(For Regular Students)

COMPUTER ORGANIZATION & ARCHITECTURE

(ETCS–204)

Instruction to Paper Setters: **Maximum Marks : 75**

1. *Question No. 1 should be compulsory and cover the entire syllabus. This question should have objective or short answer type questions. It should be of 25 marks.*
2. *Apart from Question No. 1, rest of the paper shall consist of four units as per the syllabus. Every unit should have two questions. However, student may be asked to attempt only 1 question from each unit. Each question should be 12.5 marks.*

UNIT- I : COMPUTER ARITHMETIC AND REGISTER TRANSFER LANGUAGE

Unsigned notation, signed notation, binary coded decimal, floating point numbers, IEEE 754 floating point standard, Micro-operation, Bus and Memory Transfers, Bus Architecture, Bus Arbitration, Arithmetic Logic, Shift Micro operation, Arithmetic Logic Shift Unit. **[T1,T2] [No. of hrs. 11]**

UNIT- II : INSTRUCTION SET ARCHITECTURE & COMPUTER ORGANIZATION

Levels of programming languages. assembly language instructions, 8085 instruction set architecture, Instruction Codes, Computer Registers, Computer Instructions, Timing & Control, Instruction Cycle, Memory Reference Instructions, Input-Output and Interrupts. **[T1,T2] [No. of hrs. 11]**

UNIT- III : CONTROL DESIGN

Instruction sequencing & interpretation, Hardwired & Micro Programmed (Control Unit), Micrprogrammed computers, Microcoded CPU: Pentium processor. Specifying a CPU, Design & implementation of simple CPU, General Register Organization, Stack Organization, Instruction Formats, Addressing Modes, Internal architecture of 8085 microprocessor. **[T1,T2] [No. of hrs. 11]**

UNIT- IV

Memory & Input/Output organization: Memory Technology, Main Memory (RAM and ROM Chips), Virtual memory, High-speed memories Asynchronous Data Transfers, Programmed I/O, interrupts, Direct memory Access, Serial communication, UARTs, **RS-232-C & RS-422** standard. **[T1,T2] [No. of hrs. 11]**

Rs. 54.00/-

AB Publishers

4278/3 ANSARI ROAD DARYA GANJ NEW DELHI-02

SYLLABUS

COMPUTER ARCHITECTURE (ETCS-305)

INSTRUCTIONS TO PAPER SETTERS: **MAXIMUM MARKS: 75**

1. Question No. 1 should be compulsory and cover the entire syllabus. This question should have objective or short answer type questions. It should be of 25 marks.
2. Apart from question no. 1, rest of the paper shall consist of four units as per the syllabus. Every unit should have two questions. However, student may be asked to attempt only 1 question from each unit. Each question should be of 12.5 marks.

UNIT I

Introduction and overview: Review of digital components, Evolution of computers.

Register Transfer and Microoperation: Register transfer language, register transfer, bus and memory transfer, arithmetic microoperations, logic microoperations, shift microoperations.

Basic Computer Organization and Design: Instruction codes, computer registers, computer instructions, timing & control, instruction cycle, memory reference instructions, input-output and interrupts, design of basic computer, design of accumulator logic.

[No. of Hrs: 11]

UNIT II

Microprogrammed Control Unit: Control memory, address sequencing.

Central Processing Unit: Introduction, general register organization, stack organization, instruction formats, addressing modes.

Pipeline and vector processing Parallel Processing, pipelining, arithmetic pipeline, RISC Pipeline, Vector Processing, Array Processors. **[No. of Hrs: 11]**

UNIT III

Computer Arithmetic: Introduction, addition and subtraction, multiplication algorithms, division algorithms, floating point arithmetic operation, decimal arithmetic unit, decimal arithmetic operations.

Input-Output Organization: Peripheral devices, input-output interface, asynchronous data transfer, modes of data transfer, priority interrupt, direct memory access, input-output processor. **[No. of Hrs: 11]**

UNIT IV

Memory organization: Memory hierarchy, main memory, auxiliary memory, associative memory, cache memory, virtual memory, memory management hardware.

Multiprocessors: Characteristics of multiprocessor, Interconnection Structure, Interprocessor Communication & Synchronization. **[No. of Hrs: 11]**

SYLLABUS

COMPUTER ARCHITECTURE (ETCS–305)

INSTRUCTIONS TO PAPER SETTERS: **MAXIMUM MARKS: 75**

1. Question No. 1 should be compulsory and cover the entire syllabus. This question should have objective or short answer type questions. It should be of 25 marks.
2. Apart from question no. 1, rest of the paper shall consist of four units as per the syllabus. Every unit should have two questions. However, student may be asked to attempt only 1 question from each unit. Each question should be of 12.5 marks.

UNIT I

Introduction and overview: Review of digital components, Evolution of computers.

Register Transfer and Microoperation: Register transfer language, register transfer, bus and memory transfer, arithmetic microoperations, logic microoperations, shift microoperations.

Basic Computer Organization and Design: Instruction codes, computer registers, computer instructions, timing & control, instruction cycle, memory reference instructions, input-output and interrupts, design of basic computer, design of accumulator logic.

[No. of Hrs: 11]

UNIT II

Microprogrammed Control Unit: Control memory, address sequencing.

Central Processing Unit: Introduction, general register organization, stack organization, instruction formats, addressing modes.

Pipeline and vector processing Parallel Processing, pipelining, arithmetic pipeline, RISC Pipeline, Vector Processing, Array Processors. **[No. of Hrs: 11]**

UNIT III

Computer Arithmetic: Introduction, addition and subtraction, multiplication algorithms, division algorithms, floating point arithmetic operation, decimal arithmetic unit, decimal arithmetic operations.

Input-Output Organization: Peripheral devices, input-output interface, asynchronous data transfer, modes of data transfer, priority interrupt, direct memory access, input-output processor. **[No. of Hrs: 11]**

UNIT IV

Memory organization: Memory hierarchy, main memory, auxiliary memory, associative memory, cache memory, virtual memory, memory management hardware.

Multiprocessors: Characteristics of multiprocessor, Interconnection Structure, Interprocessor Communication & Synchronization. **[No. of Hrs: 11]**

MODEL TEST PAPER-I
FIRST TERM EXAMINATION
FOURTH SEMESTER (B.TECH)
COMPUTER ORGANIZATION & ARCHITECTURE–[ETCS-204]

Time : 1½ hrs. **M.M. : 30**

Note: *Q. One is Compulsory. Attempt any two from the rest.*

Q.1.(a) Explain the overflow condition in arithmetic shift micro operation.(3)

Ans. An arithmetic shift is a micro operation that shifts a signed binary number to the left or right. An arithmetic shift-left multiplies a signed binary number by 2. An arithmetic shift-right divides the number by 2.The arithmetic shift-left inserts a 0 into R_0, and shifts all other bits to the left. An overflow occurs after an arithmetic shift left if initially, before the shift, R_{n-1} is not equal to R_{n-2}. An overflow flip-flop V_s can be used to detect arithmetic left shift overflow.

$$V_s = R_{n-1} \oplus R_{n-2}$$

If $V_s = 0$, there is no overflow, but if $V_s=1$ there is an overflow and a sign reversal after the shift. Vs must be transferred into the overflow flip-flop with the same cock pulse that shifts the register.

Q.1.(b) How many bits wide memory address have to be if the computer had 16 MB of memory? (Use the smallest value possible). **(5)**

Ans. For a system having 1024 kb = 2^{10} means 10 bits memory address.

For 16 mb =16*1024*1024 bytes = 16777216 = $2^4 + 2^{10} + 2^{10} = 2^{24}$

I.e. we have 24 bits wide memory address if we have 16 MB of memory space.

Q.1.(c) What are the two instructions needed in the basic computer in order to set E flip-flop to 1? **(2)**

Ans. (*i*) CLE clear E

(*ii*) CME complement E.

Q.2. Define IEEE 754 floating point standard. **(10)**

Ans. The **IEEE Standard for Floating-Point Arithmetic (IEEE 754)** is a technical standard for floating-point computation established in 1985 by the Institute of Electrical and Electronics Engineers (IEEE).

The standard defines:

- **Arithmetic formats:** sets of binary and decimal floating-point data, which consist of finite numbers (including signed zeros and subnormal numbers), infinities, and special "not a number" values (NaNs)
- **Interchange formats:** encodings (bit strings) that may be used to exchange floating-point data in an efficient and compact form
- **Rounding rules:** properties to be satisfied when rounding numbers during arithmetic and conversions
- **Operations:** arithmetic and other operations on arithmetic formats
- **Exception handling:** indications of exceptional conditions (such as division by zero, overflow, *etc.*)

The standard also includes extensive recommendations for advanced exception handling, additional operations (such as trigonometric functions), expression evaluation, and for achieving reproducible results.

The standard is derived from and replaces IEEE 754-1985, the previous version, following a seven-year revision process, chaired by Dan Zuras and edited by Mike Cowlishaw. The binary formats in the original standard are included in the new standard along with three new basic formats (one binary and two decimal). To conform to the current standard, an implementation must implement at least one of the basic formats as both an arithmetic format and an interchange format.

An IEEE 754 *format* is a "set of representations of numerical values and symbols". A format may also include how the set is encoded.

A format comprises:

• Finite numbers, which may be either base 2 (binary) or base 10 (decimal). Each finite number is described by three integers: s = a *sign* (zero or one), c = a *significant* (or 'coefficient'), q = an *exponent*. The numerical value of a finite number is

$$(-1)^s \times c \times b^q$$

where b is the base (2 or 10), also called *radix*. For example, if the base is 10, the sign is 1 (indicating negative), the significand is 12345, and the exponent is –3, then the value of the number is -12.345.

• Two infinities: $+\infty$ and $-\infty$.

• Two kinds of NaN: a quiet NaN (qNaN) and a signaling NaN (sNaN). A NaN may carry a *payload* that is intended for diagnostic information indicating the source of the NaN. The sign of a NaN has no meaning, but it may be predictable in some circumstances.

The possible finite values that can be represented in a format are determined by the base b, the number of digits in the significand (precision p), and the exponent parameter *emax*:

• c must be an integer in the range zero through $b^p - 1$ (*e.g.*, if b=10 and p=7 then c is 0 through 9999999)

• q must be an integer such that $1 - emax \le q+p-1 \le emax$ (*e.g.*, if p=7 and $emax$=96 then q is – 101 through 90).

Hence (for the example parameters) the smallest non-zero positive number that can be represented is 1×10^{-101} and the largest is 9999999×10^{90} (9.999999×10^{96}), and the full range of numbers is -9.999999×10^{96} through 9.999999×10^{96}. The numbers $-b^{1-emax}$ and b^{1-emax} (here, -1×10^{-95} and 1×10^{-95}) are the smallest (in magnitude) *normal numbers*; non-zero numbers between these smallest numbers are called subnormal numbers.

Zero values are finite values with significand 0. These are signed zeros, the sign bit specifies if a zero is +0 (positive zero) or –0 (negative zero).

Q.3. Explain bus arbitration. **(10)**

Ans. There are always **3 sets of wires (= bus)** in a computer:

• **Address bus** use to convey the **value of the address**

• **Data bus** use to convey the **information (data)** sent between the devices

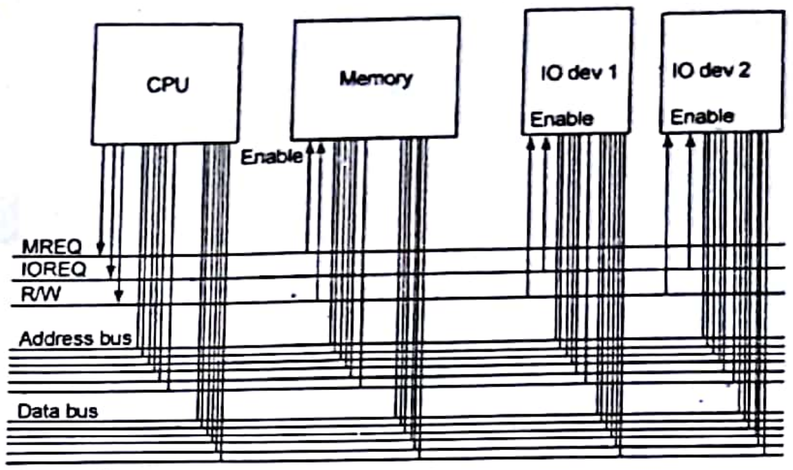
• The **control bus** use for **control signals** (such as **MREQ, Read/Write,** etc.)

The collection of **address bus, data baus** and **control bus** is called the **system bus**

• The System Bus is a bunch of **electrical paths** that are used to convey electrical signals between various devices inside the computer.

Important fact:

• A bus (any bus) **cannot** be used by **more than one device** at one time due to electrical properties of the devices (the **outputs** will be **connected together** and will cause damage).



Solution: arbbitration

• Before any device is allowed to perform a rea/write operation using the system bus, it must first **obtain permission**

A device that **starts a read/write operation** is called a **master device**

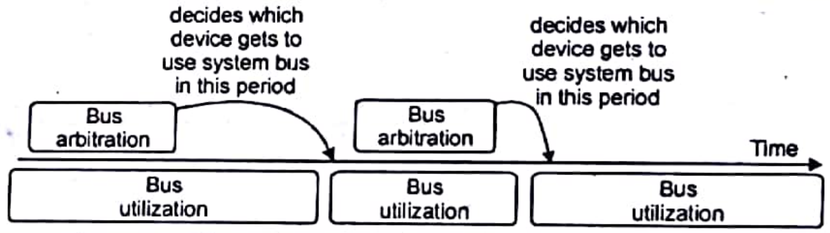
The device that it "talks" to is called the **slave device**

• There is always a **bus arbitration** period **proceeded by a bus utilization** period:

• To **speed up** the computer, the two operations (bus arbitration and bus utilization) can be and are always performed in parallel:

• While the current bus utilization period is going on, devices that want to use the bus in the next cycle can decide among themselves who will get to use the system bus in the next period.

Therefore, **bus arbitration** periods and **bus utilization** periods usually overlap:



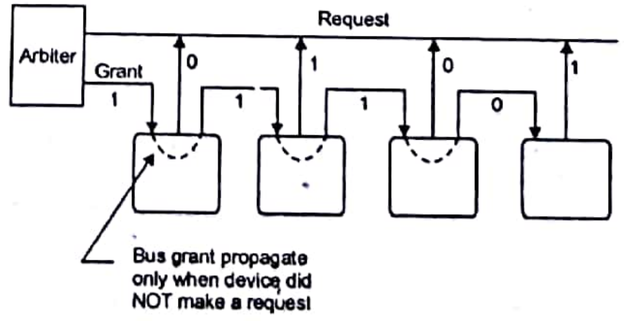
• There are 2 types of bus arbitration techniques:

→ **Centralized** (commonly used)

→ **Distributed (rarely used** - only instance I know is DEC).

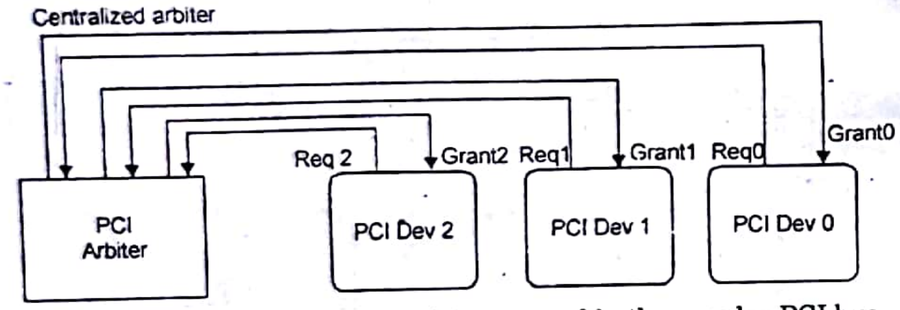
• **Example centralized** aribitration techniques:

Daisy-chained (cheap - requires no special hardware)



- The daisy chain consists of wires connecting the devices in a pre-defined order
- This ordering defines the "pecking order" of the devices.

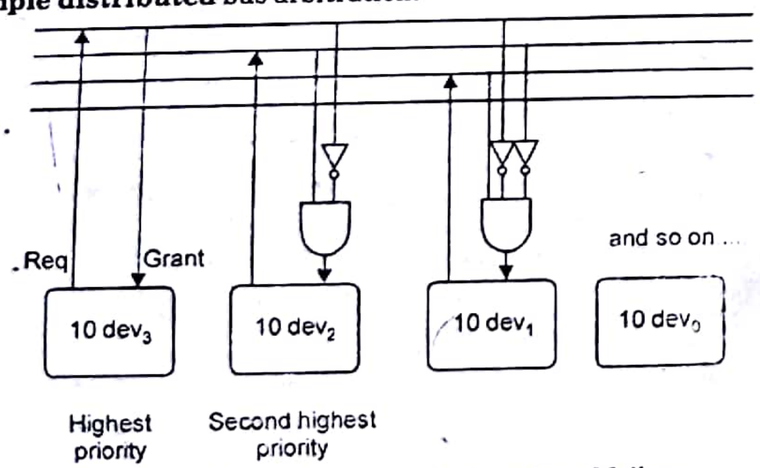
→ **Explicit centralized bus arbiter (used in PCI bus):**



- Such an explicit centralized bus arbiter is used in the popular PCI bus.
- The logic of the bus arbiter is simple and and depends on how the priorities are assigned. The following table shows the logic values when the highest numbered device receives the highest priority:

Req2	Req1	Req0	Grant2	Grant2	Grant0
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	0
1	0	0	1	0	0
1	0	1	1	0	0
1	1	0	1	0	0
1	1	1	1	0	0

- **Example distributed** bus arbitration:



Advantage of distributed bus arbiter: no single point of failure...

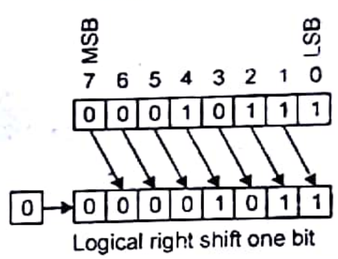
Q.4.(a) Explain shift micro operation in detail. Also draw and explain 4-bit combinational circuit shifter. **(10)**

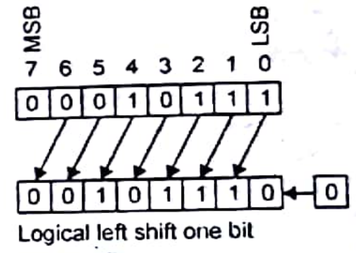
Ans. Shift micro operations are used for serial transfer of data. They are also used in conjunction with arithmetic, logic and other data – processing operations. The contents of the register can be shifted to the left or right. At the same time that the bits are shifted, the first flip-flop receives its binary information from the serial input. During

a shift right operation the serial input transfers a bit into the leftmost position. The information transferred through the serial input determines the type of shift.

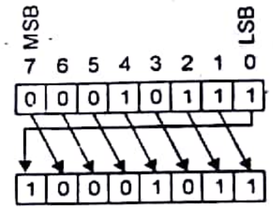
Logical Shift: a logical shift is a bitwise operation that shifts all the bits of its operand. Unlike an arithmetic shift, a logical shift does not preserve a number's sign bit or distinguish a number's exponent from its mantissa; every bit in the operand is simply moved a given number of bit positions, and the vacant bit-positions are filled in, usually with zeros (compare with a circular shift).

A logical shift is often used when its operand is being treated as a sequence of bits rather than as a number.

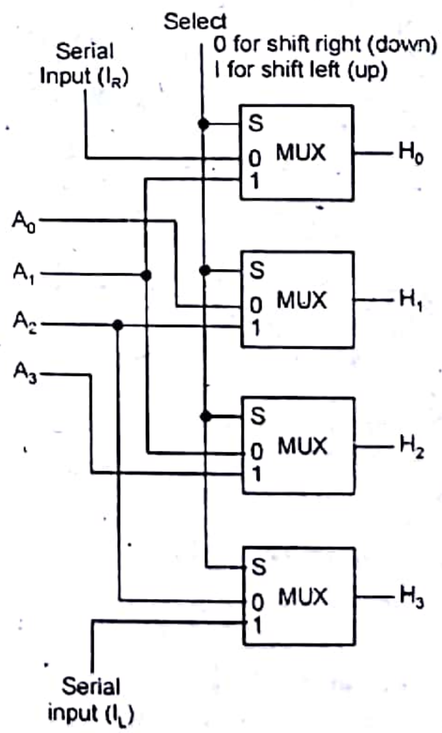




Circular Shift: A circular shift is the operation of rearranging the entries in a tuple, either by moving the final entry to the first position, while shifting all other entries to the next position, or by performing the inverse operation.



Arithmetic Shift: An arithmetic shift is a shift operator, sometimes known as a signed shift (though it is not restricted to signed operands). For binary numbers it is a bitwise operation that shifts all of the bits of its operand; every bit in the operand is simply moved a given number of bit positions, and the vacant bit-positions are filled in. Instead of being filled with all 0s, as in logical shift, when shifting to the right, the leftmost bit (usually the sign bit in signed integer representations) is replicated to fill in all the vacant positions (this is a kind of sign extension). Arithmetic shifts can be useful as efficient ways of performing multiplication or division of signed integers by powers of two.



Function table

Select	Output			
S	H_0	H_1	H_2	H_3
0	I_R	A_0	A_1	A_2
1	A_1	A_2	A_3	I_L

MODEL TEST PAPER-I
SECOND TERM EXAMINATION
FOURTH SEMESTER (B.TECH)
COMPUTER ORGANIZATION & ARCHITECTURE–[ETCS-204]

Time : 1¼ hrs. **M.M. : 30**

Note: *All questions are compulsory.*

Q.1. Explain 8085 data transfer instructions. **(15)**

Ans. DATA TRANSFER INSTRUCTIONS

Opcode Operand Description

Copy from source to destination

MOV Rd, Rs This instruction copies the contents of the source M, Rs register into the destination register; the contents of Rd, M the source register are not altered. If one of the operands is a memory location, its location is specified by the contents of the HL registers.

Example: MOV B, C or MOV B, M

Move immediate 8-bit

MVI Rd, data The 8-bit data is stored in the destination register or M, data memory. If the operand is a memory location, its location is specified by the contents of the HL registers.

Example: MVI B, 57H or MVI M, 57H

Load accumulator

LDA 16-bit address The contents of a memory location, specified by a 16-bit address in the operand, are copied to the accumulator.

The contents of the source are not altered.

Example: LDA 2034H

Load accumulator indirect

LDAX B/D Reg. pair The contents of the designated register pair point to a memory location. This instruction copies the contents of that memory location into the accumulator. The contents of either the register pair or the memory location are not altered.

Example: LDAX B

Load register pair immediate LXI Reg. pair, 16-bit data The instruction loads 16-bit data in the register pair designated in the operand.

Example: LXI H, 2034H or LXI H, XYZ

Load H and L registers direct LHLD 16-bit address The instruction copies the contents of the memory location pointed out by the 16-bit address into register L and copies the contents of the next memory location into register H. The contents of source memory locations are not altered.

Example: LHLD 2040H

Store accumulator direct STA 16-bit address The contents of the accumulator are copied into the memory location specified by the operand. This is a 3-byte instruction, the second byte specifies the low-order address and the third byte specifies the high-order address.

Example: STA 4350H

Store accumulator indirect STAX Reg. pair The contents of the accumulator are copied into the memory location specified by the contents of the operand (register pair). The contents of the accumulator are not altered.

Example: STAX B

Store H and L registers direct SHLD 16-bit address The contents of register L are stored into the memory location specified by the 16-bit address in the operand and the contents of H register are stored into the next memory location by incrementing the operand. The contents of registers HL are not altered. This is a 3-byte instruction, the second byte specifies the low-order address and the third byte specifies the high-order address.

Example: SHLD 2470H

Exchange H and L with D and E

XCHG none The contents of register H are exchanged with the contents of register D, and the contents of register L are exchanged with the contents of register E.

Example: XCHG

Copy H and L registers to the stack pointer SPHL none The instruction loads the contents of the H and L registers into the stack pointer register, the contents of the H register provide the high-order address and the contents of the L register provide the low-order address. The contents of the H and L registers are not altered.

Example: SPHL

Exchange H and L with top of stack XTHL none The contents of the L register are exchanged with the stack location pointed out by the contents of the stack pointer register. The contents of the H register are exchanged with the next stack location (SP+1); however, the contents of the stack pointer register are not altered.

Example: XTHL

Push register pair onto stack PUSH Reg. pair The contents of the register pair designated in the operand are copied onto the stack in the following sequence. The stack pointer register is decremented and the contents of the high order register (B, D, H, A) are copied into that location. The stack pointer register is decremented again and the contents of the low-order register (C, E, L, flags) are copied to that location.

Example: PUSH B or PUSH A

Pop off stack to register pair POP Reg. pair The contents of the memory location pointed out by the stack pointer register are copied to the low-order register (C, E, L,status flags) of the operand. The stack pointer is incremented by 1 and the contents of that memory location are copied to the high-order register (B, D, H, A) of the operand. The stack pointer register is again incremented by 1.

Example: POP H or POP A

Output data from accumulator to a port with 8-bit address OUT 8-bit port address The contents of the accumulator are copied into the I/O port specified by the operand.

Example: OUT F8H

Input data to accumulator from a port with 8-bit address

IN 8-bit port address The contents of the input port designated in the operand are read and loaded into the accumulator.

Example: IN 8CH

Q.2. What is the basic advantage of using interrupt initiated data transfer over transfer under program control without an interrupt? **(5)**

Ans. This mode of transfer uses the interrupt facility. While the flag is set, the computer is momentarily interrupted from proceeding with the current program and is informed of the fact that the flag has been set. An alternative to the CPU constantly monitoring the flag is to let the interface inform the computer when it is ready to transfer data. This mode of transfer uses the interrupt facility. While the CPU is running a program, it does not check the flag. However, when the flag is set, the computer is momentarily interrupted from proceeding with the current program and is informed

of the fact that the flag has been set. The CPU deviates from what it is doing to take care of the input or output transfer is completed, the computer returns to the previous program top continue what it was doing before the interrupt. The CPU responds to the interrupted signal by storing the return address from the program counter into a memory stack and then control branches to a service routine that processes the required I/O transfer. Interrupt driven data transfer is used for interlacing relatively slow I/O devices, such character printer, or an A/D converter to the µp. The idea underlying this method can be described as follows. The processor first initiates the I/O device.

Having initiated the device, the processor continues the execution of subsequent instructions in the program. Transfer of data under programmed I/O is between CPU and peripheral. In direct memory access, the interface transfers data into and out of the memory unit through the memory unit through the memory bus. The CPU initiates the transfer by supplying the interface with the starting address and the number of words needed to be transferred and then proceeds to execute other tasks. When the transfer is made, the DMA requests memory cycles through the memory bus. When the request is granted by the memory controller, the DMA transfers the data directly into the memory.

Q.3. What is asynchronous data transfer? Explain in detail. **(10)**

Ans. An asynchronous data transmission involves a mechanism called a queue. In general, a queue is a service which temporarily holds messages destined for a receiving task until that task is ready to process them. The sending task passes messages off to the queue and does not wait for a response from the receiver. The queue guarantees that if it accepts a message then that message will be delivered. Cogent's asynchronous messaging is implemented through a queue administrator, called Cascade Queue Server, or qserve. Asynchronous transmission uses start and stop bits to signify the beginning bit ASCII character would actually be transmitted using 10 bits. For example, "0100 0001" would become "1 0100 0001 0". The extra one (or zero, depending on parity bit) at the start and end of the transmission tells the receiver first that a character is coming and secondly that the character has ended. This method of transmission is used when data are sent intermittently as opposed to in a solid stream. In the previous example the start and stop bits are in bold. The start and stop bits must be of opposite polarity. This allows the receiver to recognize when the second packet of information is being sent.

It should be noted that the queue administrator never initiates a message transaction, and due to the nature of the send/receive/reply mechanism it will never block while transmitting a reply. Thus a queue administrator in this scenario will always be available to receive messages and the originator of an asynchronous message will never block.

There are a number of disadvantages to using asynchronous message passing. The sender and receiver in an asynchronous transaction must agree upon a queue name in order for the queue administrator to correctly route a message. This requires either hard-coded queue names, command-line arguments, or a queue-aware name server so tasks can discover the destination queue names. The Cascade Data Hub uses a queue-aware name server called nserve to provide maximum flexibility in asynchronous connections.

A second concern is that because asynchronous transmission introduces additional message passing overhead into the transmission mechanism, the speed of asynchronous transmission will tend to be slower than that of synchronous data transfer. Perhaps the biggest drawback of the asynchronous transmission method is that there must be a method of dealing with overflows in the queue. If the receiving task fails to collect its messages from the queue administrator, or if it cannot keep up with the rate at which messages is being sent to the queue administrator, then eventually messages will build up and the queue administrator will run out of buffer space. At this point, the queue administrator must refuse further incoming messages until the receiver has cleared

some of its pending messages. The sending task must include a contingency plan for undeliverable messages.

A sending task may react in several ways to a full queue situation:

1. Throw away data that cannot be put on the queue. This is both the most common and least acceptable response to a full queue. The most recent data will be lost, and old data currently on the queue will be retained. Once the condition causing the full queue has been cleared, the most recent data will be unavailable. In the case of process control systems, this scenario will fail to transmit the most recent process changes. Obviously, this method is not attractive when you consider the potential loss of state change information and critical alarm data.

2. Throw away the oldest data on the queue. This method is only viable if the queue is held internally to the sending task. In most cases, a task cannot walk the queue as it is provided by an external program or operating system facility. Even if the queue is available to the sender, there is no guarantee that the oldest data is the least important. In general, eliminating data solely on the basis of age is unacceptable. (In the case of the QNX queue administrator, Mqueue, this method of dealing with a full queue is not an option, as the queue is held externally to the sender.)

3. Throw away the oldest duplicate data on the queue. As above, this option is only available if the queue is internal to the sender. In addition, it implies that the data in the queue is of a known type and format, and can be examined for its similarity to new data. In the case of process control points, this is generally true, though it means that every point being transmitted must be compared to every queued message to determine whether it should replace a current message or go to the end of the queue. Depending on the implementation, this mechanism may not preserve time ordering in the queued data. The Cascade Data Hub uses a generalized external queue mechanism, and so does not provide this type of data reduction.

4. Try to send the data again later. If a task cannot deliver a message to the queue immediately, then it has the option of holding the message for later transmission. Once space is available in the queue, the message can be re-transmitted. This method actually means that the sender is implementing its own internal queue, duplicating the work of the queue facility, and so is subject to all of the same concerns regarding full buffers and data reduction. In general, this approach by itself is entirely useless, and is exactly equivalent to enlarging the size of the queue administrator's buffers. When used in combination with one of the other methods mentioned here, this approach could be very effective.

5. Try to send the data again later, throwing away old duplicate data from the retry queue only. This is the method used by the Cascade Data Hub to deal with undeliverable data. So long as the receiver is capable of keeping up with the data transmission rate, all point changes are deliverable. If the receiver allows its queue to fill, then the Cascade Data Hub will flag the point as pending delivery to the recipient and will attempt to deliver the most recent value for that point as soon as there is room in the queue. If another change to a pending point is received by the Data Hub, then the previous value is overwritten, so that when room in the queue becomes available, the new value is transmitted. In addition, the Cascade Data Hub packages many point changes into a single message whenever there are several pending points to a task, thereby increasing the bandwidth of the queue administrator during periods of high load. In this way, the Cascade Data Hub will transmit all point changes to any receiver that can handle the data rate, and guarantees that even slow receivers (such as GUI applications and applications on dial-up lines) will always receive the most recent point values even if they miss some intermediate values.

MODEL TEST PAPER-I
END TERM EXAMINATION
FOURTH SEMESTER (B.TECH)
COMPUTER ORGANISATION & ARCHITECTURE–[ETCS-204]

Time : 3 hrs. **M.M. : 70**

Note: *Attempt all questions (internal choice if indicated).*

Q.1.(a) Explain the difference between hard wired control and micro programmed control. Is it possible to have a hardwired control associated with a control memory? **(3)**

Ans.

Micro Programmed Control	Hardwired Control
• Micro programmed control is a control mechanism to generate control signals by using a memory called control storage (CS), which contains the control signals. • The controller causes instructions to be executed by issuing a specific set of control signals at each beat of the system clock. Each set of control signals issued causes one basic operation (micro-operation), such as a register transfer, to occur within the data path section of the computer.	• Hardwired control is a control mechanism to generate control signals by using appropriate finite state machine (FSM). • Hardwired control does not contain a control memory

Hardwired control does not contain a control memory.

Q.1.(b) A computer uses a memory unit with 256k words of 32 bits each. A binary instruction code is stored in one word of memory. The instruction has 4 parts: an indirect bit, an operation cycle, a register code part to specify one of the 64 registers and an address part. **(10)**

(i) How many bits are there in the operation code and register code part and address part?

(ii) Draw the instruction word format and indicate the number of bits in each part?

(iii) How many bits are there in the data & address inputs of the memory?

Ans.

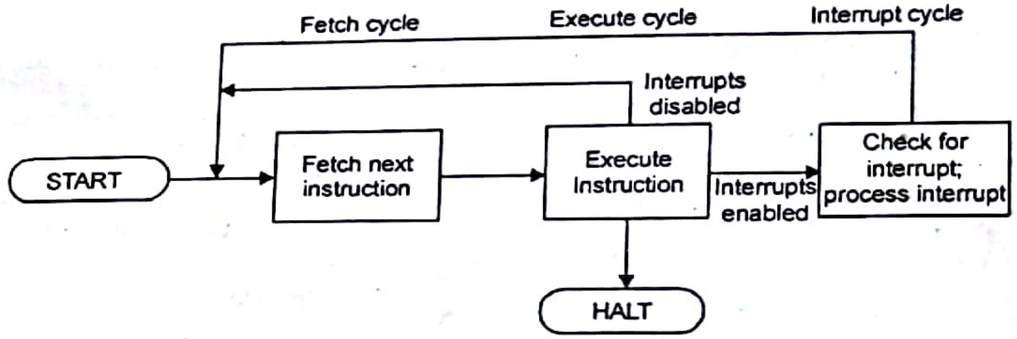
$$256k = 2^8 * 2^{10} = 2^{18}$$
$$64 = 2^6$$

(a) Address : 18 bits
Register code : 6 bits
Indirect bit : 1 bit.
32-25=7 bits for op code.

Q.1. (c) Draw and explain flow chart of program interrupt cycle. (10)

Ans. It provides several ways of suspending or terminating program execution. A program can be put on hold until a specific condition becomes TRUE using the WAIT

instruction. A program can be put on hold for a specified time period or until an event is generated in another task by the WAIT.EVENT instruction. A program can be interrupted based on a state transition of a digital input signal with the REACT and REACTI instructions. Program errors can be intercepted and handled with a REACTE instruction. Program execution can be terminated with the HALT, STOP, and PAUSE commands. These instructions interrupt the program in which they are contained. Any programs running as other tasks are not affected. Robot motion can be controlled with the BRAKE, BREAK, and DELAY instructions. (The ABORT and PROCEED monitor commands can also be used to suspend and proceed programs).



Interrupt Cycle

- Added to instruction cycle
- Processor checks for interrupt
- Indicated by presence of an interrupt signal
- If no interrupt, fetch next instruction
- If interrupt pending:
- Suspend execution of current program.
- Save context. i.e. save the address of next instruction to be executed (in current program) and any other data relevant to processor's current activity.
- Set PC to start address of interrupt handler routine
- Process interrupt
- Restore context and continue interrupted program

Q.1.(d) Differentiate between a direct and indirect addressing mode with the help of examples. **(2)**

Ans. Direct addressing mode means the operand address is contained in the instruction. In the 8085, an example is LDA 1234H, which loads the accumulator with the contents of memory location 1234H. In the 8086/8088, an example is MOV AL,[1234H], which accomplishes nearly the same thing. (AL instead of A)

Load	Reg	Address	.....

(Effective address = address as given in instruction)

This requires space in an instruction for quite a large address.

Indirect addressing mode means the operand address is contained in a register. In the 8085, an example is LDAX B, which loads the accumulator with the contents of the memory location specified in register BC. In the 8086/8088, an example is MOV AL,[BX], which accomplishes nearly the same thing. (AL instead of A and BX instead of BC)

Load	Reg 1	Base

(Effective address = contents of base register)

A few computers have this as a distinct addressing mode. Many computers just use *base plus offset* with an offset value of 0.

Q.2. Difference between Rs 232-c and Rs 422 standard. (5)

What are the primary differences between RS-232, RS-422, and RS-485 serial interfaces?

Ans. RS-232 is the most common serial interface and ships as a standard component on most Windows-compatible desktop computers. RS-232 only allows for one transmitter and one receiver on each line. RS-232 also uses a Full-Duplex transmission method. Some RS-232 boards sold by National Instruments support baud rates up to 1 Mbit/s, but most devices are limited to 115.2 kbits/s. Note that RS-422/RS-485 interface is *not* available on most IBM PCs.

RS-422 (EIA RS-422-A Standard) is the serial connection used on Apple computers. It provides a mechanism for transmitting data up to 10 Mbits/s. RS-422 sends each signal using two wires to increase the maximum baud rate and cable length. RS-422 is also specified for multi-drop applications where only one transmitter is connected to, and transmits on, a bus of up to 10 receivers.

Both protocols have multidrop capability, but RS-422 has a limit of 10. For both communication protocols, you should provide your own termination. All National Instruments RS-485 boards will work with RS-422 standards.

The following table compares mode of operation, total number of drivers and receivers, maximum cable length, and maximum data rate.

Specifications	RS-232	RS-422
Mode of Operation	Single-Ended	Differential
Total Number of Drivers and Receivers on One Line. One driver active at a time for RS-485 networks	1 Driver 1 Receiver	1 Driver 10 Receiver
Maximum Cable Length	50 ft (2500 pF)	4000 ft
Maximum Data Rate (40 ft - 4000 ft for RS-422/RS-485)	160 kbits/s (canbe up to 1Mbit/s)	10 Mbit/s

Q.3. Explain arithmetic instructions of 8085 instruction set. (15)

Ans. ARITHMETIC INSTRUCTIONS

Opcode Operand DescriptionAdd register or memory to accumulator ADD R The contents of the operand (register or memory) are M added to the contents of the accumulator and the result is stored in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers. All flags are modified to reflect the result of the addition.

Example: ADD B or ADD M

Add register to accumulator with carry ADC R The contents of the operand (register or memory) and M the Carry flag are added to the contents of the accumulator and the result is stored in the accumulator. If the operand is a memory location, its location is specified by the contents ofthe HL registers. All flags are modified to reflect the result of the addition.

Example: ADC B or ADC M

Add immediate to accumulator ADI 8-bit data The 8-bit data (operand) is added to the contents of the accumulator and the result is stored in the accumulator. All flags are modified to reflect the result of the addition.

Example: ADI 45H

Add immediate to accumulator with carry ACI 8-bit data The 8-bit data (operand) and the Carry flag are added to the contents of the accumulator and the result is stored in the accumulator. All flags are modified to reflect the result of the addition.

Example: ACI 45H

Add register pair to H and L registers DAD Reg. pair The 16-bit contents of the specified register pair are added to the contents of the HL register and the sum is stored in the HL register. The contents of the source register pair are not altered. If the result is larger than 16 bits, the CY flag is set.

No other flags are affected.

Example: DAD H

8085 Instruction

Subtract register or memory from accumulator SUB R The contents of the operand (register or memory) are M subtracted from the contents of the accumulator, and the result is stored in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers. All flags are modified to reflect the result of the subtraction.

Example: SUB B or SUB M

Subtract source and borrow from accumulator SBB R The contents of the operand (register or memory) and M the Borrow flag are subtracted from the contents of the accumulator and the result is placed in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers. All flags are modified to reflect the result of the subtraction.

Example: SBB B or SBB MSubtract immediate from accumulator SUI 8-bit data The 8-bit data (operand) is subtracted from the contents of the accumulator and the result is stored in the accumulator. All flags are modified to reflect the result of the subtraction.

Example: SUI 45H

Subtract immediate from accumulator with borrow SBI 8-bit data The 8-bit data (operand) and the Borrow flag are subtracted from the contents of the accumulator and the result is storedin the accumulator. All flags are modified to reflect the result of the subtracion.

Example: SBI 45H

Increment register or memory by 1 INR R The contents of the designated register or memory) are M incremented by 1 and the result is stored in the same place. If the operand is a memory location, its location is specified by the contents of the HL registers.

Example: INR B or INR M

Increment register pair by 1 INX R The contents of the designated register pair are incrementedby 1 and the result is stored in the same place.

Example: INX H

8085 Instruction

Decrement register or memory by 1 DCR R The contents of the designated register or memory are M decremented by 1 and the result is stored in the same place. If the operand is a memory location, its location is specified by the contents of the HL registers.

Example: DCR B or DCR M

Decrement register pair by 1 DCX R The contents of the designated register pair are decremented by 1 and the result is stored in the same place.

Example: DCX H

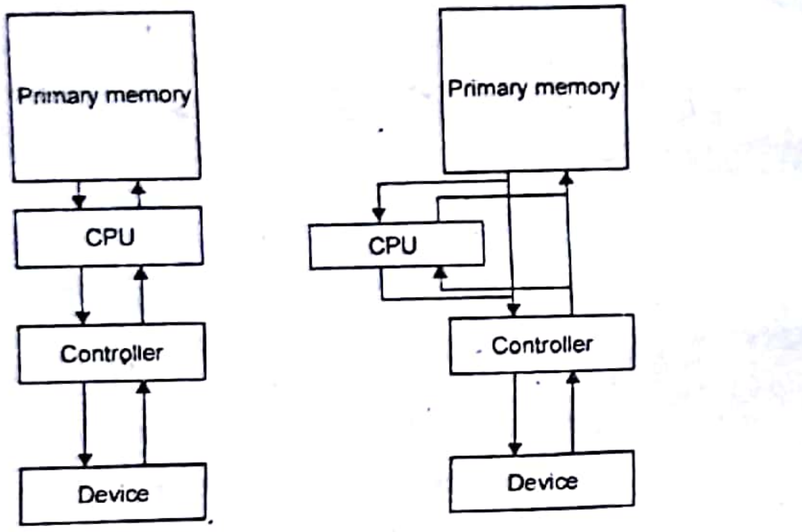
Decimal adjust accumulator DAA none The contents of the accumulator are changed from a binary value to two 4-bit binary coded decimal (BCD) digits. This is the only instruction that uses the auxiliary flag to perform the binary to BCD conversion, and the conversion procedure is described below. S, Z, AC, P, CY flags are altered to reflect the results of the operation.

If the value of the low-order 4-bits in the accumulator is greater than 9 or if AC flag is set, the instruction adds 6 to the low-order four bits.

If the value of the high-order 4-bits in the accumulator is greater than 9 or if the Carry flag is set, the instruction adds 6 to the high-order four bits.

Q.4. Why does DMA have priority over the CPU when both request a memory transfer? **(5)**

Ans. The CPU can wait to fetch instructions and data from memory without any damage occurring except loss of time. DMA usually transfers data from a device that cannot be stopped since information continues to flow so loss of data may occur.

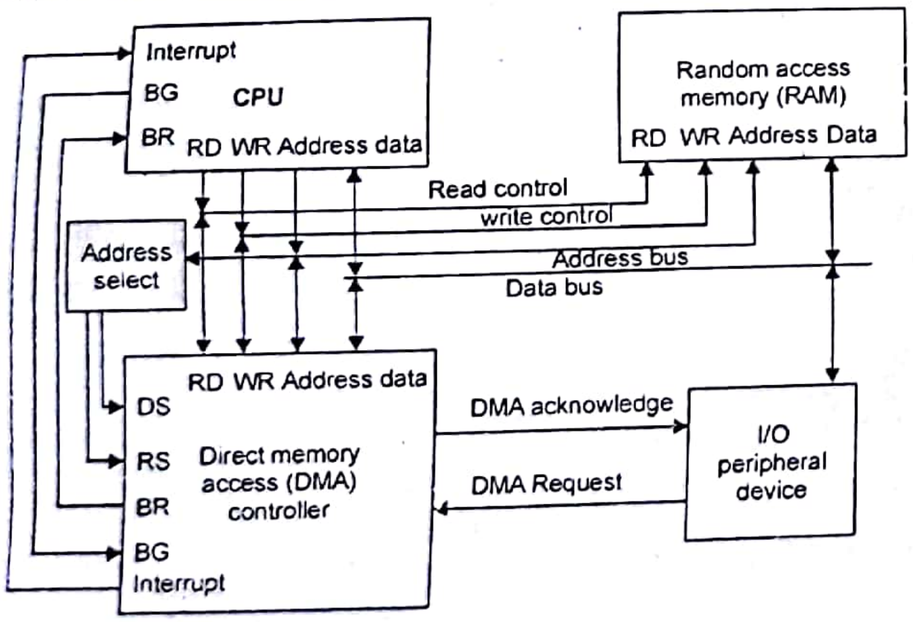


(a) Traditional I/O

(b) Direct Memory access

Q.5. What is direct memory access (DMA)? Why are the read and write control lines in a DMA controller bidirectional? **(15)**

Ans. DMA allows certain hardware subsystems within the computer to access system memory independently of the central processing unit (CPU). when the CPU is using programmed input/output, it is typically fully occupied for the entire duration of the read or write operation, and is thus unavailable to perform other work. With DMA, the CPU initiates the transfer, does other operations while the transfer is in progress, and receives an interrupt from the DMA controller when the operation is done. This feature is useful any time the CPU cannot keep up with the rate of data transfer, or



where the CPU needs to perform useful work while waiting for a relatively slow I/O data transfer. Many hardware systems use DMA, including disk drive controllers, graphics cards, network cards and sound cards. DMA is also used for intra-chip data transfer in multi-core processors. Computers that have DMA channels can transfer data to and from devices with much less CPU overhead than computers without a DMA channel. Similarly, a processing element inside a multi-core processor can transfer data to and from its local memory without occupying its processor time, allowing computation and data transfer to proceed in parallel.

DMA can also be used for "memory to memory" copying or moving of data within memory. DMA can offload expensive memory operations, such as large copies or scatter-gather operations, from the CPU to a dedicated DMA engine.

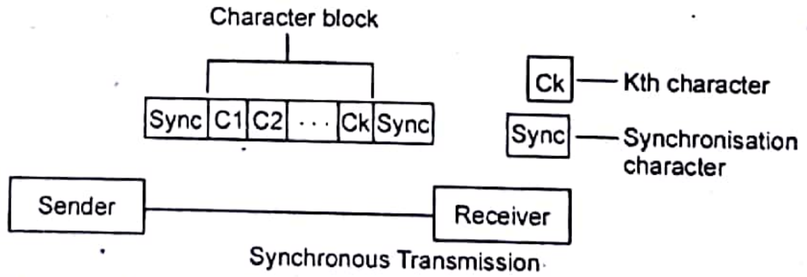
A DMA controller can generate addresses and initiate memory read or write cycles. It contains several registers that can be written and read by the CPU. These include a memory address register, a byte count register, and one or more control registers. The control registers specify the I/O port to use, the direction of the transfer (reading from the I/O device or writing to the I/O device), the transfer unit (byte at a time or word at a time), and the number of bytes to transfer in one burst.

To carry out an input, output or memory-to-memory operation, the host processor initializes the DMA controller with a count of the number of words to transfer, and the memory address to use. The CPU then sends commands to a peripheral device to initiate transfer of data. The DMA controller then provides addresses and read/write control lines to the system memory. Each time a word of data is ready to be transferred between the peripheral device and memory, the DMA controller increments its internal address register until the full block of data is transferred. So, in DMA WR and RD lines are BI-directional.

DMA transfers can either occur one word at a time, allowing the CPU to access memory on alternate bus cycles - this is called cycle stealing since the DMA controller and CPU contend for memory access.

Q.6. Differentiate between synchronous data transfer and asynchronous data transfer. Give suitable examples for each of the data transfer technique.(10)

Ans. Data transfer method in which a continuous stream of data signals is accompanied by timing signals (generated by an electronic clock) to ensure that the transmitter and the receiver are in step (synchronized) with one another. The data is sent in blocks (called frames or packets) spaced by fixed time intervals. In contrast, asynchronous transmission works in spurts and must insert a start bit before each data character and a stop bit at its termination to inform the receiver where it begins and ends. Most network protocols (such as Ethernet, SONET, Token Ring) use synchronous transmission whereas asynchronous transmission is used commonly for communications over telephone lines.



Synchronous Transmission

An asynchronous data transmission involves a mechanism called a queue. In general, a queue is a service which temporarily holds messages destined for a receiving task until that task is ready to process them. The sending task passes messages off to the queue and does not wait for a response from the receiver. The queue guarantees that if it accepts a message then that message will be delivered. Cogent's asynchronous messaging is implemented through a queue administrator, called Cascade Queue Server, or qserve. Asynchronous transmission uses start and stop bits to signify the beginning

bit ASCII character would actually be transmitted using 10 bits. For example, "0100 0001" would become "1 0100 0001 0". The extra one (or zero, depending on parity bit) at the start and end of the transmission tells the receiver first that a character is coming and secondly that the character has ended. This method of transmission is used when data are sent intermittently as opposed to in a solid stream. In the previous example the start and stop bits are in bold. The start and stop bits must be of opposite polarity. This allows the receiver to recognize when the second packet of information is being sent.

It should be noted that the queue administrator never initiates a message transaction. and due to the nature of the send/receive/reply mechanism it will never block while transmitting a reply. Thus a queue administrator in this scenario will always be available to receive messages and the originator of an asynchronous message will never block.

There are a number of disadvantages to using asynchronous message passing. The sender and receiver in an asynchronous transaction must agree upon a queue name in order for the queue administrator to correctly route a message. This requires either hard-coded queue names, command-line arguments, or a queue-aware name server so tasks can discover the destination queue names. The Cascade Data Hub uses a queue-aware name server called and serve to provide maximum flexibility in asynchronous connections.

A second concern is that because asynchronous transmission introduces additional message passing overhead into the transmission mechanism, the speed of asynchronous transmission will tend to be slower than that of synchronous data transfer. Perhaps the biggest drawback of the asynchronous transmission method is that there must be a method of dealing with overflows in the queue. If the receiving task fails to collect its messages from the queue administrator, or if it cannot keep up with the rate at which messages is being sent to the queue administrator, then eventually messages will build up and the queue administrator will run out of buffer space. At this point, the queue administrator must refuse further incoming messages until the receiver has cleared some of its pending messages. The sending task must include a contingency plan for undeliverable messages.

A sending task may react in several ways to a full queue situation:

Throw away data that cannot be put on the queue. This is both the most common and least acceptable response to a full queue. The most recent data will be lost, and old data currently on the queue will be retained. Once the condition causing the full queue has been cleared, the most recent data will be unavailable. In the case of process control systems, this scenario will fail to transmit the most recent process changes. Obviously, this method is not attractive when you consider the potential loss of state change information and critical alarm data.

Throw away the oldest data on the queue. This method is only viable if the queue is held internally to the sending task. In most cases, a task cannot walk the queue as it is provided by an external program or operating system facility. Even if the queue is available to the sender, there is no guarantee that the oldest data is the least important. In general, eliminating data solely on the basis of age is unacceptable. (In the case of the QNX queue administrator, Mqueue, this method of dealing with a full queue is not an option, as the queue is held externally to the sender.)

Throw away the oldest duplicate data on the queue. As above, this option is only available if the queue is internal to the sender. In addition, it implies that the data in the queue is of a known type and format, and can be examined for its similarity to new data. In the case of process control points, this is generally true, though it means that every point being transmitted must be compared to every queued message to determine whether it should replace a current message or go to the end of the queue. Depending on the implementation, this mechanism may not preserve time ordering in the queued data. The Cascade Data Hub uses a generalized external queue mechanism, and so does not provide this type of data reduction.

Try to send the data again later. If a task cannot deliver a message to the queue immediately, then it has the option of holding the message for later transmission. Once space is available in the queue, the message can be re-transmitted. This method actually means that the sender is implementing its own internal queue, duplicating the work of the queue facility, and so is subject to all of the same concerns regarding full buffers and data reduction. In general, this approach by itself is entirely useless, and is exactly equivalent to enlarging the size of the queue administrator's buffers. When used in combination with one of the other methods mentioned here, this approach could be very effective.

Try to send the data again later, throwing away old duplicate data from the retry queue only. This is the method used by the Cascade Data Hub to deal with undeliverable data. So long as the receiver is capable of keeping up with the data transmission rate, all point changes are deliverable. If the receiver allows its queue to fill, then the Cascade Data Hub will flag the point as pending delivery to the recipient and will attempt to deliver the most recent value for that point as soon as there is room in the queue. If another change to a pending point is received by the Data Hub, then the previous value is overwritten, so that when room in the queue becomes available, the new value is transmitted. In addition, the Cascade Data Hub packages many point changes into a single message whenever there are several pending points to a task, thereby increasing the bandwidth of the queue administrator during periods of high load. In this way, the Cascade Data Hub will transmit all point changes to any receiver that can handle the data rate, and guarantees that even slow receivers (such as GUI applications and applications on dial-up lines) will always receive the most recent point values even if they miss some intermediate values.

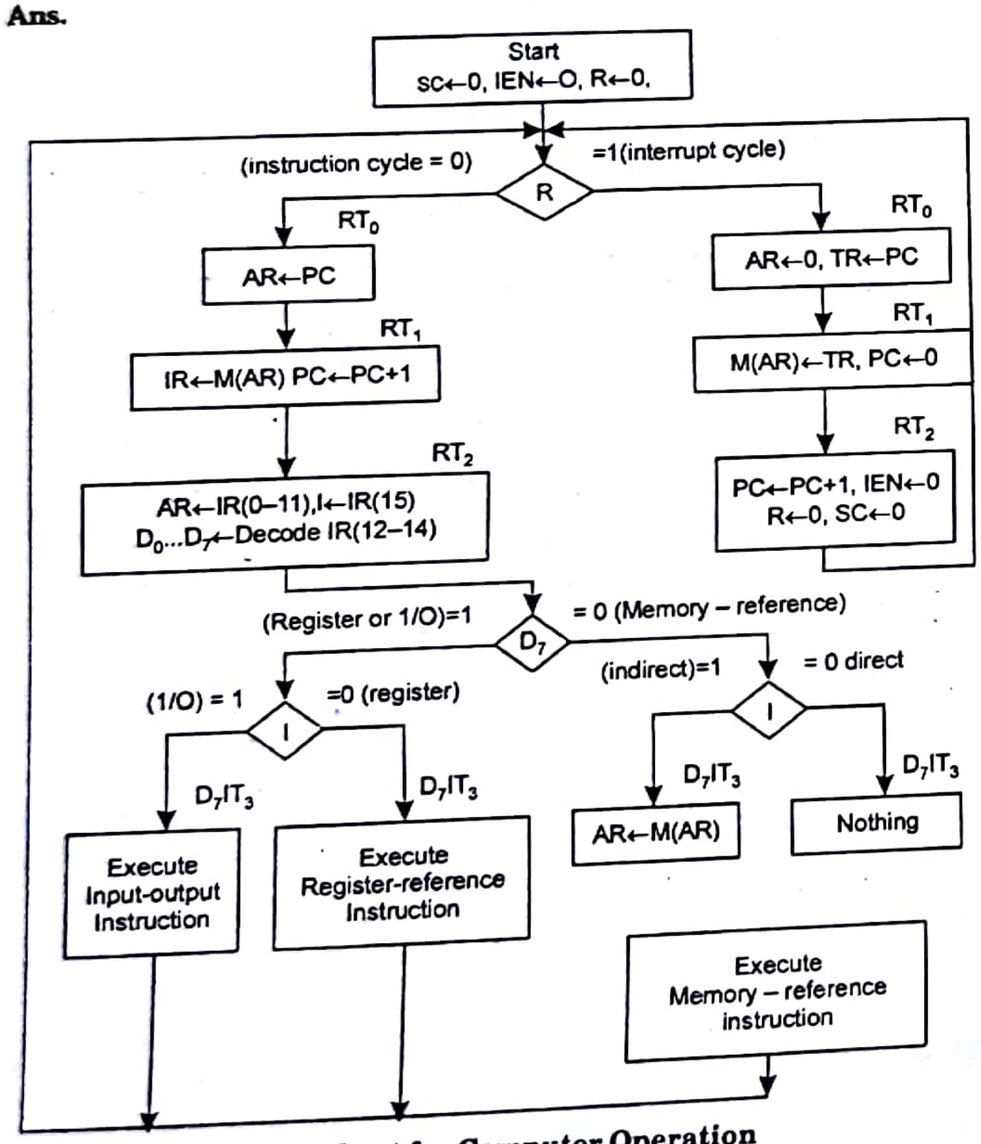
MODEL TEST PAPER-II
FIRST TERM EXAMINATION
FOURTH SEMESTER (B.TECH)
COMPUTER ORGANIZATION & ARCHITECTURE–[ETCS-204]

Time : 1½ hrs. **M.M. : 30**

Note: *All questions are compulsory.*

Q.1. (a) Explain the operation of a computer with the help of flowchart.(5)

Ans.



Flowchart for Computer Operation

Q.1.(b) Different types of memories in computer system. **(5)**

Ans. The term memory hierarchy is used in computer architecture when discussing performance issues in computer architectural design, algorithm predictions, and the

lower level programming constructs such as involving locality of reference. A 'memory hierarchy' in computer storage distinguishes each level in the 'hierarchy' by response time. Since response time, complexity, and capacity are related, the levels may also be distinguished by the controlling technology.

The many trade-offs in designing for high performance will include the structure of the memory hierarchy, i.e. the size and technology of each component. So the various components can be viewed as forming a hierarchy of memories ($m_1, m_2, ..., m_n$) in which each member m_i is in a sense subordinate to the next highest member m_{i-1} of the hierarchy. To limit waiting by higher levels, a lower level will respond by filling a buffer and then signaling to activate the transfer.

There are four major storage levels.

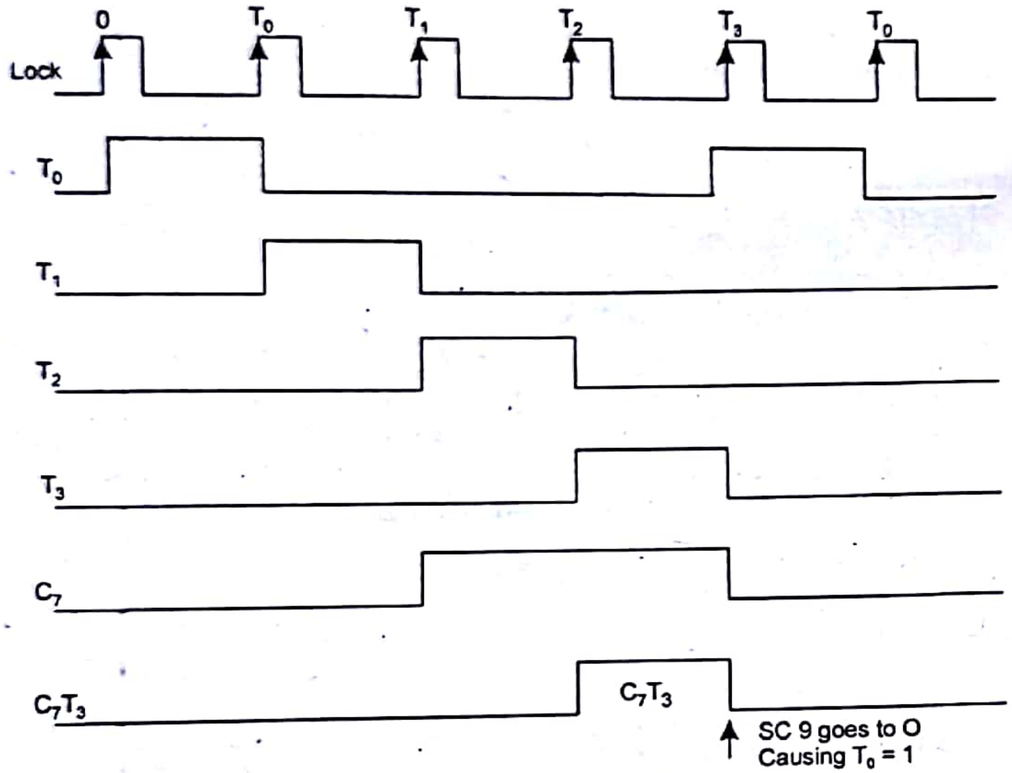
1. Internal – Processor registers and cache.
2. Main – the system RAM and controller cards.
3. On-line mass storage – Secondary storage.
4. Off-line bulk storage – Tertiary and Off-line storage.

Computer Memory Hierarchy

Q.2. (a) Draw the timing diagram by assuming that SC is cleared to 0 at time T3 if control signal is active.

C7 T3: SC<- 0, C7 is activated with positive clock transition associated with T1. **(5)**

Ans.



Q.2.(*b*) Determine the number of clock cycles that it takes to process 200 tasks in a six segment pipeline. **(5)**

Ans. 6+200-1=205 cycles.

Showing similar pipeline for 8 tasks having 13 cycles

Segment	1	2	3	4	5	6	7	8	9	10	11	12	13
1	T1	T2	T3	T4	T5	T6	T7	T8					
2		T1	T2	T3	T4	T5	T6	T7	T8				
3			T1	T2	T3	T4	T5	T6	T7	T8			
4				T1	T2	T3	T4	T5	T6	T7	T8		
5					T1	T2	T3	T4	T5	T6	T7	T8	
6						T1	T2	T3	T4	T5	T6	T7	T8

Q.3. (*a*) Write short note on memory transfer? **(5)**

Ans. The internal bus connects only registers within the CPU, so how do we get data to and from memory?

The address register (AR) is used to select a memory address, and the data register (DR) is used to send and receive data. Both these registers are connected to the internal bus. DR is a bridge between the internal BUS and the memory data BUS.

Memory can also be connected directly to the internal BUS in theory.

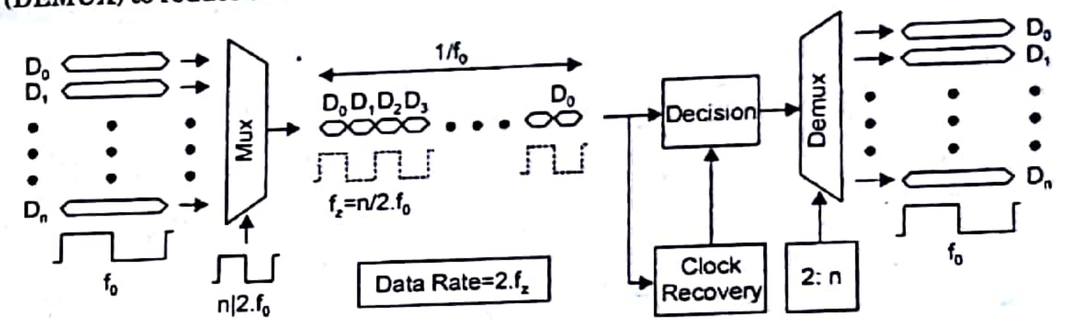
Diagram showing connections to memory unit.

$M[AR] \leftarrow DR$

$DR \leftarrow M[AR]$

Q.3.(*b*) Explain serial bus architecture and Why is it possible to achieve higher data rates using serial data buses instead of parallel ones? **(5)**

In a serial bus architecture, n parallel data bits are multiplexed (MUX) on the transmitter side. The data transfer takes place at a speed which n-times higher than the data rate of the parallel data. On the receiver side, the data have to be demultiplexed (DEMUX) to reduce the data rate which appropriate for further processing on the chip.



- Data D0...Dn are mutiplexed to one stream

⇒ Decision and demultiplexing in the receiver

⇒ **no data skew**

- Clock signal is not transmitted

⇒ Clock recovery in the receiver

⇒ **no clock skew**

- One clock recovery per link = one clock per link

⇒ **Link capacity can be linearly scaled**

- Optical transmission (laser, fiber, photodiode) can be employed

⇒ **unlimited l_{max}**

MODEL TEST PAPER-II
SECOND TERM EXAMINATION
FOURTH SEMESTER (B.TECH)
COMPUTER ORGANIZATION & ARCHITECTURE–[ETCS-204]

Time : 1½ hrs. **M.M. : 30**

Note: *All questions are compulsory.*

Q.1.(a) Explain 4 possible hardware schemes that can be used in instruction pipeline in order to minimize the performance degradation caused by instruction branching. **(5)**

• **Predict Not Taken:** Always fetch the instruction after the branch from the instruction cache, but only execute it if the branch is not taken. If the branch is not taken, the pipeline stays full. If the branch is taken, the instruction is flushed (marked as if it were a NOP), and one cycle's opportunity to finish an instruction is lost.

• **Branch Likely:** Always fetch the instruction after the branch from the instruction cache, but only execute it if the branch was taken. The compiler can always fill the branch delay slot on such a branch, and since branches are more often taken than not, such branches have a smaller IPC penalty than the previous kind.

• **Branch Delay Slot:** Always fetch the instruction after the branch from the instruction cache, and always execute it, even if the branch is taken. Instead of taking an IPC penalty for some fraction of branches either taken (perhaps 60%) or not taken (perhaps 40%), branch delay slots take an IPC penalty for those branches into which the compiler could not schedule the branch delay slot. The SPARC, MIPS, and MC88K designers designed a branch delay slot into their ISAs.

• **Branch Prediction:** In parallel with fetching each instruction, guess if the instruction is a branch or jump, and if so, guess the target. On the cycle after a branch or jump, fetch the instruction at the guessed target. When the guess is wrong, flush the incorrectly fetched target.

Q.2. Write a short note on the following: **(10)**

(a) Input – output processor

1. The CPU puts a read command on the bus addressed to the device.
2. The controller puts the data on the bus addressed to a CPU register.
3. The CPU reads this data into the register.
4. The CPU then places the data back onto the bus, addressed to somewhere in memory.
5. The memory location receives the data addressed to it.

(b) Memory interleaving: Memory interleaving is a method to increase the speed of the high end microprocessors and it is even applicable to the hard disks too. It can be of 2 types: 2 way interleaving(using 2 complete address buses) & 4 way interleaving(using complete 4 address buses). There is a controller too that generates the addresses. CPU can access alternate sections of memory..while one section is busy processing upon a word at a particular location, the other section accesses the word at the next location....resembling overlapping!!!

In an interleaved memory system, there are still two physical banks of DRAM, but logically the system sees one bank of memory that is twice as large. In the interleaved bank, the first long word of bank 0 is followed by the first long word of bank 1, which is followed by the second long word of bank 0, which is followed by the second long word of bank 1, and so on.

Q.3. Explain control instructions in 8085 instruction set. **(15)**

Ans. Opcode Operand Description

No operation

NOP none No operation is performed. The instruction is fetched and decoded. However no operation is executed.

Example: NOP

Halt and enter wait state HLT none The CPU finishes executing the current instruction and halts any further execution. An interrupt or reset is necessary to exit from the halt state.

Example: HLT

Disable interrupts

DI none The interrupt enable flip-flop is reset and all the interrupts except the TRAP are disabled. No flags are affected.

Example: DI

Enable interrupts

EI none The interrupt enable flip-flop is set and all interrupts are enabled. No flags are affected. After a system reset or the acknowledgement of an interrupt, the interrupt enable flipflop is reset, thus disabling the interrupts. This instruction is necessary to reenable the interrupts (except TRAP).

Example: EI

Read interrupt mask

RIM none This is a multipurpose instruction used to read the status of interrupts 7.5, 6.5, 5.5 and read serial data input bit. The instruction loads eight bits in the accumulator with the following interpretations.

Example: RIM

Set interrupt mask

SIM none This is a multipurpose instruction and used to implement the 8085 interrupts 7.5, 6.5, 5.5, and serial data output. The instruction interprets the accumulator contents as follows.

Example: SIM

MODEL TEST PAPER-II
END TERM EXAMINATION
FOURTH SEMESTER (B.TECH)
COMPUTER ORGANIZATION & ARCHITECTURE–[ETCS-204]

Time : 3 hrs. **M.M. : 75**

Note: *All questions are compulsory.*

Q.1. Explain UART. **(10)**

Ans. The universal asynchronous receiver/transmitter (UART) takes bytes of data and transmits the individual bits in a sequential fashion.At the destination, a second UART re-assembles the bits into complete bytes. Each UART contains a shift register, which is the fundamental method of conversion between serial and parallel forms. Serial transmission of digital information (bits) through a single wire or other medium is less costly than parallel transmission through multiple wires.

The UART usually does not directly generate or receive the external signals used between different items of equipment. Separate interface devices are used to convert the logic level signals of the UART to and from the external signalling levels. External signals may be of many different forms. Examples of standards for voltage signaling are RS-232, RS-422 and RS-485 from the EIA. Historically, current (in current loops) was used in telegraph circuits. Some signaling schemes do not use electrical wires. Examples of such are optical fiber, IrDA (infrared), and (wireless) Bluetooth in its Serial Port Profile (SPP).Communication may be *simplex* (in one direction only, with no provision for the receiving device to send information back to the transmitting device), *full duplex* (both devices send and receive at the same time) or *half duplex* (devices take turns transmitting and receiving).

Character framing: The right-most (least significant) data bit is always transmitted first. If parity is present, the parity bit comes after the data bits but before the stop bit(s).

Bit number	1	2	3	4	5	6	7	8	9	10	11
Start bit 5–8 data bits										**Stop bit(s)**	
Start	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Stop		

The idle, no data state is high-voltage, or powered. This is a historic legacy from telegraphy, in which the line is held high to show that the line and transmitter are not damaged. Each character is sent as a logic low start bit, a configurable number of data bits (usually 8, but users can choose 5 to 8 or 9 bits depending on which UART is in use), an optional parity bit if the number of bits per character chosen is not 9 bits, and one or more logic high stop bits.

The start bit signals the receiver that a new character is coming. The next five to nine bits, depending on the code set employed, represent the character. If a parity bit is used, it would be placed after all of the data bits. The next one or two bits are always in the mark (logic high, i.e., '1') condition and called the stop bit(s). They signal the receiver that the character is completed. Since the start bit is logic low (0) and the stop bit is logic high (1) there are always at least two guaranteed signal changes between characters.

If the line is held in the logic low condition for longer than a character time, this is a break condition that can be detected by the UART.

Receiver: All operations of the UART hardware are controlled by a clock signal which runs at a multiple of the data rate, typically 8 times the bit rate. The receiver tests the state of the incoming signal on each clock pulse, looking for the beginning of the start bit. If the apparent start bit lasts at least one-half of the bit time, it is valid

and signals the start of a new character. If not, it is considered a spurious pulse and is ignored. After waiting a further bit time, the state of the line is again sampled and the resulting level clocked into a shift register. After the required number of bit periods for the character length (5 to 8 bits, typically) have elapsed, the contents of the shift register are made available (in parallel fashion) to the receiving system. The UART will set a flag indicating new data is available, and may also generate a processor interrupt to request that the host processor transfers the received data.

Communicating UARTs usually have no shared timing system apart from the communication signal. Typically, UARTs resynchronize their internal clocks on each change of the data line that is not considered a spurious pulse. Obtaining timing information in this manner, they reliably receive when the transmitter is sending at a slightly different speed than it should. Simplistic UARTs do not do this, instead they resynchronize on the falling edge of the start bit only, and then read the center of each expected data bit, and this system works if the broadcast data rate is accurate enough to allow the stop bits to be sampled reliably.

It is a standard feature for a UART to store the most recent character while receiving the next. This "double buffering" gives a receiving computer an entire character transmission time to fetch a received character. Many UARTs have a small first-in, first-out FIFO buffer memory between the receiver shift register and the host system interface. This allows the host processor even more time to handle an interrupt from the UART and prevents loss of received data at high rates.

Transmitter: Transmission operation is simpler since it is under the control of the transmitting system. As soon as data is deposited in the shift register after completion of the previous character, the UART hardware generates a start bit, shifts the required number of data bits out to the line, generates and appends the parity bit (if used), and appends the stop bits. Since transmission of a single character may take a long time relative to CPU speeds, the UART will maintain a flag showing busy status so that the host system does not deposit a new character for transmission until the previous one has been completed; this may also be done with an interrupt. Since full-duplex operation requires characters to be sent and received at the same time, UARTs use two different shift registers for transmitted and received characters.

Q.2. Explain segmented page mapping used in memory management hardware. **(10)**

Ans. Memory segmentation is the division of computer's primary memory into segments or sections. In a computer system using segmentation, a reference to a memory location includes a value that identifies a segment and an offset within that segment. Segments or sections are also used in object files of compiled programs when they are linked together into a program image and when the image is loaded into memory

Memory segmentation is one method of implementing memory protection. Paging is another, and they can be combined. The size of a memory segment is generally not fixed and may be as small as a single byte. Segments usually represent natural divisions of a program such as individual routines or data tables so segmentation is generally more visible to the programmer.

A segment has a length and set of permissions associated with it. A process is only allowed to make a reference into a segment if the type of reference is allowed by the permissions, and the offset within the segment is within the range specified by the length of the segment. Otherwise, a hardware exception such as a segmentation fault is raised.

Information is also associated with a segment that indicates where the segment is located in memory. This might be the address of the first location in the segment, or the address of a page table for the segment if the segmentation is implemented with paging. In the first case, if a reference to a location within a segment is made, the offset within the segment will be added to address of the first location in the segment to give the address in memory of the referred-to item; in the second case, the offset of the segment is translated to a memory address using the page table.

Segments may also have a flag indicating whether the segment is present in main memory or not; if a segment is accessed that is not present in main memory, an exception is raised, and the operating system will read the segment into memory from secondary storage.

When a segment does not have a page table associated with it, the address of the first location in the segment is usually an address in main memory; in those situations, no paging is done. In the Intel 80386 and later, that address can either be an address in main memory, if paging is not enabled, or an address in a paged address space, if paging is enabled.

A memory management unit (MMU) is responsible for translating a segment and offset within that segment into a memory address, and for performing checks to make sure the translation can be done and that the reference to that segment and offset is permitted.

Q.3. Explain 8085 LOGICAL INSTRUCTIONS. **(15)**

Ans. Opcode Operand Description Compare register or memory with accumulator CMP R The contents of the operand (register or memory) are M compared with the contents of the accumulator. Both contents are preserved . The result of the comparison is shown by setting the flags of the PSW as follows:

if (A) < (reg/mem): carry flag is set

if (A) = (reg/mem): zero flag is set

if (A) > (reg/mem): carry and zero flags are reset

Example: CMP B or CMP M

Compare immediate with accumulator

CPI 8-bit data The second byte (8-bit data) is compared with the contents of the accumulator. The values being compared remain unchanged. The result of the comparison is shown by setting the flags of the PSW as follows:

if (A) < data: carry flag is set

if (A) = data: zero flag is set

if (A) > data: carry and zero flags are reset

Example: CPI 89H

Logical AND register or memory with accumulator ANA R The contents of the accumulator are logically ANDed with M the contents of the operand (register or memory), and the result is placed in the accumulator. If the operand is a memory location, its address is specified by the contents of HL registers. S, Z, P are modified to reflect the result of the operation. CY is reset. AC is set.

Example: ANA B or ANA M

Logical AND immediate with accumulator ANI 8-bit data The contents of the accumulator are logically ANDed with the 8-bit data (operand) and the result is placed in the accumulator. S, Z, P are modified to reflect the result of the operation. CY is reset. AC is set.

Example: ANI 86H

Exclusive OR register or memory with accumulator

XRA R The contents of the accumulator are Exclusive ORed with M the contents of the operand (register or memory), and the result is placed in the accumulator. If the operand is a memory location, its address is specified by the contents of HL registers. S, Z, P are modified to reflect the result of the operation. CY and AC are reset.

Example: XRA B or XRA M

Exclusive OR immediate with accumulator XRI 8-bit data The contents of the accumulator are Exclusive ORed with the 8-bit data (operand) and the result is placed in the accumulator. S, Z, P are modified to reflect the result of the operation. CY and AC are reset.

Example: XRI 86H

Logical OR register or memory with accumulator ORA R The contents of the accumulator are logically ORed with M the contents of the operand (register or memory),

and the result is placed in the accumulator. If the operand is a memory location, its address is specified by the contents of HL registers. S, Z, P are modified to reflect the result of the operation. CY and AC are reset.

Example: ORA B or ORA M

Logical OR immediate with accumulator ORI 8-bit data The contents of the accumulator are logically ORed with the 8-bit data (operand) and the result is placed in the accumulator. S, Z, P are modified to reflect the result of the operation. CY and AC are reset.

Example: ORI 86H

Rotate accumulator left

RLC none Each binary bit of the accumulator is rotated left by one position. Bit D7 is placed in the position of D0 as well as in the Carry flag. CY is modified according to bit D7. S, Z, P, AC are not affected.

Example: RLC

Rotate accumulator right

RRC none Each binary bit of the accumulator is rotated right by one position. Bit D0 is placed in the position of D7 as well as in the Carry flag. CY is modified according to bit D0. S, Z, P,

AC are not affected.

Example: RRC

Rotate accumulator left through carry RAL none Each binary bit of the accumulator is rotated left by one position through the Carry flag. Bit D7 is placed in the Carry flag, and the Carry flag is placed in the least significant position D0. CY is modified according to bit D7. S, Z, P, AC are not affected.

Example: RAL

Rotate accumulator right through carry RAR none Each binary bit of the accumulator is rotated right by one position through the Carry flag. Bit D0 is placed in the Carry flag, and the Carry flag is placed in the most significant position D7. CY is modified according to bit D0. S, Z, P, AC are not affected.

Example: RAR

Complement accumulator

CMA none The contents of the accumulator are complemented. No flags are affected.

Example: CMA

Complement carry

CMC none The Carry flag is complemented. No other flags are affected.

Example: CMC

Set Carry

STC none The Carry flag is set to 1. No other flags are affected.

Example: STC

Q.4. Discuss role of reverse polish notation in stack organization. **(10)**

Ans. Reverse Polish notation (RPN) is a mathematical notation in which every operator follows all of its operands, in contrast to Polish notation, which puts the operator in the prefix position. It is also known as postfix notation and is parenthesis-free as long as operator arities are fixed. The description "Polish" refers to the nationality of logician Jan Łukasiewicz, who invented (prefix) Polish notation in the 1920s.

In reverse Polish notation the operators follow their operands; for instance, to add 3 and 4, one would write "3 4 +" rather than "3 + 4". If there are multiple operations, the operator is given immediately after its second operand; so the expression written "3 " 4 + 5" in conventional notation would be written "3 4 " 5 +" in RPN: first subtract 4 from 3, then add 5 to that. An advantage of RPN is that it obviates the need for parentheses that are required by infix. While "3 " 4 * 5" can also be written "3 " (4 * 5)", that means something quite different from "(3 " 4) * 5". In postfix, the former could be written "3 4 5 * "", which unambiguously means "3 (4 5 *) "" which reduces to "3 20 ""; the latter could

be written "3 4 - 5 *" (or 5 3 4 - *, if you wish to keep similar formatting), which unambiguously means "(3 4 -) 5 *".

Despite the name, reverse Polish notation is not exactly the reverse of Polish notation, for the operands of non-commutative operations are still written in the conventional order (e.g. "/ 6 3" in Polish notation and "6 3 /" in reverse Polish both evaluating to 2, whereas "3 6 /" in reverse Polish notation would evaluate to ½).

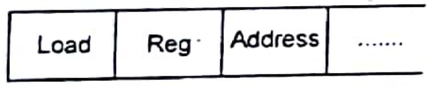
The algorithm for evaluating any postfix expression is fairly straightforward:

- While there are input tokens left
- Read the next token from input.
- If the token is a value
  - Push it onto the stack.
- Otherwise, the token is an operator (operator here includes both operators, and functions).
  - It is known *a priori* that the operator takes n arguments.
  - If there are fewer than n values on the stack
  - (Error) The user has not input sufficient values in the expression.
  - Else, Pop the top n values from the stack.
  - Evaluate the operator, with the values as arguments.
  - Push the returned results, if any, back onto the stack.
- If there is only one value in the stack
- That value is the result of the calculation.
- If there are more values in the stack
- (Error) The user input has too many values.

Q.5. Discuss the various addressing modes schemes used by CPU. Provide example for each addressing mode scheme. **(10)**

Ans. (*i*) Direct address mode

Absolute/Direct



(Effective address = address as given in instruction)

This requires space in an instruction for quite a large address.

Some RISC machines have a special *Load Upper Literal* instruction which places a 16-bit constant in the top half of a register. An *OR literal* instruction can be used to insert a 16-bit constant in the lower half of that register, so that a full 32-bit address can then be used via the register-indirect addressing mode, which itself is provided as "base-plus-offset" with an offset of 0.

(*ii*) Indirect address mode

Register indirect

(Effective address = contents of base register)

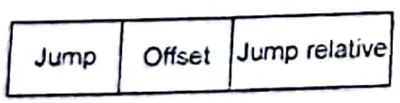
A few computers have this as a distinct addressing mode. Many computers just use *base plus offset* with an offset value of 0.

An indirect address instruction needs three references to memory:

(1) Read instruction
(2) Read effective address
(3) Read operand

(*iii*) Relative address mode

PC-relative



(Effective PC address = next instruction address + offset, offset may be negative)

The effective address for a PC-relative instruction address is the offset parameter added to the address of the next instruction. This offset is usually signed to allow reference to code both before and after the instruction.

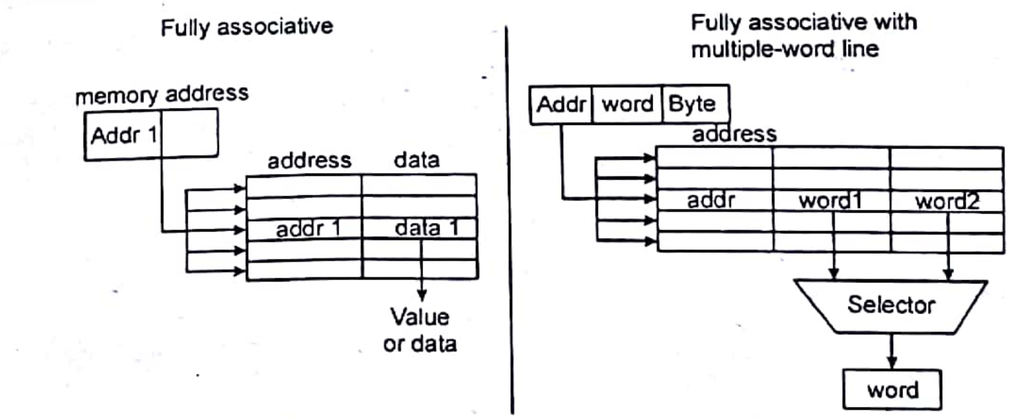
This is particularly useful in connection with jumps, because typical jumps are to nearby instructions.

Another advantage of program-relative addressing is that the code may be position-independent, i.e. it can be loaded anywhere in memory without the need to adjust any addresses.

Some versions of this addressing mode may be conditional referring to two registers ("jump if reg1=reg2"), one register ("jump unless reg1=0") or no registers, implicitly referring to some previously-set bit in the status register. See also conditional execution below.

Q.6. What is associative memory? With the help of block diagram explain its hardware organization and its working. **(10)**

Ans. A storage unit of digital computers in which selection (entry) is performed not according to concrete address but rather according to a preset combination (association) of attributes characteristic of the desired information. Such attributes can be part of a word (number), attached to it for detection among other words, certain features of the word itself (for example, the presence of specific codes in its digits), the absolute value of a word, its presence in a preset range, and so on.



The operation of an associative memory is based on the representation of all information in the form of a sequence of zones according to properties and characteristic attributes. In this case the retrieval of information is reduced to the determination of the zone according to the preset attributes by means of scanning and comparison of those attributes with the attributes that are stored in the associative memory. There are two basic methods of realizing the associative memory. The first is the construction of a memory with storage cells that have the capability of performing simultaneously the functions of storage, nondestructive reading, and comparison. Such a method of realizing an associative memory is called network parallel-associative—that is, the required sets of attributes are preserved in all the memory cells, and the information that possesses a given set of attributes is searched for simultaneously and independently over the entire storage capacity. Card indexes for edge-punched cards are prototypes of such an associative memory. Thin-film kryotrons, transfluxors, biaxes, magnetic thin films, and so on are used as storage elements of network-realized associative memories.

The second method of realizing an associative memory is the programmed organization (modeling) of the memory. It consists of the establishment of associative connections between the information contained in the memory by means of ordered arrangement of the information in the form of sequential chains or groups (lists) connected by linkage addresses whose codes are stored in the same memory cells. This

procedure is the more suitable for practical realization in dealing with large volumes of information because it provides for the use of conventional accumulators with address reference.

The use of an associative memory considerably facilitates the programming and solution of informational-logical problems and accelerates by hundreds (thousands) of times the speed of retrieval, analysis, classification, and processing of data.

Q.7. Discuss the various modes of data transfer techniques. **(10)**

Ans. 1. Programmed I/O

Programmed input/output (PIO) is a method of transferring data between the CPU and a peripheral, such as a network adapter or an ATA storage device.

In general, programmed I/O happens when software running on the CPU uses instructions that access I/O address space to perform data transfers to or from an I/O device. This is in contrast to Direct Memory Access (DMA) transfers.

The best known example of a PC device that uses programmed I/O is the ATA interface; however, this interface can also be operated in any of several DMA modes. Many older devices in a PC also use PIO, including legacy serial ports, legacy parallel ports when not in ECP mode, the PS/2 keyboard and mouse ports, legacy MIDI and joystick ports, the interval timer, and older network interfaces.

2. Interrupt- initiated I/O

The CPU issues commands to the I/O module then proceeds with its normal work until interrupted by I/O device on completion of its work.

For input, the device interrupts the CPU when new data has arrived and is ready to be retrieved by the system processor. The actual actions to perform depend on whether the device uses I/O ports, memory mapping.

For output, the device delivers an interrupt either when it is ready to accept new data or to acknowledge a successful data transfer. Memory-mapped and DMA-capable devices usually generate interrupts to tell the system they are done with the buffer.

Although Interrupt relieves the CPU of having to wait for the devices, but it is still inefficient in data transfer of large amount because the CPU has to transfer the data word by word between I/O module and memory.

Below are the basic operations of Interrupt:

- CPU issues read command
- I/O module gets data from peripheral whilst CPU does other work
- I/O module interrupts CPU
- CPU requests data
- I/O module transfers data

3. Direct Memory Access (DMA)

Direct Memory Access (DMA) means CPU grants I/O module authority to read from or write to memory without involvement. DMA module controls exchange of data between main memory and the I/O device. Because of DMA device can transfer data directly to and from memory, rather than using the CPU as an intermediary, and can thus relieve congestion on the bus. CPU is only involved at the beginning and end of the transfer and interrupted only after entire block has been transferred.

Direct Memory Access needs a special hardware called DMA controller (DMAC) that manages the data transfers and arbitrates access to the system bus. The controllers are programmed with source and destination pointers (where to read/write the data), counters to track the number of transferred bytes, and settings, which includes I/O and memory types, interrupts and states for the CPU cycles.

DMA increases system concurrency by allowing the CPU to perform tasks while the DMA system transfers data via the system and memory busses. Hardware design is complicated because the DMA controller must be integrated into the system, and the system must allow the DMA controller to be a bus master. Cycle stealing may also be necessary to allow the CPU and DMA controller to share use of the memory bus.

FIRST TERM EXAMINATION [FEB. 2015]
FOURTH SEMESTER [B.TECH]
COMPUTER ORGANIZATION AND ARCHITECTURE [ETCS-204]

Time. 1 Hour **M.M. : 30**

Note: *Q.No.1 is compulsory. Attempt any two more questions from the rest.*

Q.1. (a) Represent the floating point in the IEEE standard format for the given number $1.00010100 \times 2^{-10}$ **(2)**

Ans. The IEEE 754 single precision representation is given by

1	00010100	10000000000
1 bit	8 bits	23 bits
S	E	M

Q.1. (b) Represent the following conditional statement by two register transfer statements with control functions. **(2)**

If (P = 1) then (R1 → R2) else if (Q = 1) then (R1 ← R3)

Ans. P : R1 ← R2 ; If (P = 1) then (R1 ← R2)

P'Q : R1 ← R3 ; else if (Q = 1) the (R1 ← R3)

Q.1. (c) Starting from an initial value of R = 11011101, determine the sequence of binary values in R after a logical shif left, followed by a circular shift right, followed by a logical shift Right and a circular shift left. **(2)**

Ans. Starting from an initial value of R = 11011101, determine the sequence of binary values in R after each operation in the sequence: (1) a logical shift-left, (2) followed by an arithmetic shift-right, (3) followed by another arithmetic shift-right, and (4) followed, finally, by a circular shift-left. Show all your work.

Left shift (11011101) leads to 10111010

Arithmetic Shift Right (10111010) leads to 11011101

Arithmetic Shift Right (01011101) leads to 11101110

Circular Shift Left (00101110) leads to 11011101

Q.1. (d) Explain what operation will take place when the following instructions are executed LX1 rp, data; LDA addr, LHLD addr & STA addr **(2)**

Ans. LXI rp, data :- (Load register pair immediate) [rp] <—— data

LDA addr :- (Load Accumulator Direct) [A] <—— [addr]

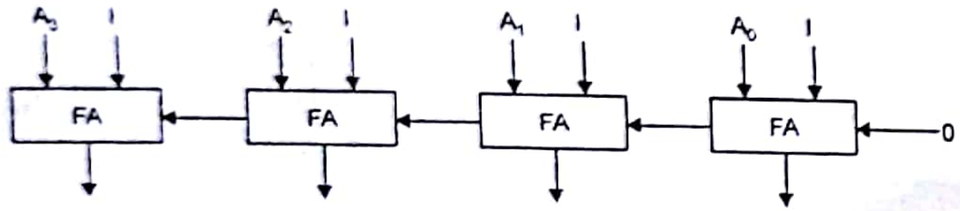
LHLD addr :- (Load H-L Pair Direct) [L] <——[addr]

[H] <——[addr +1]

STA addr :- (Store Accumulator Direct) [addr] <——[A]

Q.1. (e) Design a 4-bit combinational circuit decrementer using four full-adder circuits. **(2)**

Ans. A – 1 = A + 2's complement of 1 = A + 1111



Q.2. (a) Show the step by step multiplication process using booth algorithm. When the binary numbers (+ 15) × (– 13) are multiplied. Assume 5-bit registers that hold signed numbers. The multiplicand is + 15. **(6)**

Ans.

A		0	1	1	1	1				15
X	x	1	0	0	1	1				–13
Y		–1	0	1	0	–1				recoded multiplier
Add –A	+	1	0	0	0	1				
Shift		1	1	0	0	0	1			
Shift Only		1	1	1	0	0	0	1		
Add A	+	0	1	1	1	1				
		0	1	0	1	1	0	1		
Shift		0	0	1	0	1	1	0	1	
Shift Only		0	0	0	1	0	1	1	0	1
Add –A	+	1	0	0	0	1				
	1	0	0	1	1	1	1	0	1	
Shift	1	1	0	0	1	1	1	1	0	1 –195

Q.2. (b) What are the various registers of 8085? **(4)**

Ans. 8085 Microprocessor has 8-bit registers: A,B,C,D,E,H,L,F and two 16-bit registers PC andSP. These registers can be classified as :

1. General Purpose Registers.
2. Temporary Registers:
   - (a) Temporary Data Registers
   - (b) W and Z Registers
3. Special Purpose Registers:
   - (a) Accumulator
   - (b) Flag Registers
   - (c) Instruction Registers
4. Sixteen Bit Registers
   - (a) Program Counter (PC)
   - (b) Stack Pointer (SP)

(1) General Purpose Registers: B,C,D,E,H and L are 8 bit General Purpose Registers can be used as a separate 8-bit register or as 16-bit register pairs BC, DE and HL. When used in register pair mode, the high order bytes resides in the first register i.e in B and the lower order bytes resides in the second register when BC is used as a register pair.

HL pair is also function as a data pointer or memory pointer. They are also called Scratchpad Register, as user can store data in them. The efficient programmer prefers to use these registers to store intermediate results.

(2) Temporary Registers:

(a) Temporary Data Registers: The ALU has two Inputs, one input is supplied by the accumulator and other from temporary data registers. The programmer cannot access this temporary data registers. However, it is internally used for execution of most of the arithmetic and logic instructions.

(b) W and Z Registers: W and Z register are temporary registers. These registers are used to hold 8 bit data during execution of some instruction. These registers are not available for programmer since 8085 microprocessor uses them internally.

(3) Special Purpose Registers:-

(a) Accumulator (Register A): It is a tri-state 8-bit register. It is extensively used in arithmetic logic, load and store operation, as well as in Input-Output operation. Result of ALU are stored in this register.

(b) Flag Registers: It is 8 bit register in which 5 of the bit carry significant information in the form of flags: S(Sign flag), Z(Zero flag), AC (Auxiliary flag), P(Parity flag) and CY (Carry flag) as shown in fig.

D7	D6	D5	D4	D3	D2	D1	D0
S	Z	X	AC	X	P	X	CY

(Flag register)

(c) Instruction Registers: In a typical processor operation, the processor first fetches the opcode of instruction from memory. The CPU stores this opcode in a register called the instruction register.

(4) Sixteen Bit Registers:-

(a) Program counter (PC): The program counter is a special purpose register which a t a given time stores the address of the next instruction.

(b) Stack Register (SP): The stack is reserved area of the memory in the RAM where temporary information may be stored. A 16-bitnstack pointer is used to hold the address of the most recent stack entry.

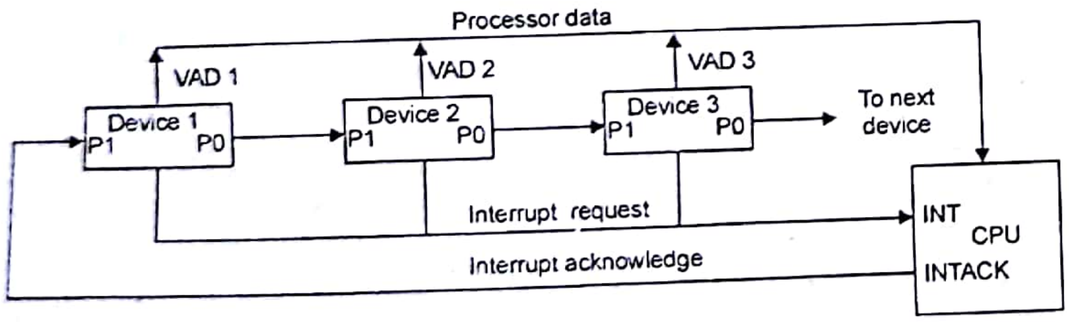
Q.3. (a) Explain block diagram of daisy chaining priority interrupt. **(6)**

Ans. Daisy-Chaining Priority

The daisy-chaining method of establishing priority consists of a serial connection of all devices that request an interrupt. The device with the highest priority is placed in the first position, followed by lower-priority devices up to the device with the lowest

priority, which is placed last in the chain. This method of connection between three devices and the CPU is shown in Fig. The interrupt request line is common to all devices and forms a wired logic connection. If any device has its interrupt signal in the low-level state, the interrupt line goes to the low-level state and enables the interrupt input in the CPU. When no interrupts are pending, the interrupt line stays in the high-level state and no interrupts are recognized by the CPU. This is equivalent to a negative logic OR operation. The CPU responds to an interrupt request by enabling the interrupt acknowledge line. This signal is received by device 1 at its PI (priority in) input. The acknowledge signal passes on to the next device through the PO (priority out) output only if device 1 is not requesting an interrupt. If device 1 has a pending interrupt, it blocks the acknowledge signal from the next device by placing a 0 in the PO output. It then proceeds to insert its own interrupt vector address (VAD) vector address (VAD) into the data bus for the CPU to use during the interrupt cycle.

A device with a 0 in its PI input generates a 0 in its PO output to inform the next-lower-priority device that the acknowledge signal has been blocked. A device that is requesting an interrupt and has a 1 in its PI input will intercept the acknowledge signal by placing a 0 in its PO output. If the device does not have pending interrupts, it transmits the acknowledge signal to the next device by placing a 1 in its PO output. Thus the device with PI = 1 and PO = 0 is the one with the highest priority that is requesting an interrupt, and this device places its VAD on the data bus. The daisy chain arrangement gives the highest priority to the device that receives the interrupt acknowledge signal from the CPU. The farther the device is from the first position, the lower is its priority.



Q.3. (b) Explain BCD subtraction using 9's complement for (87 – 39).

Ans. 9's complement of 39 is 60

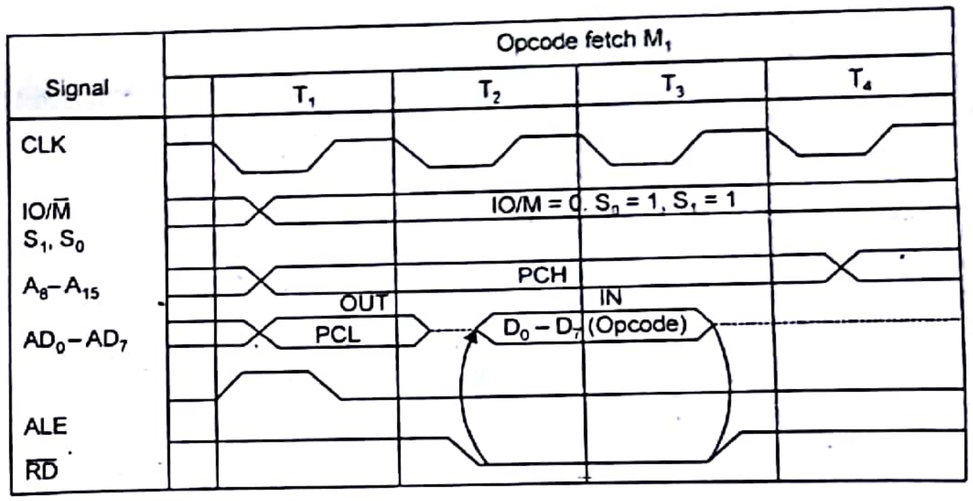
Adding 60 to 87

$$\begin{array}{r} 87 \\ +60 \\ \hline 147 \\ \quad\;\; 1 \\ \hline 48 \end{array}$$

BCD Subtraction using 9's complement of (87-39) is 48

Q.4. (a) Draw and explain the timing diagram for fetch operation ?

Ans. Opcode fetch machine cycle



The Opcode fetch cycle, fetches the instructions from memory and delivers it to the instruction register of the microprocessor. For any instruction cycle, Opcode fetch is the first machine cycle. We know that each machine cycle may have 3 to 6 T-states. This Opcode fetch machine cycle consists of 4 T-states.

T1 State: During the T1 state, the contents of the program counter are placed on the 16 bit address bus. The higher order 8 bits are transferred to address bus (A8-A15) and lower order 8 bits are transferred to multiplexed A/D (AD0-AD7) bus.

After the address bits are transferred, the ALE (address latch enable) signal goes high. As soon as ALE goes high, the memory latches the AD0-AD7 bus. At the middle of the T state the ALE goes low and the complete 16-bit address is made available for the Opcode fetch machine cycle.

T2 State: During the beginning of this state, the RD' signal goes low to enable memory. It is during this state, the selected memory location is placed on D0-D7 of the Address/Data multiplexed bus.

T3 State: In the previous state the Opcode is placed in D0-D7 of the A/D bus. In this state of the cycle, the Opcode of the A/D bus is transferred to the instruction register of the microprocessor. Now the RD' goes high after this action and thus disables the memory from A/D bus.

T4 State: In this state the Opcode which was fetched from the memory is decoded.

Thus the cycle completes after 4 T-states.

Q.4. (b) The content of accumulator are 93H and the contents of register C are B7H, add both contents.

Ans. If accumulator content = 93H and content of register C = B7H, then addition of two will be as:

			CY	D7	D6	D5	D4	D3	D2	D1	D0
A	= 93H	=		1	0	0	1	0	0	1	1
C	= B7	=		1	0	1	1	0	1	1	1
			1		1	1		1	1	1	← Carry
Sum (A)	= 1 4AH	=	1	0	1	0	0	1	0	1	0
	CY										

Status of the flag register after addition:

Flag Register	D7	D6	D5	D4	D3	D2	D1	D0
	S 0	Z 0	–	AC 0	–	P 0	–	CY 0

SECOND TERM EXAMINATION [APRIL 2015]
FOURTH SEMESTER [B.TECH]
COMPUTER ORGANIZATION AND ARCHITECTURE [ETCS-204]

Time. 1 Hour **MM : 30**

Note: *Q.No.*1 *is compulsory. Attempt any two more questions from the rest.*

Q.1. (*a*) Define the following: **(2)**

(*i*) Microoperation:-

Ans. A microoperation is an elementary operation performed with the data stored in registers. The microoperations in digital computers are classified into four categories:-

1. Register transfer microoperations which transfer binary information from one register to another.

2. Arithmetic microoperation which performs arithmetic operations on numeric data stored in registers.

3. Logic microoperations which perform bit manipulation operations on non-numeric data stored in registers.

4. Shift microoperation which perform shift operations on data stored in registers.

(ii) Microinstruction:

Ans. Each word in control memory contains within it a microinstruction. The microinstruction specifies one or more microoperations for the system. A sequence of microinstructions constitutes a microprogram.

The Microinstruction code format for the control memory is :

3	3	3	2	2	7
F1	F2	F3	CD	BR	AD

Here,

F1,F2,F3 : Microoperation fields

CD : Condition for Branching

Br : Branch field

AD: Address field

Q.1. (*b*) Express (3 x 4) + (5 x 6) to reserve polish notation. **(2)**

Ans.

PostFix	Infix	Answer
3 4 +	3 + 4	7
3 5 6 + *	(5 + 6) * 3	33
2 4 / 5 6 – *	(2/4)*(5-6)	– 0.5
2 3 ↑	2^3	8

Q.1. (c) How is logical memory address different from physical memeory address. **(3)**

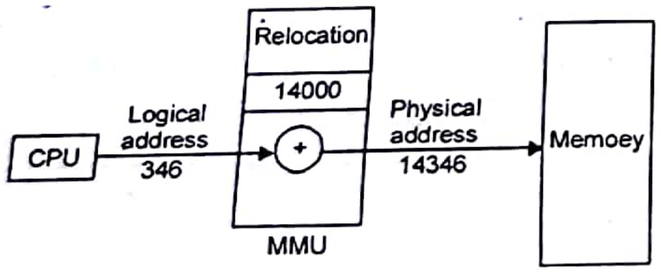
Ans. (1) An address generated by the CPU is commonly referred to as a logical a logical address. The set of all logical addresses generated by a program is known as logical address space. Whereas, an address seen by the memory unit- that is, the one loaded into the memory-address register of the memory- is commonly referred to as physical address. The set of all physical addresses corresponding to the logical addresses is known as physical address space.

(2) The compile-time and load-time address-binding methods generate identical logical and physical addresses. However, in the execution-time address-binding scheme, the logical and physical-address spaces differ.

(3) The user program never sees the physical addresses. The program creates a pointer to a logical address, say 346, stores it in memory, manipulate it, compares it to other logical addresses–all as the number 346. Only when a logical address is used as memory address, it is relocated relative to the base/relocation register. The memory-mapping hardware device called the memory- management unit(MMU) converts logical addresses into physical addresses.

(4) Logical addresses range from 0 to max. User program that generates logical address thinks that the process runs in locations 0 to max. Logical addresses must be mapped to physical addresses before they are used. Physical addresses range from (R+0) to (R + max) for a base/relocation register value R.

(5) Example:



Mapping from logical to physical addresses using memory management unit (MMU) and relocation/base register. The value in relocation/base register is added to every logical address generated by a user process, at the time it is sent to memory, to generate corresponding physical address.

In the above figure, base/ relocation value is 14000, then an attempt by the user to access the location 346 is mapped to 14346.

Q.1. (d) Differentiate between "hit" and "miss" with respect to cache memory. **(2)**

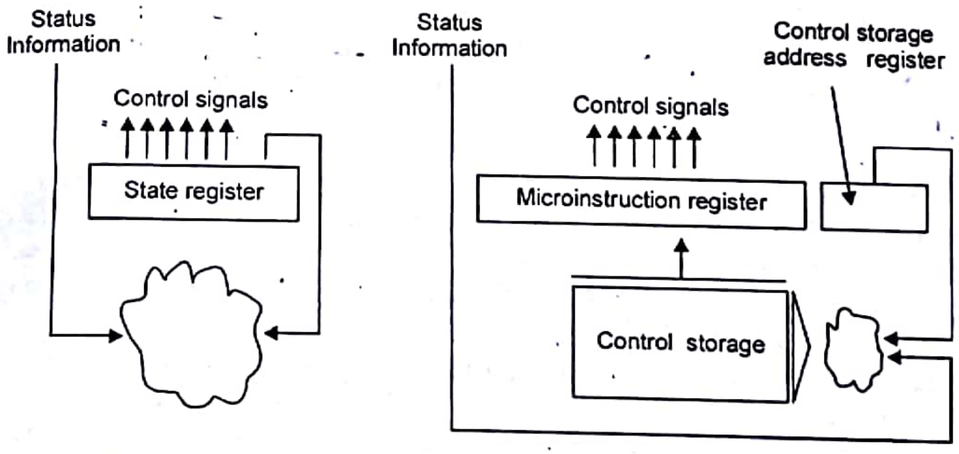
Ans. The performance of cache memory is frequently measured in terms of a quantity called hit ratio. When the CPU refers to memory and finds the word in cache , it is said to produce a hit.

If the word is not found in cache, it is in main memory and it counts as a miss.

Q.1. (e) Explain the difference between hardwired control and microprogrammed control.

Ans. Micro programmed control: Micro programmed control is a control mechanism to generate control signals by using a memory called control storage (CS), which contains the control signals. Although micro programmed control seems to be advantageous to CISC machines, since CISC requires systematic development of sophisticated control signals, there is no intrinsic difference between these 2 control mechanisms.

Hard-wired control: Hardwired control is a control mechanism to generate control signals by using appropriate finite state machine (FSM). The pair of "microinstruction-register" and "control storage address register" can be regarded as a "state register" for the hardwired control. Note that the control storage can be regarded as a kind of combinational logic circuit. We can assign any 0, 1 values to each output corresponding to each address, which can be regarded as the input for a combinational logic circuit. This is a truth table.



(a) Hardwired control (b) Microprogrammed control

Q.2. (a) An instruction is stored at location 300 with its address field at location 301. The address field has the value 400. A processor register R1 contains the number 200. Evaluate the effective address if the addressing mode of the instruction is:

(i) Direct (ii) Immediate (iii) Relative (iv) Register Indirect (v) Index with R1 as the index register. **(5)**

Ans. Location _ Contents

300 _ opcode; the instruction operation code

301 _ 400; address field of the above instruction

(a) Direct addressing

Direct addressing means that the address field contains the address of memory location the instruction is supposed to work with (where an operand "resides").

Effective address would therefore be 400.

(b) Immediate addressing

Immediate addressing means that the address field contains the operand itself.

Effective address would therefore be 301.

(c) Relative addressing

Relative addressing means that the address field contains offset to be added to the program counter to address a memory location of the operand.

Effective address would therefore be 301 + 400 = 701.

(d) Register indirect addressing

Register indirect addressing means that the address of an operand is in the register. The address field in this case contains just another operand.

Effective address would therefore be in R1 = 200.

(e) Indexed addressing with R1 as index register

There are several possible indexed addressing modes but in this case (there is an address field) it is co called "indexed absolute" addressing.

In indexed absolute addressing the effective address is calculated by taking the contents of the address field and adding the contents of the index register.

Effective address would therefore be 400 + R1 = 400 + 200 = 600.

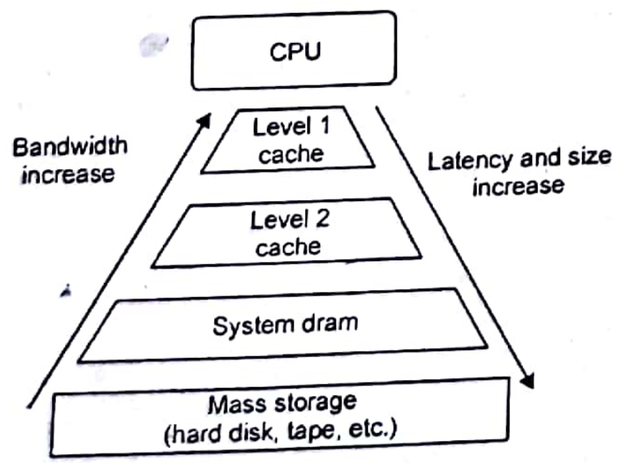
Q.2 (b) Discuss Memory hierarchy?

Ans. The hierarchical arrangement of storage in current computer architectures is called the memory hierarchy. It is designed to take advantage of memory locality in computer programs. Each level of the hierarchy is of higher speed and lower latency, and is of smaller size, than lower levels.

Most modern CPUs are so fast that for most program workloads the locality of reference of memory accesses, and the efficiency of the caching and memory transfer between different levels of the hierarchy, is the practical limitation on processing speed. As a result, the CPU spends much of its time idling, waiting for memory I/O to complete.

Memory Hierarchy

The memory hierarchy in most computers is as follows:



- Processor registers - fastest possible access (usually 1 CPU cycle), only hundreds of bytes in size
- Level 1 (L1) cache - often accessed in just a few cycles, usually tens of kilobytes
- Level 2 (L2) cache - higher latency than L1 by 2× to 10×, often 512KB or more
- Level 3 (L3) cache - (optional) higher latency than L2, often multiple MB's
- Main memory (DRAM) - may take hundreds of cycles, but can be multiple gigabytes
- Disk storage - hundreds of thousands of cycles latency, but very large

Q.3 (a) What is an instruction format? Explain different types of instruction format? **(5)**

Ans: It is the function of the control unit within the CPU to interpret each instruction code and provide the necessary control functions needed to process the instruction The bits of the instruction are divided into group called fields. The most common fields found found in instruction formats are:-

1. An operation code field that specifies the operation to be performed.
2. An address field that designated a memory address or a processor register.
3. A mode field that specifies the way the operand or the effective address is determined.

Opcode	Mode	Address

Computers may have instructions of several different lengths containing varying number of addresses. The number of address field in the instruction format of a computer depends on the internal organization of its registers. Most computers fall into one of three types of CPU organization.

(1) Single Accumulator organization ADD X AC ® AC + M [×]

(2) General Register Organization ADD R1, R2, R3 R ® R2 + R3

(3) Stack Organization PUSH X

Three address Instruction

Computer with three addresses instruction format can use each address field to specify either processor register are memory operand.

ADD R1, A, B A1 ® M [A] + M [B]

ADD R2, C, D R2 ® M [C] + M [B] X = (A + B) * (C + A)

MUL X, R1, R2 M [X] R1 * R2

The advantage of the three address formats is that it results in short program when evaluating arithmetic expression. The disadvantage is that the binary-coded instructions require too many bits to specify three addresses.

Two Address Instruction

Most common in commercial computers. Each address field can specify either a processes register on a memory word.

MOV R1, A R1 ® M [A]

ADD R1, B R1 ® R1 + M [B]

MOV	R2, C	R2 ® M [C]	X = (A + B) * (C + D)
ADD	R2, D	R2 ® R2 + M [D]	
MUL	R1, R2	R1 ® R1 * R2	
MOV	X1 R1	M [X] ® R1	

One Address instruction

It used an implied accumulator (AC) register for all data manipulation. For multiplication/division, there is a need for a second register.

LOAD	A	AC ® M [A]	
ADD	B	AC ® AC + M [B]	
STORE	T	M [T] ® AC	X = (A+B) × (C + A)

All operations are done between the AC register and a memory operand. It's the address of a temporary memory location required for storing the intermediate result.

LOAD	C	AC ® M (C)	
ADD	D	AC ® AC + M (D)	
ML	T	AC ® AC + M (T)	
STORE	X	M [×] ® AC	

Zero – Address Instruction

A stack organized computer does not use an address field for the instruction ADD and MUL. The PUSH & POP instruction, however, need an address field to specify the operand that communicates with the stack (TOS ® top of the stack)

PUSH	A	TOS ® A
PUSH	B	TOS ® B
ADD		TOS ® (A + B)
PUSH	C	TOS ® C
PUSH	D	TOS ® D
ADD		TOS ® (C + D)
MUL		TOS ® (C + D) * (A + B)
POP	X	M [X] TOS

Q.3.(*b*) Explain BUN (Branch unconditionally) and BSA (Branch and save return address) instruction with example.

Ans. BUN: Branch Unconditionally

This **instruction** has the responsibility to transfer the entire program t the **instruction** which is specified by the effective address. We now know that program counter (PC) holds the address of the instruction to be read from the memory and program is a set of **instructions** to be carried out to accomplish the particular task. BUN instruction allows the programmer to specify an instruction out of the program and modify the program.

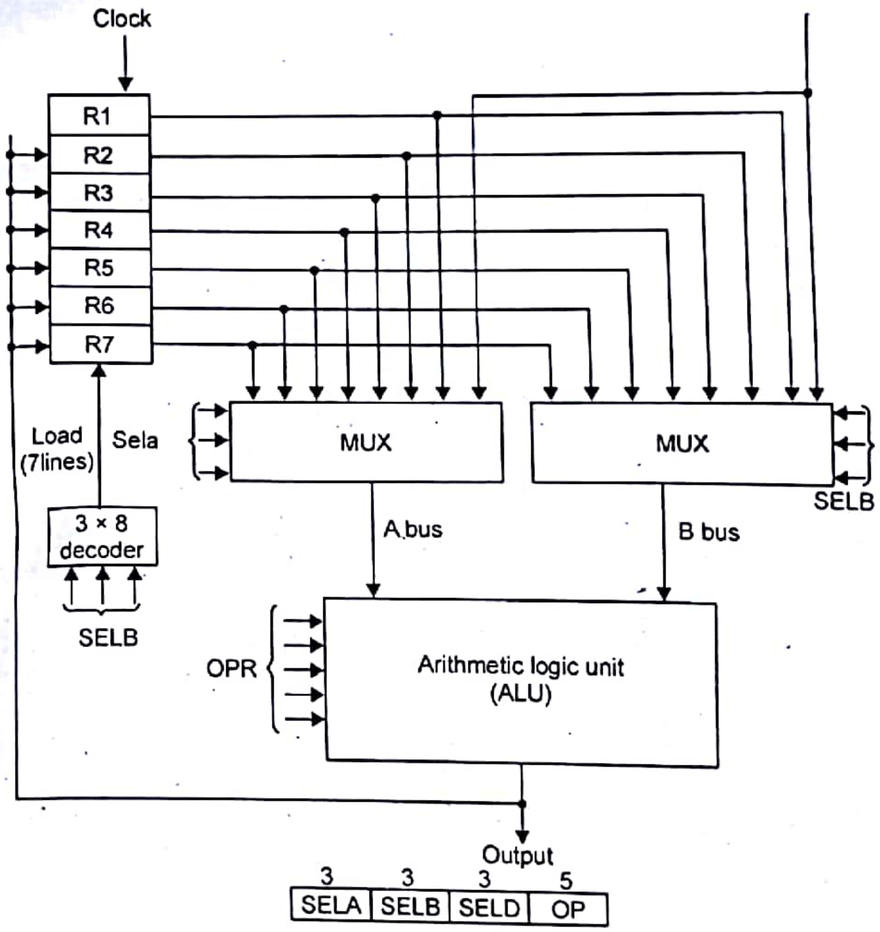
BSA : Branch and Save Return Address

As the name tells the function of this **instruction**, it allows the branching in th execution of instruction. By branching we mean that the **instructions** can have sub routin

or procedure. When this instruction is executed, it stores the address of the next instruction to be executed as PC (Program counter).

Q.4. (a) Draw the general register organization of basic computer, having seven registers. Also show all connections with Arithmetic Logical Unit (ALU) and control word. **(4)**

Ans.



Control word

Q.4. (*b*) Explain LRU and FIFO page replacement algorithm.

Ans. Least Recently Used (LRU) Page Replacement Algorithm

In this algorithm, the page that has not been used for the longest period of time has to be replaced.

Advantages of LRU Page Replacement Algorithm:

1. It is amenable to full statistical analysis.

2. Never suffers from Belady's anomaly.

Most Frequency (MFU) Used Page Replacement Algorithm

Actually MFU algorithm thinks that the page which was used most frequently will not be needed immediately so it will replace the MFU page

Example: Consider the following reference string:7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

Buffer size:3 String : 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

7 7 7 2 2 2 0 4 2 2 0 0 2 2 2 0 0 7 7 7

0 0 0 0 3 3 3 3 3 3 3 3 3 3 3 3 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

END TERM EXAMINATION [JUNE 2015]
FOURTH SEMESTER [B.TECH]
COMPUTER ORGANIZATION AND ARCHITECTURE [ETCS-204]

Time. 3 Hours **M.M. : 75**

Note: *Attempt any five question including Q no. 1 which is compulsory. Internal choice is indicated.*

Q.1. (a) What do you mean by base in number system?

Ans. In mathematical numeral systems, the **radix** or **base** is the number of unique digits, including zero, used to represent numbers in a positional numeral system. For example, for the decimal system (the most common system in use today) the radix is ten, because it uses the ten digits from 0 through 9.

In any standard positional numeral system, the number x and its base y are conventionally written as $(x)_y$ although for base ten the subscript is usually assumed and not written, as it is the most common way to express value. For example, $(100)_{10}$ (in the decimal system) represents the number one hundred, while $(100)_2$ (in the binary system with base 2) represents the number four.

Q.1.(b) What is Accumulator?

Ans. In a computer's central processing unit (CPU), an **accumulator** is a register in which intermediate arithmetic and logic results are stored.

Without a register like an accumulator, it would be necessary to write the result of each calculation (addition, multiplication, shift, etc.) to main memory, perhaps only to be read right back again for use in the next operation. Access to main memory is slower than access to a register like the accumulator because the technology used for the large main memory is slower (but cheaper) than that used for a register. Early electronic computer systems were often split into two groups, those with accumulators and those without.

Modern computer systems often have multiple general purpose registers that operate as accumulators, and the term is no longer as common as it once was. However, a number of special-purpose processors still use a single accumulator for their work, in order to simplify their design

Q.1.(c) What is the role of implied mode in addressing mode?

Ans. Implied mode: In this mode the operands are specified implicitly in the definition of the instruction. For example, the instruction "complement accumulator" is an implied mode instruction because the operand in the accumulator register is implied in the definition of the instruction.

Q1.(d) What is handshake problem?

Ans. Various handshaking problems are in circulation, the most common one being the following. In a room of n people, how many different handshakes are possible?

The answer is $\binom{n}{2} = n(n-1)/2$. To see this, enumerate the people present, and consider one person at a time. The first person may shake hands with $n-1$ other people. The next person may shake hands with $n-2$ other people, not counting the first person

again. Continuing like this gives us a total number of $(n-1)+(n-2)+....+2+1$ handshakes, which is exactly the answer given above.

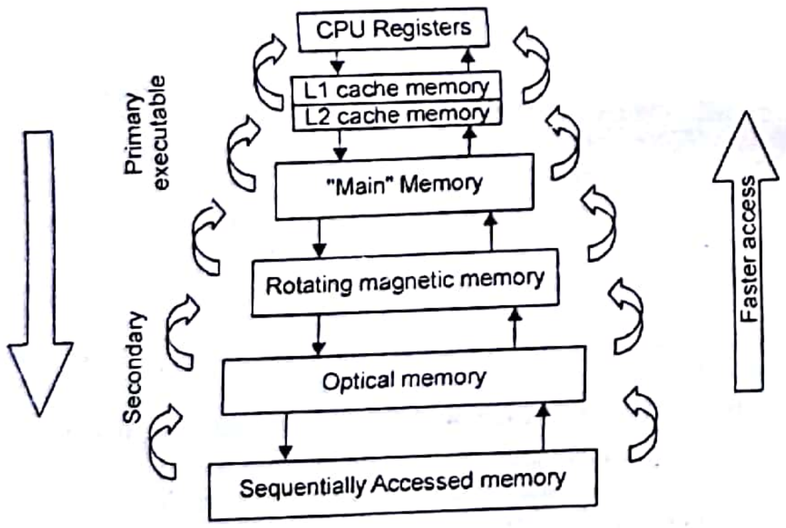
Another popular handshake problem starts out similarly with $n > 1$ people at a party. Not being able to shake hands with yourself, and not counting multiple handshakes with the same person, the problem is to show that there will always be two people at the party, who have shaken hands the same number of times.

The solution to this problem uses Dirichlet's box principle. If there exists a person at the party, who has shaken hands zero times, then every person at the party has shaken hands with at ssmost $n-2$ other people. Since there are $n-1$ possible handshakes (from 0 to $n-2$), among peoplssssse there must be two who have shaken hands the same amount of times. If there exists no person, who has shaken hands zero times, then all of the party guests have shaken hands at least once. This also amounts to possible handshakes (from 1 to $n-1$).

Q1.(e) Explain the memory hierarchy with performace?

Ans. The hierarchical arrangement of storage in current computer architectures is called the memory hierarchy. It is designed to take advantage of memory locality in computer programs. Each level of the hierarchy is of higher speed and lower latency, and is of smaller size, than lower levels.

Most modern CPUs are so fast that for most program workloads the locality of reference of memory accesses, and the efficiency of the caching and memory transfer between different levels of the hierarchy, is the practical limitation on processing speed. As a result, the CPU spends much of its time idling, waiting for memory I/O to complete.



Memory Hierarchy

The memory hierarchy in most computers is as follows:

* Processor registers - fastest possible access (usually 1 CPU cycle), only hundreds of bytes in size
* Level 1 (L1) cache - often accessed in just a few cycles, usually tens of kilobytes
* Level 2 (L2) cache - higher latency than L1 by 2× to 10×, often 512KB or more
* Level 3 (L3) cache - (optional) higher latency than L2, often multiple MB's
* Main memory (DRAM) - may take hundreds of cycles, but can be multiple gigabytes
* Disk storage - hundreds of thousands of cycles latency, but very large.

Q.1.(f) What is the transfer rate of an 8-track magnetic tape whose speed is 120 inches/second and density is 1600 bits/inch?

Ans. We Know, Transfer Rate = Number of bytes on a track × Rotation time

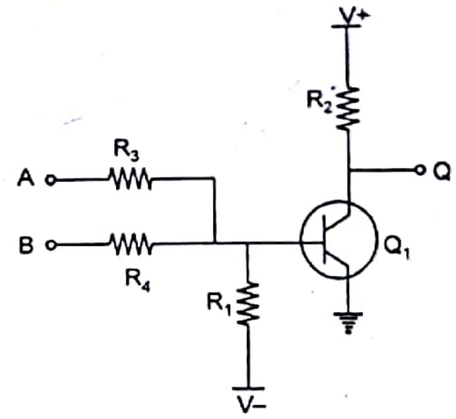
Here, Number of Bytes on a track = 1600 bits/inch

Rotation Time = 120 inches/sec.

Therefore, Transfer rate = 1600 x 120 = 192000ms/track

Q1. (g) What are the advantages of RTL?

Ans. Resistor-transistor logic (RTL) is a class of digital circuits built using resistors as the input network and bipolar junction transistors (BJTs) as switching devices. RTL is the earliest class of transistorized digital logic circuit used; other classes include diode-transistor logic (DTL) and transistor-transistor logic (TTL).



Schematic of basic two-input RTL NOR gate

The primary advantage of RTL technology was that it used a minimum number of transistors. In circuits using discrete components, before integrated circuits, transistors were the most expensive component to produce. Early IC logic production (such as Fairchild's in 1961) used the same approach briefly, but quickly transitioned to higher-performance circuits such as diode-transistor logic and then transistor-transistor logic (starting 1963 at Sylvania), since diodes and transistors were no more expensive than resistors in the IC.

Q.1.(h) Write assembly language program to subtract two 8 bit numbers?

Ans.

```
; 2000H = 24
; 2001H = 43
LDA 2000H ;
MOV B,A ;
LDA 2001H ;
SUB B ;
DAA ;
STA 2002H ;
HLT ;
```

Q.1. (i) Difference between a direct and indirect instruction?

Ans. Direct addressing has the address of a memory location as the operand in an instruction. For instance the instruction MOV R0 00H, R0, when converted to machine language is the physical address of register R0. The instruction moves 0 to R0.

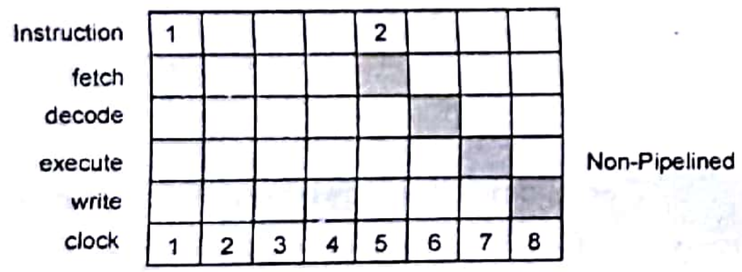
Indirect addressing has a pointer to the memory location as the operand in an instruction. For instance the instruction MOV @R0 00H, when converted to machine language, @R0 becomes whatever is stored in R0, and that is the address used to move 0 to. It can be whatever is stored in R0.

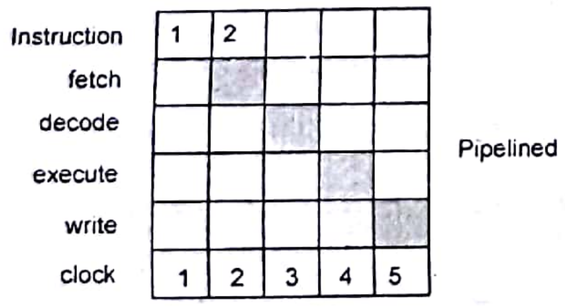
Q.1. (j) How performance of instruction improved using pipelining?

Ans. Pipelining is a technique used to improve the execution throughput of a CPU by using the processor resources in a more efficient manner.

The basic idea is to split the processor instructions into a series of small independent stages. Each stage is designed to perform a certain part of the instruction. At a very basic level, these stages can be broken down into:

- **Fetch Unit** Fetch an instruction from memory
- **Decode Unit** Decode the instruction be executed
- **Execute Unit** Execute the instruction
- **Write Unit** Write the result back to register or memory



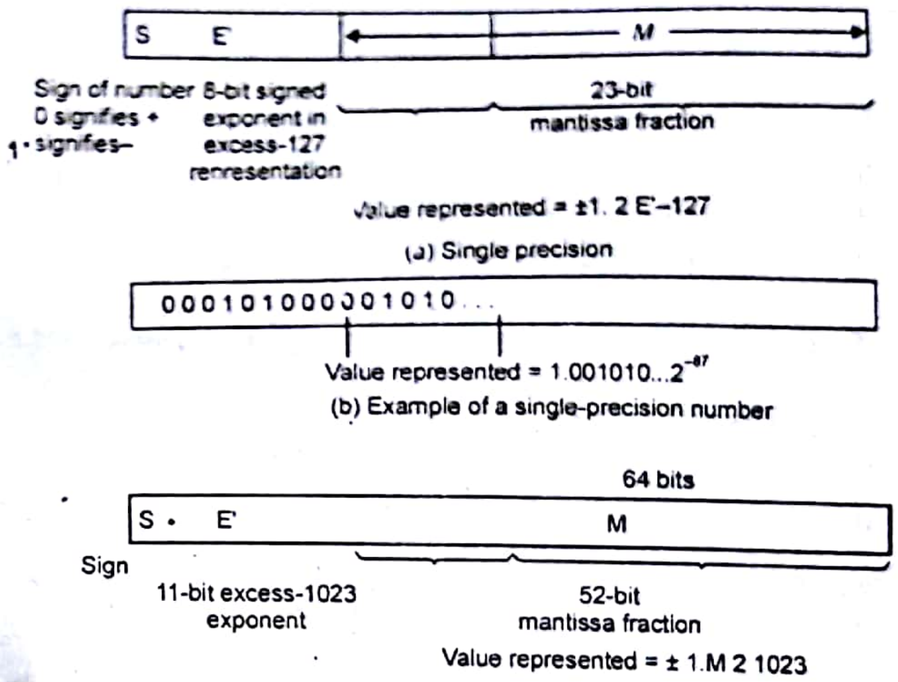


On a non-pipelined CPU, when a instruction is being processed at a particular stage, the other stages are at an idle state – which is very inefficient. If you look at the diagram, when the 1st instruction is being decoded, the Fetch, Execute and Write Units of the CPU are not being used and it takes 8 clock cycles to execute the 2 instructions.

On the other hand, on a pipelined CPU, all the stages work in parallel. When the 1st instruction is being decoded by the Decoder Unit, the 2nd instruction is being fetched by the Fetch Unit. It only takes 5 clock cycles to execute 2 instructions on a pipelined CPU. Increasing the number of stages in the pipeline will not always result in an increase of the execution throughput.

Q.2 (a) Represent floating point in IEEE standard format for given No. $1.001010\_0\times2^{-87}$ and explain difference in Single precision and double precision numbers? **(8)**

Ans.



Single Precision

The IEEE single precision floating point standard representation requires a 32 bit word, which may be represented as numbered from 0 to 31, left to right.

- The first bit is the **sign** bit, S,
- the next eight bits are the **exponent** bits, 'E', and
- the final 23 bits are the **fraction** 'F':

S EEEEEEEE FFFFFFFFFFFFFFFFFFFFFFF
0 1 8 9 31

The value V represented by the word may be determined as follows:

- If E=255 and F is nonzero, then V=NaN ("Not a number")
- If E=255 and F is zero and S is 1, then V=-Infinity
- If E=255 and F is zero and S is 0, then V=Infinity
- If 0<E<255 then V=(-1)**S * 2 ** (E-127) * (1.F) where "1.F" is intended to represent the binary number created by prefixing F with an implicit leading 1 and a binary point.
- If E=0 and F is nonzero, then V=(-1)**S * 2 ** (-126) * (0.F). These are "unnormalized" values.
- If E=0 and F is zero and S is 1, then V=-0
- If E=0 and F is zero and S is 0, then V=0

In particular,

0 00000000 00000000000000000000000 = 0

1 00000000 00000000000000000000000 = -0

0 11111111 00000000000000000000000 = Infinity
1 11111111 00000000000000000000000 = -Infinity

0 11111111 00000100000000000000000 = NaN
1 11111111 00100010001001010101010 = NaN

0 10000000 00000000000000000000000 = +1 * 2**(128-127) * 1.0 = 2
0 10000001 10100000000000000000000 = +1 * 2**(129-127) * 1.101 = 6.5
1 10000001 10100000000000000000000 = -1 * 2**(129-127) * 1.101 = -6.5

0 0000000 00000000000000000000000 = +1 * 2**(1-127) * 1.0 = 2**(-126)
0 00000000 10000000000000000000000 = +1 * 2**(-126) * 0.1 = 2**(-127)
0 00000000 00000000000000000000001 = +1 * 2**(-126) *
0.00000000000000000000001 =
2**(-149) (Smallest positive value)

Double Precision

The IEEE double precision floating point standard representation requires a 64 bit word, which may be represented as numbered from 0 to 63, left to right.

- The first bit is the **sign** bit, S,
- the next eleven bits are the **exponent** bits, 'E', and
- the final 52 bits are the **fraction** 'F':

S EEEEEEEEEEE
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
0 1 11 12
63

The value V represented by the word may be determined as follows:

- If E=2047 and F is nonzero, then V=NaN ("Not a number")
- If E=2047 and F is zero and S is 1, then V=-Infinity
- If E=2047 and F is zero and S is 0, then V=Infinity
- If 0<E<2047 then V=(– 1)**S * 2 ** (E – 1023) * (1.F) where "1.F" is intended to represent the binary number created by prefixing F with an implicit leading 1 and a binary point.
- If E=0 and F is nonzero, then V=(– 1)**S * 2 ** (– 1022) * (0.F) These are "unnormalized" values.
- If E=0 and F is zero and S is 1, then V = – 0
- If E=0 and F is zero and S is 0, then V = 0

Q.2 (b) An 8 bit register contains binary value 10011100. What is register value after arithmetic shift right? Starting from initial no. 10011100 determine register value after an arithmetic shift left and state whether this is overflow. (6.5)

Ans. R = 10011100

Arithmetic shift right: 11001110

Arithmetic shift left: 00111000 overflow because a negative number changed to positive

Q.3 (a) What do you mean by bus arbitration? What are the advantages and disadvantages of using parallel arbitration? (6)

Ans. Bus Arbitration

So far we have seen the operation of the bus from the master's point of view and using only one master on the bus.

The I2C bus was originally developed as a multi-master bus. This means that more than one device initiating transfers can be active in the system.

When using only one master on the bus there is no real risk of corrupted data, except if a slave device is malfunctioning or if there is a fault condition involving the SDA / SCL bus lines.

When MCU 1 issues a start condition and sends an address, all slaves will listen (including MCU 2 which at that time is considered a slave as well). If the address does not match the address of CPU 2, this device has to hold back any activity until the bus becomes idle again after a stop condition.

As long as the two MCU's monitor what is going on on the bus (start and stop) and as long as they are aware that a transaction is going on because the last issued command was not a STOP, there is no problem.

Let's assume one of the MCU's missed the START condition and still thinks the bus is idle, or it just came out of reset and wants to start talking on the bus which could very well happen in a real-life scenario. This could lead to problems.

Fortunately, the physical bus setup helps us out. Since the bus structure is a wired AND (if one device pulls a line low it stays low), you can test if the bus is idle or occupied.

When a master changes the state of a line to HIGH, it MUST always check that the line really has gone to HIGH. If it stays low then this is an indication that the bus is occupied and some other device is pulling the line low.

Therefore the general rule of thumb is: If a master can't get a certain line to go high, it lost arbitration and needs to back off and wait until a stop condition is seen before making another attempt to start transmitting.

Since the previous rule says that a master loses arbitration when it cannot get either SCL or SDA to go high when needed, this problem does not exist. It is the device that is sending the '0' that rules the bus. One master cannot disturb the other master's transmission because if it can't detect one of the lines to go high, it backs off, and if it is the other master that can't do so, it will behave the same.

This kind of back-off condition will only occur if the two levels transmitted by the two masters are not the same. Therefore, let's have a look at the following figure, where two MCUs start transmitting at the same time:

The two MCU's are accessing a slave in write mode at address 1111001. The slave acknowledges this. So far, both masters are under the impression that they "own" the bus.

Now MCU1 wants to transmit 01010101 to the slave, while MCU 2 wants to transmit 01100110 to the slave. The moment the data bits do not match anymore (because what the MCU sends is different than what is present on the bus) one of them loses arbitration and backs off. Obviously, this is the MCU which did not get its data on the bus. For as long as there has been no STOP present on the bus, it won't touch the bus and leave the SDA and SCL lines alone (yellow zone).

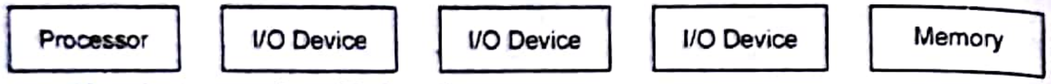
The moment a STOP was detected, MCU2 can attempt to transmit again.

From the example above we can conclude that is the master that is pulling the line LOW in an arbitration situation that always wins the arbitration. The master which wanted the line to be HIGH when it is being pulled low by the other master loses the bus. We call this a loss of arbitration or a back-off condition.

When a MCU loses arbitration, it has to wait for a STOP condition to appear on the bus. Then it knows that the previous transmission has been completed.

Advantages of Buses

- **Versatility:**

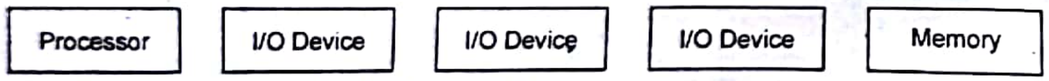


– New devices can be added easily

– Peripherals can be moved between computer systems that use the same bus standard

- Low Cost:
- A single set of wires is shared in multiple ways
- Manage complexity by partitioning the design

Disadvantage of Buses



- It creates a communication bottleneck

– The bandwidth of that bus can limit the maximum I/O throughput

- The maximum bus speed is largely limited by:

– The length of the bus

– The number of devices on the bus

– The need to support a range of devices with:

- Widely varying latencies

Widely varying data transfer rates

Q.3 (b) Register A holds the 8-bit binary value 11011001. Determine the B operand and the logic micro-operation to be performed in order to change the value in A to: **(6.5)**

a. 01101101

b. 11111101

(a) 01101101

11011001
XOR
10110100
01101101

(b) 11111101

11011001
OR
111111101
11111101

Q.4 (a) What are the features of 8085 microprocessor? Explain addressing mode of 8085.

Ans. (1) 8085 microprocessor is an 8 bit microprocessor. I.e. it can accept or provide 8 bit data simultaneously.

(2) 8085 microprocessor is a single chip, NMOS device implemented with 6200 transistors.

(3) 8085 microprocessor requires a single +5V DC power supply.

(4) 8085 microprocessor provides on chip clock generator, therefore there is no need of external clock generator, but it requires external tuned circuit like LC, RC or crystal.

(5) 8085 microprocessor requires two phase, 50% duty cycle, TTL clock. These clock signals are generated by an internal clock generator.

(6) The maximum clock frequency of 8085 microprocessor is 3MHz where as minimum clock frequency is 500 KHz.

(7) 8085 provides 74 instructions with the following addressing modes:

- register
- direct
- immediate
- indirect
- implied.

(8) The data bus is multiplexed with the address bus, hence it requires external hardware to separate data lines from address lines (this is one of the disadvantage of 8085).

(9) 8085 microprocessor provides 16 address lines, therefore it can access 2^16 = 64K bytes of memory.

(10) It generates 8 bit I/O address, hence it can access 2^8 = 256 input ports and 256 output ports.

(11) It performs the following arithmetic and logical operations.

- 8 bit, 16 bit binary addition
- 2 digit BCD addition
- 8 bit binary subtraction
- logical AND, OR, EXOR
- complement and shift operations.

(12) 8085 microprocessor has five hardware interrupts: TRAP, RST 5.5, RST 6.5, RST 7.5, INTR

(13) The hardware interrupt capability of 8085 microprocessor can be increased by providing external hardware.

(14) 8085 microprocessor has capability to share its bus with external bus controller (direct memory access controller); for transferring large amount of data from memory to I/O and vice versa.

(15) 8085 microprocessor provides one accumulator, one flag register, 6 general purpose registers and two special purpose registers.

(16) It provides status for advanced control signals. (Advanced control signals are used in large systems).

(17) 8085 microprocessor can be used to implement three chip microcomputer (8085, 8155, 8355)

16) 8085 microprocessor provides two serial I/O lines which are SOD and SID; it means, serial peripherals can be interfaced with 8085 microprocessor directly.

There are five addressing modes in 8085.

1. Immediate Addressing Mode: - An immediate is transferred directly to the register.

Eg: MVI A, 30H (30H is copied into the register A)

MVI B,40H(40H is copied into the register B).

2. Register Addressing Mode: - Data is copied from one register to another register.

Eg: - MOV B, A (the content of A is copied into the register B)

MOV A, C (the content of C is copied into the register A).

3. Direct Addressing Mode: - Data is directly copied from the given address to the register.

Eg: - LDA 3000H (The content at the location 3000H is copied to the register A).

4. Indirect Addressing Mode: - The data is transferred from the address pointed by the data in a register to other register.

Eg: - MOV A, M (data is transferred from the memory location pointed by the regiser to the accumulator).

5.Implied Addressing Mode: - This mode doesn't require any operand. The data is specified by opcode itself.

Eg: - RAL

CMP

Q4 (b) Derive the control gates associated with the programm counter PC in the basic computer. **(6)**

Ans. (Assume that ZDR = 1 if DR = 0; ZAC = 1, if AC = 0)

$$INR\ (PC) = RT_1 + RT_2 + D_6T_6ZDR + PB_9(FGI) + PB_8(FGO) + rB_4(AC_{15})' + rB_3(AC_{15}) + rB_2ZAC + rB_1E'$$

$$LD\ (PC) = D_4T_4 + D_5T_5$$

$$CLR\ (PC) = RT_1$$

Q.5. (a) An output program resides memory starting from address 2300. It is executed after the computer recognizes an interrupt when FGO becomes as 1 (while IEN=1).

(a) What instruction must be placed at address 1?

(b) What must be the last two instructions of the output program?

Ans. *(a) 0 BUN (2300)10*

(b) ION

1 BUN 0 (Branch indirect with address 0)

Q.5. (b) Write a program loop, using a pointer and a counter, that clears to 0 the contents of hexadecimal locations 500 through 5FF. **(6.5)**

Ans.

```
LDA NBR          /Initialize counter with
CMA                    /2's compl. of NBR
INC
STA CTR          /save -NBR to counter
```



```
LDA ADR          /Save start address to
STA PTR          /PTR
LOP,CLA          /Clear AC
STA PTR I        /Reset memory word
ISZ PTR          /Increment pointer
ISZ CTR          /increment counter
BUN LOP          /Branch to LOP (CTR < 0)
HLT                    /Halt when CTR = 0
NBR,HEX FF             /Nbr of cleared words
CTR,-                  /Counter
ADR,HEX 500      /Start address
PTR,-                  /Pointer
```

Q.6.(a) Differentiate between hardwired control and micro programmed control. It is possible to have a hardwired control associated with a control memory? **(6.5)**

Ans. For each instruction, the control unit causes the CPU to execute a sequence of steps correctly. In reality, there must be control signals to assert lines on various digital components to make things happen. For example, when we perform an Add instruction in assembly language, we assume the addition takes place because the control signals for the ALU are set to "add" and the result is put into the AC. The ALU has various control lines that determine which operation to perform. The question we need to answer is, "How do these control lines actually become asserted?"

We can take one of two approaches to ensure control lines are set properly. The first approach is to physically connect all of the control lines to the actual machine instructions. The instructions are divided up into fields, and different bits in the instruction are combined through various digital logic components to drive the control lines. This is called hardwired control, and is illustrated in figure (1)

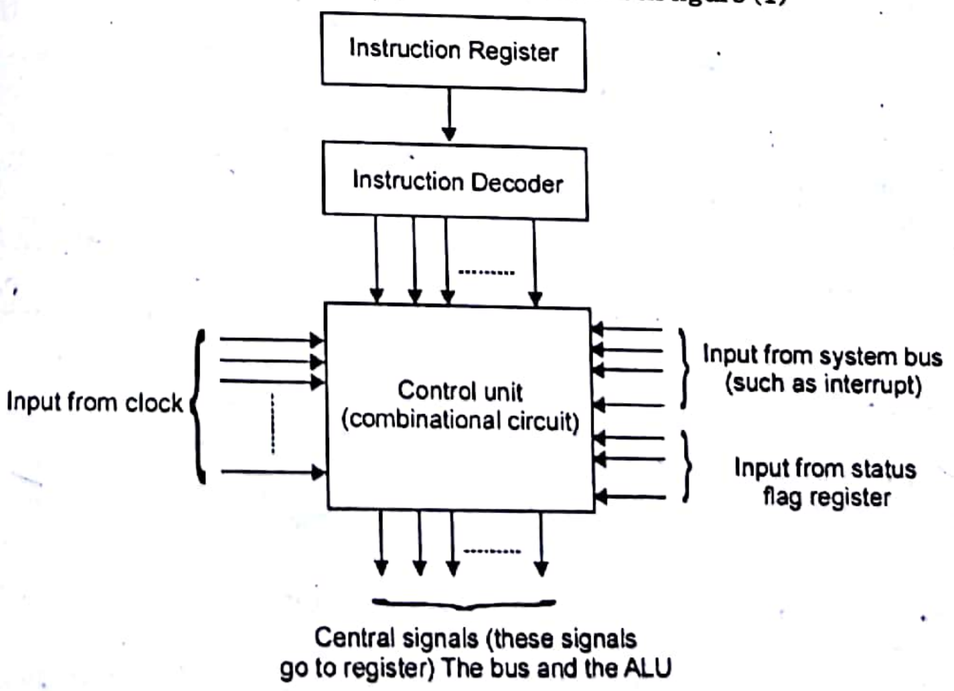


Fig. Hardwired Control Organization

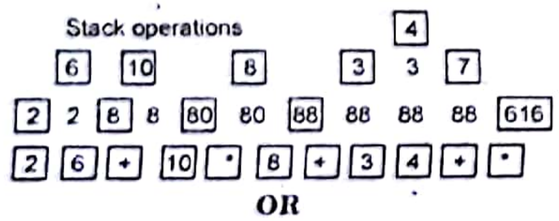
The advantage of hardwired control is that is very fast. The disadvantage is that the instruction set and the control logic are directly tied together by special circuits that are complex and difficult to design or modify. If someone designs a hardwired computer and later decides to extend the instruction set, the physical components in the computer must be changed. This is prohibitively expensive, because not only must new chips be fabricated but also the old ones must be located and replaced.

Microprogramming is a second alternative for designing control unit of digital computer (uses software for control). A control unit whose binary control variables are stored in memory is called a **microprogrammed control unit**. The control variables at any given time can be represented by a string of 1's and 0's called a **control word** (which can be programmed to perform various operations on the component of the system). Each word in control memory contains within it a **microinstruction.** The microinstruction specifies one or more **microoperatiotins** for the system. A sequence of microinstructions constitutes a **microprogram.** A memory that is part of a control unit is referred to as a **control memory.**

Q.6. (b) Convert the following numerical arithmetic expression into reserve polish notation and show stack operation for evaluating result. (3+4)[10(2+6)+8] (6)

Ans. Solution.

26 + 10* 8 + 34+*



OR

Q.7. (a) The memory unit of a computer has 256K words of 32 bits each. The computer has an instruction format with four fields: an operation code field, a mode field, to specify one of seven addressing modes, a register address field to specify one of 60 processor registers, and a memory address. Specify the instruction format and the number of bits in each field if the instruction is one memory word long. (6.5)

Ans. To address 256K we need 18 bits since 256K = 2^18.

To enumerate 7 addressing modes we need 3 bits since 2^3 = 8 what is the first higher number of combinations.

To enumerate 60 registers we need 6 bits since 2^6 = 64 what is the first higher number of combinations.

The rest of bits in the 32-bit word we can use for operation codes (opcodes), 32 - 18 - 3 - 6 = 5. So there can be 2^5 = 32 opcodes.

The instruction format can be:

31...27 - 26...24 - 23...18 - 17...0

opcode - mode - reg_addr - mem_addr

where reg_addr stands for "register address" and mem_addr stands for "memory address"

Q.7. (b) Explain the three types of CPU organization with the help of example? (6)

Ans. Computers may have instructions of several different lengths containing varying number of addresses. The number of address fields in the instruction format of

a computer depends on the internal organization of its registers. Most computers fall into one of the three types of CPU organizations:

(i) **Single Accumulator Organization:** In this type of organization all operations are performed on an implied accumulator. The instruction format uses only one address field. For example, the instruction that loads the accumulator with the contents of a memory location.

Load X

Where X is the address of the source operand. This results in the operation AC ← M (X). AC is the accumulator and M(X) symbolizes the memory word located at address X.

(ii) **General Register Organisation :** In this organization, the instruction format needs 2 or 3 register address fields according to the operation.

For example, an instruction for addition may be written as

ADD R1, R2, R3.

It denotes the operation R1 <-R2 -f- R3

The same ADD instruction needs only two register address fields if the destination register is one of the source registers, i.e. if the operation is

R1 R1 + R2

Then the instruction is ADD R1, R2

The instruction may also contain one memory address field and one register address field. For example, the instruction,

ADD R1, X

Specifies the operation R1 -R1 + M [X]

(iii) **Stack Organization :** In this organization, the computers will have PUSH and POP instructions which require an address field. For Kathplê, the instruction PUSH X will push the word at address Xonto the top of the stack. The operation - type instructions do not need any address field. For example, the instruction

ADD

Consists of only opcode and no address field. It has the effect of popping the top two numbers from the stack, adding them, and pushing the sum onto the stack. Thus all the operands are implied to be in the stack.

Most of the computers fall into one of the above three types of organizations. Some computers combine features from more than one organizational structure.

Q 8. (a) What is cache memory? State the advantages of cache memory? (3)

Ans. Cache memory is a small-sized type of volatile computer memory that provides high-speed data access to a processor and stores frequently used computer programs, applications and data. It stores and retains data only until a computer is powered up.

Cache memory is the fastest memory in a computer. It is typically integrated on the motherboard and directly embedded on the processor or main random access memory (RAM).

Cache memory provides faster data storage and access by storing an instance of programs and data routinely accessed by the processor. Thus, when a processor requests data that already has an instance in the cache memory, it does not need to go to the main memory or the hard disk to fetch the data.

Cache memory can be primary or secondary cache memory, where primary cache memory is directly integrated or closest to the processor. In addition to hardware-based cache, cache memory also can be a disk cache, where a reserved portion on a disk stores and provide access to frequently accessed data/applications from the disk.

Advantages: The advantages of cache memory are as follows:

- Cache memory is faster than main memory.
- It consumes less access time as compared to main memory.
- It stores the program that can be executed within a short period of time.
- It stores data for temporary use.

Q.8. (b) Explain DMA burst-transfer. **(3)**

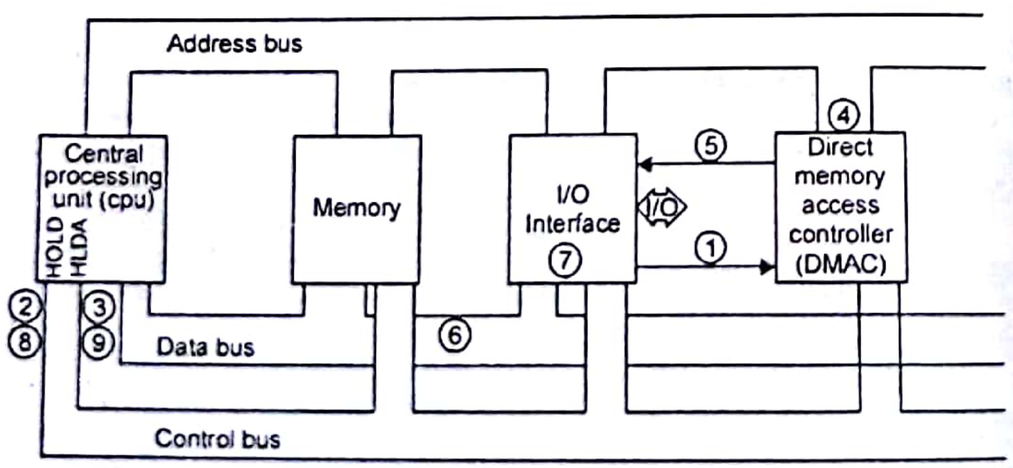
Ans. During any given bus cycle, one of the system components connected to the system bus is given control of the bus. This component is said to be the master during that cycle and the component it is communicating with is said to be the slave. The CPU with its bus control logic is normally the master, but other specially designed components can gain control of the bus by sending a bus request to the CPU. After the current bus cycle is completed the CPU will return a bus grant signal and the component sending the request will become the master.

Taking control of the bus for a bus cycle is called cycle stealing. Just like the bus control logic, a master must be capable of placing addresses on the address bus and directing the bus activity during a bus cycle. The components capable of becoming masters are processors (and their bus control logic) and DMA controllers. Sometimes a DMA controller is associated with a single interface, but they are often designed to accommodate more than one interface.

The 8086 microprocessor receives bus requests through its HOLD pin and issues grants from the hold acknowledge (HLDA) pin. A request is made when a potential master sends a 1 to the HOLD pin. Normally, after the current bus cycle is complete the 8086 will respond by putting a 1 on the HLDA pin. When the requesting device receives this grant signal it becomes the master. It will remain master until it drops the signal to the HOLD pin, at which time the 8086 will drop the grant on the HLDA pin. One exception to the normal sequence is that if a word, which begins at an odd address is being accessed, then two bus cycles are required to complete the transfer and a grant will not be issued until after the second bus cycle.

When a DMA controller becomes master it places an address on the address bus and sends the interface the necessary signals to cause it to put data on, or receive data from, the data bus. Since the DMA controller determines when the bus request is dropped, it can return control to the CPU after each data byte is transferred and then request control again when the next data byte is ready, or it can retain control until the entire block is moved. The former is the usual case because this allows the CPU to continue its work until the next data byte is available.

DMA Block Transfer



Data transfer with a DMA controller

During a block input byte transfer, the following sequence occurs as the data byte is sent from the interface to the memory:

1. The interface sends the DMA controller a request for DMA service.
2. A Bus request is made to the HOLD pin (active High) on the 8086 microprocessor and the controller gains control of the bus.
3. A Bus grant is returned to the DMA controller from the Hold Acknowledge (HLDA) pin (active High) on the 8086 microprocessor.
4. The DMA controller places contents of the address register onto the address bus.
5. The controller sends the interface a DMA acknowledgment, which tells the interface to put data on the data bus. (For an output it signals the interface to latch the next data placed on the bus.)
6. The data byte is transferred to the memory location indicated by the address bus.
7. The interface latches the data.
8. The Bus request is dropped, the HOLD pin goes Low, and the controller relinquishes the bus.
9. The Bus grant from the 8086 microprocessor is dropped and the HLDA pin goes Low.
10. The address register is incremented by 1.
11. The byte count is decremented by 1.
12. If the byte count is non-zero, return to step 1, otherwise stop.

Q8. (c) A block set associative cache memory consist of 128 blocks divided into 4 block sets. **(6.5)**

The main memory consists of 16, 384 blocks and each blocks contains 256 eight bit words.

(i) How many bits are required for addressing the main memory?

(ii) How many bits are needed to represent the TAG SET, WORD fields?

Ans. (i) The cache is divided into 16 sets of 4 lines each. Therefore, 4 bits are needed to identify the set number.

Main memory consists of 4K = 212 blocks. Therefore, the set plus tag lengths must be 12 bits and therefore the tag length is 8 bits.

Each block contains 128 words. Therefore, 7 bits are needed to specify the word.

Main memory address =	TAG	SET	WORD
	8	4	7

(*ii*) 8 left most bits = tag 5 middle bits = line number; 3 rightmost bits = byte number.

TAG	Line	WORD
8	5	3

Q.9. (a) Explain Virtual Memory. **(3)**

Ans. Virtual memory (VM) is a feature developed for the kernel of an operating system (OS) that simulates additional main memory such as RAM (random access memory) or disc storage. This technique involves the manipulation and management of memory by allowing the loading and execution of larger programs or multiple programs simultaneously. It also allows each program to operate as if it had infinite memory, and is often considered more cost effective than purchasing additional RAM.

Virtual memory permits software to use additional memory by utilizing the hard disc drive (HDD) as temporary storage. Most central processing units (CPUs) provide memory management units (MMUs) that support virtual memory. The MMU supports the "page tables" that are used to transform the "real" and "virtual" addresses located in memory and on the HDD.

An OS that uses virtual memory frees up space by transferring data from the HDD which is not immediately required. When the data is needed, it is copied back to the HDD. When all RAM is being used, VM swaps data to the HDD and then back again. Thus, VM allows a larger total system memory; however, complicated code writing is required.

Q.9. (b) How memory management is done. **(3.5)**

Ans. Memory Management

Main Memory refers to a physical memory that is the internal memory to the computer. The word main is used to distinguish it from external mass storage devices such as disk drives. Main memory is also known as RAM. The computer is able to change only data that is in main memory. Therefore, every program we execute and every file we access must be copied from a storage device into main memory.

All the programs are loaded in the main memeory for execution. Sometimes complete program is loaded into the memory, but some times a certain part or routine of the program is loaded into the main memory only when it is called by the program, this mechanism is called **Dynamic Loading**, this enhance the performance.

Also, at times one program is dependent on some other program. In such a case, rather than loading all the dependent programs, CPU links the dependent programs to the main executing program when its required. This mechanism is known as **Dynamic Linking**.

Swapping: A process needs to be in memory for execution. But sometimes there is not enough main memory to hold all the currently active processes in a timesharing system. So, excess process are kept on disk and brought in to run dynamically. Swapping is the process of bringing in each process in main memory, running it for a while and then putting it back to the disk.

Contiguous Memory Allocation: In contiguous memory allocation each process is contained in a single contiguous block of memory. Memory is divided into several fixed size partitions. Each partition contains exactly one process. When a partition is free, a process is selected from the input queue and loaded into it. The free blocks of memory are known as *holes*. The set of holes is searched to determine which hole is best to allocate.

Memory Protection: Memory protection is a phenomenon by which we control memory access rights on a computer. The main aim of it is to prevent a process from accessing memory that has not been allocated to it. Hence prevents a bug within a process from affecting other processes, or the operating system itself, and instead results in a segmentation fault or storage violation exception being sent to the disturbing process, generally killing of process.

Memory Allocation:

Memory allocation is a process by which computer programs are assigned memory or space. It is of three types :

1. First Fit

The first hole that is big enough is allocated to program.

2. Best Fit

The smallest hole that is big enough is allocated to program.

3. Worst Fit

The largest hole that is big enough is allocated to program.

Fragmentation

Fragmentation occurs in a dynamic memory allocation system when most of the free blocks are too small to satisfy any request. It is generally termed as inability to use the available memory.

In such situation processes are loaded and removed from the memory. As a result of this, free holes exists to satisfy a request but is non contiguous i.e. the memory is fragmented into large no. Of small holes. This phenomenon is known as **External Fragmentation.**

Also, at times the physical memory is broken into fixed size blocks and memory is allocated in unit of block sizes. The memory allocated to a space may be slightly larger than the requested memory. The difference between allocated and required memory is known as **Internal fragmentation** i.e. the memory that is internal to a partition but is of no use.

Paging

A solution to fragmentation problem is Paging. Paging is a memory management mechanism that allows the physical address space of a process to be non-contagious. Here physical memory is divided into blocks of equal size called **Pages**. The pages belonging to a certain process are loaded into available memory frames.

Page Table

A Page Table is the data structure used by a virtual memory system in a computer operating system to store the mapping between *virtual address* and *physical addresses*.

Virtual address is also known as Logical address and is generated by the CPU. While Physical address is the address that actually exists on memory.

Segmentation

Segmentation is another memory management scheme that supports the user-view of memory. Segmentation allows breaking of the virtual address space of a single process into segments that may be placed in non-contiguous areas of physical memory.

Segmentation with Paging

Both paging and segmentation have their advantages and disadvantages, it is better to combine these two schemes to improve on each. The combined scheme is known as 'Page the Elements'. Each segment in this scheme is divided into pages and each segment is maintained in a page table. So the logical address is divided into following 3 parts :

- Segment numbers(S)
- Page number (P)
- The displacement or offset number (D)

Q.9. (c) How many 128x8 RAM chips are required to provide a memory capacity of 2048 bytes? **(2)**

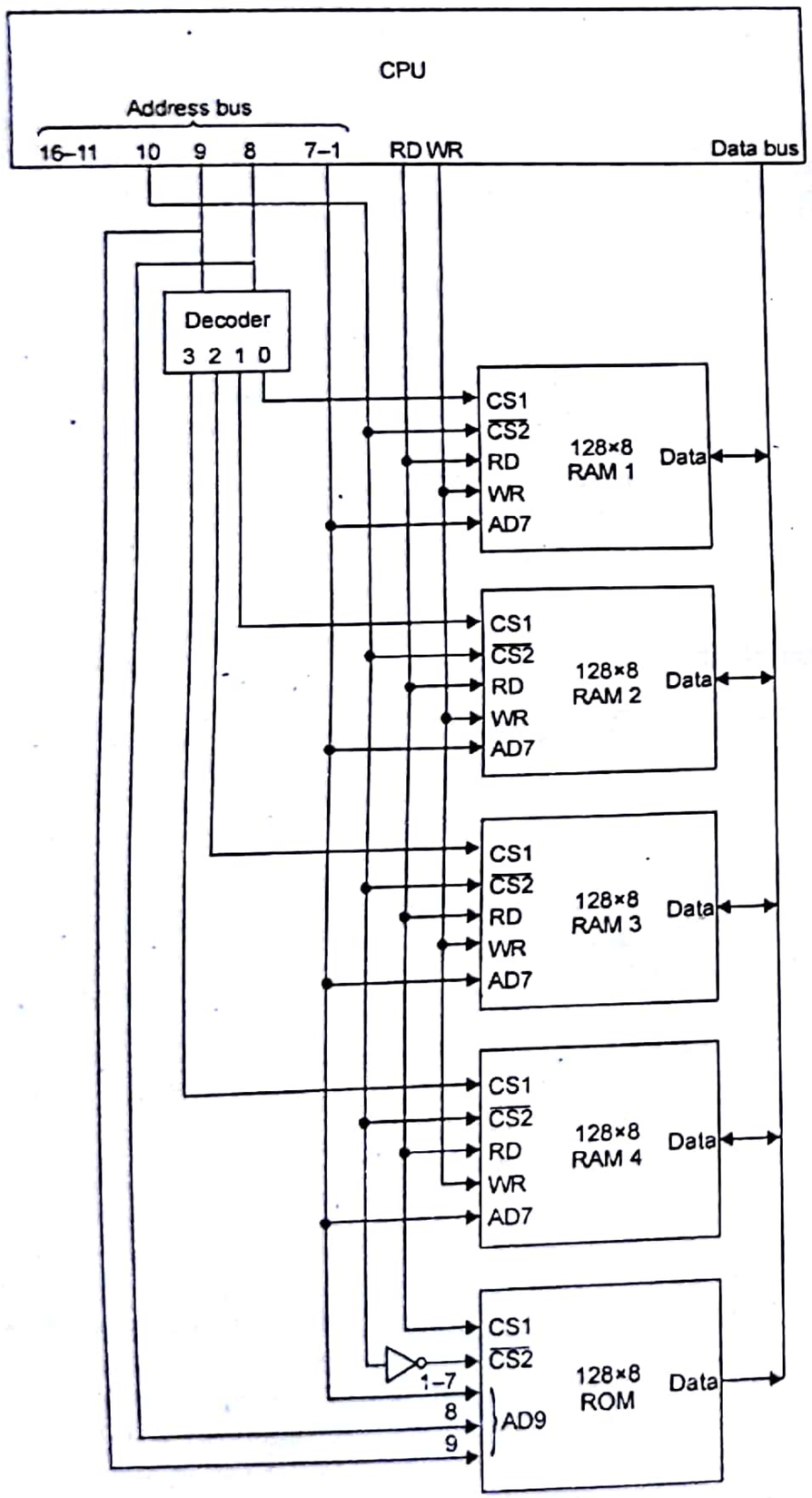
Ans. 128 x 8 RAM => 128 Bytes (8bits) / chip => Numbers of chips = 2048 / 128 = 16

Q.9. (d) How many lines of the address be must be used to access 2048 bytes of memory? How many of these lines ssssswill be common to all chips? **(2)**

Ans. $2048 = 2^d$, Where d is nbr of address lines => d = 11 We have 16 chips => 4 bits for chip select Number of common address lines is 11 - 4 = 7 Summary: 4 bits (MSB) to select the correct chip and 7 bits (LSB) to select the memory location inside the selected chip.

Q.9. (e) How many lines must be decoded for chips select? Specify the size of the decoders. **(2)**

Ans. We need 4 x 16 line decoder



FIRST TERM EXAMINATION [FEB. 2016] FOURTH SEMESTER [B.TECH] COMPUTER ORGANIZTION AND ARCHITECTURE [ETCS–204]

Time: 1.30 hrs. **M.M. : 30**

Note: *Q. No. 1 is compulsory and attempt any two from the remaining questions.*

Q.1. (a) Explain BUN (Branch unconditionally) instruction with example. **(2)**

Ans. BUN- Branch Unconditionally: This instruction has the responsibility to transfer the entire program to the instruction which is specified by the effective address. We now know that program counter (PC) holds the address of the instruction to be read from the memory and program is a set of instructions to be carried out to accomplish the particular task. BUN instruction allows the programmer to specify an instruction out of the program and modify the program.

Example:

D_4T_4: PC <- AR, SC<-0

Q.1. (b) Represent 1460.125_{10} in single precision format. **(2)**

Ans. STEP 1-

convert decimal in binary fractional form:

10110110100.001

STEP 2-

normalize binary fractional number

1.0110110100001*2^10

STEP 3-

convert 8 bit excess 127 notation:

20+127=137 10001001

STEP 4-

convert mantissa in hidden bit format:

0110110100001

STEP 5-

write down in 32 bit format:

0 10001001 01101101000010000000000

Q.1. (c) Define register transfer language (RTL). **(2)**

Ans. Register transfer language (RTL) is a kind of intermediate representation (IR) that is very close to assembly language, such as that which is used in a compiler. It is used to describe data flow at the register-transfer level of an architecture. Micro-operations can be expressed in terms of a Register Transfer Language (RTL).

Q.1. (d) Explain instruction cycle. **(2)**

Ans. An instruction cycle (sometimes called a fetch–decode–execute cycle) is the basic operational process of a computer. It is the process by which a computer retrieves a program instruction from its memory, determines what actions the instruction dictates, and carries out those actions. This cycle is repeated continuously by a computer's central processing unit (CPU), from boot-up to when the computer is shut down.

Q.1. (c) Explain the different shift operations. **(2)**

Ans. Shift micro operations are the operations in which the contents of the register can be shifted to left or right. Shift micro operations are used for serial transfer of data. They can also be used in lieu with arithmetic, logic and other data-processing operations. There are three types of shift operations:

1. Logical shift: Logical shift can be defined as the shift of the bits to the right or left serially. Let us suppose the symbol for logical right shift as shr and for logical left shift as shl.

Ex: Shift micro operationR1 shl R1

2. Circular shift: Circular shift also named as rotate shift circulates the bits among the ends of the register without losing information. We can achieve this by connecting the output terminal to the input terminal of the register. They also shift only one bit at a single time. For example

Ex: Circular micro operationR cir R

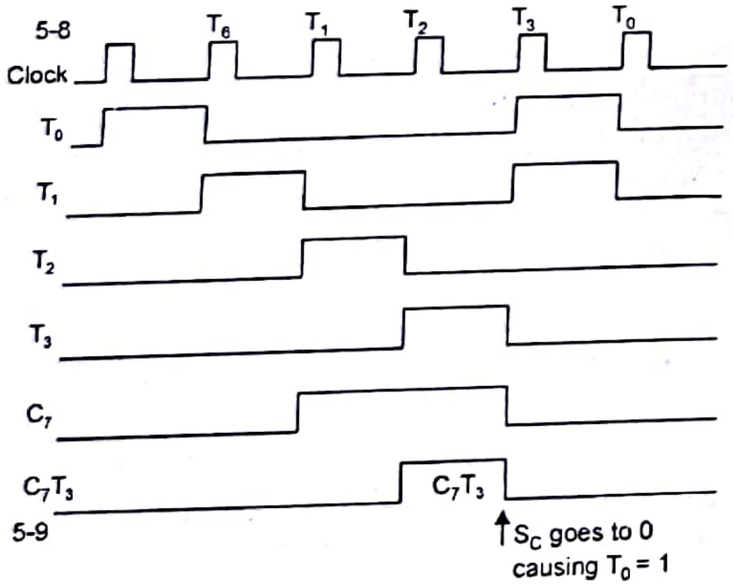
3. Arithmetic Micro-operation: The main purpose of Arithmetic Micro-operation is to perform arithmetic operation on numeric data.

Q. 2. (a) Draw & explain a timing diagram assuming that SC is cleared to 0 at time T_3 if control signal C_7 is active

C_7T_3 : SC4–0

C_7 is activated with the positive clock transition associated with T **(5)**

Ans.



Q.2. (b) List any 4 arithmetic micro-operations & describe what functions are performed by them. **(5)**

Ans. Arithmetic Microoperations

Types of Microoperations

1. Register Transfer

2. Arithmetic (Addition, subtraction, ...) Data is numeric, and bits with a word are interdependent.

3. Logic (AND, OR, ...) Data is not numeric, and bits are independent of each other. The same logical operation is applied to each bit in a word in parallel.

4. Shift. Data may or may not be numeric. All bits are moved the same number of positions left or right.

Arithmetic Microoperations:

Example	Description
R3 ← R1 + R2	Addition
R3 ← R1 – R2 (R1 + R2' + 1)	Subtraction
R2 ← R2'	Complement (really a logic operation)
R2 ← -R2 (R2' + 1)	Negation
R1 ← R1 + 1	Increment
R1 ← R1 – 1	Decrement

Q.3. (a) Explain the three different types of instruction code formats of a basic computer namely-memory reference, register-reference and input/output instruction. **(5)**

Ans. All Basic Computer instruction codes are 16 bits wide. There are 3 instruction code formats:

Memory-reference instructions take a single memory address as an operand, and have the format:

```
15 14  12 11      0
+-----------------------+
| I | OP  | Address  |
+-----------------------+
```

Here 12-bits are used for storing the address of the operands. Three bits are used for operation code and one bit is used for addressing mode. The opcode has the binary value from 000 to 110. The addressing mode I has two values 0 and 1. example: BUN- Branch Unconditionally

Register-reference instructions operate solely on the AC register, and have the following format :

```
  15  14 12 11    0
+-------------------+
| 0 | 111 | OP    |
+-------------------+
```

These Instructions are Recognised by opcode 111 with addressing mode 0.

example: INC- Increment AC

Input/output instructions have the following format:

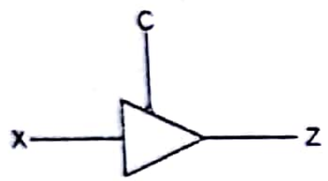
```
15  14 12 11    0
+-------------------+
| 0 | 111 | OP    |
+-------------------+
```

In the Input/Output Instruction, The MSB is 1 . The op-code bits is 111 and the remaining bits are used to store the other address. Example: INP- Input character to AC

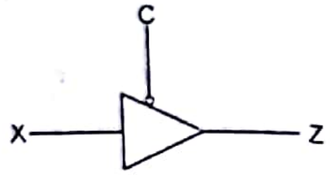
Q.3. (b) Explain three-state bus buffer with diagram. **(5)**

Ans. A tri-state buffer is a useful device that allows us to control when current passes through the device, and when it doesn't.

Here's two diagrams of the tri-state buffer.



Tri-state buffer with active high control



Tri-state buffer with active low control

A tri-state buffer has two inputs: a data input x and a control input c. The control input acts like a valve. When the control input is active, the output is the input. That is, it behaves just like a normal buffer. The "valve" is open.

When the control input is not active, the output is "Z". The "valve" is open, and no electrical current flows through. Thus, even if x is 0 or 1, that value does not flow through.

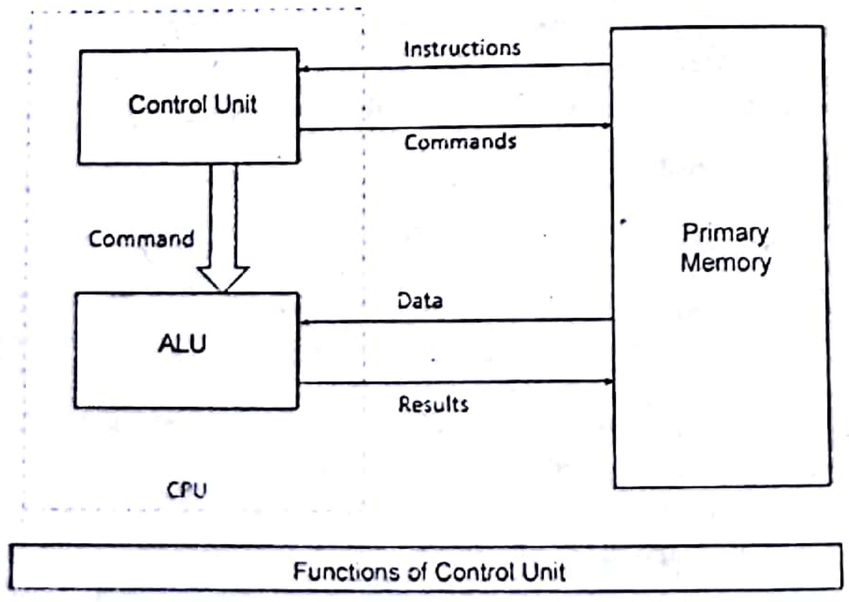
Here's a truth table describing the behavior of a active-high tri-state buffer.

c	x	z
0	0	Z
0	1	Z
1	0	0
1	1	1

In this case, when the output is Z, that means it's high impedance, neither 0, nor 1, *i.e.*, no current.

Q.4. (*a*) Draw & explain control unit of basic computer. (5)

Ans. The control unit (CU) is a component of a computer's central processing unit (CPU) that directs the operation of the processor. It tells the computer's memory, arithmetic/logic unit and input and output devices how to respond to a program's instructions. It directs the operation of the other units by providing timing and control signals. Most computer resources are managed by the CU. It directs the flow of data between the CPU and the other devices. John von Neumann included the control unit as part of the von Neumann architecture.Thus control unit controls the complete work-flow of the CPU. But control unit doesn't take inputs, give outputs, process data or store



Functions of Control Unit

data itself, what control unit do is, it controls these operations when they are performed by respective devices. The purpose of control unit to run the whole computer. And control unit is ran by the instructions stored in RAM and ROM. So, control unit receives instructions which are stored in RAM and ROM and controls operations of other connected units or devices through those instructions.

Q.4. (*b*) What is difference between a direct and an indirect address. Explain with example. (5)

Ans. Direct addressing means the instruction refers directly to the address being accessed. That is, the instruction encoding itself contains the address of the location. Depending on the instruction set, it may also allow computing a small index relative to the address.direct addressing has the address of a memory location as the operand in an instruction.

Ex:

8086:

MOV AX, [01234h] ; direct. Loads loc DS:01234h into AX

MOV AX, [01234h+BX]; direct-indexed. Loads loc DS:(01234h+BX) into AX

Indirect addressing uses an address held in a register or other location to determine what memory location to read or write. The idea here is that the instruction itself isn't directly telling you the address to access, but rather indirectly telling the CPU where to find that address. The processor may also allow you to add a small offset to the indirect address, giving an indirect-indexed addressing mode.indirect addressing has a pointer to the memory location as the operand in an instruction.

Ex:

8086:

MOV AX, [BX] ; Indirect via BX. Loads DS:BX into AX

MOV AX, [01234h + BX] ; Indirect-indexed. Loads DS:(01234h + BX) into AX

SECOND TERM EXAMINATION [APRIL–2016]
FOURTH SEMESTER [B.TECH]
COMPUTER ORGANIZTION AND ARCHITECTURE [ETCS–204]

Time: 1½ hrs. **M.M. : 30**

Note: *Q. No. 1 is compulsory and attempt any two from the remaining questions*

Q.1. (a) What is control word? **(2)**

Ans. There are 14 binary selection inputs in the units, and their combined value specified a control word. It consists of four fields three fields contain three bits each, and one field has five bits. The three bits of SEL A select a source register for the A input of the ALU. The three bits of SEL B select a source register for the B input of the ALU. The three bit of SEC D select a destination register using the decoder and its seven load outputs. The five bits of OPR select one of the operations in the ALU. The 14-bit control word when applied to the selection inputs specify a particular micro-operation.

Q.1. (b) What do you mean by addressing modes? **(2)**

Ans. Addressing modes are an aspect of the instruction set architecture in most central processing unit (CPU) designs. The various addressing modes that are defined in a given instruction set architecture define how machine language instructions in that architecture identify the operand(s) of each instruction. An addressing mode specifies how to calculate the effective memory address of an operand by using information held in registers and/or constants contained within a machine instruction or elsewhere.

Q.1 (c) What is strobe control? **(2)**

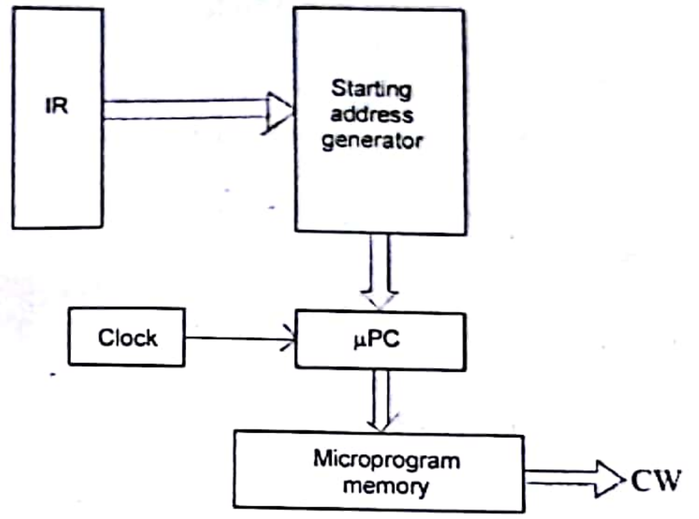
Ans. In computer or memory technology, a strobe is a signal that is sent that validates data or other signals on adjacent parallel lines. In memory technology, the CAS (column address strobe) and RAS (row address strobe) signals are used to tell a dynamic RAM that an address is a column or row address.

Q.1. (d) What is cache memory? **(2)**

Ans. Cache memory, also called CPU memory, is random access memory (RAM) that a computer microprocessor can access more quickly than it can access regular RAM. This memory is typically integrated directly with the CPU chip or placed on a separate chip that has a separate bus interconnect with the CPU. The basic purpose of cache memory is to store program instructions that are frequently re-referenced by software during operation. Fast access to these instructions increases the overall speed of the software program.

Q.1. (e) Draw block diagram of general configuration of a microprogrammed control unit. **(2)**

Ans.



Basic organization of a microprogrammed control

Q.2. (a) Evaluate the following arithmetic expressions using 2-address, 1-address and 0-address instructions to show the effect on computer program.

$$X = (A+B) * < (C - D),$$ **(6)**

Ans. 2 address :

MOV R1,A
ADD R1,B
MOV R2,C
SUB R2,D
MUL R1,R2
MOV X,R1

1 address:

LOAD A
ADD B
STORE T
LOAD C
SUB D
MUL T
STORE X

0 address:

PUSH A
PUSH B
ADD
PUSH C

PUSH D
SUB
MUL
POP X

Q.2. (b) The control memory has 4096 words of 24 bits each.

(i) How many bits are there in each of the four inputs going into the multiplexers?

(ii) What are the number of inputs in each multiplexer and how many multiplexers are needed?

(iii) How many bits are there in the control address register? **(4)**

Ans. Control memory = 2^12 * 24

(i) 12 bits

(ii) 12 multiplexers, each of size 4-to-1 line

(iii) 12 bits

Q.3. (a) What is basic advantage of using interrupt-initiated data transfer over transfer under program control without interrupt **(4)**

Ans. In the interrupt initiated data transfer, the processor verifies the request and transfer the control ISR to perform the task and its resumes back with the useful task while, the processor has to waste its time by performing all the task, for example when a print command is given in the interrupt initiated , it gives control over to ISR and resumes the work back where as without interrupt the processor has to wait unless the print document is transferred to the printer.

It does not require the continuous checking of I/O port by the computer . The computer checks the port only when an I/O interrupt is encountered . It saves processing time. The CPU issues commands to the I/O module then proceeds with its normal work until interrupted by I/O device on completion of its work.

For input, the device interrupts the CPU when new data has arrived and is ready to be retrieved by the system processor. The actual actions to perform depend on whether the device uses I/O ports, memory mapping.

For output, the device delivers an interrupt either when it is ready to accept new data or to acknowledge a successful data transfer. Memory-mapped and DMA-capable devices usually generate interrupts to tell the system they are done with the buffer.

Although Interrupt relieves the CPU of having to wait for the devices, but it is still inefficient in data transfer of large amount because the CPU has to transfer the data word by word between I/O module and memory. Below are the basic operations of Interrupt:

- CPU issues read command
- I/O module gets data from peripheral whilst CPU does other work
- I/O module interrupts CPU
- CPU requests data
- I/O module transfers data

Q.3. (b) A computer uses a memory unit with 256K words of 32 bits each. A binary instruction code is stored in one word of memory. The instruction has four parts: an indirect bit, an operation code, a register code part to specify one of 64 registers, and an address part. **(4)**

(i) How many bits are there in the operation code, the register code part and the address part?

(ii) Draw the instruction word format and indicate the number of bits in each part.

(iii) How many bits are there in the data and address inputs of the memory. **(6)**

Ans.

$$256K = 2^3 \times 2^{10} = 2^{18}$$
$$64 = 2^6$$

(a) Address : 18 bits $64 = 2^6$
Register code : 6 bits

Indurect bit : $\frac{1}{25}$ bits

$32 - 25 = 7$ sits for opcode,

(b)

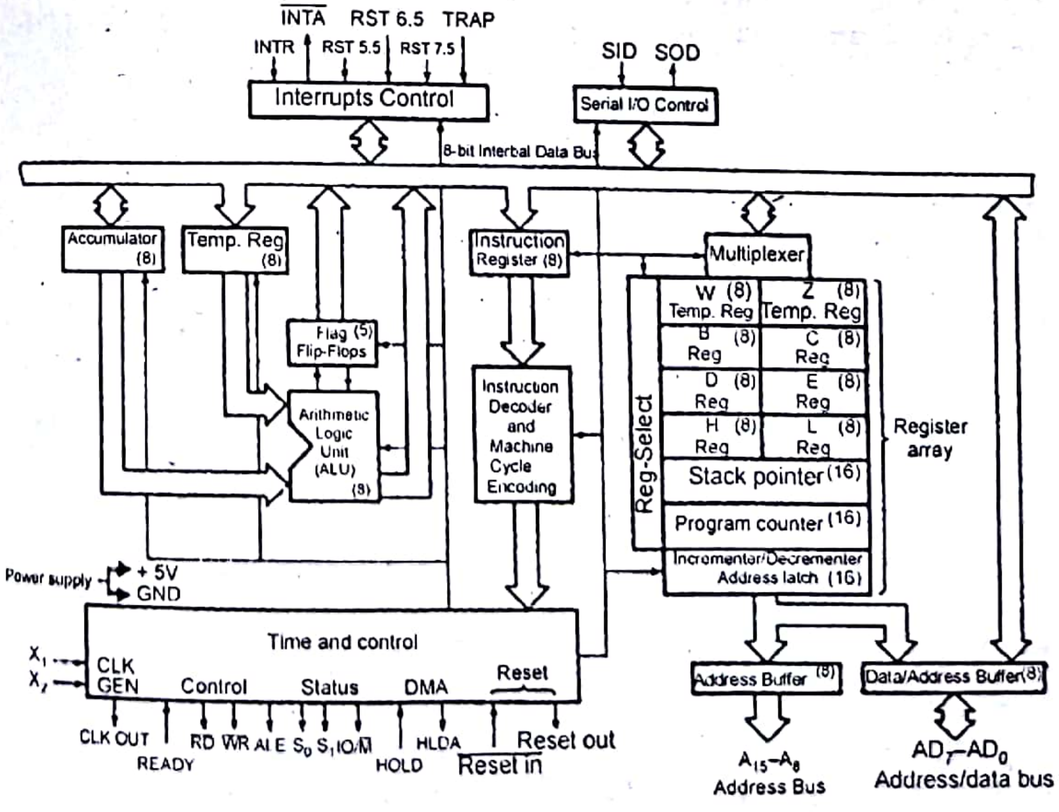
1	7	6	18
I	opcode	Register	Aodiess

= 32 bits

(c) Data : 32 bits ; address: 18 bits.

Q.4. (a) Draw the internal architecture of 8085 microprocessor. **(4)**

Ans.



Q.4. (b) Compare Direct mapping, Associative mapping and Set associative mapping. **(6)**

Ans. In a **direct-mapped** cache, each address in main memory has one and only one place in the cache in which it can be stored. If two addresses in main memory map to the same place in the cache, only one of those two can be resident in cache at the same time. Direct-Mapped Cache is simplier (requires just one comparator and one multiplexer), as a result is cheaper and works faster. Given any address, it is easy to identify the single entry in cache, where it can be. A major drawback when using DM cache is called a conflict miss, when two different addresses correspond to one entry in the cache. Even if the cache is big and contains many stale entries, it can't simply evict those, because the position within cache is predetermined by the address.

In a **set-associative** cache, each address in main memory has some number of places in the cache in which it can reside. Thus there are no two addresses that cannot reside in cache at the same time. A set-associative scheme is a hybrid between a fully associative cache, and direct mapped cache. It's considered a reasonable compromise between the complex hardware needed for fully associative caches (which requires parallel searches of all slots), and the simplistic direct-mapped scheme, which may cause collisions of addresses to the same slot (similar to collisions in a hash table).

In a **fully-associative cache,** any address in main memory can map to any address in the cache. This makes the cache to most flexible, but it's generally not practical. The problem is that the operation of checking if a particular address is resident in the cache is very complex in a fully-associative.Associative Cache is much more complex, and it allows to store an address into any entry. There is a price for that. In order to check if a particular address is in the cache, it has to compare all current entries (the tags to be exact). Besides in order to maintain temporal locality, it must have an eviction policy. Usually approximation of LRU (least recently used) is implemented, but it is also adds additional comparators and transistors into the scheme and of course consumes some time. cache (because it could be anywhere), and that operation needs to be very fast.

END TERM EXAMINATION [MAY. 2016]
FOURTH SEMESTER [B.TECH]
COMPUTER ORGANIZTION AND ARCHITECTURE [ETCS-204]

Time: 3 hrs. **M.M. : 75**

Note: *Attempt any five questions including Q. No. 1 which is compulsory.*

Q.1. Attempt All.

Q.1. (a) Differentiate between Micro-operation and Macro operation. **(5)**

Ans.

Micro Operation	Macro Operation
Micro-operations (also known as a micro-ops or μops) are detailed low-level instructions used in some designs to implement complex machine instructions	A Macro instruction is a line of computer program coding that results in one or more lines of program coding in the target programming language, sets variables for use by other statements
Micro-operations perform basic operations on data stored in one or more registers, including transferring data between registers or between registers and external buses of the central processing unit (CPU), and performing arithmetic or logical operations on registers.	In a typical fetch-decode-execute cycle, each step of a macro-instruction is decomposed during its execution so the CPU determines and steps through a series of micro-operations.
The execution of micro-operations is performed under control of the CPU's control unit, which decides on their execution	The use of macro instructions was initiated for two main purposes: to reduce the amount of program coding that had to be written by generating several assembly language statements from one macro instruction and to enforce program writing standards
The micro-operations in computers are classified into the following categories: Register transfer micro-operations, Arithmetic micro-operations, Logic micro-operations and Shift micro operations **Example:** R3 ← R1 + R2 R3 ← R1 + (1's complement of R2)	A macro-operation is usually divided into • operation code, operand address, addressing mode, etc. • basic addressing modes Immediate, Direct, Indirect Example: for the EXIT macro instruction, a list of define constant instructions and peripheral access by access methods including macros such as • PEN, CLOSE, READ and WRITE

Q.1. (b) Differentiate between access time and cycle time of a memory. (5)

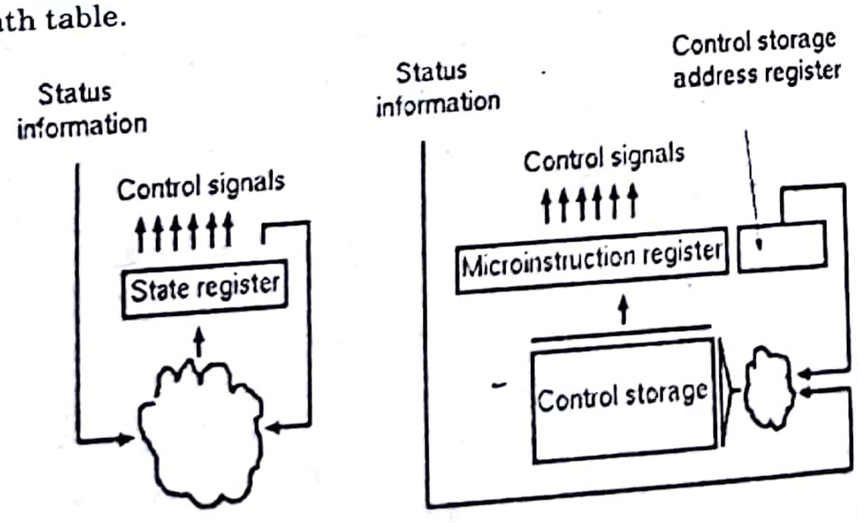
Ans. Memory access time is corresponding to the time interval between the read/write request and the availability of the data. The time required by a processor to access data or to write data from and to memory chip is referred as access time. Cache memory has the shortest access time. *Access time* is also frequently used to describe the speed of disk drives. Disk access times are measured in milliseconds (thousandths of a second), often abbreviated as *ms*. Fast hard disk drivesfor personal computers boast access times of about 9 to 15 milliseconds. The access time for disk drives includes the time it actually takes for the *read / write head* to locate a sector on the disk (called the *seek time*). This is an average time since it depends on how far away the head is from the desired data.

Cycle time represents the minimum time interval between two successive accesses. The memory cycle time is a measurement of how quickly two back-to-back accesses of a memory chip can be made. Note that a DRAM chip's cycle time is usually much longer than its access time, which measures only a single access. This is because there is a latency between successive memory accesses. Cycle time is the time, usually measured in nanosecond s, between the start of one random access memory access to the time when the next access can be started.

Q.1. (c) Differentiate between Hardwired and Micro Programmed control unit. (5)

Ans. Micro programmed control: Micro programmed control is a control mechanism to generate control signals by using a memory called control storage (CS), which contains the control signals. Although micro programmed control seems to be advantageous to CISC machines, since CISC requires systematic development of sophisticated control signals, there is no intrinsic difference between these 2 control mechanisms.

Hard-wired control: Hardwired control is a control mechanism to generate control signals by using appropriate finite state machine (FSM). The pair of "microinstruction-register" and "control storage address register" can be regarded as a "state register" for the hardwired control. Note that the control storage can be regarded as a kind of combinational logic circuit. We can assign any 0, 1 values to each output corresponding to each address, which can be regarded as the input for a combinational logic circuit. This is a truth table.



(a) Hardwired control

(b) Microprogrammed control

Q.1. (d) Differentiate between Asynchronous Data Transfers and synchronous data transfer. (5)

Ans. Synchronous: In synchronous data transfers, the sender and receiver take some time to communicate before they make the exchange. This communication outlines the parameters of the data exchange. This usually involves establishing which end, sender or receiver, will be in control of the transfer. Here, the two parties also ensure they are using the same timing; that is, they know when each burst ends and another begins. They also set parameters for resetting their clocks during the transfer to make sure they don't drift away from the agreed-upon timing.A synchronous data transfer is the generation and transferring of data on a Clock with known phase and frequency. The data is synchronized according to the Setup and Hold requirements of the known clock.Synchronous messaging involves a client that waits for the server to respond to a message. Messages are able to flow in both directions, to and from.

Asynchronous: In asynchronous, or "best effort" transfers, sender and receiver do not establish the parameters of the information exchange. Rather, the sender places extra bits of data before and after each burst that indicate when each burst begins and ends. It then sends the information, and it is up to the receiver to determine how to reset its clock to match the timing of the signal. Unlike synchronous transfers, the receiver does not take time to communicate to the sender information about what it received. Whereas, asynchronous data transfer could be the one where a data is generated on, say, CLOCK 1 and it is sampled on any other clock, say, CLOCK 2 (Where clock 1 and clock 2 are asynchronous to each other and has no known phase and frequency relationship). Then the data will arrive asynchronously to CLOCK 2 and it will be an asynchronous data transfer. Asynchronous messaging involves a client that does not wait for a message from the server.

Q.1. (e) Differentiate between Unsigned notation and, signed notation. Find range of 2 byte integer in both cases. (5)

Ans. Signed variables, such as signed integers will allow you to represent numbers both in the positive and negative ranges.Unsigned variables, such as unsigned integers, will only allow you to represent numbers in the positive.Unsigned and signed variables of the same type (such as int and byte) both have the same range (range of 65,536 and 256 numbers, respectively), but unsigned can represent a larger magnitude number than the corresponding signed variable. Signed variables use one bit to flag whether they are positive or negative. Unsigned variables don't have this bit, so they can store larger numbers in the same space, but only nonnegative numbers, e.g. 0 and higher.Unsigned variables are variables which are internally represented without a mathematical sign (plus or minus) can store 'zero' or positive values only. Let us say the unsigned variable is n bits in size, then it can represent 2^n (2 power n) values - 0 through (2^n –1). A signed variable on the other hand, 'loses' one bit for representing the sign, so it can store values from -(2^(n–1) –1) through (2^(n–1)) including zero. Thus, a signed variable can store positive values, negative values and zero.

Here n= 2*8=16 bits

So for unsigned notation range is from – 0 to 65535

while for signed notation range is from – –32768 to +32767

Q.2. (a) What is RS 232-C standard? Explain the signals associated with it. (6)

Ans. In telecommunications, RS-232 is a standard for serial communication transmission of data. It formally defines the signals connecting between a DTE (data

terminal equipment) such as a computer terminal, and a DCE (data circuit-terminating equipment or data communication equipment), such as a modem. The RS-232 standard is commonly used in computer serial ports. The standard defines the electrical characteristics and timing of signals, the meaning of signals, and the physical size and pinout of connectors. An RS-232 serial port was once a standard feature of a personal computer, used for connections to modems, printers, mice, data storage, uninterruptible power supplies, and other peripheral devices. However, RS-232 is hampered by low transmission speed, large voltage swing, and large standard connectors. In modern personal computers, USB has displaced RS-232 from most of its peripheral interface roles. Many computers do not come equipped with RS-232 ports and must use either an external USB-to-RS-232 converter or an internal expansion card with one or more serial ports to connect to RS-232 peripherals. Nevertheless, RS-232 devices are still used, especially in industrial machines, networking equipment, and scientific instruments.

The various signals associated with RS-232 are:

Transmitted Data	Carries data from DTE to DCE.	TxD
Request To Send	DTE requests the DCE prepare to transmit data.	RTS
Received Data	Carries data from DCE to DTE.	RxD
Ready To Receive	DTE is ready to receive data from DCE. If in use, RTS is assumed to be always asserted.	RTR
Clear To Send	DCE is ready to accept data from the DTE.	CTS

Q.2. (*b*) What are the advantages of byte addressing mechanism over word addressing mechanism and what are their disadvantages? (6.5)

Ans. Byte addressing refers to hardware architectures which support accessing individual bytes of data rather than only larger units called words, which would be word-addressable.

The basic unit of digital storage is called a bit. In most common computer architectures, 8 bits are grouped together to form a byte. Byte addressable memory refers to architectures where data can be accessed 8 bits at a time. In computer architecture, a word is an ordered set of bytes or bits that is the normal unit in which information may be stored, transmitted, or operated on within a given computer. If a computer's memory is word-addressable then each word in memory is assigned its own memory address. That means that the processor is able to address and fetch only complete words from the memory.

As a simple example, consider a computer with a 16–bit address space. The machine would have 65,536 (64K = 216) addressable entities. The maximum memory size would depend on the size of the addressable entity.

Byte Addressable	64 KB
16-bit Word Addressable	128 KB
32-bit Word Addressable	256 KB

For a given address space, the maximum memory size is greater for the larger addressable entities. This may be an advantage for certain applications, although this advantage is reduced by the very large address spaces we now have: 32–bits now and

64–bits soon. The advantages of byte-addressability are clear when we consider applications that process data one byte at a time. Access of a single byte in a byte-addressable system requires only the issuing of a single address. In a 16–bit word addressable system, it is necessary first to compute the address of the word containing the byte, fetch that word, and then extract the byte from the two-byte word. Although the processes for byte extraction are well understood, they are less efficient than directly accessing the byte. For this reason, many modern machines are byte addressable.

Q.3. Discuss different addressing modes used in computer systems using examples. (12.5)

Ans. Addressing modes are the ways how architectures specify the address of an object they want to access. In GPR machines, an addressing mode can specify a constant, a register or a location in memory.

IMMEDIATE MODE: The operand is an immediate value is stored explicitly in the instruction.

INDEXED MODE: The address of the operand is obtained by adding to the contents of the general register (called index register) a constant value. The number of the index register and the constant value are included in the instruction code. Index Mode is used to access an array whose elements are in successive memory locations. The content of the instruction code, represents the starting address of the array and the value of the index register, and the index value of the current element. By incrementing or decrementing index register different element of the array can be accessed.

INDIRECT MODE: The effective address of the operand is the contents of a register or main memory location, location whose address appears in the instruction. Indirection is noted by placing the name of the register or the memory address given in the instruction in parentheses. The register or memory location that contains the address of the operand is a pointer. When an execution takes place in such mode, instruction may be told to go to a specific address. Once it's there, instead of finding an operand, it finds an address where the operand is located.

DIRECT MODE: The address of the operand is embedded in the instruction code.

DISPLACEMENT MODE: Similar to index mode, except instead of a index register a base register will be used. Base register contains a pointer to a memory location. An integer (constant) is also referred to as a displacement. The address of the operand is obtained by adding the contents of the base register plus the constant. The difference between index mode and displacement mode is in the number of bits used to represent the constant.

Addressing modes	Example Instruction	Meaning
Register	Add R4,R3	R4 <- R4 + R3
Immediate	Add R4, #3	R4 <- R4 + 3
Displacement	Add R4, 100(R1)	R4 <- R4 + M[100+R1]
Indexed	Add R3, (R1 + R2)	R3 <- R3 + M[R1+R2]
Direct	Add R1, (1001)	R1 <- R1 + M[1001]

Q.4. (a) Explain zero-address, one-address and two-address instructions with examples. **(6)**

Ans. Zero-Address Instructions: A stack-organized computer does not use an address field for the instructions ADD and MUL. The PUSH and POP instructions, however, need an address field to specify the operand that communicates with the stack.

```
PUSH A      //Top of Stack <- A
PUSH B      //Top of Stack <- B
ADD         //Top of Stack <- (A + B)
PUSH C      //Top of Stack <- C
PUSH D      //Top of Stack <- D
ADD         //Top of Stack <- (C + D)
MUL         //Top of Stack <- (C + D) * (A + B)
POP X       //Top of Stack <- M[X] -< Top of Stack
```

One-Address Instructions:

One-address instructions use an Accumulator (AC) register for all data manipulation.

```
LDA A       //AC  <- M[A]
ADD B       //AC  <- AC + M[B]
STA T       //M[T] <- AC
LDA C       //AC  <- M[C]
ADD D       //AC  <- AC + M[D]
MUL T       //AC  <- AC * M[T]
STA X       //M[X] <- AC
```

Two-Address Instructions:

Two-address instructions are the most common in commercial computers. Each address field can specify either a processor register or a memory word.

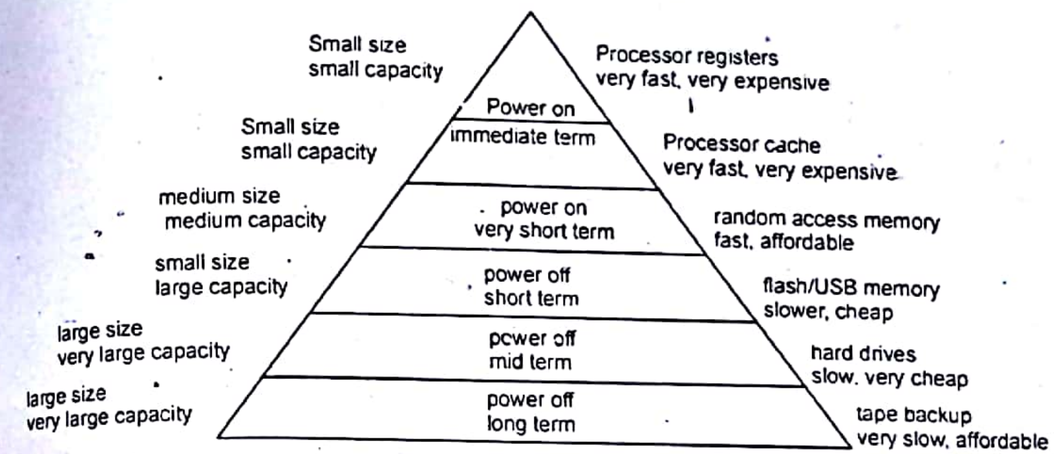
```
MOV R1, A   //R1  <- M[A]
ADD R1, B   //R1  <- R1 + M[B]
MOV R2, C   //R2  <- M[C]
ADD R2, D   //R2  <- R2 + M[D]
MUL R1, R2  //R1  <- R1 * R2
MOV X, R1   //M[X] <- R1
```

Q.4. (b) Explain the need of memory hierarchy with the help of a block diagram? What is the reason for not having one large memory unit for storing all information at one place? **(6-5)**

Ans. In computer architecture the memory hierarchy is a concept used to discuss performance issues in computer architectural design, algorithm predictions, and lower level programming constructs involving locality of reference. The memory hierarchy in computer storage separates each of its levels based on response time. Since response time, complexity, and capacity are related, the levels may also be

distinguished by their performance and controlling technologies. Designing for high performance requires considering the restrictions of the memory hierarchy, i.e. the size and capabilities of each component. Each of the various components can be viewed as part of a hierarchy of memories (m1, m2,...,mn) in which each member mi is typically smaller and faster than the next highest member mi+1 of the hierarchy. To limit waiting by higher levels, a lower level will respond by filling a buffer and then signaling to activate the transfer.

Computer Memory Hierarchy



A large single unit is not used whereas the memory hierarchy is used since it increases performance by saving time based on locality of reference. A "memory hierarchy" in computer storage distinguishes each level in the "hierarchy" by response time. Each level of the hierarchy is of higher speed and lower latency, and is of smaller size, than lower levels.

Q.5. The 8-bit registers A, B, C and D are loaded with the value $(F2)_H$, $(B9)_H$ and $(EA)_H$ respectively. Determine the register content after the execution of the following sequence of micro-operations sequentially. Where Shl=shift left.shr=shift right and cir=circular. **(12.5)**

(i) A ← A + B, C ← C + shl(d

(ii) C ← C + D, B ← B + 1

(iii) A ← A-C .

(iv) A ← shr(B) ⊕ cir(d)

Ans. A = 11110010

B = 10111001

C = 11101010

D = 10010101 (assumed value)

(i) A← A+B = 1-CARRY 10101011

C ← C+SHL(D) = 11101010 + 00101010 = 1-CARRY 00010100

(ii) C ← C+D = 1-CARRY 01101001 , B ← B+1= 10111010

(iii) A ← A-C = 00001000

(iv) A ← SHR(B) exor cir(D)

= 01011100 EXOR 00101011= 01110111

Q.6. (a) Explain 8085 instruction set architecture and its organization in detail. **(6.5)**

Ans. Instruction sets are instruction codes to perform some task. It is classified into five categories.

CONTROL INSTRUCTIONS:

NOP - No operation is performed, i.e., the instruction is fetched and decoded.

HLT - Halt and enter wait state. The CPU finishes executing the current instruction and stops further execution. An interrupt or reset is necessary to exit from the halt state.

LOGICAL INSTRUCTIONS:

CMP - Compare the register or memory with the accumulator

CPI - Compare immediate with the accumulator

ARITHMETIC INSTRUCTIONS:

ADD - Add register or memory, to the accumulator

SUB - Subtract the register or the memory from the accumulator

DATA TRANSFER INSTRUCTIONS:

MOV - Copy from the source (Sc) to the destination(Dt)

LDA - Load the accumulator

8085 consists of the following functional units

Accumulator: It is an 8-bit register used to perform arithmetic, logical, I/O & LOAD/ STORE operations. It is connected to internal data bus & ALU.

Arithmetic and logic unit: As the name suggests, it performs arithmetic and logical operations like Addition, Subtraction, AND, OR, etc. on 8-bit data.

General purpose register: There are 6 general purpose registers in 8085 processor, i.e. B, C, D, E, H & L. Each register can hold 8-bit data.

Program counter: It is a 16-bit register used to store the memory address location of the next instruction to be executed.

Stack pointer: It is also a 16-bit register works like stack, which is always incremented/decremented by 2 during push & pop operations.

Temporary registers: It is an 8-bit register, which holds the temporary data of arithmetic and logical operations.

Flag register: It is an 8-bit register having five 1-bit flip-flops, which holds either 0 or 1 depending upon the result stored in the accumulator.

It has the following configuration:

-8-bit data bus

-16-bit address bus, which can address upto 64KB

-A 16-bit program counter

- 16-bit stack pointer

-Six 8-bit registers arranged in pairs: BC, DE, HL

Q.6. (b) Discuss about Input-Output and Interrupts in detail. **(6)**

Ans. INPUT-OUTPUT: In computing, input/output or I/O (or, informally, io or IO) is the communication between an information processing system, such as a computer,

and the outside world, possibly a human or another information processing system. Inputs are the signals or data received by the system and outputs are the signals or data sent from it. The term can also be used as part of an action; to "perform I/O" is to perform an input or output operation. I/O devices are used by a human (or other system) to communicate with a computer. For instance, a keyboard or computer mouse is an input device for a computer, while monitors and printers are output devices. Devices for communication between computers, such as modems and network cards, typically perform both input and output operations. The designation of a device as either input or output depends on perspective. Mouse and keyboards take physical movements that the human user outputs and convert them into input signals that a computer can understand; the output from these devices is the computer's input. Similarly, printers and monitors take signals that a computer outputs as input, and they convert these signals into a representation that human users can understand. From the human user's perspective, the process of reading or seeing these representations is receiving input; this type of interaction between computers and humans is studied in the field of human–computer interaction.

INTERRUPTS: In system programming, an interrupt is a signal to the processor emitted by hardware or software indicating an event that needs immediate attention. An interrupt alerts the processor to a high-priority condition requiring the interruption of the current code the processor is executing. Hardware interrupts are used by devices to communicate that they require attention from the operating system. Internally, hardware interrupts are implemented using electronic alerting signals that are sent to the processor from an external device, which is either a part of the computer itself, such as a disk controller, or an external peripheral. A software interrupt is caused either by an exceptional condition in the processor itself, or a special instruction in the instruction set which causes an interrupt when it is executed. The former is often called a trap or exception and is used for errors or events occurring during program execution that are exceptional enough that they cannot be handled within the program itself. **(6)**

Q.7. (a) Discuss the procedure to implements a simple CPU. **(6)**

Ans. Hardwired into a CPU's circuitry is a set of basic operations it can perform, called an instruction set. Such operations may involve, for example, adding or subtracting two numbers, comparing two numbers, or jumping to a different part of a program. Each basic operation is represented by a particular combination of bits, known as the machine language opcode; while executing instructions in a machine language program, the CPU decides which operation to perform by "decoding" the opcode. In general, a CPU executes an instruction by fetching it from memory, using its ALU to perform an operation, and then storing the result to memory. Beside the instructions for integer mathematics and logic operations, various other machine instructions exist, such as those for loading data from memory and storing it back, branching operations, and mathematical operations on floating-point numbers performed by the CPU's floating-point unit (FPU).

Control unit: The control unit of the CPU contains circuitry that uses electrical signals to direct the entire computer system to carry out stored program instructions. The control unit does not execute program instructions; rather, it directs other parts of the system to do so. The control unit communicates with both the ALU and memory.

Arithmetic logic unit: The arithmetic logic unit (ALU) is a digital circuit within the processor that performs integer arithmetic and bitwise logic operations. The inputs

to the ALU are the data words to be operated on (called operands), status information from previous operations, and a code from the control unit indicating which operation to perform.

Memory management unit: Most high-end microprocessors (in desktop, laptop, server computers) have a memory management unit, translating logical addresses into physical RAM addresses, providing memory protection and paging abilities, useful for virtual memory. Simpler processors, especially microcontrollers usually don't include an MMU.

Q.7. (b) Starting from an initial value of R=10011101. Determine the sequence of binary values in R after logical shift left, followed by a circular shift-right followed by a logical shift right and a circular shift left. **(6.5)**

Ans. R= 10011101

LOGICAL SHIFT LEFT- 00111010

CIRCULAR SHIFT RIGHT- 00011101

LOGICAL SHIFT RIGHT- 00001110

CIRCULAR SHIFT LEFT- 00011100

Q.8. Write short notes on any two of the following: **(6.25×2 = 12.5)**

Q.8.(*a*) Bus Architecture and Bus Arbitration

Ans. Bus Architecture: Bus is a group of wires that connects different components of the computer. It is used for transmitting data, control signal and memory address from one component to another. A bus can be 8 bit, 16 bit, 32 bit and 64 bit. A 32 bit bus can transmit 32 bit information at a time. A bus can be internal or external.

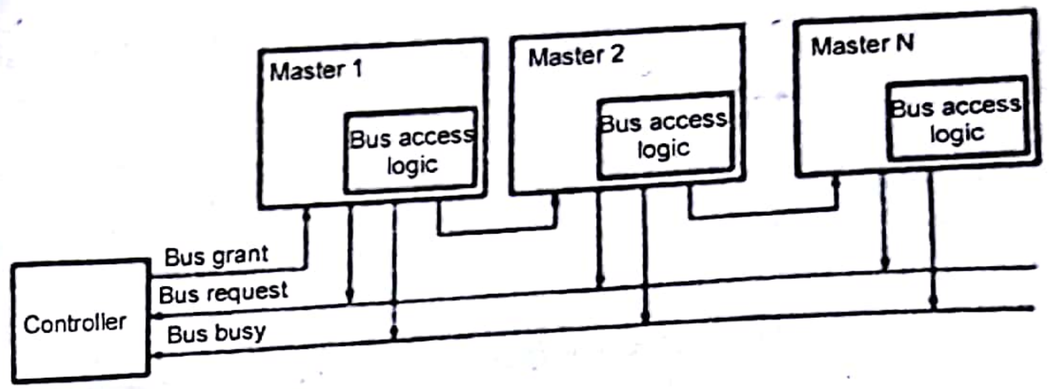
Types of bus:

- **Data bus:** Data bus carries data from on component to another. It is uni-directional for input and output devices and bi-directional for memory and CPU.
- **Control bus:** Control bus carries control signal. CU of CPU uses control signal for controlling all the components. It is uni-directional from CPU to all other components.
- **Address bus:** Address bus carries memory address. A memory address is a numerical value used for identifying a memory location. Computer performs all its task through the memory address. CU of CPU sends memory address to all the components. So, address bus is also uni-directional from CPU to all other components.

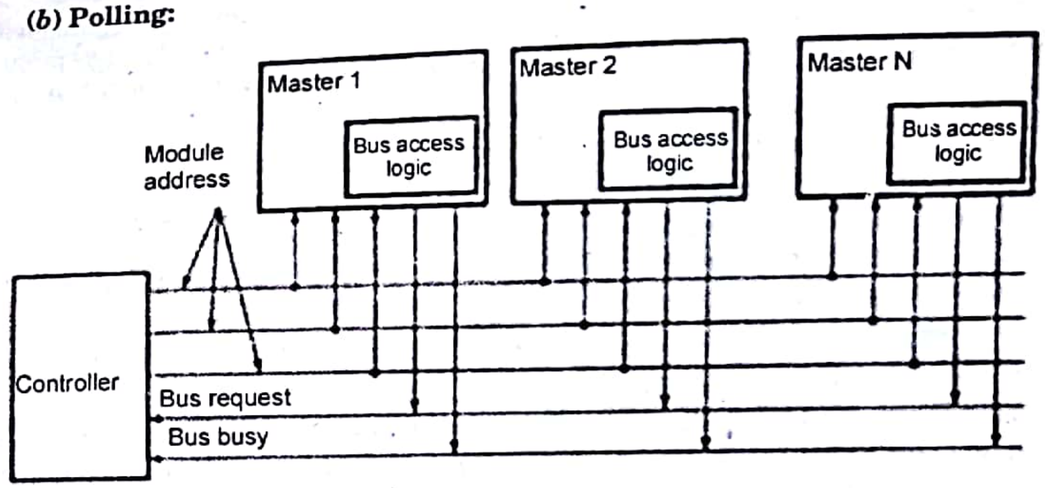
Bus Arbitration: Bus arbitration is the process by which the next device to become the bus master is selected and bus mastership is transferred to it. The selection of bus master is usually done on the priority basis.There are two approaches to bus arbitration: Centralized and distributed.

1. CENTRALIZED ARBITRATION: In centralized bus arbitration, a single bus arbiter performs the required arbitration. The bus arbiter may be the processor or a separate controller connected to the bus.

(*a*) Daisy chaining: It is simple and cheaper method. All masters make use of the same line for bus request. In response to the bus request the controller sends a bus grant if the bus is free.



(*b*) Polling:



In this the controller is used to generate the addresses for the master. Number of address line required depends on the number of master connected in the system.

2. DISTRIBUTED ARBITRATION: In distributed arbitration, all devices participate in the selection of the next bus master. In this scheme each device on the bus is assigned a 4-bit identification number. The number of devices connected on the bus when one or more devices request for the control of bus, they assert the start-arbitration signal and place their 4-bit ID numbers on arbitration lines, ARB0 through ARB3.

Q.8. (*b*) Levels of programming languages.

Ans. LEVELS OF PROGRAMMING LANGUAGES:

A programming language is used by a human programmer to direct a computer to accomplish a specific set of steps which lead to a desired outcome.These programming languages can be divided into three major groups. They are:

- Machine
- Assembly
- High-level

Machine language is the "native tongue" of the computer, the language closest to the hardware itself. Each unique computer has a unique machine language. A machine language program is made up of a series of binary patterns (e.g., 01011100) which represent simple operations that can be accomplished by the computer (e.g., add two operands, move data to a memory location). Machine language programs are executable,

meaning that they can be run directly. Programming in machine language requires memorization of the binary codes and can be difficult for the human programmer.

Assembly language represents an effort to make programming easier for the human. The machine language instructions are replaced with simple pneumonic abbreviations (e.g., ADD, MOV). Thus assembly languages are unique to a specific computer (machine). Prior to execution, an assembly language program requires translation to machine language. This translation is accomplished by a computer program known as an Assembler. Assemblers are written for each unique machine language.

High-level languages are more English-like and, therefore, make it easier for programmers to "think" in the programming language. High-level languages also require translation to machine language before execution. This translation is accomplished by either a compiler or an interpreter. Compilers translate the entire source code program before execution. Interpreters translate source code programs one line at a time.

Q.8. (c) RS-232-C and RS-422 standard.

Ans. RS-232-C STANDARD: RS-232C is a long-established standard ("C" is the current version) that describes the physical interface and protocol for relatively low-speed serial data communication Networks between computers and related devices. RS-232C is the interface that your computer uses to talk to and exchange data with your modem and other serial devices. RS-232C is the interface between your Communication networks and other communication networks.

RS-422 STANDARD: RS-422, also known as TIA/EIA-422, is a technical standard originated by the Electronic Industries Alliance that specifies electrical characteristics of a digital signaling circuit. Differential signaling can transmit data at rates as high as 10 Mbit/s, or may be sent on cables as long as 1500 meters. Some systems directly interconnect using RS-422 signals, or RS-422 converters may be used to extend the range of RS-232 connections. The standard only defines signal levels; other properties of a serial interface, such as electrical connectors and pin wiring, are set by other standards.

FIRST TERM EXAMINATION [FEB. 2017]
FOURTH SEMESTER [B.TECH.]
COMPUTER ORGANISATION AND ARCHITECTURE [ETCS–204]

Time : 1½ Hrs. **M.M. : 30**

Note: *Attempt Q. No. 1 which is compulsory and two more questions from recmaining.*

Q.1. (a) What do you mean by base of a number system? **(2)**

Ans. Let's look at base-two, or binary, numbers. How would you write, for instance, 12_{10} ("twelve, base ten") as a binary number? You would have to convert to base-two columns, the analog of base-ten columns. In base ten, you have columns or "places" for $10^0 = 1$, $10^1 = 10$, $10^2 = 100$, $10^3 = 1000$, and so forth. Similarly in base two, you have columns or "places" for $2^0 = 1$, $2^1 = 2$, $2^2 = 4$, $2^3 = 8$, $2^4 = 16$, and so forth.

Q.1. (b) What is an accumulator? **(2)**

Ans. An accumulator is a register for short-term, intermediate storage of arithmetic and logic data in a computer's CPU (central processing unit). The term "accumulator" is rarely used in reference to contemporary CPUs, having been replaced around the turn of the millennium by the term "register." In a modern computers, any register can function as an accumulator.

Q.1. (c) What are the advantages of RTL? **(2)**

Ans. In computer science, **register transfer language (RTL)** is a kind of intermediate representation (IR) at the register-transfer level of an architecture. Academic papers and textbooks often use a form of that is very close to assembly language, such as that which is used in a compiler. It is used to describe data flow RTL as an architecture-neutral assembly language.

Q.1. (d) Difference between direct and indirect instruction. **(2)**

Ans. Direct addressing means the instruction refers directly to the address being accessed. That is, the instruction encoding itself contains the address of the location. Depending on the instruction set, it may also allow computing a small index relative to the address. When used that way, you can think of that as a direct-indexed mode. Indirect addressing uses an address held in a register or other location to determine what memory location to read or write. The idea here is that the instruction itself isn't directly telling you the address to access, but rather indirectly telling the CPU where to find that address. The processor may also allow you to add a small offset to the indirect address, giving an indirect-indexed addressing mode.

Q.1. (e) Explain BUN (Branch Unconditionally) instruction with example. **(2)**

Ans. BUN stands for branch unconditionally and this instruction allows one to select an instruction from a program (which is a group of instructions) and gives him access to modify the program as he wants to. Example of how to write this instruction in programs is as follows:

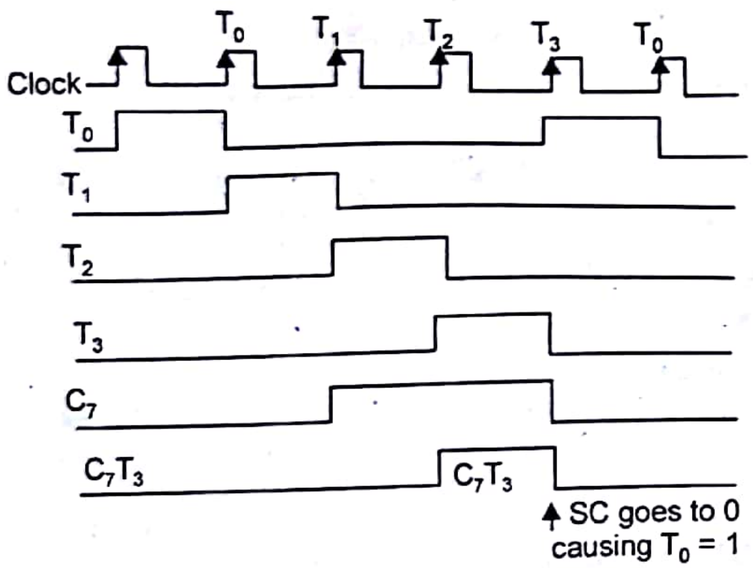
Example of D_4T_4: PC <– AR, SC <– 0

Q.2 (a) Draw and explain a timing diagram assuming that the SC is cleared to 0 at time T3 if the control signal C7 is active.

C7T3 : SC –> 0

C7 is activated with the positive clock transition associated with T2. (6)

Ans.

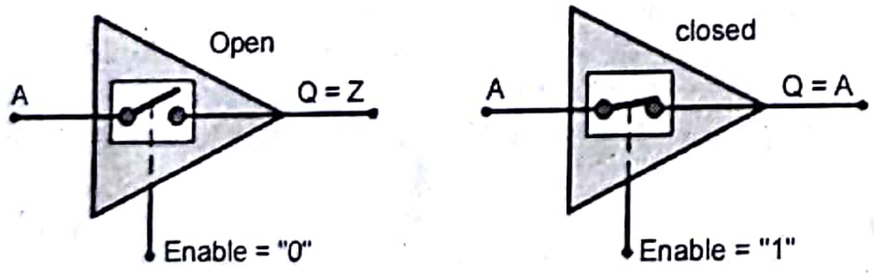


Q.2. (b) Explain three state buffer with diagram. (4)

Ans. A Tri-state Buffer can be thought of as an input controlled switch with an output that can be electronically turned "ON" or "OFF" by means of an external "Control" or "Enable" (EN) signal input. This control signal can be either a logic "0" or a logic "1" type signal resulting in the Tri-state Buffer being in one state allowing its output to operate normally producing the required output or in another state were its output is blocked or disconnected.

Then a tri-state buffer requires two inputs. One being the data input and the other being the enable or control input as shown.

Tri-state Buffer Switch Equivalent



When activated into its third state it disables or turns "OFF" its output producing an open circuit condition that is neither at a logic "High" or "Low", but instead gives an output state of very high impedance, **High-Z**, or more commonly Hi-Z. Then this type of device has two logic state inputs, "0" or a "1" but can produce three different output states, "0", "1" or " Hi-Z " which is why it is called a "Tri" or "3-state" device.

Q.3. (a) Register A holds the 8-bits binary 11011001. Determine the B operand and the logic micro-operation to be performed in order to change the value in A to: (5)

(i) 01101101 (ii) 11111101

Ans.

(a) selective complement A= 11011001 B=10110100

(b) selective set A= 11011001 B= 11111101

Q.3. (b) Derive the control gates associated with program counter PC in basic computer. (5)

Ans. The inputs to this circuit come from the two decoders, the I flip-flop, and bits 0-11 of IR. The other inputs are bits 0-15 of AC to check if AC = 0 and to detect the sign bit AC(15); bits 0-15 of DR to check if DR = 0; and the values of seven flip-flops.

- The outputs of the control logic circuit are:

1. Signals to control the inputs of the nine registers
2. Signals to control the read and write inputs of memory
3. Signals to set, clear, or complement the F/Fs
4. Signals for S2 S1 S0 to select a register for the bus
5. Signals to control the AC adder and logic circuit

Control of Registers and Memory:

- Suppose that we want to derive the gate structure associated with the control inputs of AR. We scan to find all the statements that change the content of AR:

$$R'T_0: AR \leftarrow PC$$

$$R'T_2: AR \leftarrow IR(0\text{-}11)$$

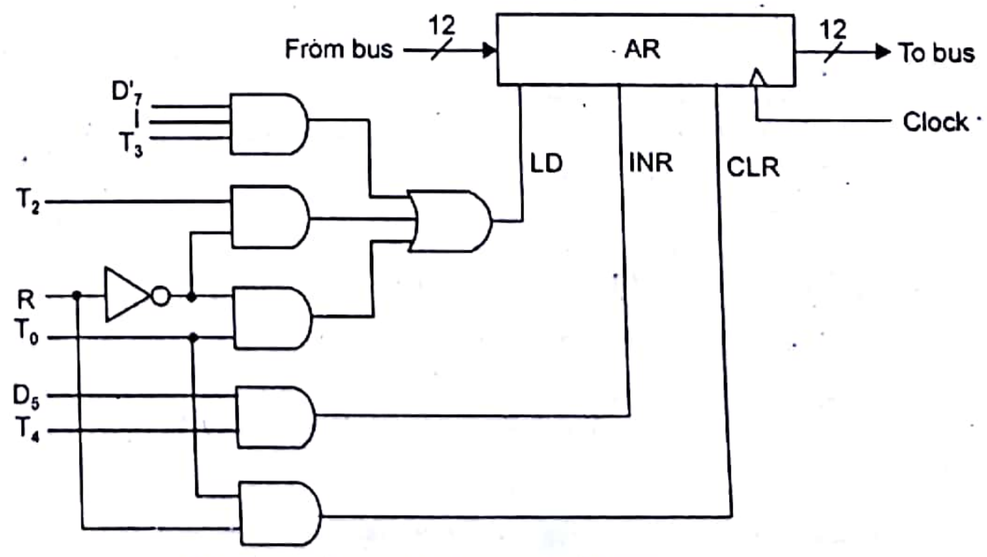


Fig. Control gates associated with AR.

$$D_7T_3: AR \leftarrow M[AR]$$

$$RT_0: AR \leftarrow 0$$

$$D_5T_4: AR \leftarrow AR + 1$$

- The control functions can be combined into three Boolean expressions as follows:

$$LD(AR) = RT_0 + RT_2 + D_7T_3$$

$$CLR(AR) = RT_0$$

$$INR(AR) = D_5T_4$$

The control logic gate associated with AR is shown in Fig.

Q.4. (a) Explain the three different types of instruction code formats of a basic computer namely as memory reference, register-reference, and I/O reference instruction. **(5)**

Ans. Memory-reference instruction: Uses 12-bit address, 1-bit to specify the addressing mode I and 3-bits for the Opcode.

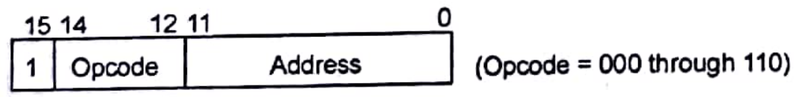
Opcode = 000 to 110 (I=0: 0xxx to 6xxx, I=1: 8xxx to Exxx).

Register-reference instruction: It is recognized by the Opcode 111 with a 0 in the left most bit of the instruction. It specifies an operation on or a test of the AC register.

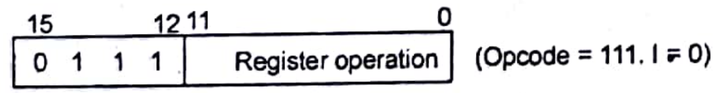
7xxx (7800 to 7001): CLA, CMA, etc.

Input-output instruction: It is recognized by the Opcode 111 with a 1 in the left most bit of the instruction. The remaining 12-bits are used to specify the type of I/O operation or test performed.

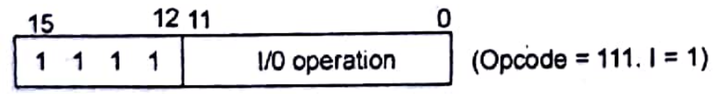
Fxxx (F800 to F040): INP, OUT, ION, etc



(a) Memory – reference instruction



(b) Register – reference instruction



(c) Input – output instruction

Figure: Basic computer Instruction formats.

Q.4. (b) An 8-bit register contains binary value 10011100. What is register value after arithmetic shift right? Starting from initial number 10011100 determine register value after an arithmetic shift left and state whether there is overflow. **(5)**

Ans. R = 10011100 Arithmetic shift right: 11001110 Arithmetic shift left: 00111000 overflow because a negative number changed to positive. Selective complement Selective set.

END TERM EXAMINATION [MAY-JUNE 2017]
FOURTH SEMESTER [B.TECH]
COMPUTER ORGANISATION AND ARCHITECTURE [ETCS-204]

Time : 3 Hrs. **M.M. : 75**

Note: *Attempt all questions as directed. Internal choice is indicated*

Q.1. (a) Differentiate between "hit" and "miss" with respect to cache memory. **(2.5)**

Ans. A **cache hit** means that the user's request for an object is satisfied by the cache entity (computer cache/http web cache/CDN edge server etc.), say, the requested object is exactly presented in the entity. As a result, the cache entity sends back object data to the end user to facilitate a quick response.On the other side, a **cache miss** regarding a particular cache server means that the requested object is not available in the storage , so the request message must be fowarded to cache entities at higher levels. In the worst case , the user request is delivered all the way along to the original storage server (such as an remote HTTP server.

Q.1. (b) How interrupts are handled? Explain. **(2.5)**

Ans. If there is an interrupt present then it will trigger the interrupt handler, the handler will stop the present instruction which is processing and save its configuration in a register and load the program counter of the interrupt from a location which is given by the interrupt vector table. After processing the interrupt by the processor interrupt handler will load the instruction and its configuration from the saved register, process will start its processing where it's left. This saving the old instruction processing configuration and loading the new interrupt configuration is also called as context switching.The interrupt handler is also called as Interrupt service routine (ISR). There are different types of interrupt handler which will handle different interrupts.

Q.1. (c) Explain little endian and big endian data storage machanisms. **(2.5)**

Ans. Big-endian and little-endian are terms that describe the order in which a sequence of bytes are stored in computer memory. Big-endian is an order in which the "big end" (most significant value in the sequence) is stored first (at the lowest storage address). Little-endian is an order in which the "little end" (least significant value in the sequence) is stored first. For example, in a big-endian computer, the two bytes required for the hexadecimal number 4F52 would be stored as 4F52 in storage (if 4F is stored at storage address 1000, for example, 52 will be at address 1001). In a little-endian system, it would be stored as 524F (52 at address 1000, 4F at 1001).

Q.1. (d) Express A*B+*(B*D+C*E) into reverse polish notation. **(2.5)**

Ans. REVERSE POLISH NOTATION: AB*BD*CE*+*+

Q.1. (e) Differentiate between direct and indirect instruction. **(2.5)**

Ans.Difference between direct and Indirect address Instructions are

Direct Address	Indirect Address
If the Address part has the address of an operand, then the instruction is said to have a direct address.	If the address bits of the instruction code are used as an actual operand, it is termed as indirect addressing.

Two memory references are needed to access data.	Three memory references are made to access the actual data
Limited address space	Large address space
No additional calculation to workout effective address	Effective address = Address (Address Operand)
Faster memory access	Usually slower than direct addressing mode.

Q.1. (f) Briefly list the types of interrupts with the help of suitable examples. **(2.5)**

Ans. The different types of interrupts are:

1. Hardware interrupts: If the signal for the processor is from external device or hardware is called hardware interrupts, Example: from keyboard we will press the key to do some action this pressing of key in Keyboard will generate a signal which is given to the do action, such interrupts are called hardware interrupts. Hardware interrupts can be classified into two types they are

- **Maskable Interrupt:** The hardware interrupts which can be delayed when a much highest priority interrupt has occurred to the processor
- **Non Maskable Interrupt:** The hardware which cannot be delayed and should process by the processor immediately,

2. Software Interrupts: Software interrupt can also divided in to two types they are

- **Normal Interrupts:** The interrupts which are caused by the software instructions are called normal interrupts.
- **Exception:** unplanned interrupts while executing a program is called Exception For example: while executing a program if we got a value which should be divided by zero is called a exception.

Q.1. (g) Differentiate between RISC and CISC. **(2.5)**

Ans.

	RISC	CISC
Acronym	It stands for 'Reduced Instruction Set Computer'.	It stands for 'Complex instruction Set Computer'.
Definition	The RISC processors have a smaller set of instructions with few addressing nodes.	The CISC processors have a larger set of instructions with many addressing nodes.
Memory unit	It has no memory unit and uses a separate hardware to implement instructions	It has a memory unit to implement complex instructions.
Program	It has a hard-wired unit of programming.	It has a micro-programming unit.
Design	It is a complex complier design.	It is an easy complier design
Calculations	The calculations are faster and precise	The calculations are slow and precise

Q.1. (h) Explain updating techniques used in cache design. **(2.5)**

Ans. As there are fewer cache lines than main memory blocks, an algorithm is needed for mapping main memory blocks into cache lines. Further, a means is needed for determining which main memory block currently occupies a cache line. The choice of the mapping function dictates how the cache is organized. Three techniques can be used: direct, associative, and set associative.

• **DIRECT MAPPING:** The simplest technique, known as direct mapping, maps each block of main memory into only one possible cache line.

The direct mapping technique is simple and inexpensive to implement.

• **ASSOCIATIVE MAPPING:** Associative mapping overcomes the disadvantage of direct mapping by permitting each main memory block to be loaded into any line of the cache

• **SET-ASSOCIATIVE MAPPING:** Set-associative mapping is a compromise that exhibits the strengths of both the direct and associative approaches. With set-associative mapping, block can be mapped into any of the lines of set

Q.1. (i) Explain Virtual memory. Also state "Locality of Reference" principle. **(2.5)**

Ans. In computing, virtual memory is a memory management technique that provides an "idealized abstraction of the storage resources that are actually available on a given machine" which "creates the illusion to users of a very large memory.

In computer science, **locality of reference**, also known as the principle of **locality**, is a term for the phenomenon in which the same values, or related storage locations, are frequently accessed, depending on the memory access pattern. There are two basic types of **reference locality** – temporal and spatial **locality**.

Q.1. (j) What are the different kinds of operation used in CPU design? **(2.5)**

Ans. The micro-operations in computers are classified into the following categories:

• Register transfer micro-operations: These type of micro operations are used to transfer from one register to another binary information.

• Arithmetic micro-operations : These micro-operations are used to perform on numeric data stored in the registers some arithmetic operations.

• Logic micro-operations: These micro operations are used to perform bit style operations / manipulations on non numeric data.

• Shift micro operations: As their name suggests they are used to perform shift operations in data store in registers.

Q.2. (a) Starting from an initial value R= 11011101, determine the sequence of binary values in R after a logical shift left, followed by a circular shift right, followed by a logical shift right and a circular shift. **(4)**

Ans.

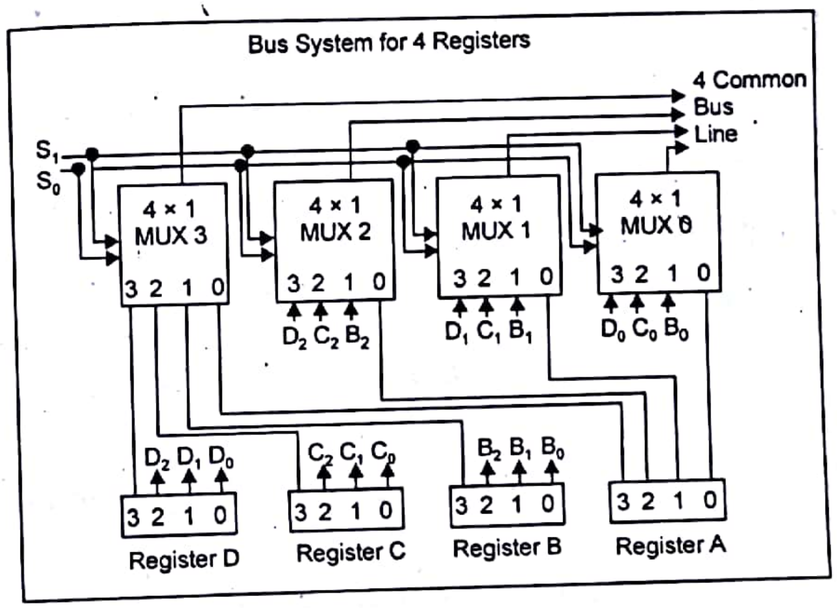
Operation	1	1	0	1	1	1	0	1
Logical shift left	1	0	1	1	1	0	1	0
Circular shift right	0	1	0	1	1	1	0	1
logical shift right	0	0	1	0	1	1	1	0
Circular shift left	0	1	0	1	1	1	0	0

So final answer would be 01011100

Q.2. (b) Draw an explain bus system for four registers. Also represent the conditional statement by two register transfer statements with control functions.

• If (P = 1) then (R -> R2) else if (Q = 1) then (R1 <- R3) **(6)**

Ans. A bus is a set of common wires that carries data between registers. – There is a separate wire for every bit in the registers. – There are also a set of control signals which determines which register is selected by the bus at a particular time. A bus can be constructed using multiplexer which enable a sets of registers to share a common bus for data transfer. The select lines S1 and S0 indicate which of four register will have its contents transferred to the bus. In general, a bus system will multiplex k registers of n bit each to produce a n–line common bus. It will require n k x 1 multiplexers. The bus is connected to the inputs of all destination registers. and will activate the load control of the selected register when it is ready to transfer data.



Statement

P : R <– R2 , Q: R1 <– R3

Q.2. (c) Define the term Micro-operation and micrco instruction with the help of an example. **(2.5)**

Ans. Microoperation : an elementary digital computer operation. **Micro-operations** (also known as a **micro-ops** or **μops**) are detailed low-level instructions used in some designs to implement complex machine instructions (sometimes termed *macro-instructions* in this context).

Example: The basic **arithmetic micro operations** are Decrement, Increment and Subtraction.

Microinstruction: an instruction stored in control memory.A micro-instruction is a simple command that makes the hardware operate properly. The format is unique to each computer. Example:

For example, although all are x86 chips, the microcode for Intel's Pentium 4, Pentium M and AMD's Athlon are not the same. The software programmer never sees microinstructions, and they are not documented for the public.

OR

Q.3. (a) Convert the decimal 61.5867 into its binary equivalent. **(2)**

Ans. Conversion steps:

1. Divide the number by 2.
2. Get the integer quotient for the next iteration.
3. Get the remainder for the hex digit.
4. Repeat the steps until the quotient is equal to 0.

$(61.587)_{10} = (111101.1001011001000101101)_2$

Q.3. (b) Convert – 6.75 (written in decimal) to floating point representation (single precision). **(4)**

Ans.

	sign	Exponent	Mantissa
Value	–1	2^2	1.6875
Encoded as:	1	129	5767168

Binary:

You entered –6.75

V ue actually stored in float: –6.75 +1

Error due to conversion: 0.00

Binary Representation 11000000110110000000000000000000 –1

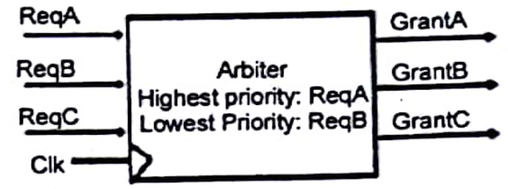
Hexadecimal Representation 0xc 0d80000

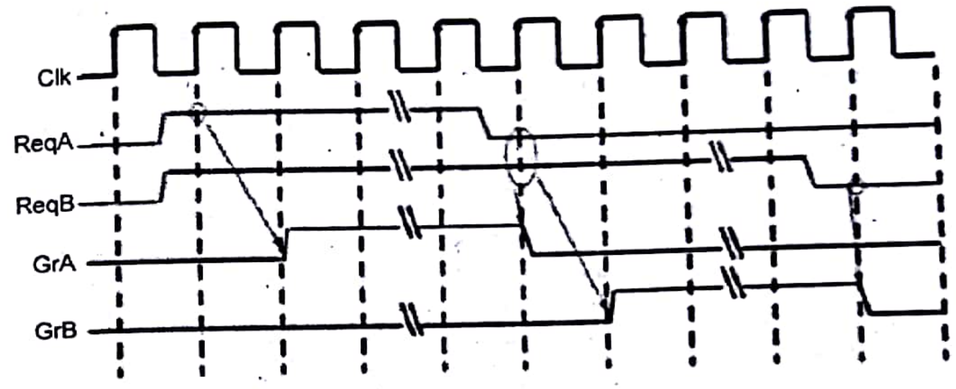
Q.3. (c) What do you mean by bus arbitration? Explain serial and parallel bus arbitration in detail. **(6.5)**

Ans. The device that is allowed to initiate data transfers on the bus at any given time is called the bus master. In a computer system there may be more than one bus master such as processor, DMA controller etc.They share the system bus. When current master relinquishes control of the bus, another bus master can acquire the control of the bus. Bus arbitration is the process by which the next device to become the bus master is selected and bus mastership is transferred to it. The selection of bus master is usually done on the priority basis.There are two approaches to bus arbitration: Centralized and distributed.

In serial bus, if there are multiple masters, then multiple master decides whether it acquires the bus or not using data pattern. In case where master needs to drive 0, it would drive other wise it would release. Master would always figure out whehter it has acquired the bus or not by checking the bus being equal to 1 when it is not driving. If it is driven as 0, then it would understand that it is occupied by other master. So it would release the bus.In the centralized, parallel arbitration scheme, the devices independently request the bus by using multipliers.A centralized arbiter chooses from among the devices requesting bus access and notifies the selected device that it is now the bus master via one of the grant line. Consider an example where A has the highest priority and C the lowest. Since A has the highest priority, Grant A will be asserted even though both requests A and B are asserted. Device A will keep Request A asserted until it no longer needs the bus so when Request A goes low, the arbiter will disable Grant A. Since

Request B remains asserted (Device B has not gotten the bus yet) at this time, the arbiter will then assert Grant B to grant Device B access to the bus.Similarly, Device B will not disable Request B until it is done with the bus.





Q.4. (a) Explain instruction cycle(fetch) and (decode) in detail. Also explain the working of computer registers used in it. **(6.5)**

Ans. Each computer's CPU can have different cycles based on different instruction sets, but will be similar to the following cycle:

1. Fetch the instruction: The next instruction is fetched from the memory address that is currently stored in the program counter (PC), and stored in the instruction register (IR). At the end of the fetch operation, the PC points to the next instruction that will be read at the next cycle.

2. Decode the instruction: During this cycle the encoded instruction present in the IR (instruction register) is interpreted by the decoder.

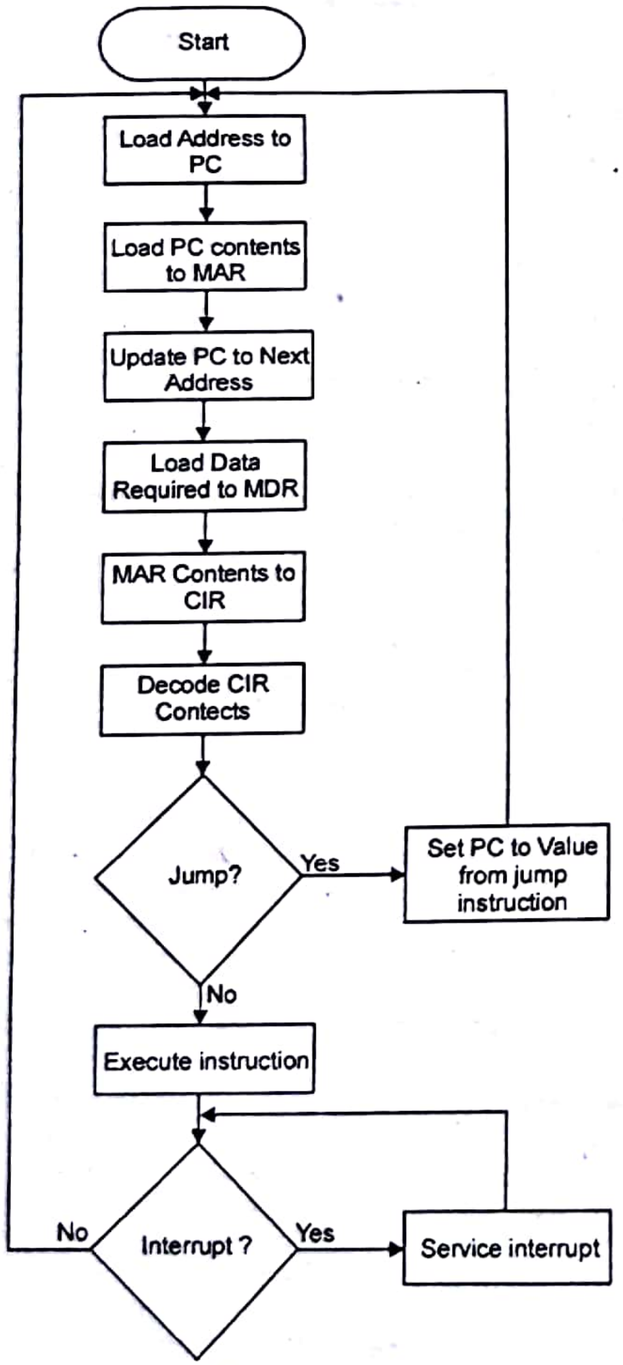
3. Read the effective address: In case of a memory instruction (direct or indirect) the execution phase will be in the next clock pulse. If the instruction has an indirect address, the effective address is read from main memory, and any required data is fetched from main memory to be processed and then placed into data registers (Clock Pulse: T_3). If the instruction is direct, nothing is done at this clock pulses. If this is an I/O instruction or a Register instruction, the operation is performed (executed) at clock Pulse.

4. Execute the instruction: The control unit of the CPU passes the decoded information as a sequence of control signals to the relevant function units of the CPU to perform the actions required by the instruction such as reading values from registers, passing them to the ALU to perform mathematical or logic functions on them, and writing the result back to a register. If the ALU is involved, it sends a condition signal back to the CU. The result generated by the operation is stored in the main memory, or sent to an output device. Based on the condition of any feedback from the ALU, Program Counter may be updated to a different address from which the next instruction will be fetched.

The cycle is then repeated. The various registers used are :

MAR stand for *Memory Address Register:* This register holds the memory addresses of data and instructions. This register is used to access data and instructions from memory during the execution phase of an instruction.

Program Counter: The **program counter (PC)**, commonly called the **instruction pointer** (IP) in Intel x86 microprocessors, and sometimes called the **instruction address register**, or just part of the instruction sequencer in some computers, is a processor register



Instruction cycle

Accumulator Register: This Register is used for storing the Results those are produced by the System. When the CPU will generate

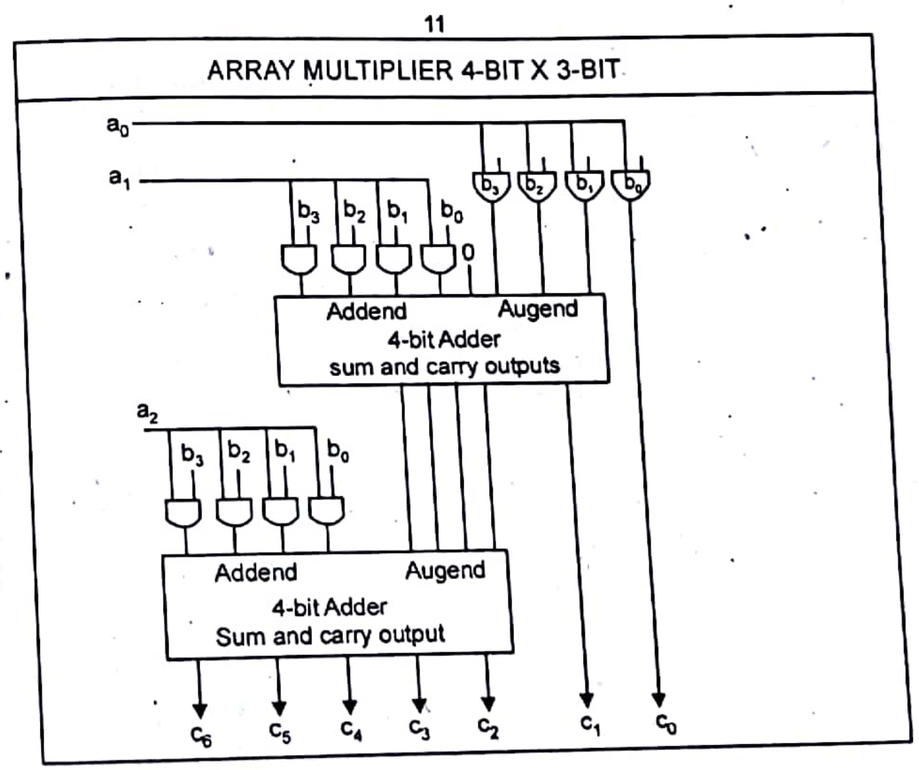
Index Register: A hardware element which holds a number that can be added to (or, in some cases, subtracted from) the address portion of a computer instruction to form an effective address. Also known as **base register**

Memory Buffer Register*:** MBR stand for ***Memory Buffer Register. This register holds the contents of data or instruction read from, or written in memory. It means that this register is used to store data/instruction coming from the memory or going to the memory.

Data Register: A register used in microcomputers to temporarily store data being transmitted to or from a peripheral device.

Q.4. (b) Draw an array mutliple circuit that multiplies as binary number of 4 bits with a number of 3 bits. **(3)**

Ans.



Q.4.(c) For the expression X = (A + B)* (C + D) when evaluated on stack machine how many number of machine instructions an required. **(3)**

Ans. Total 8 instuctions are required. Of which 5 are the stack operation instructions.

- Evaluate X = (A + B) * (C + D)
- PUSH A TOS ← A
- PUSH B TOS ← B
- ADD TOS ← (A + B)
- PUSH C TOS ← C

- PUSH D TOS ← D
- ADD TOS ← (C + D)
- MUL TOS ← (C + D)*(A+B)
- POP X M[X] ← TOS

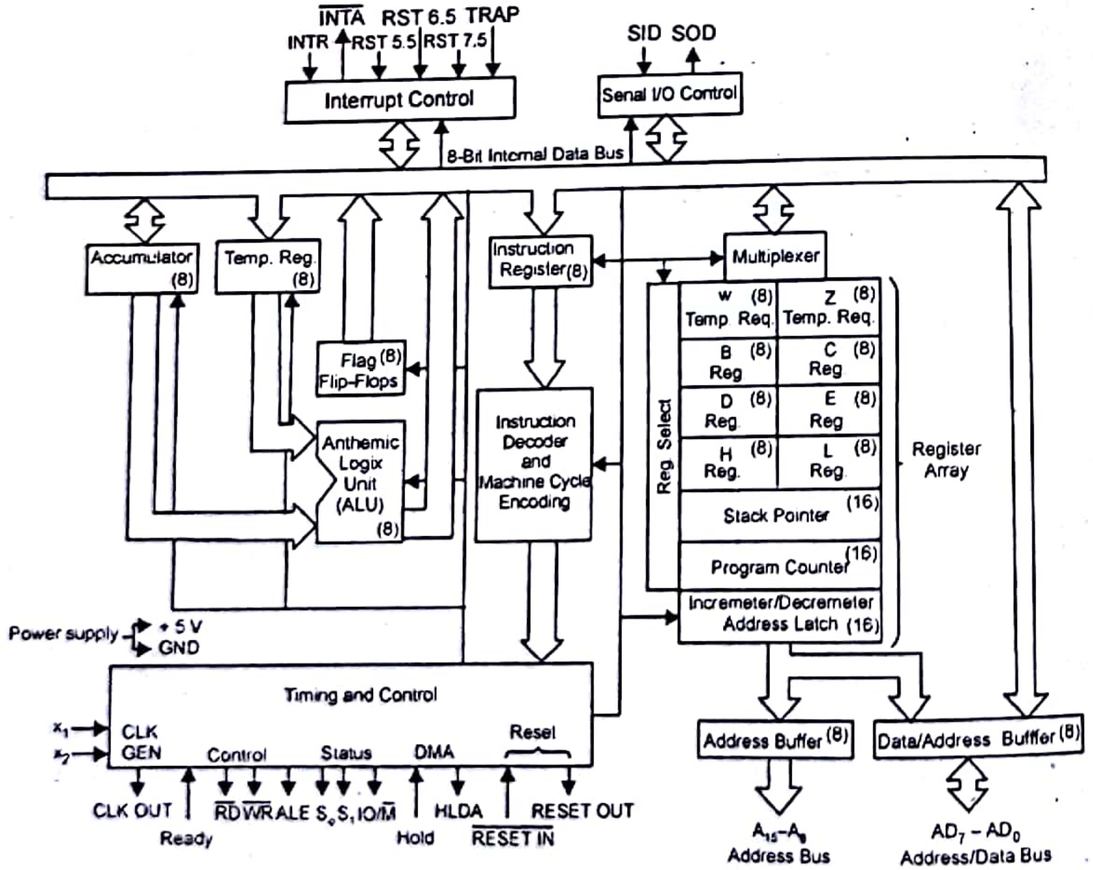
OR

Q.5. (a) What are different types of flags used in 8085 microprocessor? On what architecture is 8085 based? Explain. **(6.5)**

Ans. ALU of 8085 have five flip flops whose states (set/reset) are determined by the result data of other registers and accumulator. They are called as Zero, Carry, Sign, Parity and Auxiliary-Carry flags.

- Zero Flag (Z): When an arithmetic operation results in *zero* , the flip-flop called the Zero flag - which is set to one.
- Carry flag (CY): After an addition of two numbers, if the sum in the accumulator is larger than eight bits, then the flip-flop uses to indicate a *carry* called the Carry flag – which is set to one.
- S-Sign (S): It is set to 1, if bit D7 of the result = 1; otherwise reset. D7 is the first digit of a binary number.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z		AC		P		CY



- P-Parity (P): If the result has an even number of 1s, the flag is set to 1; for an odd number of 1s the flag is reset.
- AC-Auxiliary Carry (AC): In an arithmetic operation, when a carry is generated by digit D3 and passed to digit D4, the AC flag is set. Generally this flag is used internally for Binary Coded Decimals (BCD).

8085 is pronounced as "eighty-eighty-five" microprocessor. It is an 8-bit microprocessor designed by Intel in 1977 using NMOS technology.

It has the following configuration

- 8-bit data bus
- 16-bit address bus, which can address upto 64KB
- A 16-bit program counter
- A 16-bit stack pointer
- Six 8-bit registers arranged in pairs: BC, DE, HL
- Requires +5V supply to operate at 3.2 MHZ single phase clock

Q.5. (b) Consider the following program segment used to execute on a hypothetical machine instructors are: Inst 1, Inst2, Inst3, Inst3, Inst5, Inst6, Inst7

Size(in words): 2,1,1,2,1,2,1 respectively.

Assume the word size of the instruction is 32 bit and the program has been loaded into the memory with starting address of 1000 (Decimal onwards). What could be the value present in PC during the execution of Inst6. (Memory byte addressable). **(6)**

Ans. 32 bits= 4 bytes

So 1 word= 4 bytes and 2 words = 8 bytes

instr1- location is 1000-1007 as it is 2 words

instr2- location is 1008-1011 , it is 1 word

instr3- location is 1012-1015 , it is 1 word

instr4- location is 1016-1023 ,it is 2 words

instr 5 - location is 1024- 1027 , it is 1 word

instr 6- location is 1028- 1035 , it is 2 words

therefore, the location is 1036

Q.6. (a) Explain instruction formats with examples. **(6)**

Ans. Zero-Address Instructions

A stack-organized computer does not use an address field for the instructions **ADD** and **MUL**. The **PUSH** and **POP** instructions, however, need an address field to specify the operand that communicates with the stack.

```
PUSH A          //Top of Stack <- A
PUSH B          //Top of Stack <- B
```

```
ADD          //Top of Stack <- (A + B)
PUSH C       //Top of Stack <- C
PUSH D       //Top of Stack <- D
ADD          //Top of Stack <- (C + D)
MUL          //Top of Stack <- (C + D) * (A + B)
POP X        //Top of Stack <- M[X] ≺ Top of Stack
```

One-Address Instructions

One-address instructions use an Accumulator (AC) register for all data manipulation.

Assume the AC contains the result of all operations.

```
LDA   A      //AC  <- M[A]
ADD   B      //AC  <- AC + M[B]
STA   T      //M[T] <- AC
LDA   C      //AC  <- M[C]
ADD   D      //AC  <- AC + M[D]
MUL   T      //AC  <- AC * M[T]
STA   X      //M[X] <- AC
```

Two-Address Instructions

Two-address instructions are the most common in commercial computers.

Each address field can specify either a processor register or a memory word.

```
MOV R1, A     //R1   <- M[A]
ADD R1, B     //R1   <- R1 + M[B]
MOV R2, C     //R2   <- M[C]
ADD R2, D     //R2   <- R2 + M[D]
MUL R1, R2    //R1   <- R1 * R2
MOV X, R1     //M[X] <- R1
```

Three-Address Instructions

Computers with three-address instruction formats can use each address field to specify either a processor register of a memory operand.

```
ADD R1, A, B     //R1   <- M[A] + M[B]
ADD R2, C, D     //R2   <- M[C] + M[D]
MUL X, R1, R2    //M[X] <- R1 * R2
```

Q.6. (b) Consider a hypothetical processor which supports expand opcode technique. A 32 bit instruction is place in 256MW memory. If there exist 10, one address instruction then how many zero-address instruction are possible. (6.5)

Ans.

8	12	12	= 32 Bits	Two address instructions
Upcode	Address 1	Address 2		

$2^8 = 256$ combinations.

256–250 = 6 combinations can be used for one address

Upcode	Address	One address Instructions

6×2^{12}

Maximum number of one address instruction:

$$= 6 \times 2^{17} = 24,576$$

OR

Q.7. (a) Differentiate between hardwired control and micro programmed control. Is it possible to have a hardwired control associated with a control memory? (6.5)

Ans. A hardwired control differs from microprogrammed control in the following ways:

Hardwired Control Unit	Microprogramed Control Unit
The Control unit whose control signals are generated by the hardware through a sequence of instruction is called a hardwired control unit.	The control unit whose control signals are generated by the data stored in control memory and constitude a program on the small scale is called a microprogrammed control unit.
The control logic of a hardwired control is implemented with gates, flip flops, decoders etc.	The control logic of a micro-programmed control is the instructions that are stored in control memory to initiate the required sequence of microoperations.
Wiring changes are made in the hardwired control unit if there are any changes required in the design.	Changes in a microprogrammed control unit are done by updating the microprogram in control memory.
Hardwired control unit are faster and known to have complex structure.	Microprogrammed control unit is comparatively slow compared but are simple in structure.

To execute instructions, a computer's processor must generate the control signals used to perform the processor's action in the proper sequence. This sequence of action can either be executed by another processor's software (for example in software emulation or simulation of processor) or in hardware. Hardware methods fall into two categories: the processor's hardware signals are generated either by hardwire control, in which the instruction bits directly generate the signals, or by microprogrammed control in which a dedicated microcontroller executes a microprogram to generate the signals.

Q.7.(b) Explain various memory based addressing modes in detail with the help of suitable diagram. **(6)**

Ans. Addressing Modes in 8085

These are the instructions used to transfer the data from one register to another register, from the memory to the register, and from the register to the memory without any alteration in the content. Addressing modes in 8085 is classified into 5 groups "

Immediate addressing mode

In this mode, the 8/16-bit data is specified in the instruction itself as one of its operand. **For example:** MVI K, 20F: means 20F is copied into register K.

Register addressing mode

In this mode, the data is copied from one register to another. **For example:**MOV K, B: means data in register B is copied to register K.

Direct addressing mode

In this mode, the data is directly copied from the given address to the register. **For example:** LDB 5000K: means the data at address 5000K is copied to register B.

Indirect addressing mode

In this mode, the data is transferred from one register to another by using the address pointed by the register. **For example:** MOV K, B: means data is transferred from the memory address pointed by the register to the register K.

Implied addressing mode

This mode doesn't require any operand; the data is specified by the opcode itself. **For example:** CMP.

Q.8. (a) Define DMA. Explain its need. Explain DMA transfer in detail with the help of suitable diagram.

Ans. Stands for "Direct Memory Access." DMA is a method of transferring data from the computer's RAM to another part of the computer without processing it using the CPU. While most data that is input or output from your computer is processed by the CPU, some data does not require processing, or can be processed by another device. In these situations, DMA can save processing time and is a more efficient way to move data from the computer's memory to other devices.

For example, a sound card may need to access data stored in the computer's RAM, but since it can process the data itself, it may use DMA to bypass the CPU. Video cards that support DMA can also access the system memory and process graphics without needing the CPU. Ultra DMA hard drives use DMA to transfer data faster than previous hard drives that required the data to first be run through the CPU.

DMA Transfer Types

Memory To Memory Transfer

In this mode block of data from one memory address is moved to another memory address. In this mode current address register of channel 0 is used to point the source address and the current address register of channel is used to point the destination address in the first transfer cycle, data byte from the source address is loaded in the

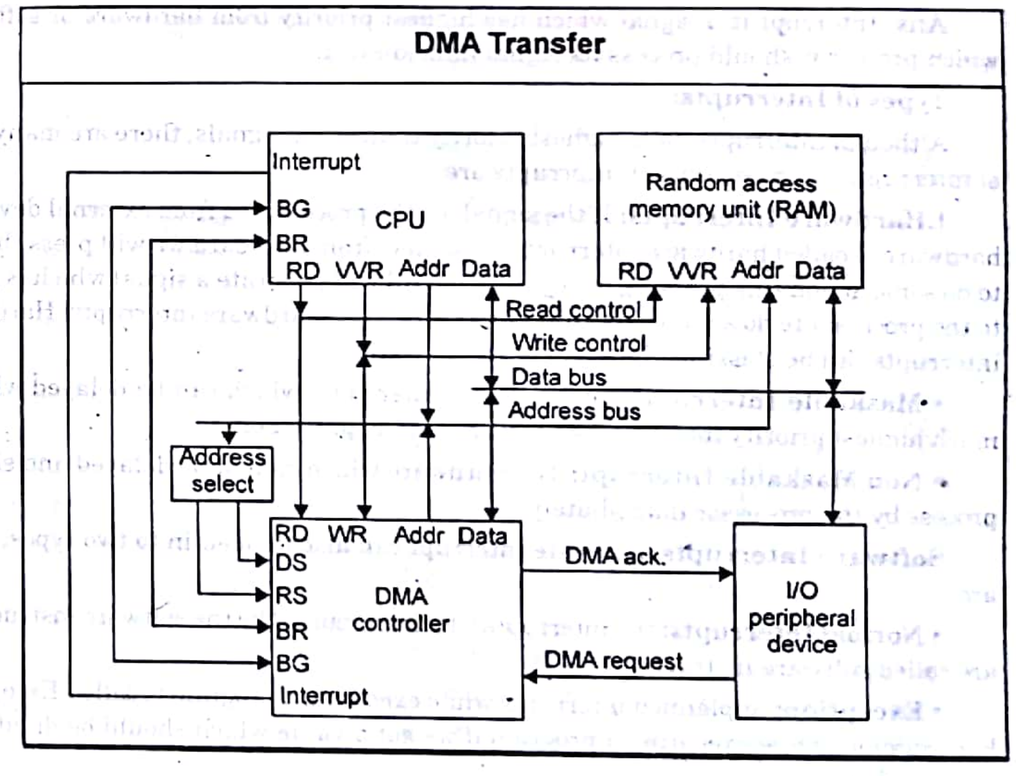
temporary register of the DMA controller and in the next transfer cycle the data from the temporary register is stored in the memory pointed by destination address.

Auto initialize

In this mode, during the initialization the base address and word count registers are loaded simultaneously with the current address and word count registers by the microprocessor. The address and the count in the base registers remain unchanged throughout the DMA service.

After the first block transfer i.e. after the activation of the EOP signal, the original values of the current address and current word count registers are automatically restored from the base address and base word count register of that channel.

DMA controller



The controller is integrated into the processor board and manages all DMA data transfers. Transferring data between system memory and an 110 device requires two steps. Data goes from the sending device to the DMA controller and then to the receiving device. The microprocessor gives the DMA controller the location, destination, and amount of data that is to be transferred. Then the DMA controller transfers the data, allowing the microprocessor to continue with other processing tasks. When a device needs to use the Micro Channel bus to send or receive data, it competes with all the other devices that are trying to gain control of the bus. This process is known as arbitration. The DMA controller does not arbitrate for control of the BUS instead; the I/O device that is sending or receiving data (the DMA slave) participates in arbitration.

Q.8. (b) Main memory size is 128 KB, cache size is 16 KB, block size is 256B. Using direct bit mapping what is the no of tags bits in physical address and what is the size of tag directory. Assume memory is byte addressable. **(5)**

Ans.

tag	index	block

Tag bits= address bit length – exponent of index- exponent of offset

Index bits = log(cache size)= log $(16*2^{10})$ = 14

Block offset = 8

Tag bits= 32- 14- 8 = 10

Tag directory= no. of lines * no. of tag bits = 14* 10 = 140

Q.9. Write short notes on:

(a) Types of interrupt and Memory hierarchy **(6.5)**

Ans. Interrupt is a signal which has highest priority from hardware or software which processor should process its signal immediately.

Types of Interrupts:

Although interrupts have highest priority than other signals, there are many type of interrupts but basic type of interrupts are

1.Hardware Interrupts: If the signal for the processor is from external device or hardware is called hardware interrupts. Example: from keyboard we will press the key to do some action this pressing of key in keyboard will generate a signal which is given to the processor to do action, such interrupts are called hardware interrupts. Hardware interrupts can be classified into two types they are

- **Maskable Interrupt:** The hardware interrupts which can be delayed when a much highest priority interrupt has occurred to the processor.
- **Non Maskable Interrupt:** The hardware which cannot be delayed and should process by the processor immediately.

Software Interrupts: Software interrupt can also divided in to two types. They are

- **Normal Interrupts:** the interrupts which are caused by the software instructions are called software instructions.
- **Exception:** unplanned interrupts while executing a program is called Exception. For example: while executing a program if we got a value which should be divided by zero is called a exception.

In computer architecture, the **memory hierarchy** separates computer storage into a hierarchy based on response time. Since response time, complexity, and capacity are related, the levels may also be distinguished by their performance and controlling technologies Memory hierarchy affects performance in computer architectural design, algorithm predictions, and lower level programming constructs involving locality of reference.

There are four major storage levels.

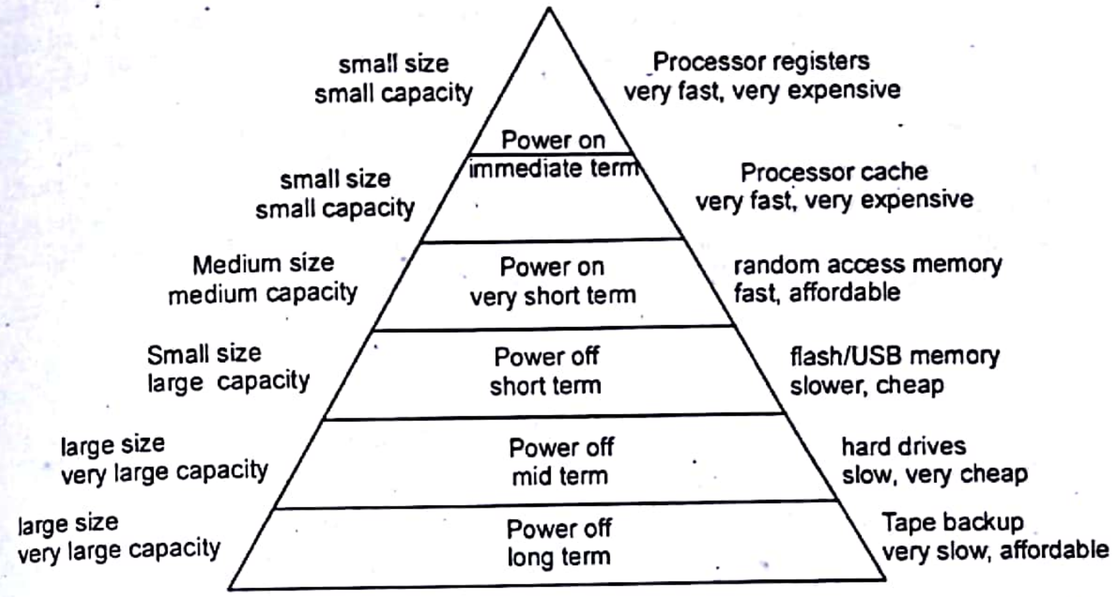
1.*Internal* – Processor registers and cache.

2.Main – the system RAM and controller cards.

3.On-line mass storage – Secondary storage.

4.Off-line bulk storage – Tertiary and Off-line storage.

Compare Memory Hierarchy



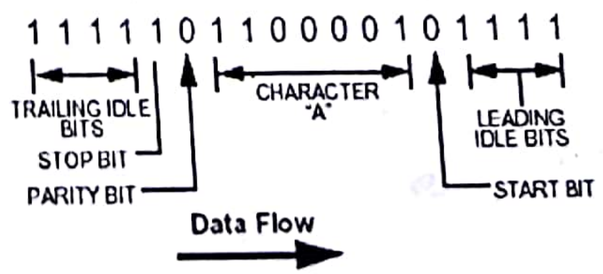
Q.9. (b) Serial communicatiorr and RS-232-C. **(6)**

Ans. In telecommunication and data transmission, **serial communication** is the process of sending data one bit at a time, sequentially, over a communication channel or computer bus. This is in contrast to parallel communication, where several bits are sent as a whole, on a link with several parallel channels.

Serial communication is used for all long-haul communication and most computer networks, where the cost of cable and synchronization difficulties make parallel communication impractical. Serial computer buses are becoming more common even at shorter distances, as improved signal integrity and transmission speeds in newer serial technologies have begun to outweigh the parallel bus's advantage of simplicity (no need for serializer and deserializer, or SerDes) and to outstrip its disadvantages (clock skew, interconnect density). The migration from PCI to PCI Express is an example.

Rs232 protocol transmission involves the sending of data one bit at a time, over a single communications line. In contrast, parallel communications require at least as many lines as there are bits in a word being transmitted (for an 8-bit word, a minimum of 8 lines are needed). RS232 Serial transmission is beneficial for long distance communications, whereas parallel is designed for short distances or when very high transmission rates are required.

Transmission Example



RS232 Standards

One of the advantages of the RS232 protocol is that it lends itself to transmission over telephone lines. The serial digital data can be converted by modem, placed onto a standard voice-grade telephone line, and converted back to serial digital data at the receiving end of the line by another modem.

Officially, RS-232 is defined as the "Interface between data terminal equipment and data communications equipment using serial binary data exchange." This definition defines data terminal equipment (DTE) as the computer, while data communications equipment (DCE) is the modem. A modem cable has pin-to-pin connections, and is designed to connect a DTE device to a DCE device.