Computer Organization and Architecture

Module 4

Design of Control Unit

Prof. Indranil Sengupta

Dr. Sarani Bhattacharya

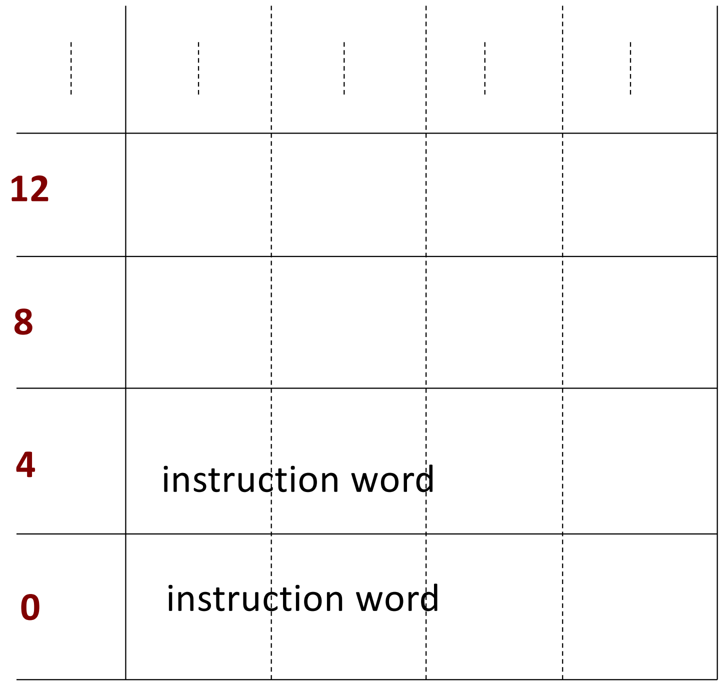
Department of Computer Science and Engineering

IIT Kharagpur

Design of Control Unit

Instructions

- Instructions are stored in main memory.
- Program Counter (*PC*) points to the next instruction.
    - MIPS32 instructions are 4 bytes (32 bits) long.
    - All instructions starts from an address that is multiple of 4 (last 2 bits 00).
    - Normally, *PC* is incremented by 4 to point to the next instruction.
    - For branch, *PC* is loaded with the address of the target instruction.

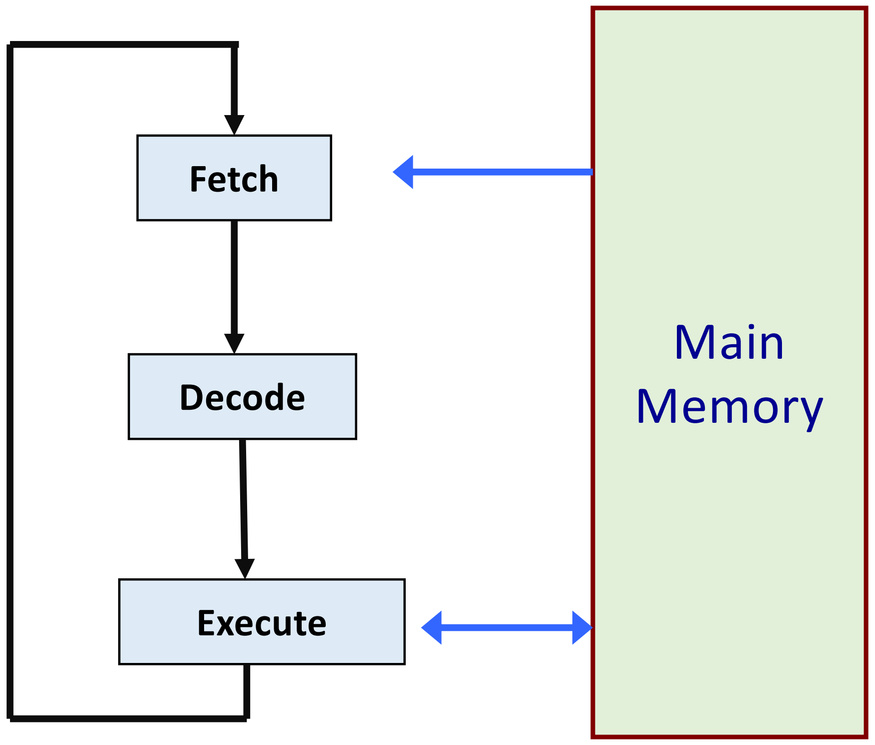


Addressing a Byte in Memory

- Each byte in memory has a unique address.
  - Memory is said to be *byte addressable*.
- Typically the instructions are of 4 bytes, hence the instruction memory is addressed in terms of 4 bytes (word length = 32 bits).
- When an instruction is executed, PC is *incremented by 4* to point to the next instruction.
  - In MIPS32, words are byte aligned.
  - Every word (including instruction) starts from a memory address that is some multiple of 4 (i.e., last two bits are `00’).

How an instruction Gets Executed?

```
repeat forever
      // till power off or
      // system failure
{
    Fetch instruction
    Decode instruction
    Execute instruction
}
```



The Fetch-Execute Cycle

1) Fetch the next instruction from memory.
2) Decode the instruction.
3) Execute the operation.
    - Get data from memory if needed (data not available in the processor).
    - Perform the required operation on the data.
    - May also store the result back in memory or register.

Registers: PC and IR

- **Program Counter (PC)** holds the address of the memory location containing the next instruction to be executed.
- **Instruction Register (IR)** contains the current instruction being executed.
- Basic processing cycle:
  - Instruction Fetch (IF)

    IR ← Mem [PC]
  - Considering the word length of the machine is 32 bits, the PC is incremented by 4 to point to the next instruction.

    PC ← PC + 4
  - Carry out the operations specified in *IR*.

Example: Add R1, R2

Address	Instruction
1000	ADD R1, R2
1004	MUL R3, R4

a)	PC	=	1000
b)	MAR	=	1000
c)	PC	=	PC + 4 = 1004
d)	MDR	=	"ADD R1, R2"
e)	IR	=	"ADD R1, R2"

(*Decode and finally execute*)

f)	R1	=	R1 + R2

May require one or more steps depending on the target architecture.

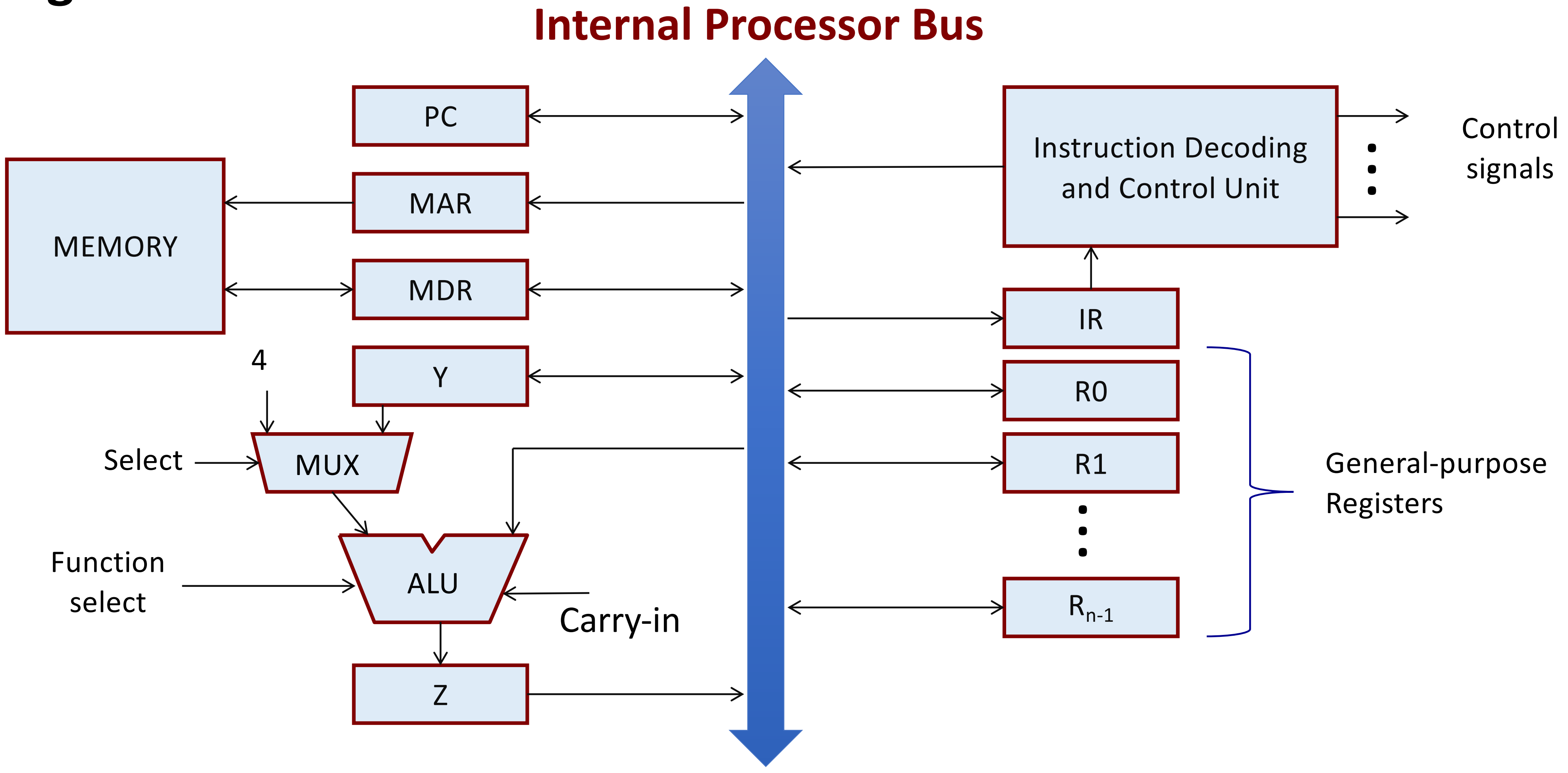
Requirement for Instruction Execution

- The necessary registers must be present.
- The internal organization of the registers must be known.
- The data path must be known.
- For instruction execution, a number of *micro-operations* are carried out on the data path.
  - An instruction consists of several micro-operations or micro-instructions.
  - Typically involves movement of data.

Kinds of Data Movement

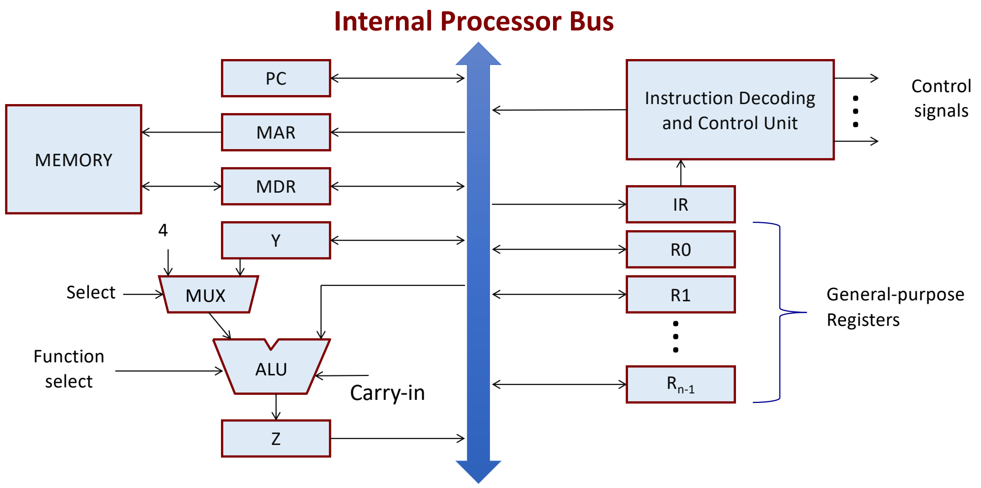
- Broadly three types:
  a) Register to Register
  b) Register to ALU
  c) ALU to Register
- Data movement is supported in the data path by:
  - The Registers
  - The Bus (single or multiple)
  - The ALU temporary Register (*Y* and *Z*)

Single Bus Organization

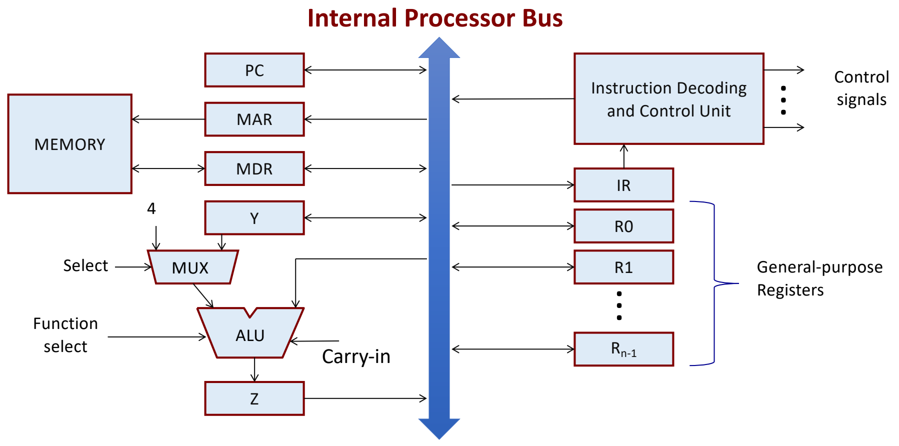


Single Internal Bus Organization

- All the registers and various units are connected using a single internal bus.
- Registers R_0-R_{n-1} are general-purpose registers used for various purposes.
- Registers *Y* and *Z* are used for storing intermediate results and never used by instructions explicitly.
- The multiplexer selects either a *constant 4* or output of register *Y*.
  - When *PC* is incremented, a constant 4 has to be added.



- The instruction decoder and control unit is responsible for performing the actions specified by the instruction loaded into *IR*.
- The decoder generates all the control signals in the proper sequence required to execute the instruction specified by the *IR*.
- The registers, the ALU and the interconnecting bus are collectively referred to as the *data path*.
- The control unit that generates the control signals in proper sequence is referred to as the *control path*.



Kinds of Operations

- Transfer of data from one register to another.

```
MOVE   R1, R2          // R1 = R2
```

- Perform arithmetic or logic operation on data loaded into registers.

```
ADD    R1, R2          // R1 = R1 + R2
```

- Fetch the content of a memory location and load it into a register.

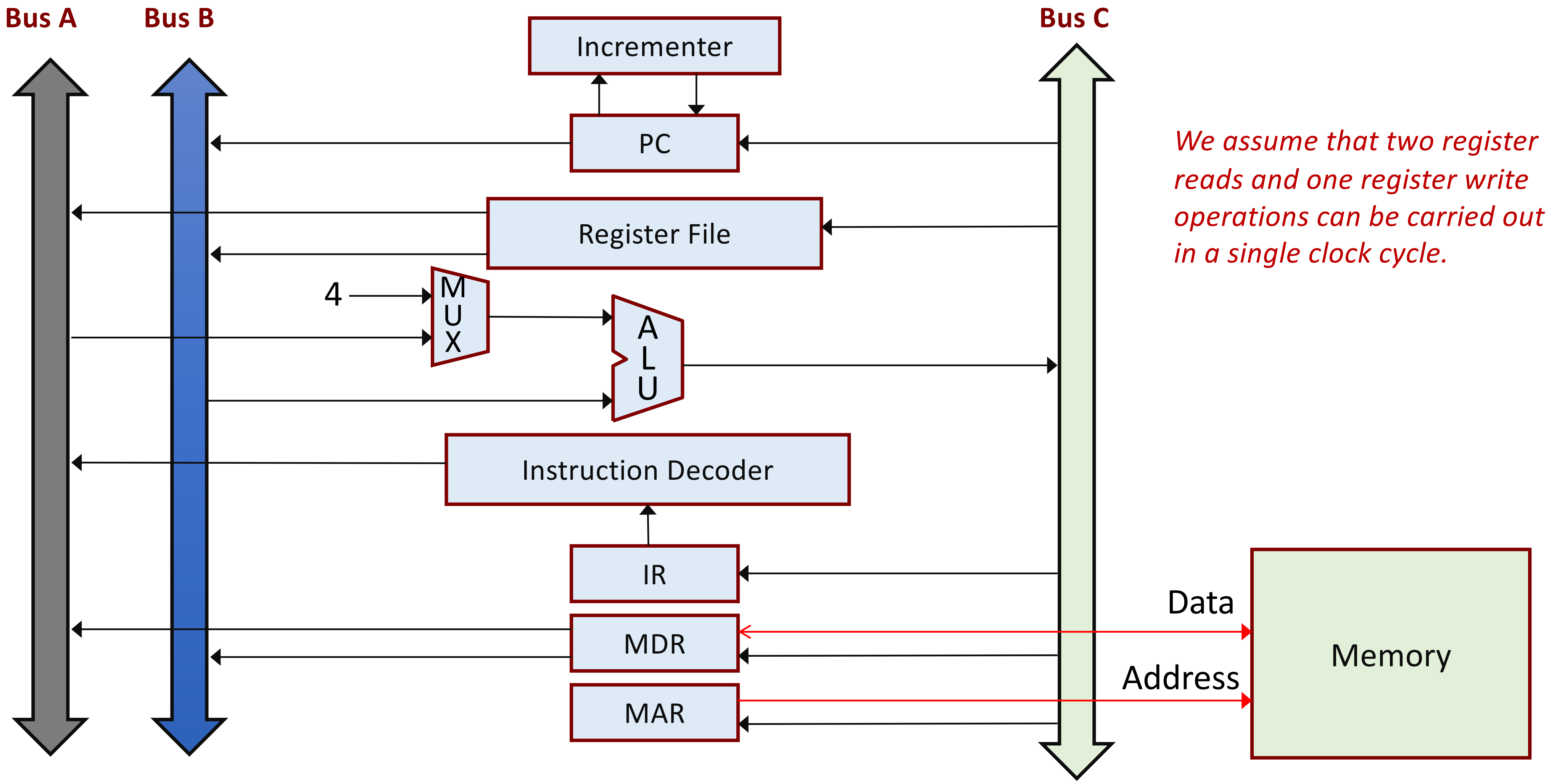
```
LOAD   R1, LOCA        // R1 = Mem[LOCA]
```

- Store a word of data from a register into a given memory location.

```
STOR   LOCA, R1        // Mem[LOCA] = R1
```

Three Bus Organization

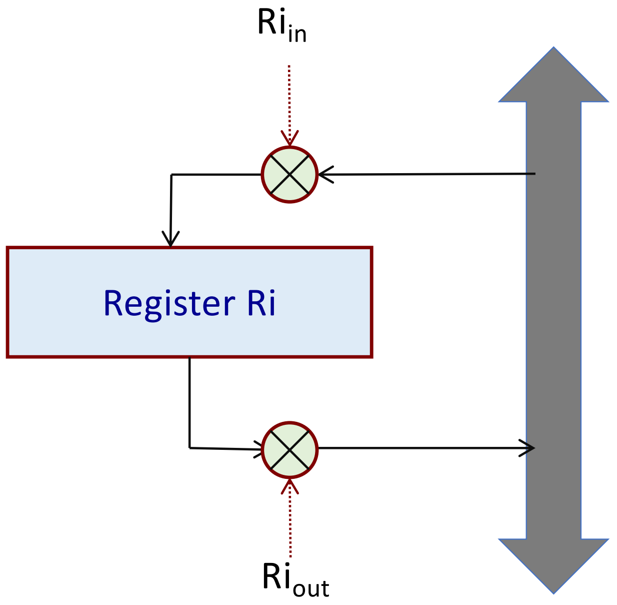
- A typical 3-bus architecture for the processor datapath is shown in the next slide.
    - The 3-bus organization is internal to the CPU.
    - Three buses allow three parallel data transfer operations to be carried out.
- Less number of cycles required to execute an instruction compared to single bus organization.



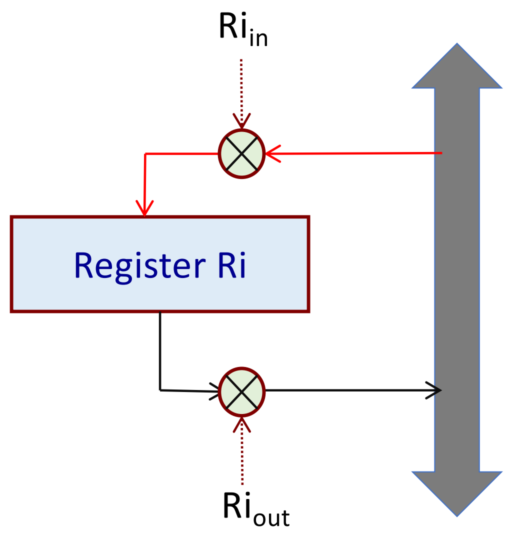
We assume that two register reads and one register write operations can be carried out in a single clock cycle.

Organization of a Register

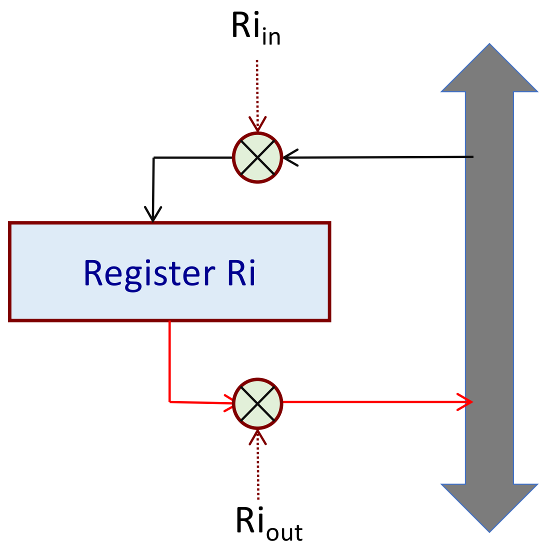
- A register is used for temporary storage of data (parallel-in, parallel-out, etc.).
- A register Ri typically has two control signals.
  - Ri_{in} : used to load the register with data from the bus.
  - Ri_{out} : used to place the data stored in the register on the bus.
- Input and output lines of the register Ri are connected to the bus via controlled switches.
  - If Ri_{out} is not selected, the register outputs are set in the high impedance state.



- When ($Ri_{in} = 1$), the data available on bus is loaded into Ri.



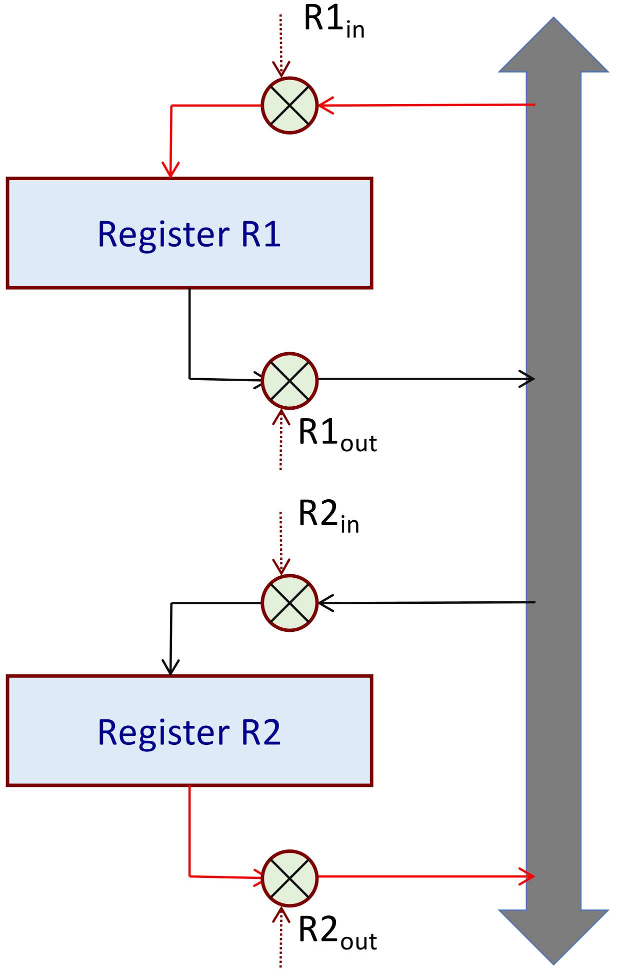
- When ($Ri_{out} = 1$), the data from register Ri are placed on the bus.



Register Transfer

MOVE R1, R2 // $R1 \leftarrow R2$

- Enable the output of R2 by setting $R2_{out} = 1$.
- Enable the input of register R1 by setting $R1_{in} = 1$.
- All operations are performed in synchronism with the processor clock.
  - The control signals are asserted at the start of the clock cycle.
  - After data transfer the control signals will return to 0.
- We write as $\underbrace{T1}_{\text{Time Step}}:\ \underbrace{R2_{out},\ R1_{in}}_{\text{Control Signals}}$



ALU Operation

ADD R1, R2 // R1 = R1 + R2

- Bring the two operands (R1 and R2) to the two inputs of the ALU.

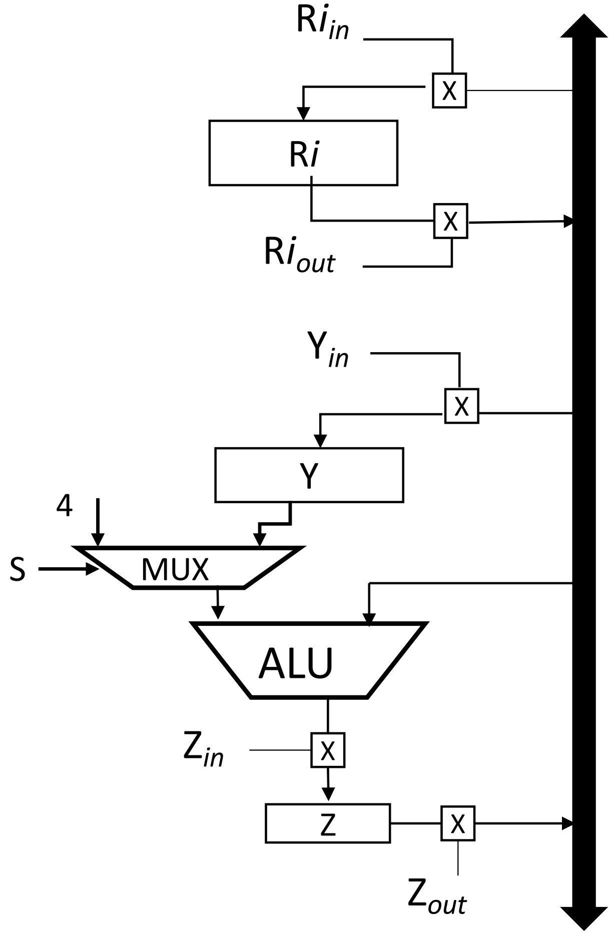
  One through Y (R1) and another (R2) directly from internal bus.

- Result is stored in Z and finally transferred to R1.

  T1: $R1_{out}$, Y_{in}

  T2: $R2_{out}$, SelectY, ADD, Z_{in}

  T3: Z_{out}, $R1_{in}$



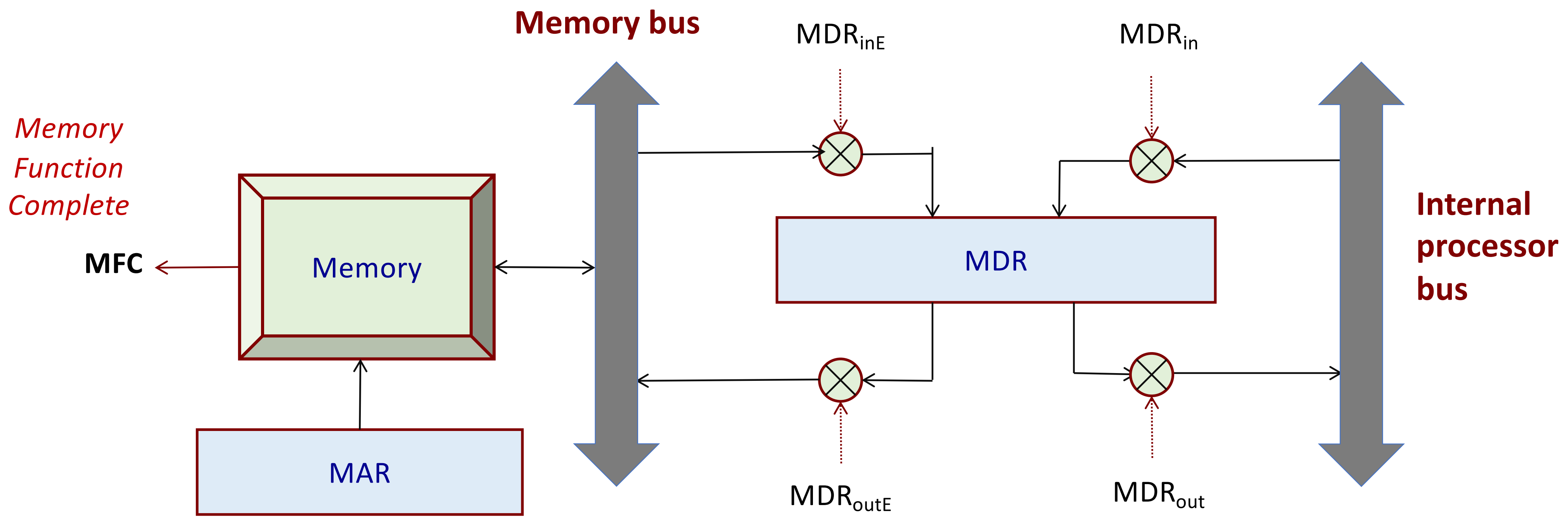
Fetching a Word from Memory

- The steps involved to fetch a word from memory:
  - The processor specifies the address of the memory location where the data or instruction is stored (move to *MAR*).
  - The processor requests a *read* operation.
    - The information to be fetched can either be an instruction or an operand of the instruction.
  - The data read is brought from the memory to *MDR*.
  - Then it is transferred to the required register or ALU for further operation.

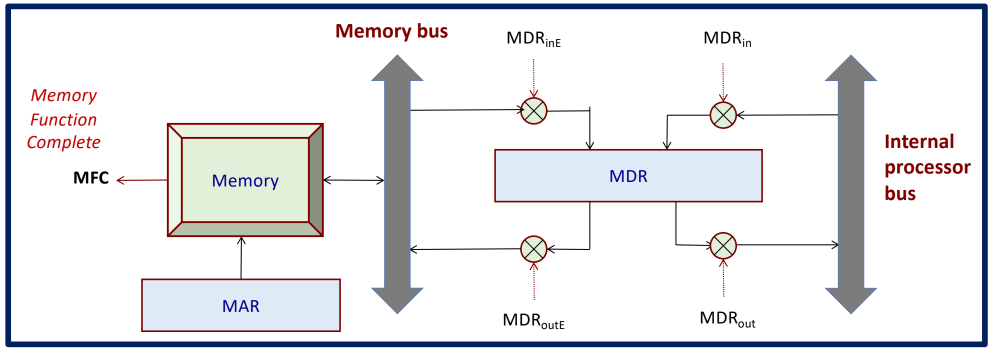
Storing a Word into Memory

- The steps involved to store a word into the memory:
  - The processor specifies the address of the memory location where the data is to be written (move to *MAR*).
  - The data to be written in loaded into *MDR*.
  - The processor requests a *write* operation.
  - The content of *MDR* will be written to the specified memory location.

Connecting MDR to Memory Bus and Internal Bus



- Memory read/write operation:
  - The address of memory location is transferred to *MAR*.
  - At the same time a *read/write* control signal is provided to indicate the operation.
  - For read, the data from memory data bus comes to *MDR* by activating MDR_{inE} .
  - For write, the data from *MDR* goes to memory data bus by activating the signal MDR_{outE} .
  - When the processor sends a read request, it has to wait until the data is read from the memory and written into *MDR*.
  - To accommodate the variability in response time, the process has to wait until it receives an indication from the memory that the read operation has been completed.
  - A control signal called *Memory Function Complete* (MFC) is used for this purpose.
    - When this signal is 1, indicates that the contents of the specified location is read and are available on the data line of the memory bus.
    - Then the data can be made available to *MDR*.



Fetch a word: MOVE R1, (R2)

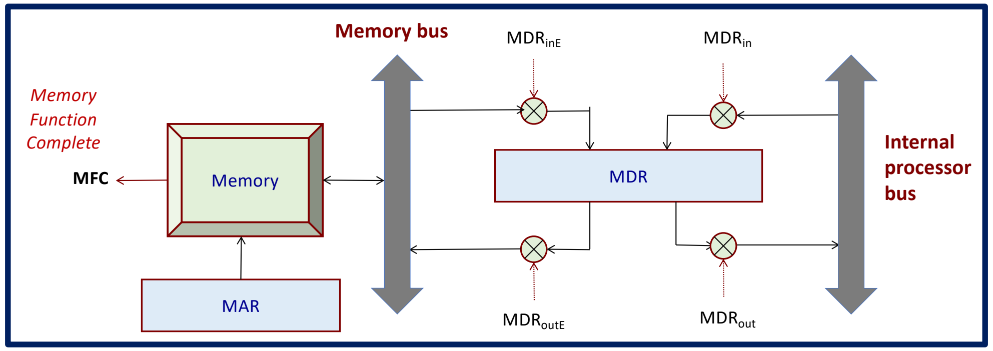
1. MAR ← R2
2. Start a Read operation on the memory bus
3. Wait for the MFC response from the memory
4. Load MDR from the memory
5. R1 ← MDR

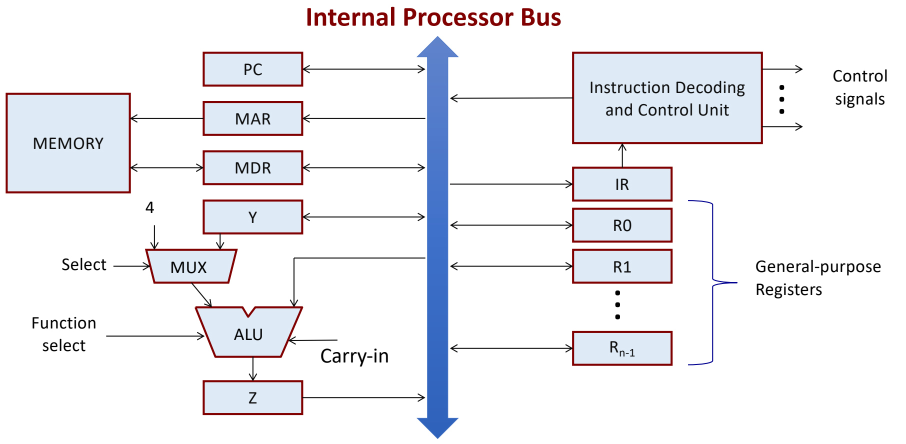
Control steps:

a) $R2_{out}$, MAR_{in}, Read

b) MDR_{inE}, WMFC

c) MDR_{out}, $R1_{in}$





Store a word: MOVE (R1), R2

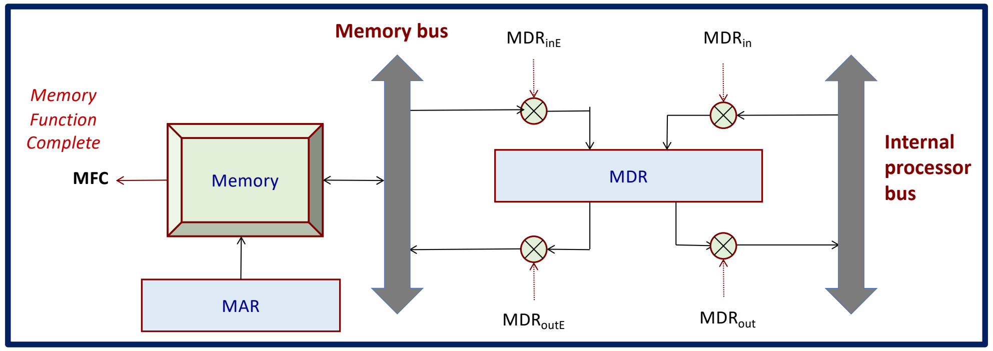
1. MAR ← R1
2. MDR ← R2
3. Start a Write operation on the memory bus
4. Wait for the MFC response from the memory

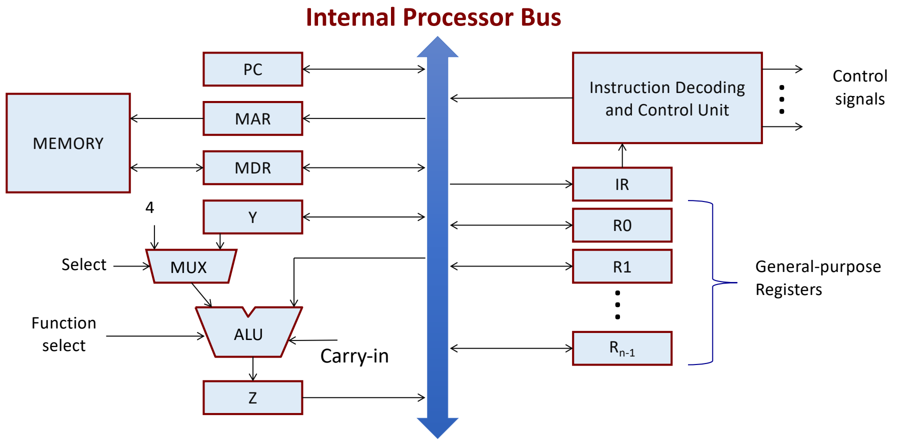
Control steps:

a) $R1_{out}$, MAR_{in}

b) $R2_{out}$, MDR_{in}, Write

c) MDR_{outE}, WMFC





Execution of a Complete Instruction

ADD R1, R2 // R1 = R1 + R2

T1: PC_{out}, MAR_{in}, Read, Select4, ADD, Z_{in}

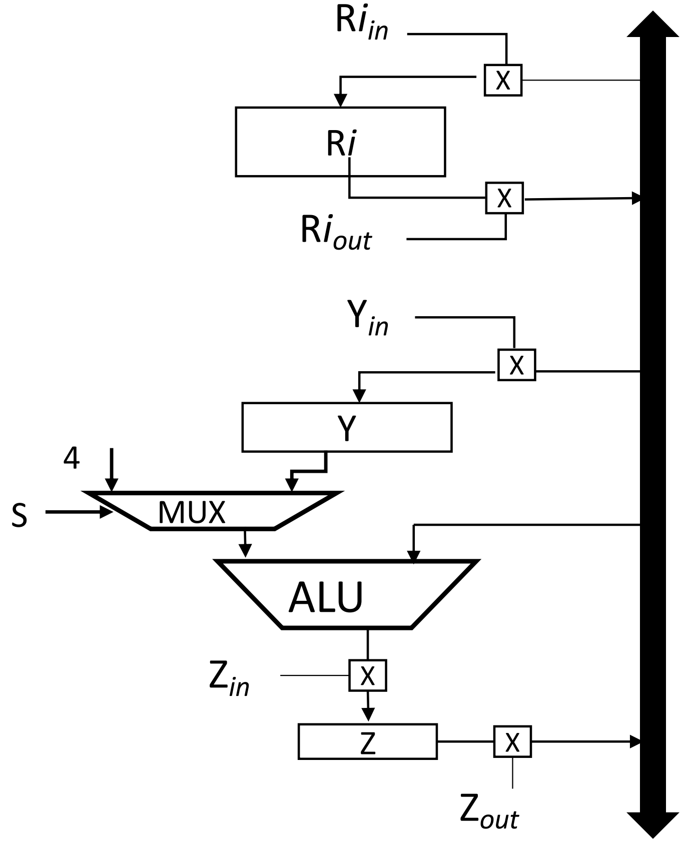
T2: Z_{out}, PC_{in}, Y_{in}, WMFC

T3: MDR_{out}, IR_{in}

T4: $R1_{out}$, Y_{in}, SelectY

T5: $R2_{out}$, ADD, Z_{in}

T6: Z_{out}, $R1_{in}$



Example for a Three Bus Organization

SUB R1, R2, R3 // R1 = R2 – R3

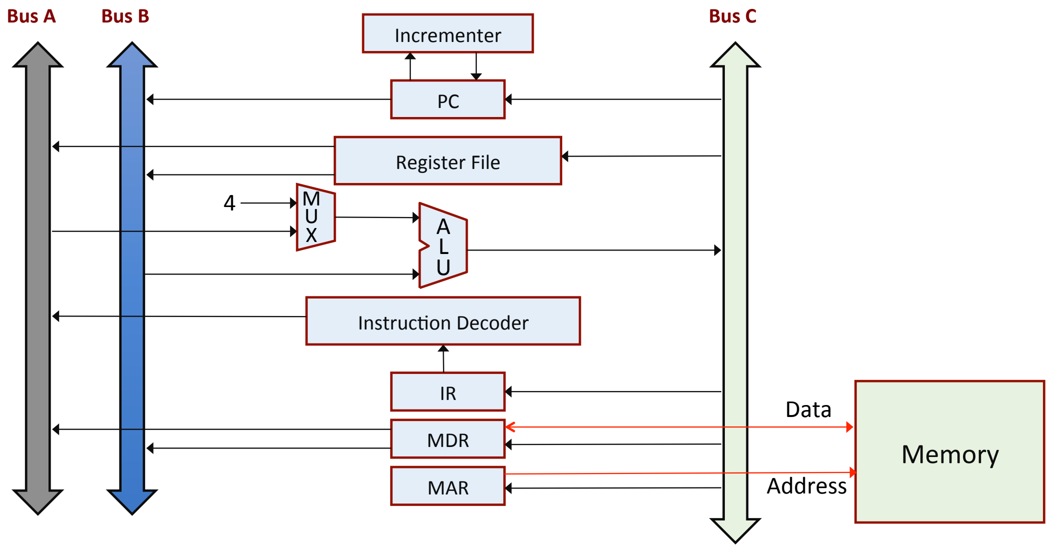
T1: PC_{out}, R = B, MAR_{in} , READ, IncPC

T2: WMFC

T3: MDR_{outB} , R = B, IR_{in}

T4: $R2_{outA}$, $R3_{outB}$, SelectA, SUB, $R1_{in}$, End

R = B means that the ALU function is selected such that data on Bus-B is transferred to the ALU output (i.e., Bus-C).



Bus A
Bus B
Incrementer
Bus C
PC
Register File
4
MUX
ALU
Instruction Decoder
IR
Data
MDR
Memory
MAR
Address