COA

Unit - III Processor & Pipelining

Processor -

The Processing Unit, executes Machine instructions and coordinates the activities of other units. This unit is often called the Instruction set Processor (ISP), (or) the Processor. It Performs the tasks of Fetching, Decoding & executing instructions of a program.

High Performance Processor. → 2 Types

1) → Pipelined Organization → Where the execution of one instruction is started before the execution of the Preceding instruction is completed.

2) → Superscalar Operation → Several instructions are Fetched & executed at the same time.

Fundamental Concepts -

To execute a program, the processor fetches One Instruction at a time & Performs the operations specified.

* Inst. are Fetched from successive Memory Location.
* Processor Keeps track of Nxt Inst. Address using PC.
* IR. (Inst. Reg)

To execute an inst., Processor has to perform the following 3 steps -

1) Fetch the contents of memory loc pointed by PC. The contents of location PC are inst. to be executed. They are loaded into IR. written as, IR ← [[PC]]

2) Assuming memory is byte addressable, increment the contents of PC by 4. PC ← [PC] + 4

3) Carry out the actions specified by instruction in IR.

Step 1 & 2 → Fetch Phase.
Step 3 → Execution Phase.

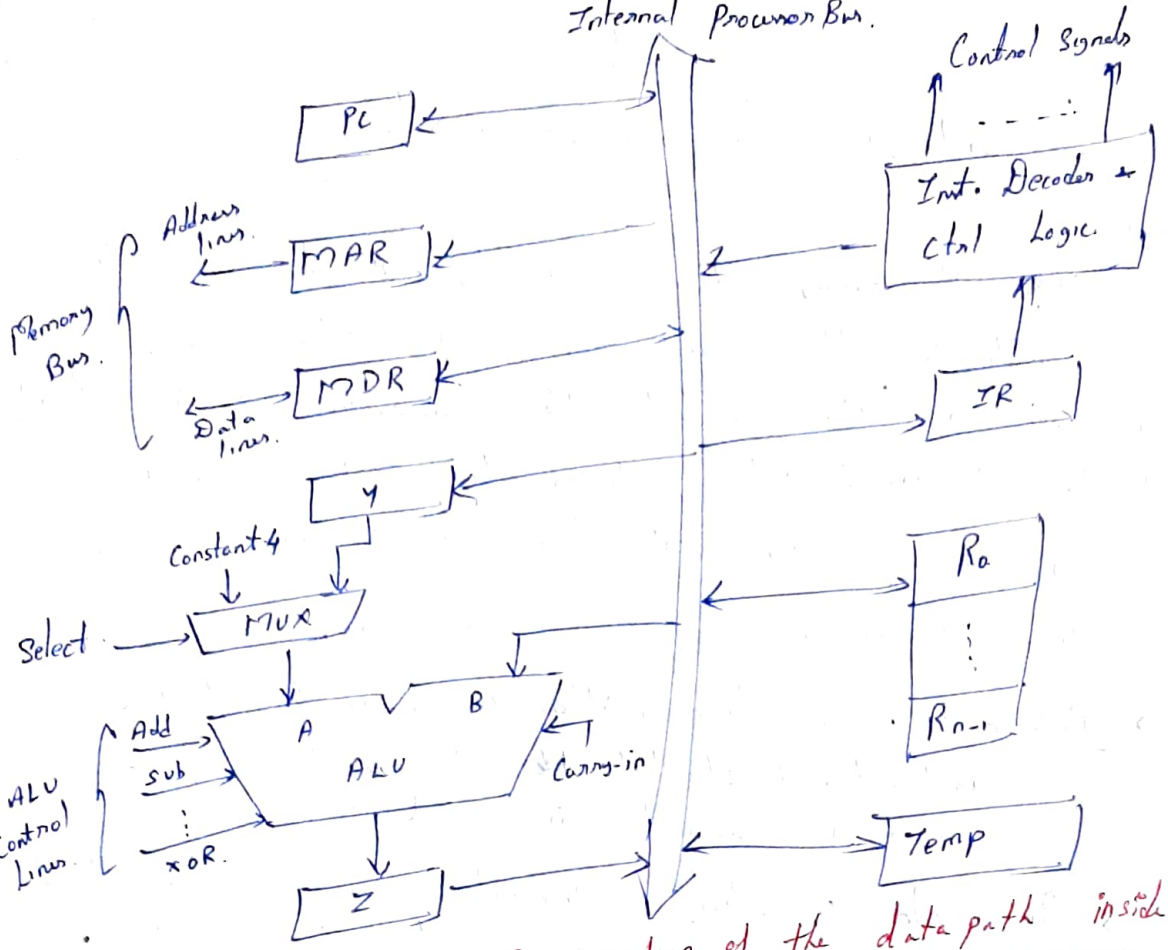

Fig 7.1 Single Bus Organization of the data path inside a Processor.

* PC
* IR
* MAR
* MDR
* R0 → Rn-1
* Temp Reg
* MUX by 4. to PC.
* ALU.

Instruction Decoder & Control Logic - is responsible for implementing the actions specified by the instruction loaded in IR Register. The Decoder generates the ctrl signals needed to select the registers involved and direct the transfer of data. The Registers, ALU & interconnecting bus are collectively referred to as data path.

An Inst. can be executed by performing one or more following operations in some specified sequence.

1) Transfer a word of data from one processor register to another or to the ALU
2) Perform arithmetic & logic operation & store result in a Processor Register.
3) Fetch the contents of given memory location & load them into a processor Register.
4) Store a Word of Data from a Processor Register into a given memory location.

Register Transfers

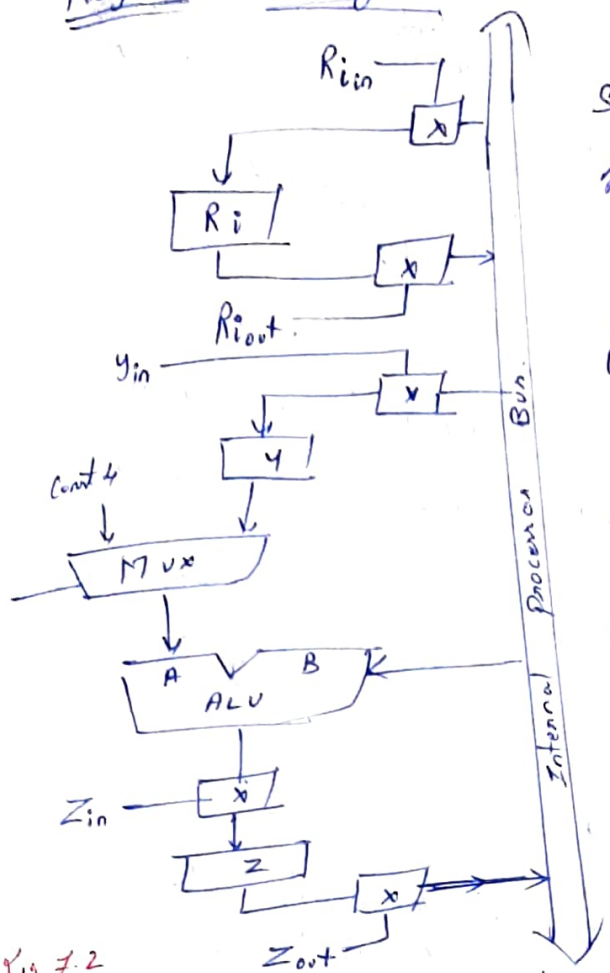

Fig 7.2
I/p & o/p gating for the Registers.

Inst. execution involves a sequence of data transfer steps from 1 register to another.

2 ctrl signals are used to place the contents of register on bus (or) to load data on bus into Registers.

The i/p & o/p of register R_i are connected to bus via switches controlled by signals R_{in} & R_{out}.

* R_{in} is set to 1 ⇒ Data on bus loaded into R_i.
* R_{out} is set to 1 ⇒ contents of Reg R_i are placed on Bus.
* R_{out} is 0 ⇒ Bus can be used for transferring data from other registers.

eg The transfer of contents of reg R_1 to R_4, can be accomplished as follows.

* Enable o/p of R_1, R_{1out} to 1 (This places the contents of R_1 on Processor Bus)
* Enable i/p of R_4, R_{4in} to 1 (This Loads data from Processor bus into Register R_4)

All operations & Data t/fs within processor take place with time periods defined by the Processor clock.

Two or more clock signals may be needed to gaurantee proper t/fs of data, is known as multiphase clocking.

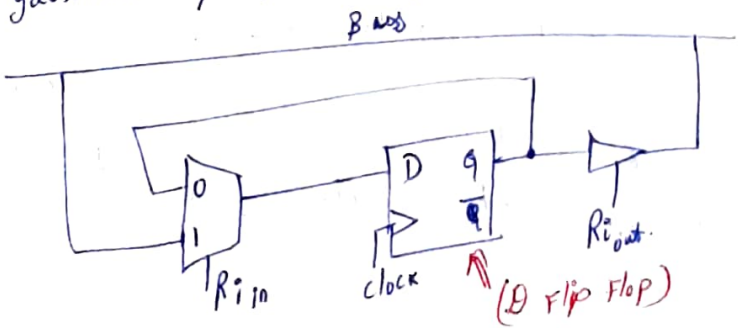

Fig 7.3
I/p & o/p gating for one Register Bit.

When Ri_{in} is 1 → multiplexer selects data on the bus

Ri_{in} is 0 → multiplexer feeds back the value currently stored in the Flip Flop.

Performing Arithmetic or Logic Operation —

ALU → Combinational circuit without internal storage

Perform operation on 2 i/p A & B.

* 1 i/p is from MUX
* Another i/p directly from Bus.

Result produced stored temporarily in register Z.

eg To add contents of R1 to R2 & store in R3

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

} only the corresponding signals will be active others will be inactive.

Fetching a word from memory —

To fetch a word, Processor need to specify addr of mem. loc. where the info. is stored & request Read operation.

⇒ The processor t/fr required addrs to MAR, o/p connected to address lines of memory bus.

At the same time processor uses ctrl lines to indicate Read operation is needed.

⇒ When Requested data are received from memory, they are stored in MDR.

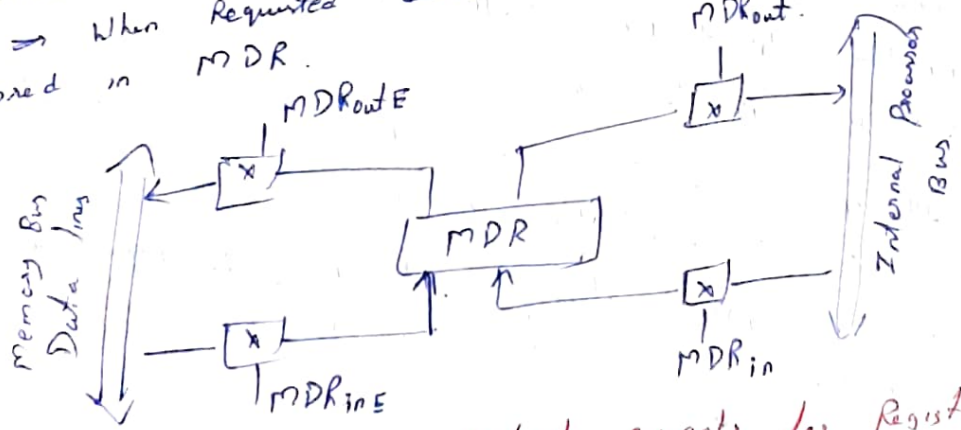

Fig 7.4 Connection & Control signals for Register MDR.

where, MDR_{in} & MDR_{out} → Ctrl connection to internal Bus.

MDR_{inE} & MDR_{outE} → ctrl connection to External Bus.

The processor waits till Read operation completed. ctrl signal called. Memory-Function completed (MFC) is used.

MFC sets signal to 1, to indicate contents of loc are read & available on data lines of memory Bus

Eg Move (R1), R2

Actions needed ⇒ 1) MAR ← [R1]
2) Start Read operation on memory Bus
3) Wait for MFC response from memory
4) Load MDR from memory Bus.
5) R2 ← [MDR].

Each action completed in 1 clock cycle. Except (3), requires 1 or more clock cycles, depending the speed of addressed device.

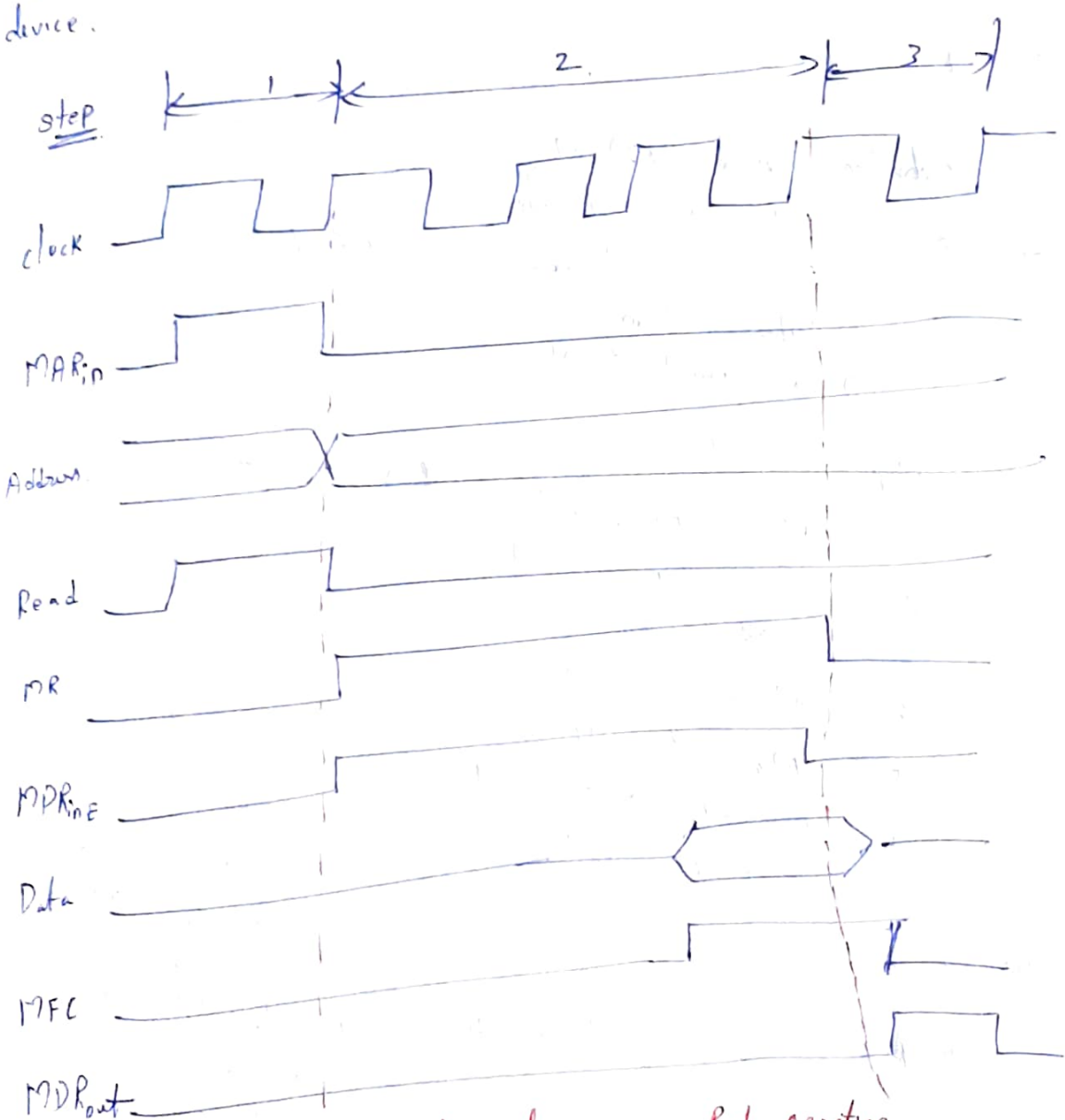

Fig 7.5 Timing of a memory Read operation.

MR requires 3 steps.

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

{ WMFC ⇒ Wait for arrival of MFC

Storing A word in memory –

* Address loaded into MAR.
* Data to be written are loaded to MDR.
* Write command is issued.

eg Move R2,(R1) Actions ⇒ 1) $R1_{out}$, MAR_{in}

2) $R2_{out}$, MDR_{in}, Write.

3) MDR_{outE}, WMFC