SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
CONTINUOUS ASSESSMENT TEST - II

WINTER SEMESTER 2025-2026

Answer Key

Course Code and Course Name: BCSE205L -Computer Architecture and Organization

1. A) Consider two different implementations of the same instruction set architecture. The instructions can be divided into four classes according to their CPI (class A, B, C and D). P1 with a clock rate of 2.5 GHz and CPIs of 1, 2, 3, and 3, and P2 with a clock rate of 3 GHz and CPIs of 2, 2, 2, and 2. Given a program with a dynamic instruction count of 1000 instructions divided into classes as follows:

10% class A, 20% class B, 50% class C, and 20% class D.

(i) Find the clock cycles required in both cases. **(4)**
(ii) Which implementation is faster and how much? **(3)**

Given

- **Processor P1**
  - Clock rate = 2.5 GHz CPI (A, B, C, D) = 1, 2, 3, 3
- **Processor P2**
  - Clock rate = 3 GHz CPI (A, B, C, D) = 2, 2, 2, 2
- **Program Instruction Count = 1000**

Instruction mix:

Class	Percentage	Instructions
A	10%	100
B	20%	200
C	50%	500
D	20%	200

(i) Find Clock Cycles Required

P1: Cycles $= (100 \times 1) + (200 \times 2) + (500 \times 3) + (200 \times 3)$
$= 100 + 400 + 1500 + 600$
$= 2600$ cycles

P2: Cycles $= 100(2) + 200(2) + 500(2) + 200(2)$
$= 200 + 400 + 1000 + 400$
$= 2000$ cycles

(ii) Which Implementation is Faster and By How Much?

$$\text{Execution Time} = \frac{\text{Clock Cycles}}{\text{Clock Rate}}$$

P1 Execution Time= $\frac{2600}{2.5 \times 10^9} = 1.04 \times 10^{-6}$ sec

P2 Execution Time=$\frac{2000}{3\times10^9}$== 0.67×10^{-6} sec

Speedup of P2 over P1 = Speedup $= \frac{T_{P1}}{T_{P2}}$

$$= \frac{1.04}{0.67} \approx 1.55$$

P2 is 1.55 times faster than P1.

B). Assume that for a given program 70% of the executed instructions are arithmetic, 10% are load/store, and 20% are branch. Given this instruction mix and the assumption that an arithmetic instruction requires 2 cycles, a load/store instruction takes 6 cycles, and a branch instruction takes 3 cycles, find the average CPI. **(3)**

Formula for Average CPI

$$\text{Average CPI} = \sum(\text{Instruction Percentage} \times \text{CPI of that instruction})$$

Step 1: Convert Percentages to Decimal

Arithmetic → 70% = 0.7 Load/Store → 10% = 0.1 Branch → 20% = 0.2

Step 2: Multiply Each with Its CPI

Arithmetic=$0.7 \times 2 = 1.4$
Load/Store=$0.1 \times 6 = 0.6$
Branch=$0.2 \times 3 = 0.6$

Step 3: Average CPI $= 1.4 + 0.6 + 0.6 = 2.6$

2. A computer system employs RAM chips of 512 MB each and ROM chips of 256 MB each. The system requires 1 GB of RAM, 512 MB of ROM, and two interface units of 512MB each. Design the memory chip layout for the above requirements. Also determine the number of RAM and ROM chips required and explain the address decoding scheme.

I. How many RAM and ROM chips are needed? **(2)**
II. Draw a memory-address map for the system. **(2)**
III. Give the address range in hexadecimal for RAM, ROM, and interface. **(3)**
IV. Show the chip layout for the above design **(3)**

Memory Chip Design Solution

Given

- RAM chip size = **512 MB**
- ROM chip size = **256 MB**
- Required RAM = **1 GB**
- Required ROM = **512 MB**
- Interface units = **2 × 512 MB**

Conversions

- $512\,MB = 2^{29}$bytes
- $256\,MB = 2^{28}$bytes
- $1\,GB = 2^{30}$bytes

System addressing assumed: **32-bit address bus**

I. How many RAM and ROM chips are needed?

Number of Chips Required

RAM $1MB = 1024KB$

$$1024KB/512KB = 2$$

RAM chips required = 2

ROM

$$512KB/128KB = 4$$

ROM chips required = 4

Interface Units

Each interface = 512 KB

$$512KB/512KB = 1$$

Two interfaces: Interface chips = 2

II. Draw a memory-address map for the system.

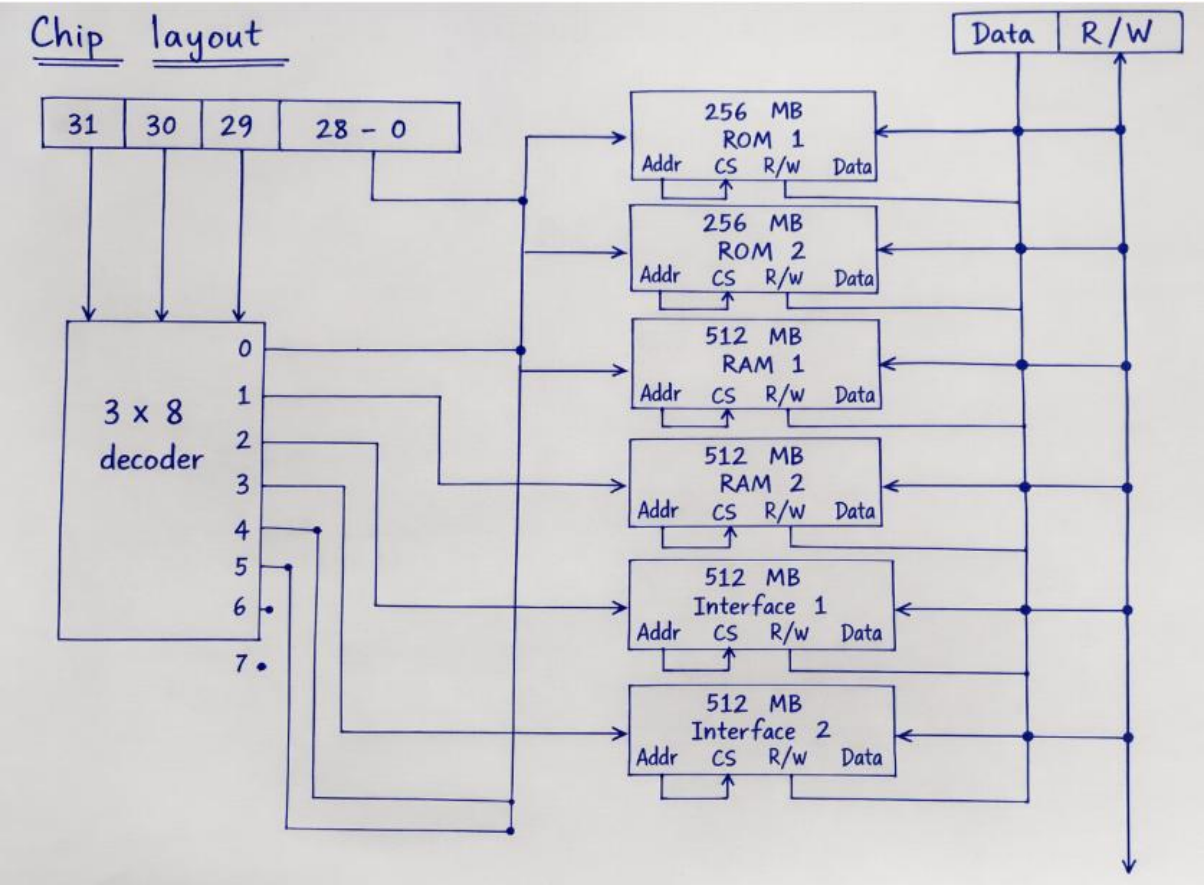
S.No	Memory	Chip Size (MB)	Required Size (MB)	Chips Needed (p)	Width Factor (q)	p × q	Address Bits (x)	y	z	Total Address Bits
1	RAM	512	1024 (1GB)	2	1	2	29	1	2	32
2	ROM	256	512	2	1	2	28	1	2	31
3	Interface	512	1024	2	1	2	29	1	2	32

III. Give the address range in hexadecimal for RAM, ROM, and interface.

Component	From	To	A31	A30	A29	A28–A0
RAM1	00000000	1FFFFFFF	0	0	0	x
RAM2	20000000	3FFFFFFF	0	0	1	x
ROM1	40000000	4FFFFFFF	0	1	0	x
ROM2	50000000	5FFFFFFF	0	1	1	x
Interface1	60000000	7FFFFFFF	1	0	0	x
Interface2	80000000	9FFFFFFF	1	0	1	x

Replace **x with 0 for From address** and **x with 1 for To address**.

Iv. Show the chip layout for the above design



3. A computer system has a 4-way set associative cache with a capacity of 256 bytes. Each cache block contains one word (4 bytes).
The sequence of byte addresses referenced by the processor is:
8, 64, 96, 128, 64, 96, 256, 192, 24

Assume that the cache is initially empty and the replacement policy is LRU (Least Recently Used).

i. Determine the block address and set number for each reference. **(5)**

ii. Construct a cache simulation table for the given memory references, indicating hit or miss for each reference. Also show the final cache contents and the total number of hits and misses. **(5)**

Given

- Cache size = 256 bytes
- Block size = 1 word = 4 bytes

$$\text{Cache blocks} = \frac{256}{4} = 64$$

4-way set associative:

$$\text{Number of sets} = \frac{64}{4} = 16$$

Set formula:

$$Set = Block\ Address \bmod 16$$

Block address:

$$Block = \frac{\text{Byte Address}}{4}$$

I.Block and Set Table:

Byte Address	Block Address	Set Number
8	2	2
64	16	0
96	24	8
128	32	0
64	16	0
96	24	8
256	64	0
192	48	0
24	6	6

II.Cache Simulation Table (with Hit/Miss)

byte #	block #	set #		set0	set0	set0	set0	set2	set6	set8
				block0	block1	block2	block3	block0	block0	block0
8	2	2	miss					**m[8]**		
64	16	0	miss	**m[64]**				m[8]		
96	24	8	miss	m[64]				m[8]		**m[96]**
128	32	0	miss	m[64]	**m[128]**			m[8]		m[96]
64	16	0	hit	**m[64]**	m[128]			m[8]		m[96]
96	24	8	hit	m[64]	m[128]			m[8]		**m[96]**
256	64	0	miss	m[64]	m[128]	**m[256]**		m[8]		**m[96]**
192	48	0	miss	m[64]	m[128]	m[256]	m[192]	m[8]		**m[96]**
24	6	6	miss	m[64]	m[128]	m[256]	m[192]	m[8]	**m[24]**	m[96]

III. Total Hits and Misses:

Type	Count
Hits	2
Misses	7

4. A) The following memory addresses are given:
000101, 000110, 000111, 001000
A memory system has 4 memory modules, each with 16 locations. The processor generates 6-bit addresses.
Divide the 6-bit address into module bits and location bits for High-Order and Low-Order Memory Interleaving, and determine the module and location for the given addresses while showing how consecutive addresses are distributed among the modules. **(3+3)**

Given

- **Memory modules = 4**
- **Locations per module = 16**

- **Address size = 6 bits**

$$\textbf{4 modules} = 2^2 \Rightarrow \textbf{2 bits for module}$$
$$\textbf{16 locations} = 2^4 \Rightarrow \textbf{4 bits for location}$$

So the 6-bit address is divided as:

Bits	Meaning
2 bits	Module number
4 bits	Location inside module

Address Division:

Interleaving Type	Address Format	Module Bits	Location Bits
High-Order Interleaving	| Module | Location |	First 2 bits (MSB)	Last 4 bits
Low-Order Interleaving	| Location | Module |	Last 2 bits (LSB)	First 4 bits

Determine Module and Location:

High-Order Interleaving- All consecutive addresses go to the same module.

Address	Module	Location
000101	00 → Module 0	0101 → 5
000110	00 → Module 0	0110 → 6
000111	00 → Module 0	0111 → 7
001000	00 → Module 0	1000 → 8

Low-Order Interleaving- Consecutive addresses go to different modules.

Address	Location	Module
000101	0001 → 1	01 → Module 1
000110	0001 → 1	10 → Module 2
000111	0001 → 1	11 → Module 3
001000	0010 → 2	00 → Module 0

Low-order interleaving gives better performance because multiple memory modules can be accessed simultaneously.

B) A system uses Interrupt-driven I/O to transfer data from an input device. The device generates one interrupt for every 8 bytes transferred. If the device transfers 256 bytes, determine:

i. The total number of interrupts generated. **(2)**

ii. Explain how interrupt-driven I/O improves CPU utilization compared to programmed I/O. **(2)**

i. Number of Interrupts Generated

$$\text{Interrupts} = \frac{\text{Total Data}}{\text{Bytes per Interrupt}}$$

$$\text{Interrupts} = \frac{256}{8}$$

$$\text{Interrupts} = 32$$

Total interrupts generated = 32

ii. Interrupt-Driven I/O, the CPU does not continuously check the device status. Instead:

1. The CPU issues an I/O command to the device.
2. The CPU continues executing other instructions.
3. When the device is ready, it sends an interrupt signal.
4. The CPU temporarily stops its current task.
5. The Interrupt Service Routine (ISR) transfers the data.
6. The CPU then returns to the previous task.

5.A) A system uses DMA to transfer a 32 KB block of data from an I/O device to main memory at a rate of 40 MB/s. The processor runs at 400 MHz and spends 400 clock cycles to initiate the DMA transfer and 800 clock cycles to complete it after the transfer finishes. Determine the percentage of processor time consumed during the DMA transfer. **(5)**

Step 1: Find Total Transfer Time

Transfer rate = 40 MB/s = 40×2^{20} bytes/sec
Data size = 32 KB = 32×2^{10} bytes

$$\text{Total Time} = \frac{\text{Total Data}}{\text{Transfer Rate}}$$

$$= \frac{32 \times 2^{10}}{40 \times 2^{20}}$$

$$= \frac{32}{40} \times 2^{-10}$$

$$= 7.81 \times 10^{-4} \text{ sec} = 0.781 \text{ ms}$$

Step 2: Find Processor Time

Processor speed = 400 MHz = 400×10^6 cycles/sec

Cycles for DMA = 400 + 800 = 1200 cycles

$$\text{Processor Time} = \frac{1200}{400 \times 10^6}$$

$$= 3 \times 10^{-6} \text{ sec} = 0.003 \text{ ms}$$

Step 3: Find % Processor Time

$$\%\text{Processor Time} = \frac{0.003}{0.781} \times 100$$
$$= 0.38\%$$

5.B) A high-speed printer is connected to a computer system. The printer is slower than the CPU, causing delays in data transfer. Explain how buffering can be used to improve data transfer between the CPU and the printer. Illustrate the concept with a simple diagram. **(5)**

Buffering is used to **temporarily store data in a memory area called a buffer** during data transfer between devices with different speeds.

In this case:

- The **CPU is faster**
- The **printer is slower**

Without buffering, the **CPU would have to wait** until the printer finishes printing each character or data block.

Working of Buffering

1. The **CPU sends data to the buffer memory**.
2. The **buffer temporarily stores the data**.
3. The **printer reads data from the buffer** at a slower rate.
4. While the printer prints, the **CPU continues processing other tasks**.