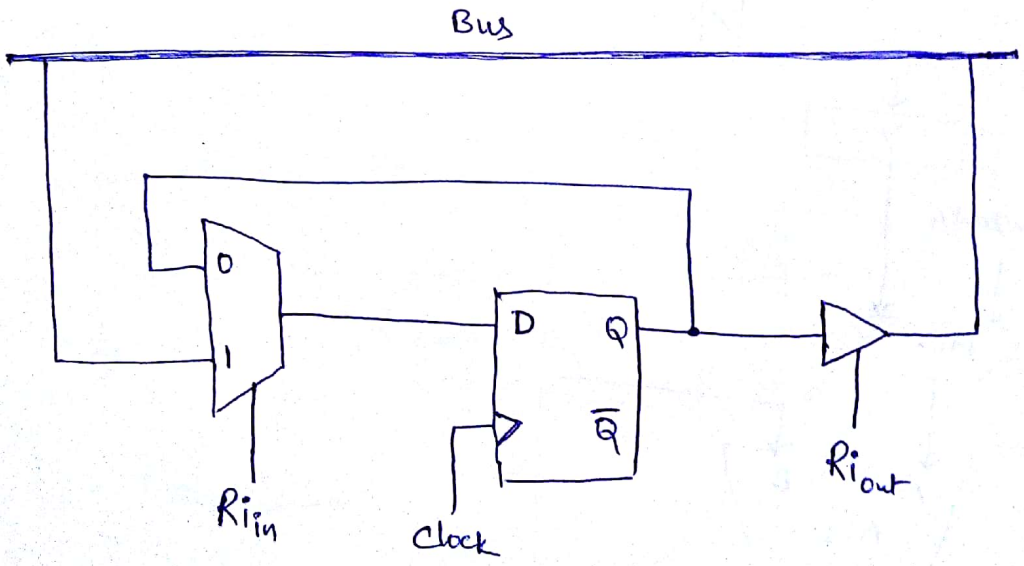
→ While Ri_{out} is equal to 0, the bus can be used for transferring data from other registers

Example:

→ To transfer the contents of register R_1 to register R_4,

- Enable the output of register R_1 by setting $R1_{out}$ to 1, This places the contents of R_1 on the processor bus.
- Enable the input of register R_4 by setting $R4_{in}$ to 1, This loads data from the processor bus into register R_4.

→ Input and Output gating for one register bit is depicted as,



→ A two-input multiplexer is used to select the data applied to the input of an edge-triggered D-flipflop

→ When the control input Ri_{in} is equal to 1, the multiplexer selects the data on the bus.

→ When Ri_{in} equal to 0, the multiplexer feeds back the value currently stored in the flip-flop.

→ When Ri_{out} is equal to 0, the gate's output is in the high-impedance (electrically disconnected) state.

→ When $Ri_{out} = 1$, the gate drives the bus to 0 (or) 1, depending on the value of Q.

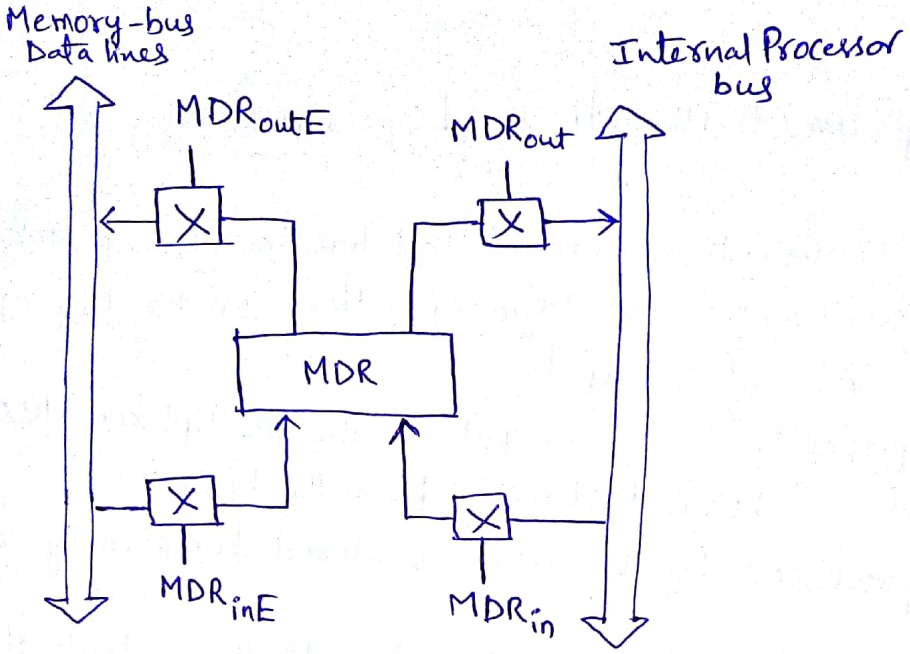
(b) Performing an Arithmetic (or) Logic Operation:

→ The ALU is a combinational circuit that has no internal storage.
→ It performs arithmetic and Logic operations on the two operands applied to its A and B inputs
→ One of the operands is the output of the multiplexer MUX and the other operand is obtained directly from the bus.
→ The result produced by the ALU is stored temporarily in register Z.
→ Therefore, a sequence of operations to add the contents of register R1 to those of register R2 and store the result in register R3 is,

1. $R1_{out}$, Y_{in}
2. $R2_{out}$, Select Y, Add, Z_{in}
3. Z_{out}, $R3_{in}$

(c) Fetching a word from memory:

→ To fetch a word of information from memory, the processor has to specify the address of the memory location where this information is stored and request a Read operation.
→ This applies whether the information to be fetched represents an instruction in a program (or) an operand specified by an instruction.
→ The processor transfers the required address to the MAR, whose output is connected to the address lines of the memory bus.
→ When the requested data are received from the memory they are stored in register MDR, from where they can be transferred to other registers in the processor.
→ The connection and control signals for register MDR is depicted as,



→ It has 4 control signals
- MDR_{in} and MDR_{out} control the connection to the internal bus.
- MDR_{inE} and MDR_{outE} control the connection to the external bus

→ As an example of a read operation, consider the instruction

MOVE (R1), R2

- The actions needed to execute this instruction are,

1. MAR ← [R1]
2. Start a Read operation on the memory bus
3. Wait for the MFC response from the memory
4. Load MDR from the memory bus
5. R2 ← [MDR]

(2) Execution of a Complete Instruction:

→ Consider the instruction

Add (R3), R1

which adds the contents of a memory location pointed to by R3 to register R1.

→ Executing this instruction requires the following actions.

- Fetch the instruction
- Fetch the first operand (the contents of the memory location pointed to by R3)
- Perform the addition
- Load the result into R1.

→ The Control sequence for execution of the instruction Add (R3), R1 is given as

Step	Action
1	PC_{out}, MAR_{in}, Read, Select4, Add, Z_{in}
2	Z_{out}, PC_{in}, Y_{in}, WMFC
3	MDR_{out}, IR_{in}
4	$R3_{out}$, MAR_{in}, Read
5	$R1_{out}$, Y_{in}, WMFC
6	MDR_{out}, SelectY, Add, Z_{in}
7	Z_{out}, $R1_{in}$, End

→ In Step1, the instruction fetch operation is initiated by loading the contents of the PC into the MAR and sending a Read request to the memory

→ The Select signal is set to Select4, which causes the multiplexer MUX to select the constant 4.

→ This value is added to the operand at input B, which is the

Contents of the PC, and the result is stored in register Z.

→ The updated value is moved from register Z back into the PC during Step 2, while waiting for the memory to respond.

→ In Step 3, the word fetched from the memory is loaded into the IR.

→ From Step 1 through 3 represents instruction fetch phase.

→ The contents of register R3 are transferred to the MAR in Step 4, and a memory read operation is initiated.

→ Then the contents of R1 are transferred to register Y in Step 5, to prepare the addition operation.

→ When the Read operation is completed, the memory operand is available in register MDR, and the addition operation is performed in Step 6.

→ The sum is stored in register Z, then transferred to R1 in Step 7.

→ From step 4 through 7 represents execution phase.

(i) BRANCH Instructions:

→ A branch instruction replaces the contents of the PC with the branch target Address.

→ This address is usually obtained by adding an offset X, which is given in the branch instruction, to the updated value of the PC

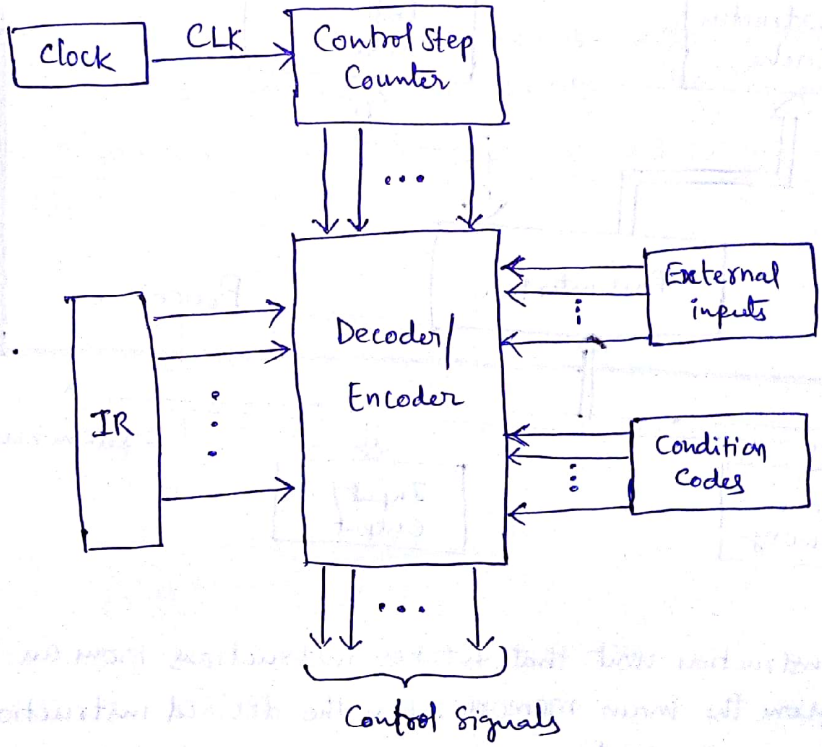
→ Control sequence for an unconditional Branch instruction is given as

Step	Action
1	PC_{out}, MAR_{in}, Read, Select4, Add, Z_{in}
2	Z_{out}, PC_{in}, Y_{in}, WMFC
3	MDR_{out}, IR_{in}
4	Offset-field-of-IR_{out}, Add, Z_{in}
5	Z_{out}, PC_{in}, End

③ Hardwired Control :

→ To execute instructions, the processor must have some means of generating the control signals needed in the proper sequence.
→ This approach has two Categories
  * Hardwired Control
  * Micro-programmed control

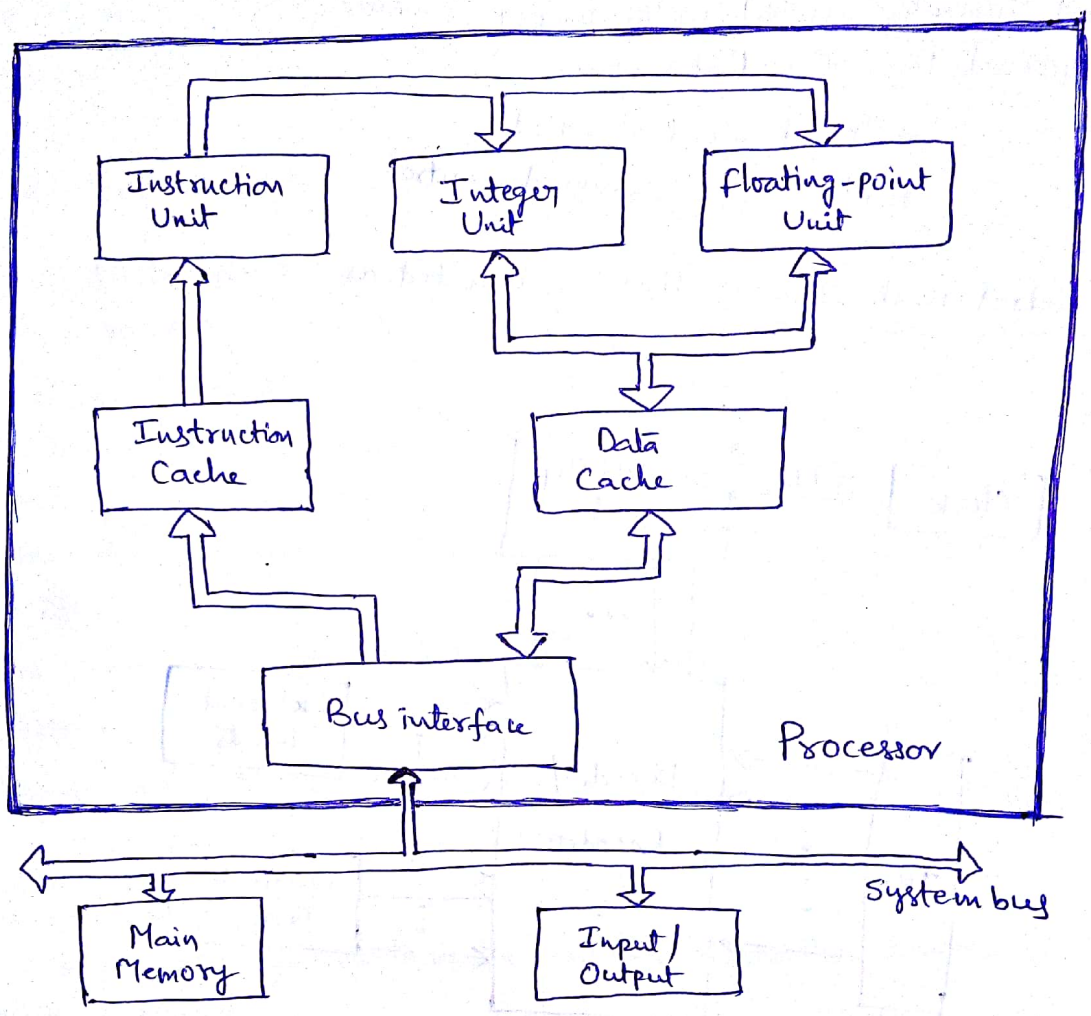
→ The control unit organization is depicted as



→ Consider the sequence of control signals.
→ Each step in this sequence is completed in one clock period.
→ A counter may be used to keep track of the control steps.
→ Each State, (or) count, of this counter corresponds to one control step.
→ The required control signals are determined by the following information.
  - Contents of the Control Step Counter
  - Contents of the instruction register
  - Contents of the condition code flags
  - External input signals, such as MFC and interrupt requests.

(i) A Complete Processor :

→ The Block diagram of a Complete Processor is depicted as



→ It has an instruction unit that fetches instructions from an Instruction Cache (or) from the main memory when the desired instructions are not already in the Cache.

→ It has seperate processing units to deal with integer data and floating-point data.

→ Using seperate caches for instruction and data is common practice in many processors today.

→ The processor is connected to the System bus and, hence, to the rest of the Computer, by means of a bus interface.

④ Microprogrammed Control:

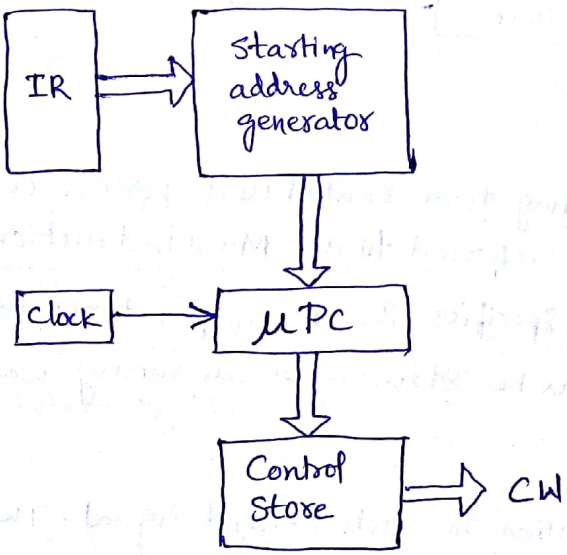
→ The control signals are generated by a program similar to machine language programs represents a scheme called Microprogrammed Control

→ A control word (CW) is a word whose individual bits represent the various control signals

→ A sequence of CWs corresponding to the control sequence of a machine instruction constitutes the microroutine for that instruction, and the individual control words in this microroutine are referred to as microinstructions.

→ The microroutines for all instructions in the instruction set of a computer are stored in a special memory called the control store.

→ The Basic organization of a microprogrammed control unit is depicted as,



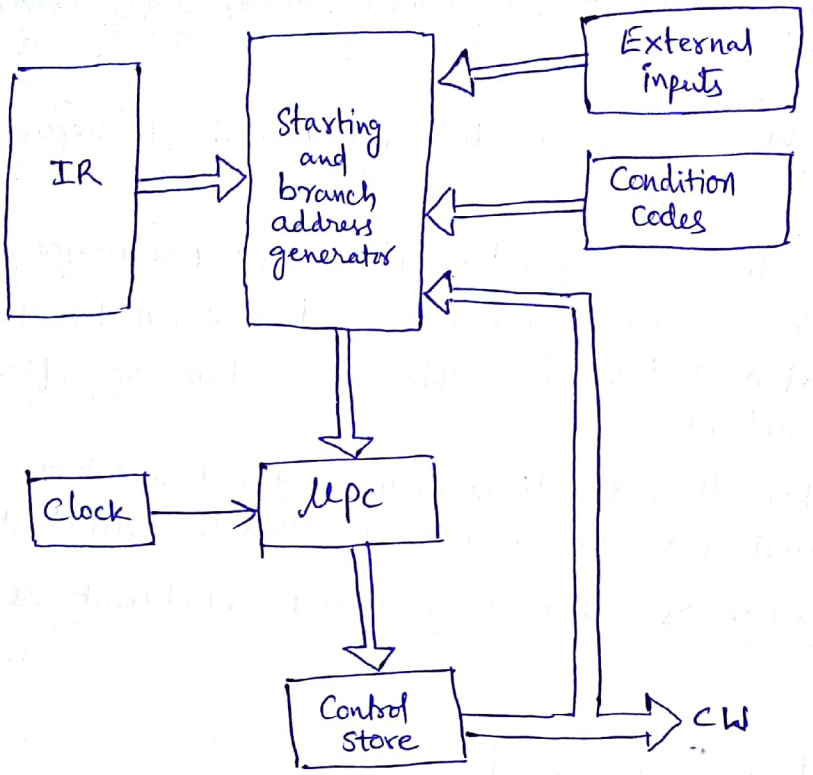
→ To read the control words sequentially from the control store, a micro-program counter (μPC) is used.

→ Everytime a new instruction is loaded into the IR, the output of the block labeled "Starting Address Generator" is loaded into the μPC.

→ The μPC is then automatically incremented by the clock, causing successive microinstructions to be read from the control store.

→ Hence, the control signals are delivered to various parts of the processor in the correct sequence.

→ Organization of the control unit to allow conditional branching in the microprogram is depicted as



(i) Microinstructions :

→ The control word belonging to a control unit possess certain instructions, usually referred to as Microinstructions.

→ Each Microinstruction specifies the Microoperations for the system

→ A microinstruction can be structured in many ways. Some of them are,

- Assign a bit position to each control signal. This scheme is not good enough, as it results in long instructions. And also because of only few bits are set to 1, it does not use the bit space properly
- Encode the microinstructions using bits, with assumptions.
- Group the mutually exclusive control signals into fields.
- Enumerate the patterns of required signals in all microinstructions.

→ There are two types of microinstruction formats.

(a) Horizontal Microinstruction Format
(b) Vertical Microinstruction Format

a) Horizontal Microinstruction format:

Internal CPU Control Signals	System bus Control Signals	Jump Condition (Indirect bit, Zero overflow, Unconditional)	Microinstruction address

→ Horizontal Microinstruction format supports
- Long formats
- express (or) resort to high degree of parallelism
- Low degree of encoding of control information

b) Vertical Microinstruction format:

Function codes	Function Codes	Jump Condition	Microinstruction Address

→ Vertical Microinstruction Format supports
- Less degree of parallelism in case of microoperations.
- Subsequently high encoding in case of control information.
- Relatively short format

(ii) Microprogram Sequencing

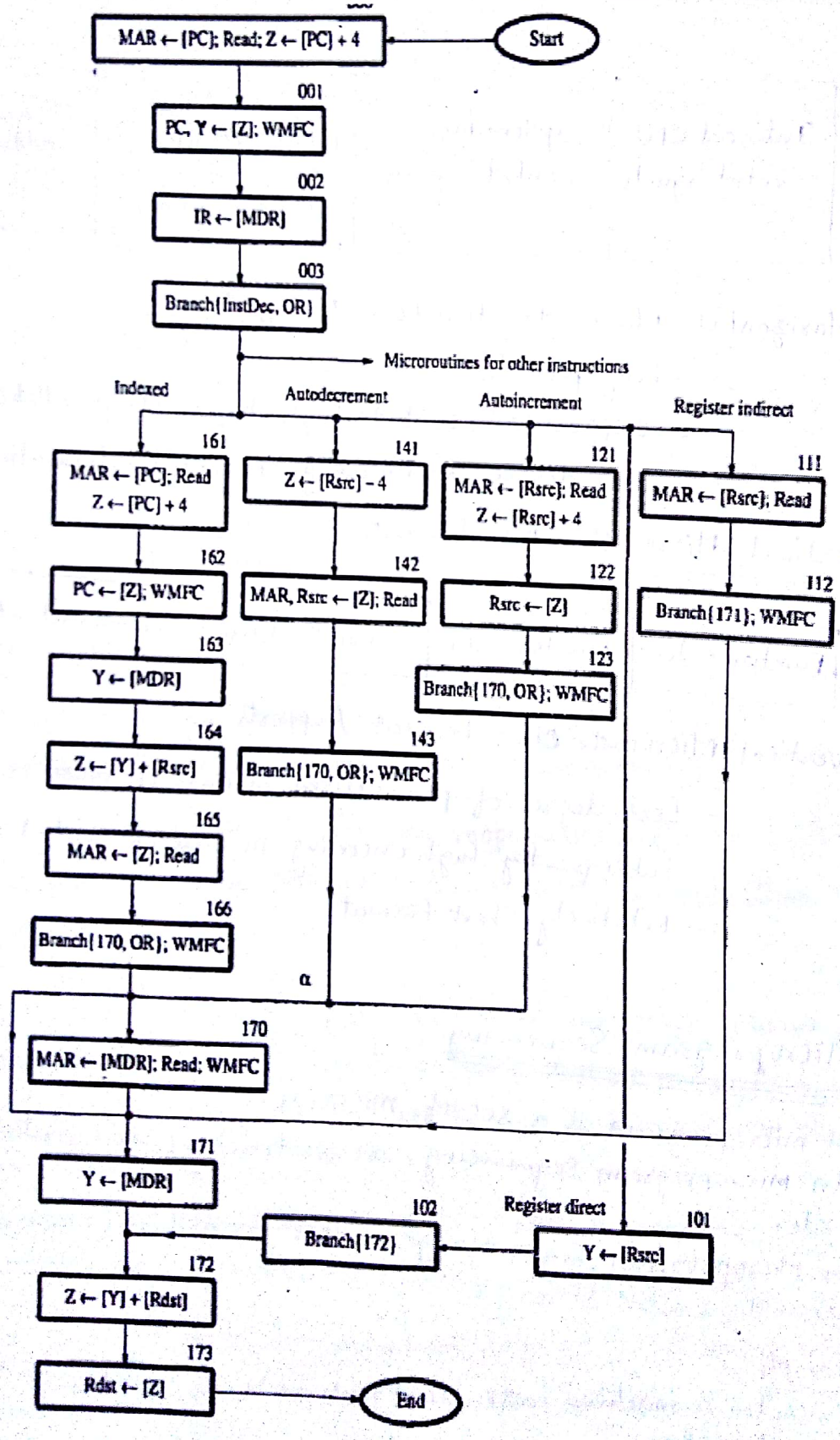
→ A microprogram is a set of microinstructions
→ In microprogram sequencing, microinstructions are executed in the sequential order.
→ The Microprogram Sequencer generates the order of executing microinstructions from the Control Store.

Example:
→ Consider a machine instruction that adds R_s and R_d and stores the result in R_d

Add R_s, R_d

→ Where, R_s is the source operand register and R_d is the destination operand register

→ Flowchart of a microprogram for the Add Rsrc, Rdst instruction is depicted as,



(iii) Wide-Branch Addressing:

→ Generating branch addresses means that the circuitry becomes more Complex

→ Example, the machine instruction fetch is completed, and an appropriate microroutine should be selected according to addressing modes

→ Here, the OPcode of a machine instruction is translated into a starting address.

→ It is possible to issue a wait for MFC (WMFC) Command in a branch micro-instruction.

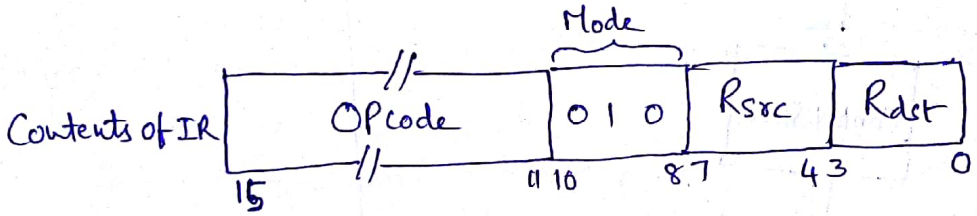
→ The WMFC signal means that the microinstruction may take several clock cycles to complete.

Example:

→ For the instruction

Add (Rsrc)+, Rdst

→ The Format of IR is depicted as



(iv) Microinstructions with Next-Address Field:

→ The purpose of branch microinstructions is to find the address of the next microinstruction to be fetched.

→ That means, they do not perform any useful operation in the data path.

→ This effects the operating speed of the System.

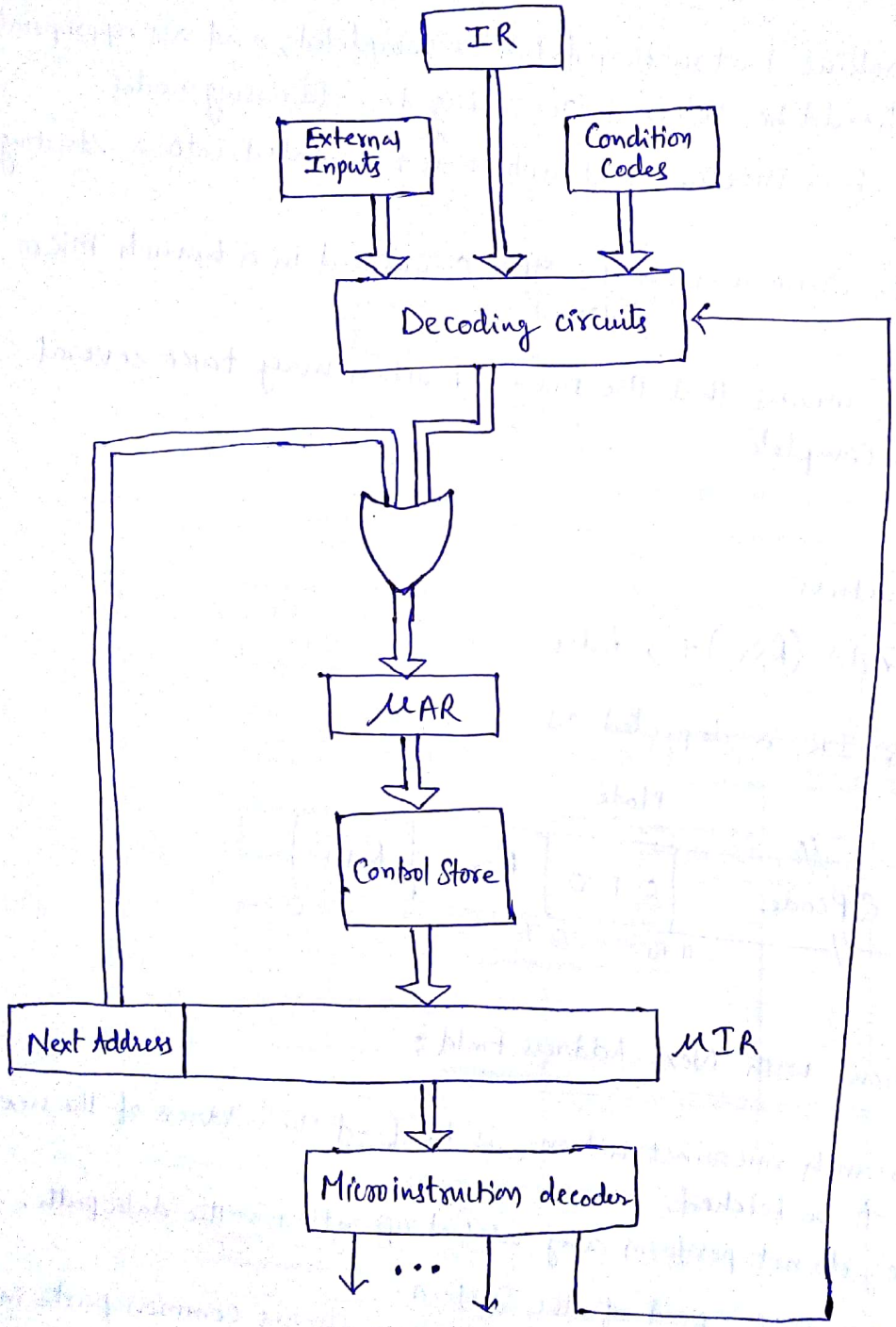
→ The situation becomes worse when the processor shares common parts in microinstructions using branch microinstruction in order to execute sequential instructions.

→ Thus, execution of one simple instruction needs several branch instructions, which badly affects the system's Operating System (OS).

→ One way to overcome the above problem is to modify the sequencing technique based on incrementable µpc.

→ An alternative is to include a special address field called "Next-address field" in each microinstruction in order to specify the address of the next microinstruction

→ The Microinstruction-Sequencing Organization is depicted as,



→ As a result, each microinstruction generates the effect of a branch microinstruction and also performs its intended function.

→ So, there is no need for a seperate μPC to store the address of next instruction.

CO - UNIT 6 - Short Answer Questions

(1) What is register transfer? Discuss..

page 6.3

(2) Briefly discuss unconditional branch instructions.

page 6.8

(3) Explain Hardwired Control

page 6.9

(4) Explain Microprogrammed Control unit

page 6.11

(5) Define a) Micro-operation b) Micro-program c) Micro-instruction

a) Microoperation:

→ A Microoperation refers to a simple (or) a complex operation that produces certain output, which is stored in a memory location (or) register.

b) Microprogram:

→ Microprogram refers to a sequence of microinstructions.
→ It is usually stored in control memory.
→ A Microprogram is executed by a micro programmed Control unit.

c) Micro Instruction:

→ The control word belonging to a control unit possess certain instructions, usually referred to as microinstructions.
→ Each Microinstruction specifies microoperations for the system.

(6) Write Micro instruction formats.

(or)

Write about a) Horizontal Microinstruction Format b) vertical Microinstruction Format

page 6.13

(7) Differentiate between Hardwired Control and Microprogrammed Control.

Microprogrammed control Unit	Hardwired control unit
1. It has slower execution speed	1. It has faster execution speed
2. Its control functions are implemented in software	2. Its control functions are implemented in hardware
3. It can easily accommodate changes such as new system specifications (or) new instructions redesign	3. It is not flexible towards any changes
4. Its design process is systematic	4. Its design process is complicated
5. It usually supports more than 100 instructions.	5. It supports less than 100 instructions
6. It can easily support operating systems and diagnostic features	6. It cannot easily support OS and diagnostic features
7. It can easily handle Large/ Complex instruction sets	7. It cannot easily handle Large/ Complex instruction sets.
8. It is used in Mainframes and Microprocessors	8. It is used in RISC microprocessors