



StudyHub

DIGITAL FUNDAMENTALS & COMPUTER ORGANIZATION

Complete Exam Preparation Guide

BCA • Semester V

Concepts • Definitions • Truth Tables • Diagrams • Solved Problems •
Model Answers

STUDYHUB COURSE GUIDE



Prepared for students on StudyHub

Structured from the University of Calicut BCA syllabus and aligned study material. This is an independent study resource, not an official University publication.



How to Use This Book

This guide is designed for examination preparation. Each topic is organized as definition → explanation → formula/equation → diagram or truth table where relevant → worked example → exam questions. The question priorities are study guidance, not guarantees of what will appear in an examination.

Syllabus coverage

Module	Focus	Hours	Marks
I	Number Systems & Boolean Algebra	10	15
II	Combinational & Sequential Logic Circuits	12	20
III	Basic Computer Organization & Microprogrammed Control	10	15
IV	Processor, Memory & I/O Organization	16	20
V	Computer Arithmetic & Types of Instruction	12	Open-ended / internal

The official syllabus explicitly covers Modules I-IV as the core theory modules and lists Module V as an open-ended component. The syllabus specifies 4-variable K-maps, SR/D/JK/T flip-flops, synchronous/asynchronous counters, ring/Johnson counters, shift registers, microprogrammed control, addressing modes, pipelining, cache mapping and DMA concepts.

Exam writing method

- Start every long answer with a precise definition.
- Use a labelled diagram whenever the question says “with neat diagram”.
- For gates and flip-flops, write the Boolean expression and truth table whenever applicable.
- For K-maps, explicitly state Gray-code order, grouping rule and simplified expression.
- For architecture questions, explain information flow and the purpose of each block.
- Do not claim that a particular question is guaranteed. Use priority labels as revision guidance.

MODULE I — NUMBER SYSTEMS AND BOOLEAN ALGEBRA

Module weight: 10 hours • 15 marks

Unit 1 — Binary Arithmetic

Binary number system: A positional number system with radix 2, using only 0 and 1. Each position represents a power of 2.

Binary addition rules: $0+0=0$; $0+1=1$; $1+0=1$; $1+1=10$ (sum 0, carry 1); $1+1+1=11$ (sum 1, carry 1).

1's complement: obtained by changing every 0 to 1 and every 1 to 0.

2's complement: obtained by taking the 1's complement and adding 1. For an n -bit signed 2's-complement number, the representable range is $-2^{(n-1)}$ to $2^{(n-1)}-1$.

Example — 8-bit 2's complement of 00010110

1's complement = 11101001
Add 1 = 11101010
Answer = 11101010

1's-complement addition

Add the operands normally. If a carry leaves the most significant bit, add that carry back to the least significant bit (end-around carry). The final result is interpreted using the chosen word width.

2's-complement addition

Add the fixed-width operands and discard the final carry beyond the most significant bit. Overflow occurs when two numbers with the same sign produce a result with the opposite sign.

Example — 8-bit subtraction: 25 – 13

25 = 00011001
13 = 00001101
2's complement of 13 = 11110011
 $00011001 + 11110011 = 1\ 00001100$
Discard carry $\rightarrow 00001100 = 12$

Overflow check

In 2's-complement addition, adding two positive numbers and obtaining a negative result, or adding two negative numbers and obtaining a positive result, indicates signed overflow.

Unit 2 — Logic Gates

Gate	Expression	Meaning
AND	$Y = A \cdot B$	1 only when both inputs are 1
OR	$Y = A + B$	1 when at least one input is 1
NOT	$Y = A'$	Inverts the input
NAND	$Y = (A \cdot B)'$	NOT of AND
NOR	$Y = (A + B)'$	NOT of OR
XOR	$Y = A \oplus B$	1 when inputs differ
XNOR	$Y = (A \oplus B)'$	1 when inputs are equal



A	B	AND	OR	NAND	NOR	XOR	XNOR
0	0	0	0	1	1	0	1
0	1	0	1	1	0	1	0
1	0	0	1	1	0	1	0
1	1	1	1	0	0	0	1

Universal gates

NAND and NOR are called universal gates because either gate family can be used to construct NOT, AND and OR, and therefore any Boolean logic network.

Function	NAND implementation	NOR implementation
NOT	$A \text{ NAND } A = A'$	$A \text{ NOR } A = A'$
AND	$(A \text{ NAND } B) \text{ NAND itself}$	$(A \text{ NOR } A) \text{ NOR } (B \text{ NOR } B)$
OR	$(A \text{ NAND } A) \text{ NAND } (B \text{ NAND } B)$	$(A \text{ NOR } B) \text{ NOR itself}$

MODULE I — BOOLEAN ALGEBRA & K-MAPS

Unit 3 — Boolean Algebra

Boolean algebra: A mathematical system used to represent and simplify logical relationships using binary variables and operations such as AND, OR and NOT.

Law	OR form	AND form
Identity	$A+0=A$	$A \cdot 1=A$
Null / domination	$A+1=1$	$A \cdot 0=0$
Idempotent	$A+A=A$	$A \cdot A=A$
Complement	$A+A'=1$	$A \cdot A'=0$
Involution	$(A')'=A$	—
Absorption	$A+A \cdot B=A$	$A \cdot (A+B)=A$
Commutative	$A+B=B+A$	$A \cdot B=B \cdot A$
Associative	$A+(B+C)=(A+B)+C$	$A(BC)=(AB)C$
Distributive	$A(B+C)=AB+AC$	$A+BC=(A+B)(A+C)$

Principle of duality

The dual of a Boolean expression is obtained by interchanging $+$ with $\cdot$ and 0 with 1 , while leaving variables and complements unchanged. Example: $A+0=A$ has dual $A \cdot 1=A$.

DeMorgan's theorems

$$(A+B)' = A' \cdot B'$$

$$(A \cdot B)' = A' + B'$$

For multiple variables: $(A+B+C)' = A'B'C'$ and $(ABC)' = A'+B'+C'$.

Unit 4 — Karnaugh Map (K-map)

Definition: A Karnaugh map is a graphical method for simplifying Boolean expressions by grouping adjacent cells containing 1s (for SOP) or 0s (for POS). The syllabus requires simplification up to four variables.

K-map rules

- Use Gray-code ordering: 00, 01, 11, 10.
- Groups must contain 1, 2, 4, 8 or 16 cells; group size is a power of two.
- Groups are rectangular and may wrap across opposite edges.
- Diagonal cells are not adjacent.
- Make groups as large as possible.
- Overlapping groups are allowed when they reduce the expression.
- For SOP, group 1s. For POS, group 0s.
- An essential prime implicant covers at least one required minterm not covered by another selected implicant.

3-variable example

$F(A,B,C)=\sum m(1,3,5,7)$. The minterms all have $C=1$. A group of four can be formed, so A and B vary and disappear. Therefore $F=C$.



4-variable example

$F(A,B,C,D) = \sum m(0,2,8,10)$. Using Gray-code rows AB and columns CD, these four cells form a valid group of four. B and D vary; A=0 and C=0 remain constant. Therefore $F = A'C'$.

SOP and POS

SOP (Sum of Products): OR of product terms, normally simplified by grouping 1s.

POS (Product of Sums): AND of sum terms, normally simplified by grouping 0s.

Exam warning

Do not use ordinary binary ordering in a K-map. Use Gray-code order. Do not form diagonal groups or groups of 3/6 cells.

MODULE I — EXAM QUESTIONS & MODEL ANSWERS

Short-answer questions (2/3 marks)

Q. What is 2's complement? Give an example.

Answer: 2's complement is obtained by complementing every bit and adding 1. Example: 00010110 → 11101001 → 11101010.

Q. State DeMorgan's theorems.

Answer: $(A+B)' = A'B'$ and $(AB)' = A'+B'$. They allow a complemented OR to be converted into AND of complements and a complemented AND into OR of complements.

Q. What is a K-map and why is it used?

Answer: A K-map is a graphical Boolean-simplification tool. It is used to minimize SOP or POS expressions by grouping adjacent cells.

Q. List the basic logic gates.

Answer: AND, OR, NOT, NAND, NOR, XOR and XNOR.

Q. What is the universal property of NAND?

Answer: NAND can realize NOT, AND and OR, so any Boolean function can be implemented using NAND gates alone.

Q. What is Gray-code ordering in a K-map?

Answer: Adjacent K-map cells differ in exactly one variable. The standard order is 00, 01, 11, 10.

5/6-mark questions

Q. Explain 1's and 2's complement arithmetic.

Model answer: Define both complements, show the bitwise procedure, explain signed representation, explain 1's-complement end-around carry and 2's-complement carry discard, and demonstrate one subtraction example.

Q. Explain all seven logic gates with expressions and truth tables.

Model answer: Define AND, OR, NOT, NAND, NOR, XOR and XNOR; provide their Boolean expressions, truth table and standard symbol in an exam diagram.

Q. Explain the universal property of NAND and NOR gates.

Model answer: Show NOT, AND and OR realization using NAND. Then show NOT, OR and AND realization using NOR. Conclude that either family is functionally complete.

Q. Explain Boolean laws and DeMorgan's theorem.

Model answer: Present identity, null, idempotent, complement, commutative, associative, distributive and absorption laws. State both DeMorgan theorems and demonstrate a simple transformation.

Q. Explain K-map simplification up to four variables.

Model answer: Explain cell arrangement, Gray code, adjacency, grouping, wrap-around, SOP/POS and prime implicants, followed by a worked 3- or 4-variable example.

Long-answer practice

1. Describe the various logic gates, their truth tables and Boolean expressions. Explain K-map simplification up to four variables with an example.

2. Explain 1's and 2's complement arithmetic, signed ranges, overflow and subtraction with worked examples.

3. Simplify a given 3-variable and 4-variable Boolean expression using K-map, showing every grouping step.

MODULE II — COMBINATIONAL & SEQUENTIAL LOGIC CIRCUITS

Module weight: 12 hours • 20 marks

Unit 5 — Adders

Combinational circuit: A digital circuit whose output depends only on the present input values; it has no stored state.

Half adder: Adds two one-bit inputs A and B and produces Sum and Carry.

$$S = A \oplus B$$

$$C = A \cdot B$$

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Ripple-carry structure



Full adder: Adds A, B and carry-in Cin. Outputs are Sum and Cout.

$$S = A \oplus B \oplus C_{in}$$

$$C_{out} = AB + AC_{in} + BC_{in}$$

A	B	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Full adder using two half adders

HA1 adds A and B → S1, C1. HA2 adds S1 and Cin → final S, C2. Final carry is C1+C2.

Ripple Carry Adder (RCA): Multiple full adders are cascaded so that each stage receives the carry generated by the previous stage. In an n-bit RCA, the carry propagates from the least significant stage toward the

most significant stage.

Key limitation: carry propagation creates delay; the worst-case delay grows with the number of stages.

Unit 6 — Encoder and Decoder

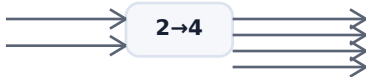
Encoder: Converts one active input among 2^n inputs into an n -bit binary code. A basic 4-to-2 encoder has four inputs and two outputs.

Active input	Y1	Y0
D0	0	0
D1	0	1
D2	1	0
D3	1	1

Decoder: Converts an n -bit binary input into one of 2^n active outputs. A 2-to-4 decoder has two inputs and four outputs.

A	B	Y0	Y1	Y2	Y3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

2-to-4 decoder block



Unit 7 — Multiplexer and Demultiplexer

Multiplexer (MUX): A data selector that chooses one of many inputs and routes it to a single output according to select lines.

For a 4-to-1 MUX: $Y = I_0 S_1' S_0' + I_1 S_1' S_0 + I_2 S_1 S_0' + I_3 S_1 S_0$.

Demultiplexer (DEMUX): A data distributor that routes one input to one of several outputs according to select lines.

4-to-1 multiplexer





MODULE II — LATCHES, FLIP-FLOPS & COUNTERS

Unit 8 — Latches and Flip-Flops

Latch: A level-sensitive storage element. Its output may respond to inputs while the enable/control level is active.

Flip-flop: A clock-controlled bistable storage element whose state changes according to the active clock edge and inputs.

Feature	Latch	Flip-flop
Sensitivity	Level-sensitive	Edge-triggered
Control	Enable level	Clock edge
Typical use	Simple storage/control	Synchronous sequential systems

SR flip-flop

S	R	Q(next)	Operation
0	0	Q	No change
0	1	0	Reset
1	0	1	Set
1	1	Invalid / forbidden in basic SR	Invalid

D flip-flop

D flip-flop eliminates the invalid SR combination by using one data input. At the active clock edge, $Q(\text{next})=D$.

D	Q(next)
0	0
1	1

JK flip-flop

J	K	Q(next)	Operation
0	0	Q	No change
0	1	0	Reset
1	0	1	Set
1	1	Q'	Toggle

T flip-flop

T	Q(next)	Operation
0	Q	Hold
1	Q'	Toggle

Unit 9 — Counters



Counter: A sequential circuit that passes through a prescribed sequence of states in response to clock pulses.

Asynchronous (ripple) counter: Only the first flip-flop receives the external clock; subsequent stages are clocked by preceding stages. Propagation delay accumulates through the chain.

Synchronous counter: All flip-flops receive the common clock, and combinational logic determines which stages toggle. This reduces ripple delay.

An n-stage binary counter normally has 2^n states.

Unit 10 — Ring, Johnson and Shift Registers

Ring counter: A shift-register counter in which the last stage feeds the first, normally producing n valid one-hot states for n stages when initialized correctly.

Johnson counter: A twisted-ring counter in which the complemented output of the last stage feeds the first. An n-stage Johnson counter normally has $2n$ distinct states.

Shift register: A group of flip-flops used to store and shift binary data. Types: SISO, SIPO, PISO and PIPO.

Type	Meaning	Main operation
SISO	Serial In Serial Out	Serial loading and serial output
SIPO	Serial In Parallel Out	Serial loading, parallel readout
PISO	Parallel In Serial Out	Parallel load, serial output
PIPO	Parallel In Parallel Out	Parallel load and parallel readout



MODULE II — EXAM QUESTIONS & MODEL ANSWERS

High-priority short answers

Q. What is a latch?

Answer: A latch is a level-sensitive bistable storage element used to store one bit.

Q. What is the difference between combinational and sequential circuits?

Answer: Combinational outputs depend only on present inputs; sequential outputs depend on present inputs and stored state.

Q. What is a multiplexer?

Answer: A MUX selects one of several input signals and routes it to one output using select lines.

Q. What is a decoder?

Answer: A decoder converts an n -bit input code into one of up to 2^n active outputs.

Q. What is a full adder?

Answer: A full adder adds A , B and C_{in} and produces Sum and C_{out} .

Q. What is a ripple-carry adder?

Answer: It is a cascade of full adders in which each carry output becomes the next stage's carry input.

Q. What is a synchronous counter?

Answer: A counter in which all flip-flops receive the same clock signal.

Q. What is a ring counter?

Answer: A shift-register counter where the last stage feeds the first, normally giving n states for n stages.

Q. What is a Johnson counter?

Answer: A twisted-ring counter using the complement of the final stage as feedback; n stages normally produce $2n$ states.

6-mark / essay questions

Q. Draw and explain a 2-to-4 decoder with truth table.

Model-answer structure: definition → labelled diagram → operation → truth table/equation → key advantage/limitation → application. Write the exact sub-points demanded by the question.

Q. Explain a full adder with truth table, equations and implementation using two half adders.

Model-answer structure: definition → labelled diagram → operation → truth table/equation → key advantage/limitation → application. Write the exact sub-points demanded by the question.

Q. Explain the universal property of NAND with suitable gate realizations.

Model-answer structure: definition → labelled diagram → operation → truth table/equation → key advantage/limitation → application. Write the exact sub-points demanded by the question.

Q. Explain a 4-to-1 multiplexer with operation, truth table and Boolean expression.

Model-answer structure: definition → labelled diagram → operation → truth table/equation → key advantage/limitation → application. Write the exact sub-points demanded by the question.

Q. Compare synchronous and asynchronous counters with diagrams.

Model-answer structure: definition → labelled diagram → operation → truth table/equation → key advantage/limitation → application. Write the exact sub-points demanded by the question.

Q. Explain SR, D, JK and T flip-flops with truth tables and applications.

Model-answer structure: definition → labelled diagram → operation → truth table/equation → key advantage/limitation → application. Write the exact sub-points demanded by the question.

**Q. Explain ring counter, Johnson counter and shift registers.**

Model-answer structure: definition → labelled diagram → operation → truth table/equation → key advantage/limitation → application. Write the exact sub-points demanded by the question.

Q. Explain the design and working of a ripple-carry adder.

Model-answer structure: definition → labelled diagram → operation → truth table/equation → key advantage/limitation → application. Write the exact sub-points demanded by the question.

MODULE III — BASIC COMPUTER ORGANIZATION & MICROPROGRAMMED CONTROL

Module weight: 10 hours • 15 marks

Unit 11 — Instruction Codes, Registers and Common Bus

Instruction: A binary-coded command that specifies an operation to be performed by the processor and the operands or their locations.

Instruction code: The complete binary representation of an instruction, including opcode and operand/address information as defined by the architecture.

Opcode: The field that specifies the operation, such as ADD, LOAD or JUMP.

Common CPU registers

Register	Purpose
PC (Program Counter)	Holds the address of the next instruction to be fetched.
IR (Instruction Register)	Holds the currently fetched instruction.
MAR / Address register	Holds a memory address for an access.
MDR / Data register	Holds data being transferred to/from memory.
Accumulator	Stores intermediate arithmetic/logic results in accumulator-based designs.
Temporary registers	Hold intermediate values during execution.

Common bus system

A common bus is a shared set of lines used to transfer binary information among registers and functional units. Source selection places one register's contents on the bus; a destination control loads another register.

Common bus concept



Unit 12 — Computer Instructions

Instruction classes include data transfer, arithmetic, logical/bit manipulation, shift, program control, conditional branch, subroutine call/return and I/O instructions. The exact instruction names depend on the processor architecture.

Unit 13 — Timing and Control

Control unit: Generates the timing and control signals that coordinate registers, ALU, memory and I/O operations.



Hardwired control: Control signals are generated by fixed combinational/sequential logic. It is generally fast but harder to modify.

Microprogrammed control: Control signals are represented by microinstructions stored in control memory. It is easier to modify and organize but introduces control-memory access overhead.

Aspect	Hardwired	Microprogrammed
Implementation	Logic circuits	Control memory + sequencer
Speed	Usually faster	Usually slower due to microinstruction sequencing
Modification	Difficult	Easier
Complex instruction sets	More difficult to design	Convenient



MODULE III — INSTRUCTION CYCLE & MICROPROGRAMMING

Unit 14 — Instruction Cycle

The instruction cycle is the sequence of operations required to fetch, decode and execute an instruction, with an interrupt check where the architecture supports it.

1. Fetch: obtain the instruction from memory.
2. Decode: determine the operation and required operands.
3. Effective-address / operand preparation: determine operand location where required.
4. Execute: perform the operation.
5. Interrupt check: determine whether an interrupt service sequence is required.
6. Next instruction: continue with the updated program counter.

Typical fetch micro-operations (architecture-dependent)

```
MAR ← PC
MDR ← Memory[MAR]
IR ← MDR
PC ← PC + 1
```

These are standard register-transfer examples; exact timing and register names depend on the processor organization being taught.

Unit 15 — Microprogrammed Control

Control memory: Memory containing microinstructions that generate the control signals required for CPU micro-operations.

Control Address Register (CAR): Holds the address of the next microinstruction.

Microinstruction register: Holds the current microinstruction/control word used to generate control signals.

Microprogrammed control path

CAR → Control Memory → IR / Signals

Address sequencing

- Sequential next address: select the next microinstruction in sequence.
- Unconditional branch: load a specified control-memory address.
- Conditional branch: choose the next address based on a condition flag.
- Opcode mapping: use instruction opcode information to enter the correct microroutine.
- Common/subroutine sequencing: reuse microinstruction sequences where appropriate.

Unit 16 — Microinstruction Format and Symbolic Microinstruction



A microinstruction typically contains a control field, a sequencing/condition field and an address field. The exact field layout is architecture-dependent.

Field	Purpose
Control field	Specifies micro-operations/control signals to activate.
Condition / sequencing field	Specifies how the next microinstruction is selected.
Address field	Provides a branch or next-address value when required.

Symbolic microinstruction: A readable notation for expressing register transfers or micro-operations, for example $R2 \leftarrow R1$ or $AC \leftarrow AC + DR$.



MODULE III — EXAM QUESTIONS & MODEL ANSWERS

Short-answer questions

Q. What is an opcode?

Answer: The opcode is the instruction field that specifies the operation to be performed.

Q. What is a common bus system?

Answer: A shared set of lines that provides a controlled path for transferring data among registers and functional units.

Q. What is an instruction cycle?

Answer: The sequence of fetch, decode, execute and related control operations required to process one instruction.

Q. What is hardwired control?

Answer: A control-unit implementation in which control signals are produced by fixed logic circuits.

Q. What is microprogrammed control?

Answer: A control-unit implementation in which control signals are represented by microinstructions stored in control memory.

Q. What is control memory?

Answer: Memory that stores microinstructions used by a microprogrammed control unit.

Q. What is address sequencing?

Answer: The process of determining the address of the next microinstruction in control memory.

Long-answer practice

Q. Explain instruction codes, CPU registers and a common bus system.

Answer framework: definition → labelled architecture/flow → operation in ordered steps → comparison or example where applicable → conclusion.

Q. Explain the complete instruction cycle with a neat flow diagram and typical fetch micro-operations.

Answer framework: definition → labelled architecture/flow → operation in ordered steps → comparison or example where applicable → conclusion.

Q. Compare hardwired and microprogrammed control.

Answer framework: definition → labelled architecture/flow → operation in ordered steps → comparison or example where applicable → conclusion.

Q. Explain the organization of a microprogrammed control unit, control memory and address sequencing.

Answer framework: definition → labelled architecture/flow → operation in ordered steps → comparison or example where applicable → conclusion.

Q. Explain microinstruction format and symbolic microinstructions.

Answer framework: definition → labelled architecture/flow → operation in ordered steps → comparison or example where applicable → conclusion.

MODULE IV — PROCESSOR, MEMORY & I/O ORGANIZATION

Module weight: 16 hours • 20 marks

Unit 17 — General Register and Stack Organization

Processor organization: The arrangement and interconnection of registers, ALU, control unit, buses and associated datapath elements used to execute instructions.

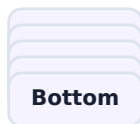
Basic processor organization



General register organization: Multiple registers are connected to a common datapath/bus and selected through control signals. The ALU performs arithmetic/logic operations on selected register values.

Stack organization: A stack is a LIFO (Last-In, First-Out) storage structure. PUSH inserts an item; POP removes the top item. A Stack Pointer (SP) identifies the current top or next stack position according to the architecture.

Stack / LIFO concept



Unit 17 — Instruction Formats

Instruction format: The bit-field layout of an instruction, commonly containing opcode plus operand/address, register and/or addressing-mode fields.

Format	Idea
3-address	Operation with three explicit operand/result locations.
2-address	Two explicit addresses; one may also hold the result.
1-address	Often accumulator-based; one explicit operand address.
0-address	Stack-based; operands are implied at the top of the stack.

Addressing modes

Mode	Effective address / operand idea	Example
Immediate	Operand is in the instruction	MOV R1,#25
Direct	Address field gives memory address	LOAD R1,1000



Mode	Effective address / operand idea	Example
Indirect	Address field points to a location containing the effective address	LOAD R1,(1000)
Register	Operand is in a register	ADD R1,R2
Register indirect	Register contains effective address	LOAD R1,(R2)
Indexed	Effective address = base/address + index	LOAD R1,A[R2]
Relative	Effective address = PC + displacement	BR +20
Base/displacement	Effective address = base register + displacement	LOAD R1,8(R2)

MODULE IV — RISC, CISC & PIPELINING

Unit 18 — RISC vs CISC

RISC (Reduced Instruction Set Computer): A design philosophy emphasizing a relatively small, regular instruction set, often with load/store organization and efficient pipelining.

CISC (Complex Instruction Set Computer): A design philosophy supporting a richer instruction set, including more complex instructions and addressing modes.

Feature	RISC	CISC
Instruction set	Smaller / regular	Larger / more complex
Instruction length	Often fixed or limited formats	Often variable
Memory operations	Typically load/store	May allow memory operands in many instructions
Control	Often simpler	Often more complex
Pipelining	Designed to be pipeline-friendly	Possible, but decoding/control can be more complex

Modern processors can combine ideas from both design traditions; the table describes the conventional architectural distinction.

Parallel processing

Parallel processing: Performing multiple computations or instruction/data operations concurrently using multiple functional units, cores, processors or other parallel resources.

Benefits include increased throughput, better utilization of hardware and faster execution of parallel workloads. Applications include scientific computing, graphics, data processing and multicore systems.

Unit 19 — Pipelining

Pipelining: Dividing instruction processing into stages so that multiple instructions can occupy different stages simultaneously. The main benefit is increased throughput; it does not automatically reduce the latency of a single instruction.

Typical instruction pipeline

IF ID EX MEM WB

Instruction moves through stages; multiple instructions overlap in time.

Typical stages: IF (Instruction Fetch), ID (Instruction Decode), EX (Execute), MEM (Memory access), WB (Write Back). Actual stage names vary by architecture.

Pipeline hazards

- **Structural hazard**: two stages require the same hardware resource simultaneously.
- **Data hazard**: an instruction depends on a result that has not yet become available.
- **Control hazard**: a branch or jump changes the instruction stream.

Arithmetic pipeline

An arithmetic pipeline divides a complex arithmetic operation into stages, allowing different data items to be processed concurrently through different stages.



Ideal speedup

For a balanced pipeline with k stages and sufficiently many independent tasks, ideal throughput improvement approaches k relative to a non-overlapped implementation. Real speedup is lower because of stage imbalance, pipeline-register overhead, stalls and hazards.

MODULE IV — MEMORY, CACHE & I/O

Unit 20 — Memory Hierarchy and Main Memory

Memory hierarchy: An arrangement of storage levels based on speed, capacity and cost per bit. Faster storage is usually smaller and more expensive per bit.

Typical memory hierarchy

Registers Cache Main Memory Secondary

Level	Typical property
Registers	Fastest, very small, directly used by CPU
Cache	Small, fast memory holding recently/frequently used blocks
Main memory	Larger working storage for running programs
Secondary storage	Large, persistent storage with higher access latency

Locality: Temporal locality means recently accessed data/instructions are likely to be accessed again.

Spatial locality means nearby addresses are likely to be accessed soon.

Main memory: The primary memory directly accessible by the processor for active program/data storage. RAM is the common read/write main-memory technology in general-purpose systems; ROM and related non-volatile technologies serve different roles.

Unit 21 — Associative and Cache Memory

Associative memory / CAM: Memory searched by content rather than only by a conventional address. A search key is compared against stored entries in parallel.

Cache memory: A small, fast memory placed between the processor and main memory. It stores copies of recently or frequently used memory blocks.

Cache hit: Requested data is found in the cache. Cache miss: data is not found and must be obtained from a lower memory level.

Hit ratio: hits / total accesses. Miss penalty: additional time required to service a miss.

Cache mapping

Direct mapping: each main-memory block maps to exactly one cache line. A common formula is:

$$\text{Cache line} = \text{Main-memory block number} \bmod \text{Number of cache lines}$$

Fully associative mapping: a block can be placed in any cache line; tag comparison determines a hit.

Set-associative mapping: cache lines are divided into sets; a block maps to one set but can occupy any line within that set.

Mapping is different from replacement policy. Mapping determines where a block may go; replacement policy determines which existing entry is evicted when a location is full.

Unit 22 — I/O Organization



Programmed I/O: CPU explicitly controls the I/O operation and repeatedly checks device status. CPU time is consumed while waiting/polling.

Interrupt-initiated I/O: The device signals the CPU when service is required, allowing the CPU to perform other work between events.

DMA (Direct Memory Access): A DMA controller transfers data between an I/O device and main memory with minimal CPU involvement after setup. DMA is particularly useful for block transfers.

DMA data path





MODULE IV — EXAM QUESTIONS & MODEL ANSWERS

Short-answer questions

Q. What is an addressing mode?

Answer: An addressing mode specifies how an instruction identifies or computes the location of its operand.

Q. What is RISC?

Answer: Reduced Instruction Set Computer; a design approach using a relatively small, regular instruction set and efficient datapaths/pipelining.

Q. What is CISC?

Answer: Complex Instruction Set Computer; a design approach supporting a richer set of more complex instructions and addressing modes.

Q. What is pipelining?

Answer: A technique that overlaps instruction-processing stages to increase instruction throughput.

Q. What is a cache hit?

Answer: A cache hit occurs when requested data is found in the cache.

Q. What is associative memory?

Answer: Memory that searches stored contents by matching a key rather than relying only on an address.

Q. What is DMA?

Answer: Direct Memory Access; a mechanism allowing data transfer between I/O and memory with minimal CPU involvement after setup.

Long-answer practice

Q. Explain general register organization and stack organization with diagrams.

Model-answer structure: definition → architecture/diagram → working → comparison/formula → advantages/limitations → applications.

Q. Explain instruction formats and addressing modes with suitable examples.

Model-answer structure: definition → architecture/diagram → working → comparison/formula → advantages/limitations → applications.

Q. Compare RISC and CISC.

Model-answer structure: definition → architecture/diagram → working → comparison/formula → advantages/limitations → applications.

Q. Explain instruction pipelining, arithmetic pipelining and pipeline hazards.

Model-answer structure: definition → architecture/diagram → working → comparison/formula → advantages/limitations → applications.

Q. Explain memory hierarchy, locality and main memory.

Model-answer structure: definition → architecture/diagram → working → comparison/formula → advantages/limitations → applications.

Q. Explain associative memory and cache memory.

Model-answer structure: definition → architecture/diagram → working → comparison/formula → advantages/limitations → applications.

Q. Explain direct, fully associative and set-associative cache mapping.

Model-answer structure: definition → architecture/diagram → working → comparison/formula → advantages/limitations → applications.

Q. Compare programmed I/O, interrupt-initiated I/O and DMA.



Model-answer structure: definition → architecture/diagram → working → comparison/formula → advantages/limitations → applications.

MODULE V — OPEN-ENDED: COMPUTER ARITHMETIC & TYPES OF INSTRUCTION

The syllabus lists this as an open-ended module of 12 hours. It is kept separate from the core Modules I-IV theory preparation.

Unit 1 — Computer Arithmetic

Addition and subtraction: practise fixed-width binary arithmetic using unsigned or complement-based signed representation as specified by the problem.

Multiplication algorithms: understand repeated add-and-shift processing and be able to trace partial products/register contents when a particular algorithm is taught.

Division algorithms: understand repeated subtract/shift processing and track quotient and remainder carefully.

When solving an arithmetic algorithm, always state the word length, sign convention and initial register values.

Unit 2 — Types of Instructions

Class	What it does
Memory-reference	Uses an address/effective address to access a memory operand.
Register-reference	Operates directly on CPU register contents.
Input / Output	Transfers or controls data/status associated with I/O devices.
Data transfer	Moves information among registers, memory and/or I/O.
Arithmetic	Performs addition, subtraction, increment, decrement and related operations.
Logical / bit manipulation	Performs AND, OR, XOR, complement and bit set/clear/test operations.
Shift	Performs logical, arithmetic or rotate-style shifts.
Program control	Changes normal sequential instruction execution.
Conditional branch	Changes control flow only when a specified condition is satisfied.
Subroutine call / return	Transfers control to a reusable routine and later resumes at the saved return point.

Internal practice questions

1. Trace one binary multiplication example using a clear add/shift table.
2. Trace one binary division example and identify quotient and remainder.
3. Classify a mixed list of instructions into the instruction classes above.
4. Compare conditional branch, subroutine call and return.
5. Explain why word length and signed interpretation must be stated before tracing an arithmetic algorithm.



STUDYHUB — QUICK REFERENCE SHEETS

Core Boolean and digital formulas

Topic	Formula / rule
Half adder	$S = A \oplus B$; $C = AB$
Full adder	$S = A \oplus B \oplus C_{in}$; $C_{out} = AB + AC_{in} + BC_{in}$
DeMorgan	$(A+B)' = A'B'$; $(AB)' = A' + B'$
XOR	$A \oplus B = A'B + AB'$
XNOR	$A \odot B = AB + A'B'$
2's complement	1's complement + 1
2's-complement range (n bits)	$-2^{(n-1)}$ to $2^{(n-1)} - 1$
Direct cache mapping	Line = Block number mod Number of cache lines
Binary counter states	Normally 2^n for n stages
Ring counter states	Normally n for n stages
Johnson counter states	Normally 2n for n stages

Logic-gate truth table

A	B	AND	OR	NAND	NOR	XOR	XNOR
0	0	0	0	1	1	0	1
0	1	0	1	1	0	1	0
1	0	0	1	1	0	1	0
1	1	1	1	0	0	0	1

Flip-flop quick table

Type	Input condition	Next state
SR	00	Hold
SR	01	Reset
SR	10	Set
SR	11	Invalid in basic SR
D	D=0/1	$Q(\text{next}) = D$
JK	00	Hold
JK	01	Reset
JK	10	Set
JK	11	Toggle
T	0	Hold
T	1	Toggle



STUDYHUB — MASTER EXAM QUESTION BANK

These are high-value practice questions assembled from the official syllabus scope and aligned study material. They are not guarantees of the exact examination paper.

Module I

1. Define 1's complement and 2's complement. Give examples.
2. Perform binary addition and subtraction using 1's and 2's complement.
3. Explain signed number range and overflow in 2's complement.
4. Explain AND, OR, NOT, NAND, NOR, XOR and XNOR with truth tables.
5. Explain the universal property of NAND and NOR.
6. State and explain Boolean algebra laws.
7. State DeMorgan's theorems and apply them.
8. Explain principle of duality.
9. Define K-map and explain its purpose.
10. Explain 3-variable K-map simplification with example.
11. Explain 4-variable K-map simplification with wrap-around.
12. Differentiate SOP and POS.

Module II

1. Define combinational and sequential circuits.
2. Explain half adder with truth table and equations.
3. Explain full adder with truth table and equations.
4. Explain full adder using two half adders.
5. Explain ripple-carry adder and propagation delay.
6. Draw and explain a 2-to-4 decoder.
7. Explain a 4-to-2 encoder.
8. Explain 4-to-1 MUX with truth table and equation.
9. Explain 1-to-4 DEMUX.
10. Differentiate latch and flip-flop.
11. Explain SR, D, JK and T flip-flops.
12. Compare synchronous and asynchronous counters.
13. Explain ring and Johnson counters.
14. Explain SISO, SIPO, PISO and PIPO shift registers.

Module III

1. Define instruction code, opcode and instruction format.
2. Explain CPU registers and their functions.
3. Explain common bus organization.
4. Classify computer instructions.
5. Explain timing and control.
6. Compare hardwired and microprogrammed control.
7. Explain the instruction cycle.



- 8. Explain fetch micro-operations.**
- 9. Explain control memory.**
- 10. Explain address sequencing.**
- 11. Explain microinstruction format.**
- 12. Explain symbolic microinstructions.**

Module IV

- 1. Explain general register organization.**
- 2. Explain stack organization and PUSH/POP.**
- 3. Explain instruction formats.**
- 4. Explain immediate, direct, indirect, register, register indirect, indexed and relative addressing.**
- 5. Compare RISC and CISC.**
- 6. Explain parallel processing.**
- 7. Explain instruction pipelining.**
- 8. Explain arithmetic pipelining.**
- 9. Explain structural, data and control hazards.**
- 10. Explain memory hierarchy and locality.**
- 11. Explain main memory.**
- 12. Explain associative memory.**
- 13. Explain cache hit, miss, hit ratio and miss penalty.**
- 14. Explain direct, fully associative and set-associative cache mapping.**
- 15. Compare programmed I/O, interrupt-initiated I/O and DMA.**



STUDYHUB — LAST-MINUTE REVISION PLAN

Priority 1 — Must know cold

- 1's and 2's complement arithmetic and subtraction
- All seven logic gates: symbol, expression and truth table
- NAND/NOR universal implementations
- Boolean laws and DeMorgan's theorem
- 3-variable and 4-variable K-map rules and simplification
- Half adder, full adder and ripple-carry adder
- MUX, DEMUX, encoder and decoder
- SR, D, JK and T flip-flops
- Synchronous vs asynchronous counters
- Ring/Johnson counters and shift registers

Priority 2 — Architecture core

- Registers and common bus
- Instruction cycle
- Hardwired vs microprogrammed control
- Control memory and address sequencing
- Instruction formats and addressing modes
- RISC vs CISC
- Pipelining and hazards
- Memory hierarchy and locality
- Cache mapping
- Programmed I/O, interrupt I/O and DMA

Final diagram checklist



- Logic-gate symbols
- Half adder and full adder
- Ripple-carry adder
- 2-to-4 decoder and 4-to-1 MUX
- Flip-flop symbols and truth tables
- Ripple and synchronous counter concept
- Ring/Johnson counter
- Shift-register types
- CPU/register organization
- Instruction cycle
- Microprogrammed control
- Pipeline
- Memory hierarchy
- Cache mapping
- DMA

Final exam rule

If a question asks for a diagram, draw it. If it asks for a truth table, write it. If it asks for a comparison, use a table. If it asks for a numerical simplification, show the intermediate steps. These are high-return presentation practices.

Source and educational note

The module structure in this guide follows the University of Calicut BCA syllabus document dated 13 June 2025. Supporting study material was used to expand definitions, worked examples, revision points and exam-oriented question forms. This PDF is an independent educational resource created for StudyHub; it is not an official University publication and priority labels do not guarantee examination questions.

StudyHub • Study Together • Grow Together

STUDYHUB

COMPLETE COVERAGE EXPANSION

Digital Fundamentals & Computer Organization — BCA Semester V

Purpose: This expansion is appended to the existing StudyHub guide after a syllabus-level coverage audit. It adds detailed worked problems, explicit circuit equations, missing Module V examples, fuller instruction classifications, exam-answer templates, and additional diagrams/quick-reference material.

Important: The official syllabus is the controlling source for scope. Modules I–IV are the core modules; Module V is explicitly an open-ended module. This document does not claim that any particular question is guaranteed.

Module	Official scope	Hours	Marks
I	Number Systems and Boolean Algebra	10	15
II	Combinational and Sequential Logic Circuits	12	20
III	Basic Computer Organization and Microprogrammed Control	10	15
IV	Processor, Memory and I/O Organization	16	20
V	Computer Arithmetic & Types of Instruction — Open-Ended	12	Open-ended / internal

1. VERIFIED COVERAGE AUDIT

The University of Calicut syllabus lists the following units. Every listed unit is explicitly represented in the combined StudyHub guide + this expansion.

Unit	Required syllabus content	Status in combined guide
1	Binary addition/subtraction; 1's/2's complement; complement addition	Covered + expanded worked examples
2	AND, OR, NOT, NOR, NAND, XOR, XNOR; NAND/NOR universal property	Covered + explicit realizations
3	Boolean operations, laws/rules, DeMorgan's theorem	Covered + expanded identities
4	K-map simplification up to 4 variables	Covered + SOP/POS + wrap-around + worked steps
5	Half Adder, Full Adder, Ripple Carry Adder	Covered + truth tables/equations
6	Encoder and Decoder (Basic Circuit Only)	Covered + basic tables/equations
7	MUX and DEMUX (Basic Circuit Only)	Covered + tables/equations
8	Latches; SR, D, JK, T flip-flops; truth tables and circuit	Covered + characteristic equations
9	Synchronous and Asynchronous Counters	Covered + operation/state sequences
10	Johnson, Ring counter, Shift Registers	Covered + sequences and types
11	Instruction codes, Registers, Common Bus	Covered + expanded register-transfer examples
12	Computer Instructions	Covered + full classification/examples
13	Hardwired and Microprogrammed Control	Covered + comparison
14	Instruction Cycle	Covered + micro-operations
15	Control Memory & Address Sequencing	Covered + sequencing cases
16	Microinstruction Format & Symbolic Microinstruction	Covered + field formats
17	General Register/Stack Organization; Instruction Formats; Addressing Modes	Covered + expanded examples
18	RISC vs CISC; Parallel Processing	Covered + comparison
19	Arithmetic/Instruction Pipelining	Covered + hazards/speedup
20	Memory Hierarchy; Main Memory	Covered + locality
21	Associative Memory; Cache Mapping	Covered + mapping worked example
22	Programmed I/O; Interrupt I/O; DMA (concepts only)	Covered + comparison
V-1	Computer arithmetic: add/subtract, multiplication, division algorithms	Expanded with algorithm examples
V-2	Specified instruction classes and examples	Expanded with examples

The official syllabus explicitly lists Modules I–IV and their unit contents, and identifies Module V as an open-ended component. The audit above follows that syllabus structure. ■filecite■turn23file0■L30-L58■

2. MODULE I — EXPANDED PROBLEM SET

Binary addition — essential rules

$0 + 0 = 0$
$0 + 1 = 1$
$1 + 0 = 1$
$1 + 1 = 10$ (sum = 0, carry = 1)
$1 + 1 + 1 = 11$ (sum = 1, carry = 1)

Q. Perform $1011 + 0110$.

Answer:

- Write the operands with equal width.
- From the LSB: $1+0=1$.
- Next: $1+1=10 \rightarrow$ write 0, carry 1.
- Next: $0+1+1=10 \rightarrow$ write 0, carry 1.
- Next: $1+0+1=10 \rightarrow$ write 0, carry 1.
- Final carry = 1. Result = 10001.

1's-complement subtraction

Q. Compute $9 - 5$ using 4-bit 1's-complement subtraction.

Answer:

- $9 = 1001$, $5 = 0101$.
- 1's complement of $0101 = 1010$.
- Add: $1001 + 1010 = 1\ 0011$.
- End-around carry: $0011 + 1 = 0100$.
- $0100 = 4$, so $9 - 5 = 4$.

Q. Compute $5 - 9$ using 4-bit 1's-complement subtraction.

Answer:

- $5 = 0101$, $9 = 1001$.
- 1's complement of $1001 = 0110$.
- Add: $0101 + 0110 = 1011$; there is no end-around carry.
- Complement $1011 \rightarrow 0100$.
- Therefore the result is $-0100 = -4$.

2's-complement subtraction

Q. Compute $25 - 13$ using 8-bit 2's complement.

Answer:

- $25 = 00011001$.
- $13 = 00001101$.
- 1's complement of $13 = 11110010$.
- 2's complement = 11110011 .
- $00011001 + 11110011 = 1\ 00001100$.
- Discard the carry beyond 8 bits.

- Result = 00001100 = 12.

Signed range and overflow

For n-bit two's-complement representation, range = $-2^{(n-1)}$ to $2^{(n-1)}-1$.

Signed overflow occurs when adding two operands with the same sign produces a result with the opposite sign.

Example: in 4-bit two's complement, $7 + 3 = 10$, which cannot be represented as a positive 4-bit signed number. The bit pattern wraps to 1010, interpreted as -6 , indicating overflow.

Universal-gate equations — write these explicitly in exams

NAND as NOT: $A \text{ NAND } A = A'$
NAND as AND: $(A \text{ NAND } B) \text{ NAND } (A \text{ NAND } B) = AB$
NAND as OR: $(A \text{ NAND } A) \text{ NAND } (B \text{ NAND } B) = A + B$
NOR as NOT: $A \text{ NOR } A = A'$
NOR as OR: $(A \text{ NOR } B) \text{ NOR } (A \text{ NOR } B) = A + B$
NOR as AND: $(A \text{ NOR } A) \text{ NOR } (B \text{ NOR } B) = AB$

Boolean algebra — additional exam identities

Law	OR form	AND form
Identity	$A + 0 = A$	$A \cdot 1 = A$
Null / domination	$A + 1 = 1$	$A \cdot 0 = 0$
Idempotent	$A + A = A$	$A \cdot A = A$
Complement	$A + A' = 1$	$A \cdot A' = 0$
Involution	$(A')' = A$	—
Absorption	$A + AB = A$	$A(A+B) = A$
Commutative	$A+B=B+A$	$AB=BA$
Associative	$A+(B+C)=(A+B)+C$	$A(BC)=(AB)C$
Distributive	$A(B+C)=AB+AC$	$A+BC=(A+B)(A+C)$

K-map — complete procedure

- For 3-variable and 4-variable maps, use Gray-code order 00, 01, 11, 10.
- For SOP, place/group 1s; for POS, place/group 0s.
- Valid group sizes are powers of two: 1, 2, 4, 8, 16.
- Groups are rectangular; diagonal-only grouping is invalid.
- Opposite edges are adjacent, so wrap-around groups are valid.
- Choose the largest useful groups first.
- Overlapping is allowed when it reduces the expression.
- Every required 1 (SOP) or 0 (POS) must be covered.
- Within a group, eliminate variables that change; retain variables that remain constant.

Q. Simplify $F(A,B,C)=\sum m(1,3,5,7)$.

Answer:

- Place 1s at minterms 1, 3, 5 and 7.
- These four cells form one group of four.
- A and B vary across the group and disappear.
- C remains 1 throughout.

- Therefore $F = C$.

Q. Simplify $F(A,B,C,D)=\Sigma m(0,2,8,10)$.

Answer:

- Use Gray-code ordering on both axes.
- The four listed cells form a valid group of four using wrap-around structure.
- B and D vary; A=0 and C=0 remain constant.
- Therefore $F = A'C'$.

Q. What is the difference between SOP and POS?

Answer:

- SOP means Sum of Products: OR of AND/product terms; K-map simplification normally groups 1s.
- POS means Product of Sums: AND of OR/sum terms; K-map simplification normally groups 0s.

3. MODULE II — EXPANDED CIRCUITS & STATE TABLES

Combinational vs sequential circuits

Feature	Combinational	Sequential
Output depends on	Present inputs only	Present inputs + stored state
Memory	No	Yes
Clock	Not inherently required	Commonly used
Examples	Adder, MUX, decoder	Counter, register, flip-flop

Half adder and full adder

Q. Explain a half adder.

Answer:

- A half adder adds two one-bit inputs A and B.
- Sum $S = A \oplus B$.
- Carry $C = AB$.
- It has no carry-in input.

Q. Explain a full adder.

Answer:

- A full adder adds A, B and carry-in C_{in} .
- Sum $S = A \oplus B \oplus C_{in}$.
- $C_{out} = AB + AC_{in} + BC_{in}$.
- A standard implementation uses two half adders and one OR gate.

A	B	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Ripple-carry adder

$FA_0 \rightarrow FA_1 \rightarrow FA_2 \rightarrow FA_3$

Each full adder receives the carry from the previous stage. The least-significant stage receives the initial carry-in. The main limitation is cumulative carry propagation delay.

Encoder / decoder

4-to-2 encoder input	Y1	Y0
D0 active	0	0
D1 active	0	1
D2 active	1	0
D3 active	1	1

For a basic 4-to-2 encoder with one active input: $Y1 = D2 + D3$; $Y0 = D1 + D3$.

A	B	Y0	Y1	Y2	Y3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

For the active-high 2-to-4 decoder above: $Y0=A'B'$, $Y1=A'B$, $Y2=AB'$, $Y3=AB$.

4-to-1 MUX

S1	S0	Selected input	Y
0	0	I0	I0
0	1	I1	I1
1	0	I2	I2
1	1	I3	I3

$$Y = I0S1'S0' + I1S1'S0 + I2S1S0' + I3S1S0.$$

1-to-4 DEMUX

S1	S0	Active output
0	0	$Y0 = D$
0	1	$Y1 = D$
1	0	$Y2 = D$
1	1	$Y3 = D$

For an active-high basic DEMUX: $Y0=DS1'S0'$, $Y1=DS1'S0$, $Y2=DS1S0'$, $Y3=DS1S0$.

Flip-flops — characteristic equations

Type	Characteristic / next-state equation	Key operation
SR	$Q(t+1)=S + R'Q(t)$ for the valid basic SR cases	Set / Reset / Hold; basic $S=R=1$ invalid
D	$Q(t+1)=D$	Stores D on active clock edge
JK	$Q(t+1)=JQ' + K'Q$	Hold / Set / Reset / Toggle
T	$Q(t+1)=T \oplus Q$	Hold when $T=0$; Toggle when $T=1$

Counters — state sequences

3-bit binary counter: $000 \rightarrow 001 \rightarrow 010 \rightarrow 011 \rightarrow 100 \rightarrow 101 \rightarrow 110 \rightarrow 111 \rightarrow 000$.

A normal n-stage binary counter has 2^n states when all states are used.

Ring counter example (4 stages, one-hot): $1000 \rightarrow 0100 \rightarrow 0010 \rightarrow 0001 \rightarrow 1000$.

Johnson counter example (4 stages, one conventional sequence): $0000 \rightarrow 1000 \rightarrow 1100 \rightarrow 1110 \rightarrow 1111 \rightarrow 0111 \rightarrow 0011 \rightarrow 0001 \rightarrow 0000$. An n-stage Johnson counter has $2n$ states.

Shift-register types

Type	Input	Output	Typical use
SISO	Serial	Serial	Delay / serial transfer
SIPO	Serial	Parallel	Serial-to-parallel conversion
PISO	Parallel	Serial	Parallel-to-serial conversion
PIPO	Parallel	Parallel	Temporary parallel storage

4. MODULE III — EXPANDED COMPUTER ORGANIZATION

Register set and common bus

Register	Typical role
PC	Address of next instruction
IR	Current instruction
AR/MAR	Memory address
DR/MDR	Data being transferred to/from memory
AC	Accumulator / intermediate arithmetic-logic result
TR	Temporary/intermediate storage

Register-transfer notation: $R2 \leftarrow R1$ means that the contents of R1 are transferred into R2 when the required control signals are asserted.

A common bus reduces the number of direct point-to-point connections. Source-selection logic chooses which register drives the bus; destination-load control selects the receiving register.

Computer instruction classes

Class	Purpose	Example form
Memory-reference	Access memory operand	LOAD R1,1000
Register-reference	Operate on registers	CLR AC
I/O	Transfer/control I/O data	IN / OUT
Data transfer	Move data between locations	MOV R1,R2
Arithmetic	Numeric operations	ADD R1,R2
Logical / bit	Bitwise operations	AND R1,R2
Shift	Move/rotate bits	SHL R1
Program control	Change control flow	JMP L1
Conditional branch	Branch on condition	BEQ L1
Subroutine call/return	Enter/leave reusable routine	CALL / RET

Instruction cycle — complete sequence

Fetch → Decode → Effective-address/operand preparation (when required) → Execute → Interrupt check → Next instruction.

Typical fetch micro-operations: $MAR \leftarrow PC$; $MDR \leftarrow \text{Memory}[MAR]$; $IR \leftarrow MDR$; $PC \leftarrow PC + 1$. Exact register names/timing depend on the processor organization.

Hardwired vs microprogrammed control

Aspect	Hardwired	Microprogrammed
Control generation	Fixed logic	Stored microinstructions
Speed	Usually faster	Usually slower due to sequencing/control-memory access
Flexibility	Low	Higher
Modification	Hardware redesign	Microprogram/control-memory changes
Complex instruction sets	More difficult	Convenient to organize

Microprogrammed control — chain

CAR → Control Memory → Microinstruction Register → Control Signals / CPU Micro-operations

Address sequencing determines the next microaddress. Common cases: sequential next address, unconditional branch, conditional branch, opcode mapping, and shared micro-routines/subroutine-style sequencing.

Microinstruction format

Field	Purpose
Control field	Specifies internal micro-operations/control signals
Condition / sequencing field	Specifies how the next microinstruction is selected
Address field	Provides a branch/next-address value when required

Horizontal-style microinstructions can expose many control signals directly; encoded/vertical-style fields reduce word width but require decoding. The exact bit layout is architecture-dependent.

Symbolic microinstruction example: $R2 \leftarrow R1$; $AC \leftarrow AC + DR$.

5. MODULE IV — EXPANDED PROCESSOR, MEMORY & I/O

General register organization

Multiple CPU registers share a datapath/bus. Control logic selects source registers, ALU operations and destination registers. The ALU may perform arithmetic/logic operations on selected register operands.

Stack organization — PUSH and POP

A stack is LIFO: the last item pushed is the first item popped.

PUSH: place a new item at the stack top and update SP according to the architecture.

POP: read/remove the top item and update SP according to the architecture.

Instruction formats

Format	Meaning
3-address	Three explicit operand/result locations; e.g. ADD R1,R2,R3
2-address	Two explicit operands; one may also be the destination
1-address	Often accumulator-based; one explicit operand address
0-address	Stack-based; operands implied at the top of the stack

Addressing modes — expanded

Mode	Effective operand/address	Example
Immediate	Operand is inside instruction	MOV R1,#25
Direct	Address field is memory address	LOAD R1,1000
Indirect	Address field points to location containing effective address	LOAD R1,(1000)
Register	Operand is in register	ADD R1,R2
Register indirect	Register contains effective address	LOAD R1,(R2)
Indexed	EA = base/address + index	LOAD R1,A[R2]
Relative	EA = PC + displacement	BR +20
Base/displacement	EA = base register + displacement	LOAD R1,8(R2)

RISC vs CISC

Feature	RISC	CISC
Instruction set	Smaller, regular	Larger, more complex
Length	Often fixed/limited formats	Often variable
Memory access	Typically load/store	May allow memory operands in many instructions
Control	Often simpler	Often more complex
Pipelining	Typically pipeline-friendly	Possible but decode/control may be more complex

Parallel processing

Parallel processing performs multiple computations or instruction/data operations concurrently using multiple functional units, cores, processors or other parallel resources.

Benefits: increased throughput, better hardware utilization, and faster execution of suitably parallel workloads.

Pipelining — performance and hazards

Typical instruction stages: IF → ID → EX → MEM → WB.

Pipelining primarily improves throughput by overlapping stages; it does not automatically reduce the latency of a single instruction.

Structural hazard: two operations require the same hardware resource simultaneously.

Data hazard: an instruction depends on a result that is not yet available.

Control hazard: a branch/jump changes the instruction stream.

For a balanced k-stage pipeline with enough independent work, ideal throughput improvement approaches k; real speedup is lower because of stalls, unequal stages, pipeline-register overhead and hazards.

Memory hierarchy and locality

Level	Typical characteristic
Registers	Fastest, very small, directly used by CPU
Cache	Small and fast; holds recently/frequently used blocks
Main memory	Larger active working storage
Secondary storage	Large persistent storage; slower access

Temporal locality: recently accessed data/instructions are likely to be accessed again.

Spatial locality: addresses near a recently accessed address are likely to be accessed soon.

Associative memory and cache

Associative memory (CAM) searches stored contents by matching a key rather than only using a conventional address. Comparisons can occur in parallel.

Cache hit: requested block/data is present. Cache miss: it must be obtained from a lower memory level. Hit ratio = hits / total accesses. Miss penalty = additional time to service a miss.

Cache mapping

Mapping	Rule	Main characteristic
Direct	Line = block number mod number of cache lines	Simple; possible conflict misses
Fully associative	Block may occupy any cache line	Flexible placement; tag comparison across lines
Set-associative	Block maps to one set, any line within that set	Compromise between direct and fully associative

Q. A cache has 8 lines. To which line does main-memory block 29 map under direct mapping?

Answer:

- Formula: cache line = block number mod number of cache lines.
- $29 \bmod 8 = 5$.
- Therefore block 29 maps to cache line 5.

Mapping vs replacement policy

Mapping determines where a block may be placed. Replacement policy determines which existing entry is evicted when a set/location is full. They are different concepts.

I/O organization

Mode	CPU involvement	Key idea
Programmed I/O	High	CPU polls device status and explicitly controls transfer
Interrupt-initiated I/O	Moderate	Device interrupts CPU when service is required
DMA	Low after setup	DMA controller transfers blocks between I/O and main memory

DMA is a concept-only topic in the official syllabus: understand the transfer path and reduced CPU involvement rather than vendor-specific controller details.

6. MODULE V — OPEN-ENDED / INTERNAL EXPANSION

The official syllabus lists Module V as an open-ended 12-hour component covering computer arithmetic algorithms and examples/classification of instruction types. It should not be treated as an ordinary replacement for Modules I–IV.

Binary multiplication — add-and-shift method

Basic idea: examine the multiplier bit-by-bit. For each 1 bit, add the appropriately shifted multiplicand; shift the multiplicand for the next bit. The exact register names and signed variant depend on the algorithm taught.

Q. Multiply 1011 (11) × 0101 (5) using partial products.

Answer:

- Multiplier bits from right to left are 1, 0, 1, 0.
- For bit 0: $1011 \ll 0 = 1011$.
- For bit 1: multiplier bit is 0, so partial product = 0000.
- For bit 2: $1011 \ll 2 = 101100$.
- For bit 3: multiplier bit is 0, so partial product = 0000000.
- Add partial products: $001011 + 101100 = 111111$ (63).

Binary division — repeated subtract/shift concept

A basic binary division algorithm compares a shifted divisor with the current partial remainder. When subtraction is possible, set the corresponding quotient bit and subtract; otherwise clear the quotient bit. Continue through the required positions.

Q. Divide 1101 (13) by 0011 (3).

Answer:

- $13 \div 3 = 4$ remainder 1.
- Binary quotient = 0100.
- Binary remainder = 0001.
- Always report both quotient and remainder and keep the chosen word width explicit.

Instruction-type examples

Instruction class	Example	Purpose
Memory-reference	LOAD R1,1000	Read memory operand
Register-reference	CLR AC	Operate directly on CPU register
I/O	IN R1,PORT	Receive I/O data
Data transfer	MOV R1,R2	Copy data
Data manipulation	AND R1,R2	Modify/process data
Arithmetic	ADD R1,R2	Addition
Logical/bit	XOR R1,R2	Bitwise operation
Shift	SHL R1	Shift bits
Program control	JMP LOOP	Unconditional control transfer
Conditional branch	BEQ DONE	Branch when condition is true
Subroutine call	CALL FUNC	Transfer to reusable routine
Subroutine return	RET	Return to saved point

Internal Module V checklist

- Binary addition and subtraction — fixed width and sign convention.

- Repeated add/shift multiplication — show partial products or register steps.
- Repeated subtract/shift division — show quotient and remainder.
- Memory-reference instructions — identify address/effective-address use.
- Register-reference instructions — identify direct register operation.
- I/O instructions — identify data/status transfer.
- Data transfer and data manipulation instructions.
- Arithmetic, logical and bit-manipulation instructions.
- Shift instructions.
- Program-control and conditional-branch instructions.
- Subroutine call and return.

7. EXAM-READY ANSWER TEMPLATES

For a 3-mark definition: definition + one key property + one example/use.

For a 6-mark circuit question: definition → labelled circuit/block diagram → truth table/equation → working → one key advantage/limitation/application.

For a 10-mark essay: definition → classification/components → neat labelled diagram → operation in steps → equations/tables → comparison/limitation → application/conclusion.

High-value answer checklists

Question type	Do not omit
Logic gate	Symbol + Boolean expression + truth table + function
K-map	Gray-code order + valid groups + wrap-around + final expression
Adder	Circuit + truth table + equations + carry explanation
Decoder/MUX	Block diagram + selection table + equation
Flip-flop	Inputs + clock + truth table + next-state behaviour
Counter	Clock arrangement + state sequence + delay/speed distinction
Instruction cycle	Fetch + decode + execute + interrupt check + micro-operations
Microprogrammed control	Control memory + CAR + sequencer + microinstruction
Addressing modes	Definition + effective-address rule + example
Pipeline	Stages + overlap + throughput + hazards
Cache	Hit/miss + hit ratio + mapping methods + formula
DMA	CPU setup + DMA controller + I/O-memory transfer + reduced CPU involvement

Common mark-losing errors

- Using ordinary binary order instead of Gray-code order in K-maps.
- Making diagonal or non-power-of-two K-map groups.
- Forgetting K-map edge/corner wrap-around.
- Confusing XOR with XNOR.
- Omitting C_{in} from a full-adder equation.
- Calling a ripple counter synchronous because it uses flip-flops.
- Confusing instruction format with addressing mode.
- Claiming pipelining always reduces single-instruction latency.
- Confusing cache mapping with replacement policy.
- Describing DMA as CPU-controlled byte-by-byte polling.
- Giving an architecture-specific fetch sequence as universally fixed without qualification.

8. CONSOLIDATED MASTER QUESTION BANK

Module I

1. Define binary number system.
2. State binary addition rules.
3. Define 1's complement and 2's complement.
4. Give the n-bit 2's-complement signed range.
5. Explain 1's-complement addition and end-around carry.
6. Explain 2's-complement addition and overflow.
7. Solve a binary subtraction using 1's complement.
8. Solve a binary subtraction using 2's complement.
9. Define AND, OR and NOT gates.
10. Explain NAND, NOR, XOR and XNOR.
11. State the universal property of NAND and NOR.
12. State Boolean algebra laws.
13. State the principle of duality.
14. State DeMorgan's theorems.
15. Define K-map.
16. Explain Gray-code ordering.
17. Explain K-map wrap-around adjacency.
18. Differentiate SOP and POS.
19. Simplify a 3-variable K-map.
20. Simplify a 4-variable K-map.

Module II

1. Define combinational circuit.
2. Differentiate combinational and sequential circuits.
3. Explain half adder with truth table and equations.
4. Explain full adder with truth table and equations.
5. Construct a full adder using two half adders.
6. Explain ripple-carry adder and propagation delay.
7. Explain 4-to-2 encoder.
8. Explain 2-to-4 decoder.
9. Explain 4-to-1 multiplexer.
10. Explain 1-to-4 demultiplexer.
11. Define latch and flip-flop.
12. Differentiate latch and flip-flop.
13. Explain SR flip-flop.
14. Explain D flip-flop.
15. Explain JK flip-flop.

16. Explain T flip-flop.
17. Compare synchronous and asynchronous counters.
18. Explain binary counter states.
19. Explain ring counter.
20. Explain Johnson counter.
21. Explain SISO, SIPO, PISO and PIPO registers.

Module III

1. Define instruction code, opcode and instruction format.
2. Explain PC, IR, MAR/AR, MDR/DR, AC and TR.
3. Explain common bus organization.
4. Classify computer instructions.
5. Define control unit.
6. Compare hardwired and microprogrammed control.
7. Explain instruction cycle.
8. Write typical fetch micro-operations.
9. Define control memory.
10. Define CAR and microinstruction register.
11. Explain address sequencing.
12. Explain opcode mapping.
13. Explain microinstruction format.
14. Explain symbolic microinstruction.
15. Differentiate control field, sequencing field and address field.

Module IV

1. Explain general register organization.
2. Explain stack organization, PUSH and POP.
3. Explain 3-, 2-, 1- and 0-address instruction formats.
4. Define addressing mode.
5. Explain immediate addressing.
6. Explain direct addressing.
7. Explain indirect addressing.
8. Explain register and register-indirect addressing.
9. Explain indexed and relative addressing.
10. Explain base/displacement addressing.
11. Compare RISC and CISC.
12. Explain parallel processing.
13. Define pipelining.
14. Explain arithmetic pipeline.
15. Explain instruction pipeline.

16. Explain structural, data and control hazards.
17. Explain memory hierarchy.
18. Explain temporal and spatial locality.
19. Explain main memory.
20. Explain associative memory/CAM.
21. Define cache hit, miss, hit ratio and miss penalty.
22. Explain direct cache mapping.
23. Explain fully associative mapping.
24. Explain set-associative mapping.
25. Differentiate mapping and replacement policy.
26. Explain programmed I/O.
27. Explain interrupt-initiated I/O.
28. Explain DMA.

Module V — Open-Ended

1. Explain binary addition and subtraction.
2. Explain multiplication by add-and-shift.
3. Explain division by subtract/shift.
4. Classify memory-reference instructions.
5. Classify register-reference instructions.
6. Explain I/O instructions.
7. Explain data-transfer instructions.
8. Explain data-manipulation instructions.
9. Explain arithmetic instructions.
10. Explain logical and bit-manipulation instructions.
11. Explain shift instructions.
12. Explain program-control instructions.
13. Explain conditional-branch instructions.
14. Explain subroutine call and return.

Coverage note

This question bank is intentionally broad. It is a study checklist, not a prediction of the exact examination paper. The official syllabus is the scope authority; supporting study material was used to expand the answer forms and practice coverage.