USN | | | | | | | | | | |

Internal Assesment Test I – Sept. 2019

Sub:	COMPUTER ORGANIZATION					Sub Code:	18CS34	Branch:	CSE
Date:	7 / 09 / 2019	Duration:	90 mins	Max Marks:	50	Sem / Sec:	3 (A,B,C)		OBE

Answer **FIVE** FULL questions selecting AT LEAST ONE question **FROM EACH PART**

	MARKS	CO	RBT

PART A

1 (a) List the steps needed to execute the machine instruction

ADD A, LOCA, R4

in terms of transfers between the components and some simple control commands. Assume that the instruction itself is stored in the memory at location LOC and that this address is initially in register PC. [10] CO1 L3

Operating steps

* Programs (list of instructions) reside in memory (usuall ~~stored there wrong~~ get there through input unit).

① PC is set to point to first instruction of program.
② This PC contents is transferred to MAR and Read c signal is sent to memory.
③ After time required to access the memory elapses, addressed word (1st instruction in this case) is read ou from memory and loaded in MDR.
④ MDR contents are transferred to IR.
⑤ If instruction involves an operation by ALU:

get operands from memory or general purpose register.
If operand resides in memory, its address is sent to MAR. Read cycle is initialized. Operand comes to MDR. It is sent to ALU. Similarly, more operands are sent to ALU (if required).
ALU performs operation and sends result to MDR.
The address of location where result is to be stored is sent to MAR, and Write cycle is initiated.
⑥ PC is incremented to point to next instruction.

NOTE: If a source of destination is a register (R), MAR, MDR steps are not required as registers are directly accessible to ALU as both reside inside processor. MAR and MDR are required only if we want to access main memory for read or write operation.

OR

2 (a)	Assume a program with 1000 instructions. 25% of the instructions take 4 clock cycles, 40% of the instructions take 5 clock cycles, and the remaining instructions take 3 clock cycles to execute respectively. Calculate the time of execution assuming basic performance equation if the clock rate of the machine is 1GHz. P=N*S/R Ans=4.05 micro sec	[5]	CO1	L3
(b)	Define Exception. Explain any 2 Kinds of Exception.	[5]	CO2	L2

Exceptions

Exception is any event that causes an interruption. I/O interrupts are one example of an exception.

Recovery from Errors

- Many computers include an error-checking code in main memory, which allows detection of errors in the stored data. If error occurs, the control hardware detects it and informs the processor by raising an interrupt.
- The processor may also interrupt a program if it detects an error or an unusual conditions while executing the instructions of program. Eg- invalid opcode, division by zero etc.

Debugging

System software usually includes a program called debugger, which helps the programmer find errors in a program. The debugger uses exceptions to provide two important facilities called trace and breakpoints.

- Trace - when a processor is operating in the trace mode, an exception occurs after execution of every instruction, using the debugging program as the exception-service routine. The debugging program enables user to examine the contents of registers, memory locations, etc. On return from debugging program, next instruction in program being debugged is executed, the debugging program is activated again. The trace exception is disabled during the execution of debugging program.
- Breakpoints - In this, the program being debugged is interrupted only at specific points selected by user. An instruction called Trap or Software-interrupt is usually provided for this purpose. Eg- user wants to interrupt program execution after instruction i. Debugging routine saves

PART B

3 (a) Explain basic instruction types with example. [6] CO1 L2

Q4 Instruction types (one-address, two-address, three-address instructions)

Basic Instruction Types

Two-address instruction

Operation Source, Destination

Move B, C
Add A, C

C ← [A] + [B]

Add contents of A & B, place sum in C. A, B contents are u

Consider three-address instructio

Operation Source1, Source2, Destination

Add A, B, C

It may happen that 2-address instruction doesn fit in one word for usual word length & addr size. In that case, we may adopt one-addres instruction. Operation Source/Destination.

A processor register, usually called the accumulat may be used for this purpose. This may be used a register to temporarily hold values.

Load	A	Copy A contents to a
Add	B	Add B contents to accu
Store	C	Store accumulator to C.

(b) Three devices, A, B, and C, are connected to the bus of a computer. I/O transfers for all three devices use interrupt control. Interrupt nesting for devices A and B is not allowed, but interrupt requests from C may be accepted while either A or B is being serviced. Suggest different ways in which this can be accomplished in each of the following cases:
(a) The computer has one interrupt-request line. [4] CO2 L3

(b) Two interrupt-request lines, INTR1 and INTR2, are available, with INTR1 having higher priority.
Specify when and how interrupts are enabled and disabled in each case.

the instruction will be completed without interruption.

4.6. (*a*) Interrupts should be enabled, except when C is being serviced. The nesting rules can be enforced by manipulating the interrupt-enable ags in the interfaces of A and B.

(*b*) A and B should be connected to $INTR_2$, and C to $INTR_1$. When an interrupt request is received from either A or B, interrupts from the other device will be automatically disabled until the request has been serviced. However, interrupt requests from C will always be accepted.

OR

4 (a) Explain addressing modes with example of each mode. [10] CO1 L2

ADDRESSING MODES

The different ways in which the location of an operand is specified in an instruction are referred to as modes.

① Immediate mode

The operand is given explicitly in the instruction.

Eg. Move #200, R0

Above instruction places value 200 in Regis

Generally, this mode is used to represent con

② Register mode

The operand is the contents of a processor reg

name of the register is given in the instruction

Eg. - Move R1, R2.

Above instruction moves the contents of register

Generally, this mode is used to access variables.

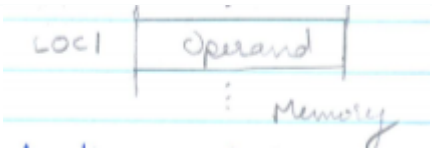
③ Absolute (Direct) mode

The address of the memory location where operand is located is given explicitly in the instruction.

Eg- Move LOC1, LOC2

Above instruction moves the operand at location LOC1 to location LOC2.

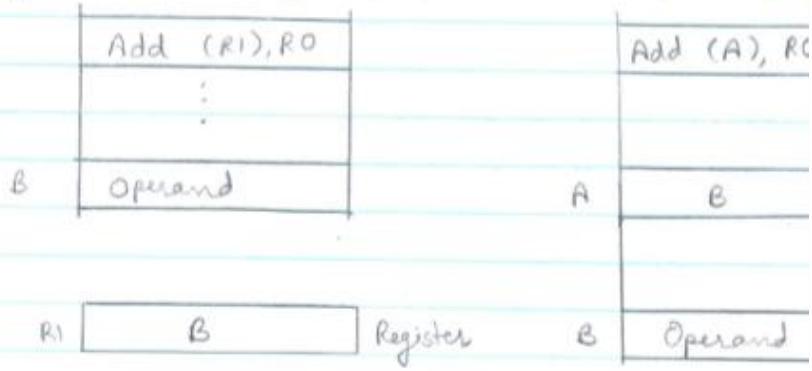
It is generally used for representing global variables.



④ Indirect mode

The contents of register or memory location given in instruction gives the effective address of the operand.

Eg.-



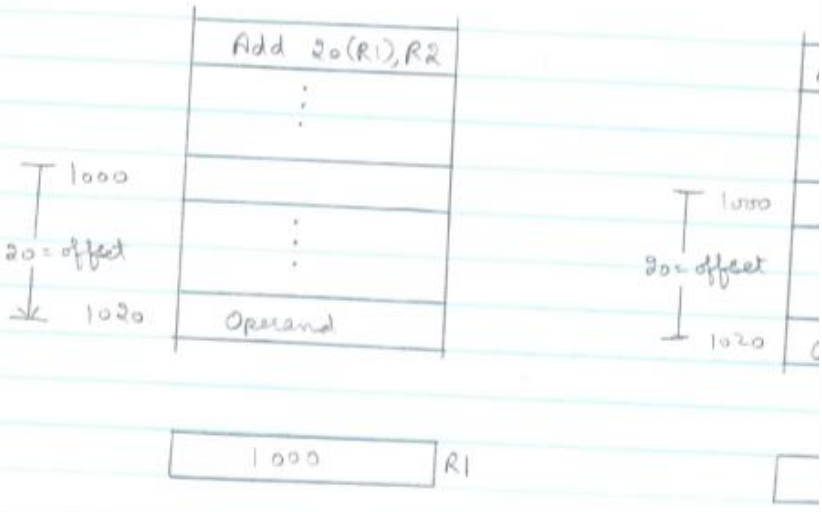
(a) Through a general purpose register (b) Through

Indirect addressing

⑤ Index mode

The effective address of the operand is adding a constant value to the content register. The register used may be a spe or a general-purpose register and is kn an index register.

Eg.



(a) offset is given as a constant (b) offset is in

Indexed addressing

Base with index mode

The effective address is the sum of the co of multiple registers.

Eg - R1 | 1000 | R2 | 20 | 1020

(R1, R2)

Base with index and offset

2 registers and a constant is used to calcu address of the operand

Eg - 20(R1, R2)

R1 | 1000 | R2 | 20 | + 20 | 1040

⑥ Relative mode

The effective address is determined by the using the program counter in place of the purpose register Ri.

Eg - 20(PC)

If PC has address 1000, then effective ad operand will be 1020.

⑦ Autoincrement mode

The effective address of the operand is the c register specified in the instruction. After access operand, the contents of this register are incremented to point to the next item in a

Eg - Add (R2)+, R0

⑧ Autodecrement mode

The contents of a register specified in th are first automatically decremented and are as effective address of the operand.

Eg - Add -(R2), R0

PART C

5 (a) Explain the operation of stack with example. [8]

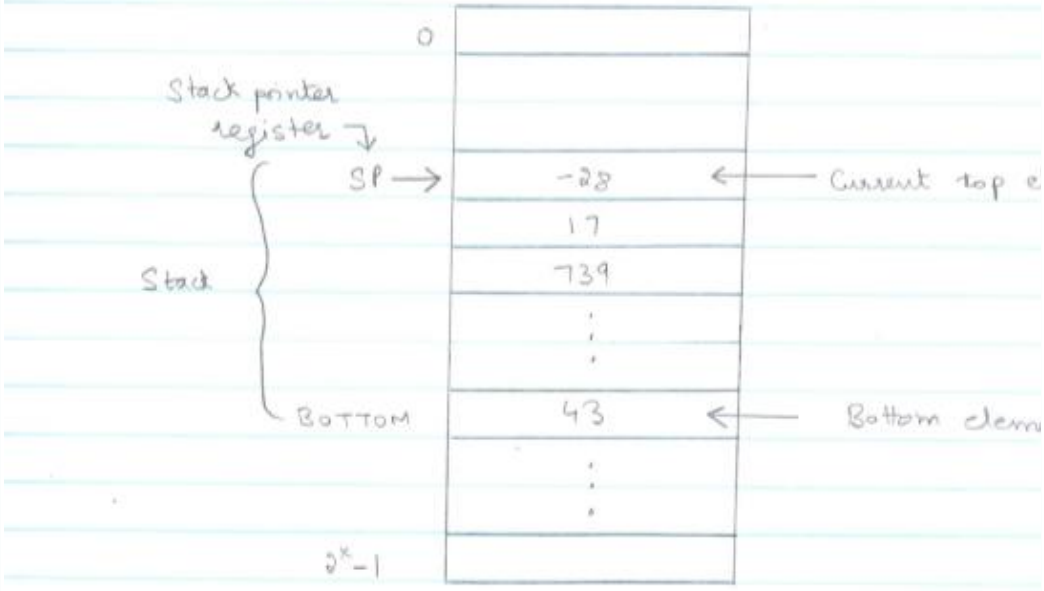
C01 L2

[2]

A stack is a list of data elements, usually bytes, with the accessing restriction that elements can be added or removed at one end of the list only. This end is called the top of stack, and the other end is called bottom. The structure is sometimes referred to as a pushdown stack. It is just like a pile of trays. It is also called as LIFO (Last In First Out) stack. Push operation is used to place a new item on the stack, pop operation is used to remove the top item from the stack.

Assume that the first element is placed in location BOTTOM, and when new elements are pushed onto the stack, they are placed in successively lower memory locations. We use a stack that grows in the direction of decreasing memory addresses.

Consider a stack containing numerical values, with 43 at the bottom and -28 at the top. 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).



A Stack of words in the memory

Assume a byte-addressable memory with a 32-bit word length.

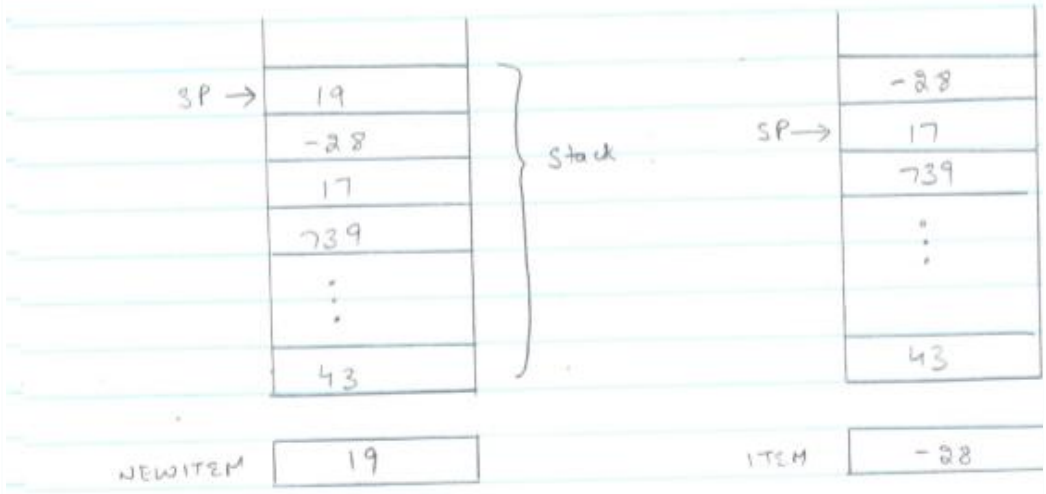
Push :
```
Subtract  #4, SP
Move      NEWITEM, (SP)
```

Pop :
```
Move      (SP), ITEM
Add       #4, SP
```

If processor has Autoincrement and Autodecrement addressing modes, push and pop can be performed using single instructions :-

Push :- `Move NEWITEM, -(SP)`

Pop :- `Move (SP)+, ITEM`



a) After push from NEWITEM b) After pop

* Consider stack size from 1500 to 2000 memory location

```
SAFEPUSH   Compare   #1500,SP     Check to see if SP
           Branch≤0  FULLERROR    an address value eq
                                  or less than 1500. I
                                  stack is full. Branch
                                  routine FULLERROR f
           Move      NEWITEM,-(SP) Otherwise, push eleme
                                  memory location NEW
                                  stack
```

(a) Routine for a safe push operation

```
SAFEPOP    Compare   #2000,SP     check to see if
           Branch>0  EMPTYERROR   an address value
                                  than 2000. If it d
                                  stack is empty - B
                                  routine EMPTYERRO
           Move      (SP)+,ITEM   Otherwise, pop the to
                                  stack into memory lo
                                  ITEM
```

(b) Routine for a safe pop operation

Differentiate between subroutine and Interrupt service Routine

OR

6 (a) Define Subroutine. Explain the concept of Subroutine Nesting with example. [7] CO2 L1,L2

SUBROUTINES

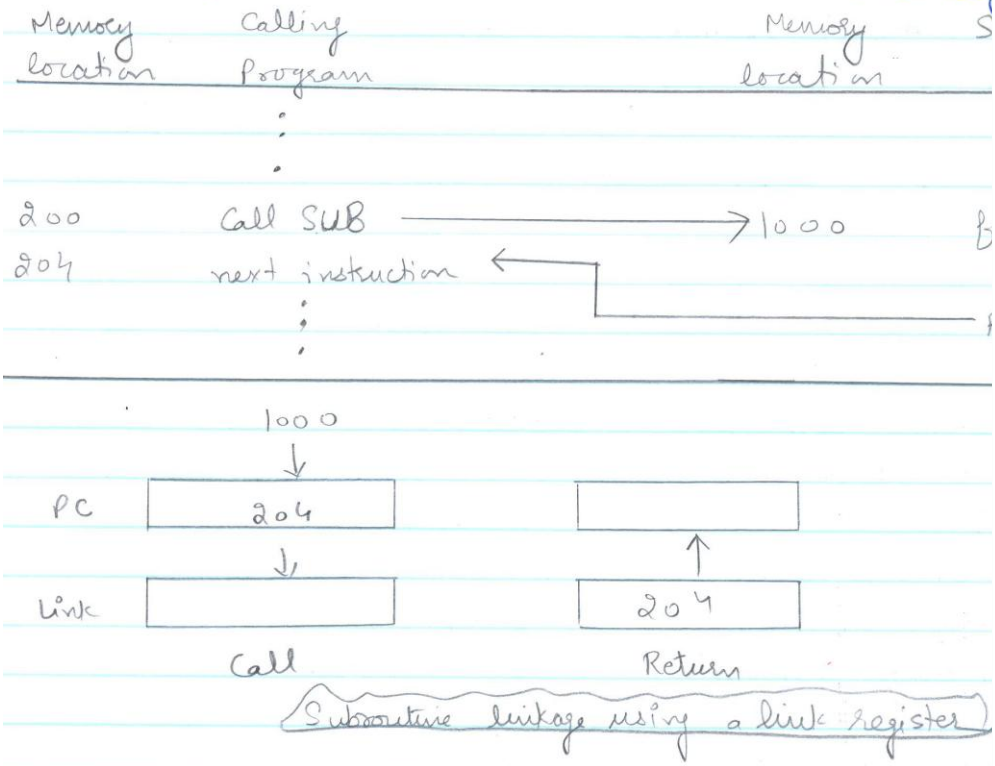
In a program, if a particular subtask is performed mo
on different data values, such subtask is usually
a subroutine. Eg subroutine to evaluate the sine f
To save space, only one copy of the instructions that
the subroutine is placed in the memory, and any
that requires the use of the subroutine simply br
to its starting location. This branching to a subrou
called as calling the subroutine. The instruction that
this branch operation is named a Call instruction. Th
is said to return (resume execution, continuing immedi
the call instruction) to the program that called it by exe
a Return instruction. Contents of PC must be sav
the Call instruction to enable correct return to the
program.
The method followed to call and return from subroutin
referred to as subroutine linkage method.
Eg- save a return address in a specific location like a
link register. When subroutine completes its task, the
instruction returns to the calling program by branching
indirectly through the link register.

Call instruction

- Store contents of PC in link register
- Branch to the target address specied by the in

Return instruction

- Branch to the address contained in the link re



Subroutine linkage using a link register

Subroutine nesting and the processor stack.

Subroutine nesting – when one subroutine calls another
The return address of second call is also stored in link
saving contents of link register at other location.
Return addresses are generated and used in a LIFO o
particular register is designated as stack pointer, SP, Th
to a stack called processor stack. The Call instructio
the contents of PC onto processor stack & loads su
address into PC. The Return instruction pops the return
from the processor stack into PC.

(b) Explain any three assembler directives with example. [3] CO1 L2

Assembler Directives

Assembler directives are instructions/statements that direct assem to do something, for eg. to set aside space for variab to include additional source files or to establish start address of the program.

These statements/instructions do not execute when object program run nor they appears in the object program. It simply provides information to assembler.

```
Eg-     SUM     EQU     200
        ORIGIN          204

N       DATAWORD        100
NUM1    RESERVE         400

        RETURN
        END     START
```

PART D

7 (a)	With a neat diagram, explain registers in DMA interface. Also, explain bus arbitration approaches in DMA.	[4+6]	CO3	L2

, DMA controller registers with bus arbitration?

DIRECT MEMORY ACCESS

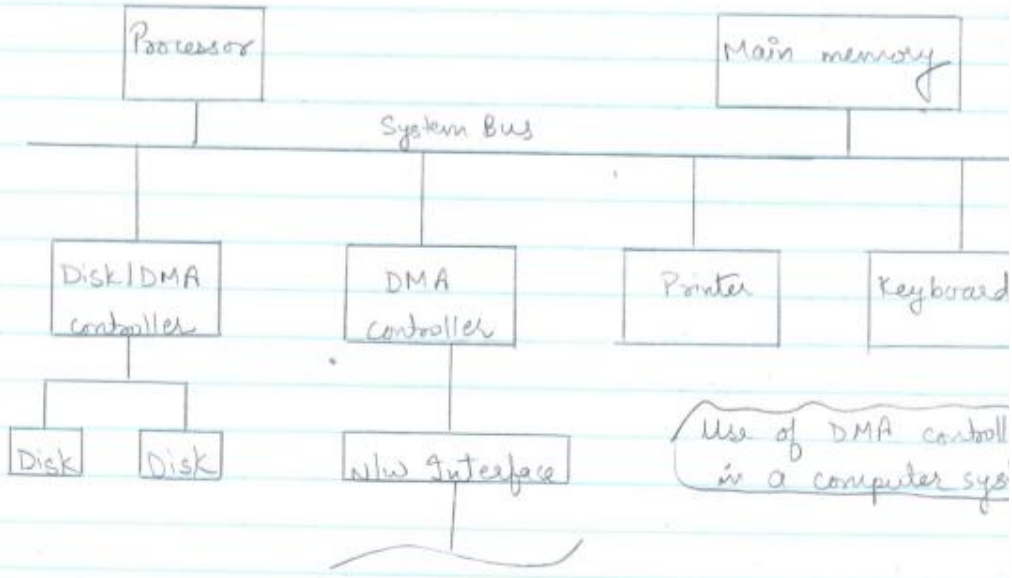
- To transfer large blocks of data at high speed, betw external devices and main memory, DMA (Direct Mem Access) approach is often used.
- DMA controller (control circuit in I/O device interface) data transfer directly between I/O device and memory, with minimal intervention of processor.
- DMA controller acts as a processor, but it is controlled
- To initiate transfer of a block of words, the process sends the following data to controller:
  - (i) The starting address of the memory block
  - (ii) The word count
  - (iii) Control to specify the mode of transfer such a or write.
  - (iv) A control to start the DMA transfer.

- DMA controller performs the requested I/O operation a sends an interrupt to the processor upon completion

	31	30		1	
Status and Control	IRQ	IE		R/$\overline{W}$	
Starting address					
word count					

Registers in a DMA interface

Bits & Flags	1	0
R/W	READ	WRITE
Done	Data transfer finishes	
IRQ	Interrupt request	
IE	Raise interrupt (enable) after Data Transfer.	



• Memory accesses by the processor and DMA Controller interwoven. DMA devices have higher priority than processor over bus control.

→ Cycle Stealing - DMA controller 'steals' memory cycl from processor, though processor originates most memo access. For each byte to be transferred, DMA interrupt processor.

→ Block or Burst mode - The DMA controller given exclus access to the main memory to transfer a block of without its interruption.

A conflict may arise if both the processor and a DMA co or two DMA controllers try to use the bus at the sa time to access the main memory. To resolve this, bus arbitration methods are required.

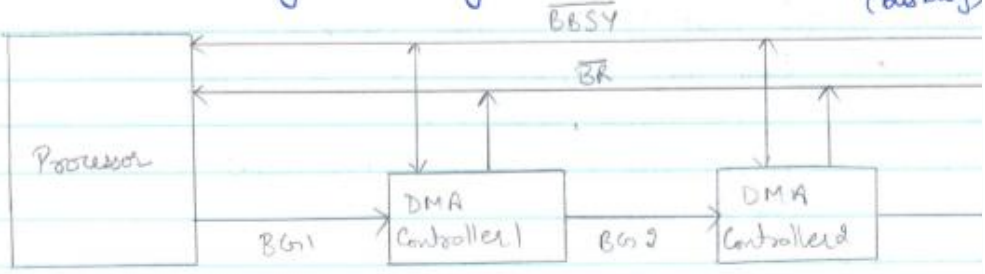
Bus Arbitration

• Bus master: device that initiates data transfers on th The next device can take control of the bus after t current master relinquishes control.
• Bus Arbitration: process by which the next dev to become master is selected.
2 main types :—

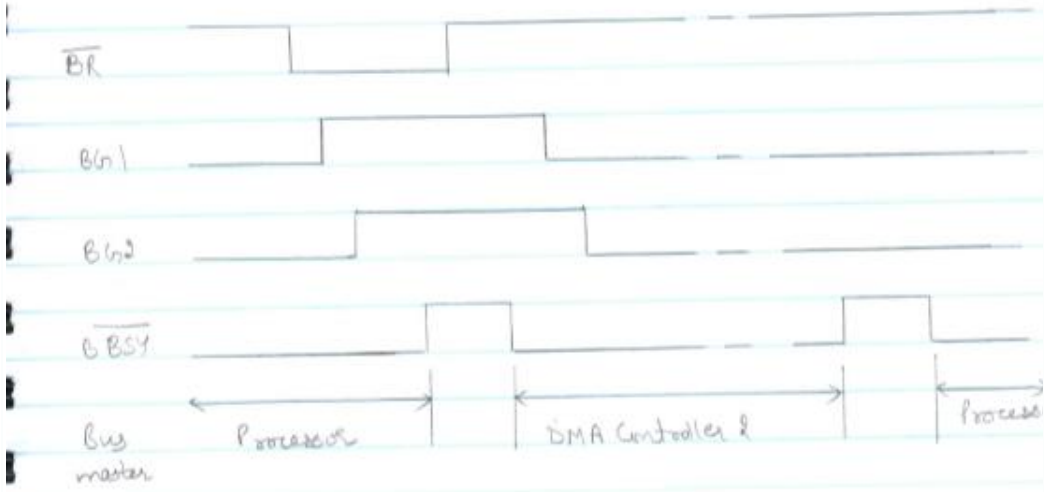
① Centralized Arbitration
② Distributed Arbitration.

- Centralized Bus Arbitration Technique

→ A 'single bus arbiter' performs the required arbitra
→ Bus arbiter may be the processor or a separate de connected to the bus (eg. DMA controller having hi priority).
→ Initially the processor will act as 'bus master'.
→ Whenever a DMA controller generates a request ($\overline{BR}$), activating Bus Request line, processor activat Bus-Grant (BG1) signal indicating DMA controller can become a master
→ BG line is connected to all DMA devices using daisy chain link.
→ If DMA controller 1 has enabled the request, it blocks signal (BG1) and acts as a master for bus arbitration thereby disabling the bus for other device use.
→ If DMA controller 1 has not enabled bus arbitration It simply forwards BG1 signal to its downstream ne DMA controllers by asserting BG2. New bus master activ (Bus Busy)



Bus arbitration using Daisy chain

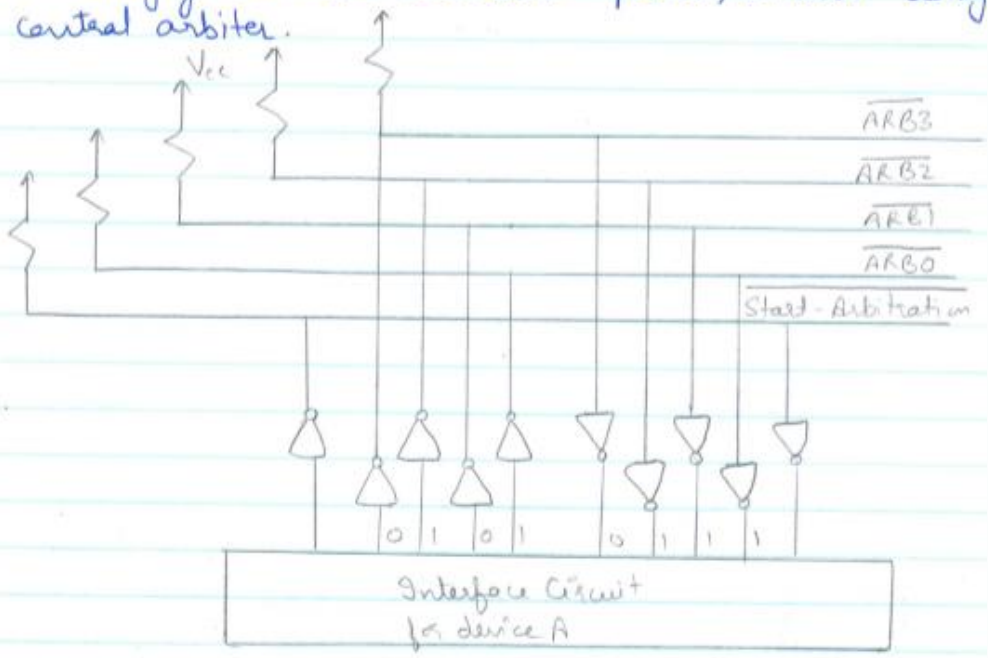


Sequence of signals for Transfer of bus mastership from to DMA controller 2

DMA controller 2 requests and acquires the bus masters the bus. At this time, it activates the bus busy to to prevent other devices from using the bus at same After it finishes its task, when bus is free again, acquires the bus if no other DMA controller request and activates Bus busy line.

Distributed Bus Arbitration

- All devices waiting to use the bus have equal res in carrying out the arbitration process, without using central arbiter.



Distributed arbitration scheme

→ Each device is identified by using a 4bit identifica number.

→ Devices start contending for bus by enabling "start- signal and place their 4-bit identification ~~number~~ the bus (4 lines $ARB_0 - ARB_3$).

→ Device having ~~has~~ highest identification number is s to get the granted service.

→ Selection procedure :

- Assume that two devices A and B have their ide numbers 5 and 6 respectively.

i.e id of A = 0101

id of B = 0110.

Device A transmits the pattern 0101 on arbitration l device B sends the pattern 0110 on arbitration li

- A code value is calculated applying "Logical OR identification numbers of contending devices.

i.e. 0101

+ 0110

0111 ← code generated.

This code generated by 'OR' operation is sent to all the contending devices.

- Each contending device, compares its own id on a lines with code value bit by bit starting from
- When it finds a mismatch in any bit place, remaining lower order bits of that device id are disabled to '0'.

device A 0101 Code 0111

mismatch

So now device A shows 0100.

device B 0110 code 0111

mismatch

- 0110 code.

now → 'OR' for new codes 0100

+0110

0110

This means device B wins the election and is chose the bus master

OR

8 (a)	Explain various methods for handling interrupts from multiple devices with neat diagram.	[10]	CO2	L2

Handling multiple devices/interrupts (polling, vectored interrupts, interrupt nesting, daisy chaining)

Handling multiple devices

- Multiple devices can initiate interrupts. They use interrupt request line.

Following techniques may be used :-

- Polling
- Vectored Interrupts
- Interrupt nesting
- Daisy chaining.

① POLLING

- The IRQ (interrupt request) bit in the status r device is set to when a device is requesting an
- The Interrupt service routine polls the I/O devic to the bus.
- The first device encountered with the IRQ bit s serviced and ISR is invoked.
- It is easy to implement, but too much time checking the IRQ of all devices, though so may not be requesting service.

② VECTORED INTERRUPTS

- Device requesting an interrupt identifies itself dire the processor.

(4 to 8 bits)

The device sends a special code to the processor bus.

- The code contains :
  identification of the device,
  starting address of ISR,
  address of the branch to ISR (if ISR not at that lo

The location pointed to by the interrupting de to store the starting address of the interru routine. This address is called interrupt vect reads it and loads it into PC.

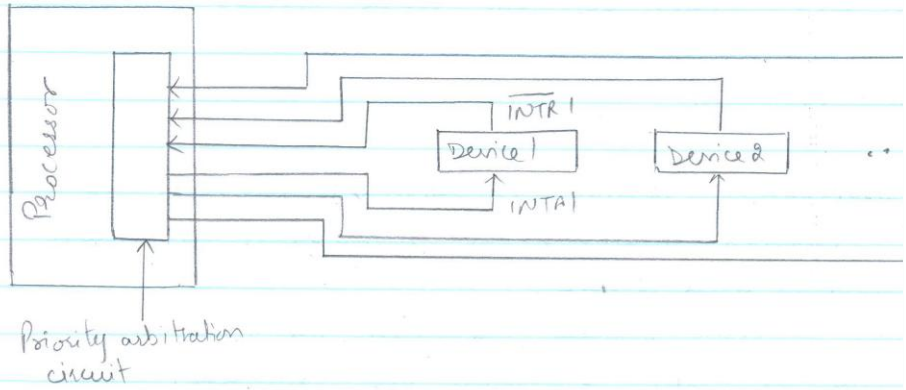
* When the processor is ready to receive in code, it may activate interrupt-acknow INTA. The I/O device responds by sending it vector code and turning off the INTR signa

② INTERRUPT NESTING

- Disabling Interrupts during execution of th not favor devices which need immediat eg, keeping track of time of day.
- Pre-emption of low priority interrupt by higher priority interrupt is known as Inte
- Only interrupts requests of higher priority accepted during execution of ISR of lower pr
- A priority level is assigned to processor which of the program that is currently being exe Only higher priority interrupts than this are a
- Processor's priority is encoded in a few bi

status word which can be changed by prog
called privileged instructions, which can be e
while processor is running in supervisor mod
executing OS routines).

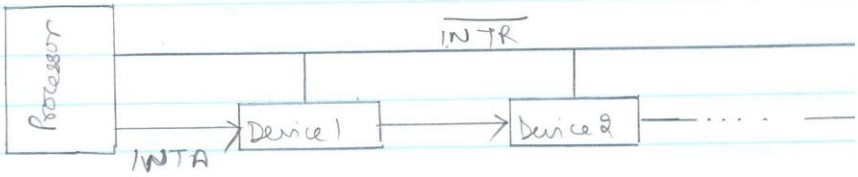
- An attempt to execute a privileged instructi
in user mode leads to a special type
called a privilege exception.
- A multiple-priority scheme can be implemented
using separate interrupt request and interru
lines for each device. Each interrupt-request li
assigned a different priority level. Interrupt requ
received over these lines are sent to a prior
arbitration circuit in the processor. A request is
only if it has a higher priority level than
assigned to the processor.



Implementation of interrupt priority using individual in
and acknowledge lines

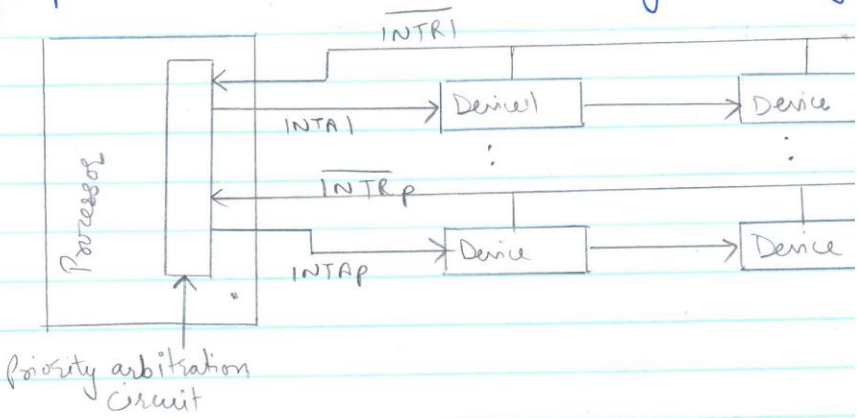
④ DAISY CHAINING

- The interrupt request line INTR is comm
  the devices.
- The interrupt acknowledgement line INTA i
  devices in a daisy chain way.
- INTA propagates serially through the device
- Device that is electrically closest to th
  gets high priority.
- Low priority device may have a danger



Daisy chaining with priority group :-

- Combining daisy chaining and interrupt vesting to
  group.
- Each group has different priority levels and with
  group devices are connected in daisy chain way



Arrangement of priority groups

PART E

9 (a)	Write about shift and rotate instruction with neat diagram and example of each.	[6] [4]	CO1	L2,L3

2) Shift and Rotate Instructions.

Shift instructions shift bits of an operand to right or left some specified number of bit positions.

Rotate instructions move the bits that are shifted out of one end of the operand back into the other end.

- Logical Shifts

Shift an operand over a number of bit positions specified in a count operand contained in the instruction. Count operand may be given as an immediate operand or in may be contained in a processor register.

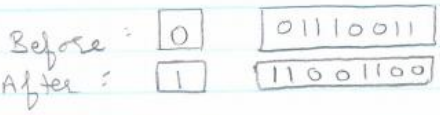
Syntax: LShiftL count, dst
LShiftR count, dst

Bits shifted out are passed through carry flag, C & then dropped.

→ Logical Shift Left

Eg-

LShiftL #2, R0

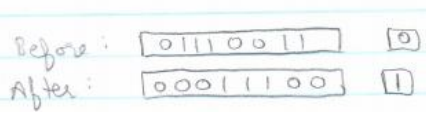


Logical Shift Left

→ Logical Shift Right.

Eg-

LShiftR #2, R0



Logical Shift Right

- Arithmetic Shifts

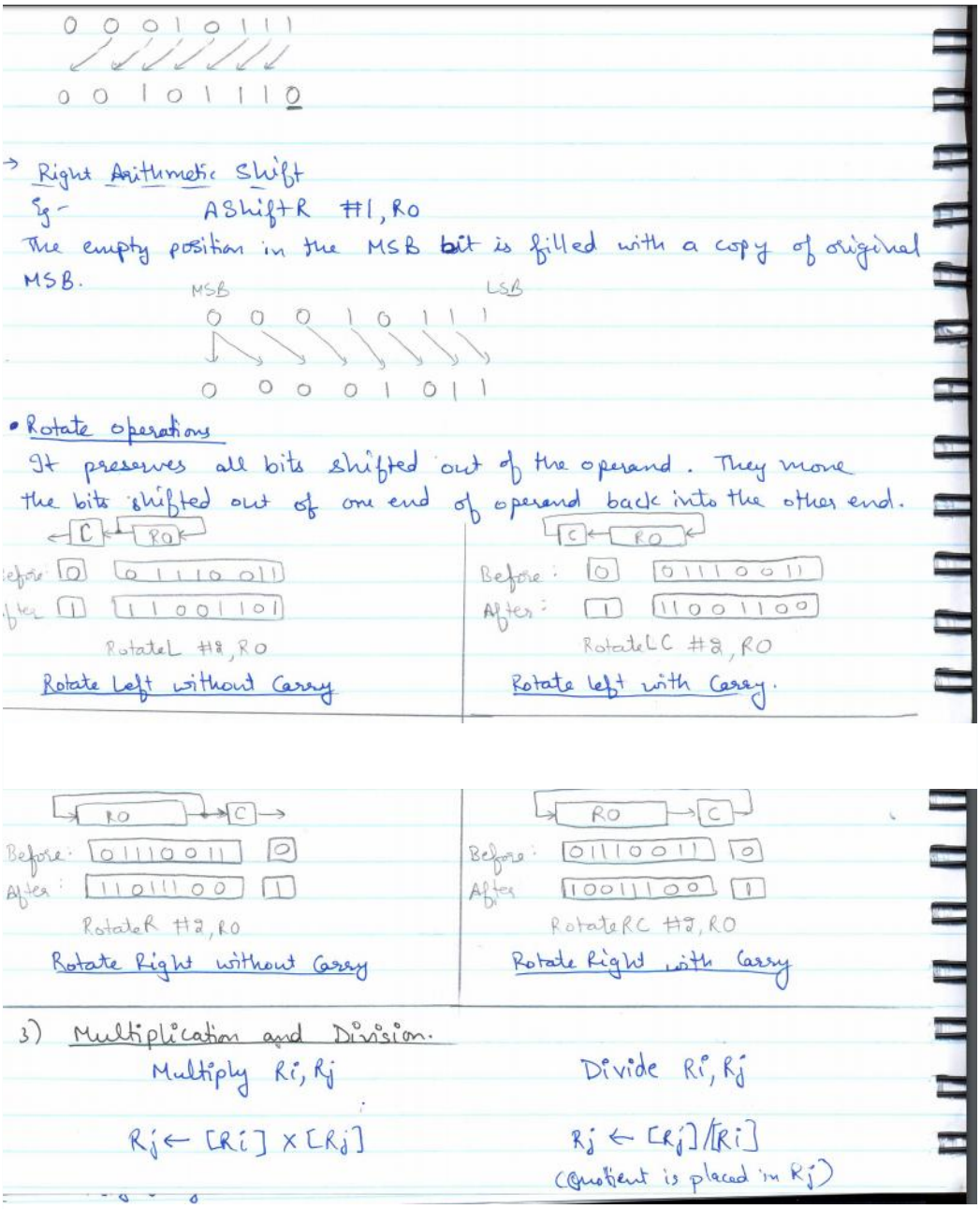
Fill-in bits are not always zero as in logical shift operations.

Syntax: AShiftL count, dst
AShiftR count, dst

→ Left Arithmetic Shift

Eg- AShiftL #1, R0

The empty position in the LSB (least significant bit) is filled with a zero.



Write a assembly program to read a character and display it.

Example :- A program that reads a line of characters and displays it.

```
        Move        #LOC, R0        Initialize pointer register R0 to
                                    point to the address of the first
                                    location in memory where characters
                                    are to be stored.
READ    TestBit     #3, INSTATUS    wait for character to be
        Branch=0    READ            entered in keyboard buffer
        MoveByte    DATAIN, (R0)    DATAIN. Transfer the character
                                    from DATAIN into memory
                                    (this clears SIN to 0)
ECHO    TestBit.    #3, OUTSTATUS   wait for display to become ready
        Branch=0    ECHO
        MoveByte    (R0), DATAOUT   Move the character just read
                                    to display buffer register. (this
                                    clears SOUT to 0)

        Compare     #CR, (R0)+      check if character just read
                                    is CR (carriage return). If
                                    not CR, then branch back &
        Branch≠0    READ            read another character.
                                    Also, increment the pointer
                                    to store next character.
```

OR

10 (a)	What is an Interrupt? With an example, illustrate the concept of interrupt.	[10]	CO2	L2

INTERRUPTS

In program-controlled I/O processor repeated device status. During this wait loop, processo performing any useful computation. There situations where other tasks can be perform waiting for the I/O device to become read device may alert the processor when it becom This alert may be sent using a hardwa called an interrupt to the processor. Ge at least one of the bus control lines, called a

Program : It consists of 2 routines.
COMPUTE - produces a set a n lines.
PRINT - send lines of output to printer a time.

without interrupts

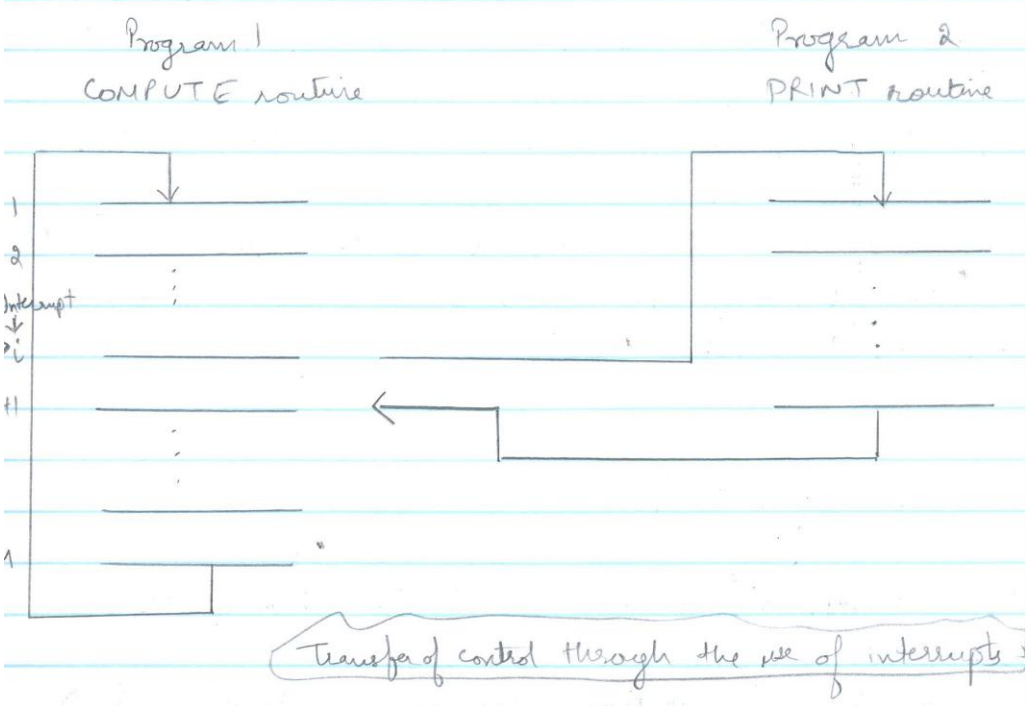
COMPUTE produces n lines.
PRINT sends 1st line
(wait for it to be printed)
send 2nd line
(wait for it to be printed)
⋮
so send n^{th} line
(wait for it to be printed)
COMPUTE produces next n lines.
PRINT sends 1st line
(wait)
send 2nd line.
(wait)
⋮
send n^{th} line
(wait)
⋮
so on.

isadvantage - Processor spends a considerable time waiting for printer to become ready.

→

With interrupts Overlapping printing and computatio
i.e., execute COMPUTE routine while printing is in

COMPUTE produces n lines
PRINT send 1st line
(suspend PRINT)
COMPUTE next n lines, printer printing
If printer ready, send ISR.
COMPUTE interrupted.
PRINT send 2nd line.
(suspend print)
COMPUTE next n lines, printer printing.
⋮
So on



Transfer of control through the use of interrupts