



VIT

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

REG.NO.:

SLOT: A2+TA2

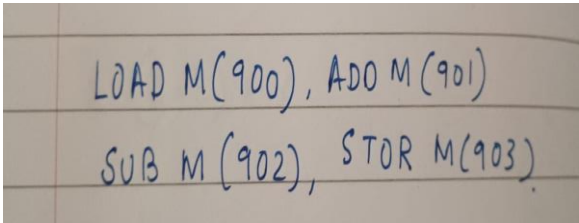
**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
CONTINUOUS ASSESSMENT TEST - I
WINTER SEMESTER 2025-2026**

Programme Name & Branch : B.Tech CSE
Course Code and Course Name : BCSE205L & Computer Architecture and Organization
Faculty Name(s) : Common to all
Class Number(s) : Common to all
Date of Examination : 27.01.2026
Exam Duration : 90 minutes **Maximum Marks: 50**

General instruction(s):

- Answer All Questions
- M - Max mark; CO – Course Outcome; BL – Blooms Taxonomy Level (1 – Remember, 2 – Understand, 3 – Apply, 4 – Analyse, 5 – Evaluate, 6 – Create)
- Course Outcomes (Type the CO statements covered in this question paper. Use the CO number as per the syllabus copy)

CO1- Differentiate Von Neumann, Harvard, and CISC and RISC architectures. Analyze the performance of machine with different capabilities. Recognize different instruction formats and addressing modes. Validate efficient algorithm for fixed point and floating point arithmetic operations.

Q. No	Question	Module	Marks	CO	BL
1.	<p>Consider the following expression using the IAS computer Instruction set and interpret the flow of the IAS computer architecture (Register flow of operations) with neat diagram.</p> $K = (M + N) - O$ <p>Assume the memory locations 900, 901, and 902 for P, Q, and R respectively.</p>  <p>1. Instruction Fetch Cycle</p> <p>PC → MAR</p> <p>Memory → MBR</p> <p>MBR → IR / IBR</p> <p>PC ← PC + 1</p>	1	7	1	BL 2



VIT[®]

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

REG.NO.:

SLOT: A2+TA2

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
CONTINUOUS ASSESSMENT TEST - I
WINTER SEMESTER 2025-2026

<p>2. Execute LOAD M(900)</p> <p>MAR ← 900</p> <p>MBR ← M(900)</p> <p>AC ← MBR</p> <p>3. Execute ADD M(901)</p> <p>MAR ← 901</p> <p>MBR ← M(901)</p> <p>AC ← AC + MBR</p> <p>4. Execute SUB M(902)</p> <p>MAR ← 902</p> <p>MBR ← M(902)</p> <p>AC ← AC – MBR</p> <p>5. Execute STOR M(903)</p> <p>MAR ← 903</p> <p>MBR ← AC</p> <p>M(903) ← MBR</p> <p>Register Function</p> <p>PC Holds next instruction address</p> <p>MAR Holds memory address</p> <p>MBR Holds instruction/data from memory</p> <p>IR Holds current instruction</p> <p>IBR Holds next instruction</p> <p>AC Holds intermediate and final result</p>				
--	--	--	--	--



SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
CONTINUOUS ASSESSMENT TEST - I
WINTER SEMESTER 2025-2026

<p>1b)</p>	<p>Compare and contrast Von Neumann and Harvard computer architectures.</p> <p>Ans:</p> <table border="1"> <thead> <tr> <th>Feature</th> <th>Von Neumann Architecture</th> <th>Harvard Architecture</th> </tr> </thead> <tbody> <tr> <td>Memory</td> <td>Single memory for both instructions and data</td> <td>Separate memory for instructions and data</td> </tr> <tr> <td>Buses</td> <td>Single bus for instructions and data</td> <td>Separate buses for instructions and data</td> </tr> <tr> <td>Instruction & Data Access</td> <td>Cannot access both simultaneously</td> <td>Can access both simultaneously</td> </tr> <tr> <td>Performance</td> <td>Slower due to Von Neumann bottleneck</td> <td>Faster due to parallel access</td> </tr> <tr> <td>Hardware Complexity</td> <td>Simple design</td> <td>More complex design</td> </tr> </tbody> </table>	Feature	Von Neumann Architecture	Harvard Architecture	Memory	Single memory for both instructions and data	Separate memory for instructions and data	Buses	Single bus for instructions and data	Separate buses for instructions and data	Instruction & Data Access	Cannot access both simultaneously	Can access both simultaneously	Performance	Slower due to Von Neumann bottleneck	Faster due to parallel access	Hardware Complexity	Simple design	More complex design	<p>1</p>	<p>3</p>	
Feature	Von Neumann Architecture	Harvard Architecture																				
Memory	Single memory for both instructions and data	Separate memory for instructions and data																				
Buses	Single bus for instructions and data	Separate buses for instructions and data																				
Instruction & Data Access	Cannot access both simultaneously	Can access both simultaneously																				
Performance	Slower due to Von Neumann bottleneck	Faster due to parallel access																				
Hardware Complexity	Simple design	More complex design																				
<p>2.</p>	<p>a) Perform 21/11 using Restoring division algorithm and explain the steps clearly and draw the flowchart. Give flowchart for the restoring division with combination of positive and negative numbers. (10)</p>	<p>2</p>	<p>10</p>	<p>1 BL 5</p>																		



**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
CONTINUOUS ASSESSMENT TEST - I
WINTER SEMESTER 2025-2026**

	<p>Step 1: Represent the numbers in binary</p> <ul style="list-style-type: none"> • Dividend $Q = 21_{10} = 10101_2$ (5 bits) • Divisor $M = 11_{10} = 01011_2$ (5 bits) <p>We assume the Accumulator (A) is initialized to 0 and has enough bits to hold intermediate results.</p> <hr/> <p>Restoring Division Algorithm (Steps)</p> <p>Registers:</p> <ul style="list-style-type: none"> • A = Accumulator (initialized to 0) • Q = Dividend register • M = Divisor register • n = Number of bits in Q <hr/> <p>Algorithm Overview:</p> <ol style="list-style-type: none"> 1. Initialize $A = 0$, $Q = \text{Dividend}$ 2. Repeat n times: <ul style="list-style-type: none"> • Shift A and Q left by 1 bit • Subtract divisor M from $A \rightarrow A = A - M$ • If $A < 0$: <ul style="list-style-type: none"> • Restore $A = A + M$ • Set least significant bit of Q = 0 • Else: <ul style="list-style-type: none"> • Set least significant bit of Q = 1 				
--	---	--	--	--	--



SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
CONTINUOUS ASSESSMENT TEST - I
WINTER SEMESTER 2025-2026

Restoring Division

Divident (Q) = 21 \Rightarrow 10101 \Rightarrow (s) 01010
 Divisor (B) = 11 \Rightarrow 001011 (s) 110109 (-2) 01011

Pass	A	Q	operation
1	000000 $\begin{array}{r} 000000 \\ 110101 \\ \hline 110110 \\ \rightarrow A < 0 \\ Q_0 = 0 \\ A = A + B \\ 110110 \\ 001011 \\ \hline 000001 \end{array}$	10101 0101 0	Shift left by 1-bit $A = A - B$ $A < 0$ $Q_0 = 0$ $A = A + B$
2	000001 $\begin{array}{r} 000001 \\ 110101 \\ \hline 110110 \\ A < 0 \\ Q_0 = 0 \\ 110110 \\ 001011 \\ \hline 000001 \end{array}$	01010 1010 0	Shift left by 1-bit $A = A - B$ $A < 0$ $Q_0 = 0$ $A = A + B$
3	000010 $\begin{array}{r} 000010 \\ 000101 \\ 110101 \\ \hline 110110 \\ A < 0 \end{array}$	10100 0100 0	Shift left by 1-bit $A = A - B$ $A < 0$ $Q_0 = 0$ $A = A + B$

Pass 1 completed

Pass 2 completed

Start

Shift left by 1-bit

$A = A - B$

Yes $A < 0$ No

Yes $Q_0 = 0$ No $Q_0 = 1$

Restoring

Count $\neq 0$



**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
CONTINUOUS ASSESSMENT TEST - I
WINTER SEMESTER 2025-2026**

	<table border="1"> <thead> <tr> <th>Pass</th> <th>A</th> <th>Q</th> <th>operation</th> </tr> </thead> <tbody> <tr> <td></td> <td> $\begin{array}{r} 111010 \\ 001011 \\ \times 000101 \\ \hline \end{array}$ </td> <td>Pass 3 completed</td> <td></td> </tr> <tr> <td>4</td> <td> $\begin{array}{r} 000101 \\ \times 001010 \\ 110101 \\ \hline 111111 \\ A < 0, Q_0 = 0 \\ 111111 \\ 001011 \\ \hline \times 001010 \end{array}$ </td> <td> $\begin{array}{r} 01000 \\ 1000 \boxed{0} \end{array}$ </td> <td> Shift left by 1-bit $A = A - B$ $A < 0, Q_0 = 0$ $A = A + B$ </td> </tr> <tr> <td>5</td> <td> $\begin{array}{r} 001010 \\ 1110101 \\ \times 010101 \\ \hline \times 001010 \end{array}$ </td> <td> $\begin{array}{r} 1000 \boxed{0} \\ 0000 \boxed{1} \end{array}$ </td> <td> Shift left by 1-bit $A = A - B$ $A > 0$, restore A $Q_0 = 1$ </td> </tr> <tr> <td></td> <td>Remainder</td> <td>Quotient</td> <td></td> </tr> <tr> <td></td> <td> Verification step:- $\begin{array}{r} 2^2 \quad 2^1 \quad 2^0 \\ 001010 \\ 8+2 \Rightarrow 10 \end{array}$ </td> <td> $\begin{array}{r} 00001^0 \\ \Rightarrow 1 \end{array}$ </td> <td> $\begin{array}{r} 1 \rightarrow \text{Quotient} \\ 11 \overline{) 21} \\ \underline{11} \\ 10 \checkmark \rightarrow \text{Remainder} \end{array}$ </td> </tr> </tbody> </table>	Pass	A	Q	operation		$\begin{array}{r} 111010 \\ 001011 \\ \times 000101 \\ \hline \end{array}$	Pass 3 completed		4	$\begin{array}{r} 000101 \\ \times 001010 \\ 110101 \\ \hline 111111 \\ A < 0, Q_0 = 0 \\ 111111 \\ 001011 \\ \hline \times 001010 \end{array}$	$\begin{array}{r} 01000 \\ 1000 \boxed{0} \end{array}$	Shift left by 1-bit $A = A - B$ $A < 0, Q_0 = 0$ $A = A + B$	5	$\begin{array}{r} 001010 \\ 1110101 \\ \times 010101 \\ \hline \times 001010 \end{array}$	$\begin{array}{r} 1000 \boxed{0} \\ 0000 \boxed{1} \end{array}$	Shift left by 1-bit $A = A - B$ $A > 0$, restore A $Q_0 = 1$		Remainder	Quotient			Verification step:- $\begin{array}{r} 2^2 \quad 2^1 \quad 2^0 \\ 001010 \\ 8+2 \Rightarrow 10 \end{array}$	$\begin{array}{r} 00001^0 \\ \Rightarrow 1 \end{array}$	$\begin{array}{r} 1 \rightarrow \text{Quotient} \\ 11 \overline{) 21} \\ \underline{11} \\ 10 \checkmark \rightarrow \text{Remainder} \end{array}$				
Pass	A	Q	operation																										
	$\begin{array}{r} 111010 \\ 001011 \\ \times 000101 \\ \hline \end{array}$	Pass 3 completed																											
4	$\begin{array}{r} 000101 \\ \times 001010 \\ 110101 \\ \hline 111111 \\ A < 0, Q_0 = 0 \\ 111111 \\ 001011 \\ \hline \times 001010 \end{array}$	$\begin{array}{r} 01000 \\ 1000 \boxed{0} \end{array}$	Shift left by 1-bit $A = A - B$ $A < 0, Q_0 = 0$ $A = A + B$																										
5	$\begin{array}{r} 001010 \\ 1110101 \\ \times 010101 \\ \hline \times 001010 \end{array}$	$\begin{array}{r} 1000 \boxed{0} \\ 0000 \boxed{1} \end{array}$	Shift left by 1-bit $A = A - B$ $A > 0$, restore A $Q_0 = 1$																										
	Remainder	Quotient																											
	Verification step:- $\begin{array}{r} 2^2 \quad 2^1 \quad 2^0 \\ 001010 \\ 8+2 \Rightarrow 10 \end{array}$	$\begin{array}{r} 00001^0 \\ \Rightarrow 1 \end{array}$	$\begin{array}{r} 1 \rightarrow \text{Quotient} \\ 11 \overline{) 21} \\ \underline{11} \\ 10 \checkmark \rightarrow \text{Remainder} \end{array}$																										
3.	a) Using Booth's algorithm, multiply (+12) × (-5) with neat flowchart based explanation.	2	7	1	BL 5																								



SCHOOL OF COMPUTER SCIENCE AND ENGINEERING CONTINUOUS ASSESSMENT TEST - I WINTER SEMESTER 2025-2026

Step 1: Choose number of bits

We need enough bits to represent ± 12 and ± 5 .

Using 5-bit 2's complement:

- $+12 = 01100$
- $-5 = 2$'s complement of 00101
 - invert: 11010
 - add 1: 11011

Booths' Algorithm

12×-5

$12 \Rightarrow 01100$	$5 \Rightarrow 00101$	<u>Rules</u> 00 - No arithmetic op 01 - add multiplier to left half of product 10 - Subtract multiplier from left half of product 11 - No arithmetic op
$5 \Rightarrow 10011$	$5 \Rightarrow 11010$	
$2's \Rightarrow \frac{\quad}{1}$	$2's \Rightarrow \frac{\quad}{1}$	
$(-5) \Rightarrow 11011$	$(-5) \Rightarrow 11011$	

Pass	left part	right part	operation
0	00000 (+2) 10100	11011 0	10 - Subtract multiplier from left half of prod
1	10100	01101 1	11 - No arithmetic op
2	1010	01101 1	11 - No arithmetic op
3	1010	01101 1	11 - No arithmetic op
4	1010	01101 1	11 - No arithmetic op
5	1010	01101 1	11 - No arithmetic op
6	1010	01101 1	11 - No arithmetic op
7	1010	01101 1	11 - No arithmetic op
8	1010	01101 1	11 - No arithmetic op
9	1010	01101 1	11 - No arithmetic op
10	1010	01101 1	11 - No arithmetic op
11	1010	01101 1	11 - No arithmetic op
12	1010	01101 1	11 - No arithmetic op
13	1010	01101 1	11 - No arithmetic op
14	1010	01101 1	11 - No arithmetic op
15	1010	01101 1	11 - No arithmetic op
16	1010	01101 1	11 - No arithmetic op
17	1010	01101 1	11 - No arithmetic op
18	1010	01101 1	11 - No arithmetic op
19	1010	01101 1	11 - No arithmetic op
20	1010	01101 1	11 - No arithmetic op
21	1010	01101 1	11 - No arithmetic op
22	1010	01101 1	11 - No arithmetic op
23	1010	01101 1	11 - No arithmetic op
24	1010	01101 1	11 - No arithmetic op
25	1010	01101 1	11 - No arithmetic op
26	1010	01101 1	11 - No arithmetic op
27	1010	01101 1	11 - No arithmetic op
28	1010	01101 1	11 - No arithmetic op
29	1010	01101 1	11 - No arithmetic op
30	1010	01101 1	11 - No arithmetic op
31	1010	01101 1	11 - No arithmetic op
32	1010	01101 1	11 - No arithmetic op
33	1010	01101 1	11 - No arithmetic op
34	1010	01101 1	11 - No arithmetic op
35	1010	01101 1	11 - No arithmetic op
36	1010	01101 1	11 - No arithmetic op
37	1010	01101 1	11 - No arithmetic op
38	1010	01101 1	11 - No arithmetic op
39	1010	01101 1	11 - No arithmetic op
40	1010	01101 1	11 - No arithmetic op
41	1010	01101 1	11 - No arithmetic op
42	1010	01101 1	11 - No arithmetic op
43	1010	01101 1	11 - No arithmetic op
44	1010	01101 1	11 - No arithmetic op
45	1010	01101 1	11 - No arithmetic op
46	1010	01101 1	11 - No arithmetic op
47	1010	01101 1	11 - No arithmetic op
48	1010	01101 1	11 - No arithmetic op
49	1010	01101 1	11 - No arithmetic op
50	1010	01101 1	11 - No arithmetic op
51	1010	01101 1	11 - No arithmetic op
52	1010	01101 1	11 - No arithmetic op
53	1010	01101 1	11 - No arithmetic op
54	1010	01101 1	11 - No arithmetic op
55	1010	01101 1	11 - No arithmetic op
56	1010	01101 1	11 - No arithmetic op
57	1010	01101 1	11 - No arithmetic op
58	1010	01101 1	11 - No arithmetic op
59	1010	01101 1	11 - No arithmetic op
60	1010	01101 1	11 - No arithmetic op
61	1010	01101 1	11 - No arithmetic op
62	1010	01101 1	11 - No arithmetic op
63	1010	01101 1	11 - No arithmetic op
64	1010	01101 1	11 - No arithmetic op
65	1010	01101 1	11 - No arithmetic op
66	1010	01101 1	11 - No arithmetic op
67	1010	01101 1	11 - No arithmetic op
68	1010	01101 1	11 - No arithmetic op
69	1010	01101 1	11 - No arithmetic op
70	1010	01101 1	11 - No arithmetic op
71	1010	01101 1	11 - No arithmetic op
72	1010	01101 1	11 - No arithmetic op
73	1010	01101 1	11 - No arithmetic op
74	1010	01101 1	11 - No arithmetic op
75	1010	01101 1	11 - No arithmetic op
76	1010	01101 1	11 - No arithmetic op
77	1010	01101 1	11 - No arithmetic op
78	1010	01101 1	11 - No arithmetic op
79	1010	01101 1	11 - No arithmetic op
80	1010	01101 1	11 - No arithmetic op
81	1010	01101 1	11 - No arithmetic op
82	1010	01101 1	11 - No arithmetic op
83	1010	01101 1	11 - No arithmetic op
84	1010	01101 1	11 - No arithmetic op
85	1010	01101 1	11 - No arithmetic op
86	1010	01101 1	11 - No arithmetic op
87	1010	01101 1	11 - No arithmetic op
88	1010	01101 1	11 - No arithmetic op
89	1010	01101 1	11 - No arithmetic op
90	1010	01101 1	11 - No arithmetic op
91	1010	01101 1	11 - No arithmetic op
92	1010	01101 1	11 - No arithmetic op
93	1010	01101 1	11 - No arithmetic op
94	1010	01101 1	11 - No arithmetic op
95	1010	01101 1	11 - No arithmetic op
96	1010	01101 1	11 - No arithmetic op
97	1010	01101 1	11 - No arithmetic op
98	1010	01101 1	11 - No arithmetic op
99	1010	01101 1	11 - No arithmetic op
100	1010	01101 1	11 - No arithmetic op

Arithmetic Shift Right

Verification Step

... 00100X



VIT[®]

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

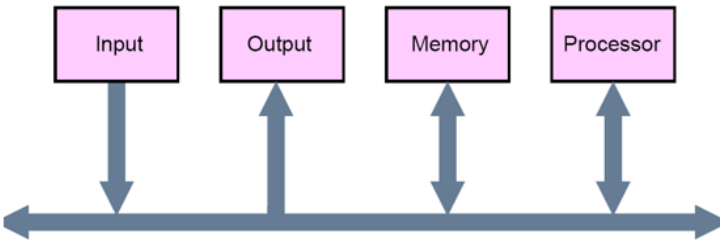
REG.NO.:

SLOT: A2+TA2

**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
CONTINUOUS ASSESSMENT TEST - I
WINTER SEMESTER 2025-2026**

	$(-1)^S \times 1.M \times 2^{(E-1023)}$																															
4.	<p>a) Consider the following instructions, classify each into the appropriate instruction set category and justify your answer:</p> <ul style="list-style-type: none"> i) MOV R1, R2 ii) ADD R3, R4 iii) JMP 200 iv) IN PORT1 v) NOP vi) AND R5, R6 vii) BCD to Binary Conversion viii) LOAD R1, M[150] <p>a) Classification of Instructions</p> <table border="1"> <thead> <tr> <th>Instruction</th> <th>Category</th> <th>Justification</th> </tr> </thead> <tbody> <tr> <td>i) MOV R1, R2</td> <td>Data Transfer Instruction</td> <td>Moves data from register R2 to register R1, transferring data between CPU registers.</td> </tr> <tr> <td>ii) ADD R3, R4</td> <td>Arithmetic Instruction</td> <td>Performs addition of contents of R3 and R4, a basic arithmetic operation.</td> </tr> <tr> <td>iii) JMP 200</td> <td>Transfer of Control</td> <td>Changes program flow by jumping to the instruction at address 200.</td> </tr> <tr> <td>iv) IN PORT1</td> <td>I/O Instruction</td> <td>Reads data from input device or port PORT1 into CPU/register.</td> </tr> <tr> <td>v) NOP</td> <td>System Control Instruction</td> <td>No operation; used to control timing or synchronization without changing system state.</td> </tr> <tr> <td>vi) AND R5, R6</td> <td>Logical Instruction</td> <td>Performs bitwise AND between registers R5 and R6, used for logical operations.</td> </tr> <tr> <td>vii) BCD to Binary Conversion</td> <td>Conversion Instruction</td> <td>Converts data from BCD format to binary, changing data representation format.</td> </tr> <tr> <td>LOAD R1, M[150]</td> <td>Data Transfer Instruction</td> <td>Loads data from memory location 150 into register R1, transferring data into the CPU.</td> </tr> </tbody> </table>	Instruction	Category	Justification	i) MOV R1, R2	Data Transfer Instruction	Moves data from register R2 to register R1, transferring data between CPU registers.	ii) ADD R3, R4	Arithmetic Instruction	Performs addition of contents of R3 and R4, a basic arithmetic operation.	iii) JMP 200	Transfer of Control	Changes program flow by jumping to the instruction at address 200.	iv) IN PORT1	I/O Instruction	Reads data from input device or port PORT1 into CPU/register.	v) NOP	System Control Instruction	No operation; used to control timing or synchronization without changing system state.	vi) AND R5, R6	Logical Instruction	Performs bitwise AND between registers R5 and R6, used for logical operations.	vii) BCD to Binary Conversion	Conversion Instruction	Converts data from BCD format to binary, changing data representation format.	LOAD R1, M[150]	Data Transfer Instruction	Loads data from memory location 150 into register R1, transferring data into the CPU.	3		1	BL 4
Instruction	Category	Justification																														
i) MOV R1, R2	Data Transfer Instruction	Moves data from register R2 to register R1, transferring data between CPU registers.																														
ii) ADD R3, R4	Arithmetic Instruction	Performs addition of contents of R3 and R4, a basic arithmetic operation.																														
iii) JMP 200	Transfer of Control	Changes program flow by jumping to the instruction at address 200.																														
iv) IN PORT1	I/O Instruction	Reads data from input device or port PORT1 into CPU/register.																														
v) NOP	System Control Instruction	No operation; used to control timing or synchronization without changing system state.																														
vi) AND R5, R6	Logical Instruction	Performs bitwise AND between registers R5 and R6, used for logical operations.																														
vii) BCD to Binary Conversion	Conversion Instruction	Converts data from BCD format to binary, changing data representation format.																														
LOAD R1, M[150]	Data Transfer Instruction	Loads data from memory location 150 into register R1, transferring data into the CPU.																														
	b) With neat diagram, explain how CPU, Memory, and I/O devices are interconnected in a computer system.	3	7	3																												

**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
CONTINUOUS ASSESSMENT TEST - I
WINTER SEMESTER 2025-2026**

	<p>The system bus consists of three types of buses:</p> <ol style="list-style-type: none"> Data Bus <ul style="list-style-type: none"> Transfers actual data between CPU, memory, and I/O devices Bidirectional Width determines the amount of data transferred at a time (e.g., 32-bit, 64-bit) Address Bus <ul style="list-style-type: none"> Carries the address of the memory location or I/O device Unidirectional (from CPU to others) Determines the maximum addressable memory Control Bus <ul style="list-style-type: none"> Carries control and timing signals Coordinates operations among components Examples: Read, Write, Interrupt, Clock signals <hr/> <p>Working of the Interconnection</p> <ol style="list-style-type: none"> The CPU places an address on the address bus. Control signals (Read/Write) are sent via the control bus. Data is transferred through the data bus. Memory or I/O device responds based on the address and control signals. I/O devices communicate with the CPU either by programmed I/O, interrupts, or DMA. <p>This answer also considered:</p> <ul style="list-style-type: none"> A group of lines that serves a connecting path for several devices is called a bus <ul style="list-style-type: none"> In addition to the lines that carry the data, the bus must have lines for address and control purposes The simplest way to interconnect functional units is to use a single bus, as shown below  <ul style="list-style-type: none"> The devices connected to a bus vary widely in their speed of operation <ul style="list-style-type: none"> Some devices are relatively slow, such as printer and keyboard Some devices are considerably fast, such as optical disks 				
--	---	--	--	--	--



SCHOOL OF COMPUTER SCIENCE AND ENGINEERING CONTINUOUS ASSESSMENT TEST - I WINTER SEMESTER 2025-2026

	<ul style="list-style-type: none"> Memory and processor units operate are the fastest parts of a computer Efficient transfer mechanism thus is needed to cope with this problem <ul style="list-style-type: none"> A common approach is to include buffer registers with the devices to hold the information during transfers Another approach is to use two-bus structure and an additional transfer mechanism <ul style="list-style-type: none"> A high-performance bus, a low-performance, and a bridge for transferring the data between the two buses. ARMA Bus belongs to this structure 				
5.	<p>Convert the following expression into assembly instructions for 0,1,2,3 address formats and calculate the total memory traffic:</p> $P=(Q \times R) - S$ <p>Use the same assumptions:</p> <ul style="list-style-type: none"> Address: 2 bytes Data: 2 bytes Opcode: 8 bits Word length: 1 byte <p>Step 1: 0-Address Machine (Stack Machine)</p> <p>0-address machines are stack-based. Operands are pushed onto the stack, operations pop operands and push the result.</p> <p>Assembly Code (0-address):</p> <pre> css PUSH Q ; Push Q onto stack PUSH R ; Push R onto stack MUL ; Pop Q,R -> multiply -> push result PUSH S ; Push S onto stack SUB ; Pop (Q*R), S -> subtract -> push result POP P ; Store result into P </pre>	3	10	1	BL 4



SCHOOL OF COMPUTER SCIENCE AND ENGINEERING CONTINUOUS ASSESSMENT TEST - I WINTER SEMESTER 2025-2026

Step 2: 1-Address Machine (Accumulator Machine)

1-address machine uses an accumulator (AC). Operations mostly involve AC and memory.

Assembly Code (1-address):

```

powershell

LOAD Q      ; AC <- Q
MUL R       ; AC <- AC * R
SUB S       ; AC <- AC - S
STORE P     ; P <- AC

```

Step 3: 2-Address Machine

2-address machine: Typically ADD R1,R2 → R1 = R1 + R2

Assembly Code (2-address):

```

powershell

MOV AC, Q   ; AC <- Q
MUL AC, R   ; AC <- AC * R
SUB AC, S   ; AC <- AC - S
MOV P, AC   ; P <- AC

```

Memory Traffic Calculation:

Step 4: 3-Address Machine

3-address machine: Each instruction can have two source operands and one destination operand.

Assembly Code (3-address):

```

pgsql

MUL Q, R, TEMP ; TEMP = Q * R
SUB TEMP, S, P  ; P = TEMP - S

```

Memory Traffic Calculation:

Instruction	Memory Access
MUL Q,R,TEMP	2 reads (Q,R) + 1 write (TEMP) = 3×2 bytes = 6 bytes?
SUB TEMP,S,P	2 reads (TEMP,S) + 1 write (P) = 3×2 bytes = 6 bytes? ↓



VIT

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

REG.NO.:

SLOT: A2+TA2

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING CONTINUOUS ASSESSMENT TEST - I WINTER SEMESTER 2025-2026

	1	2	1	
	OP	OPI	OP	
	M/Fe/ch.		M/Execul	M/T
PUSH Q	3		2	5
PUSH R	3		2	5
MUL	1		0	1
PUSH S	3		2	5
SUB	1		0	1
POP P	3		2	5
				<u>17</u>
	M/ Fetch		M/E	M/Traff
LOAD Q	3		2	5
MUL R	3		2	5
SUB S	3		2	5
STORE P	3		2	5
				<u>20</u>



VIT

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

REG.NO.:

SLOT: A2+TA2

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING CONTINUOUS ASSESSMENT TEST - I WINTER SEMESTER 2025-2026
