<u>CMSC 411 – Second Exam – Any 75 mins between 2:30 PM – 9:30 PM</u>

This exam for CMSC 411 consists of five sections, covering **multiple-choice questions**, **fill-in-the-blank questions**, **matching exercises**, **short explanations**, and **problem-solving with calculations and diagrams**. Students should allocate their 75-minute exam time efficiently across the sections to maximize their scores. The exam is open book, but it must be completed independently without collaboration.

Points Distribution:

- Section 1: Multiple-Choice Questions – 10 questions, 1 point each (Total: 10 points)
- Section 2: Fill-in-the-Blank – 5 questions, 2 point each (Total: 10 points)
- Section 3: Matching – 5 questions, 4 points each (Total: 20 points)
- Section 4: Short Explanation – 5 questions, 6 points each (Total: 30 points)
- Section 5: Problem-Solving & Doodling – 3 questions, 10 points each (Total: 30 points)
- Section 6: Extra Credit (10 points)

[10 points] Section 1: Multiple-Choice Questions (10 Questions, 1 point each)

1. **In a basic 5-stage pipelined CPU (IF, ID, EX, MEM, WB), which stage is responsible for calculating the branch target address when using early branch resolution?**
   A. IF (Instruction Fetch)
   B. ID (Instruction Decode/Register Read)
   C. EX (Execute)
   D. MEM (Memory Access)

2. **In a pipelined processor, a hazard where the next instruction needs a result that has not yet been written to the register file is called a:**
   A. Structural hazard
   B. Control hazard
   C. Load-use hazard
   D. Data hazard

3. **Which of the following best describes forwarding (bypassing) in a pipelined CPU?**
   A. Flushing the pipeline when a load is immediately followed by a dependent instruction
   B. Sending the result from one pipeline stage directly to an earlier stage that needs it
   C. Eliminating all stalls by predicting branch outcomes
   D. Using a secondary bus to fetch data from memory more quickly

4. **What is the main reason that separate instruction and data caches (Harvard architecture) are often used in pipelined processors?**
   A. They simplify the ISA by eliminating certain instructions
   B. They remove the structure hazard of having only one memory for both instructions and data
   C. They let the compiler reorder instructions more easily
   D. They reduce the size of the CPU die

5. **Which of the following is a primary difference between SRAM and DRAM?**
   A. SRAM must be refreshed regularly; DRAM does not

B. DRAM is faster and used for CPU caches; SRAM is used for main memory
C. SRAM uses transistors to store bits; DRAM uses a capacitor plus a transistor and must be refreshed
D. DRAM cells require no external power once written

6. **Suppose a computer's overall performance is heavily limited by memory accesses. Which component in the memory hierarchy can "hide" the latency of main memory from the CPU?**
A. System bus
B. Cache memory
C. DMA controller
D. Arithmetic Logic Unit

7. **In a direct-mapped cache, how does the system determine which one cache line (set) a particular memory block can occupy?**
A. By performing a range check from the lowest to highest valid block
B. Using the block address mod the number of cache lines
C. By storing a fully associative tag array and searching all lines in parallel
D. By selecting the line with the least-recently used data

8. **Which best describes the "tag" in a cache?**
A. A pointer to the next instruction to fetch
B. The part of the address used to distinguish which memory block is in a cache line
C. A value that tracks how often the block is used
D. The lower bits of the address that determine the block offset

9. **What is the main advantage of two-level paging in a virtual memory system?**
A. Eliminates the need for TLB entries
B. Allows each page to hold multiple processes' data
C. Reduces the size of page tables for processes with sparse address spaces
D. Ensures every virtual address directly translates to a fixed physical address

10. **In a typical I/O system, which of the following is true of DMA (Direct Memory Access)?**
A. It allows I/O devices to access memory independently of the CPU, reducing CPU overhead
B. It must be manually triggered by the CPU on each transferred byte
C. It is slower than polling because it always involves an interrupt
D. It only works for output (not input) devices

[10 points] Section 2: Fill-in-the-Blank (5 Questions, 2 point each)

1. ________ hazards occur when the CPU must wait to resolve the outcome of a branch before fetching further instructions.

Answer: Control (or *branch*)

2. A direct-mapped cache uses the formula (block address) mod (number of lines) to determine the ________ in which a memory block is stored.

Answer: Index

3. In a virtual memory system, a ________ is a structure that translates virtual addresses into physical addresses.

Answer: Page table

4. A ________ hazard arises when multiple instructions compete for the same hardware resource at the same time.

Answer: Structural

5. ________ prediction uses hardware to record recent branch behavior and predict whether a branch will be taken or not.

Answer: Dynamic branch

[20 points] Section 3: Matching (5 Questions, 4 points each)

1. Question 1

Synchronous Bus	A. Processor repeatedly checks device status in a loop to see if it needs attention
Asynchronous Bus	B. Uses handshake signals (no single clock) for coordinating data transfer
Polling	C. External signal that triggers execution of an I/O or service routine
Interrupt	D. Uses a common clock signal to time data transfers and control

Answer Key:

DBAC

2. Question 2

Direct-Mapped Cache	A. Indicates whether a given cache line is storing valid data
Fully Associative Cache	B. Each memory block can be placed in exactly one cache line (location)
Tag	C. A field stored in the cache line to identify which memory block is held
Valid Bit	D. Any memory block can be placed in **any** cache line

Answer Key:

BDCA

CMSC 411 – Second Exam – Any 75 mins between 2:30 PM – 9:30 PM

3. Question 3

Clock Cycle	A. Discard partially executed instructions (e.g., after misprediction)
Pipeline Hazard	B. Interval in which the pipeline can advance one step
Pipeline Stall	C. CPU must insert wait cycles, delaying the next instruction
Flush	D. A condition that prevents the pipeline from proceeding smoothly

Answer Key:

BDCA

4. Question 4

Write-Through	A. Cache write policy that updates memory at the same time the cache is updated
Write-Back	B. Policy in which modified data is kept only in the cache and written to memory later
Dirty Bits	C. A flag indicating whether a cache block has been modified
Write Allocate	D. On a write miss, the block is first fetched into the cache before updating

Answer Key:

ABCD

5. Question 5

L1 Cache	A. The fraction of all memory accesses that find the requested data in the cache
L2 Cache	B. A small, fast memory closest to the CPU, typically split into I-cache and D-cache
Hit Ratio	C. A larger but slower on-chip or on-board cache to reduce misses from main memory
Miss Penalty	D. The extra time (or cycles) needed to retrieve data from a lower-level memory when a cache miss occurs

Answer Key:

BCAD

CMSC 411 – Second Exam – Any 75 mins between 2:30 PM – 9:30 PM

[30 points] Section 4: Short Explanation (5 Questions, 6 points each)

1. How does a TLB (Translation Lookaside Buffer) speed up virtual memory?

The TLB caches recent virtual-to-physical address translations. When the CPU looks up an address, it can check the TLB instead of going through the full page table in memory, reducing address-translation time.

2. Why does increasing associativity in a cache often reduce conflict misses?

Higher associativity lets a memory block map to more than one location in the cache. This flexibility decreases the likelihood of different blocks colliding in the same cache line, reducing misses due to conflicts.

3. How do exceptions differ from interrupts?

Exceptions are triggered by events within the CPU (e.g., invalid opcode or arithmetic overflow), while interrupts originate from external devices (e.g., I/O). Both cause a jump to special handler code, but for different reasons.

4. Why does pipelining improve CPU performance?

Pipelining allows multiple instructions to overlap in execution, so more instructions complete per unit time. Even though each instruction still goes through all stages, the stages operate in parallel on different instructions.

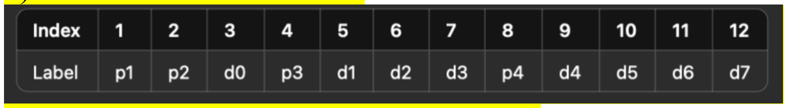
5. Why does a load-use hazard often require at least one pipeline stall, even with forwarding?

When a value is loaded from memory, it only becomes available after the MEM stage completes, so forwarding cannot supply it early enough for the following instruction. As a result, the pipeline must stall for one cycle (or more) before the data is ready.

[30 points] Section 5: Problem-Solving & Doodling (3 Questions, 10 points each)

1. You have an 8-bit data word: 1011 0010. Using even parity, construct the single-error-correcting Hamming code by inserting 4 parity bits. Your task includes
   - Mark which indices are for parity vs. data bits
   - Determine each parity bit (p1, p2, p3, p4) for even parity
   - Provide the final 12-bit Hamming code in order from index 1 to index 12.

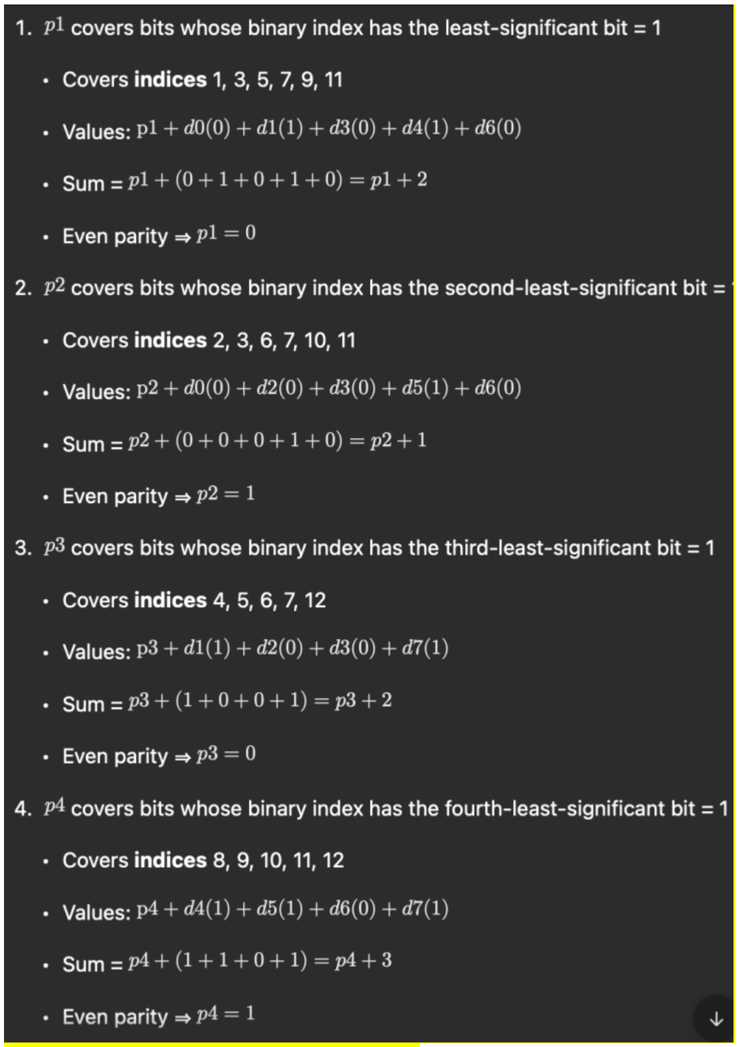
   1) Bit Positions and Labels



Index	1	2	3	4	5	6	7	8	9	10	11	12
Label	p1	p2	d0	p3	d1	d2	d3	p4	d4	d5	d6	d7

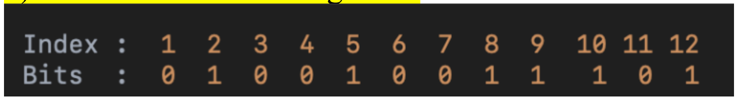
   2) Compute the Parity Bits (Even Parity)

CMSC 411 – Second Exam – Any 75 mins between 2:30 PM – 9:30 PM



1. $p1$ covers bits whose binary index has the least-significant bit = 1
   - Covers **indices** 1, 3, 5, 7, 9, 11
   - Values: $\mathrm{p1} + d0(0) + d1(1) + d3(0) + d4(1) + d6(0)$
   - Sum = $p1 + (0 + 1 + 0 + 1 + 0) = p1 + 2$
   - Even parity ⇒ $p1 = 0$
2. $p2$ covers bits whose binary index has the second-least-significant bit =
   - Covers **indices** 2, 3, 6, 7, 10, 11
   - Values: $\mathrm{p2} + d0(0) + d2(0) + d3(0) + d5(1) + d6(0)$
   - Sum = $p2 + (0 + 0 + 0 + 1 + 0) = p2 + 1$
   - Even parity ⇒ $p2 = 1$
3. $p3$ covers bits whose binary index has the third-least-significant bit = 1
   - Covers **indices** 4, 5, 6, 7, 12
   - Values: $\mathrm{p3} + d1(1) + d2(0) + d3(0) + d7(1)$
   - Sum = $p3 + (1 + 0 + 0 + 1) = p3 + 2$
   - Even parity ⇒ $p3 = 0$
4. $p4$ covers bits whose binary index has the fourth-least-significant bit = 1
   - Covers **indices** 8, 9, 10, 11, 12
   - Values: $\mathrm{p4} + d4(1) + d5(1) + d6(0) + d7(1)$
   - Sum = $p4 + (1 + 1 + 0 + 1) = p4 + 3$
   - Even parity ⇒ $p4 = 1$

3) Final 12-Bit Hamming Code



```
Index :  1  2  3  4  5  6  7  8  9  10 11 12
Bits  :  0  1  0  0  1  0  0  1  1   1  0  1
```

2. You are to design a 256 KB direct-mapped data cache for a system with 32-bit addresses. Each cache block holds 4 words, and each word is 32 bits (4 bytes).
   a) How many bits are used for the byte offset?
   b) How many bits are used for the block offset?
   c) How many bits are used for the set (index) field?
   d) How many bits are used for the tag?
   e) What is the actual physical size of this cache (in bits or bytes), assuming one valid bit per block (no dirty bits)?

(a) How many bits are used for the byte offset?
Since 1 word = 4 bytes, we need $\log_2(4)$ = **2 bits**
(b) How many bits are used for the block offset?
Since each block has 4 words, we need $\log_2(4)$ = **2 bits**
(c) How many bits are used for the set (index) field?
Total cache size = 256 KB = 2^{18} Bytes

CMSC 411 – Second Exam – Any 75 mins between 2:30 PM – 9:30 PM

Block size = 16 Bytes = 2^4 Bytes
Number of blocks = $2^{18} / 2^4 = 2^{14}$
So, index bits = $\log_2(2^{14})$ = **14 bits**

(d) How many bits are used for the tag?
Total address bits = 32
Used:

- Byte offset: 2 bits
- Block offset: 2 bits
- Index: 14 bits

Tag = 32 - (2 + 2 + 14) = **14 bits**

(e) What is the actual physical size of the cache?
Number of rows (blocks): 2^{14}
Data per block: 4 words = 4 × 32 bits = 128 bits
Tag: 14 bits
Valid bit: 1 bit
Total per row = 1 (valid) + 14 (tag) + 128 (data) = **143 bits**

Total size = 143 × 2^{14} = **143 × 16,384 = 2,342,912 bits = 286 KB**

3. You have a 32-bit virtual address space, and each page is 4 KB in size. The system uses a single-level page table where every virtual page number (VPN) is mapped to a physical frame number (PFN). Physical memory is 128 MB in total.
   a) How many bits form the page offset in each virtual address?
   b) How many bits form the virtual page number?
   c) Suppose the virtual page 0x0123 (in hexadecimal) maps to physical frame 0x0345. If the virtual address is 0x0123 ABCD, what is the corresponding physical address? Show how you split the address into page number and offset, and then recombine them with the correct physical frame number. (Full credit)

   a) A 4 KB page = 2^12 bytes, so 12 bits are used as the offset.
   b) Total address size = 32 bits
      Offset = 12 bits
      VPN = 32 − 12 = 20 bits
   c) The virtual page portion is the upper 20 bits (0x0123), and the offset is the lower 12 bits.
      a. 0x0123 ABCD in hexadecimal can be split as:
      b. VPN = 0x0123
      c. Offset = 0xBCD (the lower 12 bits)

      Given that virtual page 0x0123 maps to physical frame 0x0345, we replace the VPN in the virtual address with PFN = 0x0345 and keep the same offset.
      d. New PFN = 0x0345
      e. Same offset = 0xBCD

      Combine them:
      f. Physical address = PFN << 12 + offset = 0x0345 × 2^12 + 0xBCD
      g. In hex notation, that becomes 0x0345 BCD (often written as 0x345BCD).

      Answer: The resulting physical address is 0x0345 BCD

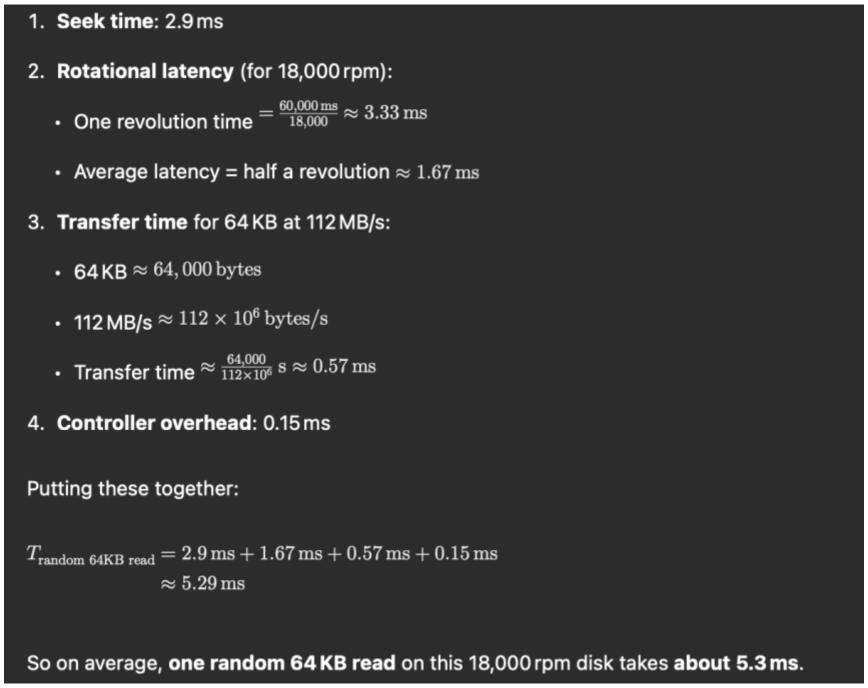
CMSC 411 – Second Exam – Any 75 mins between 2:30 PM – 9:30 PM

[10 points] Section 6: Extra Credit

A 18,000 rpm disk has:

- Average seek time = 2.9 ms
- 0.15 ms controller overhead
- Transfer rate = 112 MB/s

How long does one **random** 64 KB read take on average (in milliseconds)?



1. **Seek time**: 2.9 ms
2. **Rotational latency** (for 18,000 rpm):
   - One revolution time $= \frac{60{,}000\,\text{ms}}{18{,}000} \approx 3.33\,\text{ms}$
   - Average latency = half a revolution $\approx 1.67\,\text{ms}$
3. **Transfer time** for 64 KB at 112 MB/s:
   - 64 KB $\approx 64{,}000\,\text{bytes}$
   - 112 MB/s $\approx 112 \times 10^6\,\text{bytes/s}$
   - Transfer time $\approx \frac{64{,}000}{112\times10^6}\,\text{s} \approx 0.57\,\text{ms}$
4. **Controller overhead**: 0.15 ms

Putting these together:

$$T_{\text{random 64KB read}} = 2.9\,\text{ms} + 1.67\,\text{ms} + 0.57\,\text{ms} + 0.15\,\text{ms}$$
$$\approx 5.29\,\text{ms}$$

So on average, **one random 64 KB read** on this 18,000 rpm disk takes **about 5.3 ms**.