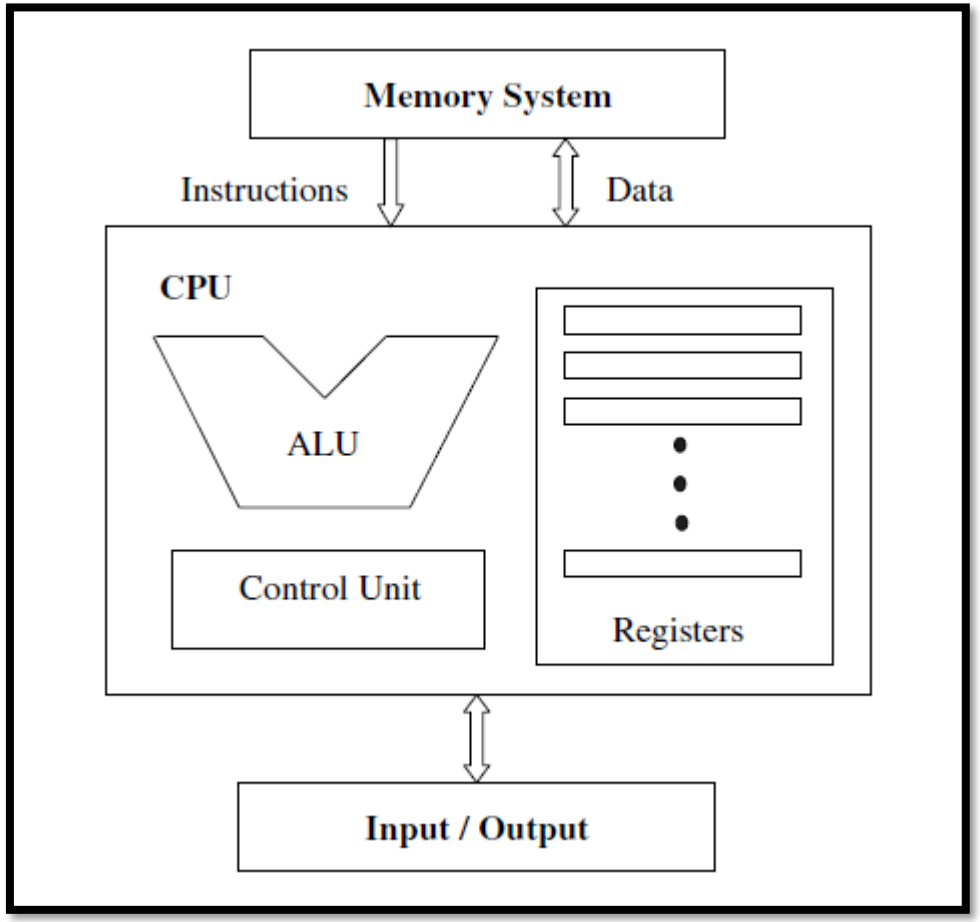
Computer System Architecture
UNIT - 3
Processor System

Syllabus

- Introduction to Processor System
- Register Files
- Datapath Design
- Complete Instruction Execution
- Control Unit
- Hardwired Control
- Microprogrammed Control

Introduction to Processor System

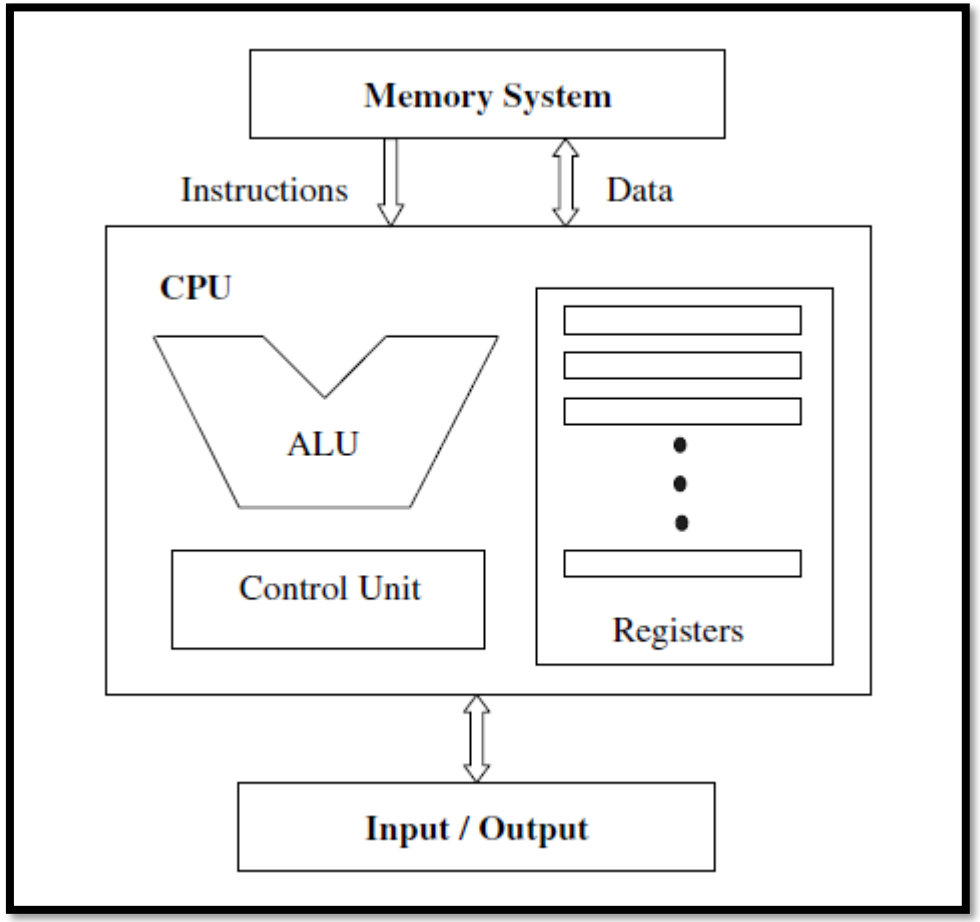
- CPU: Main component of any computer system
- Primary function: Execute a set of instructions stored in the computer's memory
- CPU consists of 3 major components
  - a set of ***Registers***
  - an ***Arithmetic Logic Unit (ALU)***
  - a ***Control Unit (CU)***



Introduction to Processor System

Register Set

- May differ from computer to computer
- Usually combination of
  - General-purpose registers
    - Can be used for any task/operation
  - Special-purpose registers
    - Dedicated to be used for specific task
    - PC: Hold the address of the instruction to be executed next
    - IR: Hold the instruction that is currently executed
    - Memory Access Registers : MAR & MDR:



Introduction to Processor System

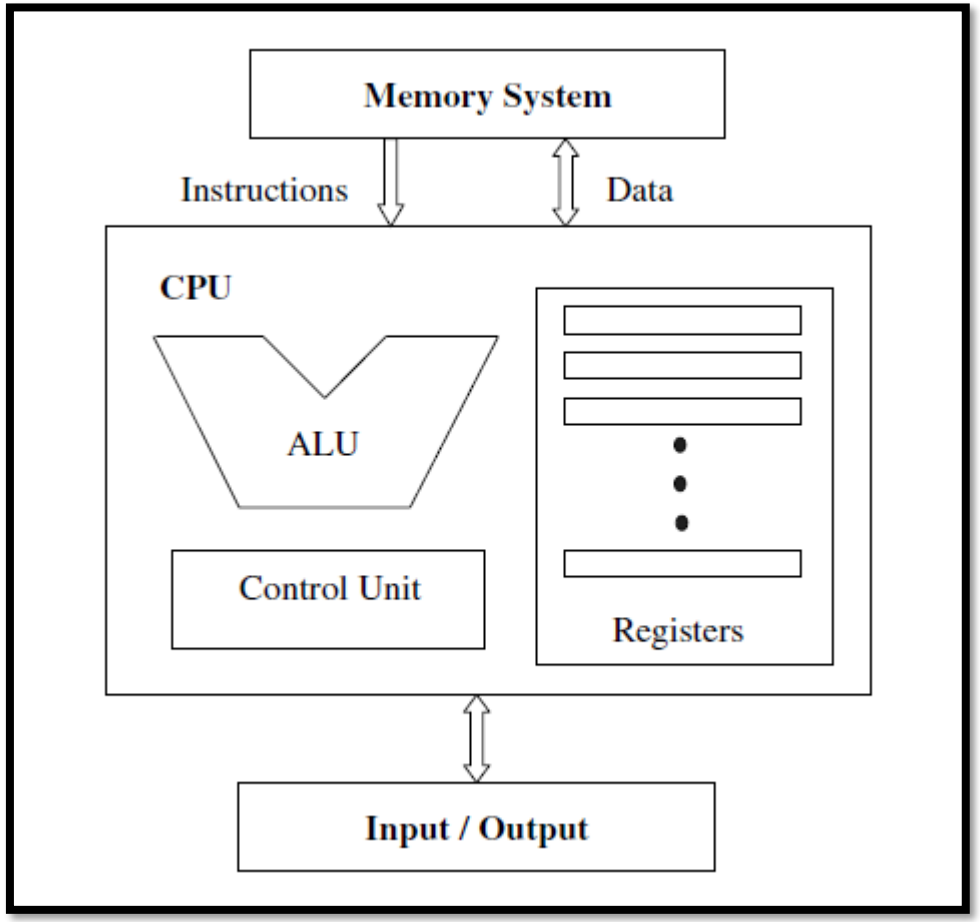
- **ALU**

  Provides the circuitry needed to perform the arithmetic, logical and shift operations demanded by the instruction set

- **CU**

  Responsible for fetching the instruction to be executed from the main memory and decoding and then executing it

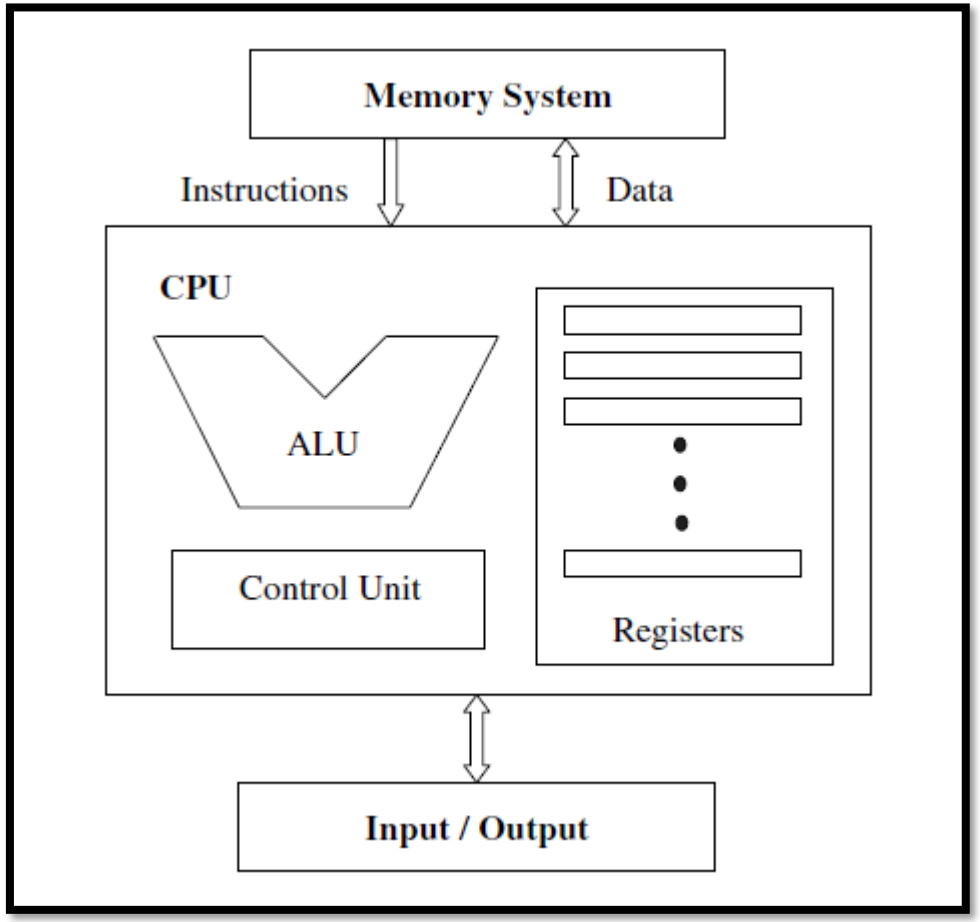
CPU fetches instructions from memory, reads and writes data from/to memory, and transfers data from/to input/output devices.



Introduction to Processor System

Steps involved in a simple execution cycle can be summarized as follows:

1. The next instruction to be executed, whose address is obtained from the PC, is fetched from the memory and stored in the IR
2. The instruction is decoded
3. Operands are fetched from the memory and stored in CPU registers, if needed
4. The instruction is executed
5. Results are transferred from CPU registers to the memory, if needed



Register Files

- Registers are essentially extremely fast memory locations within the CPU that are used to create and store the results of CPU operations and other calculations
- Different computers have different register sets
  - Differ in the number of registers, register types, and the length of each register
  - Differ in the usage of each register: GPR, SPR
  - Data registers are used only to hold data and cannot be used in the calculations of operand addresses
    - length of a data register must be long enough to hold values of most data types
  - Address registers may be dedicated to a particular addressing mode or may be used for general purpose addressing
    - must be long enough to hold the largest address

Register Files

- Number of registers in a particular architecture affects the instruction set design
- A very small number of registers may result in an increase in memory references
- Another type of registers is used to hold processor status bits, or flags
  - These bits are set by the CPU as the result of the execution of an operation
  - The status bits can be tested at a later time as part of another operation

Register Files

Memory Access Registers

- Exclusively used by the CPU and are not directly accessible to programmers
- Type: Memory Address Register (MAR) & Memory Data Register (MDR)
- Memory *write* operation
  - The *word* to be stored into the memory location is first loaded by the CPU into MDR
  - *Address* of the location into which the word is to be stored is loaded by the CPU into MAR
  - A write signal is issued by the CPU.
- Memory *read* operation
  - *Address* of the location from which the word is to be read is loaded into the MAR
  - A read signal is issued by the CPU.
  - The required word will be loaded by the memory into the MDR ready for use by the CPU.

Register Files

Instruction Fetching Registers

- Two main registers involved in fetching an instruction for execution:
  - Program Counter (PC)
  - Instruction Register (IR)
- PC contains the address of the next instruction to be fetched
- Fetched instruction is loaded in the IR for execution
- PC is updated to point to the next instruction to be executed

Register Files

Condition Registers

- Also known as *flags*
- Used for maintaining status information
- Bit set by the CPU to indicate the current status of an executing program
- Typically used for arithmetic operations, interrupts, memory protection information, or processor status

Register Files

- **Special-Purpose Address Registers**
  - Index Register
  - Segment Pointers
  - Stack Pointer
- **80x86 Registers**
- **MIPS Registers**

Register Files

- Most modern CPUs have a set of general-purpose registers(GPRs) R_0 to R_{m-1} called Register File (RF)
- Each register R_i in RF is individually addressable with address subscript '*i*'. Ex: $R_0, R_1, R_2, \ldots, R_{m-1}$
- In this way, processor is able to store the intermediate results in fast and easily accessible registers rather than storing them in external memory M
- RF performs the function of a small RAM and uses a fast RAM technology
- In most cases, RF requires 2 or more operands to be accessed simultaneously
- Hence, RF needs several ports for simultaneous reading from or writing into several registers
- Therefore, RF is often realized as a *multi-port RAM*
- Multi-port RF is built using a set of registers of proper size and multiplexer-demultiplexor
- Read operation can take place through several devices reading from same register using different ports
- Writing is normally done through one port only

Datapath Design

- CPU can be divided into a data section and a control section
- Data section is also called as datapath
  - Contains the registers and the ALU
  - Datapath is capable of performing certain operations on data items
- Control section is basically the control unit
  - Issues control signals to the datapath

Datapath Design

Data movements

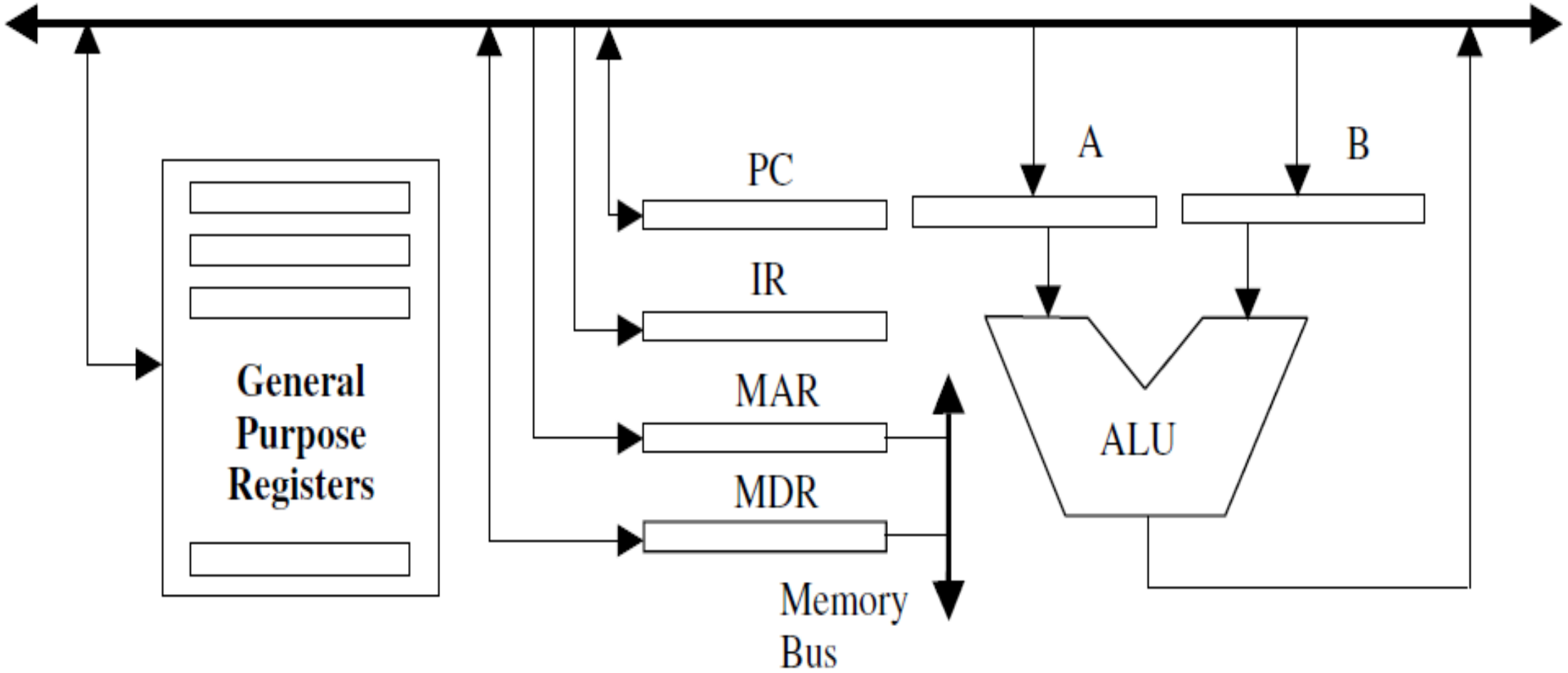
- Internal to the CPU, data move from one register to another and between ALU and registers
- Internal data movements are performed via local buses, which may carry data, instructions, and addresses
- Externally, data move from registers to memory and I/O devices, often by means of a system bus
- Internal data movement among registers and between the ALU and registers may be carried out using different organizations including one-bus, two-bus, or three-bus organizations
- Dedicated datapaths may also be used between components that transfer data between themselves more frequently
    - For example, the contents of the PC are transferred to the MAR to fetch a new instruction at the beginning of each instruction cycle
    - Hence, a dedicated datapath from the PC to the MAR could be useful in speeding up this part of instruction execution

Datapath Design

One-Bus Organization

- CPU registers and the ALU use a single bus to move outgoing and incoming data
- Since a bus can handle only a single data movement within one clock cycle, two-operand operations will need two cycles to fetch the operands for ALU
- Additional registers may also be needed to buffer data for the ALU
- This bus organization is the simplest and least expensive
- But it limits the amount of data transfer that can be done in the same clock cycle, which will slow down the overall performance.

Datapath Design (One-Bus Organization)

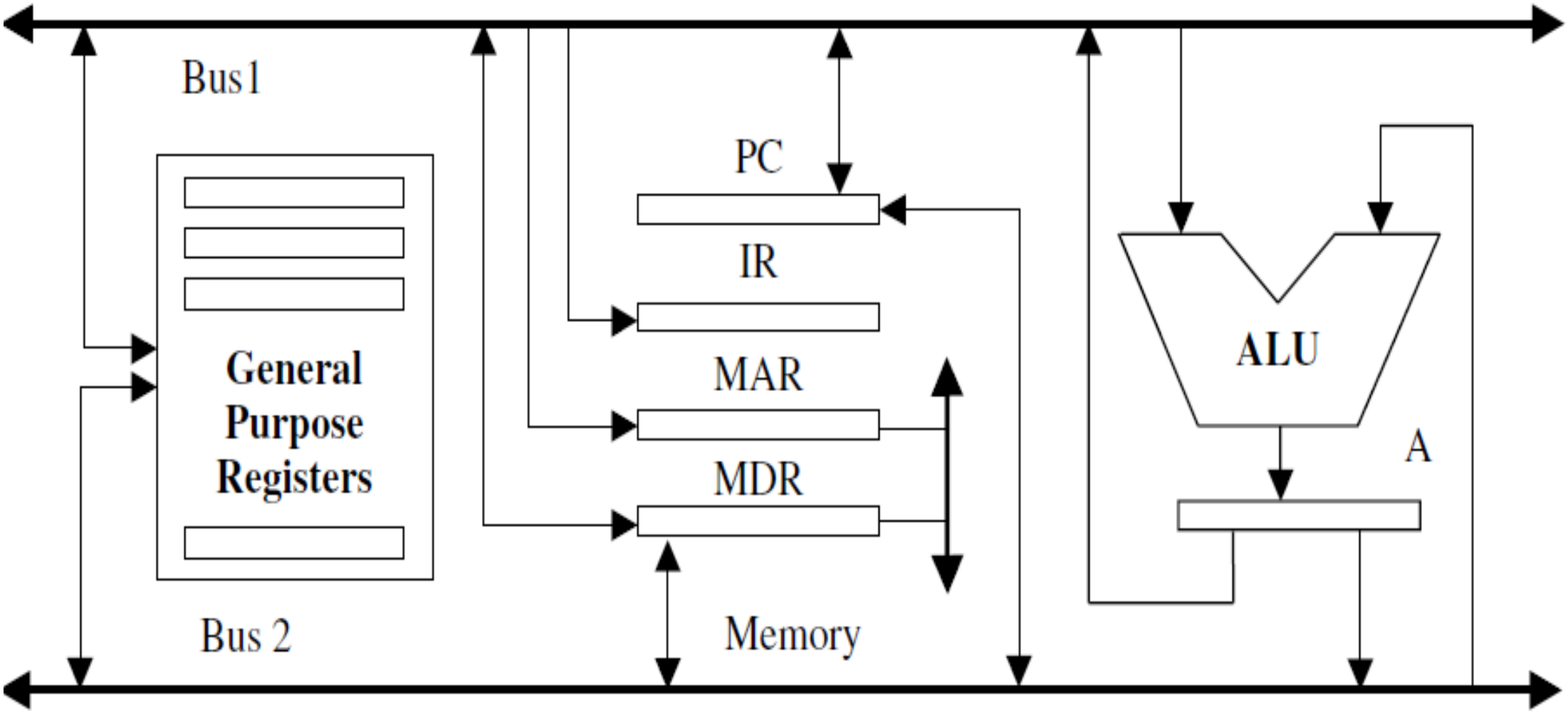


Datapath Design

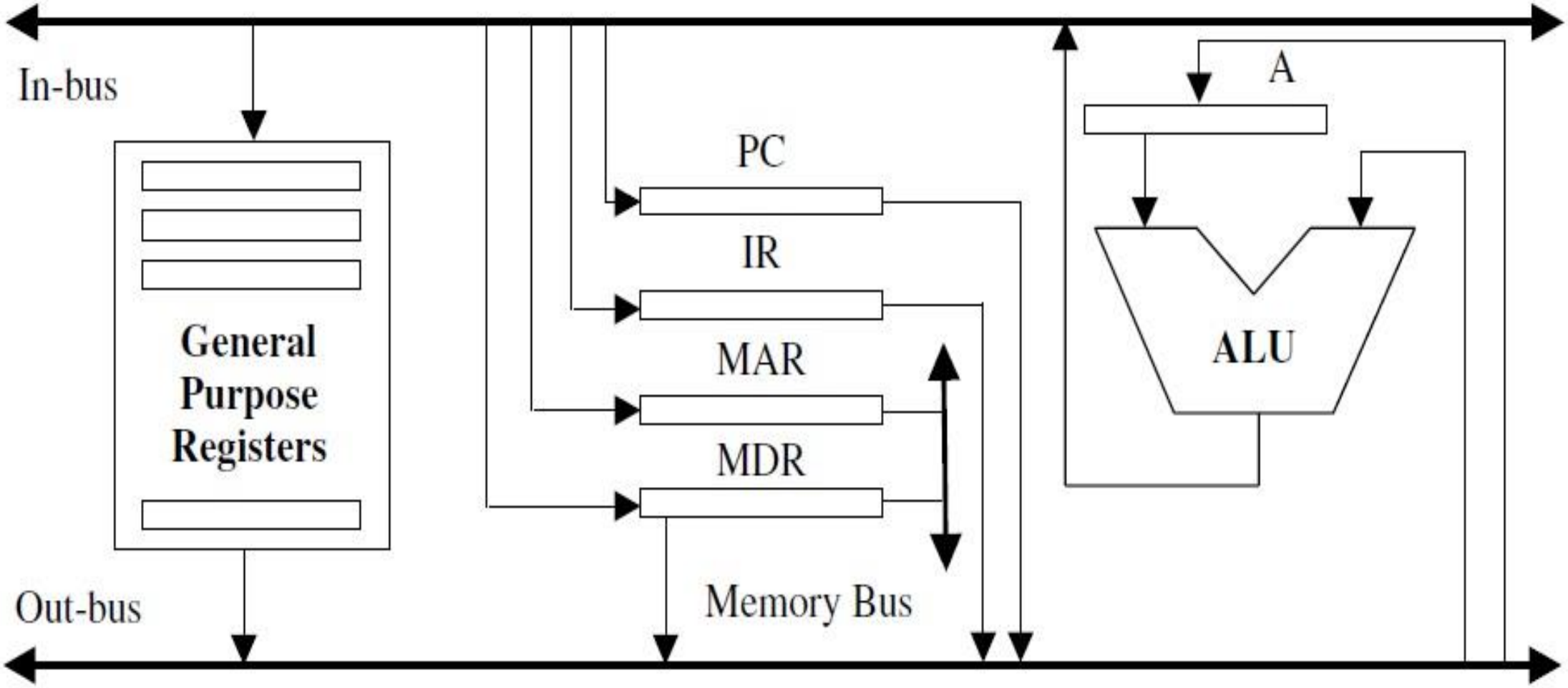
Two-Bus Organization

- Using two buses is a faster solution than the one-bus organization
- General-purpose registers are connected to both buses
- Data can be transferred from two different registers to the input point of the ALU at the same time.
- Therefore, a two-operand operation can fetch both operands in the same clock cycle
- An additional buffer register may be needed to hold the output of the ALU when the two buses are busy carrying the two operands
- In some cases, one of the buses may be dedicated for moving data into registers(in-bus), while the other is dedicated for transferring data out of the registers(out-bus).
- In this case, the additional buffer register may be used, as one of the ALU inputs, to hold one of the operands
- ALU output can be connected directly to the in-bus, which will transfer the result into one of the registers

Datapath Design (Two-Bus Organization)



Datapath Design (Two-Bus Organization)

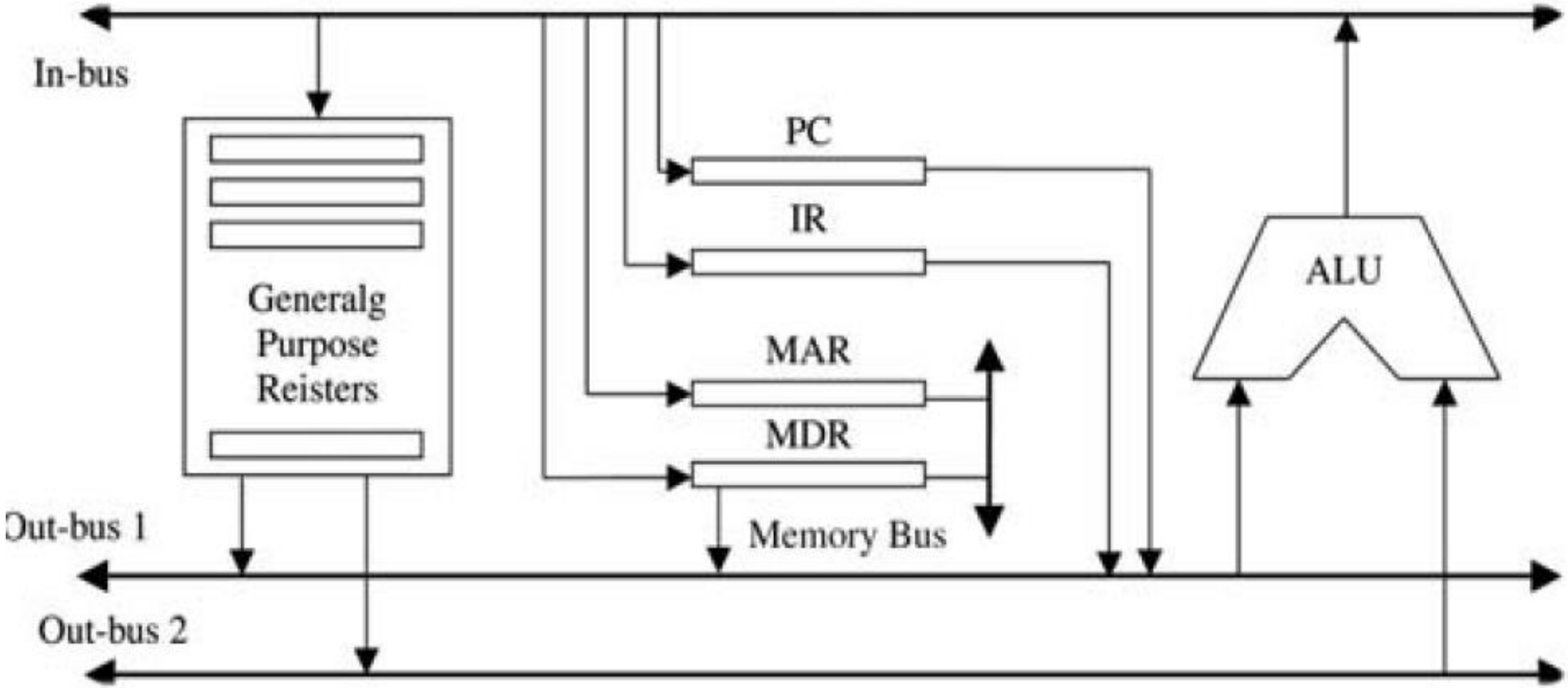


Datapath Design

Three-Bus Organization

- In a three-bus organization, two buses may be used as source buses while the third is used as destination
- Source buses move data out of registers (out-bus), and the destination bus may move data into a register (in-bus)
- Each of the two out-buses is connected to an ALU input point
- Output of the ALU is connected directly to the in-bus
- More the buses we have, the more data we can move within a single clock cycle
- However, increasing the number of buses will also increase the complexity of the hardware

Datapath Design (Three-Bus Organization)



Datapath Design

Interrupt Handling

- After execution of instruction, a test is performed to check for pending interrupts
- If there is an interrupt request waiting, the following steps take place:
  1. The contents of PC are loaded into MDR (to be saved)
  2. The MAR is loaded with the address at which the PC contents are to be saved
  3. The PC is loaded with the address of the first instruction of the interrupt handling routine
  4. The contents of MDR (old value of the PC) are stored in memory

Complete Instruction Execution

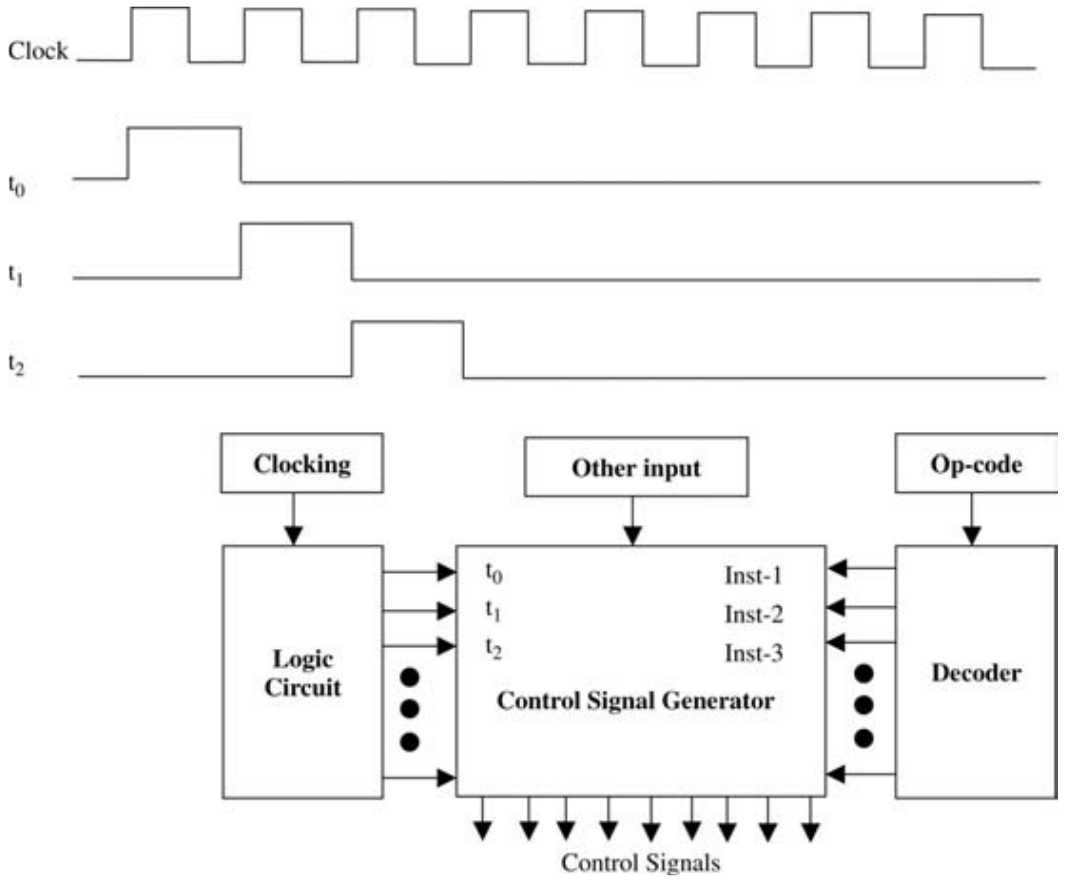
CPU executes each instruction in a series of small steps:

1. Fetch the next instruction from memory to the instruction register (IR)
   a. Contents of the PC are loaded into the MAR
   b. As a result of a memory read operation, the instruction is loaded into the MDR
   c. Contents of the MDR are loaded into the IR.
2. Change the program counter (PC) to point to the next instruction
3. Decode the instruction, just fetched
4. If the instruction uses data in memory, determine where they are
5. Fetch the data if any, into internal CPU register
6. Execute the instruction
7. Store the result in the appropriate place
8. Go to step-1 to execute the next instruction

- This sequence of steps(microoperation) is frequently referred to as **fetch-decode-execute cycle** or **instruction cycle**
- During an instruction cycle, the action of the CPU is defined by the sequence of microoperation it executes
- The time required by the CPU to execute a microoperation is the CPU cycle time or clock period

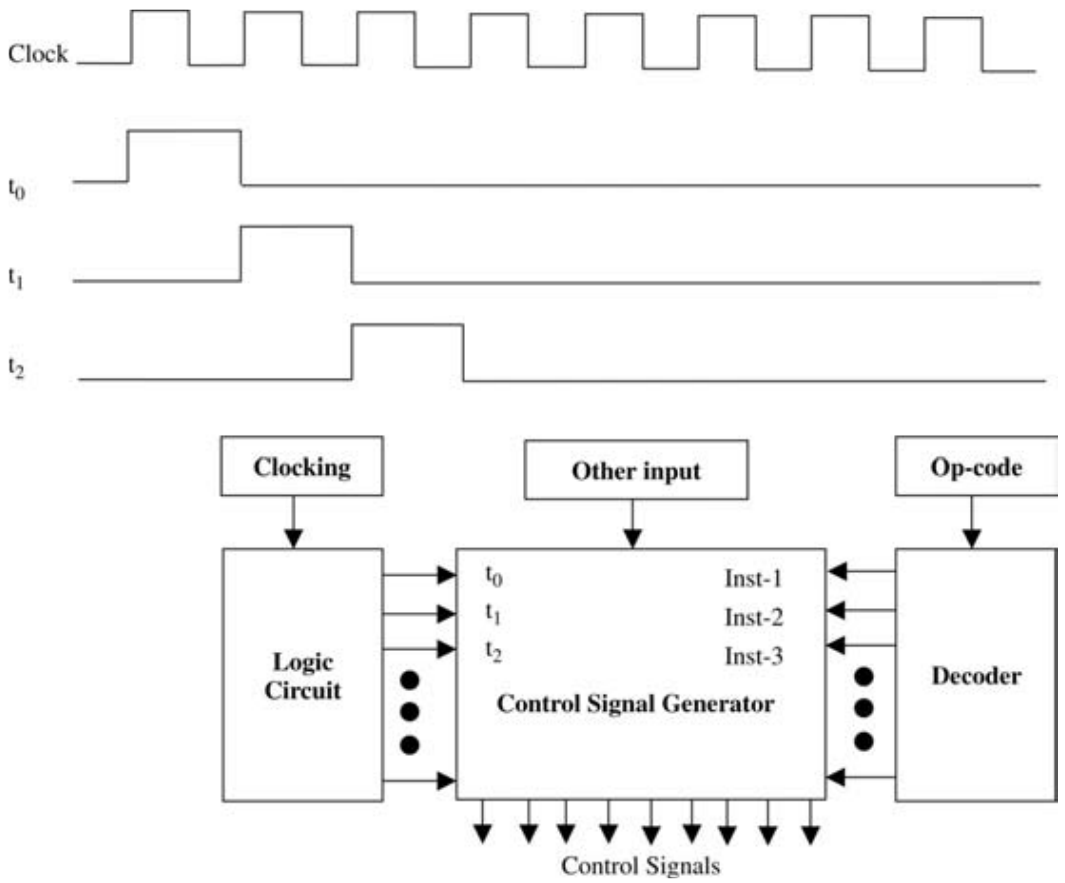
Control Unit

- The control unit is the main component that directs the system operations by sending control signals to the datapath
- These signals control the flow of data within the CPU and between the CPU and external units such as memory and I/O
- Control buses generally carry signals between the control unit and other computer components in a clock-driven manner
- The system clock produces a continuous sequence of pulses in a specified duration and frequency. A sequence of steps t0 , t1 , t2 , . . . , (t0 , t1 , t2 , . . .) are used to execute a certain instruction



Control Unit

- The op-code field of a fetched instruction is decoded to provide the control signal generator with information about the instruction to be executed
- Step information generated by a logic circuit module is used with other inputs to generate control signals
- The signal generator can be specified simply by a set of Boolean equations for its output in terms of its inputs



Control Unit

1-Bus Organization

t_0 - $A \leftarrow (R_1)$

t_1 - $B \leftarrow (R_2)$

t_2 - $R_0 \leftarrow (A) + (B)$

2-Bus Organization

t_0 - $A \leftarrow (R_1) + (R_2)$

t_1 - $R_0 \leftarrow (A)$

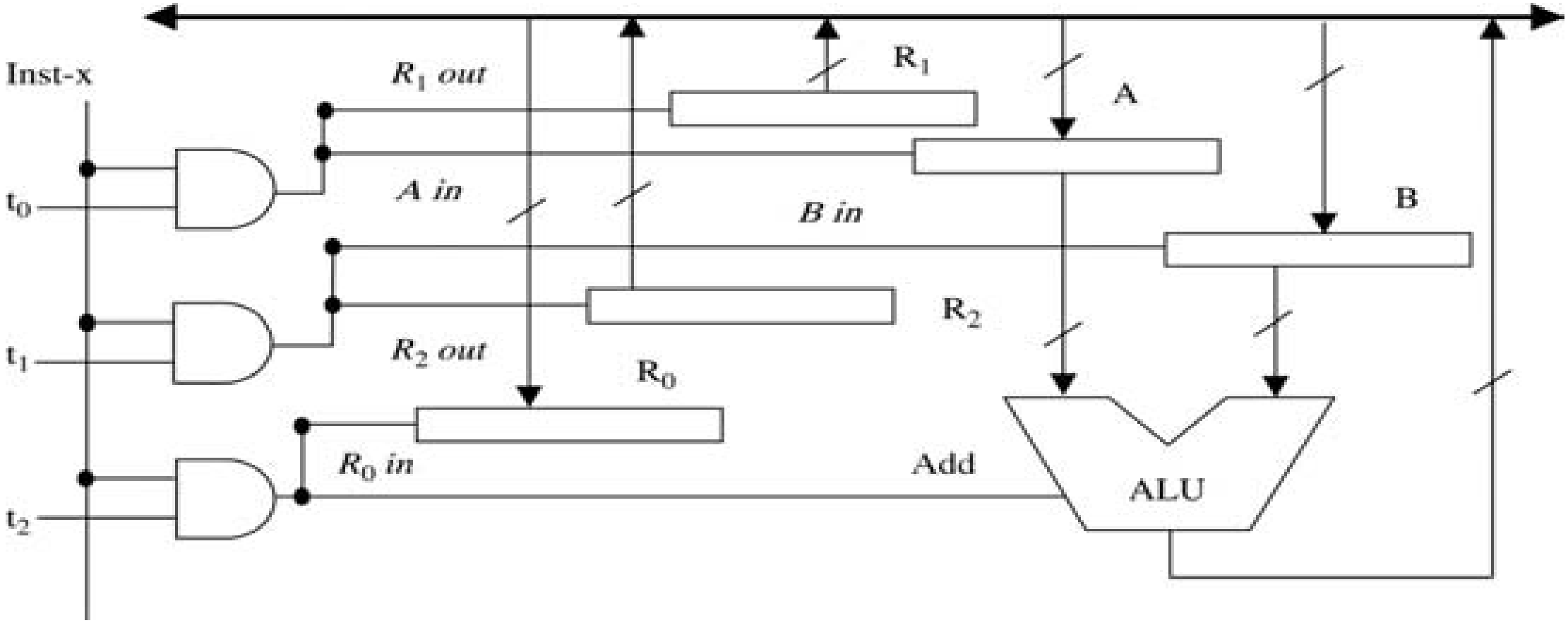
3-Bus Organization

t_0 - $R_0 \leftarrow (R_1) + (R_2)$

Step	Instruction Type	Micro-operation	Control (3-Bus Organization)
t_0	Inst-x	$R_0 \leftarrow (R_1) + (R_2)$	Select R1 as source 1 on out-bus1 (R1 out-bus1) Select R2 as source 2 on out-bus2 (R2 out-bus2) Select R0 as destination on in-bus (R0 in-bus) Select the ALU function Add (Add)

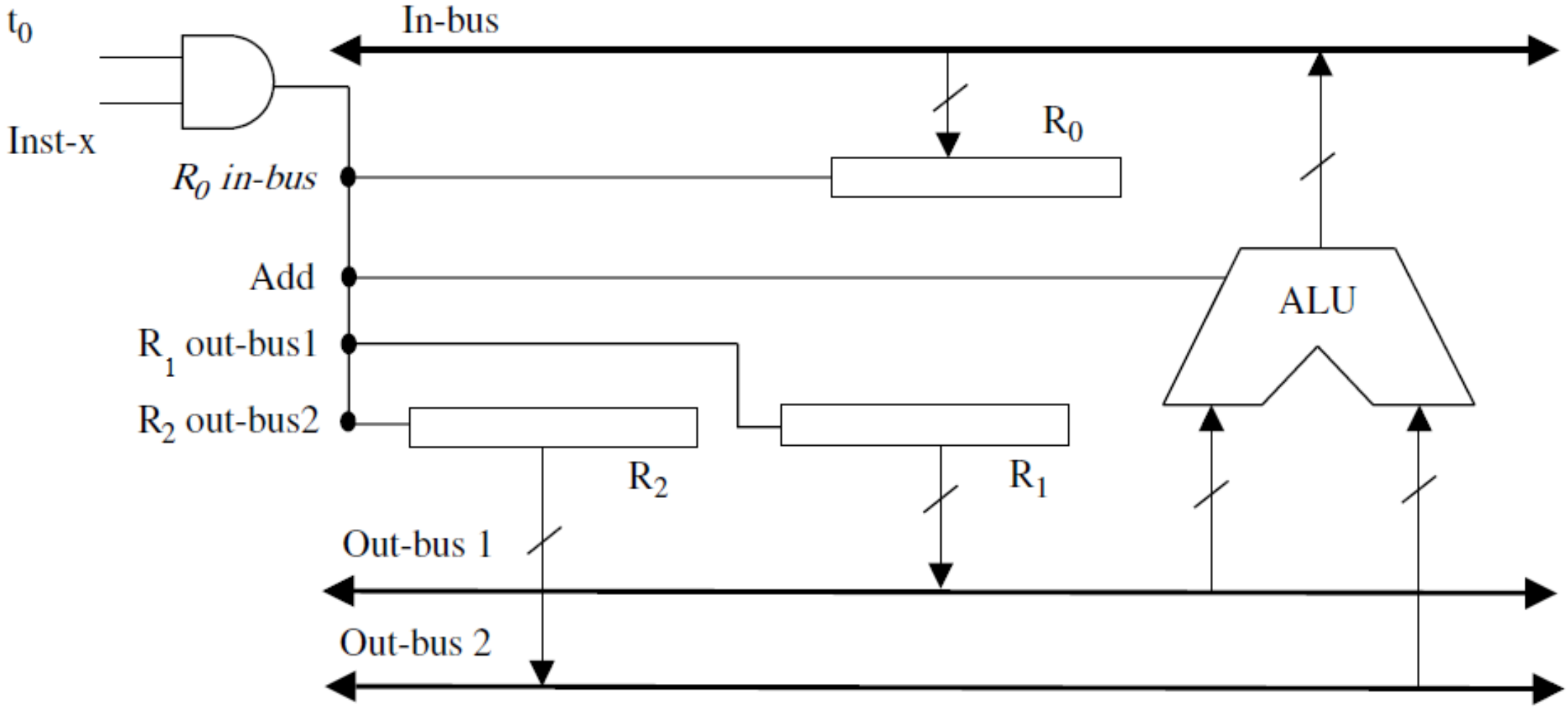
Control Unit

Example: ADD R_0, R_1, R_2 (1-Bus Organization)



Control Unit

Example: ADD R_0, R_1, R_2 (3-Bus Organization)



Control Unit

- There are mainly two different types of control units:
- Hardwired control
  - Fixed logic circuits that correspond directly to the Boolean expressions are used to generate the control signals.
  - Clearly hardwired control is faster than microprogrammed control.
  - However, hardwired control could be very expensive and complicated for complex systems.
  - Hardwired control is more economical for small control units.
  - Hardwired control will require a redesign of the entire systems in the case of any change.
- Microprogrammed control
  - Control signals associated with operations are stored in special memory units inaccessible by the programmer as control words.
  - A control word is a microinstruction that specifies one or more microoperations.
  - A sequence of microinstructions is called a microprogram, which is stored in a ROM or RAM called a control memory CM.
  - It should also be noted that microprogrammed control could adapt easily to changes in the system design.
  - We can easily add new instructions without changing hardware.

Hardwired Control Unit

- Direct implementation is accomplished using logic circuits
- For each control line, one must find the Boolean expression in terms of the input to the control signal generator

Example:

- Assume that the instruction set of a machine has
  - Three instructions: Inst-x, Inst-y, and Inst-z; and
  - Eight control lines: A, B, C, D, E, F, G, and H
- The following table shows the control lines that should be activated for the three instructions at the three steps t_0, t_1, and t_2

Step	Inst-x	Inst-y	Inst-z
t_0	D, B, E	F, H, G	E, H
t_1	C, A, H	G	D, A, C
t_2	G, C	B, C	

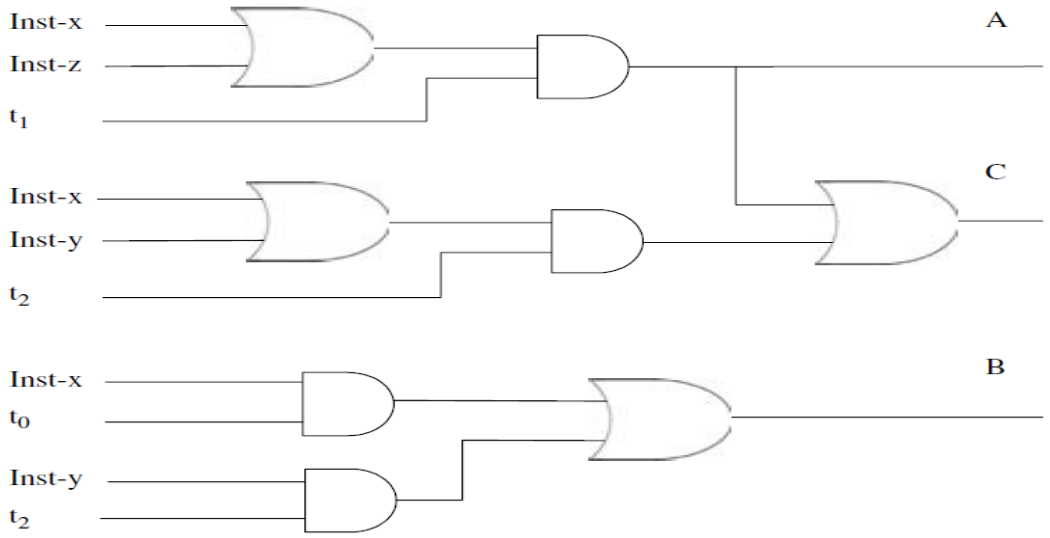
- The Boolean expressions for control lines A, B, and C can be obtained as follows:

$$A = \text{Inst-x} \cdot t_1 + \text{Inst-z} \cdot t_1 = (\text{Inst-x} + \text{Inst-z}) \cdot t_1$$

$$B = \text{Inst-x} \cdot t_0 + \text{Inst-y} \cdot t_2$$

$$C = \text{Inst-x} \cdot t_1 + \text{Inst-x} \cdot t_2 + \text{Inst-y} \cdot t_2 + \text{Inst-z} \cdot t_1$$

$$= (\text{Inst-x} + \text{Inst-z}) \cdot t_1 + (\text{Inst-x} + \text{Inst-y}) \cdot t_2$$



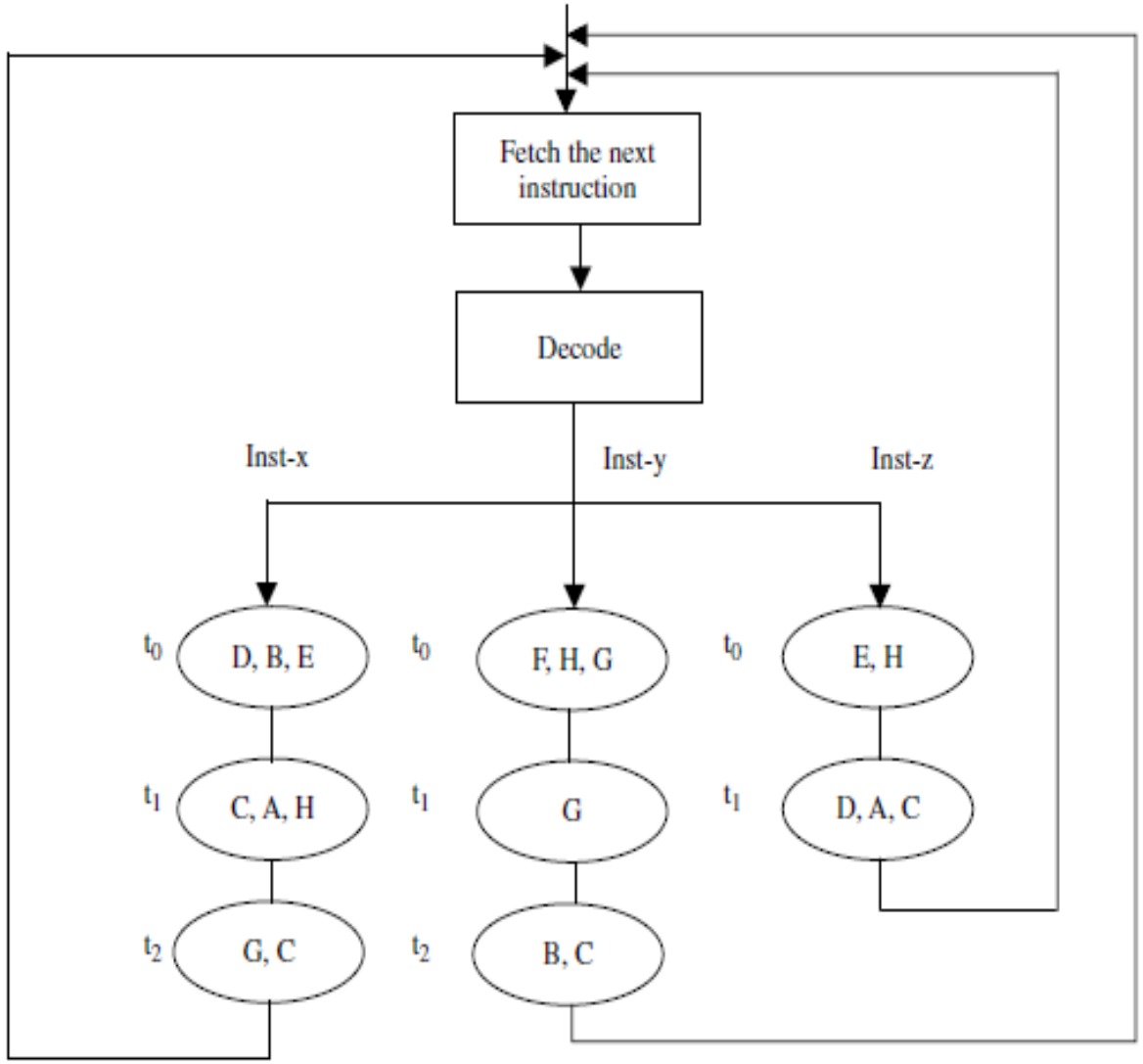
Hardwired Control Unit

- Direct implementation is accomplished using logic circuits
- For each control line, one must find the Boolean expression in terms of the input to the control signal generator

Example:

- Assume that the instruction set of a machine has
  - Three instructions: Inst-x, Inst-y, and Inst-z; and
  - Eight control lines: A, B, C, D, E, F, G, and H
- The following table shows the control lines that should be activated for the three instructions at the three steps t_0, t_1, and t_2

Step	Inst-x	Inst-y	Inst-z
t_0	D, B, E	F, H, G	E, H
t_1	C, A, H	G	D, A, C
t_2	G, C	B, C	

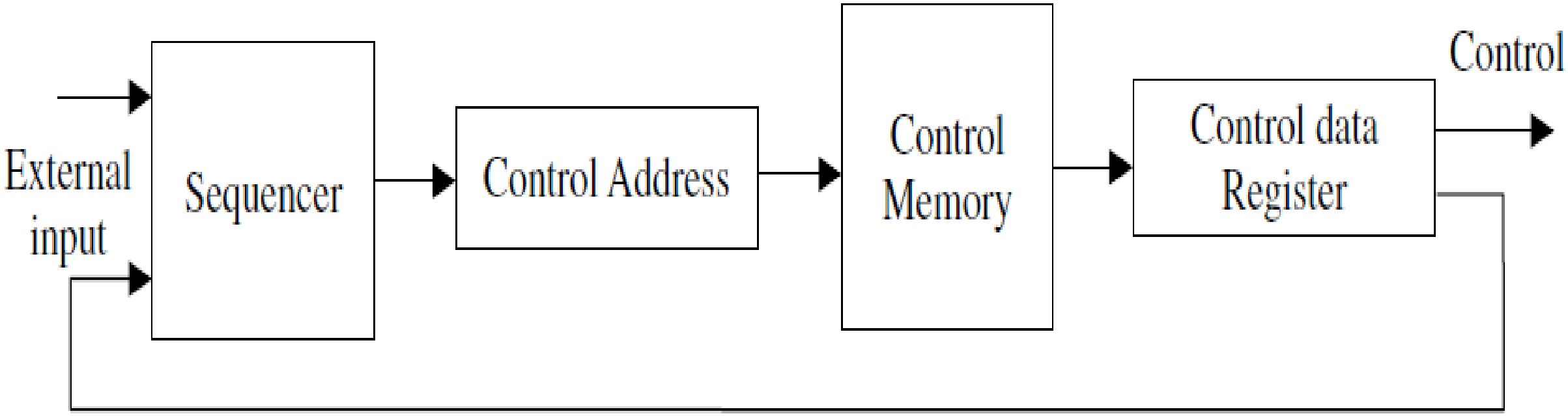


Microprogrammed Control Unit

- Idea introduced by M. V. Wilkes in early 1950s
- Microprogramming was motivated by the desire to reduce the complexities involved with hardwired control
- Each instruction is implemented using a set of micro-operations
- Associated with each micro-operation is a set of control lines that must be activated to carry out the corresponding microoperation
- The idea of microprogrammed control is to store the control signals associated with the implementation of a certain instruction as a microprogram in a special memory called a control memory (CM)
- A microprogram consists of a sequence of microinstructions
- A microinstruction is a vector of bits, where each bit is a control signal, condition code, or the address of the next microinstruction
- Microinstructions are fetched from CM the same way program instructions are fetched from main memory

Microprogrammed Control Unit

Fetching Microinstructions (Control Words)



Microprogrammed Control Unit

- When an instruction is fetched from memory, the op-code field of the instruction will determine which microprogram is to be executed
- In other words, the op-code is mapped to a microinstruction address in the control memory
- Microinstruction processor uses that address to fetch the first microinstruction in the microprogram
- After fetching each microinstruction, the appropriate control lines will be enabled
- Every control line that corresponds to a “1” bit should be turned on
- Every control line that corresponds to a “0” bit should be left off
- After completing the execution of one microinstruction, a new microinstruction will be fetched and executed
- If the condition code bits indicate that a branch must be taken, the next microinstruction is specified in the address bits of the current microinstruction
- Otherwise, the next microinstruction in the sequence will be fetched and executed

Microprogrammed Control Unit

- The length of a microinstruction is determined based on
  - the number of microoperations specified in the microinstructions,
  - the way the control bits will be interpreted, and
  - the way the address of the next microinstruction is obtained
- A microinstruction may specify one or more micro-operations that will be activated simultaneously
- The length of the microinstruction will increase as the number of parallel micro-operations per microinstruction increases
- Furthermore, when each control bit in the microinstruction corresponds to exactly one control line, the length of microinstruction could get bigger
- The length of a microinstruction could be reduced if control lines are coded in specific fields in the microinstruction
- Decoders will be needed to map each field into the individual control lines
- Using the decoders will reduce the number of control lines that can be activated simultaneously
- There is a tradeoff between the length of the microinstructions and the amount of parallelism
- It is important that we reduce the length of microinstructions to reduce the cost and access time of the control memory
- It may also be desirable that more micro-operations be performed in parallel and more control lines can be activated simultaneously