IAE ACADEMY INSTITUTE FOR ENGINEERS

CORE JOBS · INTERVIEW QUESTIONS

IAE Academy free study material · Updated 30 October 2026

Computer Organization and Architecture Interview Questions

INSIDE

- 22 questions with short answers
- 6 numericals with the working
- Grouped by topic

Companies do not publish their interview questions; these are ours.

Online, with links to the notes: **iaeacademy.in/study/computer-organization-architecture/interview-questions**

FREE COUNSELLING

Preparing for GATE or a core job? Talk to IAE Academy.

Or call **+91 84204 36524**

Every day, 10:30 am to 7:30 pm

Chat on WhatsApp

Instruction sets and addressing modes

1. What is an addressing mode, and what is the effective address?

An addressing mode is the rule an instruction uses to locate its operand. The effective address is the memory address that rule produces. In register indirect mode the register holds the address, so the effective address is the register's content. In indexed mode it is the base address plus the index register's value.

2. What is the difference between RISC and CISC instruction sets?

RISC uses a small set of simple, mostly fixed-length instructions, with memory touched only by load and store. CISC has many complex, variable-length instructions that can work on memory operands directly. RISC makes decoding and pipelining easier, while CISC gives denser code.

3. Why do some instruction sets use an expanding opcode?

An expanding opcode gives short opcodes to some instructions and longer ones to others inside a fixed-length word. Instructions with many address fields get short opcodes, and instructions with few fields use the spare bits to extend the opcode. This lets one fixed word size hold more distinct instructions.

4. A CPU has 32-bit instructions and 32 registers. An instruction has an opcode field that must encode 64 operations, two register operand fields and an immediate field that takes all the remaining bits. How wide is the immediate field?

Encoding 64 operations needs 6 opcode bits, and each of the two register fields needs 5 bits to name one of 32 registers. That uses 16 of the 32 bits, so the immediate field gets the remaining 16 bits.

> **WORKING**
>
> 1. Opcode bits: $\log_2 64 = 6$.
> 2. Each register field: $\log_2 32 = 5$ bits, so two fields use 10 bits.
> 3. Bits used so far: $6 + 10 = 16$.
> 4. Immediate bits: $32 - 16 = 16$.

ALU design and the control unit

5. What is the difference between a ripple-carry adder and a carry look-ahead adder?

A ripple-carry adder passes each carry from one full adder to the next, so its delay grows in proportion to the number of bits. A carry look-ahead adder computes the carries in parallel from generate and propagate signals, so its delay is much smaller, at the cost of more hardware.

6. Why is a microprogrammed control unit more flexible but slower than a hardwired one?

A microprogrammed unit stores the control signals for each step as microinstructions in a control memory. Changing or adding an instruction means changing stored microcode, not rewiring logic. It is slower because each step needs a control memory read, while a hardwired unit produces the signals directly through gates.

7. What is the difference between horizontal and vertical microinstructions?

A horizontal microinstruction has about one bit per control signal. It is wide, can activate many signals at once and needs no decoding. A vertical microinstruction encodes the signals into short fields that must be decoded, so it is narrower and needs less control memory, but allows less parallelism and is slower.

8. A microprogrammed control unit has a control memory of 128 microinstructions. Each microinstruction has a 36-bit control field and a next-address field that can select any word of the control memory. What is the control memory size in bits?

The next-address field must select one of 128 words, so it needs 7 bits. Each microinstruction is then $36 + 7 = 43$ bits wide, and 128 words of 43 bits give a control memory of 5504 bits.

WORKING

1. Next-address field: $\log_2 128 = 7$ bits.
2. Word width: $36 + 7 = 43$ bits.
3. Size $= 128 \times 43 = 5504$ bits.

Memory hierarchy, performance and cache mapping

9. What is locality of reference, and how does a memory hierarchy use it?

Locality of reference means programs tend to reuse recently used data (temporal locality) and to use data near it (spatial locality). A memory hierarchy keeps recently used and nearby blocks in a small fast memory. A high share of accesses is then served quickly, while the bulk of the data sits in cheaper, slower memory.

10. What is the difference between a direct-mapped and a set-associative cache?

In a direct-mapped cache each memory block can go in exactly one line, chosen by the index bits. Lookup needs one tag compare, but blocks sharing a line evict each other. In a k-way set-associative cache a block can go in any of k lines of one set. This needs k compares but gives fewer conflict misses.

11. Why do many caches use a write-back policy with a dirty bit?

Write-back updates only the cache on a write and marks the line dirty. The block goes to memory only when it is evicted. Repeated writes to one block then cost a single memory write, which saves bus traffic. Write-through writes memory every time, which keeps memory current but uses more bandwidth.

12. A cache has an access time of 2 ns and the main memory has an access time of 100 ns. The hit ratio is 0.95, and on a miss the memory is accessed only after the cache check fails. What is the average access time?

With hierarchical access, a hit costs the 2 ns cache time and a miss costs the cache check plus the 100 ns memory time. The average is $0.95 \times 2 + 0.05 \times 102 = 1.9 + 5.1 = 7$ ns.

WORKING

1. Hierarchical access: $T = h\,t_c + (1 - h)(t_c + t_m)$.
2. Hit part: $0.95 \times 2 = 1.9$ ns.
3. Miss part: $0.05 \times (2 + 100) = 5.1$ ns.
4. $T = 1.9 + 5.1 = 7$ ns.

13. A byte-addressable machine has 32-bit addresses and a 16 KB direct-mapped cache with 32-byte blocks. How many tag bits does each cache line need?

The 32-byte block needs 5 offset bits. The cache has $16384/32 = 512$ lines, so the index needs 9 bits. The tag is what remains of the address: $32 - 9 - 5 = 18$ bits per line.

WORKING

1. Block offset: $\log_2 32 = 5$ bits.
2. Number of lines: $16384/32 = 512$, so index $= 9$ bits.
3. Tag $= 32 - 9 - 5 = 18$ bits.

14. What are compulsory, capacity and conflict misses in a cache?

A compulsory miss is the first access to a block. A capacity miss happens because the cache cannot hold the working set. A conflict miss happens when too many blocks map to the same set and evict each other, a miss that a fully associative cache of the same size would avoid. So conflict misses occur in direct-mapped and set-associative caches, not in fully associative ones.

I/O interface

15. What is the difference between programmed I/O and interrupt-driven I/O?

In programmed I/O the CPU repeatedly checks the device status in a loop and moves each word itself, wasting time while the device is slow. In interrupt-driven I/O the CPU carries on with other work, and the device raises an interrupt when it is ready. Both still move each word through the CPU.

16. Why is DMA used for block transfers instead of interrupts for every word?

A DMA controller moves a whole block between device and memory without the CPU executing an instruction per word. The CPU only sets up the address, count and direction, then handles one interrupt at the end. This removes the per-word interrupt overhead, so the CPU is free for other work during large transfers.

17. What is the difference between burst mode and cycle stealing in DMA?

In burst mode the DMA controller holds the bus and transfers the whole block in one go. The CPU is blocked for that time, but the transfer is fast. In cycle stealing the controller takes the bus for one word at a time, so the CPU slows only slightly, but the block takes longer to move.

18. A device raises 2000 interrupts per second, and each interrupt takes the CPU 50 microseconds to service. What fraction of CPU time goes on servicing this device?

Each second the CPU spends $2000 \times 50\ \mu\text{s} = 100000\ \mu\text{s}$, which is 0.1 s, servicing interrupts from this device. As a fraction of each second, that is 0.1, so the device takes 10% of the CPU's time.

WORKING

1. Time per second: $2000 \times 50\ \mu\text{s} = 100000\ \mu\text{s}$.
2. That is 0.1 s in every second.
3. Fraction $= 0.1 = 10\%$.

Instruction pipelining and pipeline hazards

19. What is a read after write (RAW) hazard in a pipeline?

A RAW hazard occurs when an instruction needs a register value that an earlier instruction in the pipeline has not yet written back. Reading too early would give the old value, so the pipeline stalls the later instruction or forwards the result. It is the true data dependency between the two instructions.

20. Why does forwarding reduce stalls, and why does it not remove all of them?

Forwarding sends a result from the EX or MEM stage output straight to the input of a later instruction's EX stage, so that instruction need not wait for write-back to the register file. This removes many RAW stalls in a 5-stage pipeline. A load followed immediately by an instruction that uses its result still needs one stall, since the data appears only at the end of MEM.

21. What is a control hazard, and how does it cost cycles?

A control hazard arises on a branch: the address of the next instruction is not known until the branch is resolved, so the instructions fetched after it may be wrong. They are discarded or the pipeline stalls, costing one cycle for each stage after fetch up to the stage that resolves the branch. Branch prediction and delayed branching reduce it.

22. An instruction passes through five stages with delays of 100, 120, 150, 110 and 90 ps. The pipelined clock period is the longest stage delay plus 10 ps of register overhead, and the unpipelined machine takes the sum of the stage delays per instruction. What is the speed-up for a long run of instructions with no stalls?

Unpipelined, each instruction takes the sum of the stage delays, 570 ps. Pipelined, the clock period is the slowest stage plus overhead, $150 + 10 = 160$ ps, and with no stalls one instruction completes per cycle. The speed-up is $570/160 \approx 3.56$.

WORKING

1. Unpipelined time per instruction: $100 + 120 + 150 + 110 + 90 = 570$ ps.
2. Pipelined clock period: $150 + 10 = 160$ ps.
3. With no stalls, CPI is 1, so each instruction takes 160 ps.
4. Speed-up $= 570/160 \approx 3.56$.

FROM IAE ACADEMY

Preparing for GATE or a core job?

IAE Academy teaches GATE in seven papers in Ballygunge, Kolkata, and online. Counselling is free: ask us about batches, fees or your preparation.

FREE COUNSELLING

Chat on WhatsApp

Call +91 84204 36524

Open every day, 10:30 am to 7:30 pm.

Scan to chat on WhatsApp

GATE COURSES

Classroom batches in Ballygunge and online batches. Fees and details are on the course pages.

Scan: courses

See the courses

FREE ON IAEACADEMY.IN

Notes and solved previous-year questions for Computer Organization and Architecture, topic by topic.

Scan: the notes

Open the notes

Found this useful? Send it to a friend preparing for core-job interviews.

Send to a friend on WhatsApp

IAE Academy free study material. Companies do not publish their interview questions; these are ours. Prepared and published by IAE Academy, Kolkata. Free to read and free to share, unaltered and with attribution.

IAE INSTITUTE FOR ENGINEERS ACADEMY

GATE coaching in Kolkata since 2012 · EC, EE, IN, CS and IT, DA, BT and CE · in the classroom and online

227A Rash Behari Ave, Ballygunge, Kolkata 700019

+91 84204 36524 · iaeacademy.in · Updated 30 October 2026

Get directions