CS Exit Ethiopia

Complete Study Notes

Computer Science Exit Examination

1 selected course | 18 November 2026

Selected courses

01. Computer Organization and Architecture

https://csexitethiopia.com

abelasnake2003@gmail.com | Telegram: @Abechu7

Course contents

Each course begins on a new page. PDF bookmarks provide question, study-note section and revision-topic navigation.

Computer Organization and Architecture 6

How to use these study notes

START01 A practical study cycle

Start with a small set of questions before reading.

Mark the exact thing that stopped you: an unknown term, a rule, a calculation, a trace or a choice between alternatives.

Use the question lookup to open the relevant section.

Read one section, close it, and explain the rule in your own words.

Solve the question again and say why each other choice fails.

Then change one assumption or input and solve the variation.

This checks understanding beyond recognition.

A useful 60-minute session: 10 minutes of cold questions, 20 minutes of focused reading, 20 minutes of closed-book solving and 10 minutes updating an error log.

Adjust the timing to your needs.

Revisit mistakes after a gap, for example the next day, several days later and the following week.

These intervals are a practical starting plan, not a fixed learning law.

Record the section ID, your mistaken rule, the corrected rule and one boundary case.

Keep the log short enough to review.

Mix related courses during revision: programming with data structures; databases with security; networking with administration.

Explain how the same word changes meaning across contexts.

START02 What to emphasize

Emphasize definitions, key differences, algorithm costs, SDLC models, databases, networking and security.

Use short oral questions to check understanding and correct common misconceptions.

Combine those habits with the tracing, calculation, design and diagnosis demanded by the Ministry learning outcomes.

Allocate revision by both exam share and your error log.

A course with fewer items can still deserve more time if your foundations are weak.

The blueprint is a planning baseline, not a prediction of exact questions.

Definitions: distinguish nearby concepts, not just their names.

Calculations: write units, formulas, assumptions and intermediate steps.

Code and SQL: check validity before tracing the text; state any changes before analyzing a corrected version.

Algorithms and protocols: know the input, decision rule, state change, termination and failure cases.

Design choices: state the criterion.

A choice can change when cost, reliability, consistency or latency matters more.

Course group	Items in the 100-item blueprint	Revision implication
Web Programming	9	Reserve extra practice for HTML, CSS, JavaScript, PHP, client/server behavior and safe data handling.
Data Structures and Algorithms	7	Trace operations and compare their assumptions and costs.
Each of the other fourteen courses	6 each	Study all sixteen; do not abandon a whole course based on a short note or an alleged repeated-question list.

START03 How to handle exam traps

Underline NOT, FALSE, BEST and EXCEPT before comparing choices.

Check whether the stem supplies enough facts to select one answer.

A long scenario is not automatically an analysis question, and a familiar answer letter is not evidence.

When code is invalid, a diagram is absent, or two choices express the same rule, identify the clarification needed before answering.

Learn the correct rule and the smallest clarification needed.

Do not memorize an arbitrary key as a universal fact.

Distinguish worst case from average case, and an upper bound from a tight bound.

Check language version, SQL dialect, addressing assumptions, graph weights and tie-breaking rules.

For subnetting and scheduling, calculate rather than guess from a familiar number.

Use timed mixed papers only after you can explain your errors.

Revisit the reading sections behind recurring mistakes.

Computer Organization and Architecture

Complete Study Notes

COA01 | Architecture, organization and system structure

Architecture describes programmer-visible behavior: instructions, registers, data types, address spaces and I/O conventions.

Organization is the implementation: control signals, internal buses, pipelines and memory technology.

Different organizations can implement the same architecture; neither definition imposes a maximum lifetime.

Microarchitecture names the internal processor design.

CPU, main memory, I/O and their interconnection are the major structural parts.

The CPU contains an ALU, control unit and registers; a microcontroller is a complete integrated control system, not another mandatory CPU subunit.

A motherboard physically connects components.

Gates are fundamental logic building blocks; distinguish that level from CPU, memory, disk and motherboard components.

Computer Organization and Architecture module

COA02 | Stored programs and software roles

In the textbook Von Neumann model, instructions and data share an addressed memory system; the CPU normally advances sequentially unless control flow changes.

Harvard separates instruction and data memories or paths.

Modern systems may combine a unified address-space view with separate instruction/data caches, so “architecture used nowadays” is not an exclusive choice.

Separate program/data regions within one memory do not make a Harvard machine.

An operating system manages resources and provides services.

Applications solve user tasks.

A compiler translates source into target code; assembly is a symbolic form of machine instructions.

JIT compilation happens during program execution, often translating intermediate code.

Translation before execution alone does not distinguish JIT from an ahead-of-time compiler.

Machine-language programming depends on the register set, supported types and memory/addressing structure.

Computer Organization and Architecture module

COA03 | Gates, truth tables and symbols

AND is 1 only when all inputs are 1; OR is 1 if any input is 1.

NOT complements a bit.

NAND complements AND; NOR complements OR.

For two inputs XOR is 1 when inputs differ, and XNOR when they agree.

An enabled buffer copies its input.

A bubble at a gate output indicates inversion: an AND-shaped symbol with a bubble is NAND.

With n binary inputs, enumerate 2^n truth-table rows.

Evaluate complements, products and then sums.

A plus sign means Boolean OR, not integer addition.

Example

Worked microexample

- F=x+y'z at x=0,y=1,z=1: y'=0, so F=0 OR (0 AND 1)=0. Four independent inputs give 16 rows.

Computer Organization and Architecture module

COA04 | Boolean reduction and universal gates

Useful identities are A+0=A, A·1=A, A+A=A, AA=A, A+A'=1, AA'=0, A+AB=A and A(A+B)=A. De Morgan gives (AB)'=A'+B' and (A+B)'=A'B'.

Check a proposed simplification with a truth table if uncertain.

NAND and NOR are each universal: feeding the same input to both inputs constructs NOT; combine complemented gates to construct AND or OR.

For Karnaugh maps, use Gray-code order, group adjacent 1s in rectangles of powers of two, allow wraparound and overlap, cover every required 1, and remove variables changing within a group.

Do not group diagonally or include a required 0.

Example

Worked microexample

- AB+AB'=A(B+B')=A. A NAND A=A'; NAND followed by inversion gives AND.

Computer Organization and Architecture module

COA05 | Combinational versus sequential circuits

A combinational output depends on present inputs after propagation delay.

Adders, multiplexers, demultiplexers, encoders and decoders have no stored logical state.

A sequential circuit depends on stored state as well as inputs; registers, counters and flip-flops are examples.

This is the defensible criterion when comparing a shift register against decoder, encoder and multiplexer.

A flip-flop stores one bit while powered.

State retention during operation does not mean nonvolatile storage.

Ordinary flip-flops and SRAM lose state when power is removed.

Clocked sequential circuits update at designated clock events; combinational logic does not inherently require a clock.

Computer Organization and Architecture module

COA06 | Flip-flop and counter state traces

For an active-high SR latch, S=1,R=0 sets, S=0,R=1 resets, and S=R=0 holds; S=R=1 is forbidden for the NOR-latch model.

Active-low NAND-latch conventions differ.

A D flip-flop copies D at its active edge.

JK holds for 00, resets for 01, sets for 10 and toggles for 11.

T toggles when T=1 and holds when T=0.

Trace a sequential circuit one clock at a time from the given initial state.

A k-bit binary counter has 2^k states and wraps modulo 2^k unless extra control changes the sequence.

Ripple counters propagate changes asynchronously; synchronous counters share a clock.

Example

Worked microexample

- A 3-bit counter at 110 advances to 111, then 000. A T flip-flop starting at 0 with T inputs 1,0,1 gives states 1,1,0.

Computer Organization and Architecture module

COA07 | Adders, selectors and code converters

A half-adder has sum A XOR B and carry AB.

A full-adder includes carry-in: sum=A XOR B XOR Cin and carry=AB+Cin(A XOR B).

Cascading full-adders propagates carry across a word.

A 2^k-to-1 multiplexer uses k select bits to choose one data input.

A demultiplexer routes one input to a selected output.

A k-to-2^k decoder asserts the selected output, subject to enable and active-level conventions.

An ordinary encoder assumes one active input; a priority encoder defines which wins if several are active.

Registers are storage, not combinational code converters.

Example

Worked microexample

- For a full-adder with A=B=Cin=1, binary total is 11: sum=1 and carry=1. An 8-to-1 mux needs 3 select bits.

Computer Organization and Architecture module

COA08 | Base conversion and binary codes

In base b, a digit at position i has value digit×b^i. Binary uses 0-1, octal 0-7 and hexadecimal 0-9,A-F; F=15.

Convert a positive integer to binary by repeated division by 2 and read remainders upward.

For fractions, repeatedly multiply by 2 and read integer parts downward.

Group binary bits in threes for octal and fours for hexadecimal, outward from the radix point.

BCD encodes each decimal digit separately in four bits; BCD 12 is 0001 0010, while ordinary binary 12 is 1100.

ASCII is a character code, not a physical I/O interface.

Example

Worked microexample

- 1011_2=8+2+1=11. $2F_{16}$=47. 101.101_2=5.625. Decimal 13 gives remainders 1,0,1,1 and hence 1101_2.

Computer Organization and Architecture module

COA09 | Signed arithmetic, subtraction and overflow

For n-bit unsigned values the range is 0 through 2^n−1.

Two’s complement ranges from −2^(n−1) through 2^(n−1)−1.

Negate by inverting all n bits and adding one.

Compute A−B as A+(two’s complement of B), retaining n result bits.

Signed magnitude and one’s complement have two zero encodings; two’s complement has one.

Unsigned carry and signed overflow are different.

Signed addition overflows when equal-sign operands produce an opposite-sign result.

Keep the width fixed throughout the calculation.

Example

Worked microexample

- In 4 bits, 7−3 = 0111+1101 = 1 0100; discard carry to get 4. But 0111+0001=1000 is signed overflow, since +8 is outside −8…+7.

Computer Organization and Architecture module

COA10 | Floating point and arithmetic pipelines

A normalized binary floating-point value is sign×significand×2^exponent.

For ordinary IEEE binary32 normal values, value=(−1)^s(1.f)2^(e−127); binary64 uses 11 exponent bits and bias 1023.

Special exponent patterns represent zero/subnormal values or infinities/NaNs, so the implicit leading 1 is not universal.

For floating addition, compare exponents, shift the smaller significand to align, add or subtract, normalize and round.

Alignment may lose low-order bits.

Overflow concerns exponent range; rounding error concerns finite precision.

An arithmetic pipeline overlaps these steps for different operand pairs; it need not perform all steps in one cycle.

Example

Worked microexample

- $1.10_2 \times 2^3 + 1.00_2 \times 2^1$ becomes $(1.10_2 + 0.01_2) \times 2^3 = 1.11_2 \times 2^3 = 14$.

Computer Organization and Architecture module

COA11 | Register transfers, buses and memory cycles

R2←R1 copies R1 into R2; it does not erase R1.

A control condition P:R2←R1 performs the transfer only when P is true at the relevant clock event.

Simultaneous transfers read old register values; sequential statements can use a newly written value.

A common bus selects one source at a time, while destination load signals select receivers.

A read places an address in MAR/AR, asserts read, waits for memory and captures data in MDR/MBR/DR. A write supplies both address and data and asserts write.

The ALU computes; the control unit sequences and enables the transfers.

A microoperation is one elementary operation on register contents.

Example

Worked microexample

- With R1=3,R2=7, simultaneous R1←R2,R2←R1 swaps to 7,3. Executed sequentially, both end at 7.

Computer Organization and Architecture module

COA12 | Address, data and control buses

Address signals select a location; data signals carry values; control signals coordinate read/write, timing, requests and acknowledgements.

In the simple CPU-master model, addresses flow from CPU to memory, while data travels in either direction.

A DMA controller can also become bus master.

Aggregate control signals include both CPU outputs and device inputs; both data and control buses can therefore be bidirectional under that scope.

With a address bits there are 2^a addressable locations.

Multiply by bytes per location before calling this a byte capacity.

Word width, data-bus width, instruction width and address width need not match.

Example

Worked microexample

- 32 address bits and byte addressing give 2^32 bytes=4 GiB (often called 4 GB in introductory questions). A 16-bit data bus does not itself imply a 16-bit address space.

Computer Organization and Architecture module

COA13 | Logic masks and shifts

To set selected bits use A OR mask; to clear them use A AND NOT(mask); to toggle use A XOR mask; to retain only selected bits use A AND mask.

A XOR A clears the entire register.

Clearing with XOR is valid only with the appropriate operand, not any arbitrary mask.

Logical shifts insert zero.

Arithmetic right shift replicates the sign bit.

Left shift inserts zero and may overflow; it does not preserve the sign in every case.

Rotate returns a shifted-out bit at the opposite end.

SISO shift registers delay serial streams; SIPO collects a word and PISO serializes a parallel word.

Example

Worked microexample

- A=1010, mask=1100: set→1110, selective clear→0010, toggle→0110. Arithmetic-right 1100 gives 1110 (−4 to −2 in two's complement); logical-right gives 0110.

Computer Organization and Architecture module

COA14 | CPU registers and control-word construction

PC identifies the next instruction to fetch.

IR holds the fetched instruction; MAR holds an address used for the current memory access; MDR holds transferred data.

The accumulator holds operands/results, SP tracks the stack, and status flags report conditions. "Address of the instruction" without a phase can describe PC or MAR.

Registers and cache both store temporary information, but registers are explicit execution operands and cache holds copies of memory blocks.

A control word chooses ALU sources, destination and operation.

Use the machine's encoding table rather than guessing opcodes.

In this 14-bit control-word example, SELA, SELB and SELD each have 3 bits, followed by a 5-bit operation field.

Example

Worked microexample

- For R1←R2−R3 in this encoding, select 010,011,001 and subtraction 00101: 010 011 001 00101. Source and destination order matters.

Computer Organization and Architecture module

COA15 | Fetch, decode, execute and next-PC choice

A conceptual instruction cycle fetches an instruction, decodes opcode/addressing fields, obtains operands, executes, writes results where required and checks permitted interrupts.

A five-stage implementation may name stages IF, ID, EX, MEM and WB; these stages are not mandatory for every architecture.

A sample fetch is MAR←PC; MDR←M[MAR]; IR←MDR; PC←PC+instruction_length.

A branch can replace the sequential PC later.

Thus "which phase determines the next instruction?" needs to distinguish reading the current PC from calculating a branch target.

The control unit coordinates fetching and data flow; the ALU performs arithmetic and logic.

A write-back/result-storage phase is not another fetch.

Computer Organization and Architecture module

COA16 | Opcode fields and address-count formats

An opcode specifies the operation; operand fields locate data or registers, and a mode field explains interpretation.

A 4-bit opcode permits at most 16 encodings, not necessarily 16 implemented instructions.

A 6-bit operand-reference field has 64 encodings; what they designate depends on the machine.

Three-address ADD X,Y,Z means X←Y+Z under destination-first notation.

Two-address ADD X,Y overwrites X with X+Y. One-address accumulator ADD X means AC←AC+M[X].

Zero-address stack ADD uses two implicit top-of-stack operands.

Count explicit operand fields; an instruction such as INC X does not become an accumulator instruction merely because it has one written operand.

Computer Organization and Architecture module

COA17 | Stacks and evaluation order

A stack is last-in, first-out.

PUSH adds an item and POP removes the logical top.

A register stack uses registers; a memory stack uses RAM and a stack pointer.

Check whether SP points to the top or the next free location, whether addresses increase or decrease, and whether the update occurs before or after access.

Overflow means pushing a full stack; underflow means popping an empty one.

Zero-address arithmetic typically pops the second operand first, then the first operand; subtraction therefore needs careful order.

PUSH and POP may still contain an explicit address even on a stack machine.

Example

Worked microexample

- For (A+B)×(C+D): PUSH A; PUSH B; ADD; PUSH C; PUSH D; ADD; MUL. For a downward-growing full stack, PUSH commonly decrements SP before storing and POP reads before incrementing.

Computer Organization and Architecture module

COA18 | Effective addresses and addressing modes

Let A be an instruction field, R a register value and M[x] the contents at x. Immediate uses value A; direct uses EA=A and operand M[A]; indirect uses EA=M[A] then operand M[EA].

Register uses operand R; register-indirect uses EA=R. Indexed/base displacement uses EA=R+A; PC-relative uses the architecture-defined PC plus a signed displacement.

Auto-increment/decrement also updates the register by operand size.

A transfer between a register and memory does not uniquely specify addressing mode; either direct, indexed or indirect addressing may produce it.

Example

Worked microexample

- A=100,R=20,M[100]=300,M[300]=9: direct operand is 300; indirect operand is 9; indexed EA is 120. An immediate field 100 is the value 100, not M[100].

Computer Organization and Architecture module

COA19 | Instruction classes and control flow

Data-transfer instructions copy values.

Arithmetic/logic instructions compute values.

TEST/CMP examine values and set condition information, often without retaining the arithmetic result.

Branch instructions choose a target, conditionally or unconditionally.

CALL saves a return point; RETURN restores it.

SKIP, where supported, bypasses the next instruction when its condition holds. “Jump to the next instruction” does not clearly mean skip.

External interrupts arise from devices or timers; synchronous internal exceptions arise from the executing instruction, such as divide errors, illegal opcodes or access violations.

A software interrupt is deliberately requested by an instruction.

Arithmetic overflow trapping depends on architecture and instruction; ordinary overflow flags need not trigger an interrupt.

Computer Organization and Architecture module

COA20 | RISC, CISC and control implementation

Textbook RISC emphasizes simple operations, regular encodings, load/store memory access, plentiful registers and pipeline-friendly control.

CISC commonly supplies more complex operations, richer addressing modes and variable lengths.

Hardwired control directly generates control signals; microprogrammed control sequences stored control words.

These are tendencies, not universal guarantees.

One instruction completing per pipeline cycle is throughput, not proof that every instruction has one-cycle latency.

Modern RISC instructions can take multiple cycles and modern CISC processors can translate instructions into simpler internal operations.

Neither label guarantees lower power, cost or execution time.

The semantic gap is the distance between high-level constructs and machine operations; reducing it is associated with richer instructions, not a universal shared objective of both design approaches.

Computer Organization and Architecture module

COA21 | x86 register names and classification

In Intel x86 notation, AX/BX/CX/DX, BP/SP and SI/DI are 16-bit general-purpose registers, with conventional special roles.

AL is AX's low byte and AH its high byte; analogous byte names apply to BX/CX/DX.

EAX is 32-bit and RAX 64-bit.

FLAGS/EFLAGS/RFLAGS report status; IP/EIP/RIP identifies instruction position.

CS and SS are code/stack segment registers.

DX is programmer-visible; IR/MDR are conceptual internal registers.

SP's stack role does not remove it from Intel's general-purpose register classification.

EAX/EIP existed with the 80386, before Pentium; R8-R15 belong to the 64-bit extension, not original Pentium.

State the textbook convention when using a narrower "multipurpose" register category.

Intel Software Developer Manual, Volume 1

COA22 | x86 instruction legality and implicit operands

Assume Intel destination-first syntax and ordinary legacy x86 forms.

MOV AL,BP is invalid because operand widths differ (8 versus 16 bits).

INC CX and DEC CX are valid word operations.

Segment-register MOV has register/memory forms, not a direct immediate-to-segment form: load an immediate into AX, then MOV DS,AX.

Unsigned MUL with a byte operand multiplies AL by that byte and writes AX.

With a word operand it multiplies AX and writes DX:AX.

Thus AL=8 and BH=10 gives AX=80 after MUL BH; CL is irrelevant.

MOVS and LODS are string transfer operations, XCHG exchanges operands, and INC is arithmetic. “Different” requires the transfer-versus-arithmetic criterion.

Intel Software Developer Manual, Volume 2B

COA23 | Memory hierarchy and timing measures

A usual speed/cost-per-bit order is registers, L1/L2/L3 cache, DRAM, SSD, HDD and archival tape.

Faster levels are generally smaller and more expensive per bit.

Access time measures latency of a read/write; cycle time is the minimum interval before another operation and may include recovery.

Transfer rate measures data per time, not individual-request delay.

Temporal locality reuses recent data; spatial locality uses nearby locations.

Cache copies memory blocks near the CPU.

Hit ratio=hits/accesses; miss ratio=1−hit ratio.

Distinguish capacity, latency, bandwidth and hit ratio before comparing alternatives.

Example

Worked microexample

- For 90 hits in 100 accesses, hit ratio=0.9. If hit time is 2 ns and an additional miss penalty is 50 ns, AMAT=2+0.1×50=7 ns.

Computer Organization and Architecture module

COA24 | RAM, firmware and memory-chip capacity

SRAM uses powered bistable storage: fast and costly per bit, without periodic refresh.

DRAM stores charge and requires refresh: denser and commonly used for main memory.

Both are ordinarily volatile.

ROM and flash retain contents without power; erasable/programmed variants mean “ROM can never be modified” is too broad.

Firmware is low-level system code, commonly in nonvolatile storage.

For a chip described as 2^a×w, a is the number of address inputs and w the data width.

Combine chips in parallel to increase word width and in banks to increase the number of words.

Distinguish address decoding from data selection.

Example

Worked microexample

- Eight 1K×1 chips in parallel make 1K×8 storage: 10 shared address lines and 8 data bits. Four such banks make 4K×8 and require 12 address bits overall.

Computer Organization and Architecture module

COA25 | Cache placement, index bits and conflicts

Let B be block size in bytes and L the number of cache lines.

Block number=floor(address/B); offset=address mod B. Direct mapping uses line=block mod L. Fully associative placement permits any line.

A W-way set-associative cache has S=L/W sets and set=block mod S; any of W ways in that set may hold the block.

For power-of-two sizes, offset bits=log2 B, index bits=log2 S and remaining address bits form the tag.

Compare tags and valid bits to identify a hit.

Two blocks with the same direct-mapped index cannot reside together.

Associations reduce placement conflicts but require more tag comparisons.

Example

Worked microexample

- 16-byte blocks and four direct-mapped lines: addresses 0 and 64 have block numbers 0 and 4, both index 0, so alternating them causes conflict misses.

Computer Organization and Architecture module

COA26 | Cache replacement and write-policy traces

FIFO evicts the earliest inserted line; LRU evicts the least recently accessed line.

On each access, record hit/miss, the set involved and policy state.

Hits update LRU order but do not reset FIFO insertion order.

Direct mapping has no replacement choice within a line.

Write-through updates the lower level on each write; write-back modifies cache and marks a dirty bit, writing to the lower level on eviction.

Write-allocate fetches a missed block before writing; no-write-allocate writes below without installing it.

L1/L2 describe hierarchy levels; split cache means separate instruction/data caches and unified means combined.

Internal/external location is another classification, not synonymous with split/unified.

Example

Worked microexample

- Two fully associative lines, references A,B,A,C: all except the second A are misses. LRU evicts B for C; FIFO evicts A.

Computer Organization and Architecture module

COA27 | Virtual addresses, MMU and page faults

Virtual memory gives processes translated address spaces and protection; backing storage can hold nonresident pages.

The MMU translates virtual/logical addresses to physical addresses, using page tables and often a TLB of recent translations.

Virtual memory is not just extra disk disguised as RAM.

With page size P, page=floor(virtual_address/P), offset=virtual_address mod P and physical_address=frame×P+offset.

A valid nonresident page access faults and may require loading data; invalid or forbidden accesses can also fault.

A cache miss, TLB miss and page fault are different events; not every page fault reads a disk, since zero-fill and already-resident shared pages exist.

Example

Worked microexample

- With 1024-byte pages, address 2053 is page 2, offset 5. If page 2 maps to frame 7, physical address=7173.

Computer Organization and Architecture module

COA28 | Persistent storage and access methods

An HDD stores magnetic patterns on tracks divided into sectors; reads/writes access addressed blocks while platters rotate.

A head normally stays over a track while reading but moves radially for seeks.

It does not need continuous power to preserve stored bits.

SSDs use nonvolatile semiconductor storage without mechanical seek.

Optical media use lasers; tape is suited to sequential streaming and archives.

An Ethernet cable is a communication medium, not a disk or RAM.

In textbook access taxonomy, tape is sequential, disk is direct access with positioning delay, RAM is random access with broadly address-independent timing, and associative access matches

content. “Random-access disk” is also common informal usage; specify the taxonomy when comparing direct and random access.

Optical access ranges such as 100-300 ms are historical examples, not universal device specifications.

Computer Organization and Architecture module

COA29 | I/O interfaces, registers and commands

A peripheral performs input, output or storage outside the CPU’s execution core; CPU itself is not a peripheral.

Input units encode external information and output units present processed results.

A device controller handles device protocol/timing; its interface commonly exposes data, status and control registers.

Buffers absorb speed differences and error/status bits report conditions.

Control commands start actions, test/status commands inspect readiness or errors, read commands bring device data toward the system, and write/output commands send system data to the device. “Issue” describes sending a command, not a standard fourth command category.

A controller is hardware; a driver is software that operates it.

Computer Organization and Architecture module

COA30 | Polling, interrupts and DMA

Programmed/polled I/O repeatedly checks status and lets the CPU move each item.

Interrupt-driven I/O lets the CPU do other work until an event requests service; the handler still consumes CPU time to service the device.

DMA lets a controller move a block directly between device and memory after CPU setup.

Completion or errors can interrupt the CPU.

DMA does not eliminate CPU involvement in configuration, ownership, address mapping or completion.

Burst mode can hold the bus for a block; cycle stealing takes individual transfer opportunities, and transparent DMA uses idle opportunities.

DMA reduces per-word CPU work but competes for memory bandwidth.

Choose it for substantial blocks; mapping devices to addresses is a separate design dimension.

Example

Worked microexample

- For 1000 words, polling or interrupt transfer may require CPU work per word. DMA replaces much of that work with setup and completion, although the actual memory transfers still take time.

Computer Organization and Architecture module

COA31 | I/O address spaces, priorities and handshaking

Memory-mapped I/O assigns device registers addresses in the memory address space and uses suitable load/store operations.

Isolated/port-mapped I/O has a separate port space and special instructions such as IN/OUT.

MMIO must respect device access widths, ordering and side effects; a status read is not necessarily a harmless RAM read.

Asynchronous handshaking uses request/ready and acknowledgement signals to tolerate unequal speeds: sender presents data and requests, receiver accepts and acknowledges, then each withdraws its signal.

A single strobe lacks this feedback.

Interrupt priority selects among simultaneous requests, using schemes such as polling, daisy chains or priority hardware.

Serial sends bits over time; parallel transfers multiple bits on multiple lines, without guaranteeing higher real-world speed.

Computer Organization and Architecture module

COA32 | Named interfaces: choose by connection role

USB is a serial peripheral interconnect used by keyboards and external storage; RS-232 specifies a traditional point-to-point serial electrical/interface standard.

Both RS-232 and USB use serial communication; distinguish their interface roles and specifications.

HDMI carries digital video and audio.

Ethernet connects network devices; it is not memory.

SATA is a storage interface; legacy IDE/PATA and SCSI represent other storage connections, while iSCSI carries SCSI commands over IP.

PCI is a peripheral expansion interconnect, not a disk protocol; connecting unlike bus segments requires suitable bridge/controller logic. “Extends processor-bus connectivity” without topology does not uniquely identify PCI or Multibus.

POSIX is a software interface standard with operations such as open/read/write/close; ASCII is a character encoding.

An unqualified “standard PC disk interface” needs an era or device requirement.

USB Implementers Forum, mass storage transport specification

Texas Instruments AN-216

HDMI Licensing Administrator specifications overview

Linux libATA Developer Guide

COA33 | Analog interfaces and component identification

An ADC measures an analog input and produces a digital code; a DAC produces an analog output from a digital code.

ADC080x parts are 8-bit successive-approximation analog-to-digital converters; DAC0830 is a digital-to-analog converter, while 82C55 denotes a programmable parallel I/O interface, not an ADC.

Component names are not interchangeable with the direction of conversion.

For an ideal n-bit converter spanning a voltage range V, one code step is approximately V/2^n; actual endpoint conventions and specifications must be stated.

Quantization rounds an analog range into discrete codes; sample rate and resolution measure different properties.

Example

Worked microexample

- An ideal 8-bit ADC over a 5.12 V span has 256 codes and a 0.02 V step. Greater bit depth reduces the quantization step for the same span.

Texas Instruments ADC0804-N

COA34 | Execution-time calculations and bottlenecks

CPU time=instruction_count×CPI×clock_period=instruction_count×CPI/clock_rate.

CPI may be a weighted average across instruction classes; IPC is its reciprocal only under a consistent whole-program definition.

Raising clock frequency does not guarantee a faster program when instruction count or stalls increase.

Throughput counts completed work per unit time; latency measures one job’s delay.

When a fraction f of execution is sped up by factor s, overall speedup=1/((1−f)+f/s).

The unaffected fraction limits gains.

Compare the same workload and include memory and branch penalties instead of equating clock speed with performance.

Example

Worked microexample

- 10^6 instructions, CPI=2 and 1 GHz take 2 ms. If half the time becomes twice as fast, overall speedup=1/(0.5+0.25)=4/3, not 2.

Computer Organization and Architecture module

COA35 | Pipelines: overlap and timing

A pipeline overlaps stages of different instructions; it primarily improves throughput.

In an ideal k-stage single-issue pipeline, n instructions take k+n−1 cycles when stages are balanced and there are no stalls.

Without overlap, comparable work takes nk stage cycles.

The ideal steady-state completion rate is one instruction per cycle, while an individual instruction still spans multiple stages.

Choose the pipeline clock using the slowest stage plus register overhead.

Fill/drain overhead matters for short streams. “Fetch/decode one instruction while executing another” describes pipelining, whereas issuing several instructions in one cycle requires multiple-issue resources.

Example

Worked microexample

- A five-stage pipeline processing 10 instructions takes 14 ideal cycles versus 50 without overlap: speedup=50/14≈3.57. It does not reach the limiting factor of five for this short stream.

Computer Organization and Architecture module

COA36 | Pipeline hazards and instruction-level parallelism

Structural hazards compete for one resource.

Data hazards arise when an instruction needs a result not yet available; forwarding can supply results early, but some dependencies still stall.

Control hazards arise when the next PC depends on an unresolved branch.

Branch prediction guesses direction/target and continues speculatively; wrong guesses discard work.

Superscalar execution issues multiple instructions to multiple functional units in a cycle when dependencies permit.

Out-of-order execution schedules ready operations before stalled older ones while preserving architectural results and exception behavior.

Dynamic resource allocation assigns available internal entries/units at runtime.

These mechanisms exploit instruction-level parallelism; wider fetch improves instruction supply, not necessarily individual memory latency.

Computer Organization and Architecture module

INFLIBNET Computer Architecture, Multiple Issue Processors I

COA37 | Multiprocessing, vectors and Flynn classification

Flynn classifies instruction and data streams: SISD has one of each; SIMD applies one instruction stream to multiple data elements; MIMD uses multiple instruction and data streams; MISD has multiple instruction streams for a common data stream and is uncommon.

A vector instruction describes an operation over a sequence of elements; physical lanes may process them in batches, not all in one instant.

SMP processors share memory/I/O and have peer roles under an OS.

Distributed-memory systems communicate between separate address spaces.

More cores increase available parallel work, but serial dependencies and coordination limit speedup. “Core i7” is a product family rather than one core count; a model number is essential.

Computer Organization and Architecture module

COA38 | Historical conventions and processor-width caveats

The familiar generation convention associates first with vacuum tubes, second with transistors, third with integrated circuits and fourth with microprocessor/VLSI-era personal computers.

Stored-program machines arose in the first era.

High-level languages became prominent around the transition into the second; introduction dates do not fit every generation boundary.

Magnetic-core memory spanned more than one era, so “witnessed use” cannot identify exactly one.

Several generations improved performance.

Integration generally reduced individual component dimensions while capacity, density and processor performance grew; these are trends rather than guarantees for every model.

Intel’s 80386 introduced 32-bit x86 execution in 1985, before 80486/Pentium; the 80286 was a 16-bit predecessor.

An n-bit processor label does not force its PC, IR or address register to be n bits.

Columbia University Computing History, Computer Generations

Intel 1987 Annual Report

Computer Organization and Architecture module

COA39 | Identify architecture assumptions before answering

Main memory, registers and cache can all provide temporary storage; identify the scope and role. Memory can also include nonvolatile ROM/flash, so distinguish RAM when comparing power-off data retention with disk storage.

Architecture lifetime, power efficiency and generational improvements require assumptions; definitions alone do not make these exclusive.

For a TEST-instruction question, identify the instruction and the complete statements being compared before drawing a conclusion.

Identify a logic gate from its actual graphical symbol.

An opcode can be identified from an explicit description of its operation-field role even without a figure.

Check that the question includes the phrase, choices or diagram needed to apply the relevant architecture rule.

Computer Organization and Architecture module

COA-S01 | Model-specific core and thread counts

Intel’s desktop comparison chart lists the Core i7-920 with four physical cores and eight hardware threads. Core count and thread count are distinct specification fields: use the exact processor model and the field requested rather than inferring a count from the Core i7 family name.

Intel Core i7 Desktop Processor Comparison Chart

COA-S02 | Buses as shared communication paths

A bus is a set of common signal lines used to transfer binary information among connected units. In a parallel register bus, one line carries each bit position; control signals select a source and enable the intended destination. Storage registers and memory retain information, while the bus provides a transfer path.

Computer Organization and Architecture module · page 40

Computer Organization and Architecture module · page 6

COA-S03 | The basic single-accumulator memory instruction

In the textbook single-accumulator organization, a memory-reference instruction has an operation part and an address part. The address locates a memory operand; the operation combines that operand with the accumulator as required. The address field is not the operand value itself, and this simple format does not describe every addressing mode or instruction set.

Computer Organization and Architecture module · page 47

COA-S04 | Textbook instruction-set and control tendencies

RISC means Reduced Instruction Set Computer; CISC means Complex Instruction Set Computer. The textbook RISC design pairs a relatively large register set, load/store access and easily decoded instructions with hardwired rather than microprogrammed control and an efficient pipeline. These are design tendencies, not guarantees of one-cycle latency or lower power for every implementation.

Computer Organization and Architecture module · page 77

Computer Organization and Architecture module · page 78

COA-S05 | Instruction and stack pointers in 64-bit x86

In Intel 64-bit mode, RIP holds the offset of the next instruction to execute. RSP is the 64-bit stack pointer, extending the ESP/SP stack role. These pointers are distinct from CS and SS, which are code and stack segment registers.

Intel 64 and IA-32 Architectures Software Developer's Manual, Volume 1 · §§3.4-3.5 and 6.1

COA-S06 | Exclusive-OR notation

The circled plus symbol ⊕ denotes exclusive OR. For two Boolean inputs, A⊕B is 1 exactly when the inputs differ; an overbar or output inversion would complement this result.

Texas Instruments SNx4HC86 Quadruple 2-Input XOR Gates · description

COA-S07 | Bistable one-bit state

A flip-flop is bistable: it has two stable states representing 0 and 1. While powered, it can retain one bit until an input or designated clock event changes its state. Multiple flip-flops are required to represent a multi-bit counter state.

Computer Organization and Architecture module · page 16