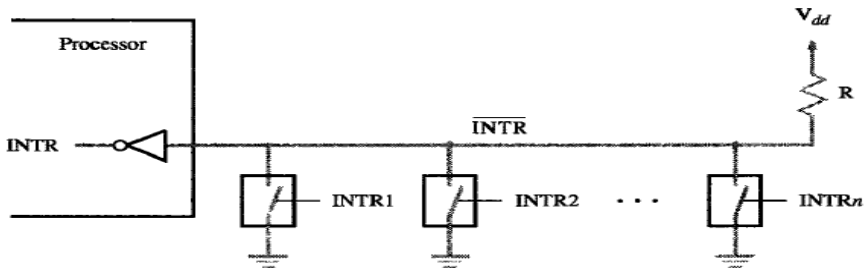


**Internal Assessment Test1–December 2022**



**(Schemes and Solutions)**

Sub:	<b>Computer Organization and Architecture</b>				Sub Code:	21CS34	Branch:	ISE																												
Date:	03/12/2022	Duration:	90min's	Max Marks:	50	Sem /Sec:	III A, B & C		OBE																											
<b>Answer any FIVE FULL Questions</b>								MARKS	CO	RBT																										
1	<p><b>Write a note on byte addressability, big endian and little endian Assignments with supportive diagrams.</b></p> <p><b>Byte Addressability-</b> As a simple example, consider a computer with a 16-bit address space. The machine would have 65,536 (<math>64K = 2^{16}</math>) addressable entities. The maximum memory size would depend on the size of the addressable entity.</p> <table style="margin-left: auto; margin-right: auto;"> <tr> <td>Byte Addressable</td> <td>64 KB</td> </tr> <tr> <td>16-bit Word Addressable</td> <td>128 KB</td> </tr> <tr> <td>32-bit Word Addressable</td> <td>256 KB</td> </tr> </table> <p><b>Big Endian</b> In this format, what is stored in the memory will be consumed in the same sequence. Because memory stores the bytes in the actual sequence. If a 32-bit word has value 0x0A0B0C0D. Then, that word will be stored as below in the memory (also showing how it will be in the register).</p> <p><b>Little Endian</b> In this format, the LSB is stored at the lowest address followed by higher significant bytes. This is opposite to how bytes were stored in BigEndian format. If a 32-bit word has value 0x0A0B0C0D. Then, that word will be stored as below in the memory (also showing how it will be in the register). To remember which is which, recall whether the least significant byte is stored first (thus, little endian) or the most significant byte is stored first (thus, big endian). Note that it is “byte” instead of “bit”. LSB also means Least Significant Byte.</p> <table style="margin-left: auto; margin-right: auto; border-collapse: collapse;"> <tr> <td style="padding: 0 10px;">The Big End</td> <td style="border: 1px solid black; padding: 2px; text-align: center;">31</td> <td style="border: 1px solid black; padding: 2px; text-align: center;">23</td> <td style="border: 1px solid black; padding: 2px; text-align: center;">15</td> <td style="border: 1px solid black; padding: 2px; text-align: center;">7</td> <td style="border: 1px solid black; padding: 2px; text-align: center;">0</td> <td style="padding: 0 10px;">The Little End</td> </tr> <tr> <td></td> <td style="border: 1px solid black; padding: 2px; text-align: center;">01</td> <td style="border: 1px solid black; padding: 2px; text-align: center;">02</td> <td style="border: 1px solid black; padding: 2px; text-align: center;">03</td> <td style="border: 1px solid black; padding: 2px; text-align: center;">04</td> <td></td> <td></td> </tr> <tr> <td></td> <td style="padding: 0 5px;">Byte 3</td> <td style="padding: 0 5px;">Byte 2</td> <td style="padding: 0 5px;">Byte 1</td> <td style="padding: 0 5px;">Byte 0</td> <td></td> <td></td> </tr> </table>						Byte Addressable	64 KB	16-bit Word Addressable	128 KB	32-bit Word Addressable	256 KB	The Big End	31	23	15	7	0	The Little End		01	02	03	04				Byte 3	Byte 2	Byte 1	Byte 0			4+3+3	CO1	L1
Byte Addressable	64 KB																																			
16-bit Word Addressable	128 KB																																			
32-bit Word Addressable	256 KB																																			
The Big End	31	23	15	7	0	The Little End																														
	01	02	03	04																																
	Byte 3	Byte 2	Byte 1	Byte 0																																
2a	<p><b>Define Processor clock, SPEC rating and Basic performance equation of computer.</b></p> <p><b>Processor clock</b> Processor circuits are controlled by a timing signal called clock. The clock designer the regular time intervals called clock cycles. To execute a machine instruction the processor divides the action to be performed into a sequence of basic steps that each step can be completed in one clock cycle. The length P of one clock cycle is an important</p> <p><b>Basic performance Equation</b></p> $T = \frac{N \times S}{R}$ <p><b>Spec Rating</b></p> <p>SPEC rating = <math>\frac{\text{Running time on the reference computer}}{\text{Running time on the computer under test}}</math></p>						2*3	CO1	L1																											

2b	<p><b>Explain Subroutine and Interrupt Service Routine (ISR).</b></p> <table border="1" data-bbox="191 86 1227 302"> <thead> <tr> <th data-bbox="191 86 764 128">Subroutine</th> <th data-bbox="764 86 1227 128">ISR</th> </tr> </thead> <tbody> <tr> <td data-bbox="191 128 764 302">A subroutine performs a function required by the program from which it is called</td> <td data-bbox="764 128 1227 302">The interrupt-Service routine may not have anything in common with the program being executed at the time the interrupt request is received</td> </tr> </tbody> </table>	Subroutine	ISR	A subroutine performs a function required by the program from which it is called	The interrupt-Service routine may not have anything in common with the program being executed at the time the interrupt request is received	4	CO1	L2																																
Subroutine	ISR																																							
A subroutine performs a function required by the program from which it is called	The interrupt-Service routine may not have anything in common with the program being executed at the time the interrupt request is received																																							
3	<p><b>A program contains 500 instructions. Out of that 25% instruction requires 5 clock cycles.40% instructions require 3 clock cycles and remaining requires 4 clock cycles for execution. Find the total time required to execute the program running in a 1GHz machine.</b></p>	5+5 (procedure and sol)	CO1	L3																																				
4	<p><b>What is an addressing mode? Explain any 5 different addressing modes with example and supportive diagrams.</b></p> <p>Table 2.1 Generic addressing modes</p> <table border="1" data-bbox="191 632 1138 1052"> <thead> <tr> <th>Name</th> <th>Assembler syntax</th> <th>Addressing function</th> </tr> </thead> <tbody> <tr> <td>Immediate</td> <td># Value</td> <td>Operand = Value</td> </tr> <tr> <td>Register</td> <td>Ri</td> <td>EA = Ri</td> </tr> <tr> <td>Absolute (Direct)</td> <td>LOC</td> <td>EA = LOC</td> </tr> <tr> <td>Indirect</td> <td>(Ri)</td> <td>EA = [Ri]</td> </tr> <tr> <td></td> <td>(LOC)</td> <td>EA = [LOC]</td> </tr> <tr> <td>Index</td> <td>X(Ri)</td> <td>EA = [Ri] + X</td> </tr> <tr> <td>Base with index</td> <td>(Ri, Rj)</td> <td>EA = [Ri] + [Rj]</td> </tr> <tr> <td>Base with index and offset</td> <td>X (Ri, Rj)</td> <td>EA = [Ri] + [Rj] + X</td> </tr> <tr> <td>Relative</td> <td>X(PC)</td> <td>EA = [PC] + X</td> </tr> <tr> <td>Autoincrement</td> <td>(Ri)+</td> <td>EA = [Ri]; Increment Ri</td> </tr> <tr> <td>Autodecrement</td> <td>-(Ri)</td> <td>Decrement Ri; EA = [Ri]</td> </tr> </tbody> </table> <p>EA = effective address Value = a signed number</p>	Name	Assembler syntax	Addressing function	Immediate	# Value	Operand = Value	Register	Ri	EA = Ri	Absolute (Direct)	LOC	EA = LOC	Indirect	(Ri)	EA = [Ri]		(LOC)	EA = [LOC]	Index	X(Ri)	EA = [Ri] + X	Base with index	(Ri, Rj)	EA = [Ri] + [Rj]	Base with index and offset	X (Ri, Rj)	EA = [Ri] + [Rj] + X	Relative	X(PC)	EA = [PC] + X	Autoincrement	(Ri)+	EA = [Ri]; Increment Ri	Autodecrement	-(Ri)	Decrement Ri; EA = [Ri]	2*5	CO1	L2
Name	Assembler syntax	Addressing function																																						
Immediate	# Value	Operand = Value																																						
Register	Ri	EA = Ri																																						
Absolute (Direct)	LOC	EA = LOC																																						
Indirect	(Ri)	EA = [Ri]																																						
	(LOC)	EA = [LOC]																																						
Index	X(Ri)	EA = [Ri] + X																																						
Base with index	(Ri, Rj)	EA = [Ri] + [Rj]																																						
Base with index and offset	X (Ri, Rj)	EA = [Ri] + [Rj] + X																																						
Relative	X(PC)	EA = [PC] + X																																						
Autoincrement	(Ri)+	EA = [Ri]; Increment Ri																																						
Autodecrement	-(Ri)	Decrement Ri; EA = [Ri]																																						
5	<p><b>Explain Interrupt Hardware in detail with diagram.</b></p>  <p><b>Figure 4.6</b> An equivalent circuit for an open-drain bus used to implement a common interrupt-request line.</p> <ul style="list-style-type: none"> <li>• Interrupt request</li> <li>• If several I/O can request an interrupt.</li> <li>• A single interrupt request line may be used as shown in figure using switch.</li> <li>• When device make interrupt request by closing switch the voltage on the line is '0'</li> <li>• INTR is logic OR</li> <li>• <math>INTR = INTR1 + INTR2 + \dots + INTRn</math></li> <li>• INTR is active low signal</li> </ul>	5+5 (diagram +theory)	CO2	L2																																				
6	<p>Explain the following methods of handling interrupt from multiple devices</p> <p>i) Vectored interrupt ii) Interrupt priority.</p> <p><b>Vectored Interrupts</b></p>	10	CO2	L2																																				

- ✓ To reduce the time involved in the polling process, a device requesting an interrupt may identify itself directly to the processor. Then, the processor can immediately start executing the corresponding interrupt-service routine.
- ✓ A device requesting an interrupt can identify itself by sending a special code to the processor over the bus. This enables the processor to identify individual devices even if they share a single interrupt-request line.
- ✓ The code supplied by the device may represent the starting address of the interrupt-service routine for that device. The code length is typically in the range of 4 to 8 bits. The remainder of the address is supplied by the processor based on the area in its memory where the addresses for interrupt-service routines are located.
- ✓ This arrangement implies that the interrupt-service routine for a given device must always start at the same location. The programmer can gain some flexibility by storing in this location an instruction that causes a branch to the appropriate routine.

### Interrupt Priority

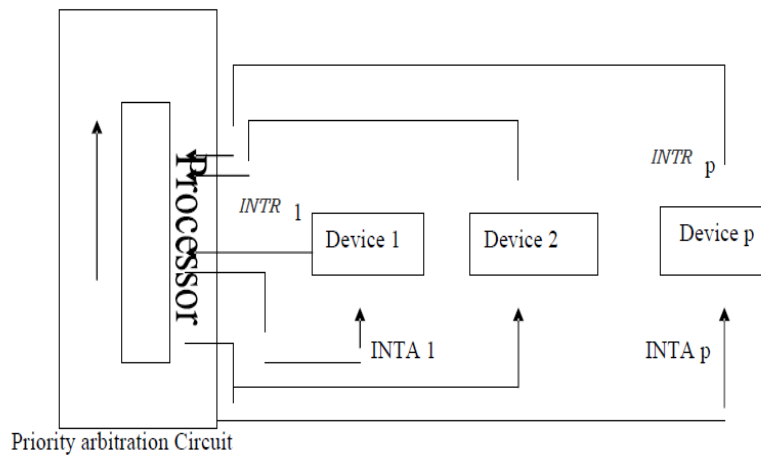


Figure2: Implementation of interrupt priority using individual interrupt-request and acknowledge lines.