Prepared by Dr K Badari Nath, Associate Professor,
Computer Science & Engineering, RV College Of Engineering

FCSD – Foundations Of Computer System Design

Unit 5 : Basic Processing Unit

Fundamental Concepts, Instruction Execution, Hardware Components, Instruction Fetch and Execution Steps, Multiple-Bus Organization, Control Signals, Hardwired Control, Basic organization of a Microprogrammed Control Unit.

- CIE/SEE Question solutions are included

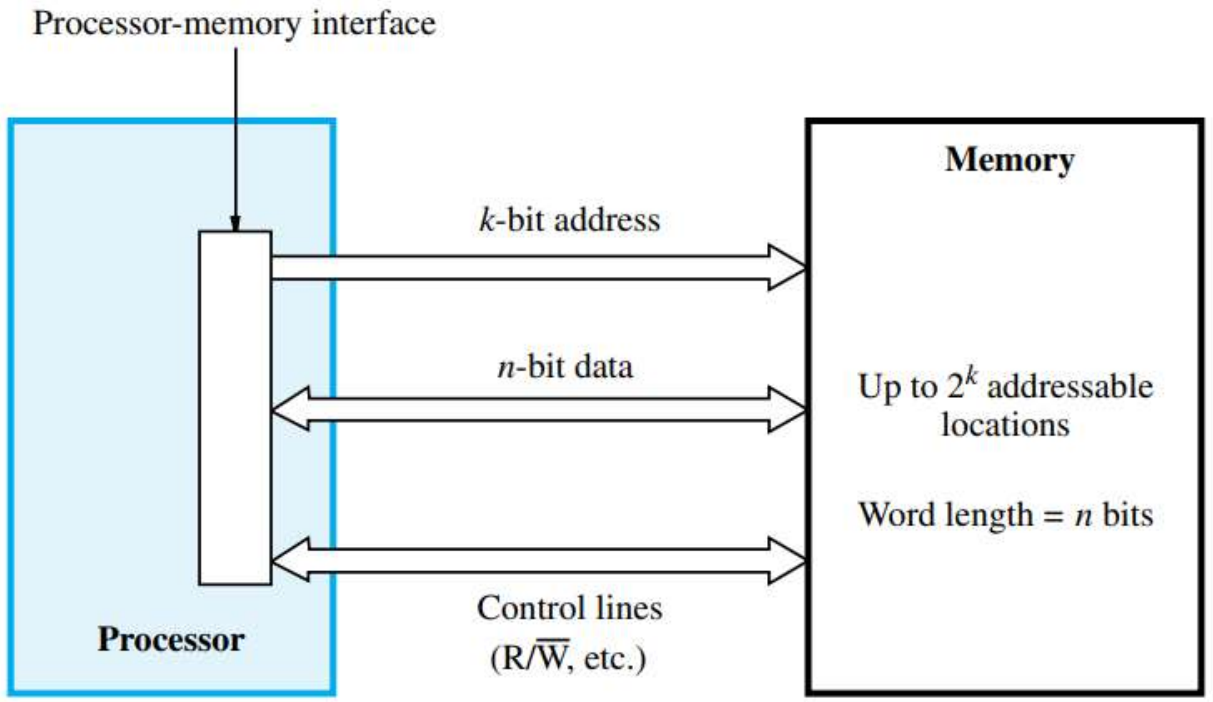
- Reference: Chapter – **Basic Processing Unit**
  (Book:"Computer Organization and Embedded Systems" by Carl Hamacher)

*https://www.youtube.com/@drkbadarinath4636
(for lecture videos)

In this presentation we focus on the processing unit, which executes machine-language instructions and coordinates the activities of other units in a computer. We examine its internal structure and show how it performs the tasks of fetching, decoding, and executing such instructions.

Processor & Memory Interface

The maximum size of the memory that can be used in any computer is determined by the addressing scheme. For example, a computer that generates 16-bit addresses is capable of addressing up to 2power16 = 64K (kilo) memory locations. Machines whose instructions generate 32-bit addresses can utilize a memory that contains up to 2power32 = 4G (giga) locations, whereas machines with 64-bit addresses can access up to 2power64 = 16E (exa) ≈ 16 × 1018 locations. The number of locations represents the size of the address space of the computer.



Connection of the memory to the processor.

The connection between the processor and its memory consists of address, data, and control lines. The processor uses the address lines to specify the memory location involved in a data transfer operation, and uses the data lines to transfer the data. At the same time, the control lines carry the command indicating a Read or a Write operation and whether a byte or a word is to be transferred. The control lines also provide the necessary timing information and are used by the memory to indicate when it has completed the requested operation.

Fundamental Concepts

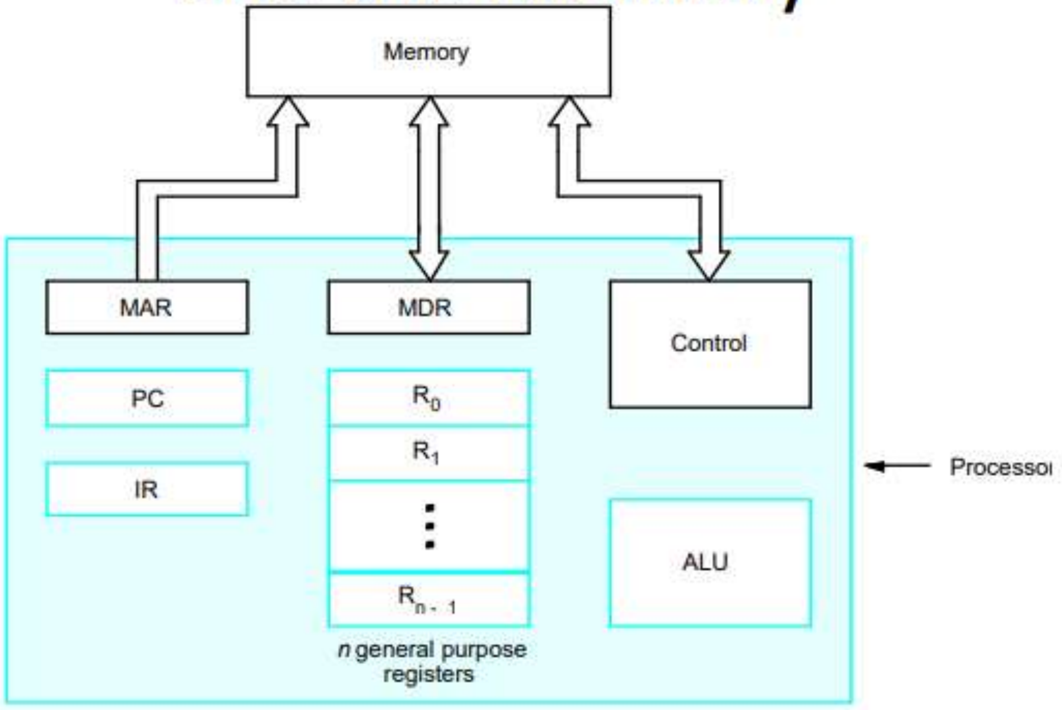
The processor uses the program counter, PC, to keep track of the address of the next instruction to be fetched and executed. After fetching an instruction, the contents of the PC are updated to point to the next instruction in sequence.
When an instruction is fetched, it is placed in the instruction register, IR, from where it is interpreted, or decoded, by the processor's control circuitry.

Consider a 32-bit computer in which each instruction is contained in one word in the memory, as in RISC-style instruction set architecture. To execute an instruction, the processor has to perform the following steps

1. Fetch the contents of the memory location pointed to by the PC. The contents of this location are the instruction to be executed; hence they are loaded into the IR. In register transfer notation, the required action is
   IR $\leftarrow$ [[PC]]
2. Increment the PC to point to the next instruction. Assuming that the memory is byte addressable, the PC is incremented by 4; that is
   PC $\leftarrow$ [PC] + 4
3. Carry out the operation specified by the instruction in the IR.

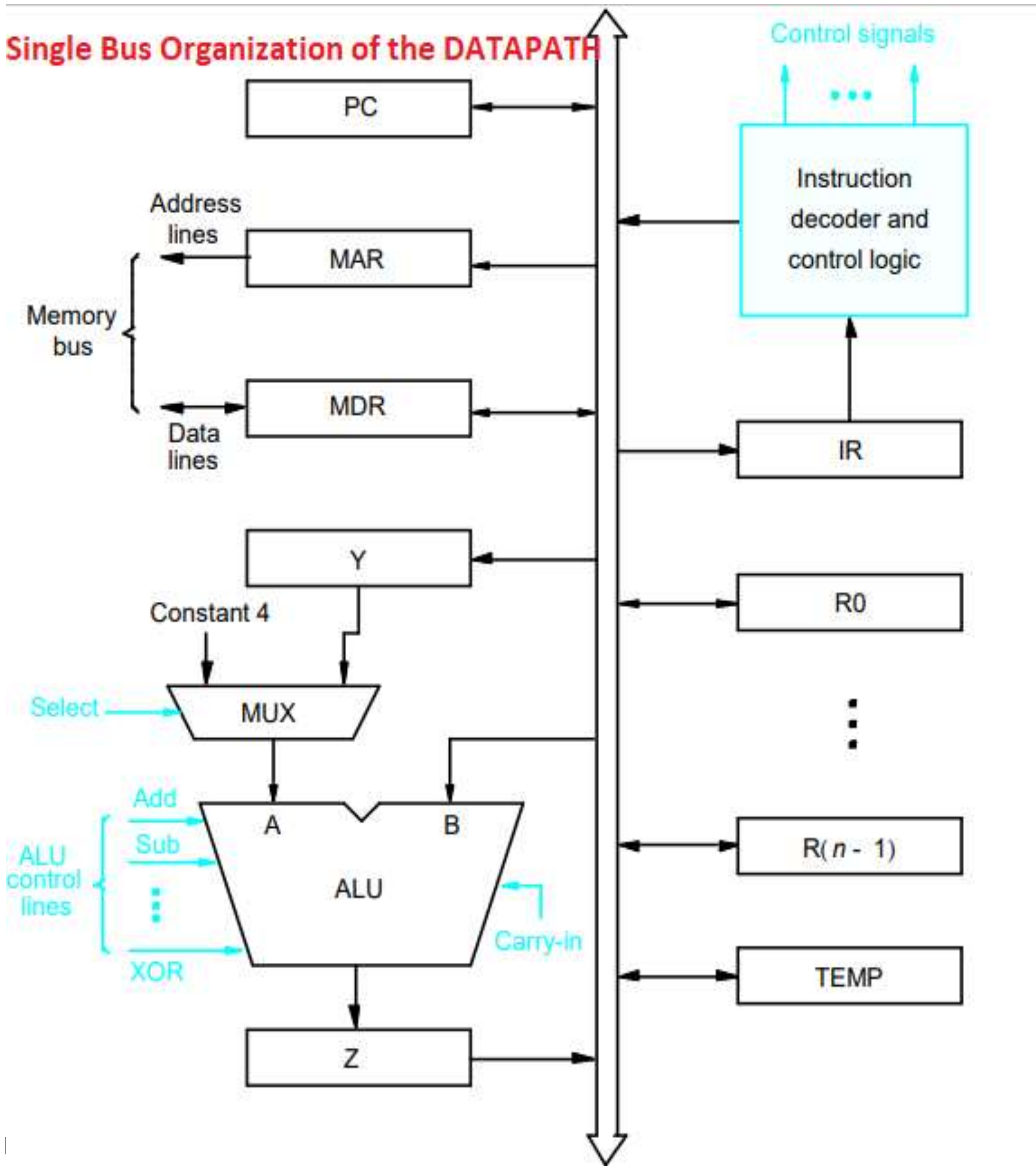
Fetching an instruction and loading it into the IR is usually referred to as the instruction fetch phase. Performing the operation specified in the instruction constitutes the instruction execution phase.

Connection Between the Processor and the Memory



Single Bus Organization of Datapath in CPU

- Figure shown an organization in which the arithmetic and logic unit (ALU) and all the registers are interconnected via a single common bus. This internal bus is different from the external bus which connects processor and memory.
- The data and address lines of the external memory bus are connected to the internal bus via MAR and MDR registers respectively. MDR – Memory data register, has two inputs and outputs, data may be loaded either from external bus or from internal bus, also the data stored in MDR may be placed on either bus. MAR – Memory address register, the input is connected to internal bus, o/p is connected to external bus.
- The control lines of the memory bus are connected to instruction decoder and control logic block
- Registers R0 –Rn are for the programmer, registers Y,Z and TEMP are for temporary storage during the execution, not for programmer usage.
- The MUX selects Y output or constant 4, and feeds to A input of ALU. The constant 4 is used to increment the value of PC by 4. (Select4 or SelectY are used to select).
- The decoder generates the control signals needed to select the registers involved and direct the transfer of data.
- The registers, the ALU, and the interconnecting bus are collectively referred to as the datapath.



The operation specified by an instruction can be carried out by performing one or more of the following operations in some specified sequence.

- Transfer a word of data from one processor register to another or to the ALU
- Fetch the contents of a given memory location and load them into a processor register.
- Perform an arithmetic or logic operation and place the result into a processor register.
- Store data from a processor register into a given memory location.

Let us consider these operations and execution of instructions using single bus processor model.

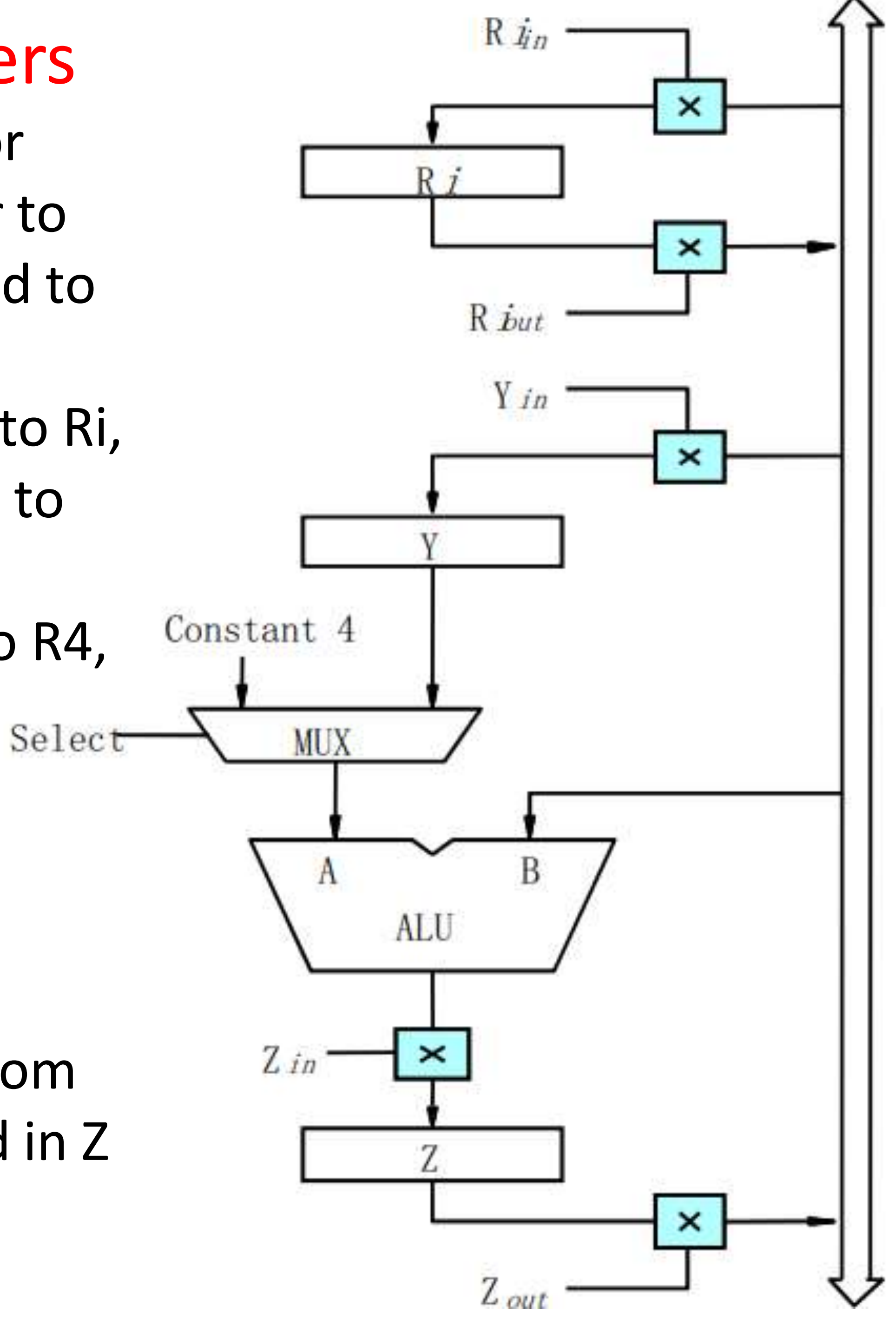
Input and Output Gating control for Registers

For each register, two control signals are provided, one for loading the data from bus to register, and one for register to the bus. The input and output of Register Ri are connected to the bus via switches controlled by the signals Ri_in and Ri_out. When Ri_in is set to ‘1’, the data moves from bus to Ri, similarly when Ri_out is set to ‘1’, the data moves from Ri to bus.

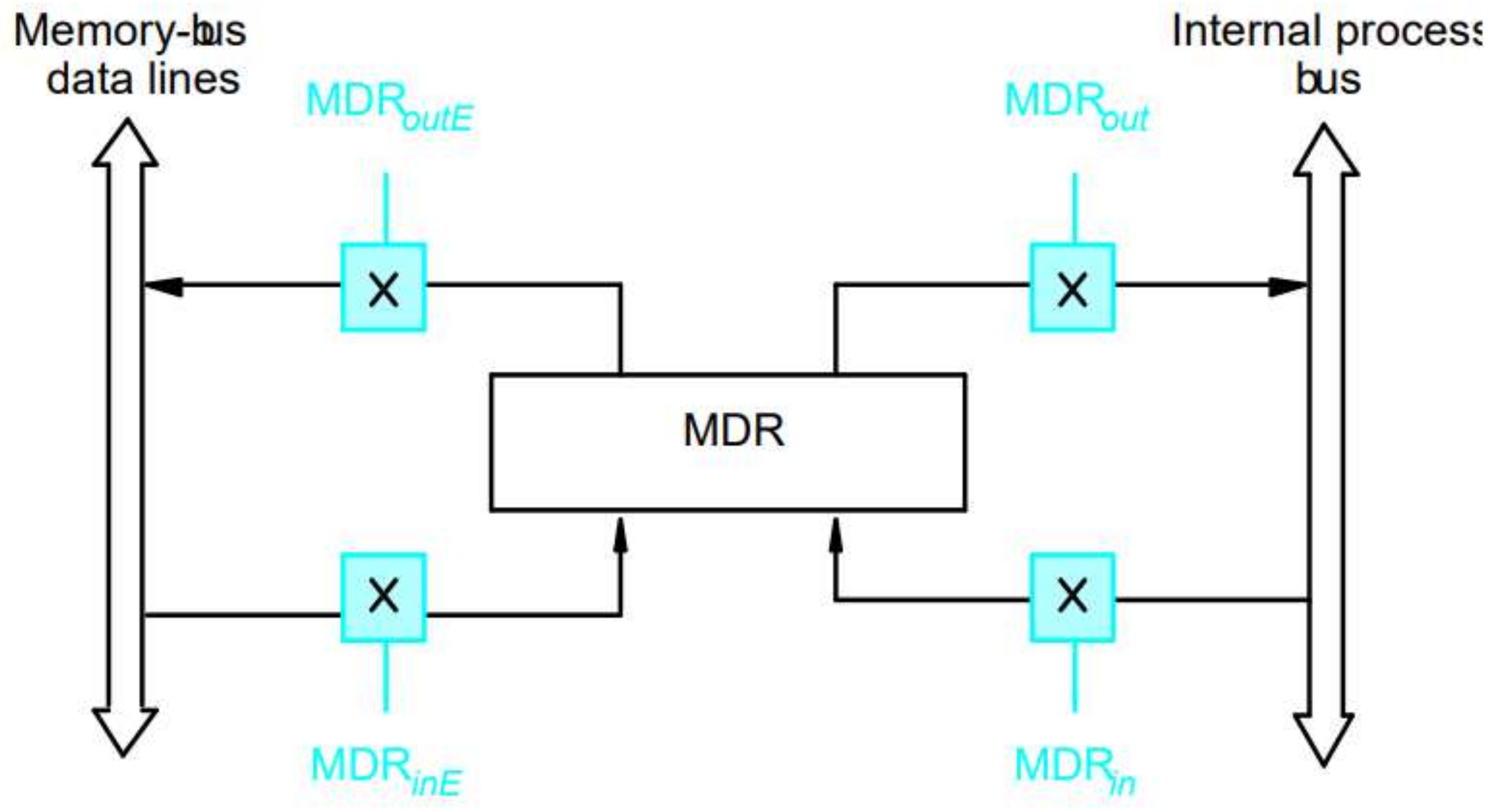
Example: suppose we wish to transfer the data from R1 to R4, then

- Send ‘1’ to R1_out, so data moves from R1 to bus
- Send ‘1’ to R4_in, so data moves from bus to R4

ALU performs operations on operands placed on A and B inputs. Input on A comes from Mux and input B comes from the bus. The result produced by ALU is temporarily stored in Z register. The function performed by ALU depends on the control signals given, like Add/Sub…



The connections for register MDR are shown in the figure. It has four control signals,MDRin and MDRout control the connection to the internal bus, and MDRin_E and MDRout_E control the connection to the external bus.



Performing an ALU operation, with registers

Write a Control Sequence for the instruction, ADD R1,R2,R3

1.R1out,Yin (contents of R1 moved to Y)
2.R2out,SelectY,Add,Zin (R2 is added with Y, answer is moved to Z
3.Zout,R3in (contents of Z moved to R3)

The signals whose names are given in any step are activated for the duration of the clock cycle corresponding to that step, all other signals are inactive. Important to note, only one data transfer can happen at a time on the bus. Hence last data transfer (step3) can't be carried out during step2.

Fetching a word from Memory

Write the Control Sequence for the instruction, involving memory read operation, assume instruction is already fetched from memory and stored in IR.

Move (R1),R2 ;

(this instn. moves the data from memory location, whose address is indicated by R1, to R2)

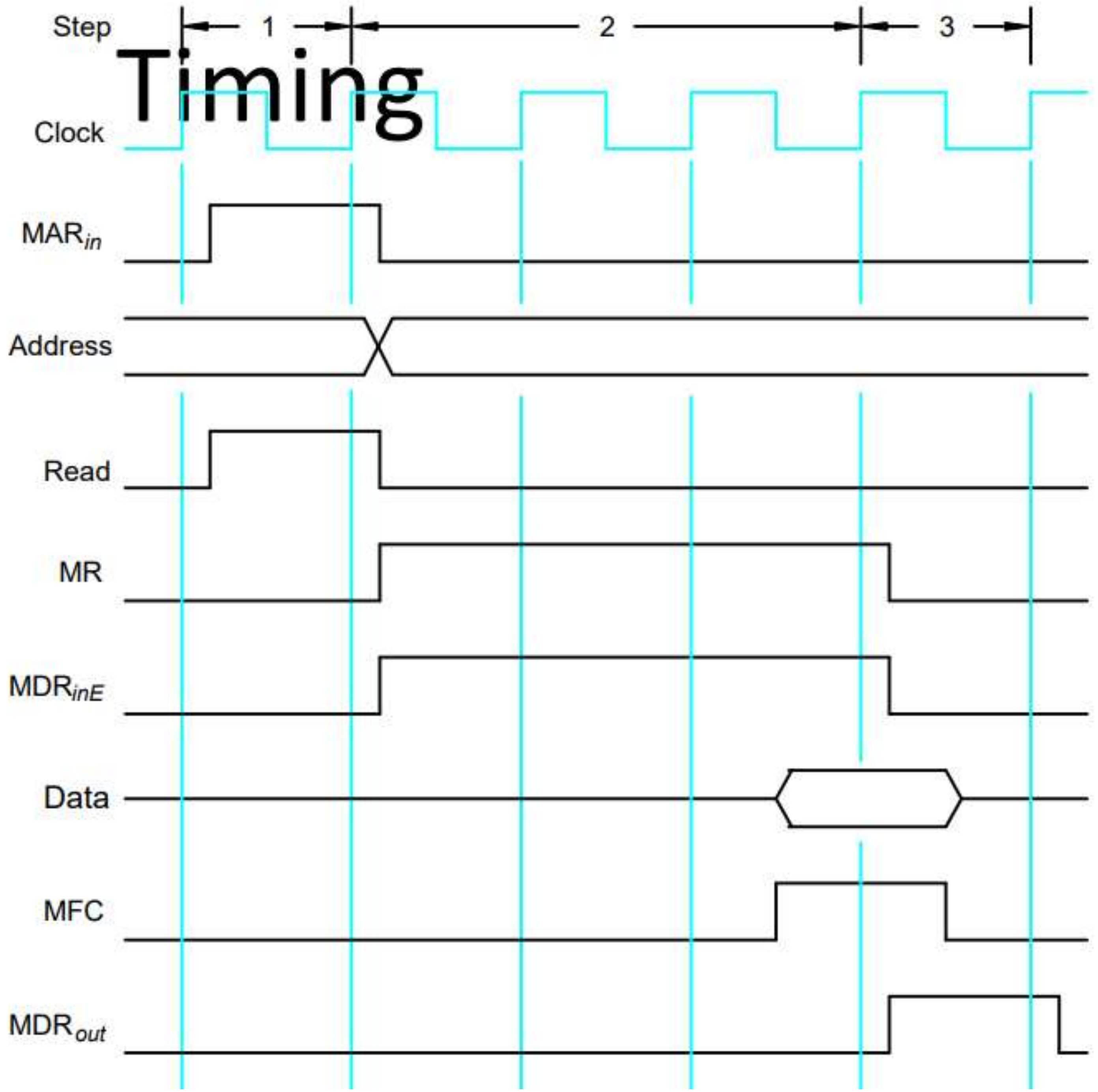
1.R1out,MARin,Read (place the memory address from R1 to MAR reg, and initiate memory read operation)
2.MDRin_E,WMFC (wait for MFC signal from memory, indicating memory read is done, data is copied to MDR)
3.MDRout,R2in (move the data from MDR to R2)

Timing of a Memory Read Operation

Assume MAR is always available on the address lines of the memory bus.

- Move (R1), R2

1. R1out, MARin, Read
2. MDRinE, WMFC
3. MDRout, R2in



Storing a Word In Memory.

Write the control sequence for memory write operation of the instruction MOV R2,(R1). (assuming the instruction is already fetched and stored in IR)

This instruction writes the data in R2 to memory location, whose address is there in R1.

1.R1out, MARin (move the memory address in R1 to MAR, since MAR is connected to memory address bus, this address is sent to Memory)
2.R2out,MDRin,Write (move the data in R2, to MDR register, initiate Write command)
3.MDRout_E,WMFC (place the data in MDR register to memory data bus, wait form memory write operation to complete, by checking the signal MFC)

Write the control sequence for the execution of complete instruction ADD (R3),R1

This instruction adds the contents of memory location pointed by R3 to R1, and store the answer in R1, Executing this instruction requires the following actions,

1. Fetch the instruction,
2. Fetch the first operand (the contents of the memory location pointed to by R3),
3. Perform the addition,
4. Load the result into R1. Control sequence is indicated below

1. PCout,MARin,Read,Select4,Add,Zin (place PC contents on MAR, start Read operation, use ALU to ADD 4 to PC contents)
2. Zout,PCin,Yin,WMFC (store the new value to PC, i.e old +4, wait for Instruction arrival from memory)
3. MDRout,IRin (instruction read from memory is moved to IR)
4. R3out,MARin,Read (move R3, which contains adds, to MAR, initiate memory Read operation)
5. R1out,Yin,WMFC (wait for the data to come from memory, at the same time, move other operand in R1 to Y, as bus free)
6. MDRout,SelectY,Add,Zin (move memory data from MDR and add to Y, and store in Z)
7. Zout,R1in,End (move the answer from Z to R1)

Write the control sequence for the execution of complete instruction ADD R2,R1

This instruction adds the contents of R2 with R1, store the answer in R1

1. Fetch the instruction,
2. Perform the addition,
3. Load the result into R1.

Control sequence is indicated below

1. PCout,MARin,Read,Select4,Add,Zin (place PC contents on MAR, start Read operation, use ALU to ADD 4 to PC contents)
2. Zout,PCin,Yin,WMFC (store the new value to PC, i.e old +4, wait for Instruction arrival from memory)
3. MDRout,IRin (instruction read from memory is moved to IR)
4. R1out,Yin (move R1 to Y)
5. R2out,SelectY,Add,Zin (move R2 to B input of ALU and add to Y, and store in Z)
6. Zout,R1in,End (move the answer from Z to R1)

Execution of Branch Instructions

- A branch instruction replaces the contents of PC with the branch target address, which is usually obtained by adding an offset X given in the branch instruction.
- The offset X is usually the difference between the branch target address and the address immediately following the branch instruction.

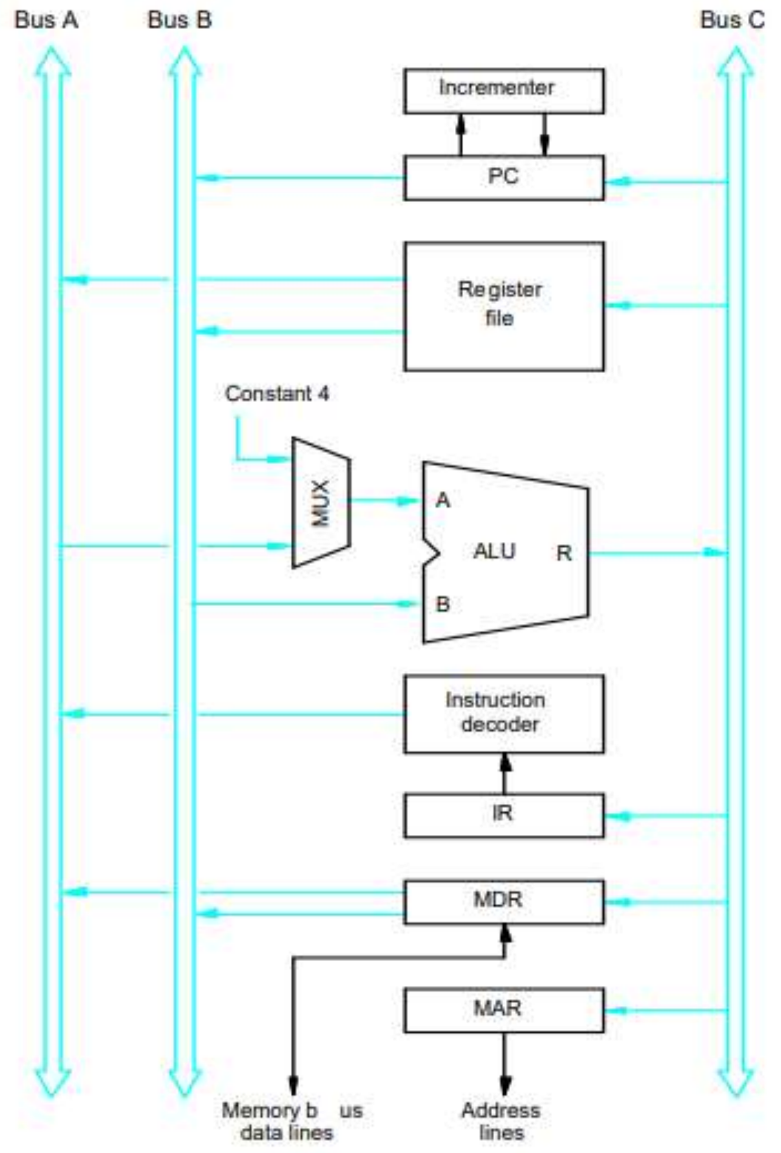
Control Sequence for unconditional branch instruction

1.PCout,MARin,Read,Select4,Add,Zin
2.Zout,PCin,Yin,WMFC
3.MDRout,IRin (get the branch instruction to IR)
4.Offset-field-of-IRout,Add,Zin (extract the offset from IR and add to PC, to get new PC adds for branch)
5.Zout,Pcin,End (move the new adds to PC, and branch happens)

Control sequence for condition branch instruction, Branch < 0

1.PCout,MARin,Read,Select4,Add,Zin
2.Zout,PCin,Yin,WMFC
3.MDRout,IRin (get the branch instruction to IR)
4.Offset-field-of-IRout,Add,Zin, if N=0 then End (it will not execute step5, if condition is not satisfied)
5.Zout,PCin,End (move the new adds to PC)

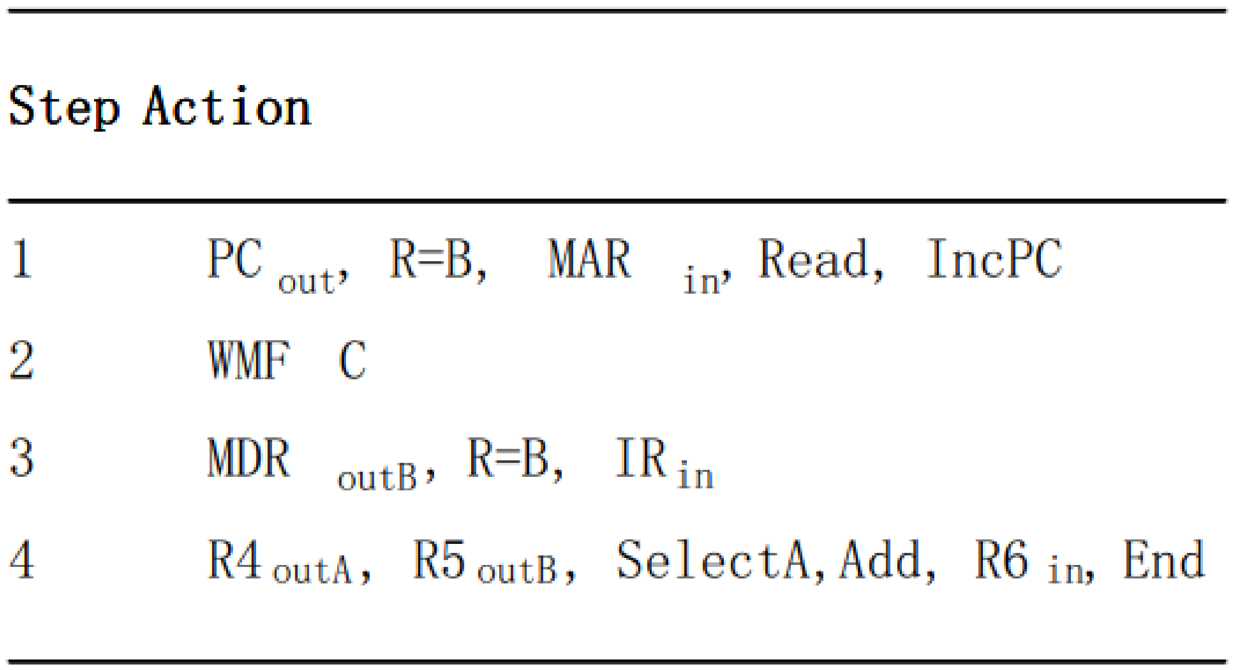
Multiple-Bus Organization (3 Bus)



- Allow the contents of two different registers to be accessed simultaneously and have their contents placed on buses A and B.
- Allow the data on bus C to be loaded into a third register during the same clock cycle.
- Incrementer unit.
- ALU simply passes one of ts two input operands unmodified to bus C

$\rightarrow$ control signal: R=A or R=B

Wrie control sequence for Add R4,R5,R6 for 3bus organization



Step	Action
1	PC_{out}, R=B, MAR_{in}, Read, IncPC
2	WMF C
3	MDR_{outB}, R=B, IR_{in}
4	$R4_{outA}$, $R5_{outB}$, SelectA, Add, $R6_{in}$, End

Hardwired Control

- To execute instructions, the processor must have some means of generating the control signals needed in the proper sequence.
- Two categories: hardwired control and microprogrammed control
- Hardwired system can operate at high speed; but with little flexibility.

Hardwired Control

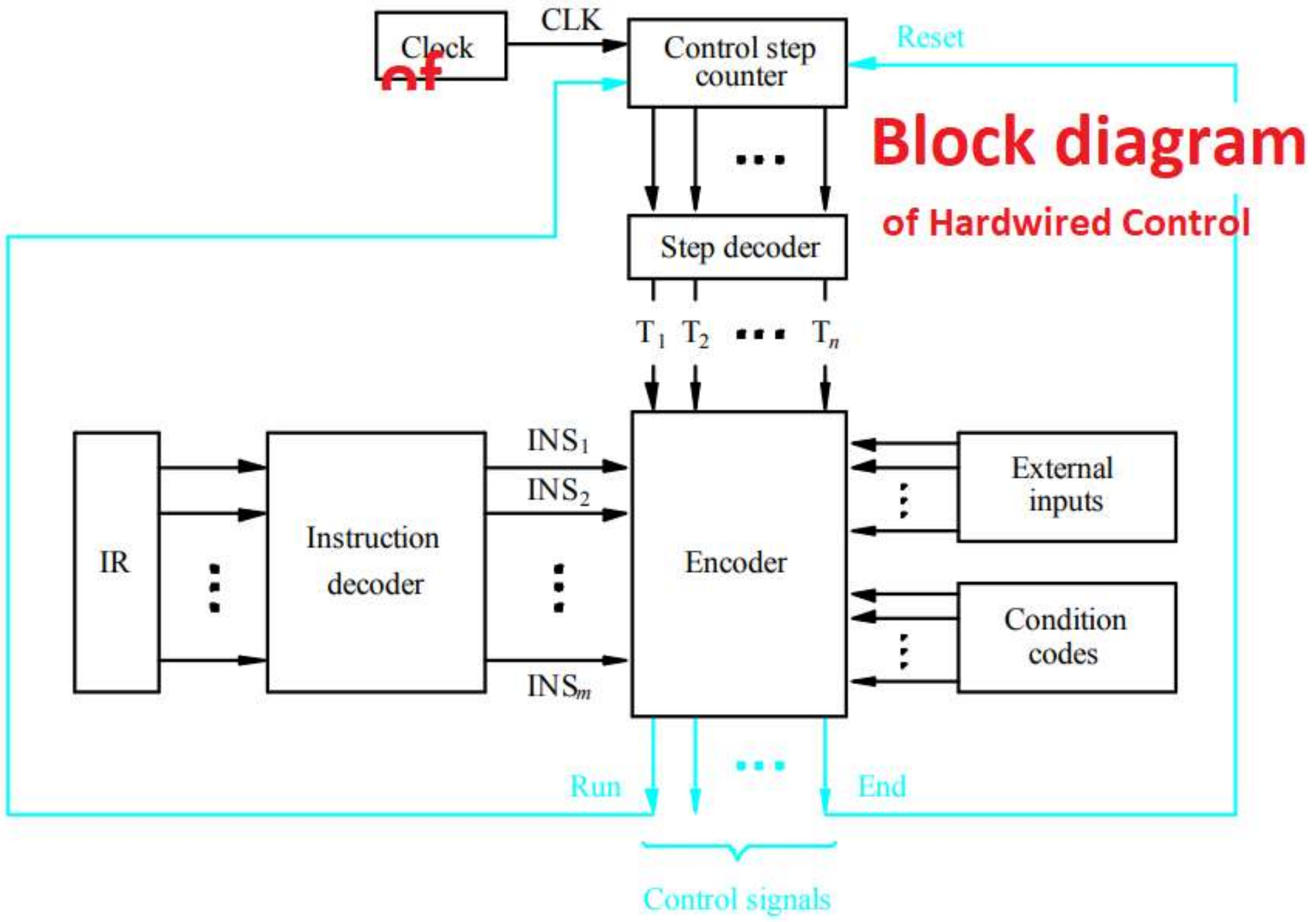
Every instruction, is sequence of control signals, generated in steps.

Each step is completed in one clock cycle. A counter is required to keep track of these steps, i.e control step counter is used, each state or count of this counter corresponds to one control step, the step decoder provides separate signal line for each step.
The output of a instruction decoder consists of separate line for each instruction.

Encoder receives the signals from External inputs like MFC,interrupts etc. Encoder also receives the data from conditional codes like Zero flag,Negative flag. Etc.
The input signals to the encoder block are combined to generate the individual control signals Zin,Yin,PCout etc
For example,

Zin = T1 + T6.ADD + T4.BR + ….

This signal is asserted during the time slot T1(first step) for all instructions, during T6 for an ADD instruction, during T4 for unconditional branch instruction, and so on…



Block diagram of Hardwired Control

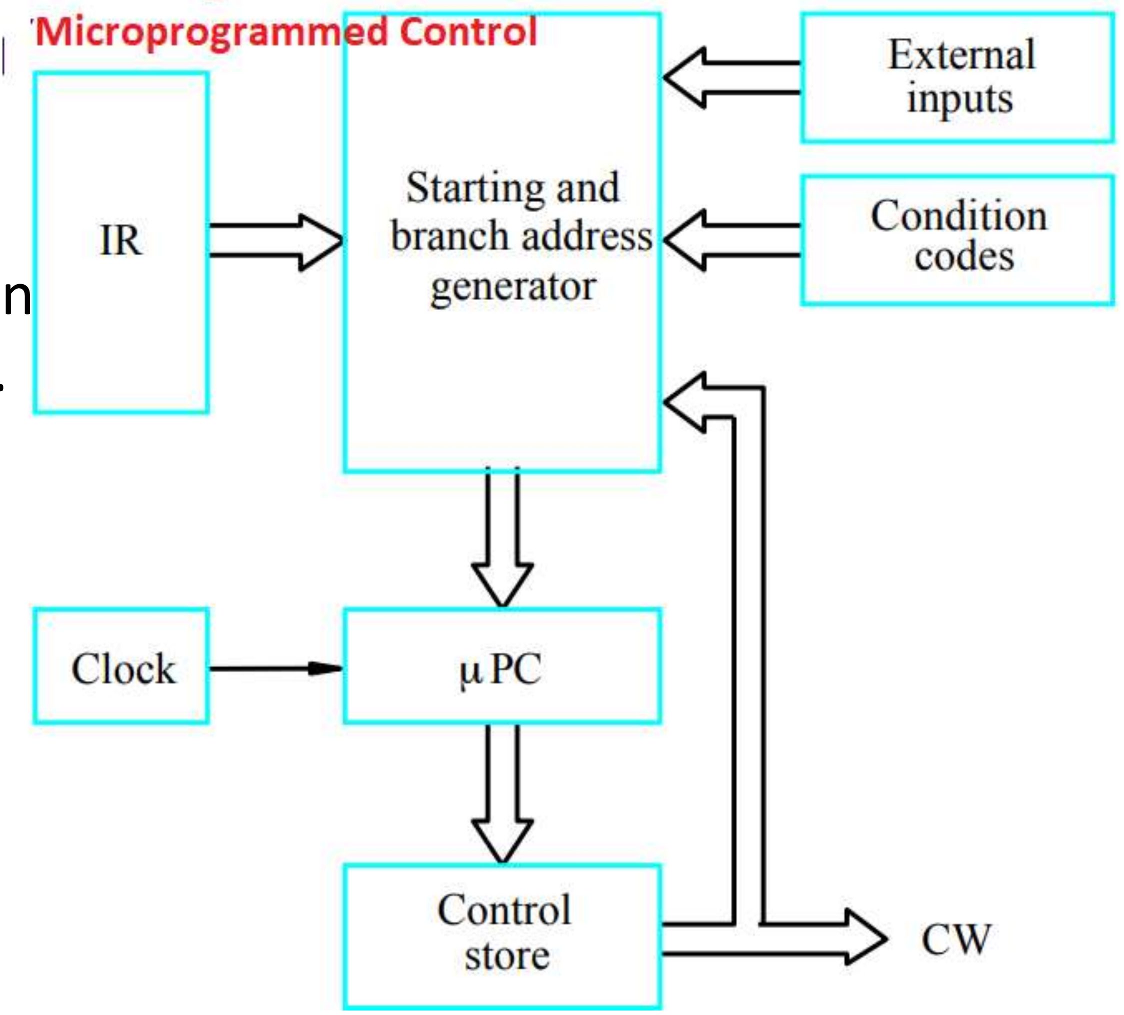
MicroProgrammed Control – very flexible

In this method control signals are generated by a program similar to machine language programs.

A control word is a word, whose individual bits represent the various control signals. Each of the control steps in the control sequence of an instruction defines a unique combination of 1s and 0s in the CW. 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 micro instructions.

The microroutines for all instructions in the instruction set of a computer are stored in a special memory called the control store. To read the control words sequentially, a micro program counter is used

Block Diagram of Microprogrammed Control


IR
Starting and branch address generator
External inputs
Condition codes
Clock
μ PC
Control store
CW