CS Exit Ethiopia

Revision Outlines

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 3

Computer Organization and Architecture

Revision Outlines

Architecture and system structure | Exam focus

- Architecture versus organization
  - Programmer-visible instructions and registers
  - Data types and address spaces
  - I/O conventions
  - Implementation: signals, buses, pipelines, memory
  - Same architecture, different organizations
  - No definition-imposed maximum lifetime
  - Microarchitecture: internal processor design
- System structural components
  - CPU, memory, I/O, interconnection
  - CPU: ALU, control unit, registers
  - Microcontroller: integrated control system
  - Microcontroller ≠ mandatory CPU subunit
  - Motherboard: physical connections
  - Gates: fundamental logic building blocks
  - Missing gates option: ambiguous answer

Stored programs and software | Exam focus

- Von Neumann versus Harvard
  - Shared addressed instructions and data
  - Sequential execution unless control changes
  - Harvard: separate memories or paths
  - Unified address space, split caches
  - Modern architecture: nonexclusive labels
  - Separate regions ≠ Harvard
- Software roles
  - OS: resources and services
  - Applications: user tasks

- Compiler: source→target code
- Assembly: symbolic machine instructions
- JIT: compilation during execution
- Intermediate-code translation
- Before execution ≠ uniquely JIT
- Machine language: registers, types, addressing

Gates and truth tables

- Logic gate behavior
  - AND: all inputs 1
  - OR: any input 1
  - NOT: bit complement
  - NAND: complemented AND
  - NOR: complemented OR
  - XOR: two inputs differ
  - XNOR: two inputs agree
  - Enabled buffer: input copied
  - Output bubble: inversion
  - AND shape plus bubble: NAND
- Truth-table construction
  - n inputs → 2^n rows
  - Complements before products before sums
  - Boolean + means OR
- Worked example Boolean evaluation example
  - F=x+y′z; x=0,y=1,z=1
  - y′=0 → F=0
  - Four inputs → 16 rows

Boolean reduction and universal gates

- Boolean identities
  - A+0=A; A·1=A
  - A+A=A; AA=A
  - A+A′=1; AA′=0

- A+AB=A; A(A+B)=A
- De Morgan transformations
- (AB)′=A′+B′
- (A+B)′=A′B′
- Truth-table equivalence check

• Universal gates and Karnaugh maps
- NAND alone; NOR alone
- Tied inputs construct NOT
- Complemented gates construct AND/OR
- Gray-code ordering
- Adjacent power-of-two rectangular groups
- Wraparound and overlap allowed
- Cover every required 1
- Eliminate changing variables
- No diagonal groups or required 0s

• Worked example Reduction and NAND construction
- AB+AB′=A(B+B′)=A
- A NAND A=A′
- NAND then inversion → AND

Combinational and sequential circuits

• Combinational versus sequential
- Present inputs plus propagation delay
- Adders, muxes, demuxes
- Encoders and decoders
- Combinational: no stored logical state
- Sequential: stored state plus inputs
- Registers, counters, flip-flops
- Shift register: sequential criterion

• Powered state retention
- Flip-flop: one bit
- Retention ≠ nonvolatility
- Flip-flops/SRAM lose unpowered state

- Clocked updates: designated clock events
- Combinational logic: no inherent clock

Flip-flops and counter traces

- Latch and flip-flop transitions
  - Active-high NOR SR latch
  - SR=10: set; 01: reset
  - SR=00: hold; 11: forbidden
  - Active-low NAND conventions differ
  - D: active-edge input copy
  - JK=00: hold; 01: reset
  - JK=10: set; 11: toggle
  - T=1: toggle; T=0: hold
- Sequential state tracing
  - Given initial state
  - One clock at a time
  - k-bit counter: 2^k states
  - Modulo 2^k wraparound
  - Extra control may alter sequence
  - Ripple: asynchronous propagation
  - Synchronous: shared clock
- Worked example Counter and T examples
  - 3-bit: 110→111→000
  - Initial T-flop state: 0
  - T inputs 1,0,1 → 1,1,0

Adders, selectors and converters

- Adder equations
  - Half-adder sum=A XOR B
  - Half-adder carry=AB
  - Full-adder includes Cin
  - Sum=A XOR B XOR Cin
  - Carry=AB+Cin(A XOR B)

- Cascading: word-wide carry propagation

• Data selection and encoding
  - 2^k-to-1 mux: k selects
  - Demux: one input→selected output
  - k-to-2^k decoder
  - Decoder enable and active level
  - Ordinary encoder: one active input
  - Priority encoder: multiple-input winner
  - Registers: storage, not combinational converters

• Worked example Adder and mux examples
  - A=B=Cin=1 → binary 11
  - Sum=1; carry=1
  - 8-to-1 mux: 3 selects

Bases and binary codes

• Positional representation
  - Digit value=digit×b^i
  - Binary digits: 0-1
  - Octal digits: 0-7
  - Hex digits: 0-9,A-F
  - F=15
  - Integer: repeated division by 2
  - Remainders read upward
  - Fraction: repeated multiplication by 2
  - Integer parts read downward

• Grouping and encodings
  - Octal: groups of three bits
  - Hex: groups of four bits
  - Group outward from radix point
  - BCD: four bits per decimal digit
  - BCD 12: 0001 0010
  - Binary 12: 1100
  - ASCII: character code, not I/O interface

- Worked example Base-conversion examples
  - 1011_2=8+2+1=11
  - $2F_{16}$=47
  - 101.101_2=5.625
  - 13: remainders 1,0,1,1
  - Reverse remainders → 1101_2

Signed arithmetic and overflow

- Fixed-width integer representations
  - Unsigned: 0…2^n−1
  - Two’s complement: −2^(n−1)…2^(n−1)−1
  - Negation: invert n bits, add one
  - A−B=A+two’s-complement(B)
  - Retain n result bits
  - Signed magnitude: two zeros
  - One’s complement: two zeros
  - Two’s complement: one zero
- Carry versus signed overflow
  - Unsigned carry ≠ signed overflow
  - Same-sign inputs, opposite-sign result
  - Fixed width throughout
- Worked example Four-bit arithmetic examples
  - 7−3: 0111+1101=1 0100
  - Discard carry → 4
  - 0111+0001=1000: signed overflow
  - +8 outside −8…+7

Floating point and arithmetic pipelines

- Binary floating-point representation
  - Sign×significand×2^exponent
  - Binary32 normal: (−1)^s(1.f)2^(e−127)
  - Binary64: 11 exponent bits
  - Binary64 bias: 1023

- Special patterns: zero/subnormal
- Special patterns: infinities/NaNs
- Implicit leading 1: not universal

• Floating addition sequence
- Compare exponents
- Shift smaller significand: alignment
- Add/subtract, normalize, round
- Alignment loses low-order bits
- Overflow: exponent range
- Rounding error: finite precision
- Arithmetic pipeline: overlapping operand pairs
- All steps ≠ one cycle

• Worked example Floating alignment example
- $1.10_2 \times 2^3 + 1.00_2 \times 2^1$
- Align: $(1.10_2 + 0.01_2) \times 2^3$
- Normalize: $1.11_2 \times 2^3 = 14$

Register transfers and memory cycles | Exam focus

• Register transfer semantics
- R2←R1: copy, preserve R1
- P:R2←R1: conditional clocked transfer
- Simultaneous transfers: old values
- Sequential transfers: newly written values
- Common bus: one selected source
- Load signals: selected destinations

• Memory read/write microoperations
- Read: address→MAR/AR
- Assert read; wait for memory
- Capture→MDR/MBR/DR
- Write: address plus data
- Assert write
- ALU: computation
- Control unit: sequencing and enables

- Microoperation: elementary register operation

- Worked example Simultaneous versus sequential example
  - Initially R1=3,R2=7
  - R1←R2,R2←R1 simultaneously → 7,3
  - Same transfers sequentially → 7,7

Address, data and control buses

- Bus signal roles
  - Address: select location
  - Data: carry values
  - Control: read/write and timing
  - Requests and acknowledgements
  - CPU-master addresses: CPU→memory
  - Data: either direction
  - DMA controller: alternate bus master
  - Control: CPU outputs/device inputs
  - Bidirectional: data and control ambiguity
- Addressability versus widths
  - a address bits → 2^a locations
  - Byte capacity=locations×bytes/location
  - Word/data-bus/instruction/address widths may differ
- Worked example Address-space example
  - 32-bit byte addressing: 2^32 bytes
  - 4 GiB; bank wording 4 GB
  - 16-bit data bus ≠ 16-bit addresses

Bit masks and shifts

- Mask operations
  - Set: A OR mask
  - Clear: A AND NOT(mask)
  - Toggle: A XOR mask
  - Retain: A AND mask
  - A XOR A: entire register cleared

- Arbitrary XOR mask ≠ clearing

• Shifts, rotates and shift registers
  - Logical shifts: zero fill
  - Arithmetic right: sign replication
  - Left shift: zero fill, overflow possible
  - Left shift may change sign
  - Rotate: wrap shifted-out bit
  - SISO: serial delay
  - SIPO: serial→parallel word
  - PISO: parallel→serial word
• Worked example Masking and shift examples
  - A=1010; mask=1100
  - Set→1110; clear→0010; toggle→0110
  - Arithmetic-right 1100→1110: −4→−2
  - Logical-right 1100→0110

CPU registers and control words | Exam focus

• CPU register roles
  - PC: next instruction to fetch
  - IR: fetched instruction
  - MAR: current memory-access address
  - MDR: transferred data
  - Accumulator: operands/results
  - SP: stack tracking
  - Status flags: conditions
  - Instruction address: PC/MAR phase ambiguity
  - Registers: explicit execution operands
  - Cache: copied memory blocks
  - Both: temporary storage
• Control-word construction
  - Select ALU sources, destination, operation
  - Machine encoding table required
  - Module control word: 14 bits

- SELA/SELB/SELD: 3 bits each
- Operation field: 5 bits

• Worked example Encoded subtraction example
- R1←R2−R3
- SELA=010; SELB=011; SELD=001
- Subtraction=00101
- 010 011 001 00101
- Source/destination order matters

Instruction cycle and next PC | Exam focus

• Conceptual instruction cycle
- Fetch; decode opcode/addressing fields
- Obtain operands; execute
- Write required results
- Check permitted interrupts
- Example stages: IF, ID, EX, MEM, WB
- Five stages: not architecturally mandatory

• Fetch and next-PC distinctions
- MAR←PC; MDR←M[MAR]; IR←MDR
- PC←PC+instruction_length
- Later branch replaces sequential PC
- Read PC versus calculate branch target
- Control unit: fetch/data-flow coordination
- ALU: arithmetic and logic
- Write-back ≠ another fetch

Instruction fields and operand counts | Exam focus

• Instruction field interpretation
- Opcode: operation
- Operand fields: data/register locations
- Mode field: interpretation
- 4-bit opcode: ≤16 encodings
- Encodings ≠ implemented instructions

- 6-bit reference: 64 encodings
- Machine determines referenced objects

• Explicit address-count formats
- Three-address ADD X,Y,Z: X←Y+Z
- Destination-first convention
- Two-address ADD X,Y: X←X+Y
- Accumulator ADD X: AC←AC+M[X]
- Zero-address ADD: implicit stack operands
- Count explicit operand fields
- INC X ≠ necessarily accumulator instruction

Stacks and operand order

• Stack organization and conventions
- LIFO; PUSH adds; POP removes
- Register stack versus RAM stack
- SP: top or next free
- Address growth: upward or downward
- SP update: before/after access
- Overflow: push full stack
- Underflow: pop empty stack

• Stack arithmetic order
- Second operand popped first
- First operand popped second
- Subtraction: operand order matters
- Stack PUSH/POP: explicit addresses possible

• Worked example Stack expression example
- (A+B)×(C+D)
- PUSH A; PUSH B; ADD
- PUSH C; PUSH D; ADD; MUL
- Downward full stack PUSH: decrement, store
- POP: read, increment

Effective addresses and addressing modes | Exam focus

- Operand addressing rules
  - A: field; R: register; M[x]: contents
  - Immediate: value A
  - Direct: EA=A; operand=M[A]
  - Indirect: EA=M[A]; operand=M[EA]
  - Register: operand R
  - Register-indirect: EA=R
  - Indexed/base displacement: EA=R+A
  - PC-relative: defined PC+signed displacement
  - Auto-increment/decrement: operand-size register update
- Transfer versus addressing mode
  - Register↔memory transfer: nonunique mode
  - Direct, indexed or indirect possible
- Worked example Addressing example
  - A=100; R=20; M[100]=300; M[300]=9
  - Direct operand: 300
  - Indirect operand: 9
  - Indexed EA: 120
  - Immediate 100 ≠ M[100]

Instruction classes and interrupts

- Instruction functions
  - Data transfer: value copying
  - Arithmetic/logic: computed values
  - TEST/CMP: condition information
  - Arithmetic result often discarded
  - Branch: conditional/unconditional target
  - CALL: save return point
  - RETURN: restore return point
  - SKIP: conditional next-instruction bypass
  - Jump-to-next wording ≠ clear skip
- Interrupt and exception origins

- External: device/timer
- Internal synchronous: executing instruction
- Divide errors; illegal opcodes
- Access violations
- Software interrupt: deliberate instruction
- Overflow traps: architecture/instruction dependent
- Overflow flag ≠ automatic interrupt

RISC, CISC and control | Exam focus

- RISC/CISC design tendencies
  - RISC: simple operations, regular encodings
  - Load/store access; plentiful registers
  - Pipeline-friendly control
  - CISC: complex operations, richer addressing
  - Variable instruction lengths
  - Hardwired: direct control signals
  - Microprogrammed: stored control-word sequences
  - Tendencies ≠ universal guarantees
- RISC/CISC interpretation traps
  - One completion/cycle: throughput, not latency
  - RISC instructions: multiple cycles possible
  - CISC: simpler internal operations possible
  - Labels ≠ power/cost/time guarantees
  - Semantic gap: high-level→machine distance
  - Richer instructions: reduced semantic gap
  - Gap reduction: not universal shared objective

x86 register names and classes

- x86 register map
  - 16-bit: AX/BX/CX/DX, BP/SP, SI/DI
  - Conventional special roles
  - AL: AX low byte
  - AH: AX high byte

- BX/CX/DX: analogous byte names
- EAX: 32-bit; RAX: 64-bit
- FLAGS/EFLAGS/RFLAGS: status
- IP/EIP/RIP: instruction position
- CS: code segment; SS: stack segment
- DX: programmer-visible
- IR/MDR: conceptual internal registers

• x86 classification caveats
- SP: general-purpose despite stack role
- EAX/EIP: 80386, before Pentium
- R8-R15: 64-bit extension
- R8-R15 ≠ original Pentium
- Narrow multipurpose category requires definition

x86 legality and implicit operands

• Legacy x86 operand legality
- Intel destination-first syntax
- MOV AL,BP: invalid 8/16-bit mismatch
- INC CX; DEC CX: valid words
- Segment MOV: register/memory forms
- Immediate→segment: no direct form
- Immediate→AX; MOV DS,AX

• Implicit MUL and instruction classes
- Byte MUL: AL×operand→AX
- Word MUL: AX×operand→DX:AX
- AL=8,BH=10; MUL BH → AX=80
- CL irrelevant
- MOVS/LODS: string transfers
- XCHG: operand exchange
- INC: arithmetic
- Different: transfer-versus-arithmetic criterion

Memory hierarchy and timing | Exam focus

- Hierarchy and timing measures
  - Speed/cost order: registers→L1/L2/L3→DRAM
  - Then SSD→HDD→archival tape
  - Faster: smaller, costlier per bit
  - Access time: read/write latency
  - Cycle time: minimum operation interval
  - Recovery may extend cycle time
  - Transfer rate: data/time, not delay
- Locality and cache metrics
  - Temporal: reuse recent data
  - Spatial: access nearby locations
  - Cache: nearby memory-block copies
  - Hit ratio=hits/accesses
  - Miss ratio=1−hit ratio
  - Capacity versus latency versus bandwidth
  - Hit ratio: separate comparison dimension
- Worked example Hit ratio and AMAT example
  - 90/100 hits → 0.9
  - Hit time: 2 ns
  - Additional miss penalty: 50 ns
  - AMAT=2+0.1×50=7 ns

RAM, firmware and chip capacity

- Memory technologies
  - SRAM: powered bistable storage
  - SRAM: fast, costly, no refresh
  - DRAM: charge storage, refresh required
  - DRAM: dense main memory
  - SRAM/DRAM: ordinarily volatile
  - ROM/flash: retain unpowered contents
  - Erasable/programmable ROM variants
  - ROM never modifiable: overbroad

- Firmware: low-level system code
- Commonly nonvolatile storage

• Memory-chip organization
- 2^a×w: words×width
- a address inputs; w data bits
- Parallel chips: increased word width
- Banks: increased word count
- Address decoding versus data selection

• Worked example Chip-combination example
- Eight 1K×1 parallel → 1K×8
- 10 shared address lines
- 8 data bits
- Four banks → 4K×8
- Overall address bits: 12

Cache placement and index bits | Exam focus

• Cache mapping equations
- B: block bytes; L: lines
- Block=floor(address/B)
- Offset=address mod B
- Direct: line=block mod L
- Fully associative: any line
- W-way: S=L/W sets
- Set=block mod S
- Any W ways within set

• Cache address decomposition
- Power-of-two sizes assumed
- Offset bits=log2 B
- Index bits=log2 S
- Remaining address bits: tag
- Hit: matching tag and valid bit
- Same direct index: mutual conflict
- Associativity: fewer placement conflicts

- More tag comparisons

- Worked example Direct-mapped conflict example
  - B=16 bytes; L=4 lines
  - Addresses 0,64 → blocks 0,4
  - Both index 0
  - Alternating accesses: conflict misses

Cache replacement and write policies | Exam focus

- Replacement trace rules
  - FIFO: earliest inserted victim
  - LRU: least recently accessed victim
  - Track hit/miss, set, policy state
  - Hits update LRU order
  - Hits preserve FIFO insertion order
  - Direct mapping: no victim choice
- Write policies and cache classifications
  - Write-through: lower-level update each write
  - Write-back: cache update, dirty bit
  - Dirty eviction: lower-level write
  - Write-allocate: fetch missed block
  - No-write-allocate: write below, no installation
  - L1/L2: hierarchy levels
  - Split: separate instruction/data caches
  - Unified: combined instruction/data cache
  - Internal/external: separate classification
  - Location ≠ split/unified
- Worked example FIFO versus LRU trace
  - Two fully associative lines
  - A,B,A,C: second A hits
  - Remaining accesses miss
  - LRU evicts B
  - FIFO evicts A

Virtual addresses and page faults

- Virtual-memory mechanisms
  - Translated process address spaces
  - Protection
  - Nonresident pages: backing storage
  - MMU: virtual/logical→physical translation
  - Page tables
  - TLB: recent translations
  - Virtual memory ≠ merely disk RAM
- Paging equations and fault distinctions
  - Page=floor(virtual_address/P)
  - Offset=virtual_address mod P
  - Physical_address=frame×P+offset
  - Valid nonresident access: page fault
  - Invalid/forbidden accesses can fault
  - Cache miss ≠ TLB miss
  - TLB miss ≠ page fault
  - Page fault ≠ mandatory disk read
  - Zero-fill; resident shared pages
- Worked example Address-translation example
  - P=1024; virtual_address=2053
  - Page=2; offset=5
  - Page 2→frame 7
  - Physical_address=7×1024+5=7173

Persistent storage and access methods

- Storage media mechanisms
  - HDD: magnetic tracks and sectors
  - Addressed blocks; rotating platters
  - Reading: head over track
  - Seeking: radial head movement
  - HDD bits survive power removal
  - SSD: nonvolatile semiconductors, no mechanical seek

- Optical media: lasers
- Tape: sequential streaming and archives
- Ethernet cable: communication, not disk/RAM

• Access-method taxonomy
- Tape: sequential access
- Disk: direct, positioning delay
- RAM: random, broadly address-independent timing
- Associative: content matching
- Informal disk random-access usage
- Direct/random options require taxonomy
- Optical 100-300 ms: historical example
- Device timings: not universal

I/O interfaces and commands | Exam focus

• Peripheral and interface roles
- Peripheral: input/output/storage beyond execution core
- CPU ≠ peripheral
- Input units: external information encoding
- Output units: processed-result presentation
- Controller: device protocol and timing
- Interface: data/status/control registers
- Buffers: speed mismatch absorption
- Error/status bits: condition reporting

• I/O command categories
- Control: start actions
- Test/status: readiness/errors
- Read: device→system
- Write/output: system→device
- Issue: sending, not command category
- Controller: hardware
- Driver: controlling software

Polling, interrupts and DMA | Exam focus

- I/O transfer mechanisms
  - Polling: repeated status checks
  - CPU moves each item
  - Interrupt-driven: other work until event
  - Handler still consumes CPU time
  - DMA: controller device↔memory block transfer
  - CPU setup first
  - Completion/errors may interrupt CPU
- DMA responsibilities and modes
  - CPU: configuration and ownership
  - Address mapping and completion
  - Burst: bus held for block
  - Cycle stealing: individual transfer opportunities
  - Transparent: idle opportunities
  - Reduced per-word CPU work
  - Competes for memory bandwidth
  - Substantial blocks favor DMA
  - Device address mapping: separate dimension
- Worked example Thousand-word transfer comparison
  - Polling/interrupts: CPU work per word
  - DMA: setup plus completion
  - Actual memory transfers still take time

I/O spaces, priorities and handshaking | Exam focus

- Memory-mapped versus isolated I/O
  - MMIO: device registers in memory space
  - Suitable load/store operations
  - Isolated: separate port space
  - Special IN/OUT instructions
  - Device access widths matter
  - Ordering and side effects
  - Status read ≠ harmless RAM read

- Handshaking and interrupt priority
  - Asynchronous request/ready plus acknowledgement
  - Sender: data then request
  - Receiver: accept then acknowledge
  - Withdraw signals after acceptance
  - Unequal-speed tolerance
  - Single strobe: no feedback
  - Priority: simultaneous interrupt selection
  - Polling, daisy chains, priority hardware
  - Serial: bits over time
  - Parallel: multiple bits/lines
  - Parallel ≠ guaranteed faster

Interfaces by connection role | Exam focus

- Named interface roles
  - USB: serial peripherals, keyboards/storage
  - RS-232: point-to-point serial electrical/interface standard
  - Serial alone: USB/RS-232 ambiguity
  - HDMI: digital video/audio
  - Ethernet: networking, not memory
  - SATA: storage interface
  - Legacy IDE/PATA and SCSI
  - iSCSI: SCSI commands over IP
- Interconnect versus software standards
  - PCI: peripheral expansion interconnect
  - PCI ≠ disk protocol
  - Unlike buses: bridge/controller logic
  - Processor-bus extension needs topology
  - PCI/Multibus: ambiguous without topology
  - POSIX: software interface standard
  - POSIX open/read/write/close
  - ASCII: character encoding
  - Standard disk interface requires era/requirements

Analog interfaces and component identity

- Conversion direction and named components
  - ADC: analog input→digital code
  - DAC: digital code→analog output
  - ADC080x: 8-bit successive-approximation ADC
  - DAC0830: DAC
  - 82C55: programmable parallel I/O
  - 82C55 ≠ ADC
  - Component identity versus conversion direction
- Converter resolution
  - Ideal n-bit voltage span V
  - Approximate code step=V/2^n
  - Endpoint conventions/specifications required
  - Quantization: analog ranges→discrete codes
  - Sample rate ≠ resolution
- Worked example ADC step-size example
  - 8 bits; 5.12 V span
  - 256 codes
  - Step=5.12/256=0.02 V
  - Same span, more bits: smaller steps

Execution time and bottlenecks | Exam focus

- CPU performance equations
  - CPU_time=instruction_count×CPI×clock_period
  - CPU_time=instruction_count×CPI/clock_rate
  - CPI: instruction-class weighted average
  - IPC=1/CPI: consistent whole-program definitions
  - Higher frequency ≠ faster program
  - Increased instruction count/stalls
  - Throughput: completed work/time
  - Latency: individual job delay
- Amdahl’s law
  - f: accelerated execution fraction

- s: local speedup factor
- Speedup=1/((1−f)+f/s)
- Unaffected fraction limits gains
- Same workload comparison
- Include memory and branch penalties
- Clock speed ≠ performance

• Worked example Performance calculation examples
- 10^6 instructions; CPI=2; 1 GHz
- CPU time=2 ms
- f=0.5; s=2
- Speedup=1/(0.5+0.25)=4/3
- Overall speedup ≠ 2

Pipeline overlap and timing | Exam focus

• Ideal pipeline timing
- Different instructions overlap stages
- Primary gain: throughput
- Balanced k-stage single-issue pipeline
- n instructions: k+n−1 cycles
- No stalls assumed
- Nonoverlapped work: nk stage cycles
- Steady-state: one instruction/cycle
- Individual latency: multiple stages

• Pipeline clock and overhead
- Clock: slowest stage plus register overhead
- Short streams: fill/drain cost
- Fetch/decode overlaps another execution
- Multiple issue: several instructions/cycle
- Multiple-issue resources required

• Worked example Five-stage timing example
- k=5; n=10
- Ideal cycles=14
- Nonoverlapped cycles=50

- Speedup=50/14≈3.57
- Short stream: below limiting fivefold gain

Hazards and instruction-level parallelism | Exam focus

- Pipeline hazard classes
  - Structural: shared-resource competition
  - Data: unavailable required result
  - Forwarding: early result supply
  - Some dependencies still stall
  - Control: unresolved branch next PC
  - Prediction: guess direction/target
  - Speculative continuation
  - Misprediction: discard work
- Instruction-level parallelism mechanisms
  - Superscalar: multiple instructions/units per cycle
  - Dependencies constrain issue
  - Out-of-order: ready before stalled older operations
  - Preserve architectural results and exceptions
  - Dynamic allocation: runtime entries/units
  - Wider fetch: improved instruction supply
  - Wider fetch ≠ lower memory latency

Multiprocessing, vectors and Flynn classes | Exam focus

- Flynn stream classification
  - SISD: one instruction, one data
  - SIMD: one instruction, multiple data
  - MIMD: multiple instruction/data streams
  - MISD: multiple instructions, common data
  - MISD uncommon
  - Vector instruction: sequence of elements
  - Physical lanes may batch elements
  - Vector ≠ simultaneous entire sequence
- Multiprocessor organization and scaling

- SMP: shared memory/I/O
- Peer processors under OS
- Distributed memory: separate address spaces
- Interprocessor communication
- More cores: available parallel work
- Serial dependencies and coordination limits
- Core i7: product family
- Core count requires model

Historical generations and processor widths

- Generation conventions and caveats
  - First: vacuum tubes
  - Second: transistors
  - Third: integrated circuits
  - Fourth: microprocessor/VLSI-era PCs
  - Stored programs: first era
  - High-level languages: transition into second
  - Dates cross generation boundaries
  - Magnetic-core memory spans eras
  - Witnessed-use clue: nonunique generation
  - Multiple generations improved performance
- Integration and x86 width history
  - Smaller individual components
  - Increased capacity, density, performance
  - Trends ≠ every-model guarantees
  - 80386: 32-bit x86, 1985
  - Before 80486/Pentium
  - 80286: 16-bit predecessor
  - n-bit CPU ≠ uniform register widths
  - PC/IR/address widths may differ

Underspecified and malformed questions

- Question-scope ambiguities

- Temporary storage: RAM/registers/cache
- Explicit role determines interpretation
- Memory includes ROM/flash
- Power-off disk/memory choice: RAM assumption
- Architecture lifetime requires assumptions
- Power efficiency requires assumptions
- Generational gains: not exclusive definitions

• Malformed bank-item handling

- TEST options displaced into stem
- Reconstructed intent ≠ unique answer
- Gate question: actual symbol required
- Absent drawing: definitions insufficient
- Opcode role explicit: figure unnecessary
- Identify minimal missing phrase
- Repair options or diagram
- Weak key ≠ authority