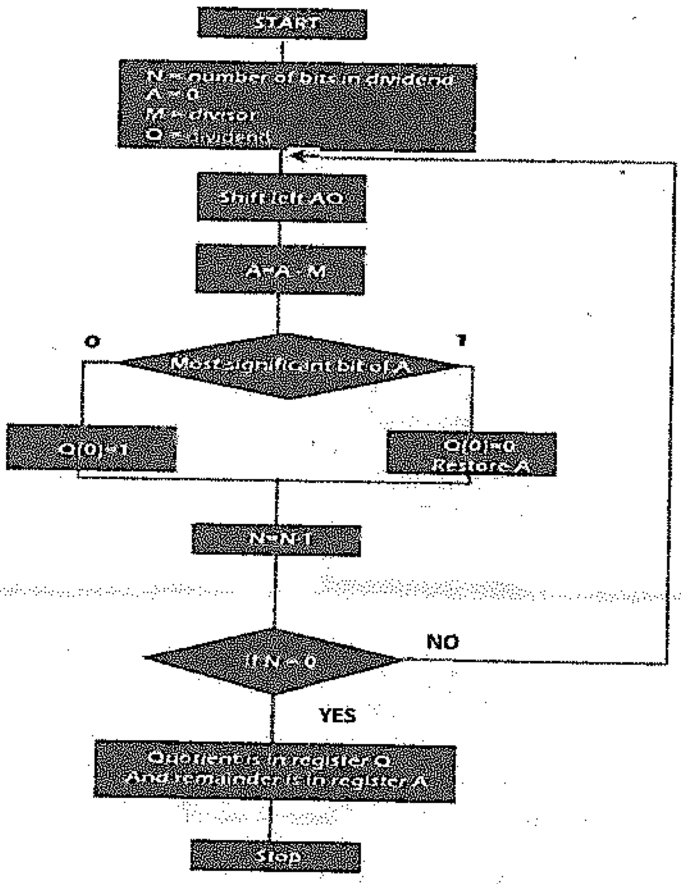
Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Annexure – III (Answer Key)

Academic Year (2022-23)
Year:Third Semester: V
Program: B. Tech. (Computer Engineering) **Max. Marks: 75**
Subject: Processor organization and Architecture **Time:**
Date: **Duration: 3 Hours**

REGULAR EXAMINATION

ANSWER KEY

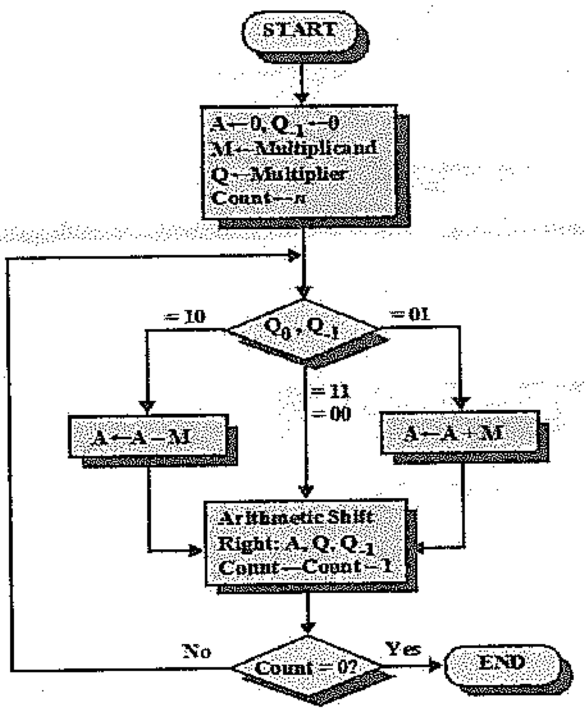
Question No.		Max. Marks
Q1 (a)	**Restoring Division Algorithm Flowchart**  START N = number of bits in dividend A = 0 M = divisor Q = Dividend Shift left AQ A=A - M 0 1 Most significant bit of A Q[0]=1 Q[0]=0 Restore A N=N-1 NO If N = 0 YES Quotient is in register Q And remainder is in register A Stop Q= Dividend = 11 (1011) M= Divisor = 3 (00011)	[10]

N	M	A	Q	Operation
4	00011	00000	1011	Initialize
	00011	00001	011_	Shift left AQ
	00011	11110	011_	A = A - M
	00011	00001	0110	Q[0] = 0 And restore A
3	00011	00010	110_	Shift left AQ
	00011	11111	110_	A = A - M
	00011	00010	1100	Q[0] = 0
2	00011	00101	100_	Shift left AQ
	00011	00010	100_	A = A - M
	00011	00010	1001	Q[0] = 1
1	00011	00101	001_	Shift left AQ
	00011	00010	001_	A = A - M
	00011	00010	0011	Q[0] = 1

After Division
Quotient = Q= 3 (0011)
Remainder = A=2 (0010)

OR

Booths Algorithm



M = -7 = (1001) and –M = M' + 1 = 0111

Q = 3 = (0011)
Value of SC = 4, because the number of bits in Q is 4.
Qn=1Qn=1 according to last bit of Q and Qn+1Qn+1 set as 0 at initially.

Q_n	Q_{n+1}	M = 1001 -M = M' +1 = 0111	AC	Q	Q_{n+1}	SC
1	0		0000	0011	0	4
		10 Hence, AC = AC - M	AC = AC - M AC = 0000 – 0111 = 0111 0111			
		Arithmetic Right Shift operation on AC Q Q_{n+1}	0011	1001	1	3
1	1	11 Hence, Arithmetic Right Shift operation on AC Q Q_{n+1}	0001	1100	1	2
0	1	01 Hence, AC = AC + M	AC = AC + M AC = 0001 + 1001 = 1010 1010			
		Arithmetic Right Shift operation on AC Q Q_{n+1}	1101	0110	0	1
0	0	00 Hence, Arithmetic Right Shift operation on AC Q Q_{n+1}	1110	1011	0	0

As, (-7) * 3 = -21

Value stored in AC & Q registers = 11101011

(-7) * 3 = 2's complement of
11101011= −(00010100+1)2=−(00010101)2−(00010100+1)2=−(00010101)2
= - (16 + 4 + 1) = -21

Q1 (b) **Differentiate between Von Neumann Model and Harvard Architecture** [5]

Von-Neumann Architecture	Harvard Architecture
The Von Neumann Architecture is an ancient type of computer architecture that follows the concept of a stored-program computer.	Harvard Architecture is a modern type of computer architecture that follows the concept of the relay-based model by Harvard Mark 1.
Single memory to be shared by both code and data.	Separate memories for code and data.
Processor needs to fetch code in a separate clock cycle and data in another clock cycle. So it requires two clock cycles.	Single clock cycle is sufficient, as separate buses are used to access code and data.
The speed of execution of the Von Neumann Architecture is comparatively slower.	The overall speed of execution of Harvard Architecture is comparatively faster.
Simple in design.	Complex in design.
This method comes to play in the case of small computers and personal computers.	This architecture is best for signal processing as well as microcontrollers.
This architecture basically requires less space.	This architecture comparatively requires more space.
It is comparatively cheaper in cost than Harvard Architecture.	It is comparatively more expensive than the Von Neumann Architecture.

Q2 (a)

Illustrate Cache Mapping techniques with neat Sketch

[8]
Any 2
techniques

Mapping Techniques

1) **Associative mapping**

1) The fastest and most flexible cache organization uses an associative memory.
2) In associative mapping cashes are made of associative memory.
3) Associative memory is used to store both the address and content of memory word.
4) It permits any location in cache to store any word from main memory.
5) That is it enables any word from main memory at any place in cache memory.
6) The figure 3.1 shows three words presently stored in the cache.
7) The address value of 15 bits is shown as a five digit octal number and its corresponding 12 bit word is shown as a four digit octal number.
8) A CPU address of 15 bits is placed in the argument register and the associative memory is searched for a matching address.
9) If the address is found, the corresponding 12 bit data is read and sent to the CPU.
10) If the address is not matched, then CPU accesses the main memory for the data.
11) The address and data pair is then transferred to the associative cache memory.
12) If the cache is full, an address-data pair must be displaced to make room for a pair that is needed and not presently in the cache.

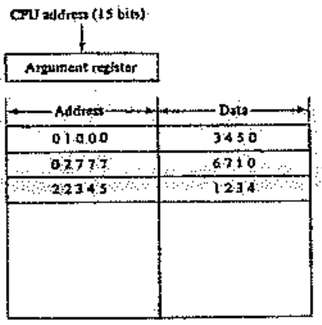


Fig:3.1

2) **Direct Mapping**

1) Associative memories are expensive compared to random-access memories because of the added matching logic associated with each cell.
2) It maps each block of main memory into only one possible cache line.
3) That is it assigns each memory block to a specific line in the cache.
4) The CPU address of 15 bits is divided into two fields.
5) The nine least significant bits constitute the *index field and the remaining six bits form the tag field.*

6) The number of bits in the index field is equal to the number of address bits required to access the cache memory.
7) The number of bits for tag field is the number of bits required to represent the address of Main Memory - number of bits required to represent the address of Cache Memory.
8) In the general case, there are 2^k words in cache memory and 2^n words in main memory.
9) The n-bit memory address is divided into two fields as *k* bits for the *index* field and *n - k* bits for the *tag* field.
10) The direct mapping cache organization uses the n-bit address to access the main memory and the k-bit index to access the cache.
11) The internal organization of the words in the cache memory is as shown in figure 3.2

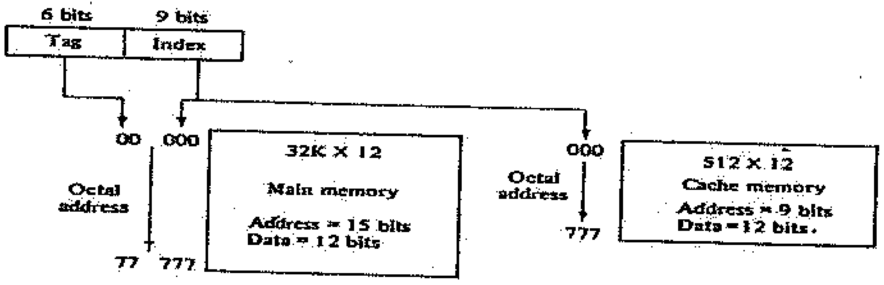
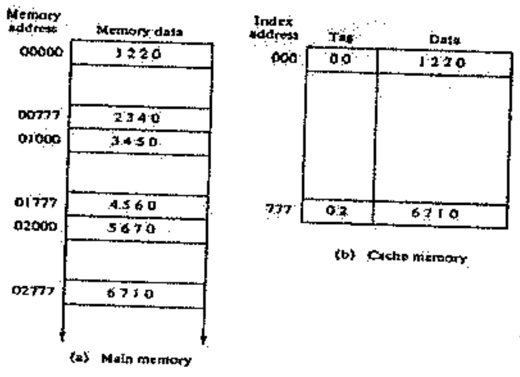


Fig: 3.2

1) Eg: Suppose that the CPU now wants to access the word at address 02777.
2) So the index value is 777 and tag value is 02.
3) The index value 777 is used as address to access the cache memory.
4) In the figure, index address 777 is available, and then tag value 02 is compared with tag value in the cache memory associated with index address 777.
5) Here tag value of CPU address is match with tag value of cache with index address also.
6) So desired data is available in cache memory.
7) So it is hit operation.



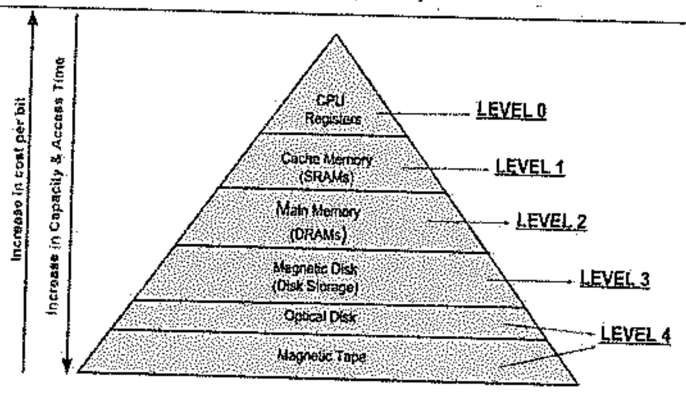
3) **Set-Associative Mapping**

1) To overcome the disadvantage of direct mapping, i.e. two words with the same index in their address but with different tag values cannot reside in cache memory at the same time, Set- associative mapping is proposed.
2) Here each word of cache can store two or more words stored under the same index address, creating a Set.
3) Each data word is stored together with its tag.
4) The number of tag-data items in one word is said to form a set.
5) An example of a set-associative cache organization for a set size of two is shown in Figure 3.3
6) Each index address refers to two data words and their associated tags.
7) Each tag requires six bits and each data word has 12 bits, so the word length is 2(6 + 12) = 36 bits.
8) An index address of nine bits can accommodate 512 words.
9) Thus, the size of cache memory is 512*36.
10) It can accommodate 1024 words of main memory since each word of cache contains two data words.
11) As Shown in fig 3.4 the words stored at addresses 01000 and 02000 of main memory are stored in cache memory at index address 000 with different tag values 01 and 02 respectively.
12) Similarly, the words at addresses 02777 and 00777 of main memory are stored in cache memory at index address 777 with different tag values 01 and 02 respectively.
13) When the CPU generates a memory request, the index value of the address is used to access the cache.
14) The tag field of the CPU address is then compared with both tags in the cache. If tags are match then it is a *hit* operation.
15) If the tag field of the CPU address and tags in the cache are not matched then it is a *miss* Operation.
16) When a miss occurs in a set-associative cache and the set is full, it is necessary to replace one of the tag-data items with a new value.

Index	Tag	Data	Tag	Data
000	01	3450	02	5670
777	02	6710	00	2340

Q2 (b)	**Explain Memory Hierarchy and its characteristics.**	[7]

Memory Hierarchy:



MEMORY HIERARCHY DESIGN

Fig:4.1

- In our computer system, a processor and many memory devices have been used.
- However, the main problem is, these devices are expensive.
- So the memory organization of the system is done with the help of the memory hierarchy.
- It has various levels of memory with different access times and performance rates.
- But all these can give us an exact purpose, such that the access time can be reduced.
- Therefore the memory hierarchy was developed based on the program.
- The memory hierarchy is the arrangement of various types of storage on a computing system based on access speed.
- It organizes computer storage according to response time.
- Since response time, complexity, and capacity are all connected, the levels can also be distinguished by their performance and controlling technologies.
- As shown in fig: 4.1 the computer memory has a pyramid-like structure.
- It is used to describe the different levels of memory.
- This Memory Hierarchy Design is divided into 2 types:

  A) Primary or internal memory
  It consists of CPU registers, Cache Memory, Main Memory, and these are directly accessible by the processor.

  B) Secondary or external memory
  It consists of a Magnetic Disk, Optical Disk, Magnetic Tape, which are accessible by processor via I/O Module.

A) **Internal memory**

- It is also called primary memory.
- Internal memory is a part of computer that, when running, can store small amounts of data that need to be accessed quickly.
- It consists of RAM, ROM, and cache memory.

B) **External memory**

- It is also known as secondary memory.
- Since it has a huge capacity, it stores massive data.

	• The critical property of external memory is that stored information will not be lost whenever the computer switches off. • It consists of magnetic tape, a magnetic disk, and an optical disk. • The memory characteristics mainly include: • **Access Time** • The access time in the memory hierarchy of a computer system is the interval of the time between the data availability and request to read or write. • **Capacity** • It is the amount of information that can be stored. • The capacity increases as we move from top to bottom in the hierarchy • **Performance** • In old times, designing a computer system was done without memory hierarchy. • The gap of speed between the main memory and the CPU registers is enhanced because of the huge inconsistency in access time, which will cause the lower performance of the system. • So, the enhancement in the memory was mandatory. • This enhancement was designed in the memory hierarchy model due to which the system's performance increase. • **Cost per bit** • As we will move from bottom to top in the system's hierarchy, the cost per bit increases, • i.e., Internal Memory is costlier than External Memory.	
Q3 (a)	**Addressing modes of 8086** • The way of specifying data to be operated by an instruction is known as **addressing modes.** • This specifies that the given data is an immediate data or an address. • It also specifies whether the given operand is register or register pair. • Addressing mode indicates a way of locating data or operands **1) Immediate addressing mode** • In this type of addressing, immediate data is a part of instruction, and appears in the form of successive byte or bytes. • The immediate addressing mode is commonly used to load a register or memory with some initial data. • **Example:** MOV AX, 0005H. • In the above example, 0005H is the immediate data directly stored in AX register. • The immediate data may be 8- bit or 16-bit in size. **2) Register addressing mode** • In the register addressing mode, the data is stored in a register and it is referred using the particular register. • In this type of addressing mode both the operands are registers. • **Example:** MOV BX, AX • The contents of Ax register are transferred to BX register. **3) Direct addressing mode**	[10]

- In the direct addressing mode, a 16-bit memory address (offset) directly specified in the instruction as a part of it.
- **Example:** MOV AL, [5000H]
- The above instruction transfers the contents of [5000] location to Ax register

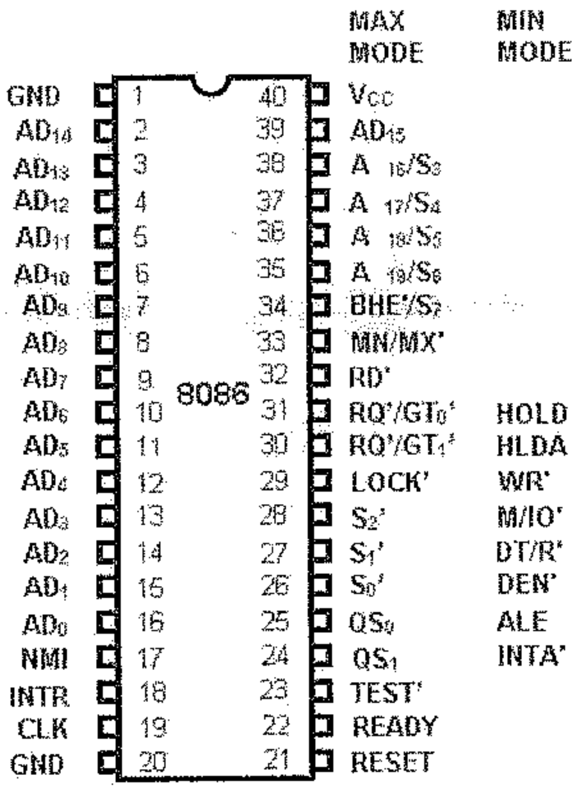
4) **Register InDirect addressing mode**
- Sometimes, the address of the memory location which contains data or operands is determined in an indirect way, using the offset registers.
- The mode of addressing is known as register indirect mode.
- In this addressing mode, the offset address of data is in either BX or SI or DI
- Example: MOV SI, 0009H
- MOV AL, [SI]
- In above example contents of memory location specified by SI is moved to AL that is AL = 60H

5) **Index addressing mode**
- In this addressing mode, the memory address is formed by adding the contents of Index register (SI or DI) and 8-bit/16-bit displacement within the instruction.
- **Example:** MOV AX, [SI+8]
- In above example SI is the index register and 8 is the displacement from SI.
- The contents of location [SI + 8] are copied to AX register.

6) **Based addressing mode**
- In this addressing mode, the memory address is the sum of the base register (BX or BP) and 8-bit/16-bit displacement within the instruction.
- **Example:** MOV AX, [BP+2]
- In above example BP is the base register and 2 is the displacement from BP.
- The contents of location [BP + 2] are copied to AX register.

7) **Based and Index addressing mode**
- In this addressing mode, the contents of the base register (BX or BP) and the contents of an Index register (SI or DI). Are added to calculate the address of the operand.
- **Example:** MOV AX, [BP+DI]
- BX= 0100 and DI=0009 BX+DI = 0109
- The contents of location 0109H are moved to Ax register

8) **Based and Index addressing with displacement**
- In this type of addressing mode the effective address is the sum of index register, base register and displacement.
- Offset= [BX] + [SI] + displacement.
- **Example:**
- MOV AX, [BX+SI+05]

	OR **Explain different pins used in 8086 micro processor with neat diagrams.**  MAX MODE MIN MODE GND 1 AD14 2 AD13 3 AD12 4 AD11 5 AD10 6 AD9 7 AD8 8 AD7 9 AD6 10 AD5 11 AD4 12 AD3 13 AD2 14 AD1 15 AD0 16 NMI 17 INTR 18 CLK 19 GND 20 8086 40 Vcc 39 AD15 38 A16/S3 37 A17/S4 36 A18/S5 35 A19/S6 34 BHE'/S7 33 MN/MX' 32 RD' 31 RQ'/GT0' HOLD 30 RQ'/GT1' HLDA 29 LOCK' WR' 28 S2' M/IO' 27 S1' DT/R' 26 S0' DEN' 25 QS0 ALE 24 QS1 INTA' 23 TEST' 22 READY 21 RESET All pins description	

Q3 (b)	**Construct an ALP to transfer a block of data from one segment to another using string instructions.**
	DATA SEGMENT
	STRING1 DB 01H,02H,03H,04H,05H ;
	STRING2 DB 5 DUP(0)
	DATA ENDS
	
	CODE SEGMENT
	ASSUME CS:CODE,DS:DATA
	START: MOV AX,DATA
	    MOV DS,AX
	    MOV ES,AX
	    LEA SI,STRING1
	    LEA DI,STRING2
	    MOV CX,05H
	    CLD
	    REP MOVSB
	    MOV AH,4CH
	    INT 21H
		[5]

```
CODE ENDS
END START
```

Q4 (a) [8]

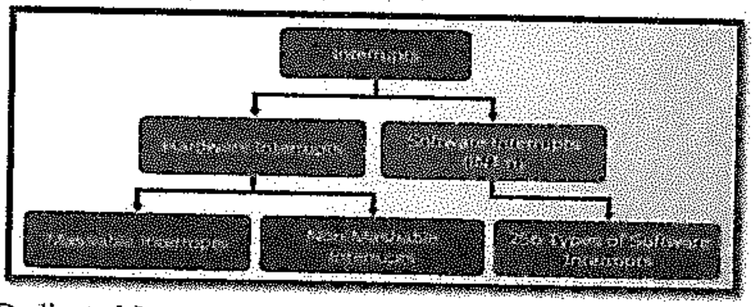
What are interrupts ? list different types of interrupts in 8086 microprocessor. Explain dedicated interrupts.

Interrupts in 8086

- While the CPU is executing a program, an interrupt breaks the normal execution of instructions, diverts its execution to some other program called Interrupt Service Routine (ISR).
- Whenever an interrupt occurs the processor completes the execution of the current instruction and starts the execution of an Interrupt Service Routine (ISR) or Interrupt Handler.
- ISR is a program that tells the processor what to do when the interrupt occurs.
- At the end of the ISR the last instruction should be IRET.
- After the execution of ISR, control returns back to the main routine where it was interrupted.

Types of Interrupts

- In general there are two types of Interrupts:
    - Internal (or) Software Interrupts are generated by a software instruction and operate similarly to a jump or branch instruction.
    - External (or) Hardware Interrupts are caused by an external hardware module.



Dedicated Interrupts

- 1) Divide by Zero Interrupt (Type 0):
    - When the quotient from either a DIV or IDIV instruction is too large to fit in the result register; 8086 will automatically execute type 0 interrupt.
    - To avoid this interrupt, user can check before division that divisor is not zero.
- 2. Single Step Interrupt (Type 1):
    - The type 1 interrupt is single step trap.
    - In the single step mode, system will execute one instruction and wait for further direction from user.
    - This feature is useful for debugging assembly language programs.
- 3. Non Maskable Interrupt (Type 2):
    - As the name suggests, this interrupt cannot be disabled by any software instruction.
    - This interrupt is activated by low to high transition on 8086 NMI input pin.

- In response, 8086 will do a type 2 interrupt.
- 4. Break Point Interrupt (Type 3):
  - The type 3 interrupt is used to implement breakpoint function in the system.
  - When this interrupt occurs a program would execute up to its break point.
  - The type 3 interrupt is produced by execution of the INT 3 instruction.
- 5. Overflow Interrupt (Type 4):
  - This interrupt occurs if the overflow flag (OF) is set in the flag register.
  - The OF flag is set if the signed result of an arithmetic operation on two signed number is too large to be represented in destination register or memory location.
  - Thus this interrupt is used to capture overflow errors.

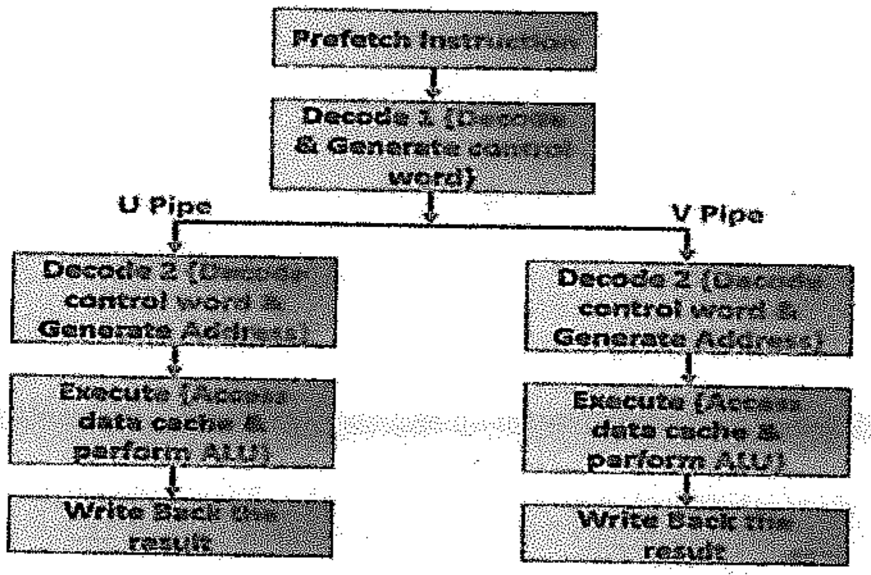
OR

Program for sorting 5 numbers in ascending order – 5M

```
DATA SEGMENT
STRING1 DB 99H,12H,56H,45H,36H
DATA ENDS

CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START: MOV AX,DATA
MOV DS,AX

MOV CH,04H

UP2: MOV CL,04H
LEA SI,STRING1

UP1: MOV AL,[SI]
MOV BL,[SI+1]
CMP AL,BL
JC DOWN
MOV DL,[SI+1]
XCHG [SI],DL
MOV [SI+1],DL

DOWN: INC SI
DEC CL
JNZ UP1
DEC CH
JNZ UP2

CODE ENDS
END START
```

Q4 (b)

Explain the 5 Stage Integer Pipeline in Pentium Processor.


Prefetch Instruction
Decode 1 (Decode & Generate control word)
U Pipe
V Pipe
Decode 2 (Decode control word & Generate Address)
Decode 2 (Decode control word & Generate Address)
Execute (Access data cache & perform ALU)
Execute (Access data cache & perform ALU)
Write Back the result
Write Back the result

- The salient feature of Pentium is that it supports superscalar architecture.
- For execution of multiple instructions concurrently, Pentium microprocessor issues two instructions in parallel to the two independent integer pipelines known as U and V pipelines.
- Each of these two pipelines has 5 stages, as shown in Fig. 5.2.
  - Prefetch (PF)
  - Decode-1 (D1)
  - Decode-2 (D2)
  - Execute (E)
  - Write Back (WB)

1) Prefetch (PF) :

- In the prefetch stage of integer pipeline of the Pentium processor, instructions are fetched from the instruction cache.
- After fetching, the CPU aligns the codes properly.
- As the instructions are of variable lengths, the initial opcode bytes of each instruction must be properly aligned.
- After completion of the prefetch stage, the decode stages D1 and D2 will be executed.

2) Decode-1 (D1) :

- In the decode-1 (D1) pipeline stage, the CPU decodes the instruction and generates a control word.
- In this stage the processor decodes two instructions in parallel and checks their dependency.
- If they are not dependent then they are given to U and V pipes and generates the control word.

3) Decode-2 (D2):

- In decode-2(D2) stage the control word from D1 stage is decoded to complete the instruction decoding.
- In D2 stage the addresses for operands that reside in memory are calculated.

[7]

4) Execute (E):

- The execution stage is used for both ALU operations and data cache access.
- The data cache is used for data operands and ALU performs arithmetic logic computations.

5) Write Back (WB):

- The final stage of the five-stage pipeline is Write Back (WB).
- In the WB stage, the CPU updates the contents of registers and status of the flag register after completion of execution.

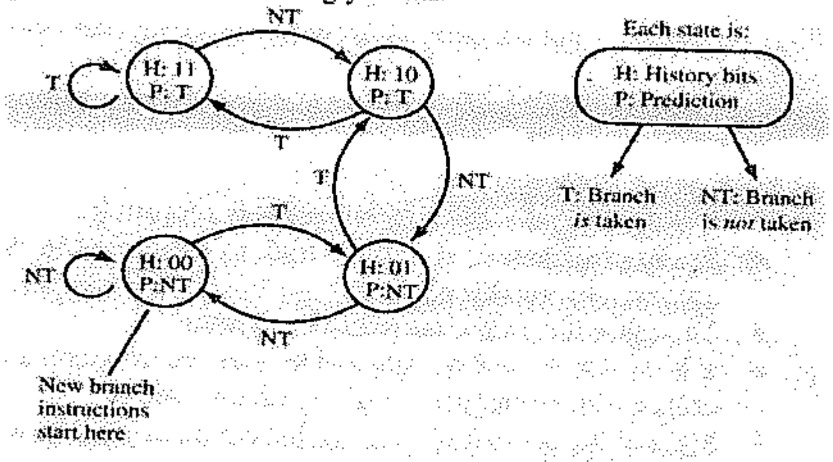
OR

Explain the Branch Prediction Concept in Pentium Processor.

Branch Prediction

- The Pentium microprocessor utilizes branch prediction logic to decrease the time needed for a branch caused by internal delays.
- Branch prediction is a technique used in CPU design that attempts to guess the outcome of a conditional operation and prepare for the most likely result.
- The gain produced by Pipelining can be reduced by the presence of program transfer instructions e.g. JMP, CALL, RET etc.
- They change the sequence causing all the instructions that entered the pipeline after program transfer instructions as invalid.
- Thus no work is done as the pipeline stages are reloaded.
- To avoid this problem, Pentium uses a scheme called Dynamic Branch Prediction.
- In this scheme, a prediction is made for the branch instruction currently in the pipeline.
- The prediction will either be taken or not taken.
- If the prediction is true then the pipeline will not be flushed and no clock cycles will be lost.
- If the prediction is false then the pipeline is flushed and starts over with the current instruction.
- It is implemented using 4 way set associated cache with 256 entries.
- This is called **Branch Target Buffer (BTB).**
- The BTB is a special cache that stores the instruction and target addresses of any branch instructions that have been encountered.
- Along with the addresses for each instruction, the BTB also stores two history bits that indicate the execution history of the last two branch instructions.
- The history bits are initially set to 11 when a new target address is placed in to BTB.
- When the corresponding branch instruction is encountered, the history bits are updated.
- The repeated failures to take a branch cause the history bits to become 00 and the prediction to become not taken.
- Now when the first time that a branch instruction enters the pipeline, the BTB uses its source memory to perform a lookup in the cache.
- Since the instruction was never seen before, it is BTB miss. It predicts that the branch will not be taken even though it is unconditional jump instruction.

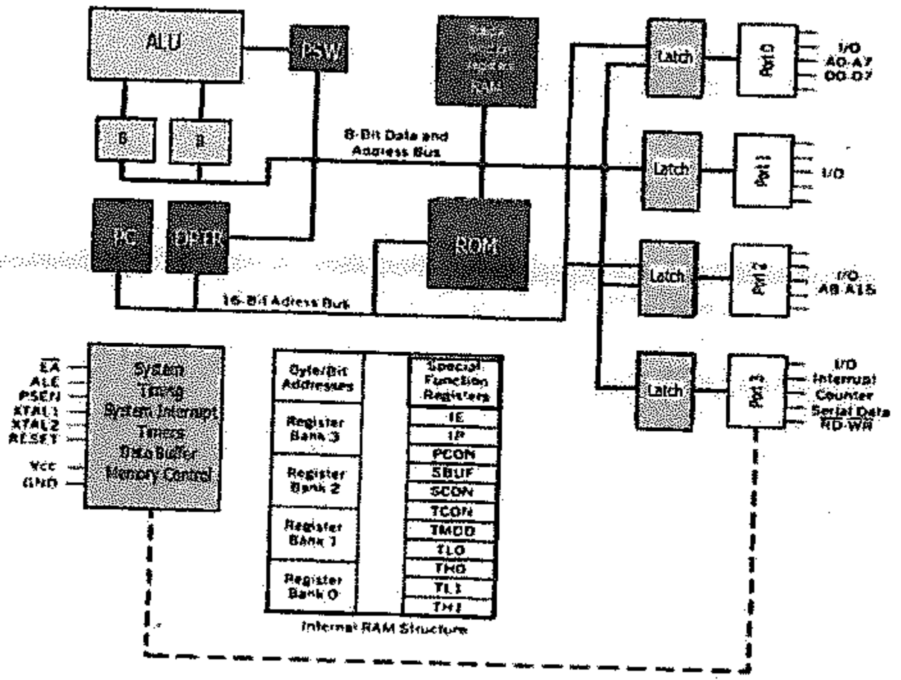
- When the instruction reaches the EU(execution unit), the branch will either be taken or not taken.
- If taken, the next instruction to be executed will be fetched from the branch target address. If not taken, there will be a sequential fetch of instructions.
- When a branch is taken for the first time, the execution unit provides feedback to the branch prediction. The branch target address is sent back which is recorded in BTB.
- A directory entry is made containing the source memory address and history bit is set as strongly taken.



Q5 (a)

Sketch the Architectural Block diagram of 8051 Microcontroller and Explain different components.

[10]



1. Oscillator and clock generator:
   - All operations in a microcontroller are synchronized by the help of an oscillator clock.

- The oscillator clock generates the clock pulses by which all internal operations are synchronized.
- A resonant network connected through pins XTAL1 and XTAL2 forms up an oscillator.

2. ALU:

- It is 8 bit unit.
- It performs arithmetic and logical operations like addition, subtraction, multiplication, division, increment and decrement, AND, OR and EX-OR etc.
- It manipulates 8 bit and 16 bit data.
- 8051 micro controller contains 34 general purpose registers or working registers.
- Two of them are called math registers A & B and 32 are bank of registers.

a. Accumulator(A-reg):

- It is 8 bit register and it is bit and byte accessible.
- Result of arithmetic & logic operations performed by ALU is accumulated by this register.
- Therefore it is called accumulator register.
- It is used to store 8 bit data and to hold one of operand of ALU units during arithmetical and logical operations.
- Most of the instructions are carried out on accumulator data.

b. B-register:

- It is special 8 bit math register.
- It is bit and byte accessible.
- It is used in conjunction with A register as Input operand for ALU.
- It is dedicated for Multiplication and Division.
- It is used as general purpose register to store 8 bit data.

c. PSW:

- It is the 8-bit register but only 6-bits are used by 8051
- It is bit and byte accessible.
- It has 4 conditional flags or math flags which sets or resets according to condition of result.
- It has 3 control flags, by setting or resetting bit required operation or function can be achieved.

3.Program counter(PC):

- 8051 has a 16-bit program counter .
- The program counter always points to the address of the next instruction to be executed.
- After execution of one instruction the program counter is incremented to point to the address of the next instruction to be executed.
- It is the contents of the PC that are placed on the address bus to find and fetch the desired instruction.
- Since the PC is 16-bit width ,8051 can access program addresses from 0000H to FFFFH, a total of 64kB of code.

4. Data pointer register(DTPR):

- It is a 16-bit register.
- DPTR, as the name suggests, is used to point to data.

- It is used by a number of commands which allow the 8051 to access external memory.
- It is typically used by the programmer to transfer data from External RAM.
- When the 8051 accesses external memory it will access external memory at the address indicated by DPTR.
- This DPTR can also be used as two 8-registers DPH and DPL.

5. Stack pointer(SP):

- RAM locations from 08H to 1FH can be used as stack.
- Stack is used to store the data temporarily.
- Stack is last in first out (LIFO)
- Stack pointer (SP) is a 8 bit register
- It indicates current RAM address available for stack or it points the top of stack.
- Initially by default at 07H because first location of stack is 08H.
- This 8-bit register is incremented before the data is stored onto the stack using PUSH or CALL instructions.

6. Input / output Ports

- There are four input output ports available P0, P1, P2, P3.
- Each port is 8 bit wide and has special function register P0, P1, P2, P3 .
- These registers are bit addressable means each bit can be set or reset by the Bit instructions (SETB for high, CLR for low) independently.
- The data at any port which is transmitting or receiving is in these registers.
- The port 0 can perform dual works.
- It is also used as Lower order address bus (A0 to A7) multiplexed with 8 bit data bus. P0.0 to P0.7 is AD0 to AD7 respectively.
- The address bus and data bus is demultiplexed by the ALE signal and latch.
- Port 1 is a true I/O port as it doesn't have any alternative functions as in P0, but this port can be configured as general I/O only.
- Port 2 can be used as I/O port as well as higher order address bus A8 to A15.
- Port 3 also have dual functions it can be worked as I/O as well as each pin of P3 has specific function.

Q5 (b)

List different instruction Sets of 8051 Microcontroller. Illustrate different Arithmetic instructions of 8051 Microcontroller with Example

Instruction Set of 8051

- The instructions of 8051 can be broadly classified under the following headings.
- 1. Arithmetic Instructions
- 2. Data transfer Instructions
- 3. Logical Instructions
- 4. Program Branching Instructions
- 5. Bit Manipulation Instructions / Boolean Variable Manipulation Instructions

[5]

- **1. Arithmetic Instructions:** Arithmetic instructions perform several basic arithmetic operations such as addition, subtraction, division, multiplication etc.
- After execution, the result is stored in the first operand.

Addition

- In this group, we have instructions to

i. Add the contents of A with immediate data with or without carry.

- i. ADD A, #45H
- ii. ADDC A, #B4H

ii. Add the contents of A with register Rn with or without carry.

- i. ADD A, R5
- ii. ADDC A, R2

iii. Add the contents of A with contents of memory with or without carry using direct and indirect addressing

- i. ADD A, 51H
- ii. ADDC A, 75H
- iii. ADD A, @R1
- iv. ADDC A, @R0

Subtraction

- In this group, we have instructions to

i. Subtract the contents of A with immediate data with or without carry.

- i. SUBBA, #45H
- ii. SUBB A, #B4H

ii. Subtract the contents of A with register Rn with or without carry.

- i. SUBB A, R5
- ii. SUBB A, R2

iii. Subtract the contents of A with contents of memory with or without carry using direct and indirect addressing

- i. SUBB A, 51H
- ii. SUBB A, 75H
- iii. SUBB A, @R1
- iv. SUBB A, @R0

- **Multiplication**
- **MUL AB.**
- This instruction multiplies two 8 bit unsigned numbers which are stored in A and B register.
- After multiplication the lower byte of the result will be stored in accumulator and higher byte of result will be stored in B register.
- Eg. MOV A,#03H ;*[A]=03H*
- MOV B,#02H ;*[B]=02H*
- MUL AB ;*[A] x [B] = 03 x 05 = 0006*
  ;*[A]=06H, [B]=00H*
- **Division**
- **DIV AB.**
- This instruction divides the 8 bit unsigned number which is stored in A by the 8 bit unsigned number which is stored in B register.
- After division the Quotient will be stored in accumulator and remainder will be stored in B register.
- Eg. MOV A,#05H ;*[A]=05H*

	• MOV B,#02H ;[B]=02H • DIV AB ; Q = 02 and R = 01 ;[A] = 02H, [B]=01H	