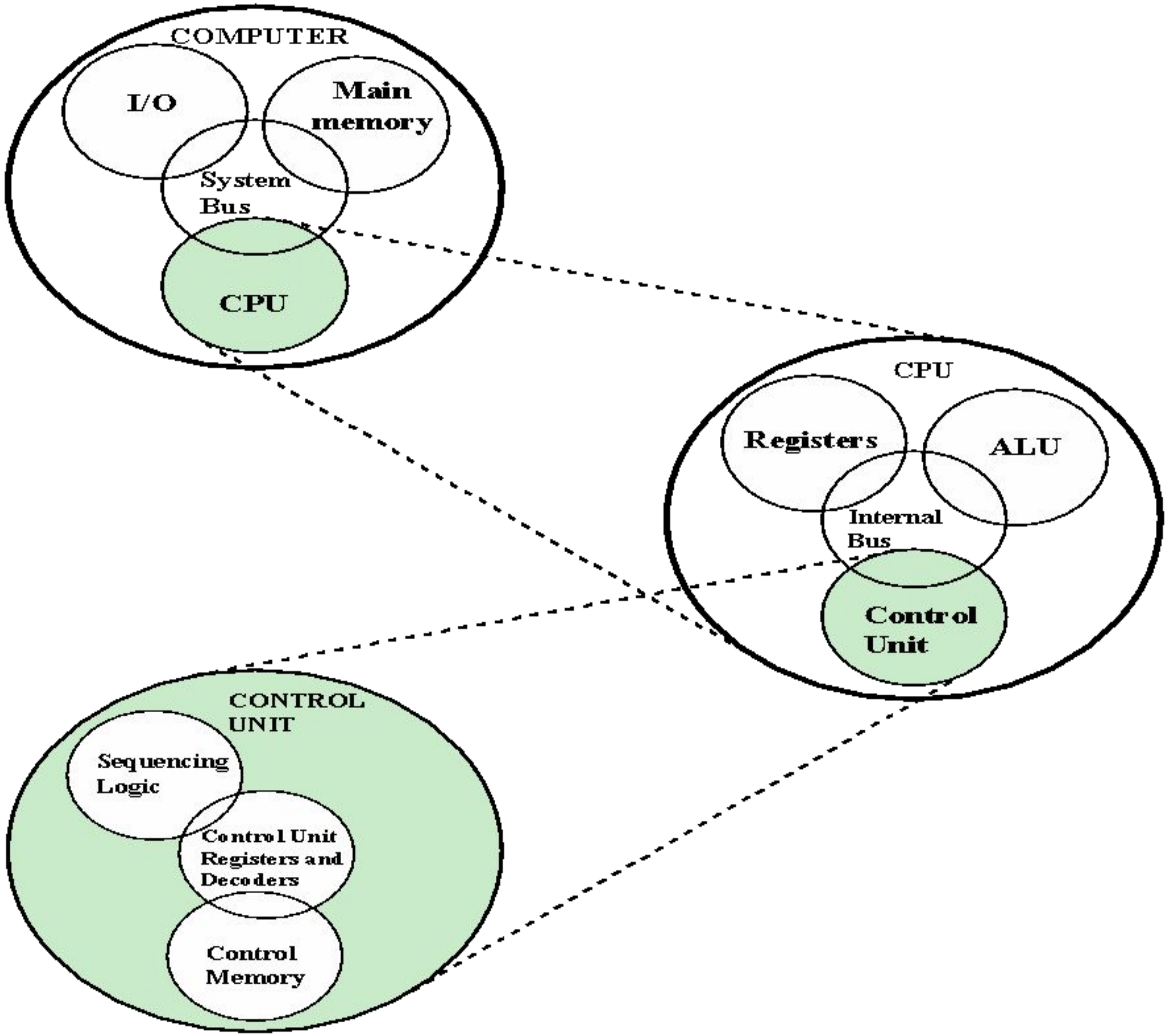
ALU and Data Path

Santosh Kumar Verma
Mentor
GeeksforGeeks

ALU and Data Path in Computer Organization

- As per the developments, a real need of complex operations are more required.
- These operations are handled by the ALU.



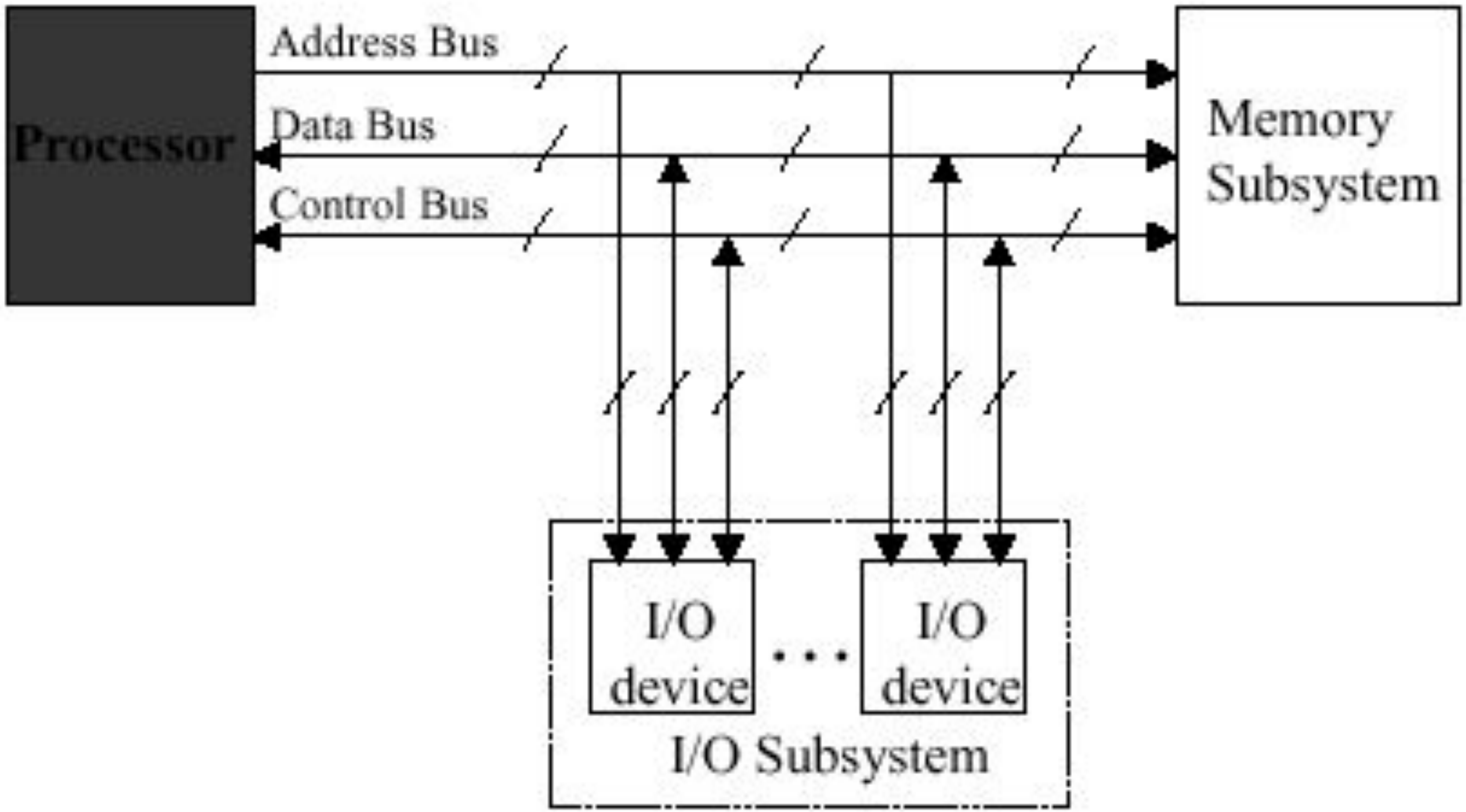
Structure

The **central processing unit (CPU)** can be divided into two sections:

1. **Data section:** Memory, registers, ALU (adders, multiplier, etc.), and communication buses. Each step (fetch, decode, execute, save the result) requires communication (data transfer) paths between memory, registers and ALU. It is also known as the ***data path.***

2. **Control section:** Data path for each step is set up by control signals that set up dataflow directions on communication buses and select ALU and memory functions. Control signals are generated by a control unit consisting of one or more finite state machines.

Data Path


Processor
Address Bus
Data Bus
Control Bus
Memory
Subsystem
I/O
device
I/O
device
I/O Subsystem

BUS

There are mainly three type of bus:-

1. **Address bus:** Transfers memory addresses from the processor to components like storage and input/output devices. It’s one-way communication.

2. **Data bus:** carries the data between the processor and other components. The data bus is bidirectional.

3. **Control bus:** carries control signals from the processor to other components. The control bus also carries the clock’s pulses. The control bus is unidirectional.

The bus can be dedicated,i.e., it can be used for a single purpose or it can be multiplexed.

Registers

The following list of five registers for in-out signal data storage:

1. **Program Counter:** A CPU register in the computer processor which has the address of the next instruction to be executed from memory.

2. **Instruction Register:** holds the instruction currently being executed or decoded

3. **Memory Address Register** (MAR): is the CPU register that either stores the memory address from which data will be fetched, or the address to which data will be sent and stored.

4. **Memory Data Register** (MDR): is the register that stores the data being transferred to and from the immediate access storage. Also known as memory buffer register (MBR).

5. **General Purpose Register (GPR):** are used to store temporary data within the microprocessor. It is a multipurpose register.

A data path is a collection of functional units such as ***arithmetic logic units*** or multipliers that perform data processing operations, ***registers,*** and ***buses.***

Along with the control unit, it composes the central processing unit (CPU). A larger data path can be made by joining more than one data path using multiplexers.

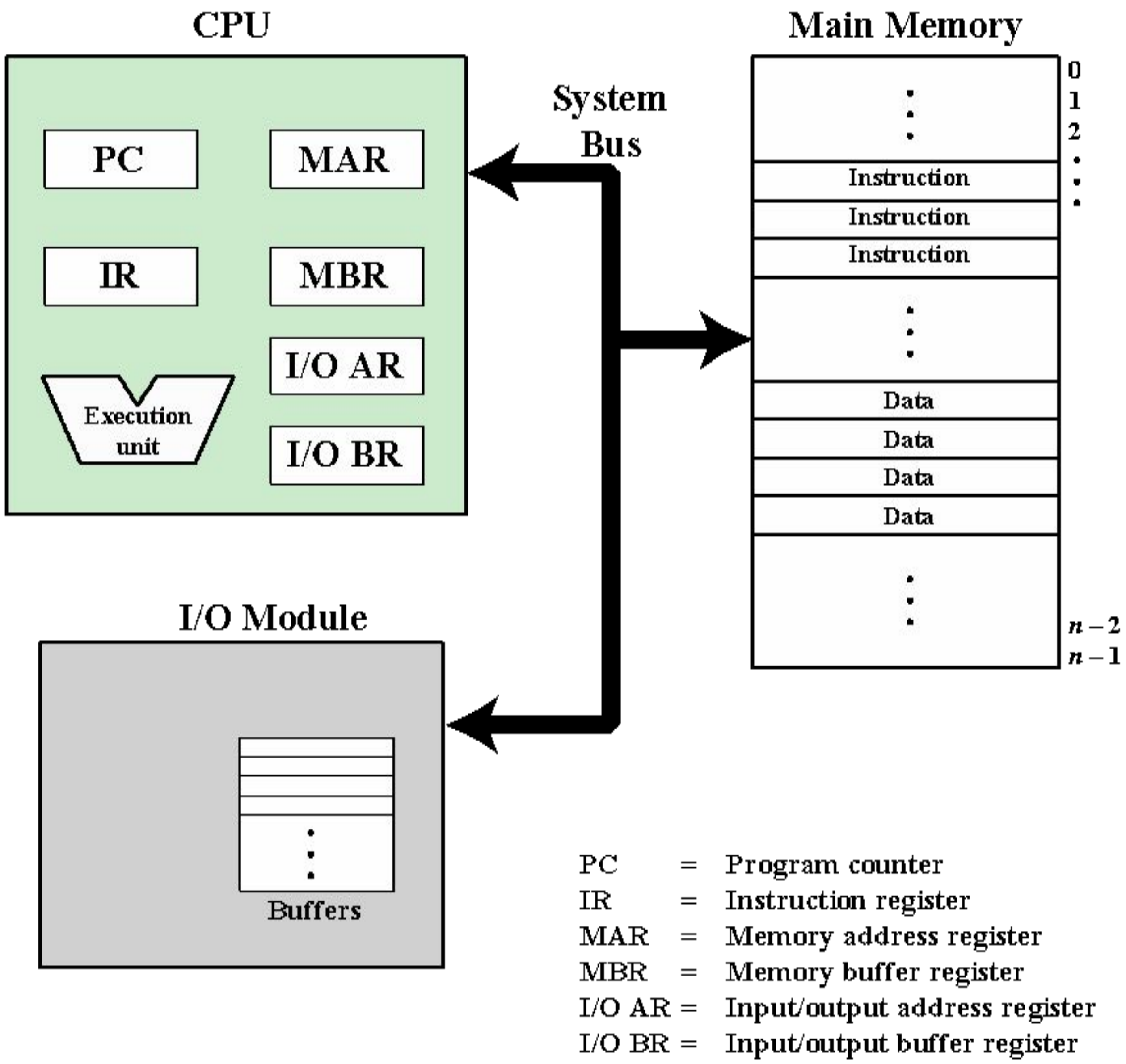
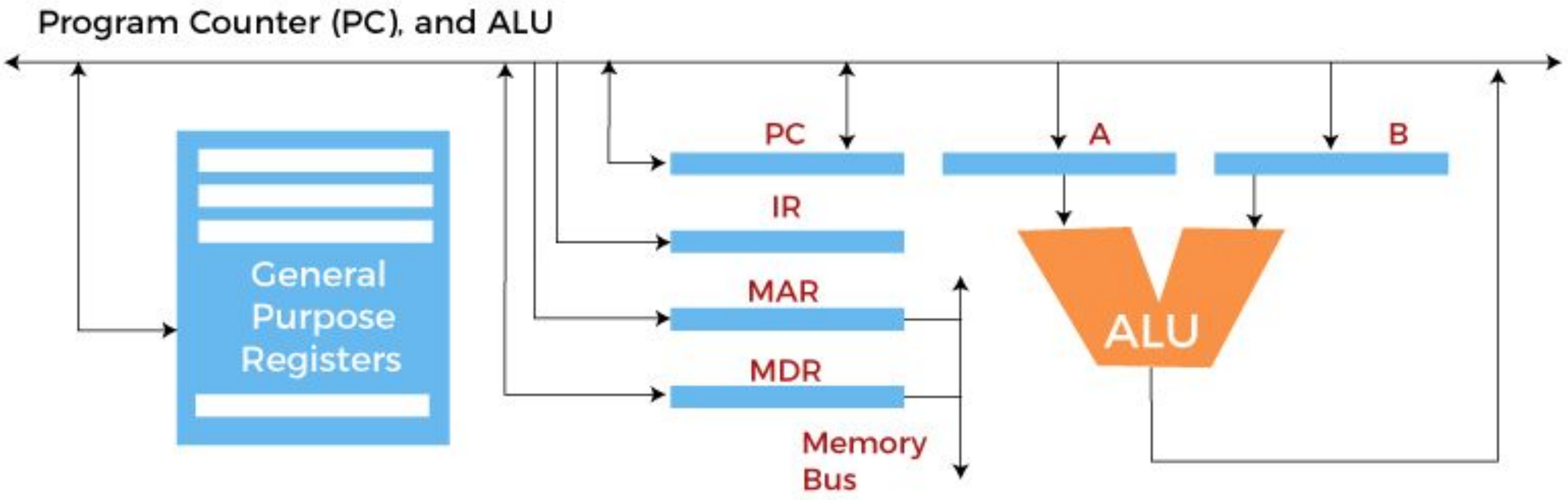


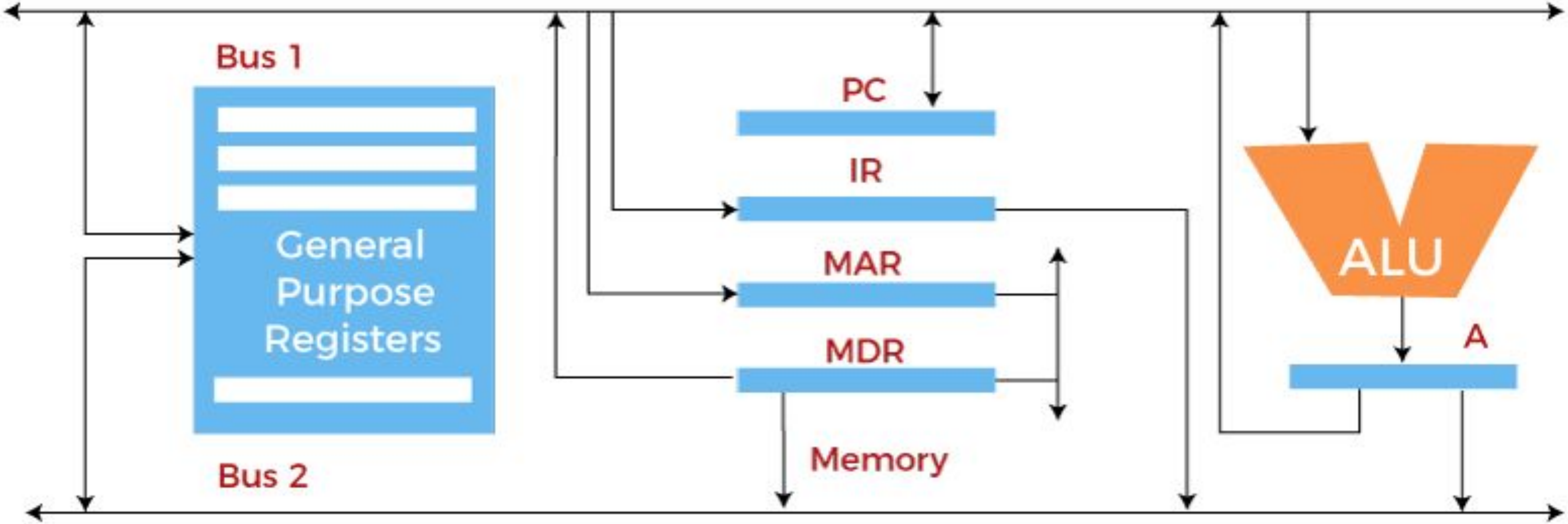
Figure 3.2 Computer Components: Top-Level View

One Bus organization



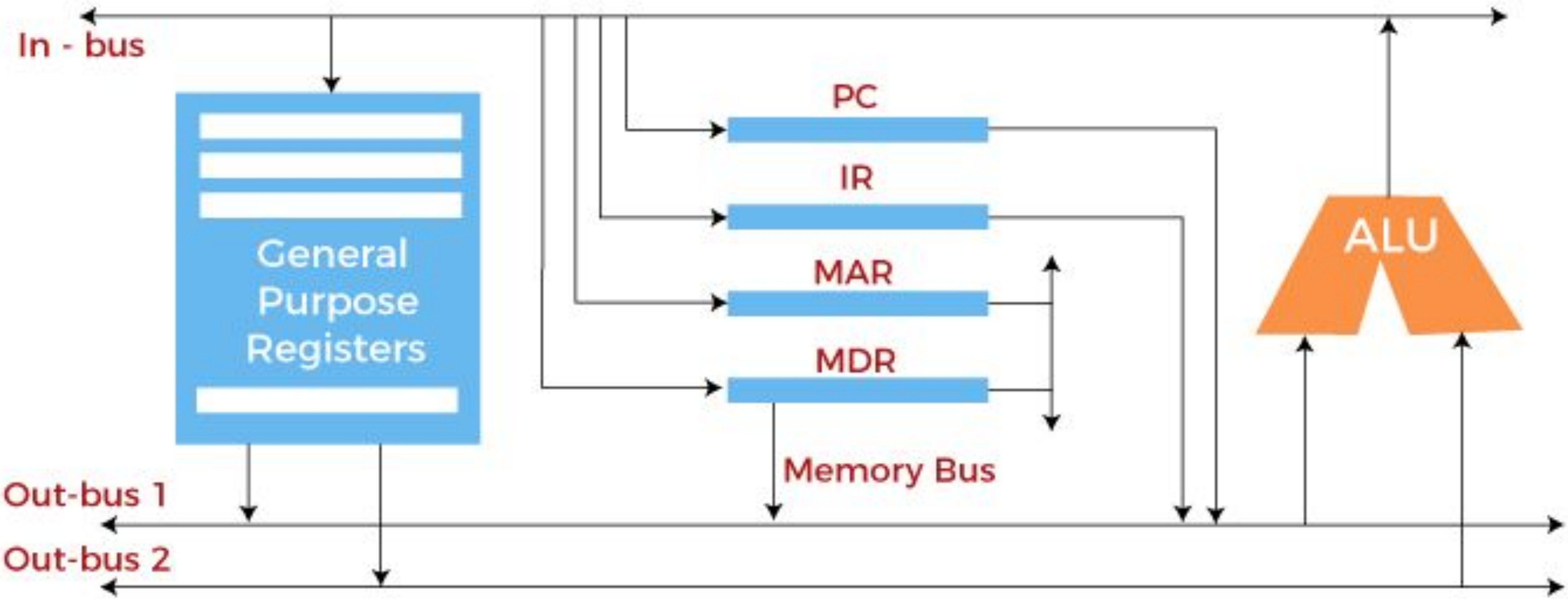
One-bus datapath

Two Bus organizations

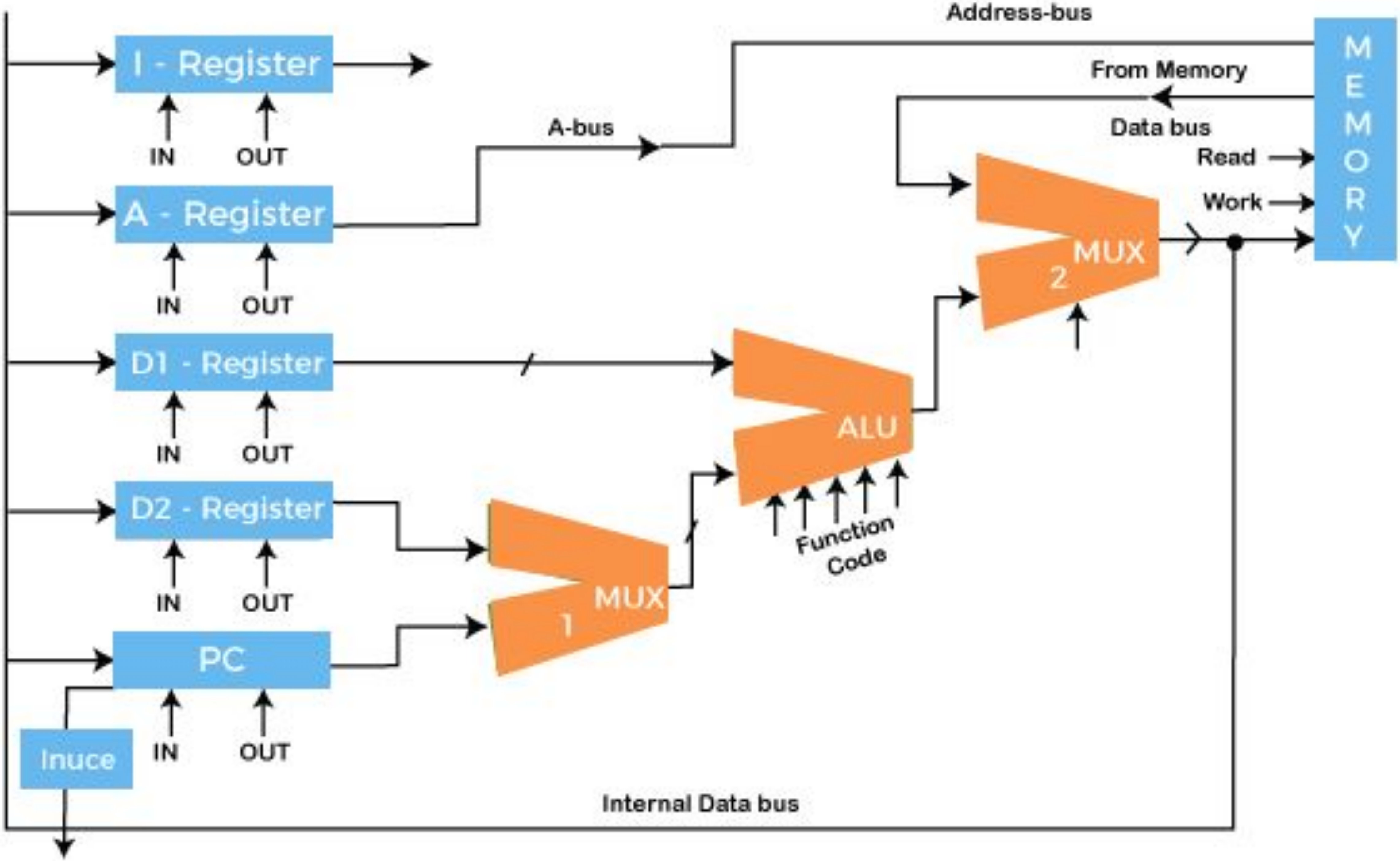


Two Bus Datapath

Three Bus organization



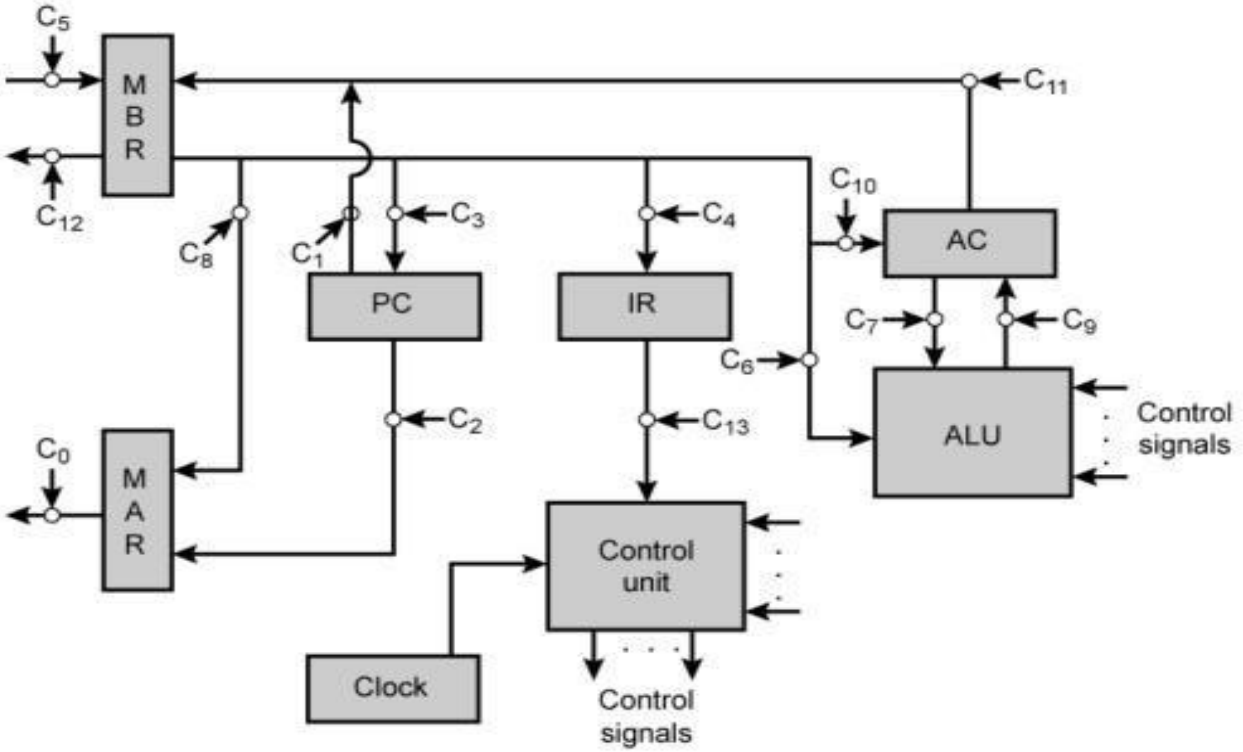
Three Bus Datapath

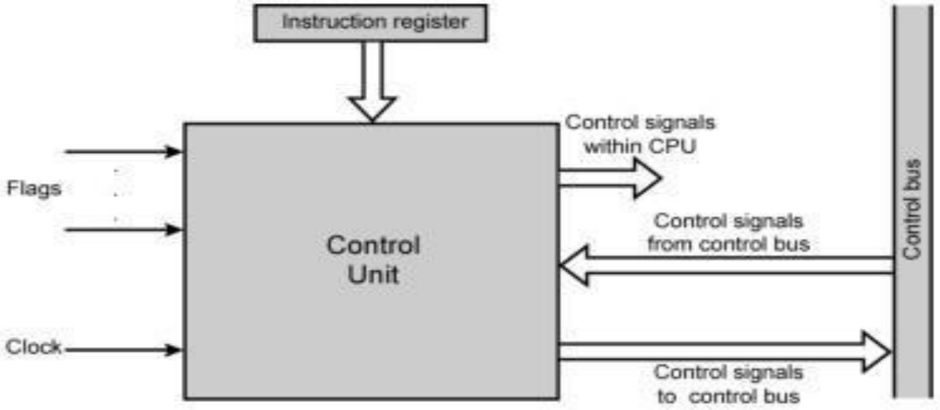


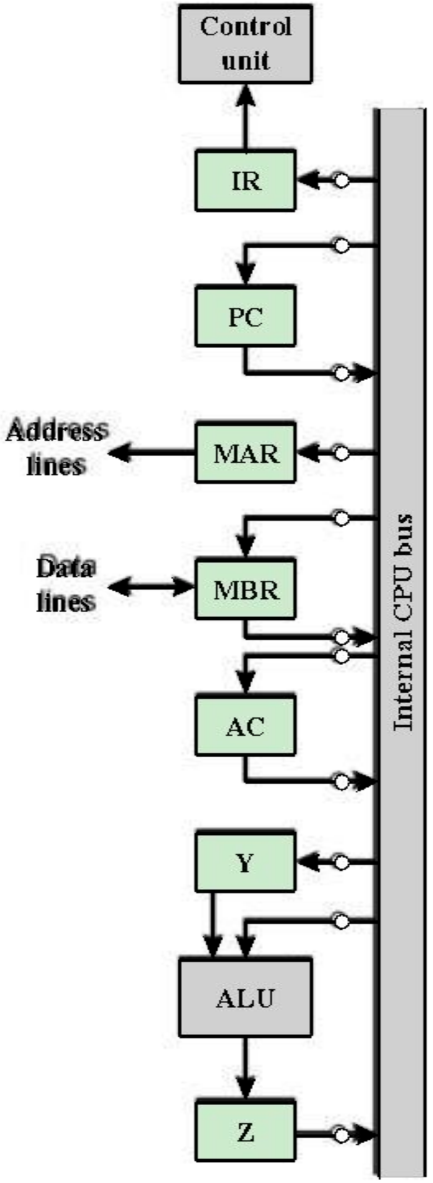
Address-bus
I - Register
IN
OUT
From Memory
A-bus
Data bus
Read
Work
A - Register
IN
OUT
MUX
2
M
E
M
O
R
Y
D1 - Register
IN
OUT
ALU
D2 - Register
IN
OUT
Function
Code
MUX
1
PC
Inuce
IN
OUT
Internal Data bus

Control Unit and Operations

A Simple Computer & its Control Unit



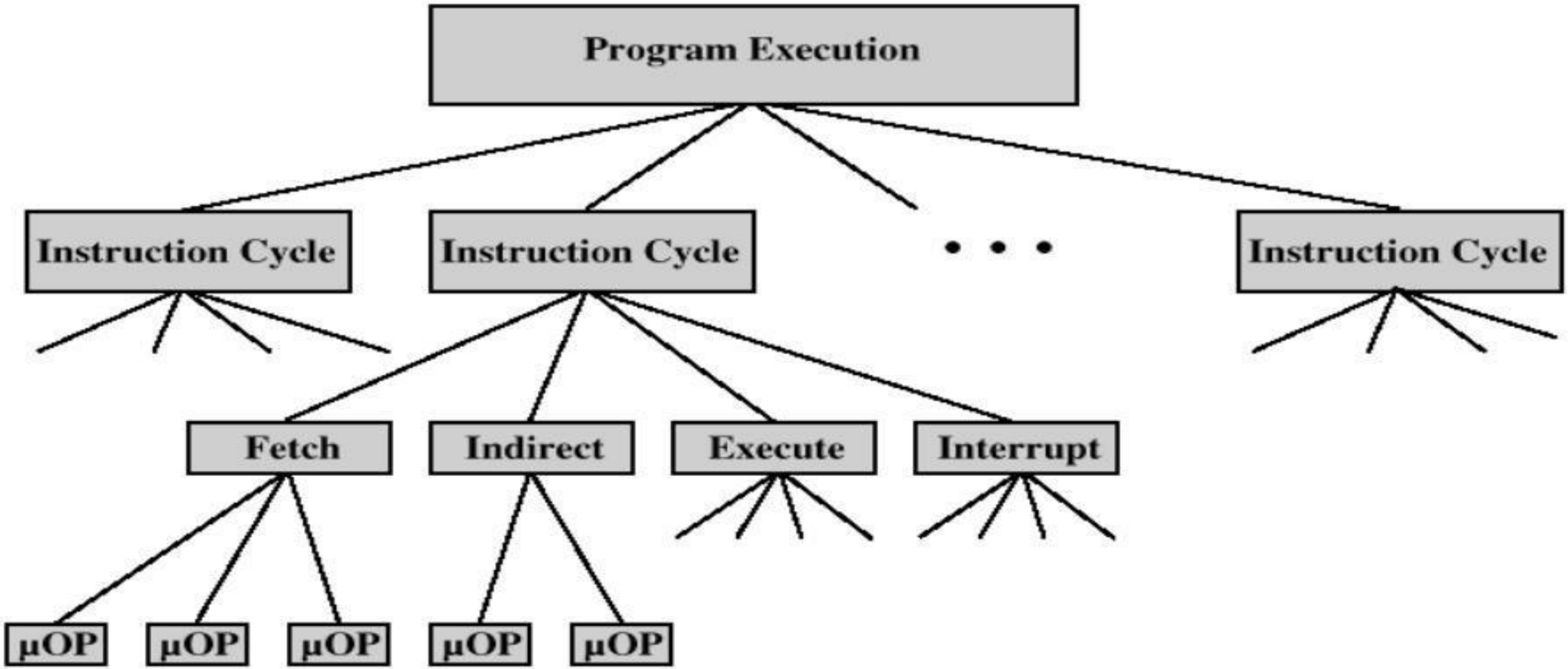




Instruction Micro-Operations

- A computer executes a program of instructions (or instruction cycles)
- Each instruction cycle has a number to steps or phases:
    - **Fetch**,
    - **Indirect** (if specified),
    - **Execute**,
    - **Interrupt** (if requested)
- These can be seen as micro-operations
    - Each step does a modest amount of work
    - Atomic operation of CPU

Constituent Elements of its Program Execution



Types of Micro-operation

- Transfer data between registers
- Transfer data from register to external
- Transfer data from external to register
- Perform arithmetic or logical ops

Functions of Control Unit

- Sequencing
  - Causing the CPU to step through a series of micro-operations
- Execution
  - Causing the performance of each micro-op
- This is done using Control Signals

Control Signals

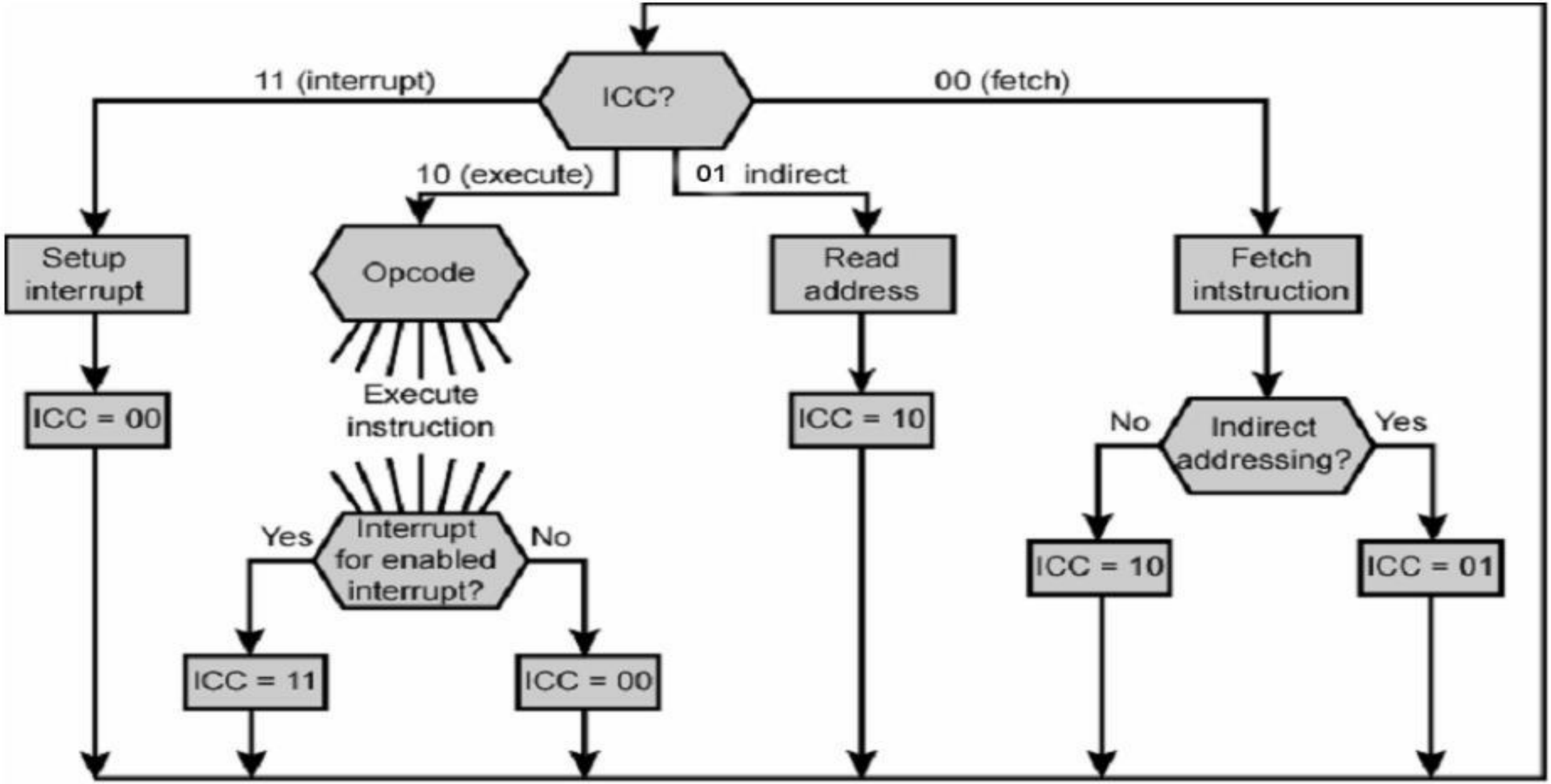
- Clock
  - One micro-instruction (or set of parallel micro-instructions) per clock cycle
- Instruction register
  - Op-code for current instruction
  - Determines which micro-instructions are performed
- Flags
  - State of CPU
  - Results of previous operations
- From control bus
  - Interrupts
  - Acknowledgements

Control Signals - output

- Within CPU
  - Cause data movement
  - Activate specific functions
- Via control bus
  - To memory
  - To I/O modules

Examples of Control Signal Sequence

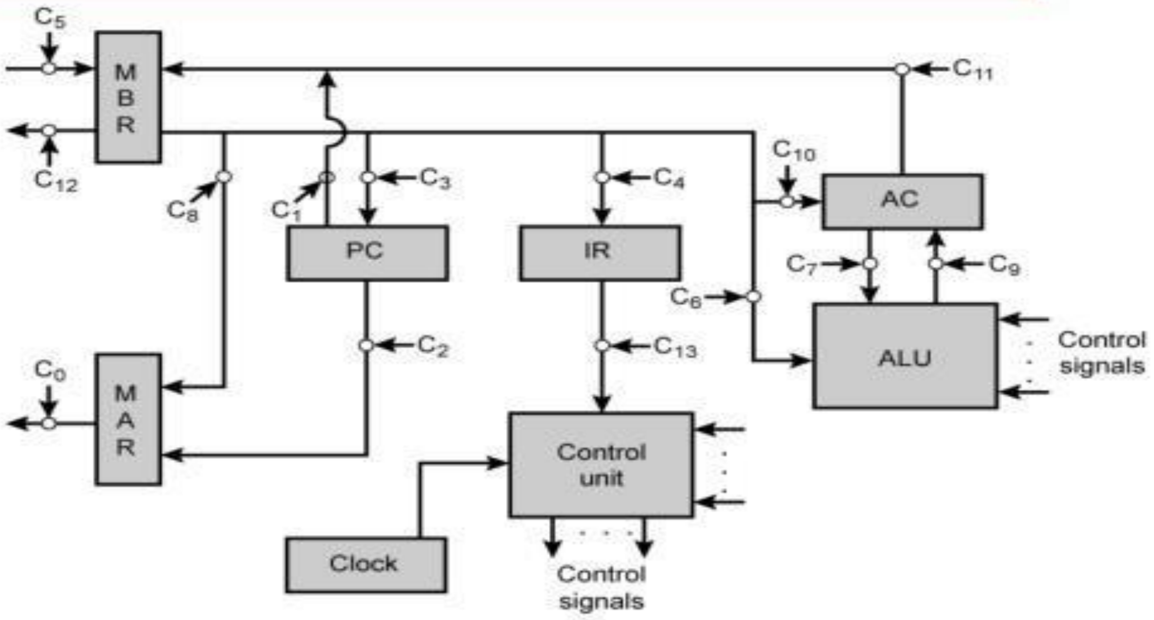
Flowchart for Instruction Cycle



Fetch - 4 “Control” Registers Utilized

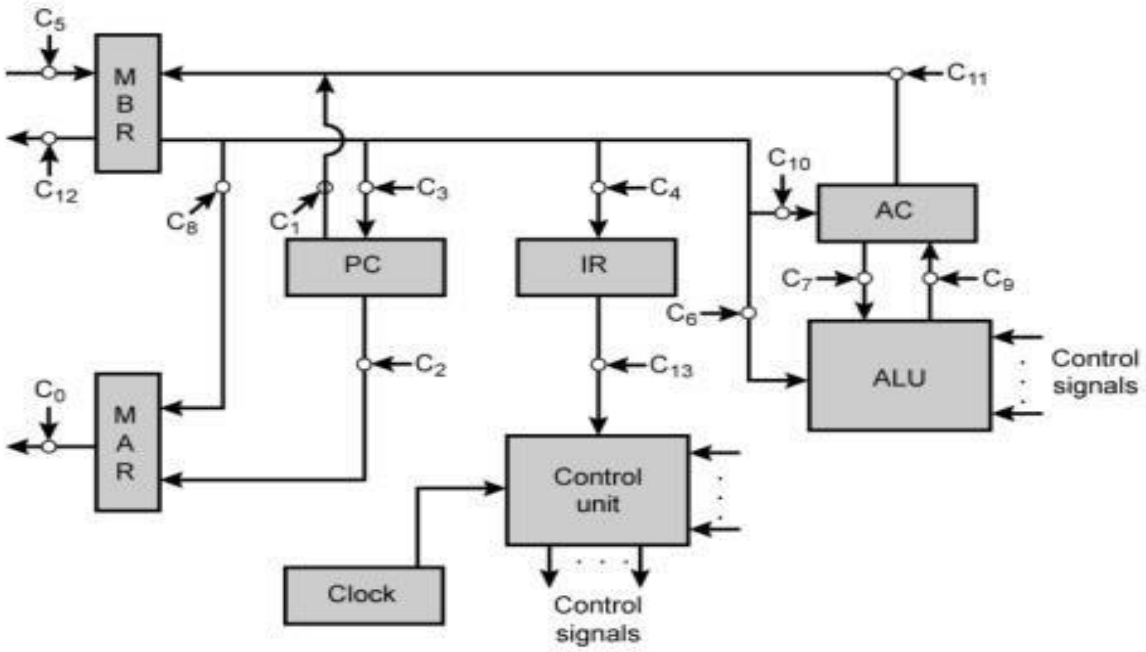
- Program Counter **(PC)**
  - Holds address of next instruction to be fetched
- Memory Address Register **(MAR)**
  - Connected to address bus
  - Specifies address for read or write op
- Memory Buffer Register **(MBR)**
  - Connected to data bus
  - Holds data to write or last data read
- Instruction Register **(IR)**
  - Holds last instruction fetched

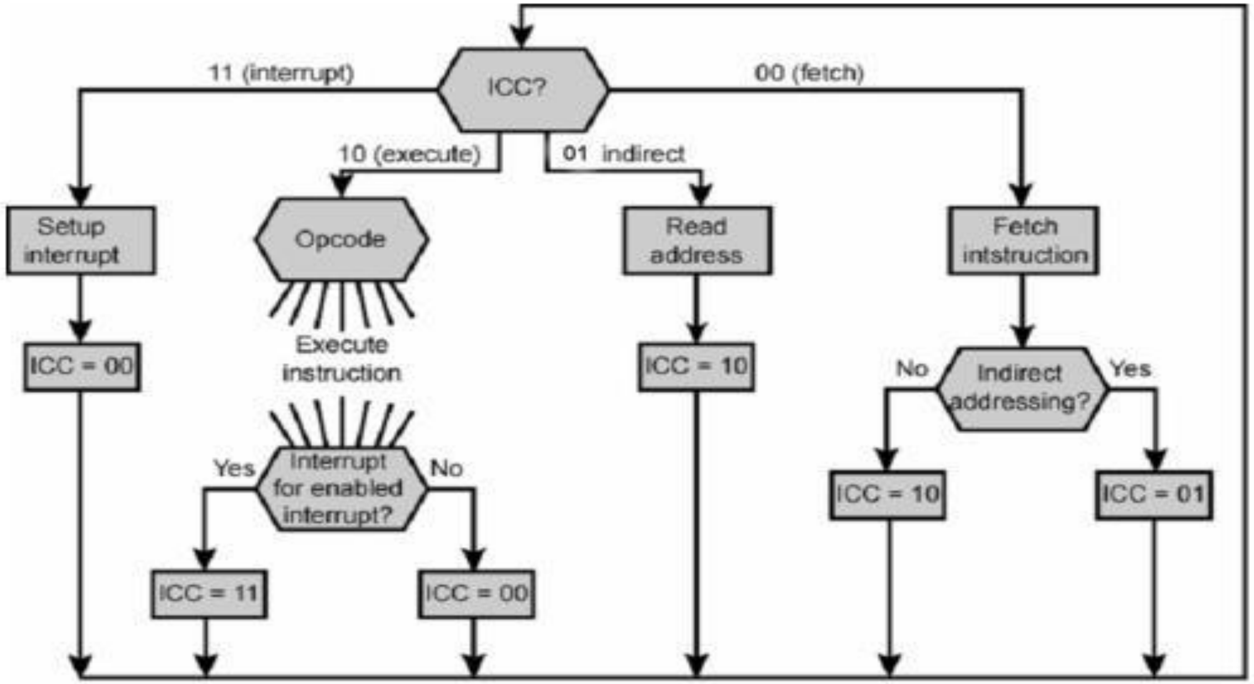
Fetch Cycle



- Address of next instruction is in PC
- Address (MAR) is placed on address bus
  t1: MAR ← (PC)
- Control unit issues READ command
- Result (data from memory) appears on data bus
- Data from data bus copied into MBR
  t2: MBR ← (memory)
- PC incremented by 1 (in parallel with data fetch from memory)
  PC ← (PC) +1
- Data (instruction) moved from MBR to IR
  t3: IR ← (MBR)
- MBR is now free for further data fetches

Fetch Cycle





Fetch Cycle:

t1: MAR ← (PC)
t2: MBR ← (memory)
PC ← (PC) +1
t3: IR ← (MBR)

Fetch Cycle

- Let Tx be the time unit of the clock. Then:

t1: MAR $\leftarrow$ (PC)
t2: MBR $\leftarrow$ (memory)
PC $\leftarrow$ (PC) +1
t3: IR $\leftarrow$ (MBR)

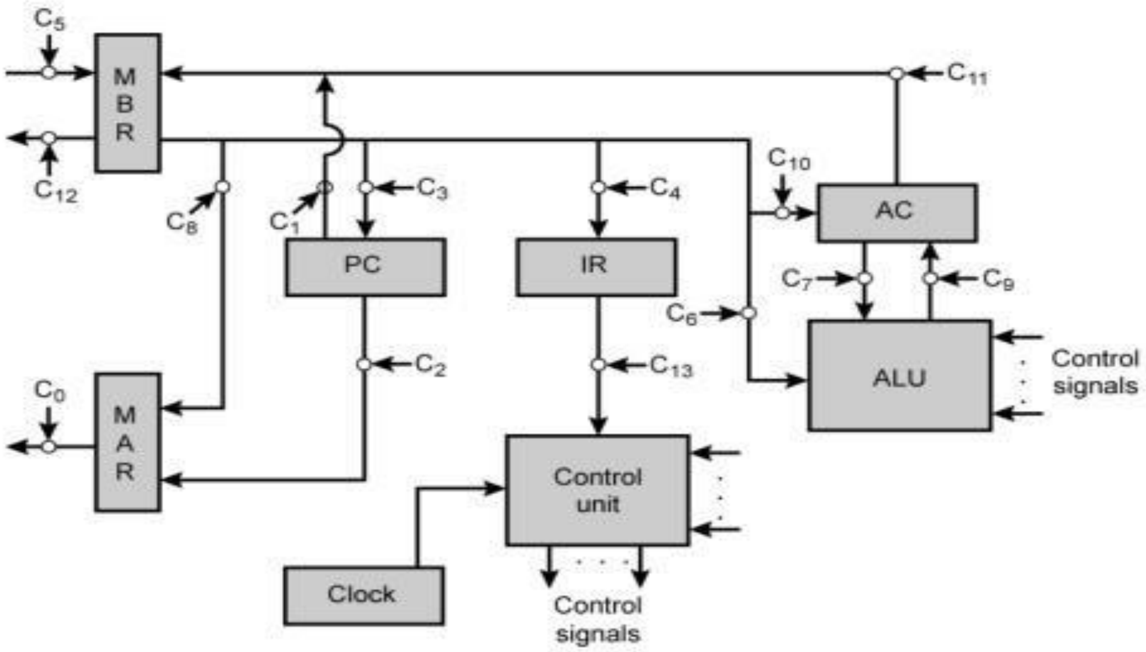
- Is this equally correct? Why?

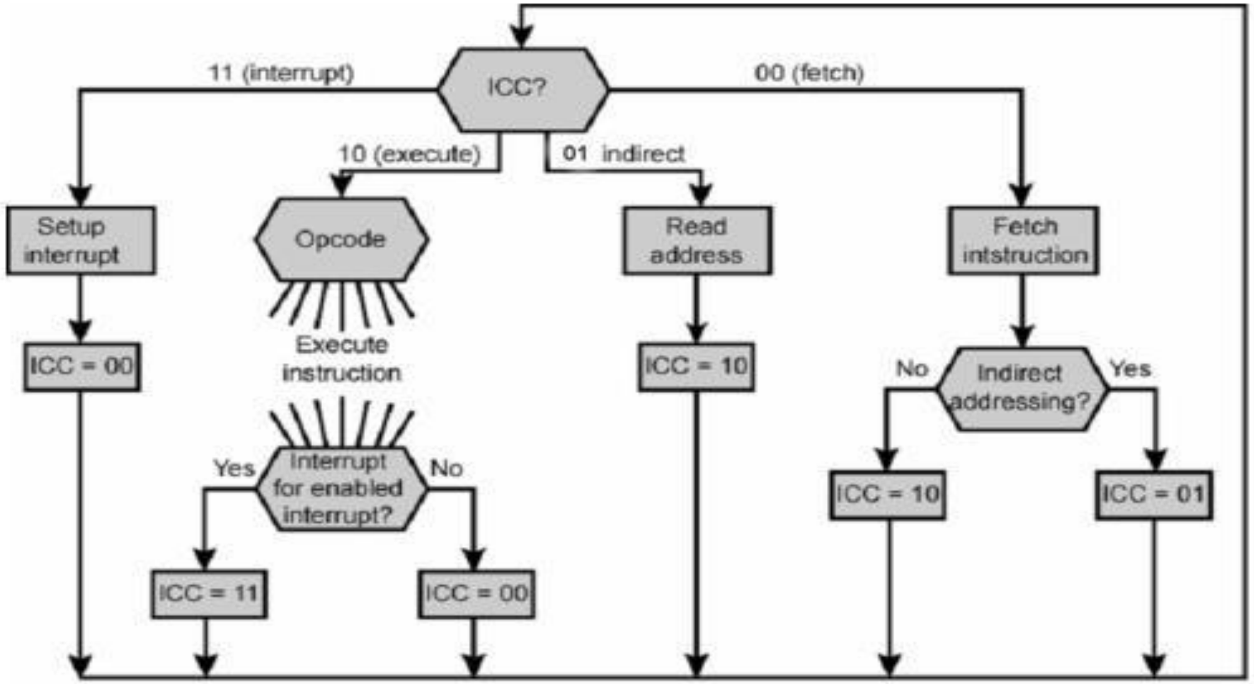
t1: MAR $\leftarrow$ (PC)
t2: MBR $\leftarrow$ (memory)
t3: PC $\leftarrow$ (PC) +1
IR $\leftarrow$ (MBR)

Basic Rules for Clock Cycle Grouping

- Proper sequence must be followed
  - MAR ← (PC) must precede MBR ← (memory)
- Conflicts must be avoided
  - Must not read & write same register at same time
  - MBR ← (memory) & IR ← (MBR) must not be in same cycle
- Also: PC ← (PC) +1 involves addition
  - Use ALU ?
  - May need additional micro-operations

Indirect Cycle



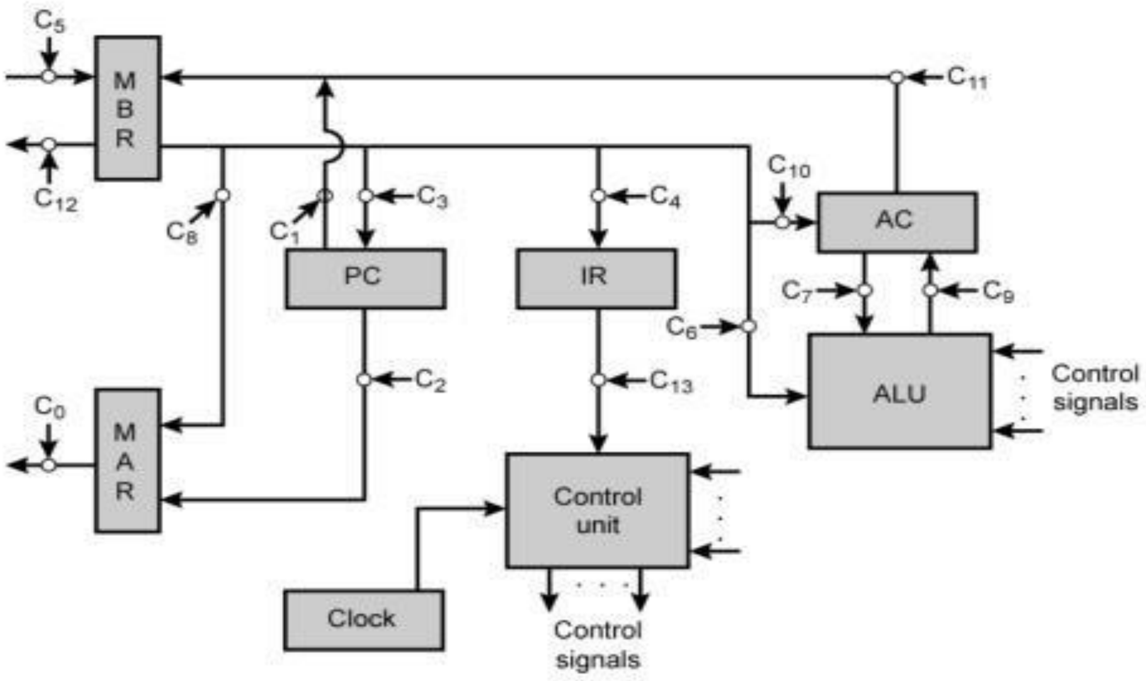


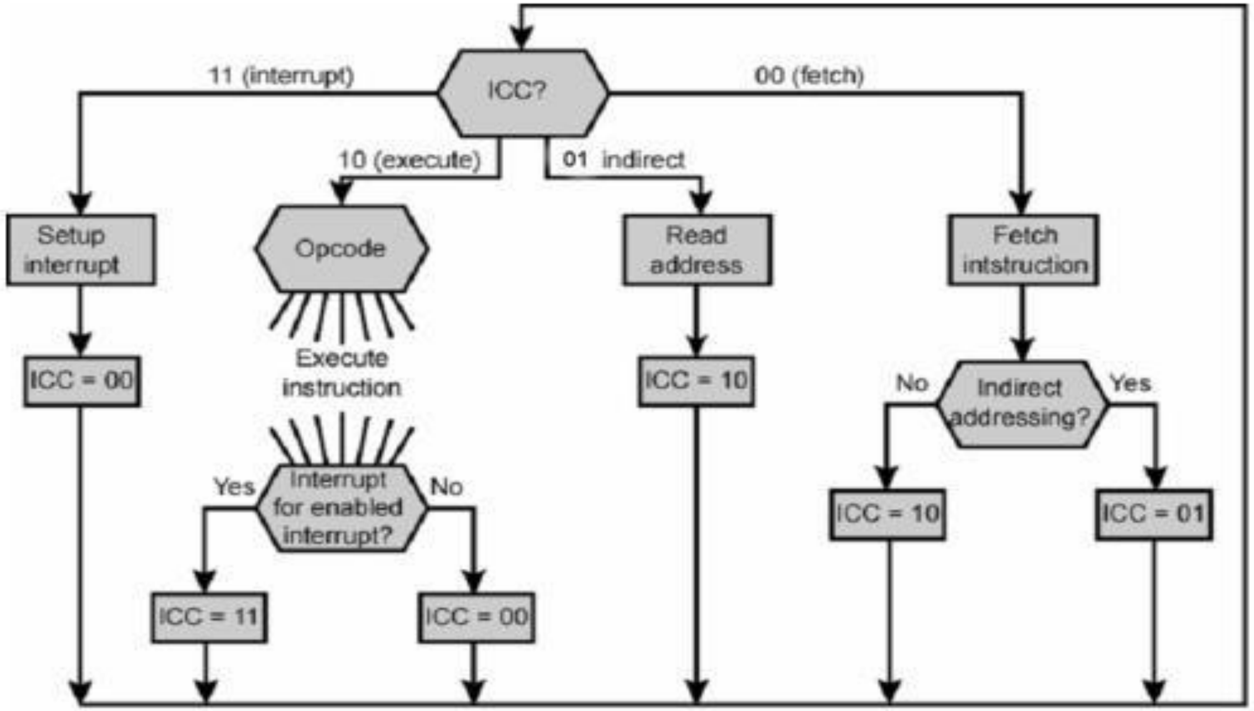
Indirect Cycle:

t1: MAR ← ($IR_{address}$)
t2: MBR ← (memory)
t3: $IR_{address}$ ← ($MBR_{address}$)

- IR is now in same state as if direct addressing had been used
- (What does this say about IR size?)

Interrupt Cycle



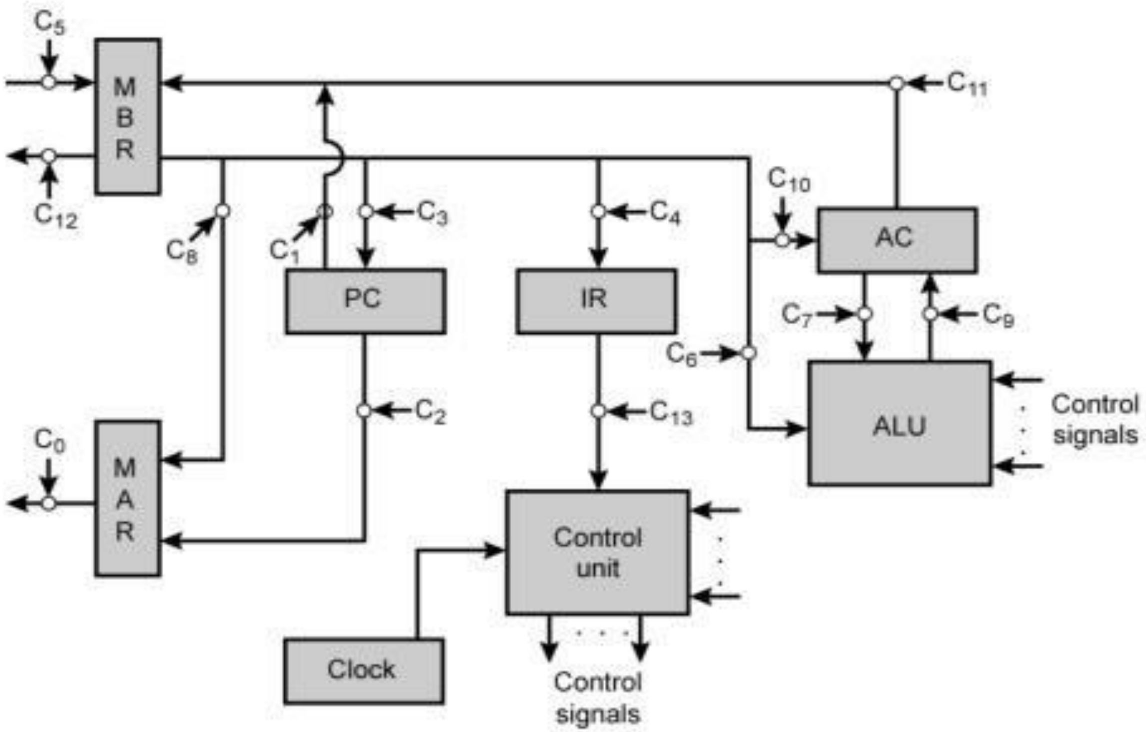


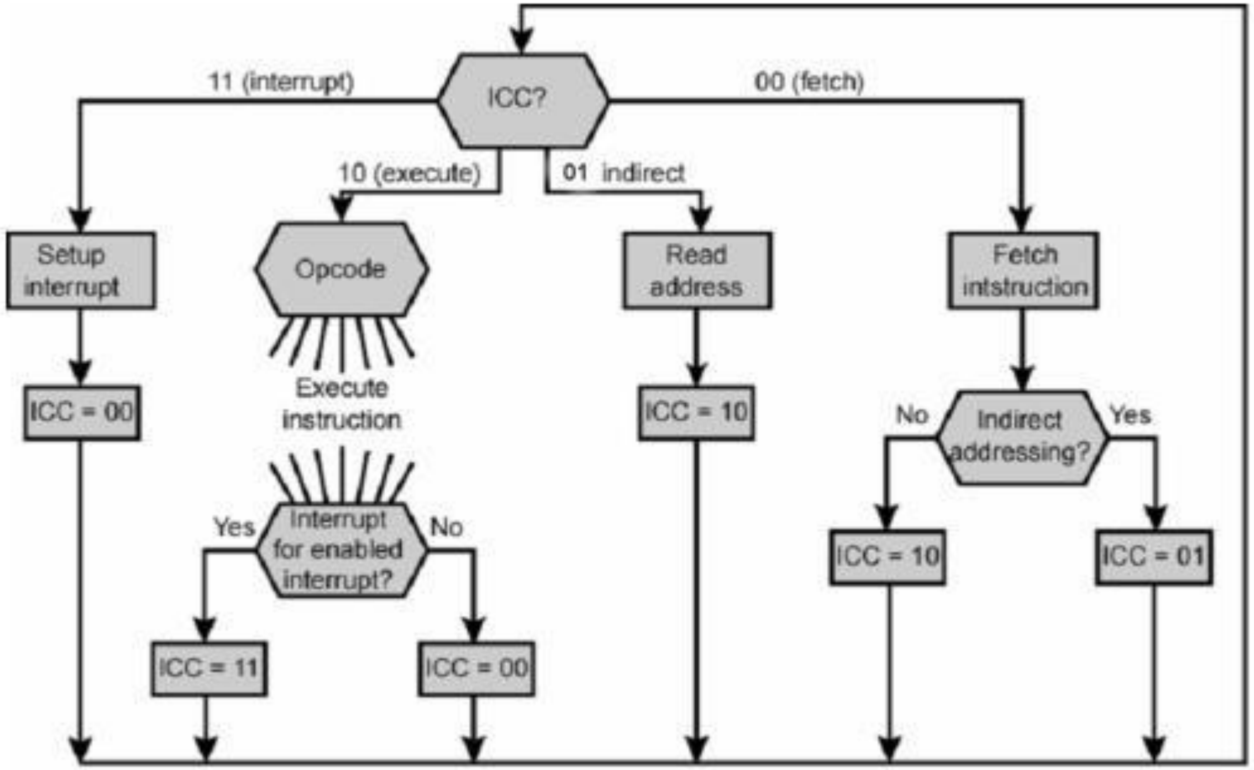
Interrupt Cycle:

t1: MBR ← (PC)
t2: MAR ← save-address
PC ← routine-address
t3: memory ← (MBR)

- This is a minimum. May be additional micro-ops to get addresses
  - N.B. saving context is done by interrupt handler routine, not micro-ops

Execute Cycle: ADD R1, memory



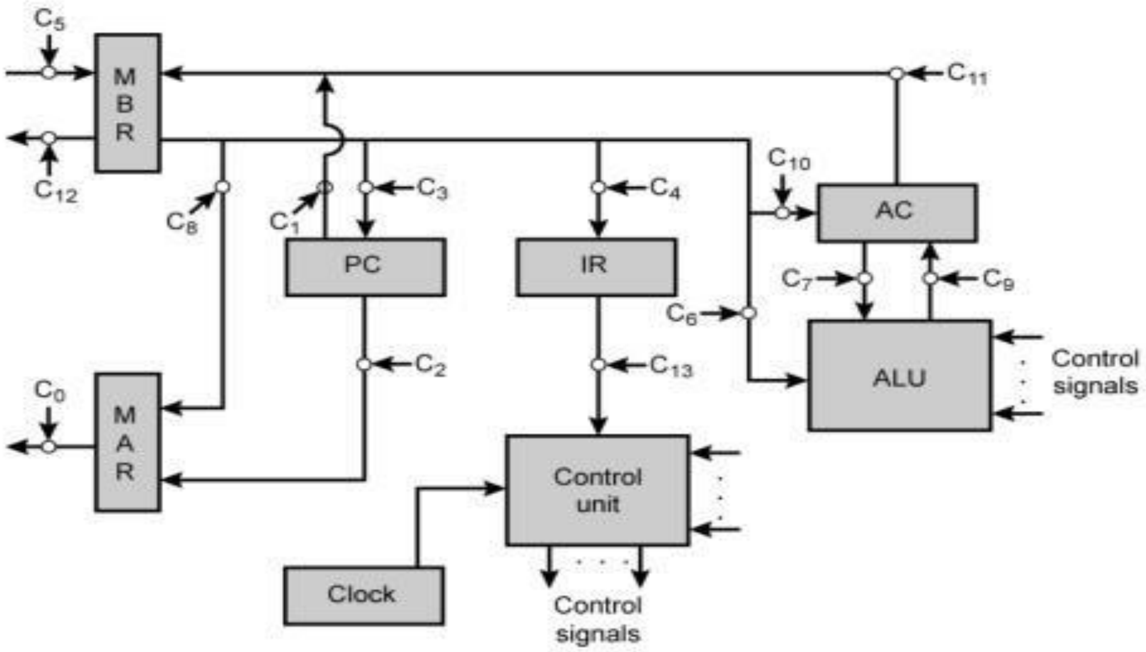


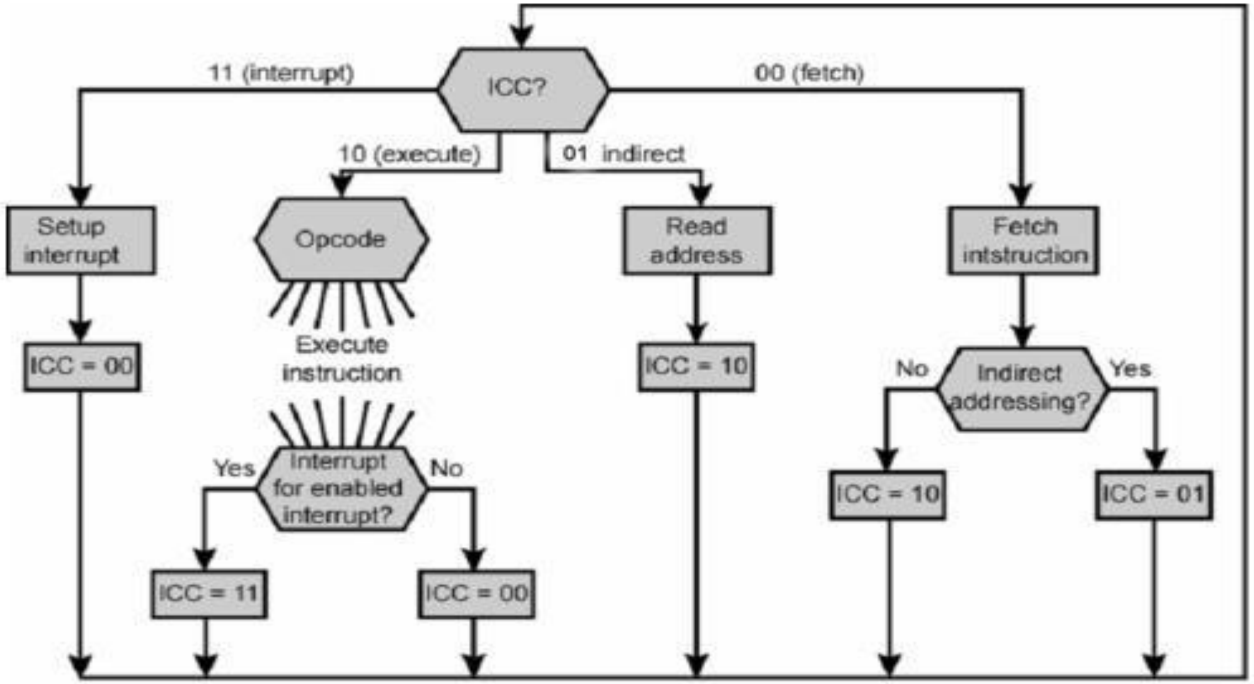
Execute Cycle: ADD R1, X

t1: MAR ← ($IR_{address}$)
t2: MBR ← (memory)
t3: R1 ← R1 + (MBR)

- Different for each instruction
- Note no overlap of micro-operations

Execute Cycle: ISZ X



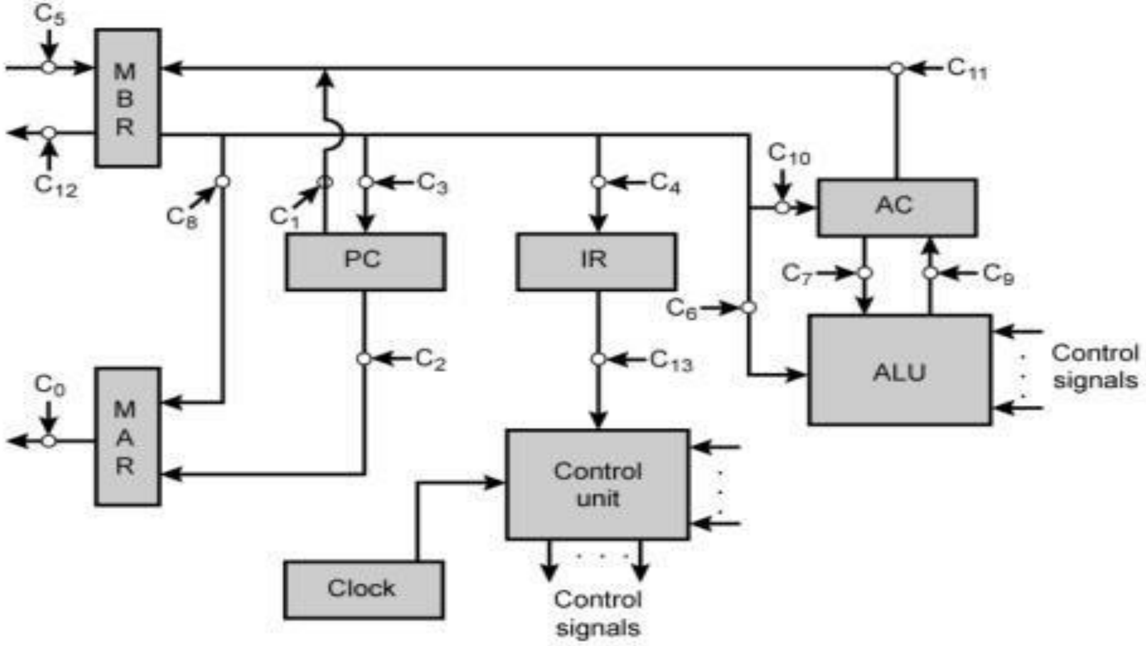


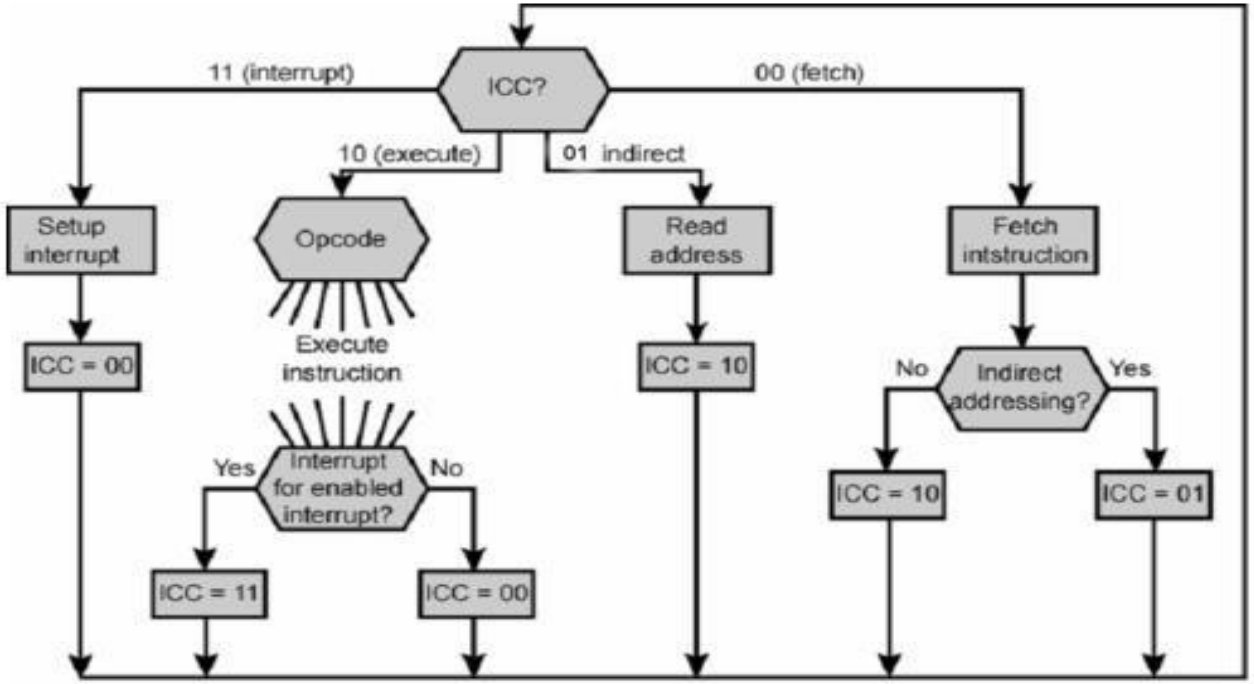
Execute Cycle: ISZ X (inc and skip if zero)

t1: MAR ← ($IR_{address}$)
t2: MBR ← (memory)
t3: MBR ← (MBR) + 1
t4: memory ← (MBR)
if (MBR) == 0 then
PC ← (PC) + 1

- Notes:
  - "if" is a single micro-operation
  - Micro-operations done during t4

Execute Cycle: BSA X





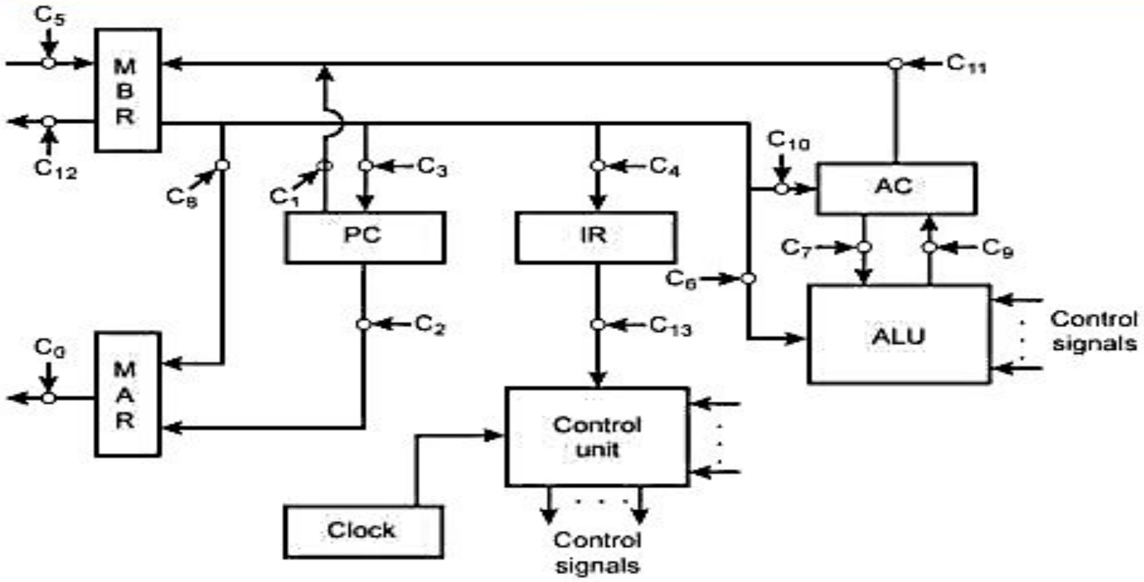
Execute: BSA X (Branch and Save Address)

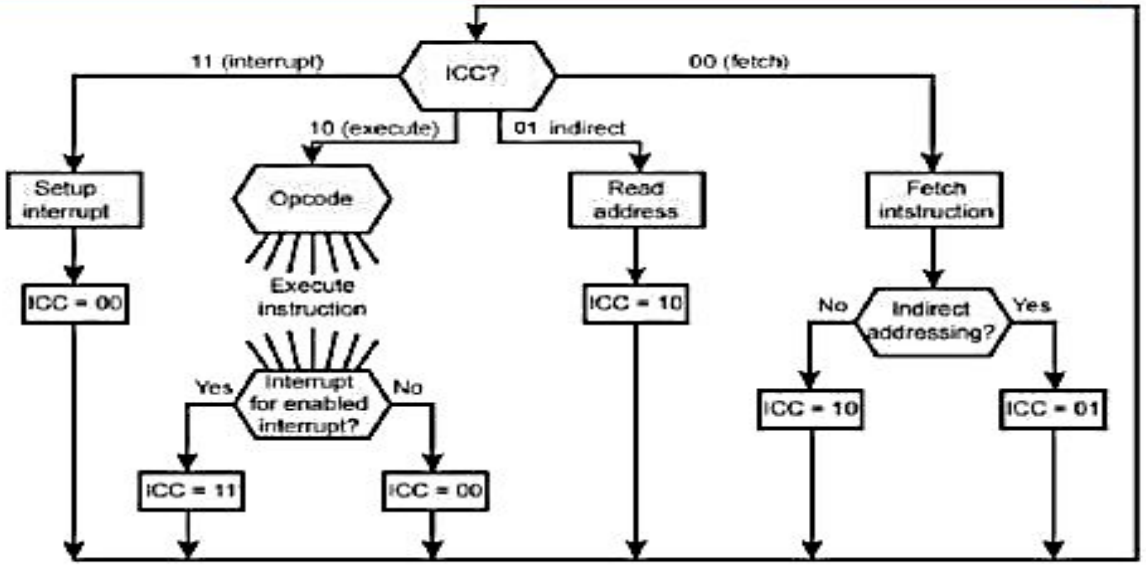
t1: MAR ← ($IR_{address}$)
MBR ← (PC)
t2: PC ← ($IR_{address}$)
memory ← (MBR)
t3: PC ← (PC) + 1

- BSA X - Branch and save address
  Address of instruction following BSA
  is saved in X

- Execution continues from X+1

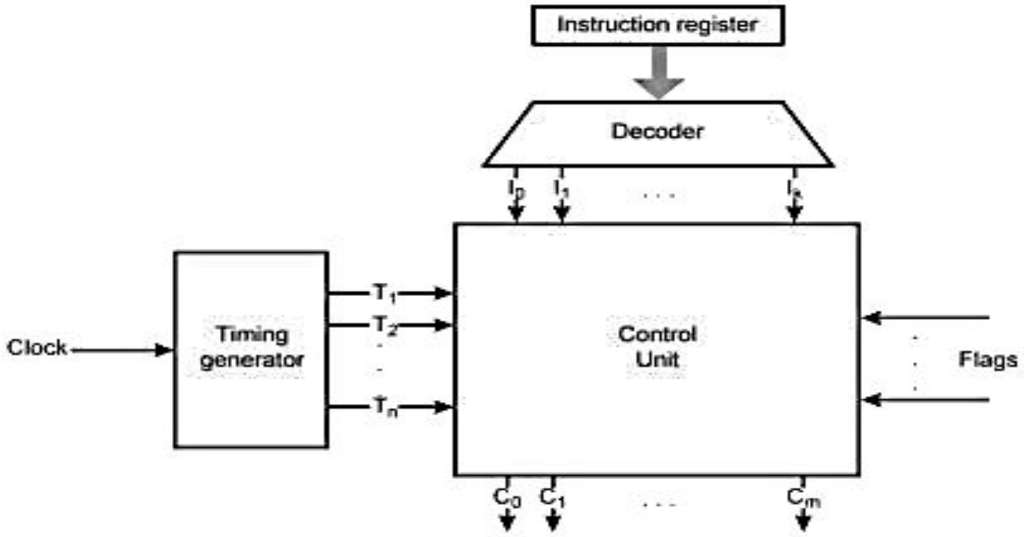
Control Signals



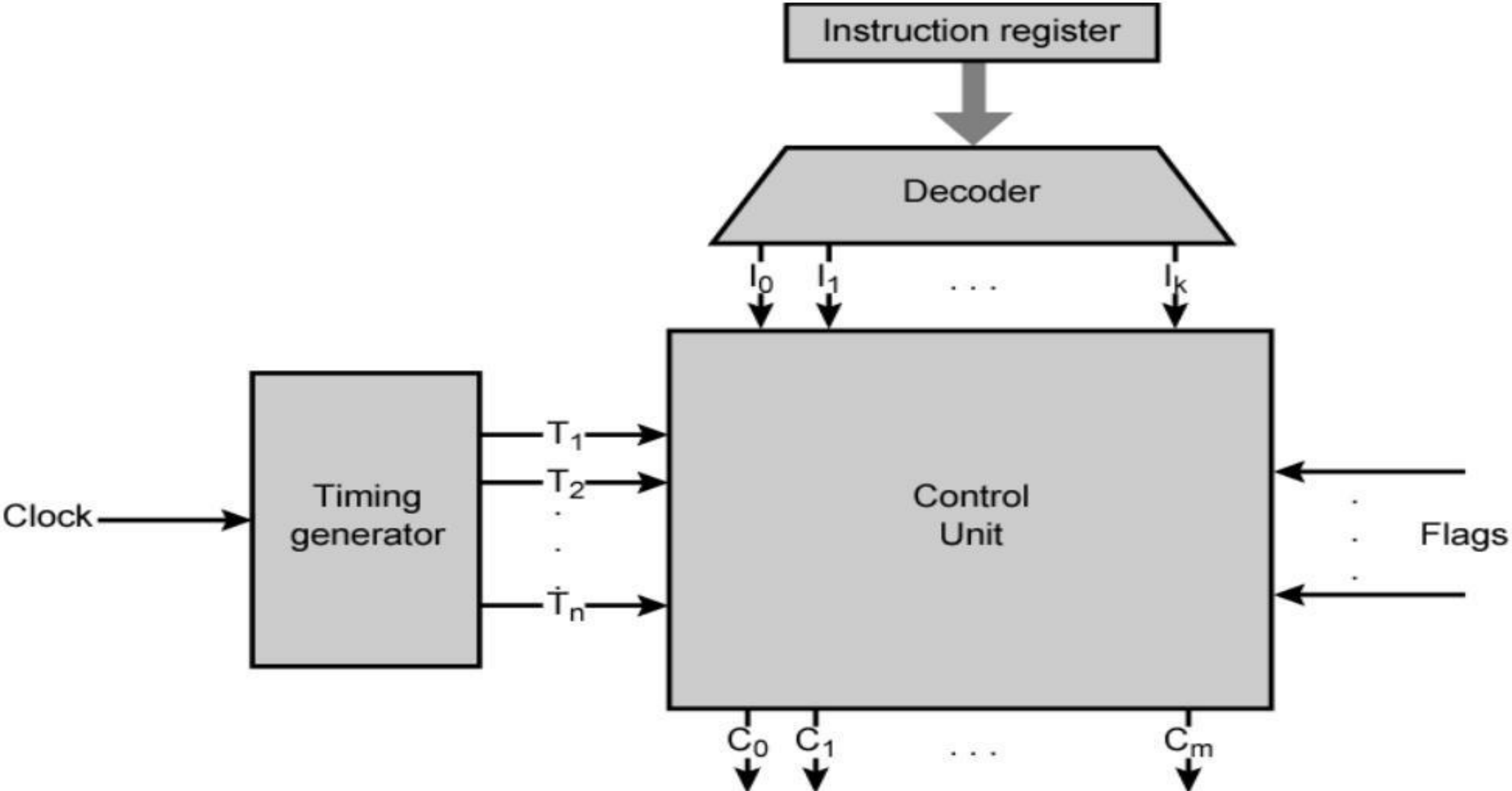


Micro-operations	Timing	Active Control Signals
Fetch:	t_1: MAR ← (PC)	C_2
	t_2: MBR ← Memory PC ← (PC) + 1	C_5, C_R
	t_3: IR ← (MBR)	C_4
Indirect:	t_1: MAR ← (IR(Address))	C_8
	t_2: MBR ← Memory	C_5, C_R
	t_3: IR(Address) ← (MBR(Address))	C_4
Interrupt:	t_1: MBR ← (PC)	C_1
	t_2: MAR ← Save-address PC ← Routine-address	
	t_3: Memory ← (MBR)	C_{12}, C_W

C_R = Read control signal to system bus.
C_W = Write control signal to system bus.



Control Unit with Decoded Inputs



Internal Organization

- Usually a single internal bus
- Gates control movement of data onto and off the bus
- Control signals control data transfer to and from external systems bus
- Temporary registers needed for proper operation of ALU

Control Unit Design

a. Hardwired

b. Micro programmed

Hardwired Implementation

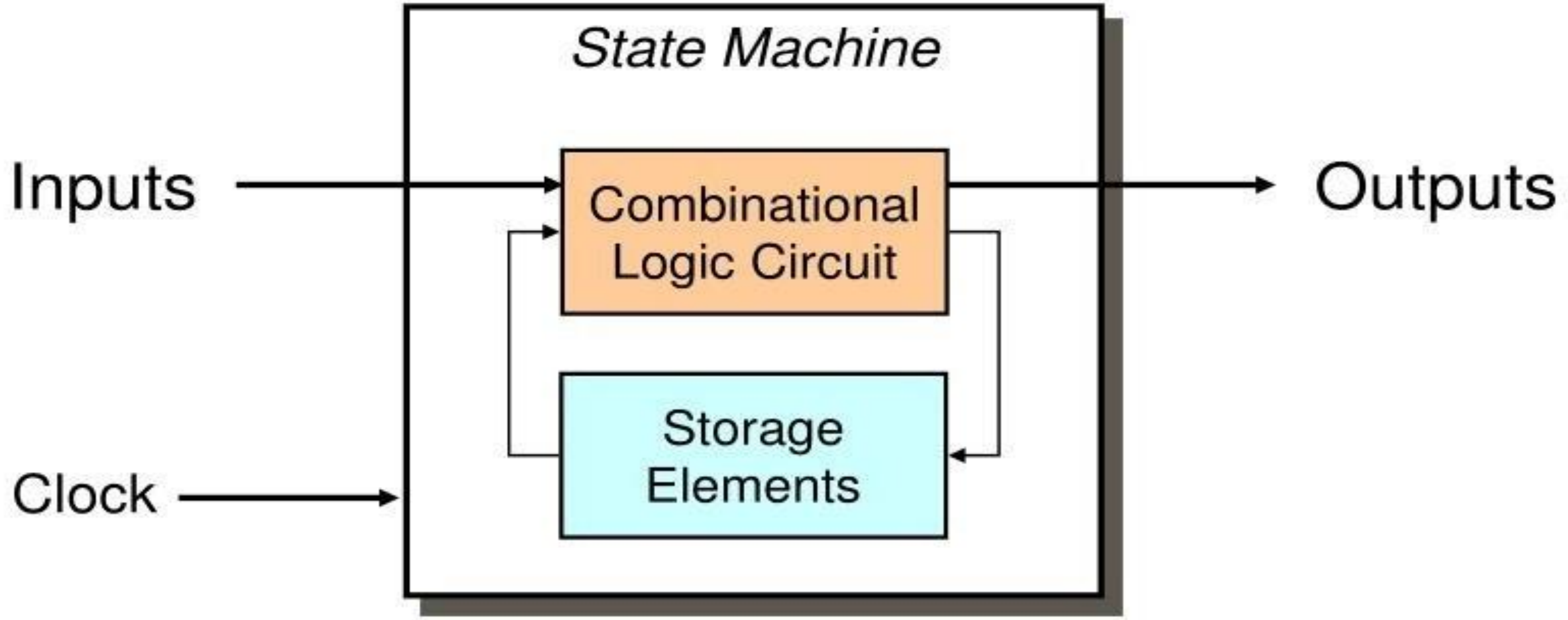
- » Control unit is viewed as a sequential logic circuit
- » Used to generate fixed sequences of control signals
- » Implemented using any of a variety of "standard" digital logic techniques
- » Principle advantages
  - ◆ High(er) speed operation
  - ◆ Smaller implementations (component counts)
- » Modifications to the design can be hard to do
- » Favored approach in RISC style designs

Hard Wired Control Unit

- The Cycles (Fetch, Indirect, Execute, Interrupt) are constructed as a State Machine
- The Individual instruction executions can be constructed as State Machines
  - Common sections can be shared. There is a lot of similarity
- One ALU is implemented. All instructions share it

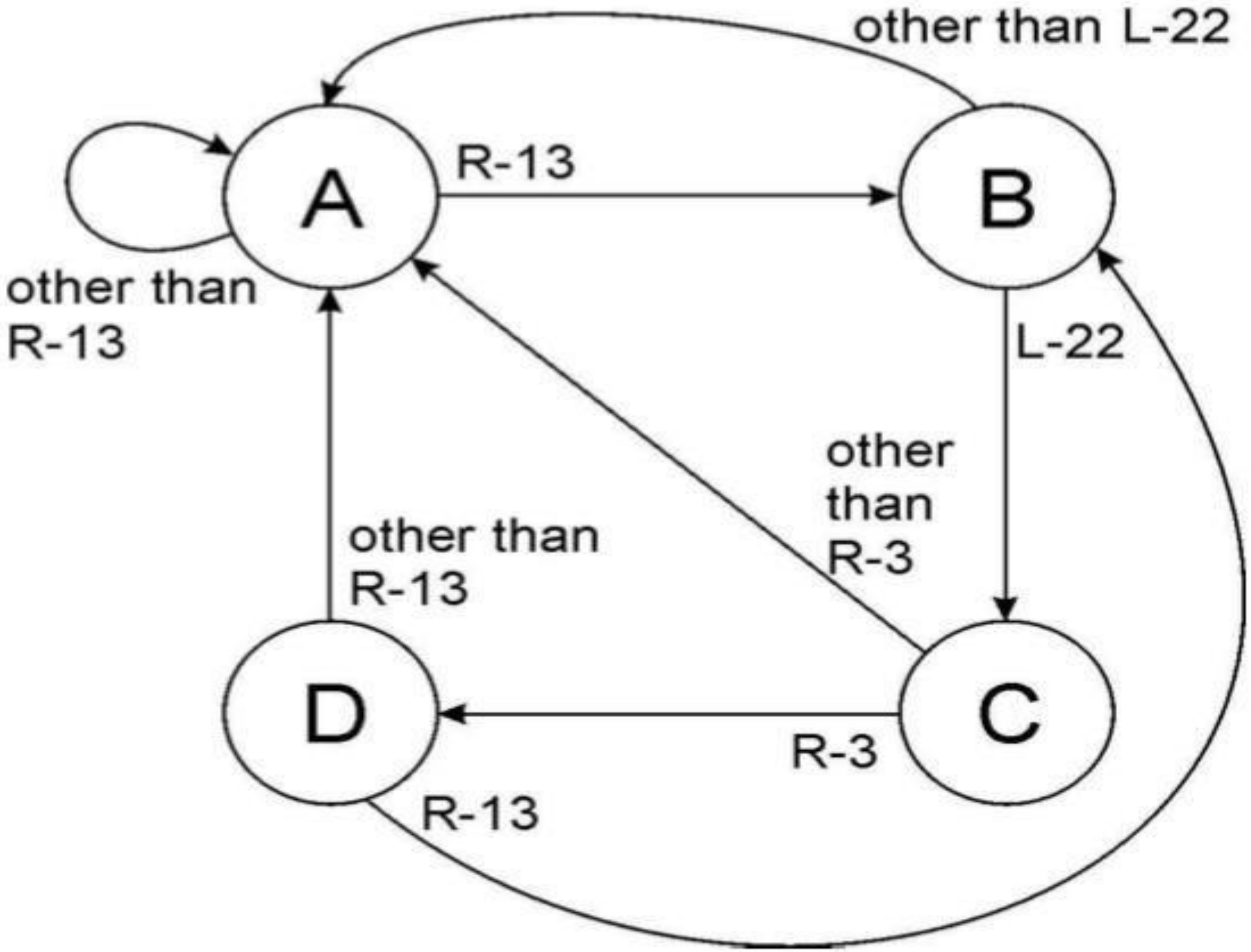
State Machine

- Combinational logic
  - Determine outputs at each state.
  - Determine next state.
- Storage elements
  - Maintain state representation.

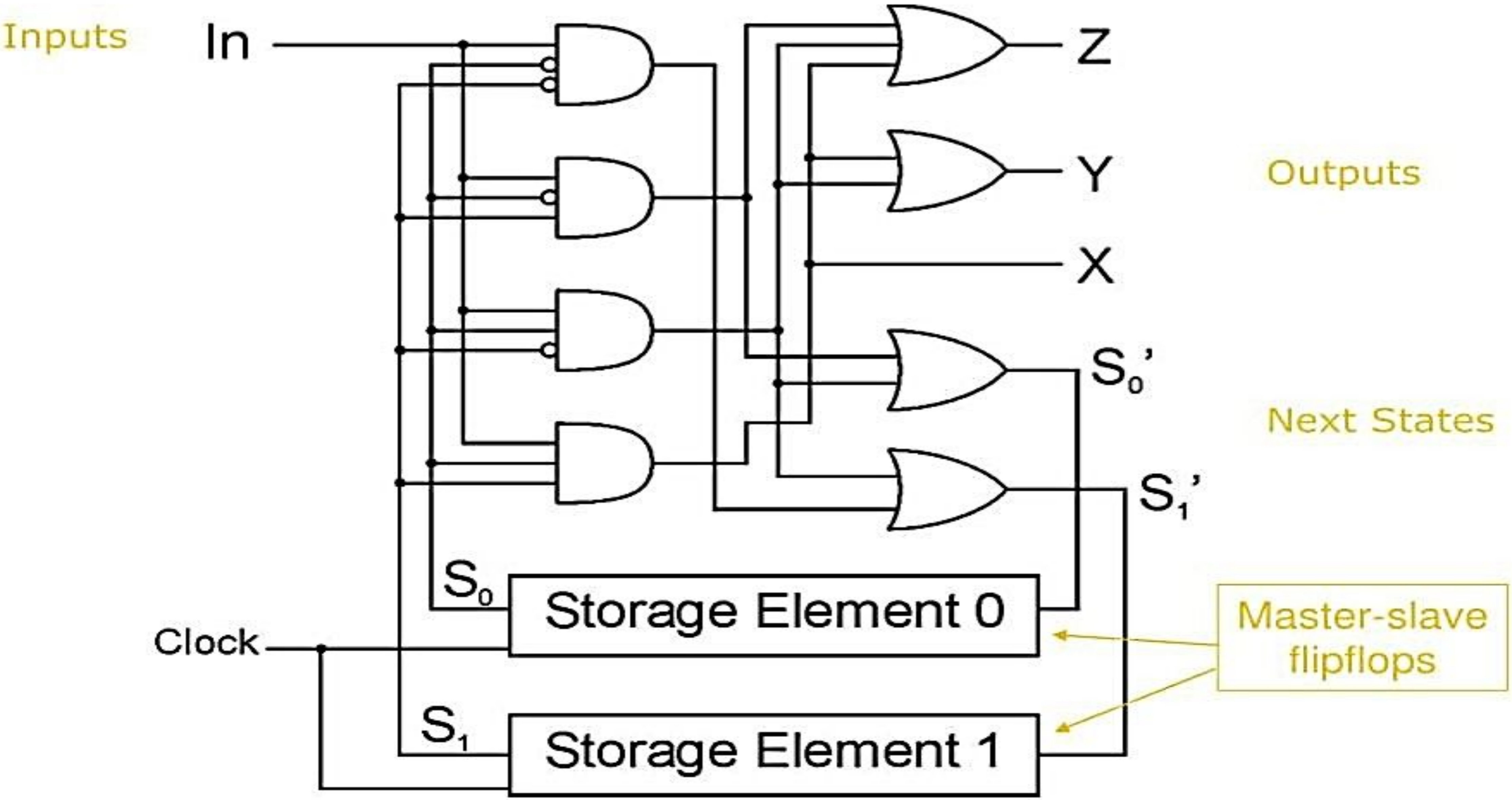


State Diagram

Shows states and actions that cause transitions between states.



Example State Machine



Problems With Hard Wired Designs

- Sequencing & micro-operation logic gets complex
- Difficult to design, prototype, and test
- Resultant design is inflexible, and difficult to build upon (Pipeline, multiple computation units, etc.)
- Adding new instructions requires major design and adds complexity quickly.

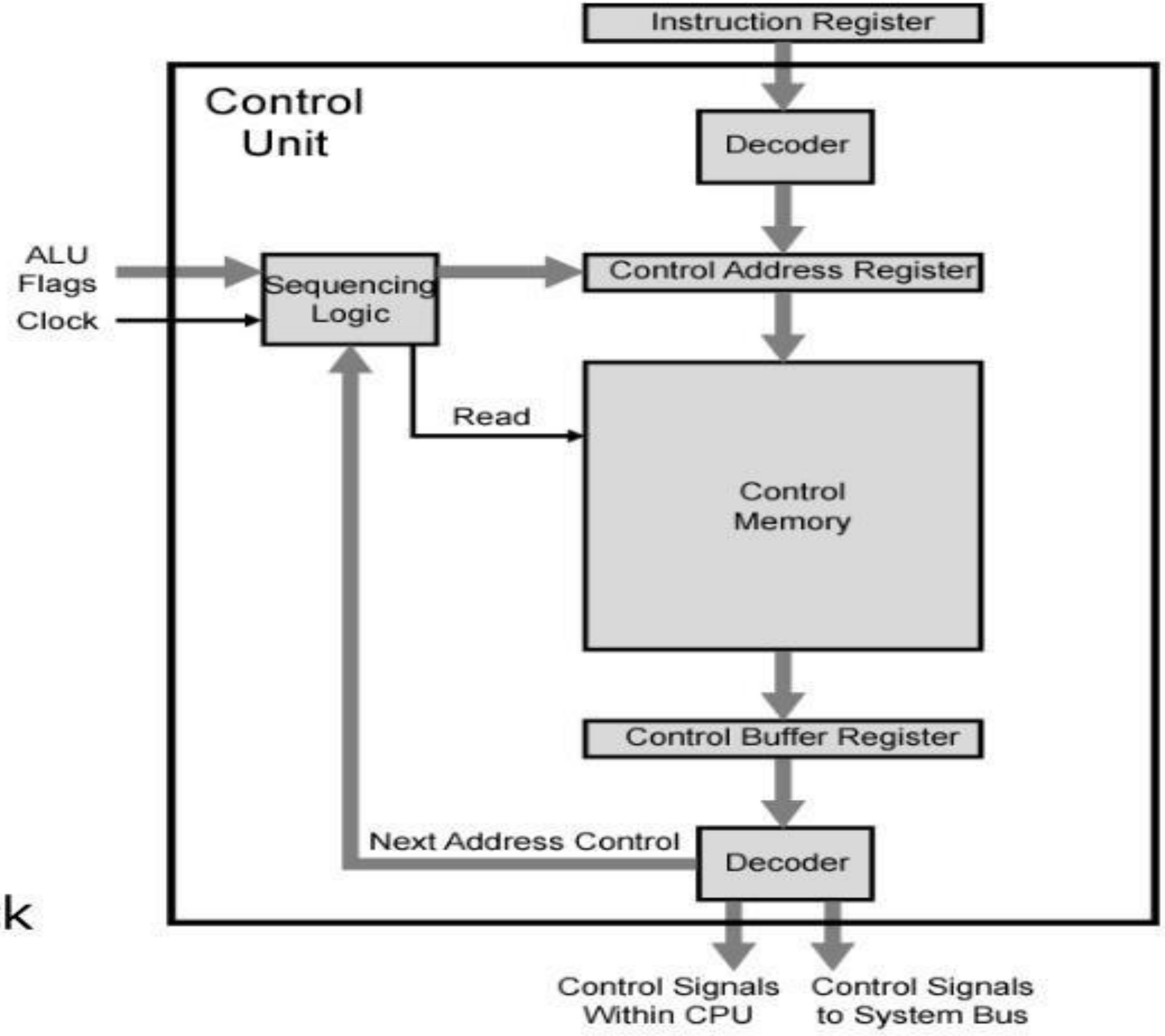
Control Unit Organization

The Control Memory contains sequences of microinstructions that provide the control signals to execute instruction cycles, e.g. Fetch, Indirect, Execute, and Interrupt.

Tasks of Control Unit:

- Microinstruction sequencing
- Microinstruction execution

May be expected to complete instruction execution in "1" clock cycle. How is this possible?



Recall: Micro-sequencing

Micro-operations	Timing	Active Control Signals
Fetch:	t_1: MAR ← (PC)	C_2
	t_2: MBR ← Memory PC ← (PC) + 1	C_5, C_R
	t_3: IR ← (MBR)	C_4
Indirect:	t_1: MAR ← (IR(Address))	C_8
	t_2: MBR ← Memory	C_5, C_R
	t_3: IR(Address) ← (MBR(Address))	C_4
Interrupt:	t_1: MBR ← (PC)	C_1
	t_2: MAR ← Save-address PC ← Routine-address	
	t_3: Memory ← (MBR)	C_{12}, C_W

C_R = Read control signal to system bus.
C_W = Write control signal to system bus.

Simple Control Memory

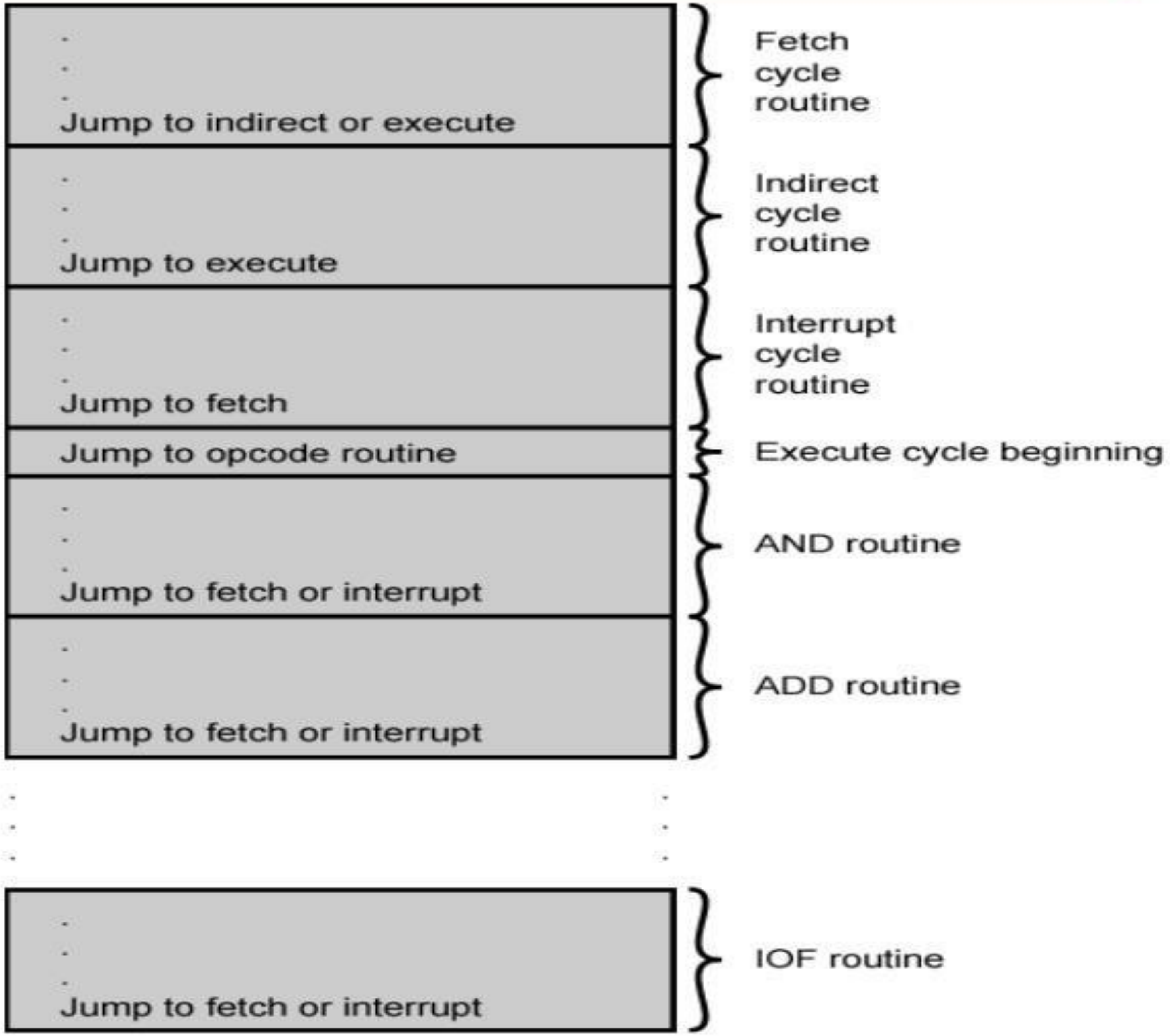
- I1-I4 Control Memory addresses
- O1-O16 Control Signals

I1	I2	I3	I4	O1	O2	O3	O4	O5	O6	O7	O8	O9	O10	O11	O12	O13	O14	O15	O16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

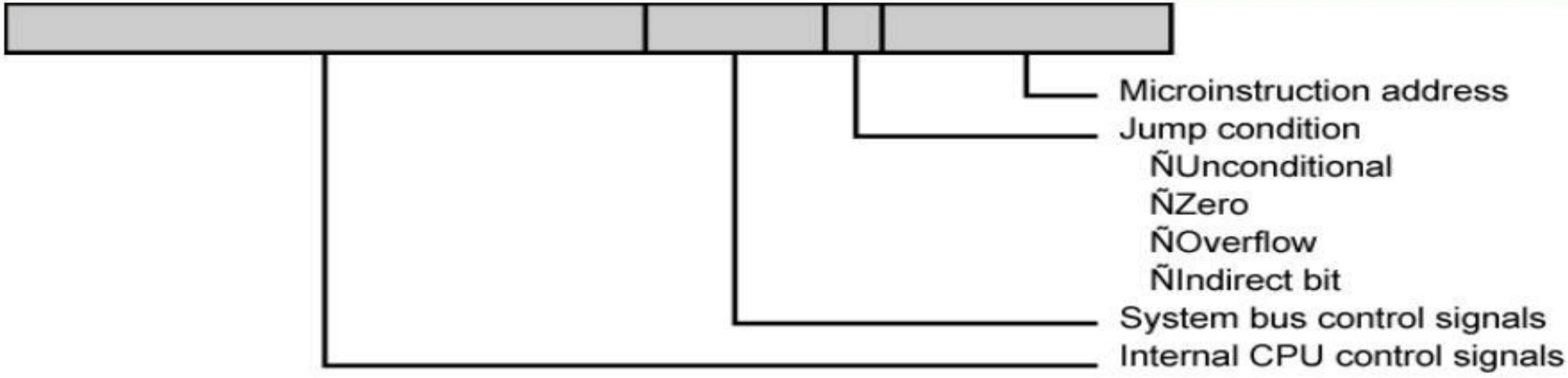
Example of Control Memory Organization

Microinstructions:

- Generate Control Signals
- Provide Branching
- Do both



Typical Microinstruction Formats



(a) Horizontal microinstruction

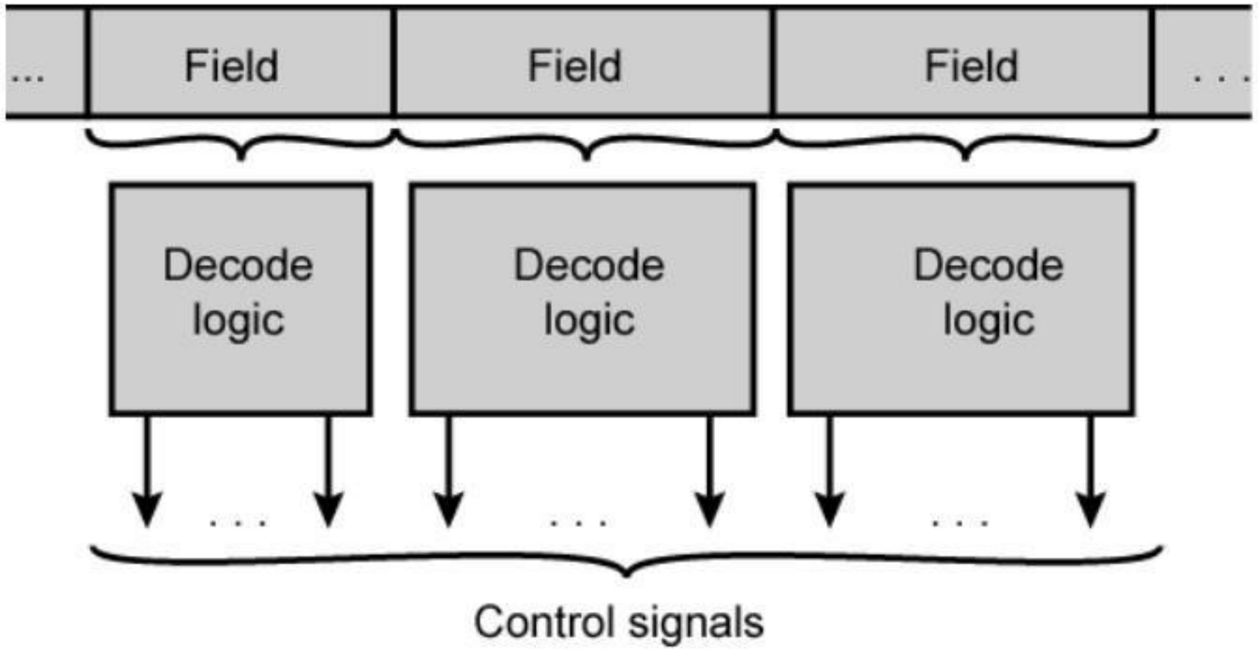
(b) Vertical microinstruction

Horizontal vs Vertical Microprogramming

Horizontal Microprogrammed
- Unpacked
- Hard
- Direct

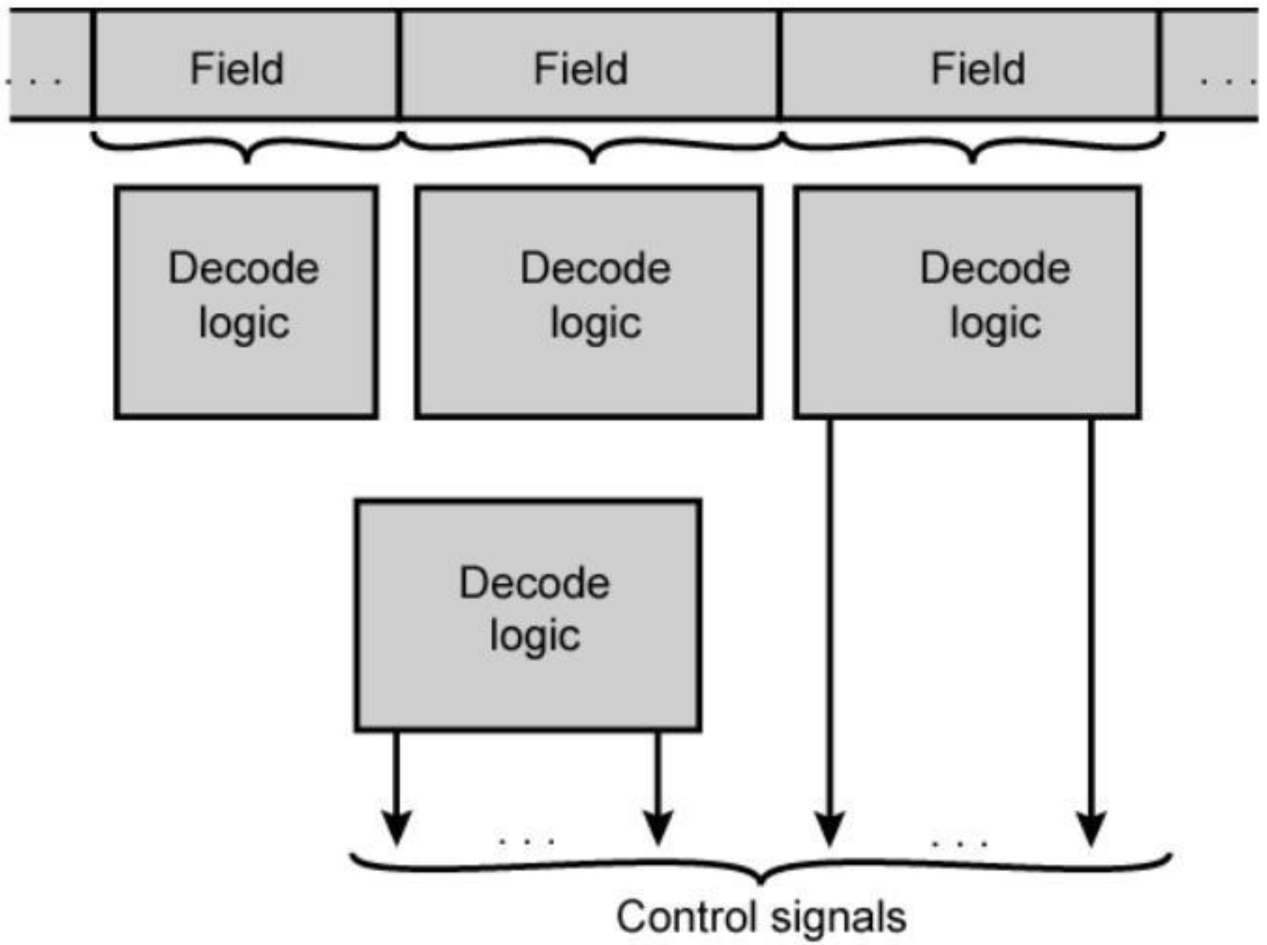
Vertical Microprogrammed
- Packed
- Soft
- Indirect

Microinstruction Encoding
Direct Encoding



(a) Direct encoding

Microinstruction Encoding
Indirect Encoding



(b) Indirect encoding

Horizontal Micro-programming

- Wide control memory word
- High degree of parallel operations possible
- Little encoding of control information
- Fast

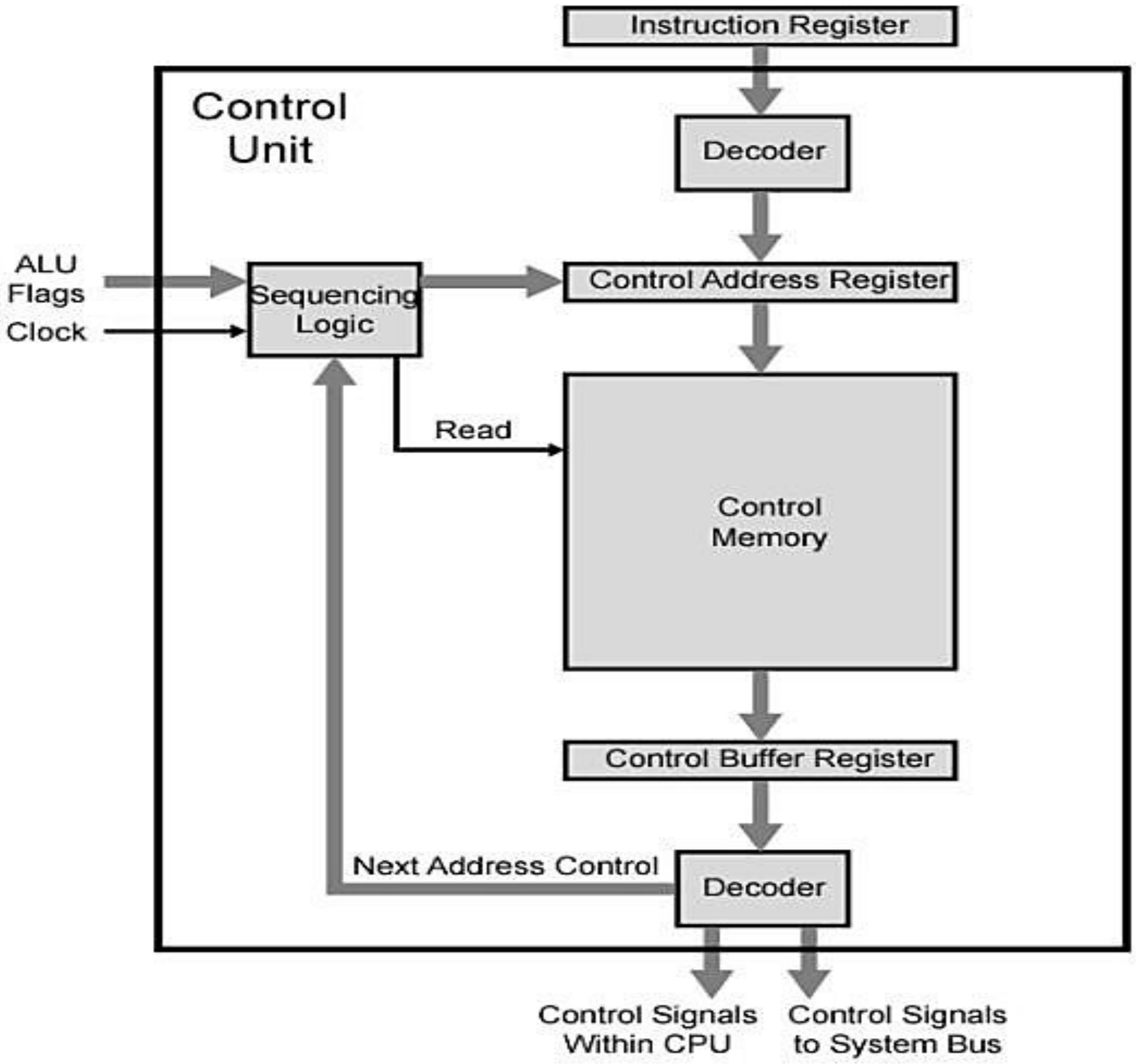
Vertical Micro-programming

- Width can be much narrower
- Control signals encoded into function codes – need to be decoded
- More complex, more complicated to program, less flexibility
- More difficult to modify
- Slower

Next Address Decision

- Depending on ALU flags and control buffer register:
  - Get next instruction
    - Add 1 to control address register
  - Jump to new routine based on jump microinstruction
    - Load address field of control buffer register into control address register
  - Jump to machine instruction routine
    - Load control address register based on opcode in IR

Microprogrammed Control Unit



Advantages and Disadvantages of Microprogramming

Advantage:

- Simplifies design of control unit
  - Cheaper
  - Less error-prone
  - Easier to modify

Disadvantage:

- Slower

Design Considerations

- Necessity of speed
- Size of microinstructions
- Address generation
  - Branches
    - Both conditional and unconditional
    - Based on current microinstruction, condition flags, contents of IR
    - Based on format of address information
      - Two address fields
      - Single address field
      - Variable format

Address Generation

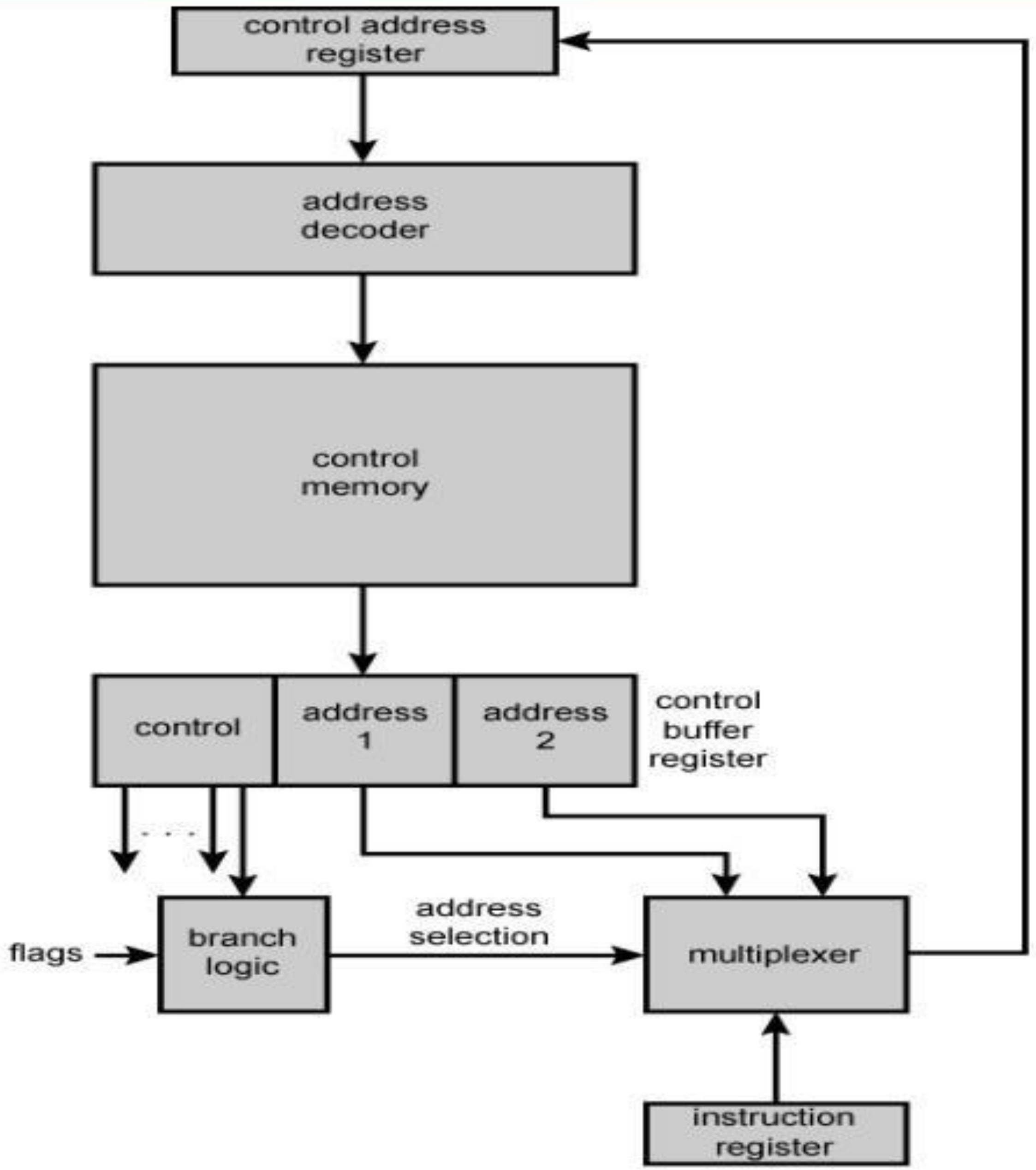
Explicit	Implicit
Two-field	Mapping
Unconditional Branch	Addition
Conditional branch	Residual control

Branch Control: Two Address Fields

Branch based upon:

- Instruction Opcode
- Address 1
- Address 2

Does require a wide microinstruction, but no address calculation is needed

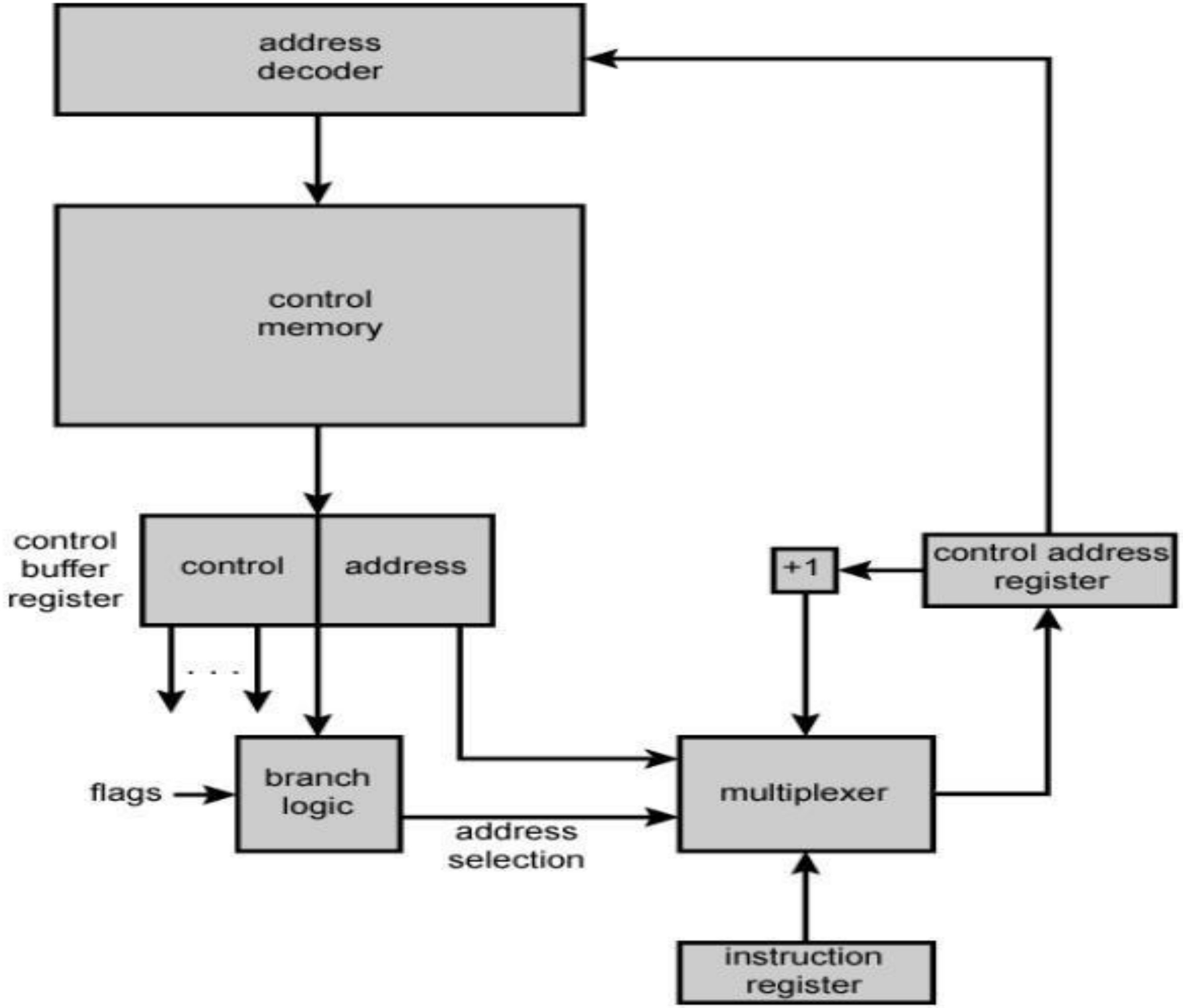


Branch Control: Single Address Field

Branch based upon:

- Next instruction
- Address
- Opcode

Does require more circuitry, e.g. adder

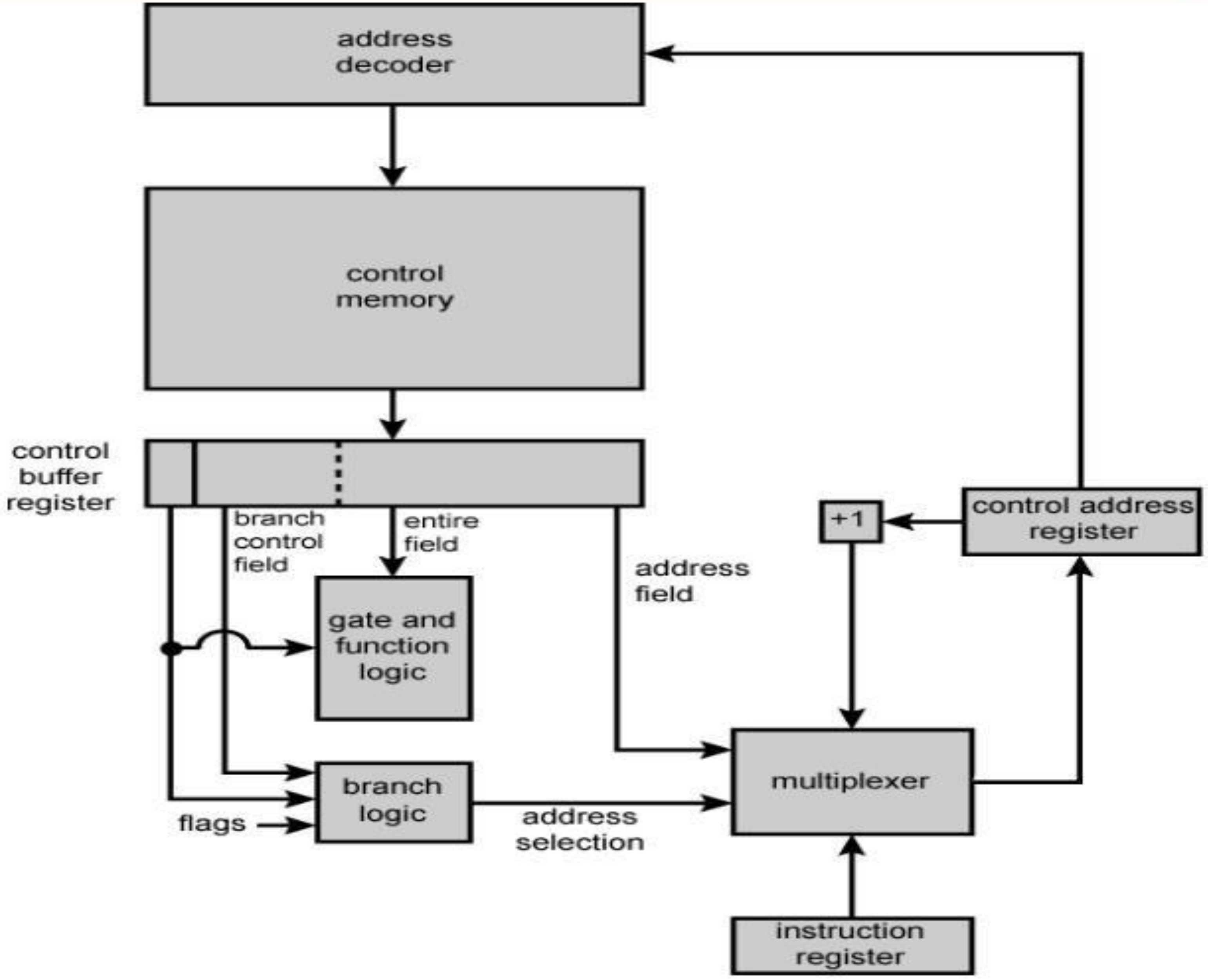


Branch Control: Variable Format

One bit determines microinstruction format:

- Control signal format
- Branch format

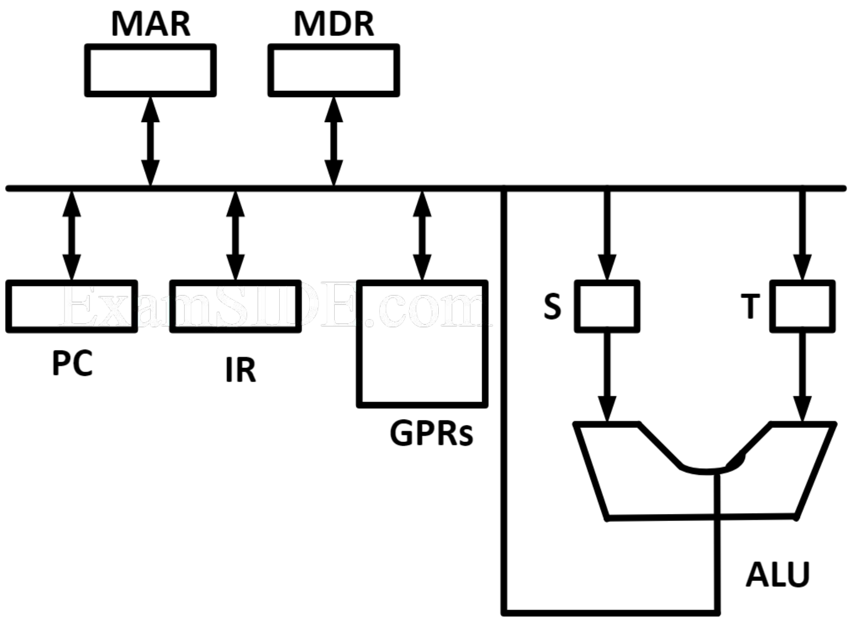
Does require even more circuitry, and is slowest.



GATE CSE 2005

Consider the following data path of a CPU

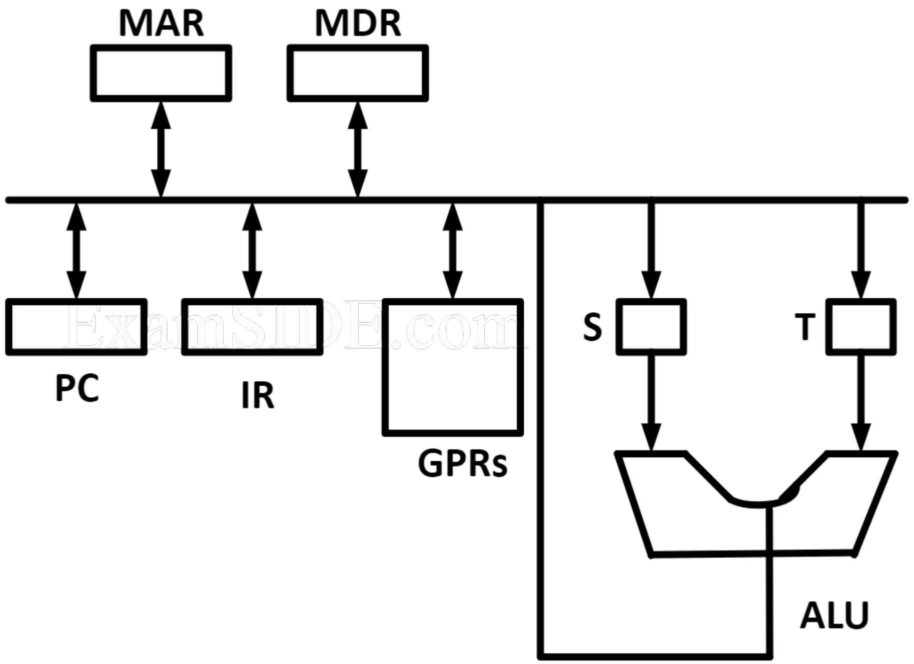
The, ALU, the bus and all the registers in the data path are of identical size. All operations including incrementation of the PC and the $GPRs$ are to be carried out in the ALU. Two clock cycle are needed for memory read operation-the first one for loading address in the MAR and the next one for loading data from the memory but into the MDR.
The instruction "add $R0$, $R1$" has the register transfer interpretation $R0$<=$R0$+$R1$. The minimum number of cycles needed for execution cycle of this instruction is



A. 2
B. 3
C. 4
D. 5

GATE CSE 2005

Consider the following data path of a CPU



The, *ALU*, the bus and all the registers in the data path are of identical size. All operations including incrementation of the *PC* and the *GPRs* are to be carried out in the *ALU*. Two clock cycle are needed for memory read operation-the first one for loading address in the *MAR* and the next one for loading data from the memory but into the *MDR*.

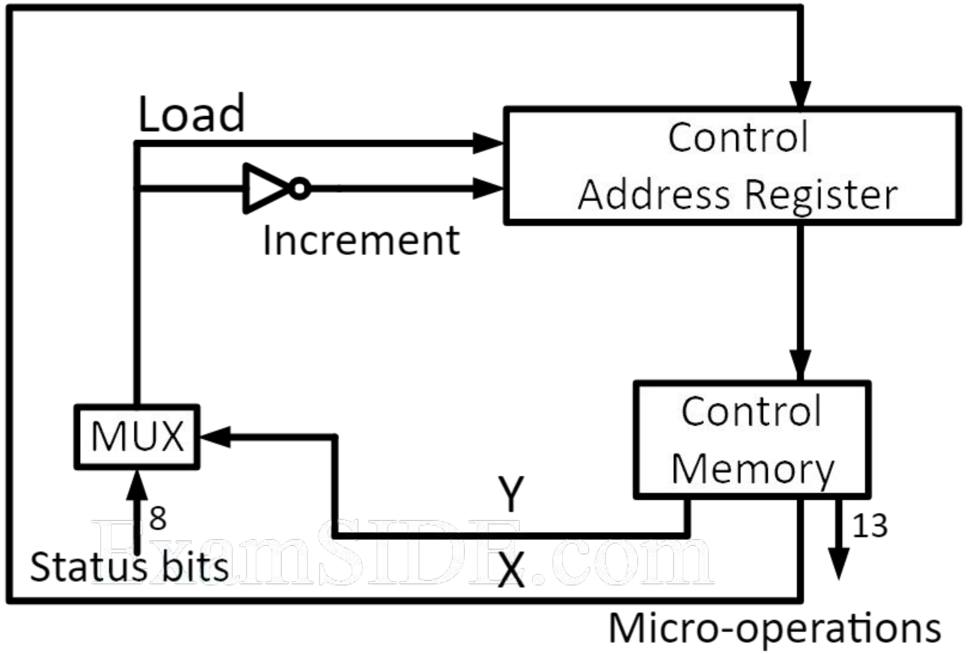
The instruction "*call Rn,sub*" is a two word instruction. Assuming that *PC* is incremented during the fetch cycle of the first word of the instruction, its register transfer interpretation is *Rn* □*PC*+1; *PC*□*M*[*PC*];
The minimum number of *CPU* clock cycles needed during the execution cycle of this instruction is :

A. 2
B. 3
C. 4
D. 5

GATE CSE 2004

The microinstructions stored in the control memory of a processor have a width of 26 bits. Each microinstruction is divided into three fields: a micro-operation of 13 bits, a next address field (X), and a MUX select field (Y). There are 8 status bits in the inputs of the MUX.

How many bits are there in the X and Y fields, and what is the size of the control memory in number of words?



A. $10, 3, 1024$
B. $8, 5, 256$
C. $5, 8, 2048$
D. $10, 3, 512$

GATE CSE 2013

Consider the following sequence of micro-operations

$MBR \leftarrow PC$

$MAR \leftarrow X$

$PC \leftarrow Y$

$Memory \leftarrow MBR$

Which one of the following is a possible operation performed by this sequence?

A. Instruction fetch
B. Operand fetch
C. Conditional branch
D. Initiation of interrupt service

GATE CSE 2001

Consider the following datapath of a simple non-pipelined *CPU*. The registers *A*, *B*, *A*1, *A*2, *MDR*, the bus and the *ALU* are 8-bit wide. *SP* and *MAR* are 16-bit registers. The *MUX* is of size 8×(2:1) and the *DEMUX* is of size 8×(1:2). Each memory operation takes 2 *CPU* clock cycles and uses *MAR* (Memory Address Register) and *MDR* (Memory Data register). *SP* can be decremented locally.

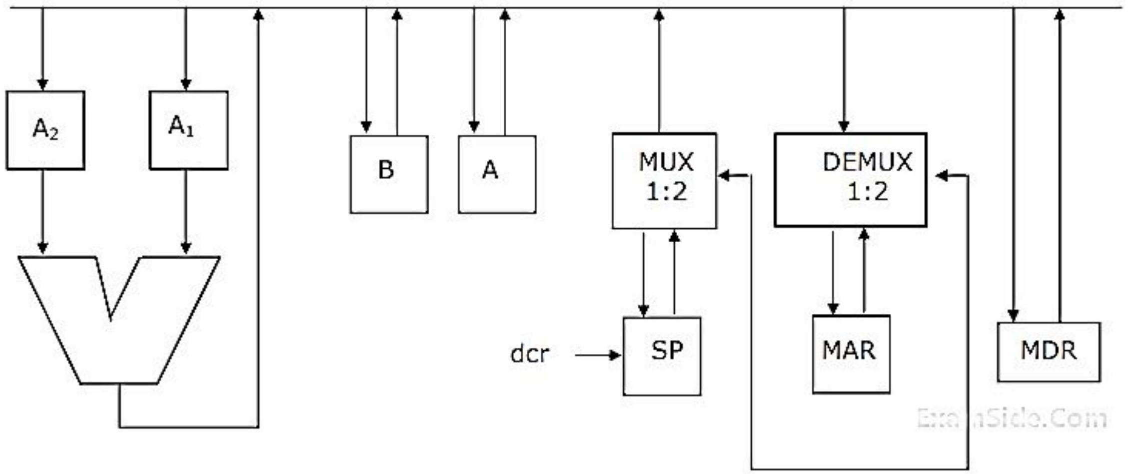
The *CPU* instruction "*push r*", where *r*=*A* or *B*, has the specification

$M[SP] \leftarrow r$

$SP \leftarrow SP - 1$

How many *CPU* clock cycles are needed to execute the "*push r*" instruction?

A. 2
B. 3
C. 4
D. 5



Horizontal micro programming

A. does not require use of signal decoders.

B. Results in larger sized micro instructions then vertical micro programming

C. Uses one bit for each control signal

D. All of the above

GATE CSE 2001, 1999

Arrange the following configuration for CPU in decreasing order of operating speeds:
Hard wired control, Vertical microprogramming, Horizontal microprogramming.

1. Hard wired control, Vertical microprogramming, Horizontal microprogramming.
2. Hard wired control, Horizontal microprogramming, Vertical microprogramming.
3. Horizontal microprogramming, Vertical microprogramming, Hard wired control.
4. Vertical microprogramming, Horizontal microprogramming, Hard wired control.

GATE CSE 2008

Consider a CPU where all the instructions require 7 clock cycles to complete execution. There are 140 instructions in the instruction set. It is found that 125 control signals are needed to be generated by the control unit. While designing the horizontal microprogrammed control unit, single address field format is used for branch control logic. What is the minimum size of the control word and control address register?

A 125, 7

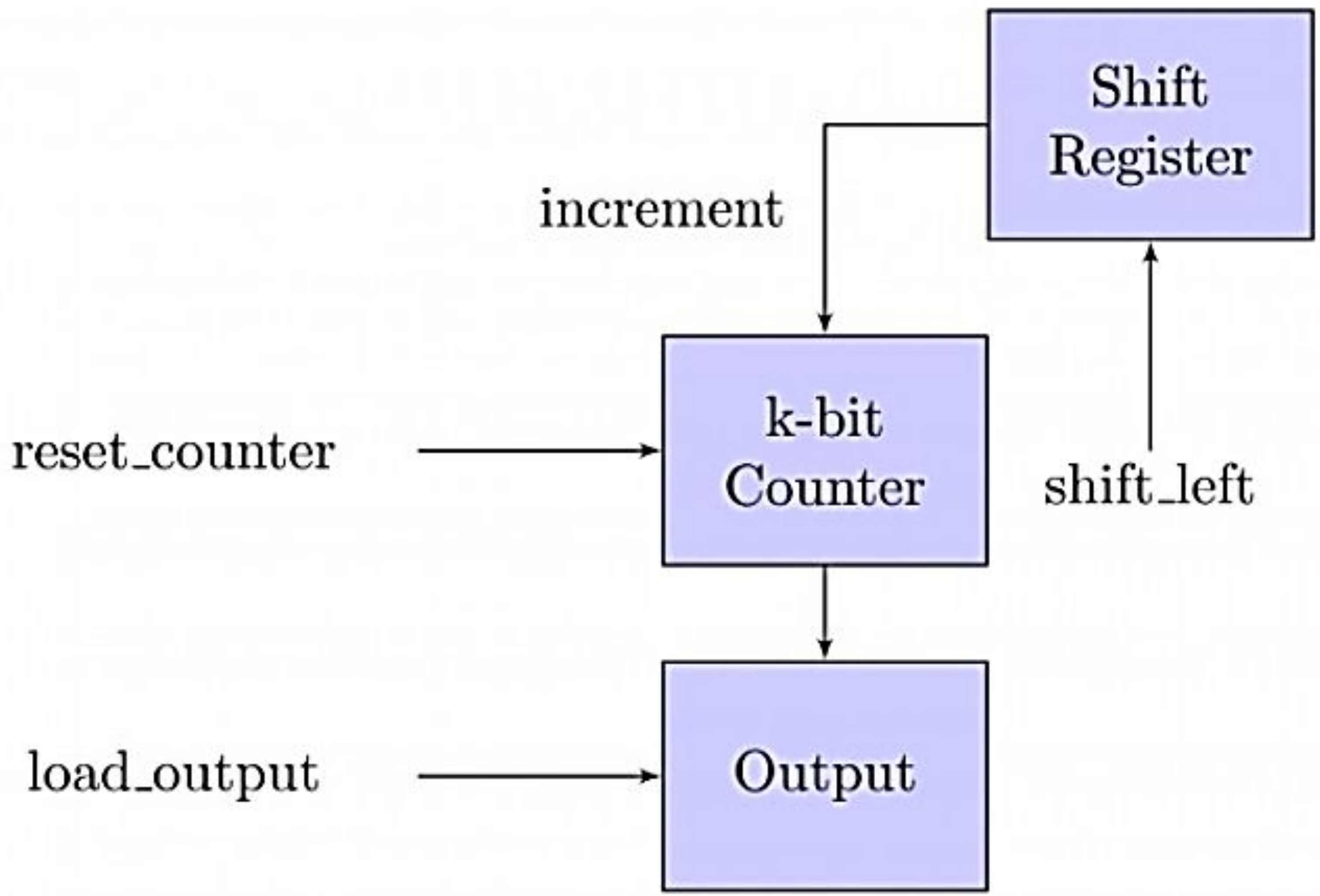
B 125, 10

C 135, 9

D 135, 10

The data path shown in the figure computes the number of 1s in the 32-bit input word corresponding to an unsigned even integer stored in the shift register.

The unsigned counter, initially zero, is incremented if the most significant bit of the shift register is 1.



The microprogram for the control is shown in the table below with missing control words for microinstructions $I_1, I_2, \ldots I_n$.

Microinstruction	Reset Counter	Shift left	Load output
BEGIN	1	0	0
I1	?	?	?
:	:	:	:
In	?	?	?
END	0	0	1

The counter width (k), the number of missing microinstructions (n), and the control word for microinstructions $I_1, I_2, \ldots I_n$ are, respectively,

A 32,5,010

B 5,32,010

C 5,31,011

D 5,31,010

GATE CSE 2005

An instruction set of a processor has 125 signals which can be divided into 5 groups of mutually exclusive signals as follows:

Group 1 : 20 signals, Group 2 : 70 signals, Group 3 : 2 signals, Group 4 : 10 signals, Group 5 : 23 signals.

How many bits of the control words can be saved by using vertical microprogramming over horizontal microprogramming?

A. 0

B. 103

C. 22

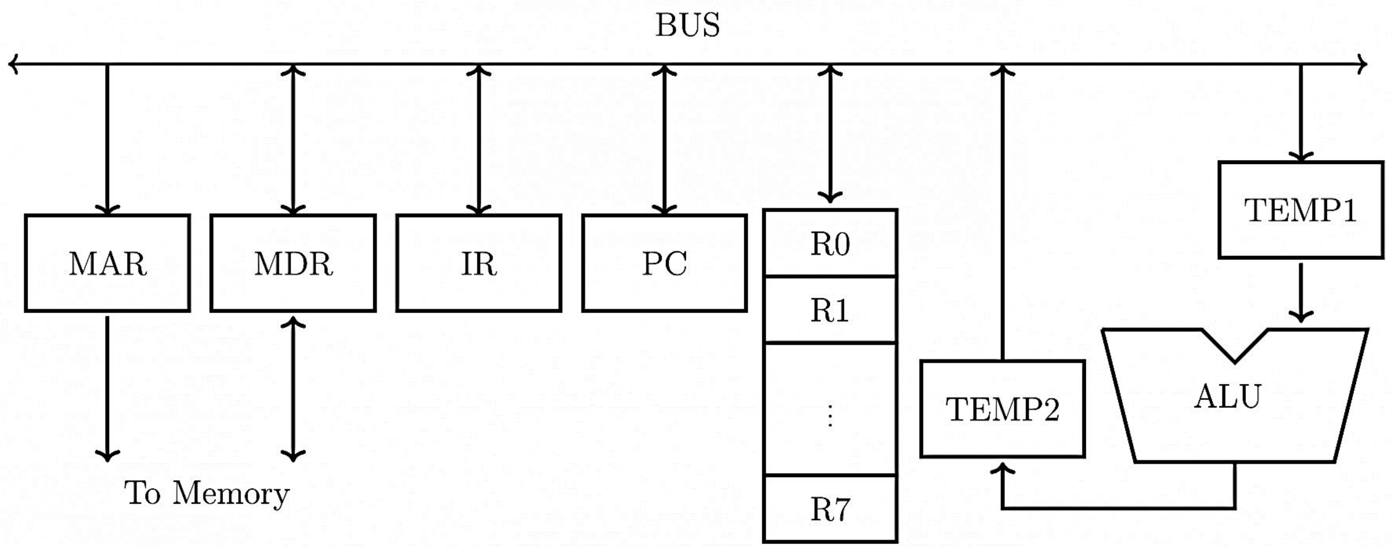
D. 55

GATE CSE 2020

Consider the following data path diagram.

Consider an instruction: $R0 \leftarrow R1 + R2$. The following steps are used to execute it over the given data path. Assume that PC is incremented appropriately. The subscripts r and w indicate read and write operations, respectively.

1. $R2_r$, TEMP1_r, ALU_{add}, TEMP2_w
2. $R1_r$, TEMP1_w
3. PC_r, MAR$_w$, MEM$_r$
4. TEMP2_r, R0_w
5. MDR$_r$, IR$_w$



Which one of the following is the correct order of execution of the above steps?

A. 2,1,4,5,3
B. 1,2,4,3,5
C. 3,5,2,1,4
D. 3,5,1,2,4