USN : _____

CMR INSTITUTE OF TECHNOLOGY, BANGALORE
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INTERNAL ASSESSMENT – I

Semester: 3 Date: 20/12/2021

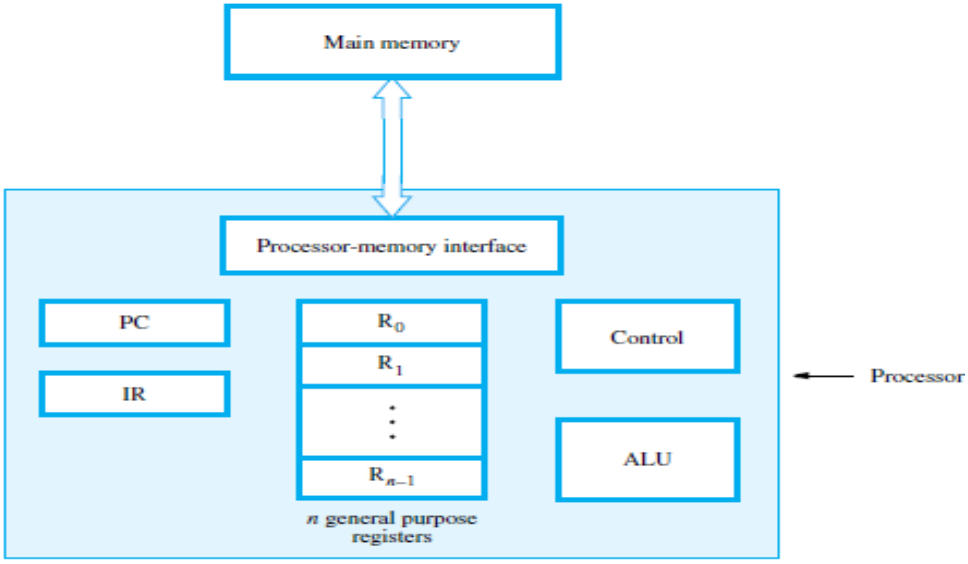
Subject: COMPUTER ORGANIZATION (18CS34)

Faculty: Prof. Savitha S, Dr. R. Kesavamoorthy Max Marks: 50

Instructions to Students:

ANSWER ANY 5 Question(s)

Marks CO BT/CL

Question #		Description	Marks Distribution		M
1	a)	With a neat diagram, explain the basic operational concept of a computer  **Figure 1.2** Connection between the processor and the main memory. • The **processor** contains ALU , control-circuitry and many registers. • The processor contains 'n' general-purpose registers **R0** through **Rn-1**. • The **IR** holds the instruction that currently need to be decoded and executed. • The **control-unit** generates the timing-signals that determine when a given action is to take place. • The **PC** contains the memory-address of the next-instruction to be fetched & executed. • During the execution of an instruction, the contents of PC are updated to point to next instruction.	1M 2M	5M	

- The **MAR** holds the address of the memory-location to be accessed.
- The **MDR** contains the data to be written into or read out of the addressed location.
- MAR and MDR facilitate the communication with memory.

IR –Instruction Register
PC- Program counter
MAR-Memory Address Register
MDR- Memory Data Register

2M

- The address of first instruction gets loaded into PC.
- The contents of PC (i.e. address) are transferred to the MAR & control-unit issues Read signal to memory.
- After certain amount of elapsed time, the first instruction is read out of memory and placed into MDR.
- Next, the contents of MDR are transferred to IR. At this point, the instruction can be decoded & executed.
- To fetch an operand, its address is placed into MAR & control-unit issues Read signal. As a result , the operand is transferred from memory into MDR, and then It is transferred from MDR to ALU. Likewise required number of operands is fetched into processor.
- Finally, ALU performs the desired operation.

  - If the result of this operation is to be stored in the memory, then the result is sent to the MDR.
  - The address of the location where the result is to be stored is sent to the MAR and a Write cycle is initiated.
  - At some point during execution, contents of PC are incremented to point to next instruction in the program.

Example:

- An Instruction consists of 2parts, 1) Operation code (Op code) and 2)Operands.

OPCODE	OPERANDS

- The data/ operands are stored in memory.
- The individual instruction is brought from the memory to the processor.
- Then, the processor performs the specified operation.
- Let us see a typical instruction

ADD LOCA, R0

- This instruction is an addition operation. The following are the steps to execute

the instruction:

Step1: Fetch the instruction from main-memory into the processor.
Step2: Fetch the operand at location LOCA from main-memory into the processor.
Step3: Add the memory operand(fetched contents of LOCA) to the contents of register R0.

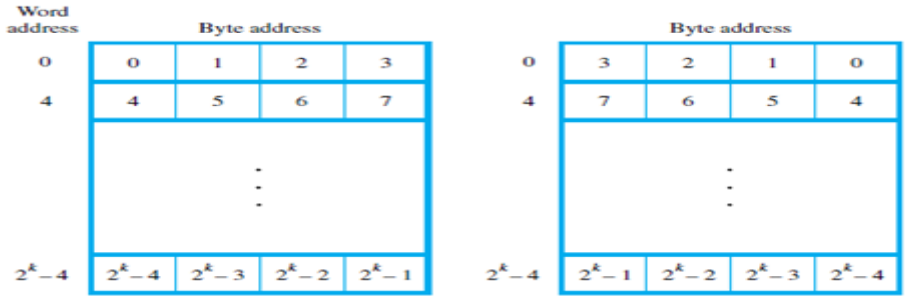
		Step4: Store the result in R0.		
	b)	What is performance measurement? Explain the overall SPEC rating for the computer in a program suit. 1 Performance measurement is the measure of how well a processor operates for a given bench mark. 2 SPEC selects & publishes the standard programs along with their test results for different application domains.(SPEC-System Performance Evaluation Corporation). 3 SPEC Rating is given by SPEC Rating = (Running time of the reference Computer) / (Running time of the Computer Under test) • If SPEC rating is 50, than computer under test is 50 times as fast as reference-computer. • The test is repeated for all the programs in the SPEC suite. • Then, the geometric mean of the results is computed. • Let SPEC ¡=Rating for program 'i' in the suite. • Overall SPEC rating for the computer is given by $\text{SPEC rating} = \left(\prod_{i=1}^{n} \text{SPEC}_i\right)^{\frac{1}{n}}$ Where n= no. of programs in the suite.	2M 1M 2M	5M

2	a)	Show how the below expression will be executed in one address, two address and three address processors in an accumulator organization.
		
		X = A x B + C x D
		
		One Address X=A*B+C*D
		Load A
		MULTIPLY B
		Store X
		Load C
		MULTIPLY D
		Store Y
		Load X
		ADD Y
		Store S
		
		Two Address X=A*B+C*D
		Mov A,R1
		Multiply B,R1
		Mov C, R2
		Multiply D,R2
		Add R1,R2
		Mov R2,X
		
		Three Address
		Multiply A, B, R1
		Multiply C, D, R2
		Add R1, R2, X
			2M+
			1.5M+
			1.5M
				5M

b) With relevant figure, discuss about the Big Endian and Little Endian assignments.

- There are two ways in which byte-addresses are arranged (Figure2.3).
  1) **Big- Endian:** Lower byte-addresses are used for the more significant bytes of the word.
  2) **Little- Endian:** Lower byte-addresses are used for the less significant bytes of the word

  In both cases, byte-addresses 0,4,8.....are taken as the addresses of successive words in the memory.



(a) Big-endian assignment (b) Little-endian assignment

Figure 2.3 Byte and word addressing.

2.5M

5M

- Consider a 32-bit integer (in hex): 0**x**12345678 which consists of 4 bytes:12, 34,56,and 78.
  - Hence this integer will occupy 4 bytes in memory.
  - Assume, we store it at memory address starting 1000.

    - **On little-endian, memory will look like**

2.5M

Address	Value
1000	78
1001	56
1002	34
1003	12

    - **On big-endian, memory will look like**

Address	Value
1000	12
1001	34
1002	56
1003	78

3 Define addressing mode. Explain the various types of addressing modes with example.

The term addressing modes refers to the way in which the operand of an instruction is specified. Information contained in the instruction code is the value of the operand or the address of the operand. Following are the main addressing modes that are used on various platforms and architectures. Listing out all the addressing modes.

2M

Register Addressing Mode

- The operand is the content of a processor register. Register name is specified in the instruction.

1M

		• Effective Address of the Operand: Register name specified in the instruction **ADD R0, R1** ; R1 <- R0 + R1 **Immediate Addressing Mode** • The operand is given explicitly in the instruction • Effective Address of the Operand: Operand value given in the instruction **ADD #10, R1** ; R1 <- 10 + R1 **Direct (or Absolute) Addressing Mode** • The operand is a Memory location. The address of the memory location is given in the instruction explicitly. • Effective Address of the Operand: Address of the memory location given directly in the instruction ADD LOCA, R1 **Indirect Addressing Mode** • Here neither the operands nor the their addresses are given explicitly. The instruction provides the information from which the address of the operand is determined i.e., the instruction provides effective address of the operand using register or memory location. The indirection is denoted by () sign around register or memory. • Effective Address of the Operand: (Ri) or (LOCA) is the contents of a register or the memory location whose address appears in the instruction **Ex1: ADD LOCA, R1** **Ex2: ADD (R1), R2** **Index Addressing Mode** • The effective address of the operand is generated by adding a constant value to the contents of a register specified in the instruction. The register in this case is called as Index register. • The operation is indicated as X(Ri). • Effective Address of the Operand: X+Ri where X is a constant value (signed integer) and Ri is the index register. **Ex1: ADD 5(R1), R2** **Ex2: ADD 3(R1, R3), R2** **Relative Addressing Mode** • In this mode the content of the program counter is added to the address part of the instruction to obtain the effective address. • Effective Address: X+PC where X is a constant value (signed integer) and PC is	1M 1M 1M 1M 1M	10M

		the contents of the program counter. **Auto increment Addressing Mode** • This is indirect mode with a modification. The effective address of the operand is the contents of a pointer register specified in the instruction. After accessing the operand, the contents of this pointer register is incremented automatically to point to the next entity. • The mode is denoted by (Ri)+, where Ri is the pointer register. • The + sign indicates that Ri is incremented after the operation. • The increment operation is depending on the size of the accessed operand. Thus, the increment value is 1 for byte-size operands, 2 for word-size (16-bit) operands and 4 for long-word (32-bit) operands. • This mode is useful when operands are stored consecutively in memory i.e., for array manipulation **Move (R1)+, R2** **Auto decrement Addressing Mode**This mode is useful to access an array in the reverse order. The value of the pointer register specified in the instruction is decremented first and this value is used as the effective address of the operand. • The mode is denoted by -(Ri), where Ri is the pointer register. • The - sign indicates that Ri is decremented before accessing the operand. • The decrement operation is depending on the size of the accessed operand. Thus, the decrement value is 1 for byte-size operands, 2 for word-size (16-bit) operands and 4 for long-word (32-bit) operands. • This two modes (Autoincrement and Autodecrement) are useful to implement a data structure called Stack. **Move -(R1), R2**	1M 1M	
4	a)	Interpret the subroutine stack frame work with example. • SP, FP and Stack Frame **Stack Pointer** A processor register is used to keep track of the address of the element of the stack that is at the top at any given time. This register is called the **stack pointer (SP)**. **Frame Pointer** • In addition to the stack pointer SP, it is useful to have **another pointer register, called the frame pointer (FP)**, for convenient access to the parameters passed to the subroutine and to the local memory variables used by the subroutine. **Stack Frame** • During execution of the subroutine, there will be locations at the Top of the Stack that contain entries that are needed by the subroutine. • These locations constitute a private workspace for the subroutine, that will be created at the time the subroutine is entered and freed up when the subroutine returns control to the calling program. Such space is called a **stack frame**. • Example Program	1M	

Memory location		Instructions		Comments
Main program				
		⋮		
2000		Move	PARAM2,−(SP)	Place parameters on stack.
2004		Move	PARAM1,−(SP)	
2008		Call	SUB1	
2012		Move	(SP),RESULT	Store result.
2016		Add	#8,SP	Restore stack level.
2020		next instruction		
		⋮		
First subroutine				
2100	SUB1	Move	FP,−(SP)	Save frame pointer register.
2104		Move	SP,FP	Load the frame pointer.
2108		MoveMultiple	R0−R3,−(SP)	Save registers.
2112		Move	8(FP),R0	Get first parameter.
		Move	12(FP),R1	Get second parameter.
		⋮		
		Move	PARAM3,−(SP)	Place a parameter on stack.
2160		Call	SUB2	
2164		Move	(SP)+,R2	Pop SUB2 result into R2.
		⋮		
		Move	R3,8(FP)	Place answer on stack.
		MoveMultiple	(SP)+,R0−R3	Restore registers.
		Move	(SP)+,FP	Restore frame pointer register.
		Return		Return to Main program.
Second subroutine				
3000	SUB2	Move	FP,−(SP)	Save frame pointer register.
		Move	SP,FP	Load the frame pointer.
		MoveMultiple	R0−R1,−(SP)	Save registers R0 and R1.
		Move	8(FP),R0	Get the parameter.
		⋮		
		Move	R1,8(FP)	Place SUB2 result on stack.
		MoveMultiple	(SP)+,R0−R1	Restore registers R0 and R1.
		Move	(SP)+,FP	Restore frame pointer register.
		Return		Return to Subroutine 1.

Figure 2.28 Nested subroutines.

- Explanation for the Program

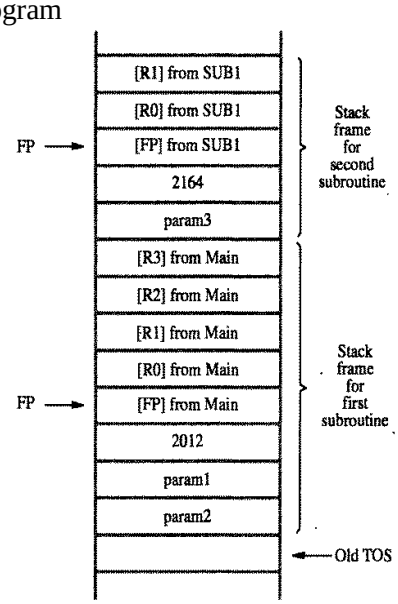


Figure 2.29 Stack frames for Figure 2.28.

2M

2M

5M

b) How the input output operations are to be performed by the processor? Write a program that reads a line of characters and displays it.

- Process of I/O Operations
  - A program monitors SIN, and when SIN is set to 1, the processor reads the contents of DATAIN.
  - When the character is transferred to the processor, SIN is automatically

1M

5M

cleared to 0.

- An analogous process takes place when characters are transferred from the processor to the display.
- A buffer register, DATAOUT, and a status control flag, SOUT, are used for this transfer. When SOUT equals 1, the display is ready to receive a character.

2M

- Program
- Explanation for the Program

```
        Move        #LOC,R0          Initialize pointer register R0 to point to the
                                       address of the first location in memory
                                       where the characters are to be stored.
READ    TestBit     #3,INSTATUS      Wait for a character to be entered
        Branch=0    READ               in the keyboard buffer DATAIN.
        MoveByte    DATAIN,(R0)      Transfer the character from DATAIN into
                                       the memory (this clears SIN to 0).
ECHO    TestBit     #3,OUTSTATUS     Wait for the display to become ready.
        Branch=0    ECHO
        MoveByte    (R0),DATAOUT     Move the character just read to the display
                                       buffer register (this clears SOUT to 0).
        Compare     #CR,(R0)+        Check if the character just read is CR
                                       (carriage return). If it is not CR, then
        Branch≠0    READ               branch back and read another character.
                                       Also, increment the pointer to store the
                                       next character.
```

2M

Figure 2.20 A program that reads a line of characters and displays it.

5

10M

Discuss in detail about

(i) Interrupt Nesting

- Priority Scheme with figure
  - Priority level of the processor is the priority of the program being executed.
  - Initialized with priority of the device whose ISR is being executed
  - Disables IR from devices with same level or low level of priority
  - Accept IR from devices with higher priority.

4M

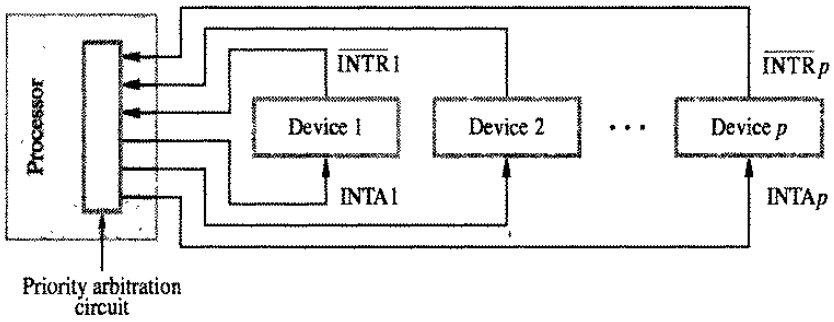
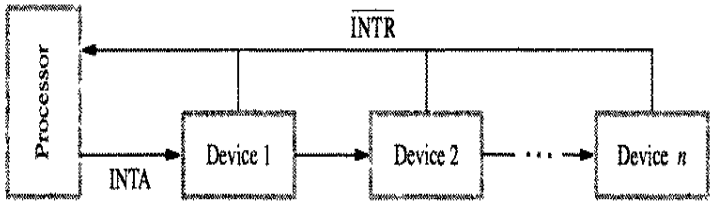
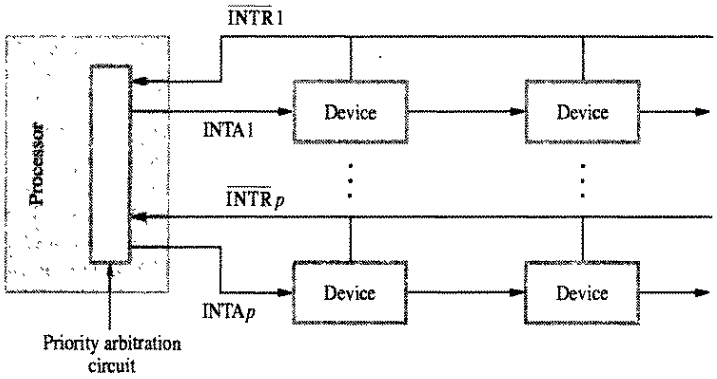


Figure 4.7 Implementation of interrupt priority using individual interrupt-request and acknowledge lines.

		(ii) Simultaneous Interrupts • Daisy Chain & Priority Groups with figure  (a) Daisy chain  (b) Arrangement of priority groups The processor must have some mechanisms to decide which request to service when simultaneous requests arrive. • INTR line is common to all devices • INTA line is connected in a daisy-chain fashion. • INTA signal propagates serially through devices. • When several devices raise an interrupt-request, INTR line is activated. • Processor responds by setting INTA line to 1. This signal is received by device 1. • Device-1 passes signal on to device 2 only if it does not require any service. • If device-1 has a pending-request for interrupt, the device-1blocks INTA signal & proceeds to put its identifying-code on data-lines. • Device that is electrically closest to processor has highest priority. • Advantage: It requires fewer wires than the individual connections. Arrangement of Priority Groups • Here, the devices are organized in groups & each group is connected at a different priority level. • Within a group, devices are connected in a daisy chain.	6M	
6		What is bus arbitration in DMA? Elucidate the different bus arbitration techniques **Bus Arbitration** • The device that is allowed to initiate data-transfers on bus at any given time is called bus master.		10M

• There can be only one bus-master at any given time.
• Bus Arbitration is the process by which next device to become the bus-master is selected &bus-mastership is transferred to that device.

2M

Centralized Method with figure

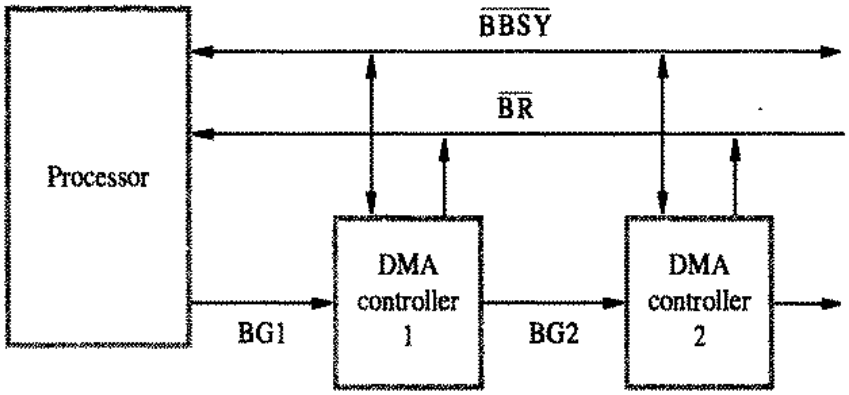
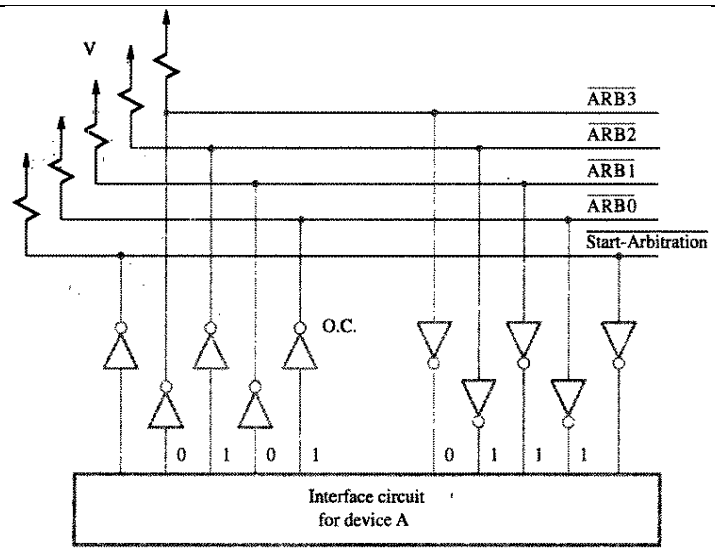


Figure 4.20 A simple arrangement for bus arbitration using a daisy chain.

4M

A single bus-arbiter performs the required arbitration.
• Normally, processor is the bus-master.
• Processor may grant bus-mastership to one of the DMA controllers.
• A DMA controller indicates that it needs to become bus-master by activating BR line.
• The signal on the BR line is the logical OR of bus-requests from all devices connected to it.
• Then, processor activates BG1 signal indicating to DMA controllers to use bus when it becomes free.
• BG1 signal is connected to all DMA controllers using a daisy-chain arrangement.
• If DMA controller-1 is requesting the bus,Then, DMA controller-1 blocks propagation of grant-signal to other devices.Otherwise, DMA controller-1 passes the grant downstream by asserting BG2.
• Current bus-master indicates to all devices that it is using bus by activating BBSY line.
• The bus-arbiter is used to coordinate the activities of all devices requesting memory transfers.
• Arbiter ensures that only 1 request is granted at any given time according to a priority scheme. (BR-Bus-Request, BG- Bus-Grant, BBSY-Bus Busy).

Distributed Method with figure

		 **Figure 4.22** A distributed arbitration scheme. • All device participate in the selection of next bus-master • Each device on bus is assigned a 4-bit identification number (ID). • When 1 or more devices request bus, they → assert Start-Arbitration signal &place their 4-bit ID numbers on four open-collector lines 0 BRA through 3 BRA . • A winner is selected as a result of interaction among signals transmitted over these lines. • Net-outcome is that the code on 4 lines represents request that has the highest ID number. • Advantage: This approach offers higher reliability since operation of bus is not dependent on any single device.	4M	