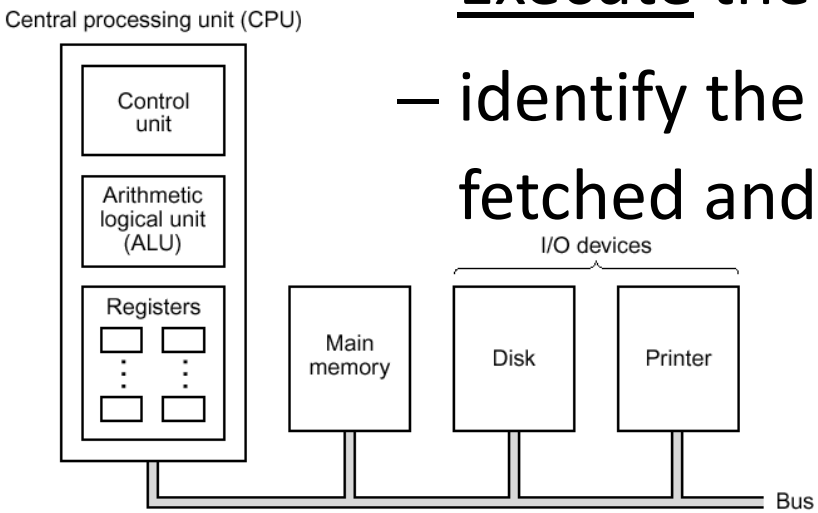
Central Processing Unit - I

Processor Organization (1)

- The aims of the processor are to
  - Fetching an instruction from memory (through the bus);
  - Fetch the instruction (fetch);
  - Execute the instruction (execute);
  - identify the next instruction ready to be fetched and executed



Processor Organization (2)

- The three components of the processor :
  (1) **The *Arithmetic and Logical Unit* (*ALU*)**
    - Made up of circuits that perform the arithmetic and logical execution within the processor.
    - It has no internal storage.
  (2) **The *Control Unit* (*CU*)**
    - It contains circuits that direct and coordinate proper sequence, interpret each instruction and apply the proper signals to the ALU and registers.

Processor Organization (3)

(3) **The *Registers***

- Registers are high speed temporary data storage area within the processor to support execution activities.
- Both instructions or data can be stored in registers for processing by the ALU.
- All processors have a certain number of registers, the exact number varies between different CPUs.

Processor Organization (4)

- The advantages of using registers:
  - Reducing instruction (program) length
    - Memory address usually requires many bits, referencing registers requires only a few bits.
  - Cutting down execution time
    - Using register in complex expressions and storing results instead of writing back to memory, the number of CPU clock cycle (time) can be reduced.

Processor Organization (5)

- Ease of programming
  - In processing a block of data consecutively in memory, one can store starting address of block in register. To retrieve data item, one can simply increment the address in register, corresponding to consecutive data in data block. In doing so, fewer number of instructions are needed.

Types of Registers (1)

– ***Instruction Register (IR)***

- It holds the instruction that is currently being executed.
- Its output is available to the control unit which generated the timing signals that control the various processing elements involved in executing the instruction.

Types of Registers (2)

- ***Program Counter (PC)***
  - It holds the address of the instruction to be fetched and executed.
  - During the execution of an instruction, its contents are updated to point to the next instruction to be executed.
- ***Stack Pointer (SP)***
  - It contains the address of a section of memory known as ***stack*** which may be used for temporary storage of data or addresses.

Types of Registers (3)

– ***Memory Address Register*** (***MAR***)

- It holds the address in memory to or from which data are to be transferred.

– ***Memory Data Register*** (***MDR***)

- It contains the data to be written into or read out of the addressed memory location.

Types of Registers (4)

- ***Status Register (SR) / Conditional Code Register (CCR) / Status Flags***
  - A register with individual bits (flags) to indicate condition of the processor as a result of an arithmetic/logical operations.
  - Common status flags:
    - Carry (C)
    - Positive result (P)
    - Zero result (Z)
    - Negative result (N)
    - arithmetic oVerflow (V)

Program Execution
Fetch and Execute Cycle (1)

- A sequence of instructions (program) to be executed by a computer are first loaded into consecutive memory locations through the input unit.
- Execution of the program starts when the Program Counter (PC) is set to point to the first instruction.
- The CPU fetches one instruction at a time and performs the specified operation.
- The CPU must keep track of the memory address of the next instruction by the PC.
- The cycle is repeated for the next instruction.

Fetch and Execute Cycle (2)

- The processor executes a program in memory is equivalent to execute the program in a series of fetch/execute operations for individual instruction as :
  - (i) At the start of the instruction, the PC contains the address of the instruction to be executed.
  - (ii) Fetch the instruction pointed to by the PC from memory into Instruction Register (IR).
  - (iii) Update PC to point to the following instruction.
  - (iv) Decode the instruction by the control unit (CU).

Fetch phase ends here.

Fetch and Execute Cycle (3)

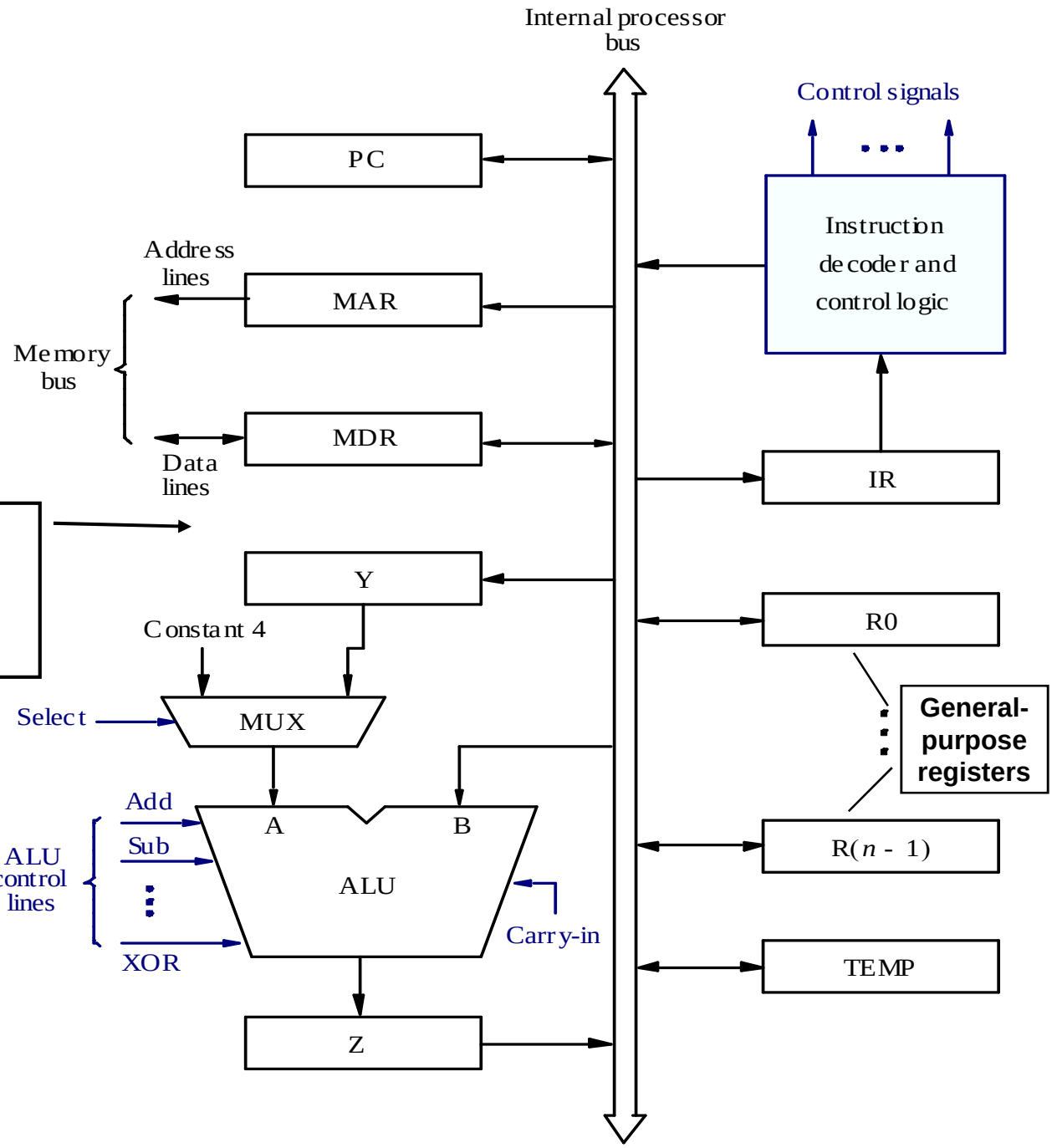
- (v) If the instruction uses data in memory, determine where they are.
- (vi) Fetch data, if any, into registers.
  - (it may be necessary to repeat fetching data, depending on the instruction)
- (vii) Execute the instruction.
- (viii) Store the results in place as specified by the instruction.
- (ix) Go to (i) and execute next instruction.

Fetch and Execute Cycle (4)

- The ***fetch cycle*** (to fetch & decode instruction) is the same for all types of instructions
- The ***execute cycle*** is determined by the instruction fetched from memory in the fetch cycle.

Single Bus Organization inside a Processor (1)

Single-bus organization



Single Bus Organization (2)

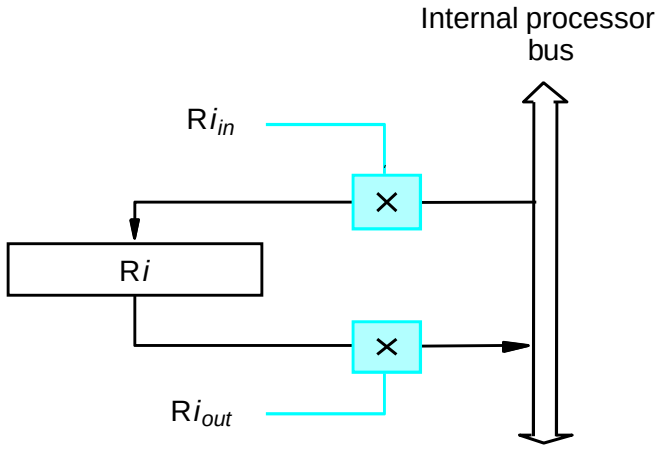
- The data and address lines of the external memory bus are connected to the internal processor bus via the memory data register, MDR, and the memory address register, MAR, respectively.
- The MUX selects either the output of register Y or a constant value 4 (it is assumed that each instruction is 4 bytes and this constant is used to increment the contents of the PC) to be provided as input A of the ALU.

Single Bus Organization (3)
Functions of the CPU

- Most of the operations needed to execute an instruction can be carried out by performing the following functions in some specified sequence.
  - Transfer a word of data from one register to another or to the ALU.
  - Perform an arithmetic or logic operation and store the result in a register.
  - Fetch the contents of a given memory location and load them into a register.
  - Store a word of data from a register into a given memory location.

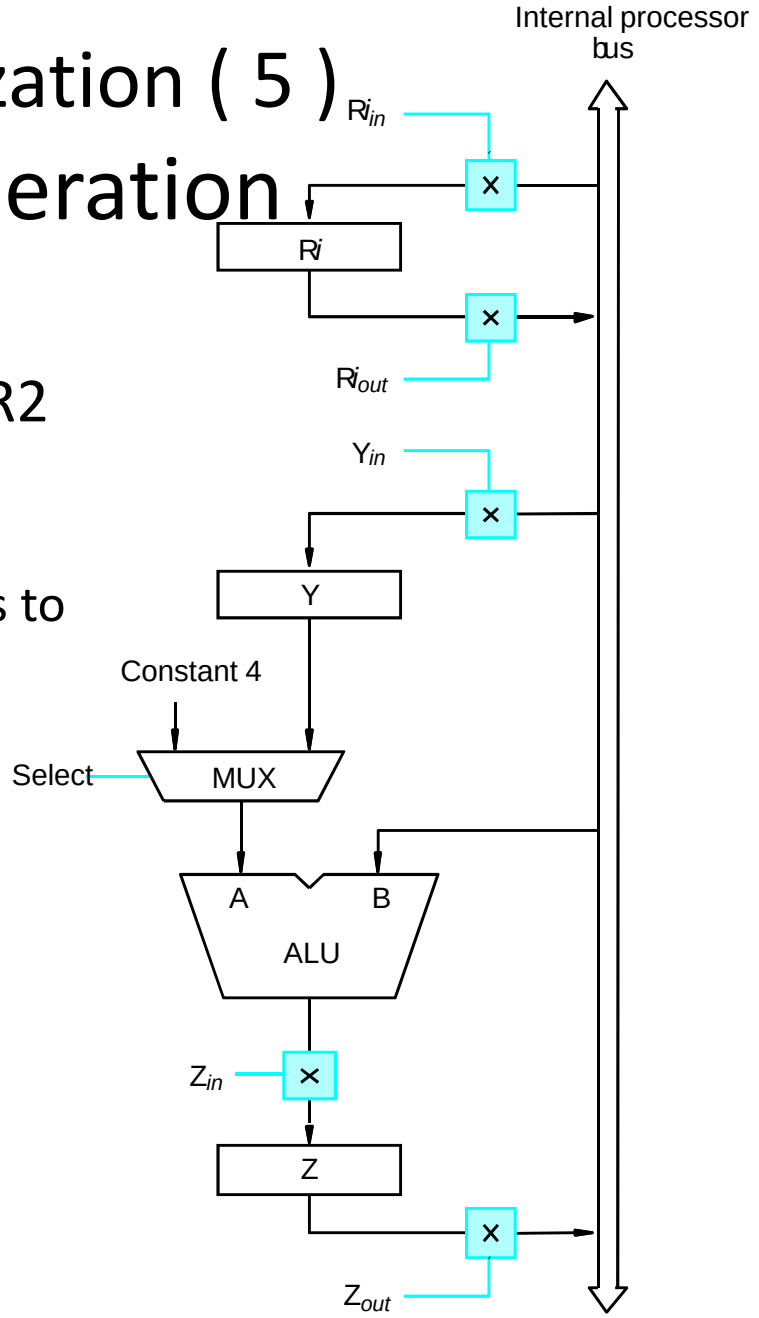
Single Bus Organization (4)
Register Transfers

- To transfer the contents of R1 to R4, the following actions are needed.
  - Enable the output of R1 by setting $R1_{out}$ to 1. This places the contents of R1 on the CPU bus.
  - Enable the input of R4 by setting $R4_{in}$ to 1. This loads data from the CPU bus into register R4.



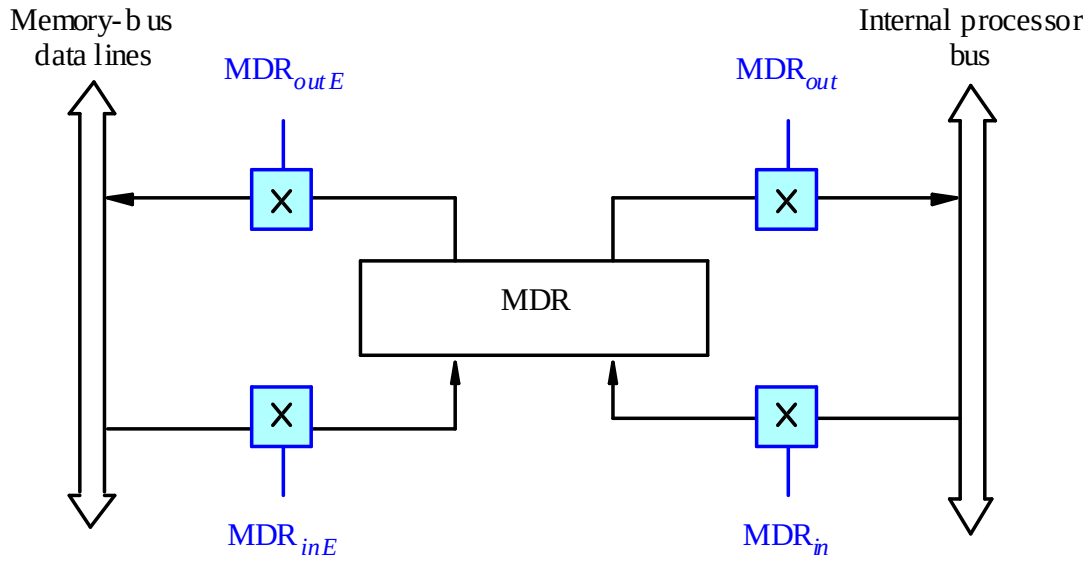
Single Bus Organization (5) Performing Operation

- To add the contents of R1 to that of R2 and store the result in R3 may be as follows. (Names are given for those signals to be activated.)
  - 1. $R1_{out}$, Y_{in}
  - 2. $R2_{out}$, SelectY, Add, Z_{in}
  - 3. Z_{out}, $R3_{in}$



Single Bus Organization (6)
Fetching from Memory

- To load the data pointed by R1 into R2.
- Assume that MAR_{out} is enabled all the time.
  - $R1_{out}$, MAR_{in}, Read
  - MDR_{inE}, WMFC
  - MDR_{out}, $R2_{in}$



Single Bus Organization (7)
Fetching from Memory

- During memory Read and Write operations, the timing must be coordinated with the (different) response of the addressed device on the memory bus.
- The WMFC (Wait for Memory-Function-Completed) signal causes the processor waits for the arrival of the MFC signal.
- The MFC signal is set by the addressed device when the contents of the specified location have been read and are available on the data lines of the memory bus.

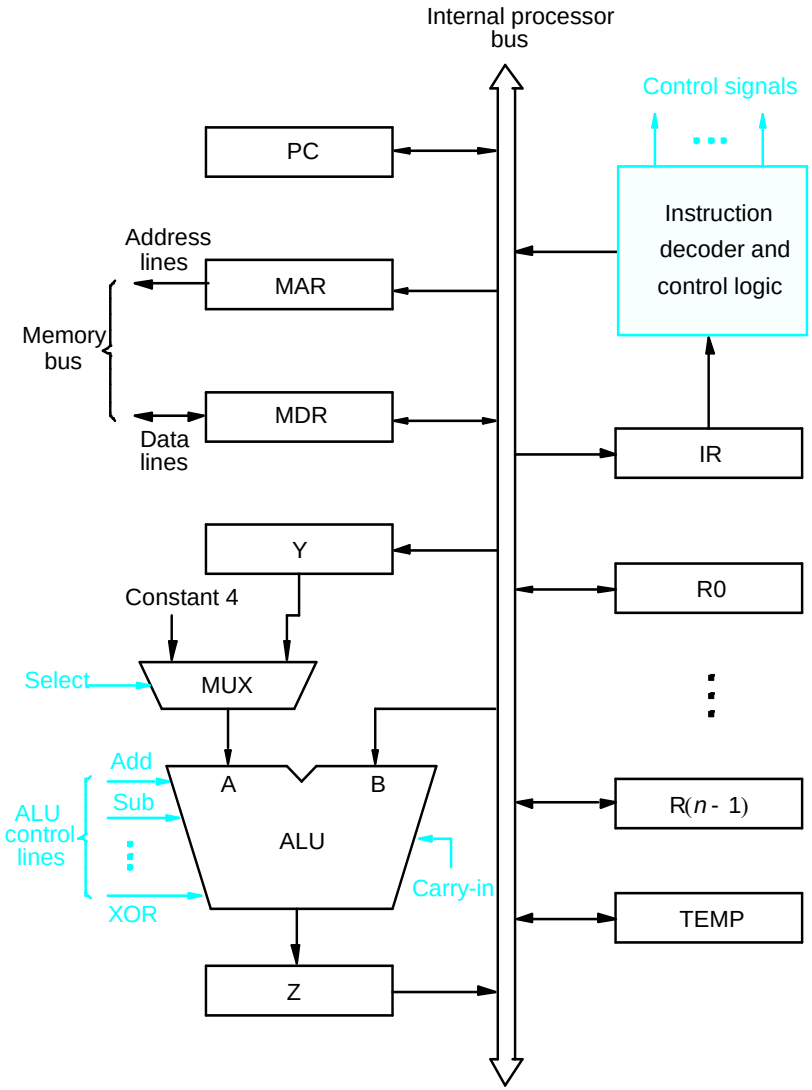
Single Bus Organization (8)
Storing in Memory

- To store the contents of R2 into the memory location specified by R1.
  - $R1_{out}$, MAR_{in}
  - $R2_{out}$, MDR_{in}, Write
  - MDR_{outE}, WMFC

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}, 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



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.
- Conditional branch

Execution of Unconditional Branch Instruction

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

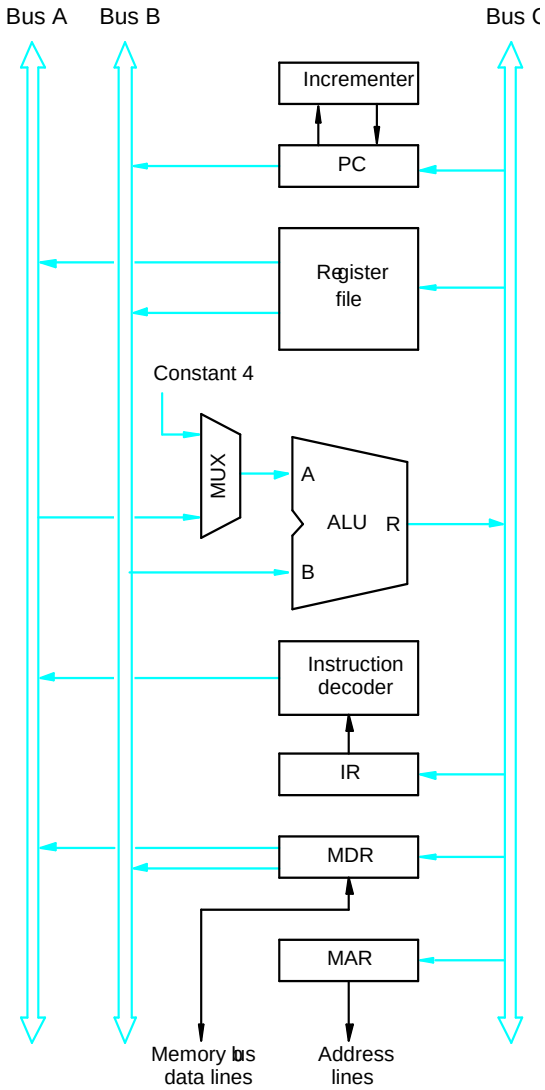
Control sequence for an unconditional branch instruction.

Execution of Conditional Branch Instruction – Branch on Negative (BN)

Step	Action
1	PC_{out}, MAR $_{in}$, Read, Select4, Add, Z_{in}
2	Z_{out}, PC_{in}, WMFC
3	MDR_{out}, IR_{in}
4	Offset-field-of-IR_{out}, Add, Z_{in}, If N=0, End
5	Z_{out}, PC_{in}, End

Control sequence for an conditional branch instruction.

Multiple-Bus Organization



Multiple-Bus Organization

- Add R4, R5, R6

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

Control sequence for the instruction. Add R4,R5,R6, for the three-bus organization