



JULY 2021: END SEMESTER ASSESSMENT (ESA) B TECH 4th SEMESTER

UE16/17CS253 /UE19CS252 – MICROPROCESSOR AND COMPUTER ARCHITECTURE

Time: 3 Hrs

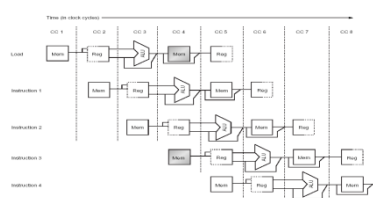
**Answer All Questions
Scheme and Solution**

Max Marks: 100

Note: ARM7TDMI – ISA stands for ARM Instruction Set Architecture,
ALP – Assembly Language Program

1	a)	<p>Write the ARM7TDMI ALP code snippet for the following code written in C-language.</p> <p>Let R1, R2, R3 contain the starting addresses of arrays X, Y and Z respectively. Use Register R5 for variable i.</p> <p>While (i ++ <= 10)</p> <pre>{ If (X [i] == Y [i]) Z [i] = X [i] * Y [i]; }</pre> <p>{Hint: X[i] is written as [R1,R5] , Y[i] as [R2,R5] and Z[i] as [R3,R5] }</p> <p>PROGRAM:</p> <pre>.DATA X:.WORD 5,4,6,7,8,1,2,3,4,9 // Array X Y:.WORD 6,4,6,7,4,1,3,3,5,9 // Array Y Z:.WORD 0 // Array Z .TEXT LDR R1, =X LDR R2, =Y LDR R3, =Z MOV R4, #0 MOV R5, #0 L2: CMP R4, #10 BEQ L1 LDR R6, [R1, R5] LDR R7, [R2, R5] ADD R5, R5, #4 ADD R4, R4, #1 CMP R6, R7 BNE L2 MUL R8, R6, R7 STR R8, [R3, #4]! B L2 L1: SWI 0X011</pre> <p>Initialization 1 mark, increment variables 2 marks, looping construct : 2 marks, storage: 1 mark.</p>	06
----------	-----------	--	-----------

b)	<p>What are the different ways of parameter passing techniques used in ARM7TDMI – ISA? Also, Consider and observe the following code snippet written using ARM7TDMI – ISA. What parameter passing technique is used in the following code snippet? Explain.</p> <pre>.text LDR R4, =A MOV R1, #52 MOV R2, #25 STMFD R13!, { R1, R2} BL LINK STR R0, [R4] SWI 0x11 LINK: LDMFD R13!, { R6, R5} ADD R0, R6, R5 MOV PC, LR .data A: .WORD 0</pre> <p>Answer: Parameter passing techniques: (2 marks).</p> <ol style="list-style-type: none"> 1. Register parameter 2. Stack parameter <p>The above code uses Stack parameter passing technique. The parameters 52 and 25 are pushed on the stack using STM instruction. A function call is made. Once transfer to the function, parameters are popped on to registers R6 and R5 respectively using LDM instruction as the variant is FD – FULL DESCENT.</p>	02 + 02 = 04
c)	<p>Is FIQ exception or interrupt accepted by the ARM processor, while the data abort exception is currently being serviced ? Explain.</p> <p>Answer: In ARM7TDMI, FIQ interrupt has higher priority than data abort hence it is accepted. Since data abort interrupt is being serviced at this point in time, as it is has lower priority compared to the device that has raised FIQ interrupt request later. The former interrupt service will be suspended and the processor will be allocated to device with FIQ request that is the later. The above operation is possible if the program has enabled FIQ interrupt. Otherwise not.</p>	01 + 02 + 01 = 04

d)	<p>i. Consider the following coding used in ARM7TDMI encoding.</p> <p>Conditional codes : 1110 - AL, GE - 1010 , LE – 1101</p> <p>Operation codes : RSB - 0011, AND – 0000, ORR - 0001</p> <p>What ARM instructions does these encoding represent?</p> <table><tr><th>Inst</th><th>Condition</th><th>F (format)</th><th>I (immediate)</th><th>OPCODE</th><th>S (Set cond Code)</th><th>Rn</th><th>Rd</th><th>Operand2</th></tr><tr><td>i.</td><td>1010</td><td>0</td><td>0</td><td>3</td><td>0</td><td>0</td><td>1</td><td>2</td></tr><tr><td colspan="9">Ans: RSBGE R1, R0, R2</td></tr><tr><td>ii.</td><td>1101</td><td>0</td><td>1</td><td>0</td><td>1</td><td>2</td><td>3</td><td>#12</td></tr><tr><td colspan="9">Ans: ANDLES R3, R2, #12</td></tr></table> <p>ii. Write the instruction to multiply a number X by 17. Use only ADD / SUB / RSB instructions to multiply a number by 17. {Assume the number is in the register R9, that is R9 = R9 x 17}.</p> <p>Answer:</p> <p>ADD R9, R9, R9, LSL #4</p>	Inst	Condition	F (format)	I (immediate)	OPCODE	S (Set cond Code)	Rn	Rd	Operand2	i.	1010	0	0	3	0	0	1	2	Ans: RSBGE R1, R0, R2									ii.	1101	0	1	0	1	2	3	#12	Ans: ANDLES R3, R2, #12									02 + 02 + 02 = 06
Inst	Condition	F (format)	I (immediate)	OPCODE	S (Set cond Code)	Rn	Rd	Operand2																																							
i.	1010	0	0	3	0	0	1	2																																							
Ans: RSBGE R1, R0, R2																																															
ii.	1101	0	1	0	1	2	3	#12																																							
Ans: ANDLES R3, R2, #12																																															
2	<p>a) Consider a Von Neumann architecture. A machine has only one memory unit. But under certain conditions, the pipeline might want to perform two operations (like read instruction / write operation) simultaneously during the same clock cycle. Is it possible? What type of hazard is introduced? Explain. How can this hazard be resolved? Explain with neat diagrams.</p> <p>Answer:</p> <p>No it is not possible.</p> <p>If certain combination of instructions can't be accommodated because of resource conflicts, the machine is said to have a structural hazard. In the above case, memory access are seen during instruction fetch and memory write or read during 4th clock cycle time. This is a conflict as memory is accessed at the same time by different instructions at the same time.</p> <p>Introduction of a bubble or memory stall cycle, it solves the problem.</p> <p>Further, even the stall cycle can be resolved by implementing Harvard architecture with split cache, Instruction & data caches.</p> <div></div> <p>Von Neumann architecture- Same memory accessed by two operations simultaneously i.e., at CC1 & CC4.</p>	01 + 01 + 02 = 04																																													

2	<div>a)</div> <div> <p>Time (in clock cycles) →</p> <p>CC 1 CC 2 CC 3 CC 4 CC 5 CC 6 CC 7 CC 8</p> <p>Load: Mem → Reg → ALU → Mem → Reg</p> <p>Instruction 1: Mem → Reg → ALU → Mem → Reg (Stall for 4 cycles)</p> <p>Instruction 2: Mem → Reg → ALU → Mem → Reg</p> <p>Stall: 4 cycles (CC 4-7)</p> <p>Instruction 3: Mem → Reg → ALU → Mem → Reg</p> <p>Memory bottleneck in load-store architecture</p> <p>Alternative way of improving performance: Harvard architecture with split cache, Instruction & data caches.</p> <p>Processor instruction cycle, Instruction Cache, Data Cache, Shared Memory, program bus, Data bus.</p> </div> <div> <p>Introduction of a bubble or memory stall cycle, It solves the problem.</p> <p>Further, even the stall cycle can be resolved by implementing Harvard architecture with split cache, Instruction & data caches.</p> <p>[Additional may be written no marks are awarded].</p> </div>	<div>01</div> <div>+</div> <div>01</div> <div>+</div> <div>02</div> <div>=</div> <div>04</div>
b)	<div> <p>Consider the following sequence of instructions written in ARM7TDMI - ISA. Assume A, B and C are memory locations storing data.</p> <pre> LDR R1, [A] LDR R2, [B] LDR R3, =C ADD R0, R1, R2 SUB R4, R0, R2 AND R5, R4, R1 STR R5, [R3]. </pre> <p>How many clock cycles does it take to execute and stall cycles are introduced in the following cases?</p> <p>Case i. No data forwarding is supported.</p> <p>Case ii. Data or operand forwarding is supported.</p> <p>Note: Write operation is performed during the first half of the clock cycle time while read operation is done during the Second half of the clock cycle time. Show the working for the same.</p> <p>Answer:</p> <p>Case 1:</p> <p># of clock cycles required to complete the instructions are: 19.</p> <p># of stalls introduced: 08</p> </div>	<div>04</div>

b)

Answer:

Inst	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
I1	IF	ID	EX	MEM	WB	-	-	-	-	-	-	-	-	-	-	-	-	-	-
I2	-	IF	ID	EX	MEM	WB	-	-	-	-	-	-	-	-	-	-	-	-	-
I3	-	-	IF	ID	EX	MEM	WB	-	-	-	-	-	-	-	-	-	-	-	-
I4	-	-	-	IF	ID	Stall	Stall	EX	MEM	WB									-
I5					IF	Stall	stall	ID	Stall	Stall	EX	MEM	WB						-
I6								IF	Stall	Stall	ID	Stall	Stall	EX	MEM	WB			-
I7											IF	Stall	Stall	ID	Stall	Stall	EX	MEM	WB

Case 2 : # of clock cycles required to complete the instructions are: 11.

of stalls introduced: 00.

Inst	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
I1	IF	ID	EX	MEM	WB	-	-	-	-	-	-	-	-	-	-	-	-	-	-
I2	-	IF	ID	EX	MEM	WB	-	-	-	-	-	-	-	-	-	-	-	-	-
I3	-	-	IF	ID	EX	MEM	WB	-	-	-	-	-	-	-	-	-	-	-	-

Inst	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
I4	-	-	-	IF	ID	EX	MEM	WB											-
I5					IF	ID	EX	MEM	WB										-
I6						IF	ID	EX	MEM	WB									-
I7							IF	ID	EX	MEM	WB								

02
+
02
=
04

03

+

03

$$=$$

06

03

+

03

$$=$$

06

3	a)	<p>A computer system with a word length of 32 bits has a 16 MB byte-addressable main memory and a 64 KB, 4-way set associative cache memory with a block size of 256 bytes. Consider the following four addresses represented in hexadecimal notation. A1 = 0x42C8A4, A2 = 0x546888, A3 = 0x6A289C, A4 = 0x5E4880. Determine which of these addresses map on to the same set in the cache memory. Show the computations. Note: Determine the number of bits required to represent word and set numbers.</p> <p>Answer:</p> <p>16MB address space has 24 address lines / bits. Matches with the given address.</p> <p>Cache Memory Capacity = 64KB which has 4 way set associativity. So, Word can be decoded with least significant 6 bits as it has 2^6 words. 64 words of 32 bits each = $64 \times 4 = 256$ bytes / block. Since it is 4 way set associative memory, it has 256 blocks. Hence requires, 2^8 sets. Requires 8 bits.</p> <table border="0"> <thead> <tr> <th></th><th>TAG bits</th><th>SET bits</th><th>WORD address</th></tr> </thead> <tbody> <tr> <td>$(42C8A4)_{16} =$</td><td>(0100 0010 11</td><td>0 0 1 0 0 0 1 0</td><td>1 0 0 1 0 0)₂</td></tr> <tr> <td>$(546888)_{16} =$</td><td>(0101 0100 01</td><td>1 0 1 0 0 0 1 0</td><td>0 0 1 0 0 0)₂</td></tr> <tr> <td>$(6A289C)_{16} =$</td><td>(0110 1010 00</td><td>1 0 1 0 0 0 1 0</td><td>0 1 1 1 0 0)₂</td></tr> <tr> <td>$(5E4880)_{16} =$</td><td>(0101 1110 01</td><td>0 0 1 0 0 0 1 0</td><td>0 0 1 0 0 0)₂</td></tr> </tbody> </table> <p>The set bits represent the mapping with respect to the cache memory block. Address 546888 and 6A289C map on to the same set in the cache.</p>		TAG bits	SET bits	WORD address	$(42C8A4)_{16} =$	(0100 0010 11	0 0 1 0 0 0 1 0	1 0 0 1 0 0) ₂	$(546888)_{16} =$	(0101 0100 01	1 0 1 0 0 0 1 0	0 0 1 0 0 0) ₂	$(6A289C)_{16} =$	(0110 1010 00	1 0 1 0 0 0 1 0	0 1 1 1 0 0) ₂	$(5E4880)_{16} =$	(0101 1110 01	0 0 1 0 0 0 1 0	0 0 1 0 0 0) ₂	02 + 04 + 01 = 07
	TAG bits	SET bits	WORD address																				
$(42C8A4)_{16} =$	(0100 0010 11	0 0 1 0 0 0 1 0	1 0 0 1 0 0) ₂																				
$(546888)_{16} =$	(0101 0100 01	1 0 1 0 0 0 1 0	0 0 1 0 0 0) ₂																				
$(6A289C)_{16} =$	(0110 1010 00	1 0 1 0 0 0 1 0	0 1 1 1 0 0) ₂																				
$(5E4880)_{16} =$	(0101 1110 01	0 0 1 0 0 0 1 0	0 0 1 0 0 0) ₂																				
	b)	<p>What are the categories of misses? Explain how to avoid address translation in cache indexing to reduce hit time?</p> <p>Answer:</p> <p>Three Categories of Miss:</p> <ol style="list-style-type: none"> Compulsory Miss. Capacity Miss. Conflict Miss. <p>Explanation of the 6th optimization technique.</p> <div style="text-align: right; background-color: #0056b3; color: white; padding: 5px; margin-top: 20px;">20 bits Page Number</div> <div style="text-align: right; background-color: #0056b3; color: white; padding: 5px; margin-top: 20px;">4 bits Frame Number</div> <p>Reduce Hit time is done by</p> <p>Not to wait for VA to PA translation. Extract Index information from Virtual Address Extract Tag Information from the Physical Address. 6th Optimization facilitate Virtual Indexing and Physically Tagging (VIPT)</p>	03 + 02 = 05																				

c)	<p>If a direct mapped cache has a hit rate of 90%, a hit time of 5ns, and a miss penalty of 100ns, what is the AMAT? If an L2 cache is added with a hit time of 25ns and a hit rate of 50%, what is the new AMAT, if the penalty for an L2 miss is 200ns? Also compute local miss rates of L1 and L2 caches and the global miss rate.</p> <p>Answer:</p> $\text{AMAT} = \text{Hit Time L1} + \text{Miss Rate L1} * [\text{Hit Time L2} + \text{Miss Rate L2} * \text{Miss Penalty L2}]$ $= 5\text{ns} + 10\% * [25\text{ns} + 50\% * 200\text{ns}]$ $= 17.5\text{ns}$ <p>Local Miss Rate L1 = 100% - 90% = 10% = 0.10.</p> <p>Local Miss Rate L2 = 100% - 50% = 50% = 0.50.</p> <p>Global Miss Rate = LMRL1 * LMRL2</p> $= 0.1 * 0.5$ $= 0.05. = 5\%.$	01 + 01 + 01 + 01 = 04
d)	<p>Assume a processor where the cycles per instruction (CPI) is 1.0 when all memory accesses hit in the cache. The only data accesses are loads and stores, and these total to 60% of the instructions. If the miss penalty is 50 clock cycles and the miss rate is 5%, how much faster would the computer be if all instructions were cache hits?</p> <p>Answer:</p> <p>First compute the performance for the computer that always hits:</p> $\text{CPU execution time} = (\text{CPU clock cycles} + \text{Memory stall cycles}) \times \text{Clock cycle}$ $= (\text{IC} \times \text{CPI} + 0) \times \text{Clock cycle}$ $= \text{IC} \times 1.0 \times \text{Clock cycle}$ <p>Now for the computer with the real cache, first we compute memory stall cycles:</p> $\text{Memory stall cycles} = \text{IC} \times \frac{\text{Memory accesses}}{\text{Instruction}} \times \text{Miss rate} \times \text{Miss penalty}$ $= \text{IC} \times (1 + 0.6) \times 0.05 \times 50$ $= \text{IC} \times 4.0$ <p>where the middle term (1 + 0.6) represents one instruction access and 0.6 data accesses per instruction. The total performance is thus</p> $\text{CPU execution time cache} = (\text{IC} \times 1.0 + \text{IC} \times 4.0) \times \text{Clock cycle}$ $= 4.0 \times \text{IC} \times \text{Clock cycle}$ <p>The performance ratio is the inverse of the execution times:</p> $\frac{\text{CPU execution time}_{\text{cache}}}{\text{CPU execution time}} = \frac{4.0 \times \text{IC} \times \text{Clock cycle}}{1.0 \times \text{IC} \times \text{Clock cycle}} = 4.0$ <p>Hence, The computer with no cache misses is 4.0 times faster.</p>	01 + 02 + 01 = 04

4	a)	<p>Assume the memory system takes 100 clock cycles of overhead and then delivers 32bytes every 4 clock cycles. That is, it can supply 32 bytes in 104 clock cycles, 64 bytes in 108 clock cycles, and so on... Compute the average memory access time for a block size of 128bytes and cache size of 64KB that has a miss rate of 1.02%. Assume hit time is 1cc.</p> <p>Answer:</p> <p>Average access time = Hit time + Miss rate x Miss penalty</p> <p>The access time for a 32- bytes is 104 cc. A block of 128 bytes takes 116cc.</p> <p>Average access time = $1 + (1.02\% \times 116)$</p> <p style="padding-left: 40px;">$= 1 + 1.1832$</p> <p style="padding-left: 40px;">$= 2.1832$ clock cycles.</p>	02 + 02 = 04
	b)	<p>Consider the following code sequence. Assume Direct mapped cache.</p> <p style="padding-left: 40px;">STR R3, 256 (R0)</p> <p style="padding-left: 40px;">LDR R1, 2048 (R0)</p> <p style="padding-left: 40px;">LDR R2, 256 (R0)</p> <p>Write-through cache maps 256 and 2048 to the same block. A four words write buffer is not checked on a read miss. Will the value in register R2 always be equal to the value in register R3 or R2? Discuss.</p> <p>Answer:</p> <ul style="list-style-type: none"> • This is a read-after-write data hazard in memory. • The data in R3 are placed into the write buffer after the STR. • The following LDR instruction uses the same cache index and is therefore a miss. • The second LDR instruction, tries to put the value in location 256 into the register R2. • This also results in a miss. • If the write buffer hasn't completed writing to location 256 in memory, • The read of location 256 will put the old, wrong value into the cache block and then into R2. • Without proper precautions, R3 would not be equal to R2! 	01 + 02 + 02 = 05

c)

What is an interrupt vector table? Write the interrupt table of ARM7TDMI processor. How simultaneous multiple interrupts are handled? Explain.

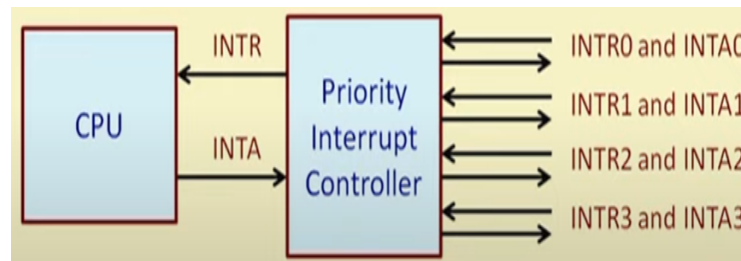
Answer:

Interrupt Vector table is as shown in the diagram. IVT is the first 1MK memory of RAM. IVT basically stores the address of the routines with respect to their corresponding locations in the memory.

Whenever an interrupt occurs, depending on the type of interrupt occurred, it refers to the corresponding location in the IVT and branches to that location to service the interrupt occurred.

Simultaneous Multiple Interrupts are handled using PIC.

- Common solution is to use a **Priority Interrupt Controller**.
 - Interrupt controller interacts with CPU on one side and multiple devices on the other side.
 - For simultaneous interrupt requests, interrupt priority is defined.
 - Interrupt controller is responsible for sending the interrupt vector to the CPU.



d)

How does user enable and disable FIQ and IRQ interrupts? Explain with appropriate code using ARM7TDMI- ISA.

Answer:

Enabling an interrupt.

cpsr value	IRQ	FIQ
Pre	<i>nzcvqjIfT_SVC</i>	<i>nzcvqjIfT_SVC</i>
Code	enable_irq MRS r1, cpsr BIC r1, r1, #0x80 MSR cpsr_c, r1	enable_fiq MRS r1, cpsr BIC r1, r1, #0x40 MSR cpsr_c, r1
Post	<i>nzcvqjIfT_SVC</i>	<i>nzcvqjIfT_SVC</i>

Disabling an interrupt.

cpsr	IRQ	FIQ
Pre	<i>nzcvqjIfT_SVC</i>	<i>nzcvqjIfT_SVC</i>
Code	disable_irq MRS r1, cpsr ORR r1, r1, #0x80 MSR cpsr_c, r1	disable_fiq MRS r1, cpsr ORR r1, r1, #0x40 MSR cpsr_c, r1
Post	<i>nzcvqjIfT_SVC</i>	<i>nzcvqjIfT_SVC</i>

02
+
03
=
05

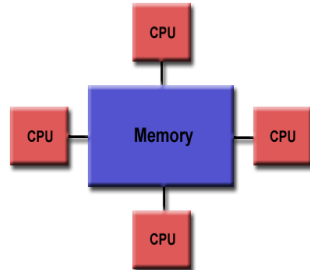
03
+
03
=
06

5 a)

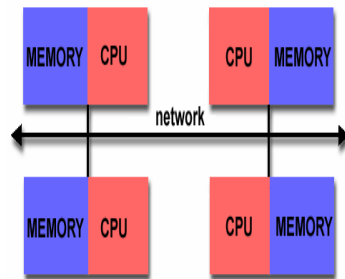
With neat diagrams discuss parallel computing memory architectures

Answer:

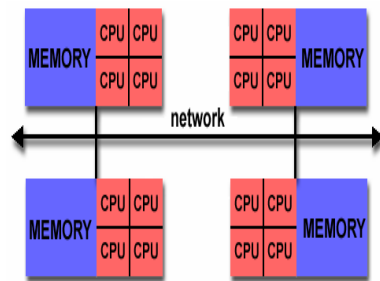
i. **Shared Memory Architecture:** Shared memory parallel computers vary widely, but generally have in common the ability for all processors to access all memory as global address space. Multiple processors can operate independently but share the same memory resources. Changes in a memory location effected by one processor are visible to all other processors. Shared memory machines can be divided into two main classes based upon memory access times: **UMA** and **NUMA**.



ii. **Distributed Memory Architecture:** Like shared memory systems, distributed memory systems vary widely but share a common characteristic. Distributed memory systems require a communication network to connect inter-processor memory. Processors have their own local memory. Memory addresses in one processor do not map to another processor, so there is no concept of global address space across all processors.



iii. **Hybrid Memory Architecture:** The largest and fastest computers in the world today employ both shared and distributed memory architectures. The shared memory component is usually a cache coherent SMP machine. Processors on a given SMP can address that machine's memory as global. The distributed memory component is the networking of multiple SMPs. SMPs know only about their own memory - not the memory on another SMP. Therefore, network communications are required to move data from one SMP to another.

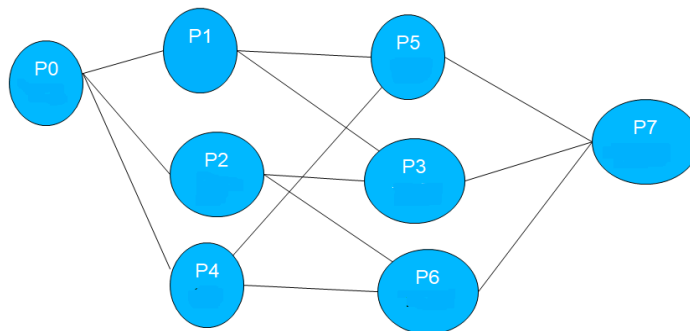


02
x
03
=
06

- b) **Explain the features of Very Long Instruction Word (VLIW) processor.**
- The processors in this architecture have multiple functional units, fetch from the Instruction cache that have the Very Long Instruction Word.
 - Multiple independent operations are grouped together in a single VLIW Instruction. They are initialized in the same clock cycle.
 - Each operation is assigned an independent functional unit.
 - All the functional units share a common register file.
 - Instruction words are typically of the length 64-1024 bits depending on the number of execution unit and the code length required to control each unit.
 - Instruction scheduling and parallel dispatch of the word is done statically by the compiler.
 - The compiler checks for dependencies before scheduling parallel execution of the instructions.

05

- c) Consider an input sequence of data elements as shown below.
7,8,1,2,9,10,3,4,5,6,11,13,16,19,21,23.

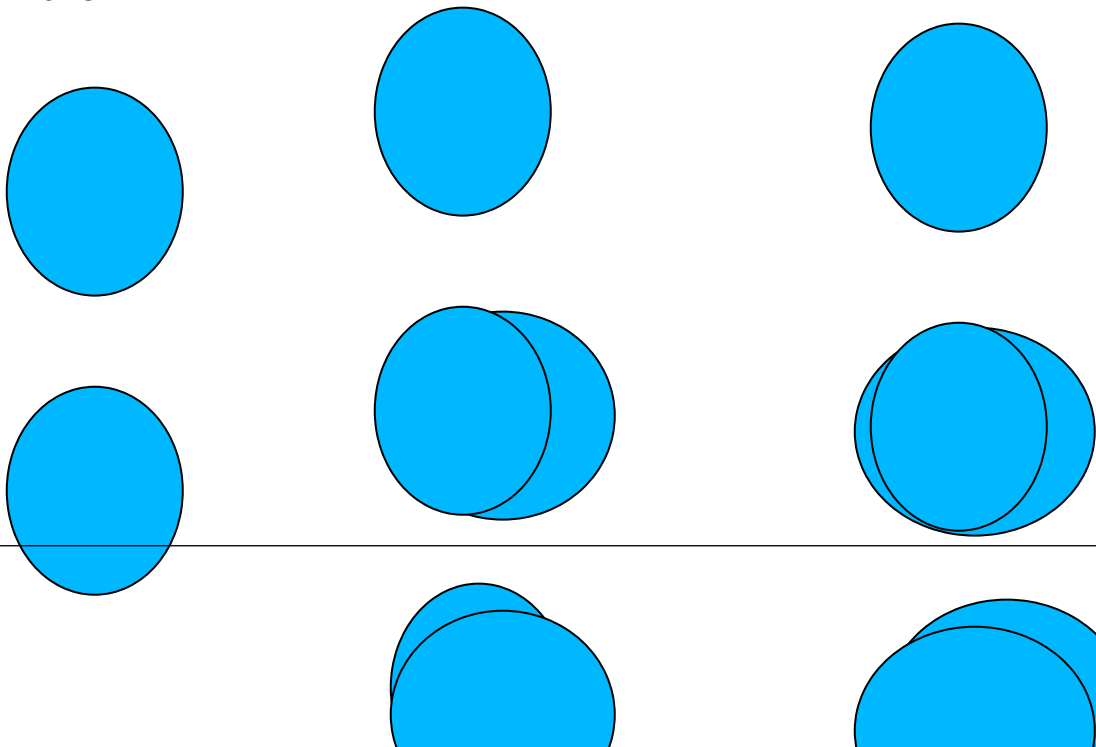


Compute the sum of all the elements using the hypercube parallel model.

Hint: Initial assignment of data items may be as follows:

Iteration1: P0 adds (7, 8), P1 adds (1,2), P2 adds (9,10) and so on where p7 adds 21 & 23. Perform the computations for each iteration to show the computations for all the iterations.

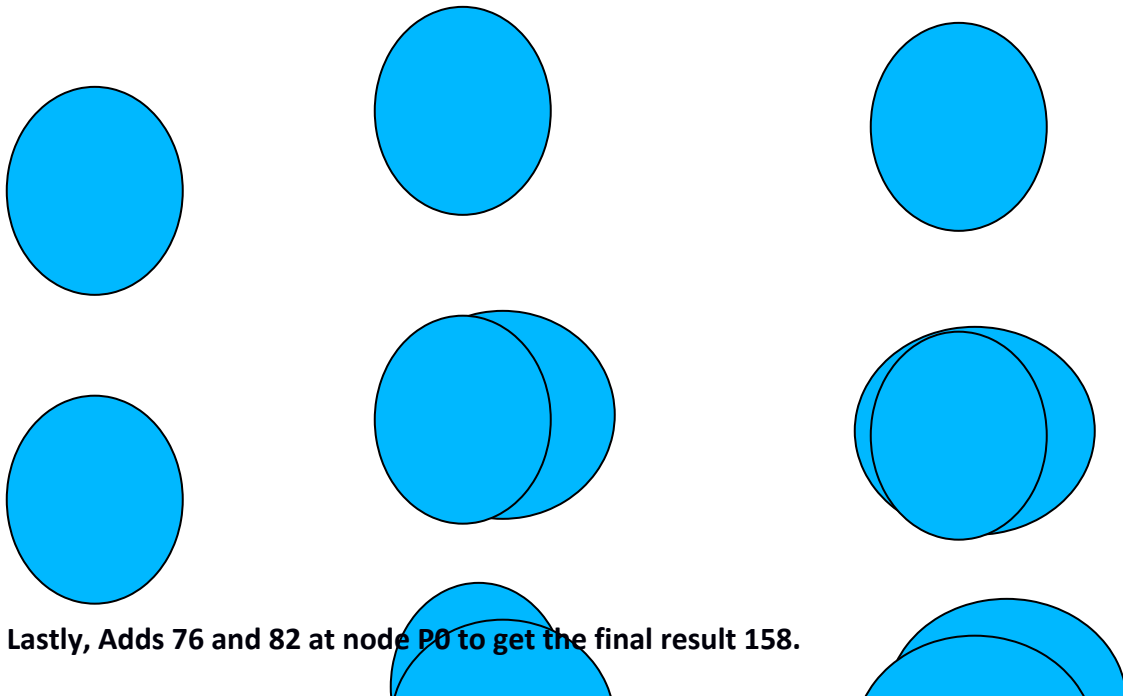
Answer:



03
+
02
=
05

c)

Answer continued:


03
+
02
=
05

d)

Consider a parallel computing system with 16 processors. Compute the speedup if the code has 4% of serial processing. Also, compute the scaled speedup if the code can be 90% parallelized.

{ Hint: Use Amdahl's and Gustafson's Laws formulae appropriately}.

Answer:

Amdahl's Law:

$$\begin{aligned}\text{Speedup} &= 1 / (1-F) + (F/N) \\ &= 1 / (1-0.96) + (0.96/16) \\ &= 10 \text{ times}\end{aligned}$$

Gustafson's Law:

$$\begin{aligned}\text{Scaled Speedup} &= N + (1-N) * S \\ &= 16 + (1-16) * 0.10 \\ &= 14.5 \text{ times}\end{aligned}$$

02
+
02
=
04