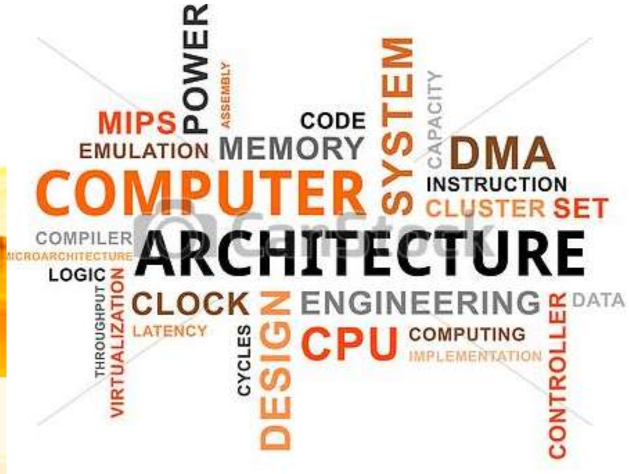
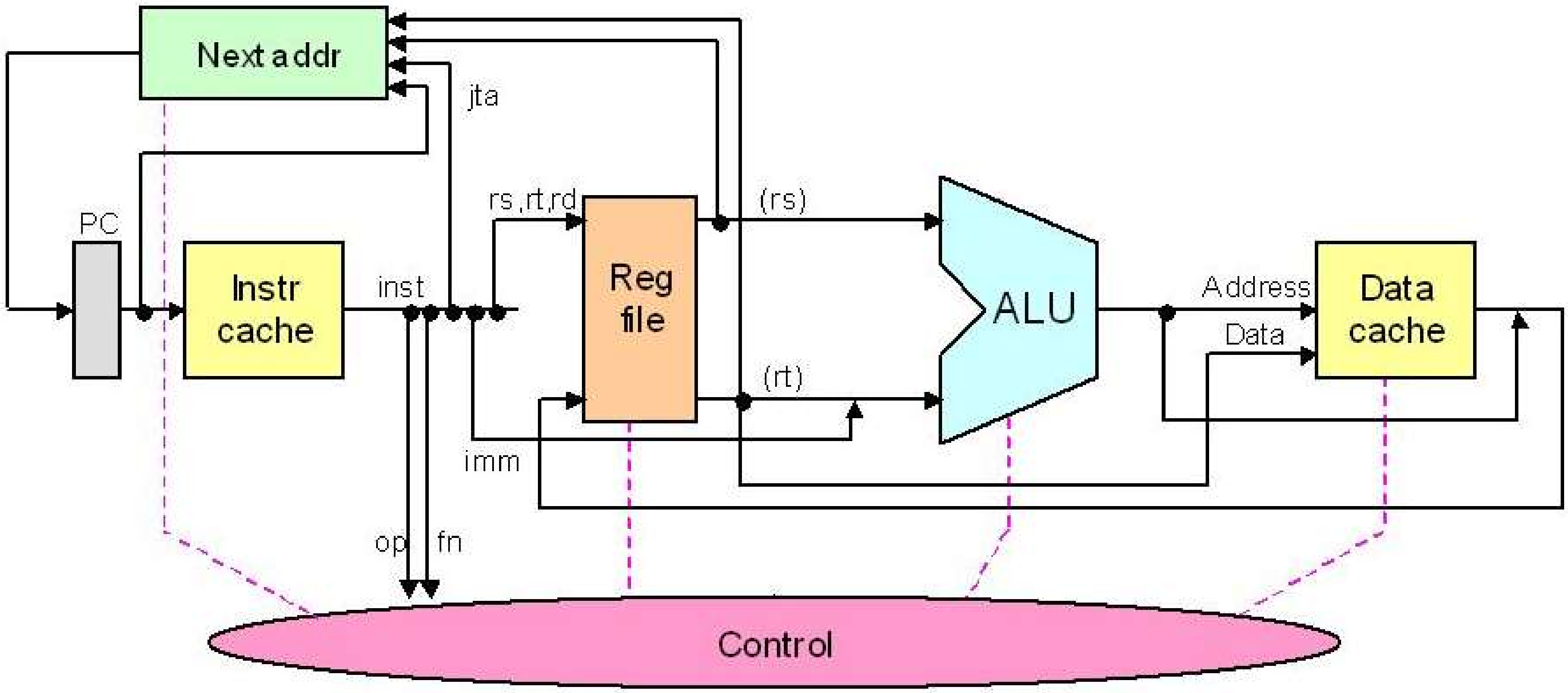
UNIT III
PROCESSOR AND PIPELINING

Fundamental concepts – **Execution of a complete instruction** – Multiple bus organization – Hardwired control – Micro programmed control – Pipelining: Basic concepts – Data hazards – Instruction hazards – Influence on Instruction sets – Data path and control consideration.



Recap the previous Class


Next addr
jta
PC
Instr cache
inst
rs,rt,rd
Reg file
(rs)
(rt)
imm
op
fn
ALU
Address
Data
Data cache
Control

Executing an Instruction

- Fetch the contents of the memory location pointed to by the PC. The contents of this location are loaded into the IR (fetch phase).

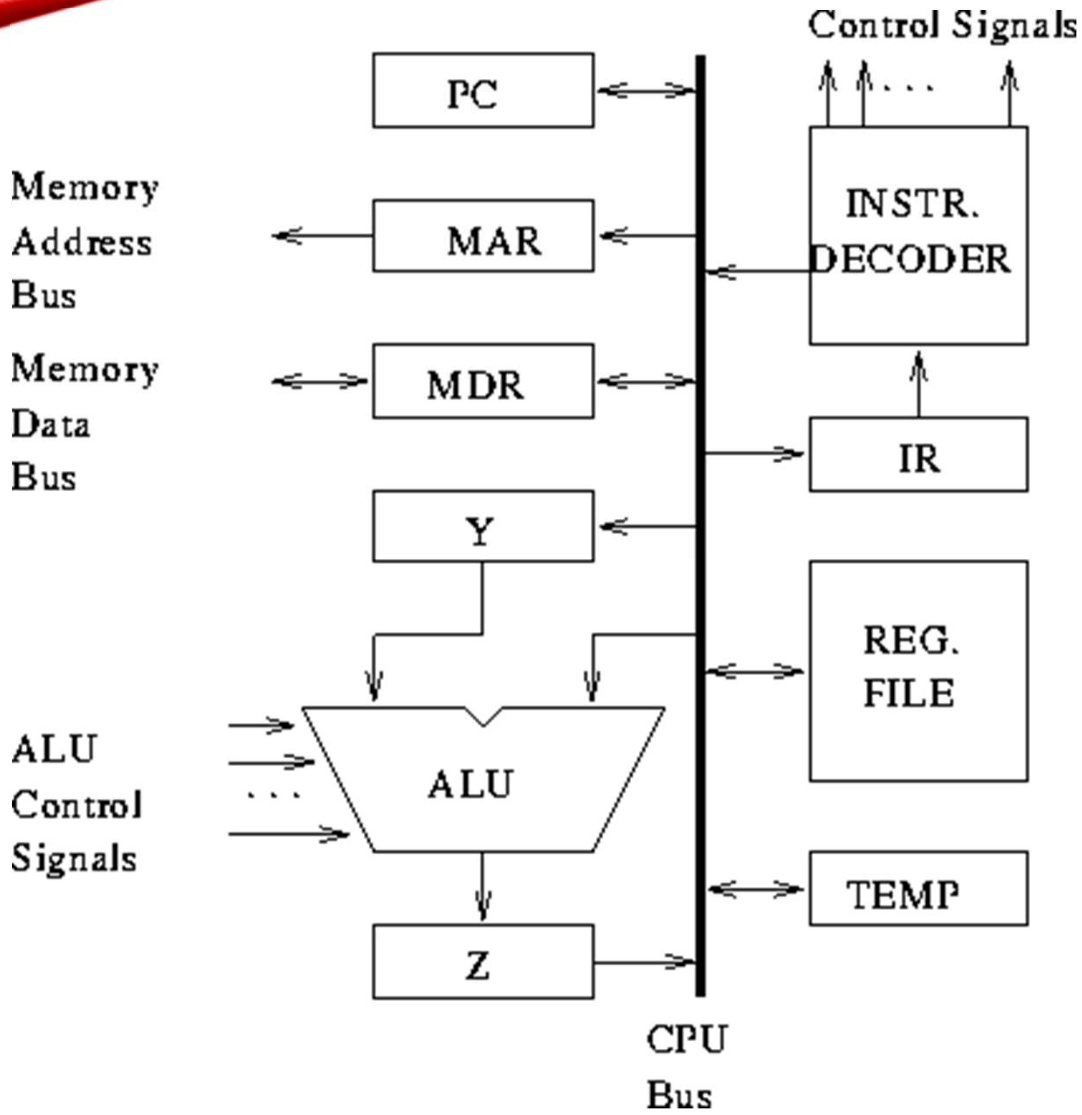
$$IR \leftarrow [[PC]]$$

- Assuming that the memory is byte addressable, increment the contents of the PC by 4 (fetch phase).

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

- Carry out the actions specified by the instruction in the IR (execution phase).

Processor Organization



Internal organization of the processor

- ALU
- Registers for temporary storage
- Various digital circuits for executing different micro operations.(gates, MUX, decoders, counters).
- **Internal path** for movement of data between ALU and registers.
- **Driver circuits** for transmitting signals to external units.
- **Receiver circuits** for incoming signals from external units.

Program Counter

- Keeps track of execution of a program
- Contains the memory address of the next instruction to be fetched and executed.

Memory Address Register

- Holds the address of the location to be accessed.
- I/P of MAR is connected to Internal bus and an O/p to external bus.

Memory Data Register

- Contains data to be written into or read out of the addressed location.
- IT has 2 inputs and 2 Outputs.
- Data can be loaded into MDR either from memory bus or from internal processor bus.

The data and address lines are connected to the internal bus via MDR and MAR

Registers

- The processor registers R0 to Rn-1 vary considerably from one processor to another.
- Registers are provided for **general purpose** used by programmer.
- **Special purpose registers-index** & stack registers.
- Registers Y,Z &TEMP are **temporary registers used by processor** during the execution of some instruction.

Multiplexer

- Select either the output of the register Y or a constant value 4 to be provided as input A of the ALU.
- Constant 4 is used by the processor to increment the contents of PC.

Arithmetic Logical Unit

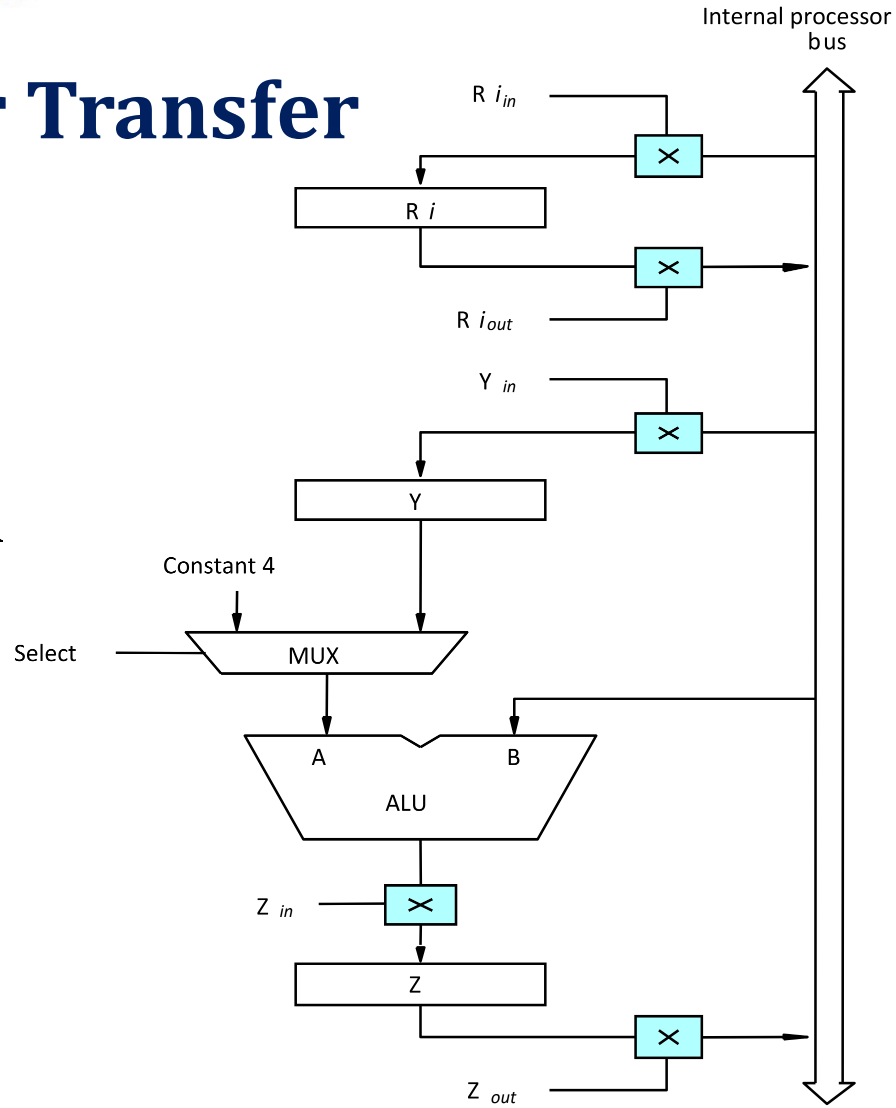
Used to perform arithmetic and logical operation.

Data Path

The registers, ALU and interconnecting bus are collectively referred to as the data path.

Register Transfer

- The input and output gates for register Ri are controlled by signals is R_{in} and Ri_{out} .
- Rin Is set to1 – data available on common bus are loaded into Ri.
- Ri_{out} Is set to1 – the contents of register are placed on the bus.
- Ri_{out} Is set to 0 – the bus can be used for transferring data from other registers .



Data transfer between two registers

Transfer the contents of R1 to R4.

1. Enable output of register R1 by setting $R1_{out}$=1. This places the contents of R1 on the processor bus.
2. Enable input of register R4 by setting $R4_{in}$=1. This loads the data from the processor bus into register R4.

Performing an Arithmetic or Logic Operation

- The ALU is a combinational circuit that has no internal storage.
- ALU gets the two operands from MUX and bus. The result is temporarily stored in register Z.
- What is the sequence of operations to add the contents of register R1 to those of R2 and store the result in R3?

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

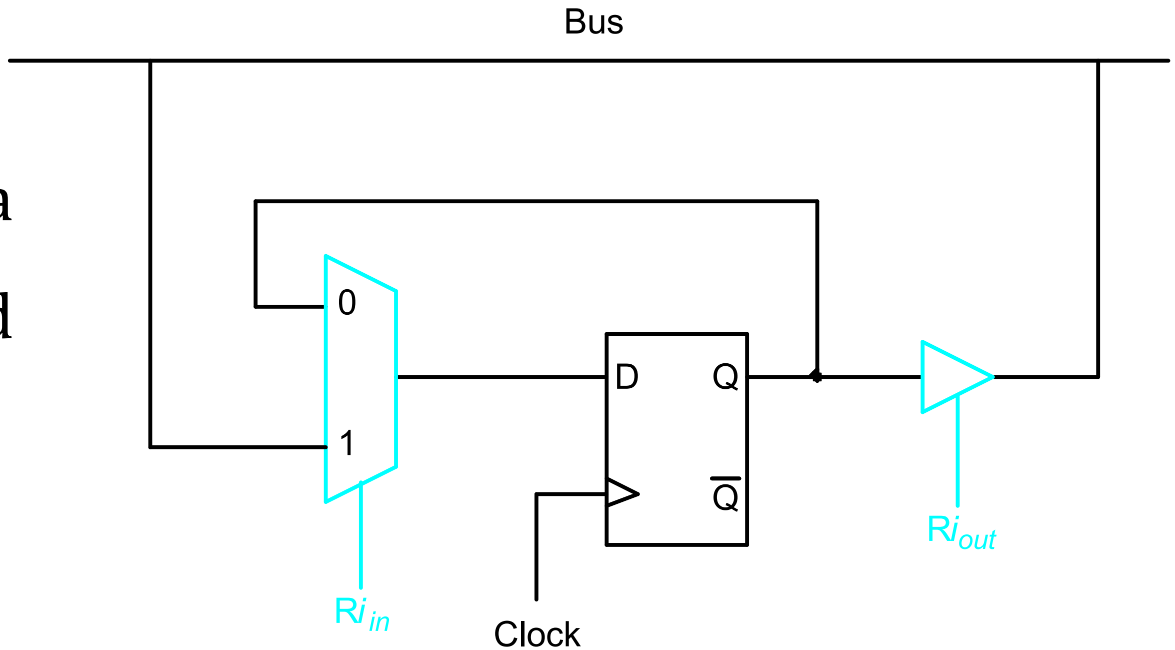
Step 1: Output of the register R1 and input of the register Y are enabled, causing the contents of R1 to be transferred to Y.

Step 2: The multiplexer's select signal is set to select Y causing the multiplexer to gate the contents of register Y to input A of the ALU.

Step 3: The contents of Z are transferred to the destination register R3.

Register Transfers

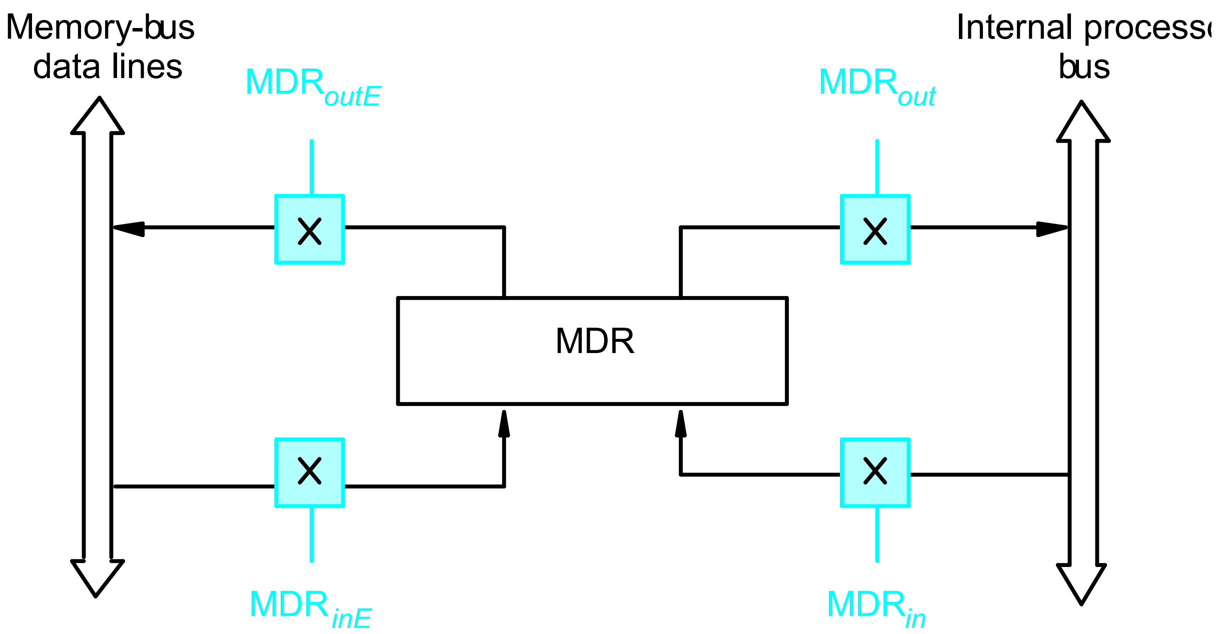
All operations and data transfers are controlled by the processor clock



Input and output gating for one register bit.

Fetching a Word from Memory

Address into MAR; issue Read operation; data into MDR.



. Connection and control signals for register MDR.

Fetching a Word from Memory

The response time of each memory access varies (cache miss, memory-mapped I/O,…).

To accommodate this, the processor waits until it receives an indication that the requested operation has been completed (Memory-Function-Completed, MFC).

Move (R1), R2

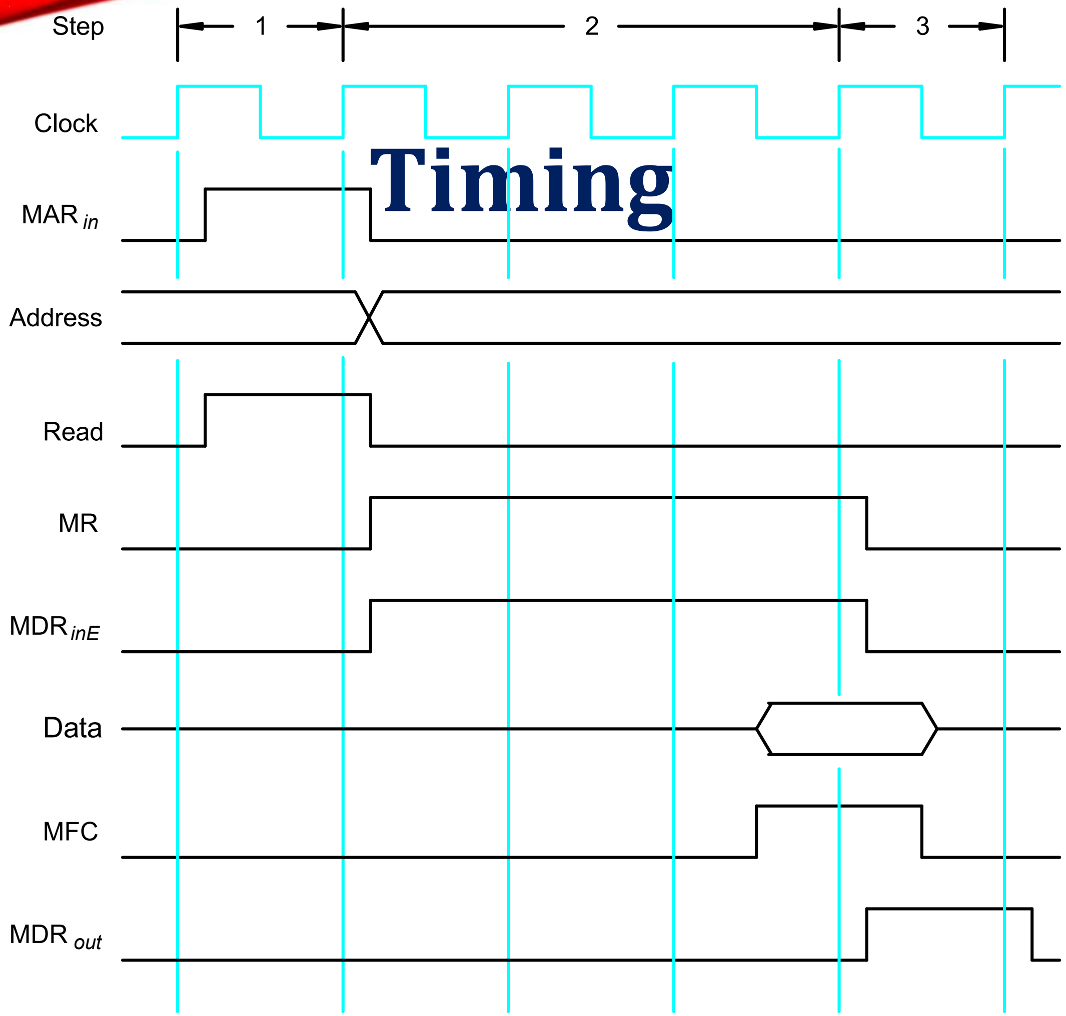
- MAR ← [R1]
- Start a Read operation on the memory bus
- Wait for the MFC response from the memory
- Load MDR from the memory bus
- R2 ← [MDR]

Timing

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

Move (R1), R2

1. $R1_{out}$, MAR_{in}, Read
2. MDR_{inE}, W_{MFC}
3. MDR_{out}, $R2_{in}$



Timing of a memory Read operation.

Storing a word in memory

Address is loaded into MAR

Data to be written loaded into MDR.

Write command is issued.

Example

- Move R2,(R1)
- $R1_{out}$,MAR_{in}
- $R2_{out}$,MDR_{in},Write
- MDR_{outE}, WMFC

Execution of a Complete Instruction

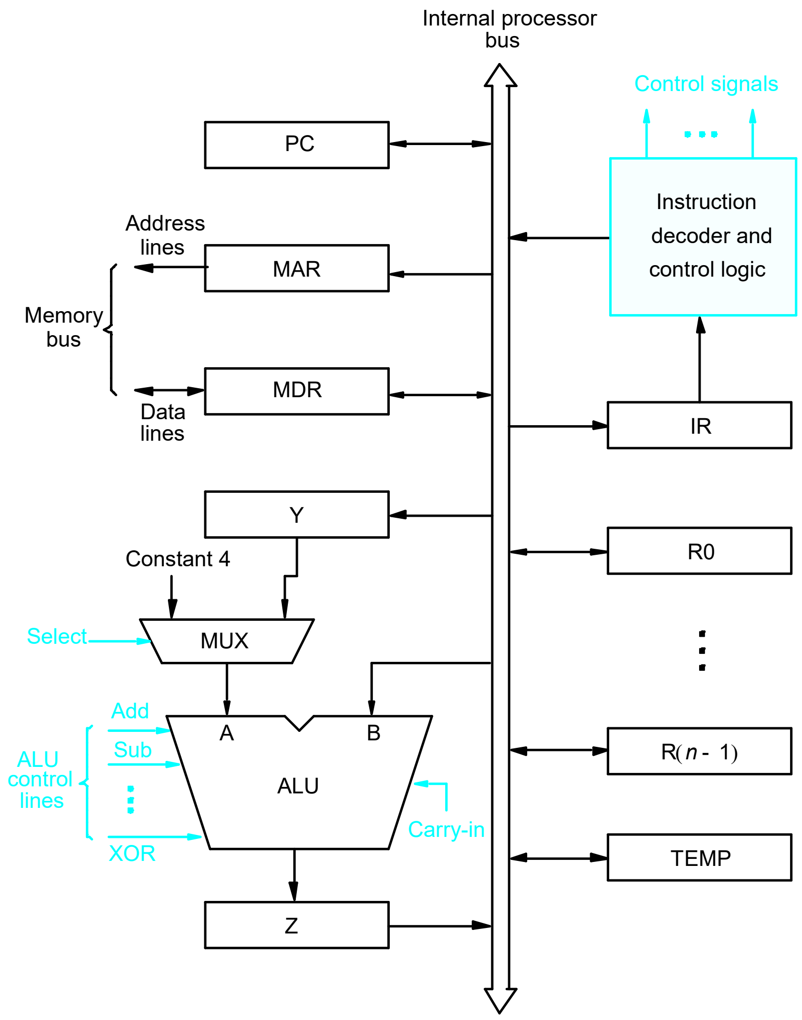
- Add (R3), R1
- Fetch the instruction
- Fetch the first operand (the contents of the memory location pointed to by R3)
- Perform the addition
- Load the result into R1

Execution of a Complete Instruction

Add (R3), R1

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

Control sequence for execution of the instruction Add (R3),R1.



Single-bus organization of the datapath inside a proc

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.
- Unconditional branch

Execution of Branch Instructions

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

. Control sequence for an unconditional branch instruction.


Thank You