RAJIV GANDHI PROUDYOGIKI VISHWAVIDYALAYA, BHOPAL

RGPV Notes

New Scheme Based On AICTE Flexible Curricula

Information Technology, IV-Semester

IT402 - COMPUTER ARCHITECTURE

★ Course Objectives :

- The objective of course is to understand the basic structure and operation of computer system.
- Students will be able to know the operation of the arithmetic unit including the algorithms & implementation of fixed-point and floating-point addition, subtraction, multiplication & division.
- To study the different ways of communicating with I/O devices and standard I/O interfaces, hierarchical memory system including cache memories and virtual memory, concept of pipeline.

UNIT-I

Computer architecture and organization, computer generations, von Neumann model, CPU organization, CPU organization, Register organization, Various CPU register, Register Transfer, Bus and Memory Transfers, Arithmetic, Logic and Shift micro-operations, Arithmetic logic shift unit.

Topics Covered :
- Basic Structure
- CPU Organization
- Registers & Register Transfer
- Bus & Memory Transfers
- ALU and its Operations

UNIT-II

The arithmetic and logic unit, Fixed-Point representation: integer representation, sign-magnitude, 1's and 2's complement and range, Integer arithmetic: negation, addition and subtraction, multiplication, division, Floating-Point representation, Floating-Point arithmetic, Hardwired micro-programmed control unit, Control memory, Micro-program sequence.

Topics Covered :
- Number Representation
- Integer Arithmetic
- Floating-Point Arithmetic
- Control Unit Design
- Micro-program Sequence

UNIT-III

Central Processing Unit (CPU), Stack Organization, Memory Stack, Reverse Polish Notation. Instruction Formats, Zero, One, Two, Three- Address Instructions, RISC Instructions and CISC Characteristics, Addressing Modes, Modes of Transfer, Priority Interrupt, Daisy Chaining, DMA, Input-Output Processor (IOP).

Topics Covered :
- CPU & Stack Organization
- Instruction Formats
- Addressing Modes
- Interrupts & DMA
- IOP

UNIT-IV

Computer memory system, Memory hierarchy, main memory: RAM, ROM chip, auxiliary and associative memory, Cache memory: associative mapping, direct mapping, set-associative mapping, write policy, cache performance, Virtual memory: address space, memory space, address mapping, paging and segmentation, TLB, page fault, effective access time, replacement algorithm.

Topics Covered :
- Memory Hierarchy
- Cache Memory
- Virtual Memory
- Paging & Segmentation
- TLB, Page Fault & EAT

UNIT-V

Parallel Processing, Pipelining General Consideration, Arithmetic Pipeline, and Instruction Pipeline, Vector Operations, Matrix Multiplication, and Memory Interleaving, Multiprocessors, Characteristics of Multiprocessors.

Topics Covered :
- Pipelining Concepts
- Vector Operations
- Multiprocessors
- Interleaving & Matrix Mult.

Text Books :
1. M. Morris Mano - Computer System Architecture
2. William Stallings - Computer Organization and Architecture
3. John P. Hayes - Computer Architecture and Organization

Reference Books :
1. David A. Patterson - Computer Organization & Design
2. Carl Hamacher - Computer Organization
3. M. Hennessy & D. Patterson - Computer Architecture

★ Understand Concepts Clearly, Practice Regularly and Score Excellent ★

IT402 - COMPUTER ARCHITECTURE

RGPV Notes

UNIT-I

9. ARITHMETIC MICRO-OPERATIONS

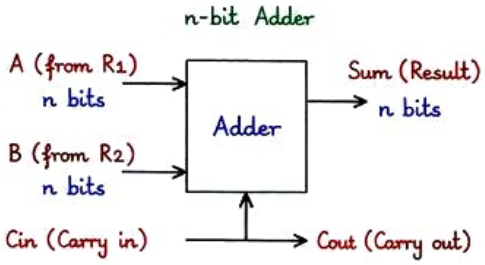
Arithmetic micro-operations are operations performed on binary numbers. These operations are used in ALU to perform arithmetic calculations. The basic arithmetic micro-operations are : Addition, Subtraction, Increment, Decrement, Multiplication and Division.

1. Addition

Addition is performed using an adder circuit. Two binary numbers are added and the result is stored in the destination register. A carry may be generated.

Symbolic representation :

$R3 \leftarrow R1 + R2$



Example (4-bit) :

R1 = 0101 (5)
R2 = 0011 (3)

Sum = 1000 (8)
Cout = 0

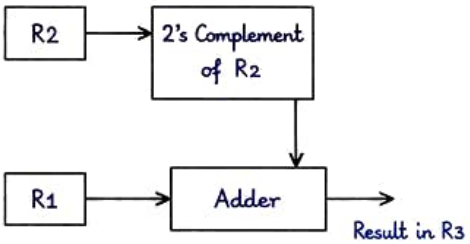
2. Subtraction

Subtraction can be performed by using 2's complement method. R1 - R2 is done by adding 2's complement of R2 to R1.

Symbolic representation :

$R3 \leftarrow R1 - R2$

Using 2's Complement



Example (4-bit) :

R1 = 0101 (5)
R2 = 0011 (3)
2's Compl. of R2 = 1101
Add → 0101
+ 1101
Result = 1 0010
Ignore carry → 0010 (2)
So, 5 - 3 = 2

3. Increment

Increment operation increases the content of a register by 1.

Symbolic representation :

$R1 \leftarrow R1 + 1$

Example : If R1 = 0111 (7)
After Increment → 1000 (8)

4. Decrement

Decrement operation decreases the content of a register by 1.

Symbolic representation :

$R1 \leftarrow R1 - 1$

Example : If R1 = 0111 (7)
After Decrement → 0110 (6)

5. Multiplication

Multiplication is repeated addition. It requires more than one clock cycle.

Algorithm (Unsigned) :

- Add multiplicand to accumulator if multiplier LSB = 1.
- Shift multiplicand left.
- Shift multiplier right.
- Repeat for n times (n = number of bits).

Symbolic representation :

$R3 \leftarrow R1 \times R2$

Example (4-bit) :

R1 = 0011 (3)
R2 = 0101 (5)

3 x 5 = 15
Result (8-bit) = 0000 1111

6. Division

Division is repeated subtraction (restoring algorithm).

Algorithm (Unsigned) :

- Compare divisor with dividend.
- If dividend ≥ divisor, subtract divisor from dividend and set quotient bit = 1.
- Else, set quotient bit = 0.
- Bring down next bit and repeat.
- Repeat until all bits are processed.

Symbolic representation :

$R3 \leftarrow R1 \div R2$ (Quotient in R3)
(Remainder in R4)

Summary : Arithmetic micro-operations are the basic operations performed by ALU. They include addition, subtraction, increment, decrement, multiplication and division. These operations are used in instruction execution, address calculation and data processing.

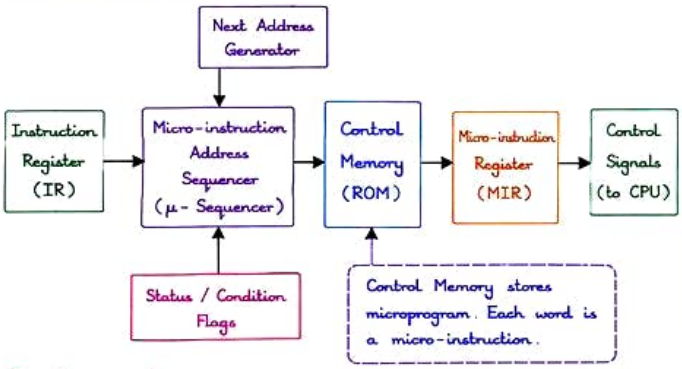
14. MICROPROGRAMMED CONTROL UNIT

Microprogrammed control unit uses control memory (microprogram) to generate control signals. It is flexible and easy to modify. ★

1. Definition :

A microprogrammed control unit is a control unit in which control signals are generated by executing a microprogram stored in a control memory (ROM). Each micro-instruction in the control memory generates the required control signals.

2. Block Diagram :



3. Components :

- Control Memory (ROM) : Stores microprogram (micro-instructions).
- Micro-instruction Register (MIR) : Holds the current micro-instruction.
- Micro-instruction Address Sequencer (μ-Sequencer) : Determines the address of next micro-instruction.
- Next Address Generator : Generates next address based on sequencing (sequential, conditional, branching).
- Status / Condition Flags : Provide condition information (Zero, Carry, Sign, Overflow, etc.).

4. Micro-instruction Format :

Control Field (Control Signals)	Condition Select	Address Field	Next Address Control
n bits	c bits	a bits	k bits

- Control Field : Specifies the control signals to be generated.
- Condition Select : Selects the status flag for testing.
- Address Field : Specifies the address of the branch target.
- Next Address Control : Specifies the type of sequencing (Sequential, Branch, Conditional Branch, Call, Return).

5. Working Principle :

1. When an instruction is fetched, the opcode is placed in IR.
2. The opcode is used to obtain the starting address of the microprogram.
3. The μ-sequencer supplies this address to the control memory.
4. The micro-instruction is read into MIR.
5. The control signution in MIR are sent to the CPU to perform micro-operations.
6. The next address is determined by μ-sequencer based on sequencing and condition flags.
7. Steps 4 to 6 repeat until the microprogram for the instruction is complete.

Example (Pseudo Sequence) : ADD R1, R2

Step	Micro-operation	Control Signals (Examples)
μ0	AR ← IR(address)	AR_in
μ1	DR ← M[AR]	Mem_read, DR_in
μ2	AC ← R1	R1_out, AC_in
μ3	AC ← AC + DR	Add, AC_in
μ4	R1 ← AC	AC_out, R1_in
μ5	SC ← 0	(End of instruction)

(Actual micro-operations depend on the architecture) ★

6. Sequencing of Micro-instructions :

- Sequential Addressing : Next μ-instruction is at the next address.
- Conditional Branching : Next address depends on condition flags.
- Unconditional Branching : Next address is given in address field.
- Call : Saves return address and branches to a subroutine.
- Return : Returns to the calling micro-instruction.

Sequencing Types	Description
Sequential	Next address = Current address + 1
Conditional Branch	If condition true → Branch to address else → Next sequential
Unconditional Branch	Always branch to given address
Call	Branch to subroutine and save return address
Return	Return to the saved return address

7. Advantages and Disadvantages :

Advantages	Disadvantages
• Easy to modify and update. • Suitable for complex instruction sets. • Control memory can be tested and verified separately. • Shorter development time. • Flexibility in adding new instructions.	• Slower than hardwired control unit (due to control memory access). • Requires control memory (more hardware). • Higher cost for large control store.

8. Hardwired vs Microprogrammed Control Unit

Feature	Hardwired Control Unit	Microprogrammed Control Unit
Implementation	Fixed logic circuits	Control memory (microcode)
Speed	Fast	Slower (memory access time)
Flexibility	Difficult to modify	Easy to modify
Design Time	Longer	Shorter
Hardware Complexity	Higher for complex ISAs	Lower for complex ISAs
Best Suitable For	RISC, Simple ISAs	CISC, Complex ISAs

9. Applications :

⇒ Used in CISC processors (e.g., x86, VAX, 68000).
⇒ Used where instruction sets are large and complex.
⇒ Used in systems where easy modification is required.
⇒ Used in microcontrollers and embedded systems.

10. Important Points (RGPV Exam)

★ Microprogrammed control unit stores microprogram in control memory.
★ Each micro-instruction generates the required control signals.
★ μ-Sequencer decides the next address of micro-instruction.
★ More flexible but slower than hardwired control unit.
★ Main components - Control Memory, MIR, μ-Sequencer, Next Address Generator, Status Flags.
★ Suitable for complex instruction sets. ★

Summary : Microprogrammed control unit uses a control memory to store microprograms. It generates control signals by executing micro-instructions. It is flexible, easy to modify and suitable for complex instruction sets, but slower than hardwired control unit. ★

UNIT-III

IT402 – COMPUTER ARCHITECTURE

RGPV NOTES

2. RISC ARCHITECTURE

Definition :

RISC (Reduced Instruction Set Computer) architecture is a type of CPU design that uses a small set of simple instructions which can be executed very quickly. These instructions are designed to perform basic operations and are executed in one clock cycle.

Basic Idea :

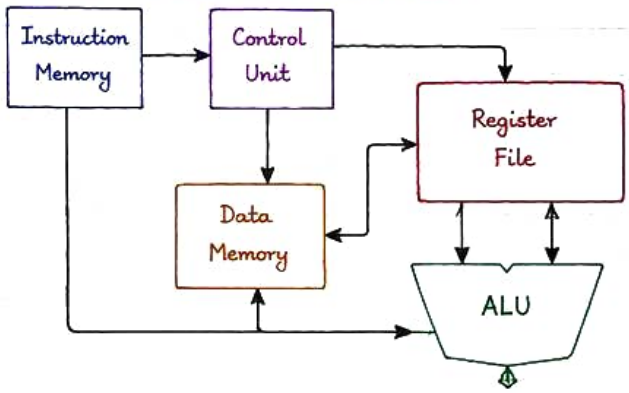
Do more work with hardware and less with software.

→ Simple instructions
→ Execute in single clock cycle
→ More registers, less memory access.

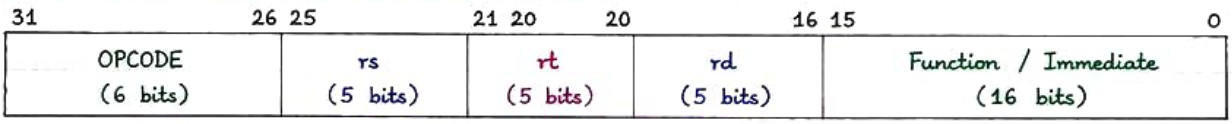
Key Features :

① Small and simple instruction set.
② Fixed length instructions.
③ Load / Store architecture (data only in registers).
④ Large number of general purpose registers.
⑤ One instruction executed in one clock cycle.
⑥ Hardwired control unit.
⑦ Pipelining can be used effectively.
⑧ Optimized for compiler.

RISC Architecture Block Diagram :



Instruction Format (Typical RISC Format) :



Note : Fixed length instruction (32 bits)

Advantages of RISC Architecture :

- High speed execution (most instructions in one clock cycle).
- Simple and regular instruction set.
- Easy to pipeline → higher throughput.
- Large number of registers → reduces memory access.
- Compilers can optimize code efficiently.
- Less hardware complexity in control unit.

Disadvantages of RISC Architecture :

- More instructions may be needed to perform complex operations.
- Code size may increase.
- Requires more registers (more chip area).
- Not suitable for applications requiring very complex instructions.

Examples of RISC Processors :

ARM, MIPS, SPARC, PowerPC, RISC-V

RISC vs CISC (Quick Comparison)

Feature	RISC	CISC
Instruction Set	Small and simple	Large and complex
Instruction Length	Fixed	Variable
Addressing Modes	Few	Many
Execution Time	One clock cycle	Multi clock cycles
Control Unit	Hardwired	Microprogrammed
Registers	Large number	Small number
Memory Access	Only through Load / Store	Any instruction can access memory
Pipelining	Easy	Difficult
Hardware Complexity	Less	More
Examples	ARM, MIPS, RISC-V	Intel x86, 8086, VAX

Important Points (Exam)

★ RISC = Reduced Instruction Set Computer.
★ RISC uses small and simple instructions.
★ Most instructions execute in one clock cycle.
★ RISC uses many registers and pipelining technique.

Summary : RISC architecture is based on the principle of using a small set of simple instructions that can be executed in one clock cycle. It is fast, efficient and suitable for modern high performance systems.

2. DIRECT MAPPING

Definition :

In direct mapping, each block of main memory can be placed in only one specific line (or block) of cache memory. The location is determined by using the lower order bits of the memory address.

Key Points :

- Simple and easy to implement.
- Less hardware cost.
- Fast access.
- But more conflict misses (compared to associative mapping).

Address Format :

Tag (t bits)	Index (k bits)	Block Offset (b bits)

$t + k + b$ = Total number of address bits

Mapping Function :

Cache Line (Index) = (Block Number) mod (Number of Cache Lines)

or

$$\text{Index} = (\text{Block Number}) \bmod (2^k)$$

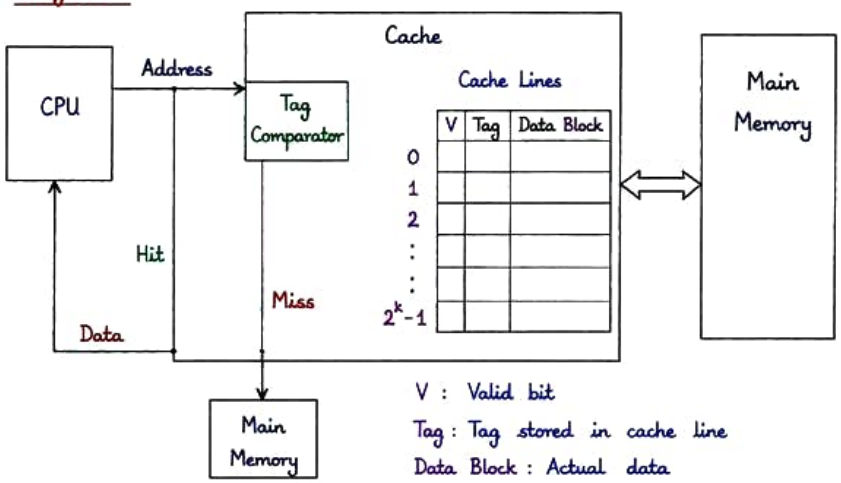
where,

Number of Cache Lines = 2^k

Working :

1. CPU generates a memory address.
2. The Index bits select a cache line.
3. The Tag bits of the address are compared with the Tag stored in that cache line.
4. If Tag matches and Valid bit = 1 ⇒ Hit. Data is sent to CPU.
5. If not match ⇒ Miss. The required block is fetched from main memory and placed in that cache line (replacing old data).

Diagram :



V : Valid bit

Tag : Tag stored in cache line

Data Block : Actual data

Example :

Main Memory = 64 KB (2^{16} bytes)

Block Size = 16 Bytes (2^4) ⇒ b = 4 bits

Cache Size = 4 KB (2^{12} bytes)

Number of Cache Lines = Cache Size / Block Size

$= 2^{12} / 2^4 = 2^8 = 256$ lines ⇒ k = 8 bits

Tag bits, t = 16 - (k + b) = 16 - (8 + 4) = 4 bits

Address Format → | Tag (4 bits) | Index (8 bits) | Block Offset (4 bits) |

Mapping Example :

Let Memory Block Number = 105

Cache Lines = 256

Index = 105 mod 256 = 105

⇒ Block 105 will be placed in Cache Line number 105.

Line No.	Stores Block No.
0	0, 256, 512, ...
1	1, 257, 513, ...
2	2, 258, 514, ...
:	:
255	255, 511, 767, ...

Advantages :

- ✓ Very easy to design.
- ✓ Requires less hardware.
- ✓ Fast access time.
- ✓ Suitable for small cache systems.

Disadvantages :

- × High conflict misses.
- × When many blocks map to same line, performance degrades.
- × Not efficient for large programs.

Used In :

- Small and low cost systems.
- Instructions cache (I-Cache) in many processors.
- Systems where simplicity is more important.

Summary : In Direct Mapping, each memory block is placed in only one specific cache line using Index = Block Number mod (Number of Cache Lines). It is simple, fast and low cost but has more conflict misses.

IT402 – COMPUTER ARCHITECTURE

10. PAGE FAULT

Definition :

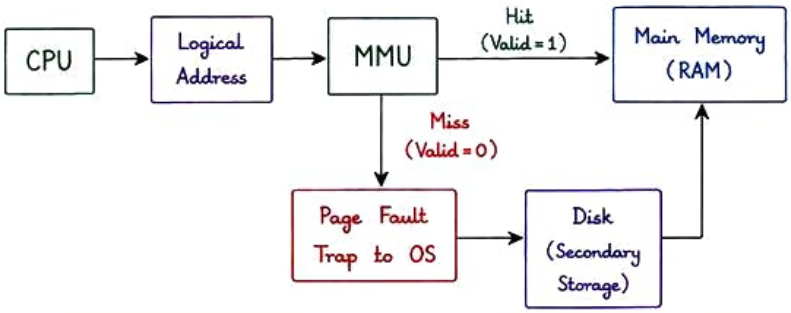
A Page Fault occurs when a process tries to access a page that is not present in main memory (RAM). This causes a trap to the operating system. The required page is then brought from secondary storage (disk) into main memory.

Causes of Page Fault :

- The page is not in main memory.
- The page table entry has Valid/Invalid bit = 0.
- First reference to a page (compulsory miss).
- Page was swapped out to make space.

How Page Fault is Handled ?

1. CPU generates logical address.
2. MMU checks page table.
3. If Valid bit = 1 → Page Hit (normal access).
4. If Valid bit = 0 → Page Fault Trap to OS.
5. OS brings required page from disk to a free frame.
6. Page table is updated and instruction is restarted.



When Page Fault Occurs ?

- When a referenced page is not in main memory.
- The valid/invalid bit of that page table entry is 0.
- Hardware generates an interrupt (trap) to the OS.

Page Table Entry Format :

Frame Number	Valid/Invalid Bit	Protection Bits	Reference Bits	Dirty Bit	Other Bits
n bits	1 bit (1 = Valid, 0 = Invalid)	r/w/x etc.	0/1	0/1	...

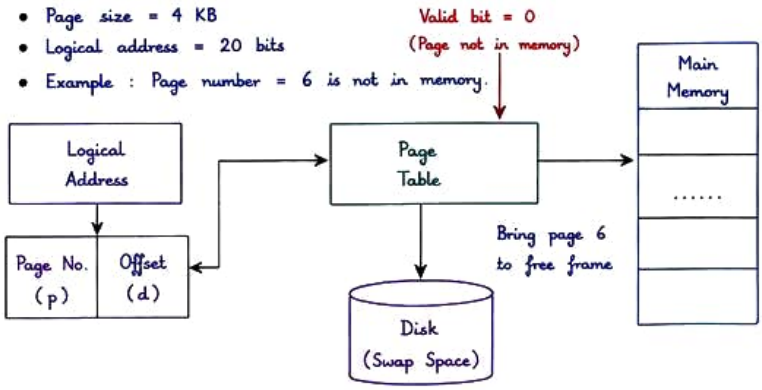
Valid = 1 → Page is in memory.
Valid = 0 → Page is not in memory (page fault if accessed).

Steps in Handling Page Fault :

1. CPU references a page.
2. MMU checks page table.
3. If Valid = 1 → Continue (no fault).
4. If Valid = 0 → Page Fault Trap.
5. OS checks if reference is valid.
   - If invalid → Abort the process (Segmentation Fault).
   - If valid → Continue.
6. Find a free frame in memory.
   If not available → Use page replacement algorithm to select a victim frame.
7. Read the required page from disk to the selected frame.
8. Update page table and restart the instruction.

Illustration :

- Page size = 4 KB
- Logical address = 20 bits
- Example : Page number = 6 is not in memory.



Types of Page Faults :

- Minor (Soft) Page Fault : Page is in disk but in swap space. No need to decrement process size.
- Major (Hard) Page Fault : Page must be read from backing store (disk). Time consuming.
- Invalid Page Fault : Reference to an invalid page (not part of process).

Effective Access Time (EAT) :

$$EAT = (1 - p) \times ma + p \times page\_fault\_time$$

where,
p = Page Fault Rate
ma = Memory Access Time
page_fault_time = Time to handle a page fault

Factors Affecting Performance :

- Page fault rate (p)
- Disk speed
- Page size
- Replacement algorithm
- Amount of free memory
- Locality of reference

Advantages :

- ✓ Allows execution of large programs.
- ✓ Better memory utilization.
- ✓ More users can be supported.
- ✓ Protection and isolation of address space.

Disadvantages :

- ✗ Page fault handling is time consuming.
- ✗ Frequent page faults degrade performance.
- ✗ Requires extra hardware and OS support.
- ✗ Thrashing may occur if page faults are high.

Summary :

A Page Fault is a normal part of virtual memory system. It occurs when a referenced page is not in main memory. The OS brings the page from disk, updates the page table and restarts the instruction. High page fault rate reduces system performance.

IT402 - COMPUTER ARCHITECTURE

14. ROM (READ ONLY MEMORY)

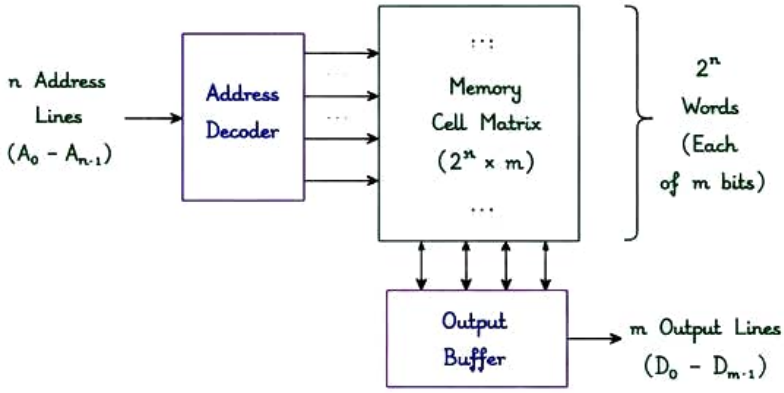
Definition :

Read Only Memory (ROM) is a type of non-volatile memory that stores data permanently. Information stored in ROM can only be read, not written, during normal operation. The data is written into ROM at the time of manufacturing or by a special programming process.

Characteristics :

- Non-volatile memory (data retained even when power is OFF).
- Data can be read but not easily modified.
- Stores fixed instructions such as bootstrap program (firmware).
- More reliable and slower than RAM.
- Used for permanent data and control programs.
- Lower cost for large volumes.

Basic Organization :



Operation :

1. Address is placed on address lines.
2. Address decoder selects the corresponding word in the matrix.
3. The stored data (m bits) is read through output buffer.
4. Data appears on output lines.

Types of ROM :

1. Mask ROM (MROM) :
   Data is programmed during manufacturing using a mask. Cannot be changed.
2. PROM (Programmable ROM) :
   User can program it once using a special programmer.
3. EPROM (Erasable PROM) :
   Can be erased using ultraviolet (UV) light and reprogrammed.
4. EEPROM (Electrically Erasable PROM) :
   Can be erased electrically and reprogrammed.
5. Flash Memory :
   A type of EEPROM that can be erased and written in blocks, widely used in pen drives, SSDs, etc.

Comparison: ROM vs RAM

Feature	ROM	RAM
Full Form	Read Only Memory	Random Access Memory
Volatility	Non-volatile	Volatile
Read / Write	Read Only (Normally)	Read & Write
Data Change	Not easily changeable	Easily changeable
Speed	Slower	Faster
Usage	Stores permanent data, firmware, bootstrap program	Stores temporary data and programs during execution
Cost	Lower for large capacity	Higher
Examples	BIOS, Firmware, Embedded code	Main memory of computer

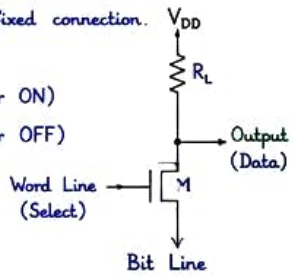
ROM Cell (Example - NMOS ROM Cell) :

A ROM cell stores data as a fixed connection.

Two states :

1 = Connection present (Transistor ON)

0 = Connection absent (Transistor OFF)



ROM Addressing Example (2^3 x 4 ROM) :

Address (A_2 A_1 A_0)	Word No.	Output (4 bits)			
		D_3	D_2	D_1	D_0
000	0	1	0	1	1
001	1	0	1	1	0
010	2	1	1	0	1
011	3	0	0	1	1
...	...	...	...	...	...
111	7	1	0	0	1

Here, 3 address lines select 1 out of 8 words. Each word has 4 bits of data.

Applications of ROM :

- Stores bootstrap program in computers.
- Used in BIOS of a computer.
- Used in embedded systems and microcontrollers.
- Stores firmware for devices.
- Used in lookup tables in digital systems.

Advantages :

- ✓ Non-volatile - data is retained.
- ✓ More reliable.
- ✓ Cheaper than RAM.
- ✓ Useful for permanent programs.

Disadvantages :

- × Cannot be modified easily.
- × Slower than RAM.
- × Not suitable for data that changes frequently.
- × Less flexible.

Summary :

ROM (Read Only Memory) is a non-volatile memory used to store permanent data and instructions. It allows only read operation during normal use and is widely used for firmware, BIOS and embedded systems.

UNIT - V

IT402 - COMPUTER ARCHITECTURE

RGPV NOTES HUB

4. PIPELINE HAZARDS

Definition :

A Pipeline Hazard is a condition that prevents the next instruction in the pipeline from executing during its next clock cycle. Hazards reduce the efficiency of pipelining and may require stalling or other techniques to handle them.

Basic Idea :

In ideal pipelining, one instruction completes in every clock cycle. But due to certain situations (hazards), the pipeline must wait (stall) or take corrective action, which reduces performance.

Hazard → Causes delay / stall / wrong result
Solution → Detect hazard and apply technique (stall, forwarding, flushing, etc.)

Where Hazards Occur ?

Hazards can occur between instructions that are close to each other in the instruction stream while they are in different stages of the pipeline.

Types of Pipeline Hazards :

1. Structural Hazards
   - Occurs when two or more instructions need the same hardware resource at the same time.
   - Example - One memory is used for both instruction fetch and data access.
   - Solution - Provide separate hardware resources or stall the pipeline.
2. Data Hazards
   - Occurs when an instruction depends on the result of a previous instruction still in the pipeline.
   - Types -
     (i) RAW (Read After Write)
     (ii) WAR (Write After Read)
     (iii) WAW (Write After Write)
   - Solution - Forwarding, Stalling, Reordering.
3. Control Hazards (Branch Hazards)
   - Occurs when the next instruction to be executed is not known due to branch/jump.
   - Example - Conditional branch, jump, interrupt.
   - Solution - Stalling, Branch prediction, Pipeline flushing.

Data Hazard Types with Example :

(i) RAW (Read After Write)
- Also called True Dependency.
- Instruction reads data that is written by a previous instruction.

Example :
```
I1 :  ADD  R1, R2, R3   ; R1 = R2 + R3
I2 :  SUB  R4, R1, R5   ; uses R1
```
Here I2 depends on result of I1.

(ii) WAR (Write After Read)
- Occurs when an instruction writes data that is needed (read) by a previous instruction.

Example :
```
I1 :  SUB  R1, R4, R5   ; reads R1
I2 :  ADD  R1, R2, R3   ; writes R1
```
Here I2 writes R1 before I1 reads it.

(iii) WAW (Write After Write)
- Occurs when two instructions write to the same location.

Example :
```
I1 :  ADD  R1, R2, R3   ; writes R1
I2 :  SUB  R1, R4, R5   ; writes R1
```
Here I2 writes R1 after I1.

Example - 5 Stage Pipeline (IF, ID, EX, MEM, WB) :

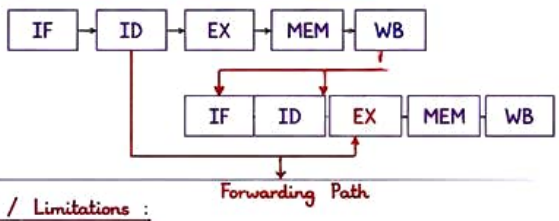
Clock Cycle →	1	2	3	4	5	6	7	8
I1 : ADD R1, R2, R3	IF	ID	EX	MEM	WB			
I2 : SUB R4, R1, R5 (RAW Hazard)		IF	ID	STALL	STALL	EX	MEM	WB
I3 : AND R6, R1, R7			IF	ID	EX	MEM	WB	

I2 needs R1 which is produced by I1 in WB stage (cycle 5). So pipeline is stalled for 2 cycles.

Techniques to Handle Hazards :

1. Stalling (Insertion of Bubbles) - Pause the pipeline by inserting empty cycles.
2. Forwarding (Bypassing) - Directly forward the result from one stage to another without waiting for it to be written back.
3. Branch Prediction - Predict the outcome of branch to avoid stalling.
4. Compiler Scheduling - Reorder instructions to minimize hazards.

Forwarding Example (RAW Hazard)



Advantages :

- ✓ Increases throughput (more instructions per unit time).
- ✓ Better hardware utilization.
- ✓ Higher performance without increasing clock frequency.
- ✓ Ideal for modern high performance processors.

Disadvantages / Limitations :

- × Hazards reduce efficiency.
- × Requires complex hardware and control logic.
- × Forwarding and prediction increase hardware cost and power.
- × Performance depends on type and frequency of hazards.

Summary : Pipeline Hazards are conditions that prevent smooth flow of instructions in a pipeline. There are three types - Structural, Data and Control Hazards. They are handled using techniques like Stalling, Forwarding, Branch Prediction and Compiler Scheduling.