MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION
(Autonomous)
(ISO/IEC - 27001 - 2005 Certified)

WINTER – 2016 EXAMINATION
Model Answer **Subject Code: 17627**

Important Instructions to examiners:
1) The answers should be examined by key words and not as word-to-word as given in the model answer scheme.
2) The model answer and the answer written by candidate may vary but the examiner may try to assess the understanding level of the candidate.
3) The language errors such as grammatical, spelling errors should not be given more Importance (Not applicable for subject English and Communication Skills).
4) While assessing figures, examiner may give credit for principal components indicated in the figure. The figures drawn by candidate and model answer may vary. The examiner may give credit for any equivalent figure drawn.
5) Credits may be given step wise for numerical problems. In some cases, the assumed constant values may vary and there may be some difference in the candidate's answers and model answer.
6) In case of some questions credit may be given by judgement on part of examiner of relevant answer based on candidate's understanding.
7) For programming language papers, credit may be given to any other program based on equivalent concept.

Q.No.	Sub Q.N.	Answer	Marking Scheme
1.		**Answer any FIVE of the following:**	**20**
	a)	**State the salient features of 80386.**	***4M***
	Ans.	**Features of 80386:** 1. It is a 132 PGA (pin grid array) with 32 bits non multiplexed data bus and 32 bits address bus. 2. It works in 3 modes: real, protected and virtual 8086 mode (V-86). 3. It can address total 232 i.e., 4GB physical memory with the help of its 32 bits address lines. 4. The integrated memory management unit in 80386 supports segmentation and paging of memory. 5. It supports the interface of 80387-DX coprocessor IC to perform the complex floating point arithmetic operations. 6. It supports 64TB virtual memory. 7. It has a integrated memory management unit which supports the virtual memory and four levels of protections. 8. It has a on chip clock divider circuitry. 9. It has BIST (built in self test) feature which tests approximately one half of the 80386 processor when RESET and BUSY are active. 10. It has breakpoint registers to provide the breakpoint traps on code (instructions) execution or data access.	***Any eight Features of 80386 processor : ½M each***

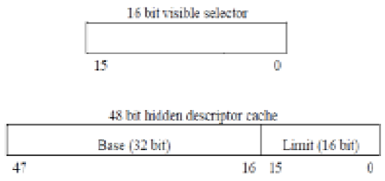
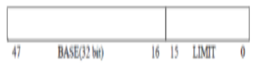
11. It supports instruction pipelining with the help of 16 bytes instruction pre fetch queue.
12. It has 8,32 bit General Purpose bits registers to store the data and address at the time of programming.
13. It has 8 debug registers DR0-DR7 for hardware debugging and control.
14. It has a 32 bit E flag register.
15. It supports the dynamic bus sizing by which the 80386 can be interfaced to 16 bits devices effectively. And also supports the 8bits, 16 bits and 32 bits operands.
16. It operates on 20 MHz and 33 MHz frequency.

b) **Distinguish between LDTR and GDTR.** ***4M***

Ans.

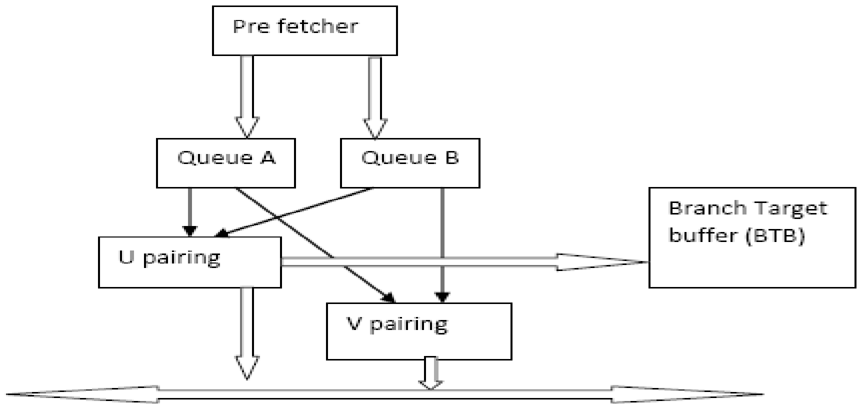
Sr No.	LDTR (LOCAL DESCRIPTOR TABLE REGISTER	GDTR (GLOBAL DESCRIPTOR TABLE REGISTER)
1	The Local Descriptor Table Register (LDTR) is a dedicated 48-bit register that contains, at any given moment, the base and size of the local descriptor table (LDT) associated with the currently executing task. Unlike GDTR, the LDTR register contains both a "visible" and a "hidden" component. Only the visible component is accessible, while the hidden component remains truly inaccessible to application programs.	The Global Descriptor Table Register (GDTR) is a dedicated 48-bit (6 byte) register used to record the base and size of a system's global descriptor table (GDT). Thus, two of these bytes define the size of the GDT, and four bytes define its base address in physical memory. LIMIT is the size of the GDT, and BASE is the starting address. LIMIT is 1 less than the length of the table, then the GDT is 16 bytes long.
2	The visible component of the LDTR is a 16- bit "selector"	There is no visible component of GDTR.
3	The dedicated, protected instructions LLDT and SLDT are reserved for loading and storing, respectively, the visible selector component of	To load the GDTR , LGDT instruction is used.

Any 4 differences between LDTR and GDTR 1M each

	the LDTR register.	
4	Structure of LDTR :  16 bit visible selector 15 0 48 bit hidden descriptor cache Base (32 bit) Limit (16 bit) 47 16 15 0	Structure of GDTR : 

c) Ans.

Explain branch prediction in Pentium. *4M*



Correct Diagram : 2M

Branch Prediction Logic:-

The Pentium processor includes branch prediction logic to avoid pipeline stalls, if correctly, predict whether or not branch will be taken when branch instruction is executed if branch prediction is not correct recycle penalty is applicable to u pipeline & 4 cycle penalty if branch is related to v pipeline.

The prediction mechanism is implemented using 4 way set associative cache with 256 entries referred as branch target buffer. Whenever branch is taken CPU enters the branch instruction address & the destination address in BTB. When an instruction is decoded CPU searches the BTB to determine presence of entry. If its present CPU uses precious history to decide to take the branch.

Relevant explanation 2M

d) Ans.

Explain the RISC processor. *4M*

RISC, or Reduced Instruction Set Computer is a type of microprocessor architecture that utilizes a small, highly-optimized set of instructions, rather than a more specialized set of instructions often found in other types of architectures.

		1. **Simple instruction set**: in a RISC machine, the instruction set contains simple basic instructions, from which more complex instructions can be composed. These instructions with less latency are preferred. 2. **Same length instructions**: each instruction is of same length, so that it may be fetched in a single operation. The traditional microprocessors from intel or Motorola support variable length instructions. 3. **Single machine cycle instruction**: Most instructions complete in one machine cycle, which allows the processor to handle several instructions at the same time. RISC processors have unity CPI (clock per instruction), which is due to optimization of each instruction on the CPU and massive pipelining embedded in a RISC processor. 4. **Pipelining**: usually massive pipelining is embedded in a RISC processor. The pipelining is key to speed up RISC machines. 5. **Very few addressing modes and formats**: unlike the CISC processors, where the numbers of addressing modes are very high. In RISC processors the addressing modes are much less and it supports few formats. 6. **Large number of registers**: the RISC design philosophy generally incorporates a larger number of registers to prevent in large amounts of interactions with memory. 7. **Micro-coding is not required**: Unlike in CISC machines, in RISC architecture, instruction micro-coding is not required. This is because of the availability of a set of simple instructions and simple instructions may be easily built into the hardware. 8. **Load and Store architecture:** the RISC architecture is primarily a Load and Store architecture, implying that all the memory accesses takes place using Load and Store type operations.	***Description/ features of RISC 4M***
	e) Ans.	**State the priority interrupts of 80286.** ***(Note: Any relevant description of interrupts shall be considered).*** If more than one interrupt signals occur simultaneously, they are processed according to their priorities as shown below: **Order** — **Interrupt** 1 — Interrupt exception 2 — Single step 3 — NMI 4 — Processor extension segment overrun 5 — INTR 6 — INT instruction	***4M*** ***Correct Priority interrupts 4M***

	f) Ans.	**Describe any four DOS interrupts.** **INT21** **1) 3CH : to create file** Registers to be used before calling the function using INT 21H: CX=File Attribute DS: DX - full file path (zero terminated) – an ASCIIZ String file descriptor; a start variable in data segment loaded to DX *Syntax:* mov ah,3Ch; function 3Ch - create a file int 21h ; transfer to DOS **2) 3DH: to open file** This function opens the indicated file Registers to be used before calling the function using INT 21H: DS: DX - an ASCIIZ String file descriptor AL=Access Code and sharing modes are as follows 00H- Open for reading mode 01H- open for writing mode 02H – open for read/write mode *Syntax*: mov ah,3Dh; function 3Dh - open the file int 21h; transfer to DOS **3) 3EH: to close the file** This function closes the indicated file Registers to be used before calling the function using INT 21H : BX = file handle *Syntax*: mov ah, 3Eh; function 3Eh - close a file int 21h; transfer to DOS **4) 3FH: to read the file** This function reads up to CX bytes from the Indicated file into the specified memory buffer. On successful return, the AX Register contains the number of bytes actually read. Registers to be used before calling the function using INT 21H: BX = file handle CX = number of bytes to read DS:DX -> buffer for data *Syntax:* mov ah,3Fh; function 3Fh – read the file int 21h; transfer to DOS **5) 40H: to write to the file** This function writes the specified number of bytes from a buffer	***4M*** ***Description of any four DOS interrupts 1M each***

to a file or device.
Registers to be used before calling the function using INT 21H:
BX = file handle
CX = number of bytes to write
DS:DX -> data to write
Syntax: mov ah,40h; function 40h - write to file
int 21h; transfer to DOS

6) 41H: to delete the file
This function deletes the specified file
Registers to be used before calling the function using INT 21H:
ASCIIZ filename DS: DX - zero terminated full paths.
Syntax: mov ah, 41h; delete file int 21h; transfer to DOS

7) 56H: to rename the file
This functions renames the given file with new name specified by ES: DI
Registers to be used before calling the function using INT 21H :
DS: DX address of ASCIIZ filename of existing file ES : DI – ASCIZ new filename
Syntax: mov ah, 56h; delete file int 21h; transfer to DOS

8) 43H: Set/Get file attribute
This function gets or sets the file attributes
Registers to be used before calling the function using INT 21H:
AL = 00H to get attributes 01H to set attributes CX = file attributes, if AL=01H. Bits can be combined DS: DX = segment: offset of ASCIIZ pathname
Syntax: mov ah, 43h; set/get file attributes int 21h; transfer to DOS

9) 57H: Set/Get file time & date
This function gets or sets the file date and time.
Registers to be used before calling the function using INT 21H:
AL = 00h 0r 01H (0 - get 1 - set)
BX = file handle
DS: DX = segment: offset of ASCIIZ pathname
Syntax: mov ah, 57h; set/get file date and time int 21h; transfer to DOS

INT 26H

INT26H Absolute Disk Write
On entry: AL Drive number (0=A, 1=B)
CX Number of sectors to write
DX Starting sector number
DS:DX Address of sectors to write

Returns: AX Error code (if CF is set; see below)
Flags DOS leaves the flags on the stack
This interrupt reads one or more sectors from a disk drive, and is comparable to the service provided by the ROM BIOS in Interrupt 13h.

INT 25h Absolute Disk Read
eads one or more sectors on a specified logical disk.

On entry: AL Drive number (0=A, 1=B)
CX Number of sectors to read
DX Starting sector number
DS:DX Buffer to store sector read

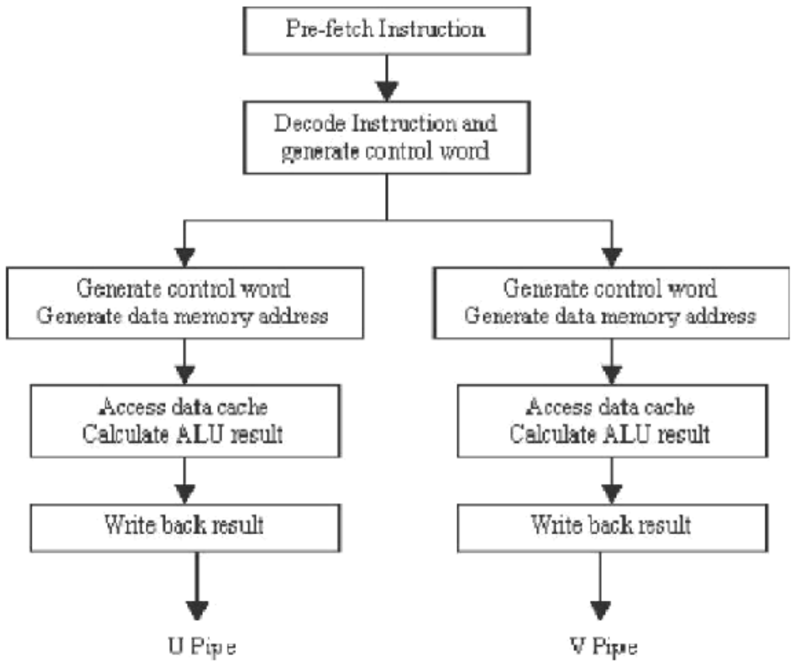
Returns: AX Error code (if CF is set; see below)
Flags DOS leaves the flags on the stack

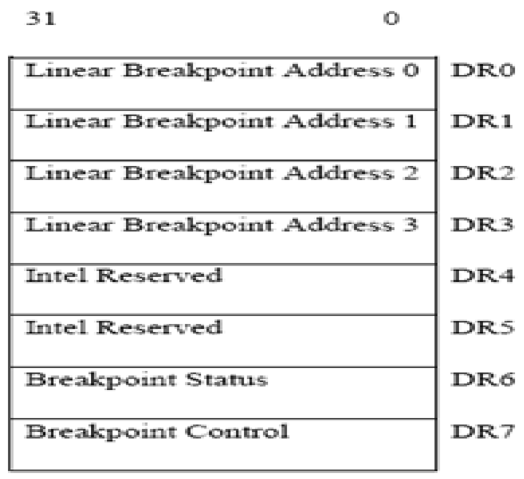
g) Ans.

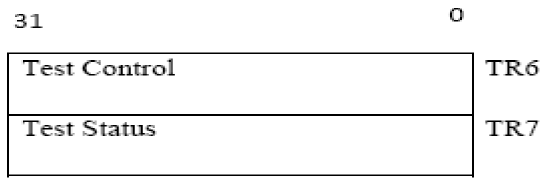
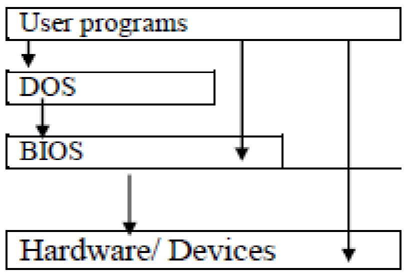
State any four differences between .COM and .EXE program. ***4M***

Sr. No.	**.COM**	**.EXE**
1	.COM file does not contain any header	.EXE file contains header
2	.COM file cannot contain relocation items.	.EXE file may contain relocation items.
3	Maximum size is 64k minus 256 bytes. For PSP and 2 bytes for stack	No limit on size; Can be of any size
4	Entry point is PSP:0100	Entry point is defined by END directive.
5	Stack size is 64K minus 256 bytes for PSP and size of executable data and code.	Stack size is defined in a program with STACK directive
6	Size of file is exact size of	Size of file is size of program

Any 4 differences between .com and .exe programs 1M each

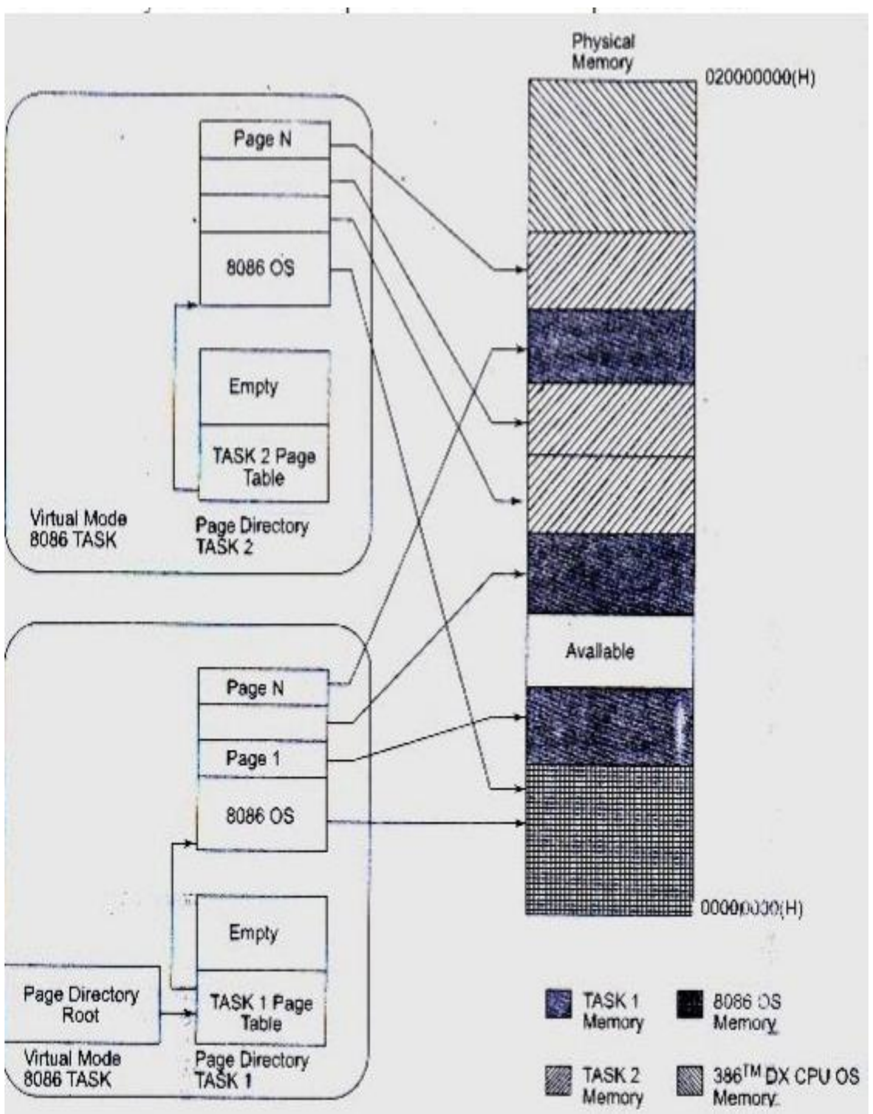
		program. | plus header (Multiple of 256 bytes)	
2.		**Attempt any FOUR of the following:**	16
	a)	**Explain the super scalar execution of pentium processor.**	*4M*
	Ans.	In case of Pentium hardware becomes enormously complex because in such a processor multiple instructions have to be issued in each cycle to the execution unit. 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.  Pre-fetch Instruction Decode Instruction and generate control word Generate control word Generate data memory address Access data cache Calculate ALU result Write back result U Pipe V Pipe	***Correct Diagram 2M***
		First stage of the pipe-line is Prefetch (PF) stage in which instructions are prefetched from the on chip instruction cache or memory. Because the Pentium processor has separate caches for instructions and data, prefetches no longer conflict with data references for access to the cache. If the requested line is not in the code cache, a memory reference is made. In the PF stage, two independent pairs of line-size (32-byte) prefetch. Buffers operate in conjunction with the branch target buffer. This allows one pre fetch buffer to pre fetch instructions sequentially, while the other pre fetches according to the branch target buffer predictions. The pre fetch buffers alternate their pre fetch paths. **The second pipe-line stage is Decode1 (D1)** in which two parallel	***Descript ion 2M***

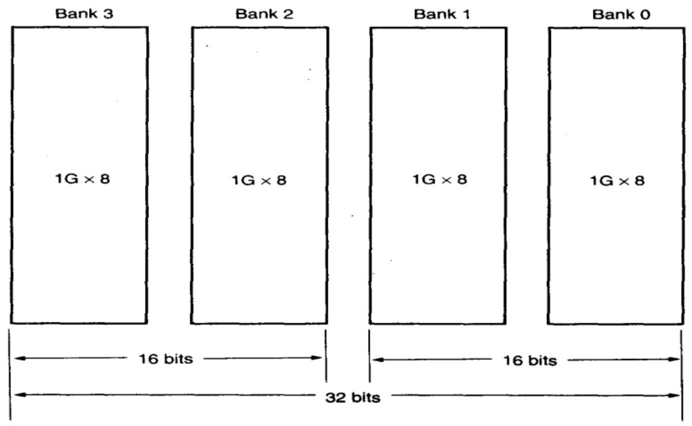
		decoders attempt to decode and issue the next two sequential instructions. The decoders determine whether on e or two instructions can be issued contingent upon the instruction pairing rules described in the section titled "Instruction Pairing Rules." The Pentium processor will decode near conditional jumps (long displacement) in the second op code map (0Fh prefix) in a single clock in either pipe-line. **The D1 stage is followed by third stage i.e. Decode 2 (D2)** in which the address of memory resident operands are calculated. **The fourth stage** **Execute (EX) stage** of the pipe line for both ALU operations and for data cache access; therefore those instructions specifying both an ALU operation and a data cache access will require more than one clock in this stage. In EX all u-pipe instructions and all v-pipe instructions except conditional branches are verified for correct branch prediction. Microcode is designed to utilize both pipe-lines and thus those instructions requiring microcode execute. **The final and fifth stage is** **Write back (WB)** where instructions are enabled to modify processor state and complete execution. In this stage v-pipe conditional branches are verified for correct branch prediction. All the registers and memory locations are updated in this stage.	
	b) Ans.	**Describe debug and test registers of 80386 microprocessor.** **Debug Register:**  31 0 Linear Breakpoint Address 0 DR0 Linear Breakpoint Address 1 DR1 Linear Breakpoint Address 2 DR2 Linear Breakpoint Address 3 DR3 Intel Reserved DR4 Intel Reserved DR5 Breakpoint Status DR6 Breakpoint Control DR7 There are eight debug registers DR0 to DR7 for hardware debugging. The DR0 to DR3 are used to store program controllable breakpoint addresses. The DR4 and DR5 are not used and are reserved by Intel. The DR6 and DR7 are used to hold the breakpoint status and	***4M*** ***Diagram for Debug Registers 1M*** ***Description 1M***

		breakpoint control information respectively. **Test registers of 80386:**  The 80386 has two test registers for page caching. The registers are TR6 – Test Control and TR7 – Test Status. TR6 & TR7 are used for translation look aside buffer (TLB). TLB holds page table address translation to reduce the no. of memory required for page table translation. The test registers are used to perform the confidence checking on the paging. TR6 is the TLB testing command register. By writing into this register, you can either initiate a write directly into the TLB or perform a mock TLB lookup. TR7 is the TLB testing data register. When a program is performing writes, the entry to be stored is contained in this register, along with cache set information.	***Diagram for Test registers 1M*** ***Description 1M***
	c) Ans.	**Explain with neat diagram DOS-BIOS interface.** **DOS-BIOS interface is as shown in the following diagram:**  BIOS contains a set of routines in a ROM to provide the device supports. The BIOS tests and initializes attached devices and provide services that are used for reading to and writing from the devices. One task of DOS is to interface with BIOS when there is a need to access its facilities. When the user program requests a service of DOS, it may transfer the request to BIOS which in turn accesses the requested device. Sometimes, a program makes a direct request to BIOS, especially for keyboard and screen services.	***4M*** ***DOS-BIOS interface diagram : 2M*** ***Description: 2M***

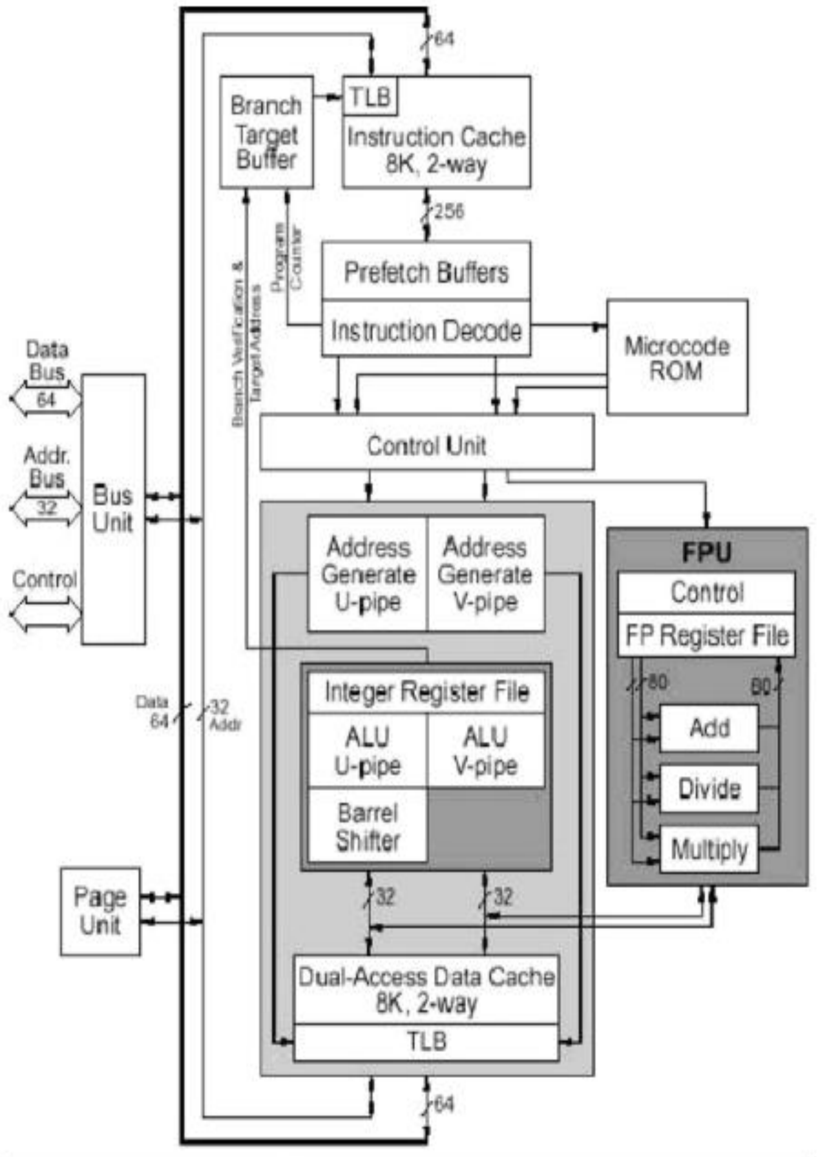
	d) Ans.	**Describe the basic features of RISC processor.** RISC, or Reduced Instruction Set Computer is a type of microprocessor architecture that utilizes a small, highly-optimized set of instructions, rather than a more specialized set of instructions often found in other types of architectures. 1. **Simple instruction set**: in a RISC machine, the instruction set contains simple basic instructions, from which more complex instructions can be composed. These instructions with less latency are preferred. 2. **Same length instructions**: each instruction is of same length, so that it may be fetched in a single operation. The traditional microprocessors from intel or Motorola support variable length instructions. 3. **Single machine cycle instruction**:most instructions complete in one machine cycle, which allows the processor to handle several instructions at the same time. RISC processors have unity CPI(clock per instruction), which is due to optimization of each instruction on the CPU and massive pipelining embedded in a RISC processor. 4. **Pipelining**: usually massive pipelining is embedded in a RISC processor. The pipelining is key to speed up RISC machines. 5. **Very few addressing modes and formats**: unlike the CISC processors, where the number of addressing modes are very high. In RISC processors the addressing modes are much less and it supports few formats. 6. **Large number of registers**: the RISC design philosophy generally incorporates a larger number of registers to prevent in large amounts of interactions with memory. 7. **Micro-coding is not required**: Unlike in CISC machines, in RISC architecture, instruction micro-coding is not required. This is because of the availability of a set of simple instructions and simple instructions may be easily built into the hardware. 8. **Load and Store architecture:** the RISC architecture is primarily a Load and Store architecture, implying that all the memory accesses takes place using Load and Store type operations.	***4M*** ***Description/ features of RISC 4M***
	e) Ans.	**Give important features of sun ultra SPARC.** It contains an integer unit, a FPU and a optional coprocessor. **The 64 bits Ultra SPARC architecture has following features**: 1. It has 14 stages non-stalling pipeline. 2. It has 6 execution units including two for integer, two for floating point, one for load/store and one for address generation units. 3. It has a large number of buffers but only one load/store unit, it	***4M***

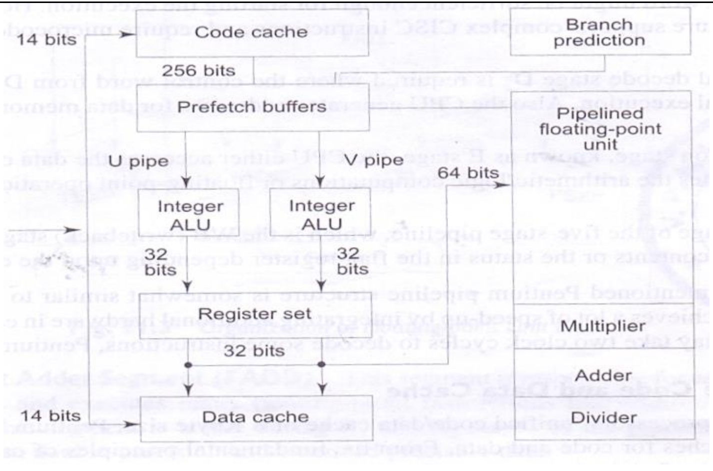
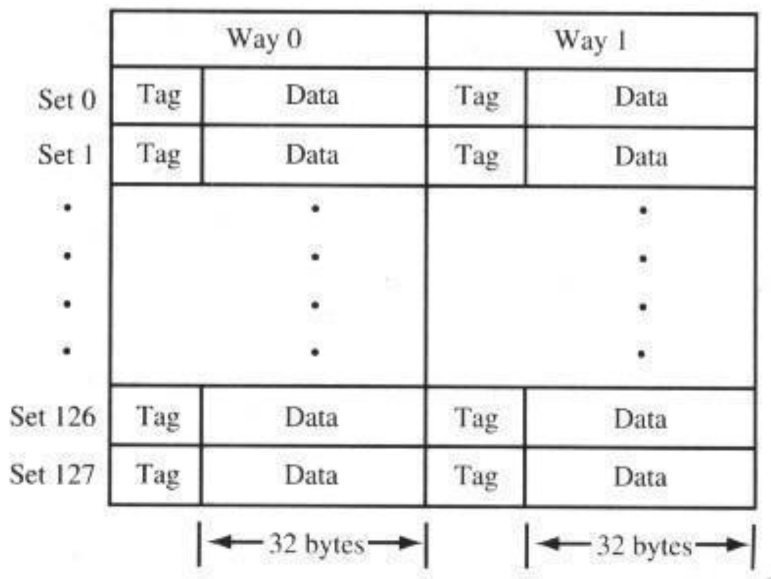
		dispatches them one instruction at a time from the instruction stream. 4. It contains 32KB L1 instruction cache, 64KB L1 data cache, 2KB prefetch cache and 2 KB write cache. It also has 1MB on chip L2 cache. 5. Like Pentium MMX it also contains the instructions to support multimedia. These instructions are helpful for the implementation of image processing codes. 6. One of the major limitations of SPARC system is its low speed compared to most of the modern processors. 7. SPARC stores multi-byte numbers using BIG endian format, i.e. the MSB will be stored at the lowest memory address. 8. It supports a pipelined floating point processor. The FPU has 5 separate functional units for performing the floating point operations. The floating point instructions can be issued per cycle and executed by the FPU unit. The source and data results are stored in 32 register files. Majority of the floating point instructions have a throughput of one cycle and a latency of three cycles. Although the single precision (32 bit) or double precision (64 bit) floating point computations can be performed by hardware, quad precision i.e. 128 bits operation can be performed only in the software.	***Any eight features of sun ultra SPARC processor 1/2M each***
3.	**a)** Ans.	**Attempt any <u>TWO</u> of the following:** **Illustrate with diagram the concept of virtual 8086 environment memory management.** The virtual 8086 mode of operation of 80386, offers an advantage of executing 8086 programs while in protected mode. The address forming mechanism in virtual 8086 mode is exactly identical with that of 8086 real mode. 1. In virtual mode, 8086 can address 1Mbytes of physical memory that may be anywhere in the 4Gbytes address space of the protected mode of 80386. Like 80386 real mode, the addresses in virtual 8086 mode lie within 1Mbytes of memory. In virtual mode, the paging mechanism and protection capabilities are available at the service of the programmers (note the 80386 supports multiprogramming; hence more than one programmer may use the CPU at a time). 2. Paging unit may not be necessarily enable in virtual mode, but may be needed to run the 8086 programs which require more than 1Mbyts of memory for memory management function. In virtual mode, the paging unit allows only 256 pages, each of 4Kbytes size. Each of the pages may be located anywhere in the maximum 4Gbytes physical memory. The virtual mode allows the	**16** ***8M*** ***Explanation: 4M***

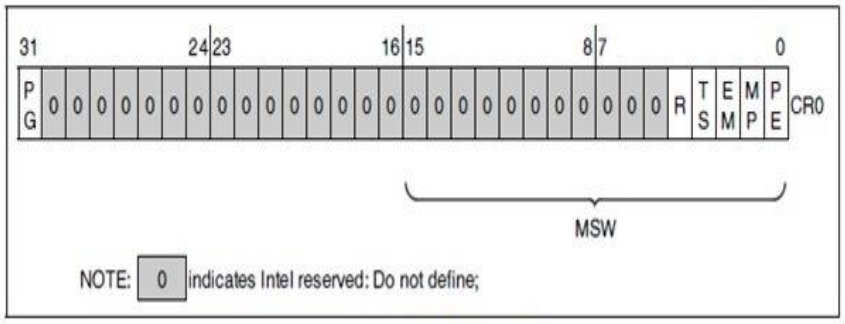
		multiprogramming of 8086 applications. The virtual 8086 mode executes all the programs at privilege level 3.  Physical Memory 020000000(H) 00000000(H) Page N 8086 OS Empty TASK 2 Page Table Virtual Mode 8086 TASK Page Directory TASK 2 Available Page N Page 1 8086 OS Empty Page Directory Root TASK 1 Page Table Virtual Mode 8086 TASK Page Directory TASK 1 TASK 1 Memory 8086 OS Memory TASK 2 Memory 386™ DX CPU OS Memory	***Correct Diagram : 4M***
	b) Ans.	**Explain with the help of neat diagram the memory organization of 80386.** The physical memory system of the 80386 is 4G bytes in size and is addressed as such. If virtual addressing is used, 64T bytes are mapped into the 4G bytes of physical space by the memory management unit and descriptors. (Note that virtual addressing allows a program to be larger than 4G bytes if a method of swapping with a very large hard disk	***8M***

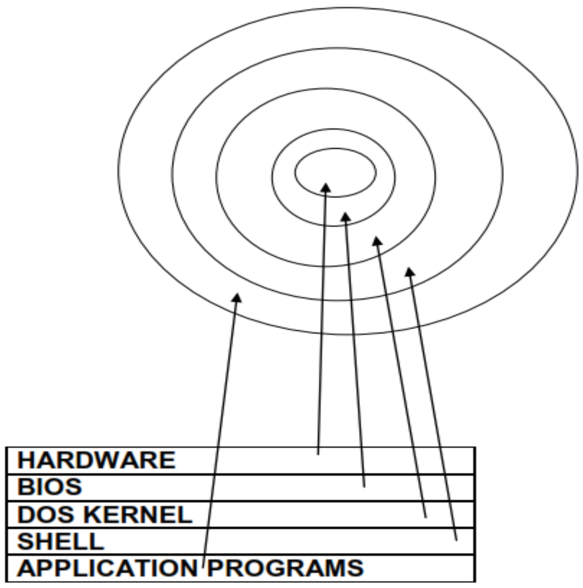
		drive exists.) Figure shows the organization of the 80386 physical memory system.  Bank 3 Bank 2 Bank 1 Bank 0 1G × 8 1G × 8 1G × 8 1G × 8 16 bits 16 bits 32 bits	***Diagram of memory organization of 80386 : 2M***
		The memory is divided into four 8-bit wide memory banks, each containing up to 1G bytes of memory. This 32-bit wide memory organization allows bytes, words, or double words of memory data to access directly. The 80386DX transfers up to a 32-bit wide number in a single memory cycle, whereas the early 8088 requires four cycles to accomplish the same transfer, and the 80386 require two cycles. The data width is important, especially with single-precision floating-point numbers that are 32 bits wide. High-level software normally uses floating-point numbers for data storage, so 32-bit memory locations speed the execution of high level software when it is written to take advantage of this wider memory. Each memory byte is numbered in hexadecimal as they were in prior versions of the family. The difference is that the 80386DX uses a 32-bit wide memory address, with memory bytes numbered from location 00000000H—FFFFFFFFH. The two memory banks in the 80386 system are accessed via $\overline{BLE}$ and $\overline{BHE}$. In the 80386DX, the memory banks are accessed via four bank enable signals. This arrangement allows a single byte to be accessed when one bank enable signal is activated by the microprocessor. It also allows a word to be addressed when two bank enable signals are activated. In most cases, a word is addressed in bank 0 and l or in bank 2 and 3. Memory location 00000000H is in bank 0, location 0000000IH is in bank 1, location 00000002H is in bank 2, and location 00000003H is in bank 3. The 80386DX does not contain address connections A0 and Al because these have been encoded as the bank enable signals.	***Description : 6M***

	c) Ans.	**Explain the hybrid architecture (i.e. RISC and CISC) of processors.** State of the art processor technology has changed significantly since RISC chips were first introduced in the early '80s. Because a number of advancements (including the ones described on this page) are used by both RISC and CISC processors, the lines between the two architectures have begun to blur. The two architectures almost seem to have adopted the strategies of the other. Because processor speeds have increased, CISC chips are now able to execute more than one instruction within a single clock. This also allows CISC chips to make use of pipelining. With other technological improvements, it is now possible to fit many more transistors on a single chip. This gives RISC processors enough space to incorporate more complicated, CISC-like commands. RISC chips also make use of more complicated hardware, making use of extra function units for superscalar execution. The two styles have become so similar that distinguishing between them is no longer relevant. However, it should be noted that RISC chips still retain some important traits. RISC chips strictly utilize uniform, single-cycle instructions. They also retain the register-to-register, load/store architecture. And despite their extended instruction sets, RISC chips still have a large number of general purpose registers. The most popular hybrid architecture has processors are the Pentium and AMD Athlon family processors which are compatible with software written for their CISC predecessors. Modern RISC processors have become CISC like by supporting more functions and support more instructions than old CISC design. Using the CISC architecture as more instructions , some applications may be run much faster such as multimedia applications, such as telecommunications encoding/ decoding , image conversions and video processing.	***8M*** ***Relevant explanation of hybrid architecture / RISC /CISC 8M***
4.	 **a)** Ans.	**Attempt any FOUR of the following:** **Draw the architecture of Pentium processor.**	**16** ***4M***

		 64 Branch Target Buffer TLB Instruction Cache 8K, 2-way 256 Prefetch Buffers Instruction Decode Program Counter Branch Verification & Target Address Microcode ROM Data Bus 64 Addr. Bus 32 Control Bus Unit Control Unit Address Generate U-pipe Address Generate V-pipe FPU Control FP Register File 80 80 Add Divide Multiply Integer Register File ALU U-pipe ALU V-pipe Barrel Shifter Data 64 32 Addr 32 32 Page Unit Dual-Access Data Cache 8K, 2-way TLB 64 OR	***Labeled architecture of Pentium processor :4M***

		 14 bits Code cache Branch prediction 256 bits Prefetch buffers Pipelined floating-point unit U pipe V pipe 64 bits Integer ALU Integer ALU 32 bits 32 bits Register set 32 bits Multiplier Adder 14 bits Data cache Divider	
	b) Ans.	**Explain the concept of separate code and data cache memory in pentium processors.** **Separate 8K B instruction and Data Cache:** The following figure shows the organization of instruction and data cache.  Way 0 Way 1 Set 0 Set 1 Set 126 Set 127 Tag Data 32 bytes 32 bytes The Pentium processor has 2 separate 8KB data and code Caches. But they need more bandwidth than the unified cache. Both the caches have TLB's associated with them. The TLBs are used to convert the linear addresses to the respective physical addresses.	*4M* ***Diagram 2M***

		As the data cache stores only 8KB data and code cache stores only instructions, the look up process speed for Pentium increases. **Advantages of separate instruction and data caches :** 1. Separate code and data cache memories effectively and efficiently executes the branch prediction. 2. Simultaneous cache look up is achieved by Pentium processor due to the separate data and code cache. 3. The separate cache memories raise the system performance i.e. an internal read request is performed more quickly than a bus cycle to memory. 4. They reduce the use of processor's external bus when the same locations are accessed multiple times.	***Explanation of Concept of separate code and data cache in Pentium 2M***
	c) Ans.	**Describe enabling and disabling of paging in 80386.** Enabling and disabling the paging in 80386 is done with the help of CR0 register . the layout of CR0 is as shown in the figure below:  31 24 23 16 15 8 7 0 PG 0 R TS EM MP PE CR0 MSW NOTE: 0 indicates Intel reserved: Do not define; Control Register 0 By setting the PG bit (Most significant Bit or bit 31) of CR0 to 1, paging can be enabled. This bit cannot be set until the processor is in protected mode. When PG is set, the PDBR (Page Directory Base Register – CR3) should already be initialized with a physical address that points to a valid page directory. To enable paging, use MOV instruction to set most significant bit of CR0, *E.g.* MOV CR0, EAX After this bit is set, page translation takes effect on the next instruction. Paging can be disabled by making the PG bit of CR0 as zero.	***4M*** ***Enabling and disabling of paging :diagram :2M*** ***Explanation :2M***
	d) Ans.	**Which function is used to "Delete file"? Explain in detail with example.** **Function 41h under DOS interrupt 21h is used to delete a file.** **AH = 41H - "UNLINK" - DELETE FILE** Entry:	***4M*** ***Delete function : 2M***

		• DS:DX -> ASCIZ filename (no wildcards, but see notes) • CL = attribute mask for deletion (server call only, see notes) Return: • CF clear if successful, AX destroyed (DOS 3.3) AL seems to be drive of deleted file • CF set on error AX = error code (02h,03h,05h) *Example program to delete a file abc.txt from c:\tasm* Code segment Movax,data Mov ds, ax Mov ah,41h Lea dx, filename Int 21h Mov ah, 4ch Int 21h Code ends Data segment Filename db “c:\tasm\abc.txt”,00 Data ends End	***syntax, usages and example : 2M***
	e) Ans.	**Explain the structure of MS-DOS with respect to its layers.** **MS DOS structure is as shown below :**  HARDWARE BIOS DOS KERNEL SHELL APPLICATION PROGRAMS	***4M*** ***Structure of MS-DOS diagram : 2M***

		MS-DOS is divided in to 3 layers which are: 1. BIOS 2. DOS kernel 3. Command processor (shell) **1.BIOS:** It is provided by the manufacturer. It contains the resident hardware dependent device drivers for the devices like: 1. Console display and keyboard. (CON) 2. Line printer (PRN) 3. Auxiliary devices. (AUX) 4. date and time (CLOCK) 5. Boot disk device (Block device) Kernel communicates with these device drivers through the I/O request packets and then the drivers translate these requests into proper commands for various controllers. BIOS is read into RAM at the time of initialization as a part of a file named IO.SYS. This file is marked with the special attribute as *hidden* and *system* **2. DOS Kernel:** Functions : 1. file and record management 2. memory management 3. character device I/O 4. swapping of programs 5. access to real time clock The DOS kernel is read into memory during system initialization from the MSDOS.SYS file on the boot disk. (The file is called IBMDOS.COM in PC-DOS.) This file is marked with the attributes hidden and system. **3. Command processor:** Functions: 1. User interface with OS. 2. Parsing and carrying out user commands. 3. Loading and execution of other programs from disk or other memory. The default shell that is provided with MS-DOS is found in a file called COMMAND.COM. Although COMMAND.COM prompts and responses constitute the ordinary user's complete perception of MS-DOS, it is important to realize that COMMAND.COM is not the operating system, but simply a special class of program running under the control of MS-DOS.	***Explanation : 2M***

	f) Ans.	**Explain floating point exceptions.** **The Pentium provides 6 floating point exceptions:** 1. Invalid operation 2. Divide by zero 3. De-normalized operand 4. Numeric overflow 5. Numeric underflow 6. Inexact result All these exceptions classes have a corresponding flag bit in the FPU status word and a mask bit in FPU control word. The exception summary (ES) flag in the status word of Pentium indicates when any of these exceptions have been detected. It also has a stack fault (SF) flag in the status word which distinguishes between the 2 types of invalid operation exceptions. When the FPU detects a floating point exception, it sets the appropriate flags in the FPU status word, then takes one of the two possible actions : 1. Handles the exceptions automatically, producing a predefined result and allow program to continue its execution without any disturbance. 2.Invokes a software exception handler to handle the exception. **1. Invalid operation** The floating point invalid exception occurs in response to two general types of operations : 1. Stack overflow or underflow 2. Invalid arithmetic operand. When the SF is set to 1, a stack operation has resulted in stack overflow or underflow. When the flag is cleared to 0, an arithmetic instruction has encountered an invalid operation. The FPU explicitly sets the SF flag when it detects a stack overflow or under flow condition, but it does not explicitly clear the flag when it detects an invalid arithmetic operand condition. As a result the state of the SF flag can be 1 following an invalid arithmetic operation exception, if it was not cleared from the last time a stack overflow or under flow condition occurred. **2. Divide by zero:** The FPU reports a floating point zero divide exception, whenever an instruction attempts to divide the operand by 0. The flag ZE for this exception is bit 2 of the FPU status word, and the mask bit ZM is bit2 of the control word. The FDIV, FDIVP, FDIVR, FDIVRP, FIDIV, FIDIVR instructions and the other instructions that perform division internally can report the	*4M* ***Any four floating point exceptions 4M***

		divide by zero exception. **3. De-normalized operand** The FPU signals the de-normal operand exception under the following conditions : 1. If an arithmetic instruction attempts to operate on a denormal operand. 2. If an attempt is made to load the denormal single or double real value into an FPU register. The flag DE for this exception is bit 1 of the FPU status word, and the mask bit (DM) is the 1 of the FPU control word. **4. Numeric overflow** This exception occurs when the rounded result of an arithmetic instruction exceeds the largest allowable finite value that will fit into the real format of the destination operand. **5. Numeric underflow** This exception occurs when the rounded result of an arithmetic instruction is less than the smallest possible normalized, finite value that will fit into the real format of the destination operand. **6. Inexact result** This exception occurs if the result of an operation is not exactly in representable in the destination format.	
5.		**Attempt any FOUR of the following:**	**16**
	a)	**State the functions of the following pins of 80386 μp (microprocess)** **(i)** $\overline{BE_3} - \overline{BE_0}$ **(ii)** BS_{16} **(iii)** $D/\overline{C}$ **(iv)** $\overline{ADS}$	***4M***
	Ans.	**(i) $\overline{BE_3} - \overline{BE_0}$ (Bus/byte enable signal)** The 32-bit Data bus supported by 80386 and the memory system of 80386 can be viewed as a 4-byte wide memory access mechanism. The four byte enable lines, $\overline{BE_3} - \overline{BE_0}$, may be used for enabling these four banks. Using these four enable signal lines, the CPU may transfer 1 byte/2bytes/3bytes or 4bytes of data simultaneously. **(ii)$\overline{BS_{16}}$(bus size 16: active low :input signal)** The bus size-16 input pin allows the interfacing of 16-bit devices with the 32-bit wide 80386 data bus. Successive 16-bit bus cycles may be executed to read a 32-bit data from a peripheral. **(iii)$D/\overline{C}$:Output signal** Data/ Control is a bus cycle definition pin that distinguishes data cycles, either memory or I/O, from control cycles which are interrupt acknowledge, halt and instruction fetching.	***1M per function of each pin***

(iv) $\overline{ADS}$(address strobe signal :active low : output signal)

The address status output pin indicates that the address bus and bus cycle definition (w/R#, D/C#, M/IO#, BE_0#-BE_3) are carrying the respective valid signals. The 80386 does not have any ALE signal and so this signal may be used for latching the address to external latches.

b) **List any four difference between real addressing mode and protected virtual (PVAM) addressing mode of 80286.** *4M*

(Note: Any other relevant description of real addressing and protected virtual addressing mode shall be considered)

Ans.

Real addressing mode	Protected Virtual Addressing Mode
80286 works as fast 8086and prepares 80286 for protected mode	This is a normal addressing mode of 80286.
Features of 80286 like memory management, virtual memory and protection are not used	All the features of 80286 are used
It uses only 20 address lines A_0-A_{19}	It uses all 24 address lines A_0-A_{23}
It accesses only 1MB memory	It can access 16MB physical memory and 1GB virtual memory
Physical address calculation is simple. Four 0's are appended on RHS of segment address and offset value is then added to get physical address	Segment register acts as selector to point to the descriptor. These descriptors provide the 24 bit base address to which offset value is added.

Any four points of difference – 1M each

c) **State any four salient features of Pentium.** *4M*

Ans. **Following are the features of Pentium:**

1) It is based on net burst micro architecture.
2) Superscalar architecture
3) Dynamic branch prediction
4) Pipelined Floating-Point Unit
5) Separate code and data caches
6) 64-bit data bus
7) Address parity
8) Support for Intel MMX technology
9) Dual power supplies—separate VCC2 (core) and VCC3 (I/O)

Any 4 features: 1M each

		voltage inputs 10) Separate 16-Kbyte, 4-way set-associative code and data caches, each with improved fully associative TLBs 11) Pool of four write buffers used by both execution pipelines 12) Enhanced branch prediction algorithm 13) New Fetch pipeline stage between Prefetch and Instruction Decode.	
	d) Ans.	**Explain non-maskable interrupts.** **Non maskable interrupts**: Non maskable interrupts provide a method of servicing very high priority interrupts. NMI is an example of non maskable interrupt. It is an external pin to the microprocessor. A common example of the use of non maskable interrupt (NMI) would be to activate a power failure routine. When a NMI is pulled high it causes an interrupt with an internally supplied vector value of 2. No interrupt acknowledgement cycle is performed by the processor when NMI occurs. While executing NMI, no further NMI is serviced until the next IRET instruction is executed or the processor is reset. If NMI occurs at the time of servicing a NMI, its occurrence will be saved and it will be processed when the servicing of the first will be over. The IF bit is cleared at the beginning of NMI interrupt to inhibit further INTR requests.	***4M*** ***Correct explanation:4M***
	e) Ans.	**Explain pentium pro-processor.** **1. Speculative Execution:** Which means that the CPU should speculate which of the next instructions can be executed earlier. As in our example just now cited, the CPU will not be able to execute the second instruction before the first instruction is executed, since the second instruction requires the value of the register, which is loaded from memory after the first instruction is executed. However, some of the next instructions may be executed earlier since they are independent of the previous instructions. The CPU may speculate this and may execute these next instructions earlier. **2. Out of Turn Execution**: Naturally the consecutive instruction execution in a sequential flow will be hampered and the CPU should be able to execute out of turn instructions. **3. Dual Independent Bus:** Pentium-Pro incorporates dual independent bus architecture to get an enhanced system bandwidth. Pentium-Pro uses two separate and independent buses- one between the CPU and the main memory and the other between the CPU and the cache memory. The CPU can thus access both, the main memory and the cache simultaneously. This obviously yields a high throughput. **4. Multiple Branch Prediction:** The concept of branch prediction in	***4M*** ***Relevant explanation-4M***

		Pentium has been extended to achieve multiple branch prediction in Pentium-Pro. Based on the past history of the branches taken, multiple branch prediction logic enhances the performance of Pentium. The processor uses an associative memory called branch target buffer for implementing this algorithm.	
	f) Ans.	**Explain design issues of RISC processor.** **1. Register Window :** The reduced hardware requirements of RISC processors leave additional space available on the chip for the system designer. RISC CPUs generally use this space to include a large number of registers (> 100 occasionally). The CPU can access data in registers more quickly than data in memory so having more registers makes more data available faster. Having more registers also helps reduce the number of memory references especially when calling and returning from subroutines. The RISC processor may not be able to access all the registers it has at any given time provided that it has many of it. Most RISC CPUs have some global registers which are always accessible. The remaining registers are windowed so that only a subset of the registers is accessible at any specific time. To understand how register windows work, we consider the windowing scheme used by the Sun SPARC processor. The processor can access any of the 32 different registers at a given time. (The instruction formats for SPARC always use 5 bits to select a source/destination register which can take any 32 different values. Of these 32 registers, 8 are global registers that are always accessible. The remaining 24 registers are contained in the register window. The register window overlap. The overlap consists of 8 registers in SPARC CPU. Notice that the organizations of the windows are supposed to be circular and not linear; meaning that the last window overlaps with the first window. Example: the last 8 registers of window 1 are also the first 8 registers of window 2. Similarly, the last 8 registers of window 2 are also the first 8 registers of window 3. The middle 8 registers of window 2 are local; they are not shared with any other window. 2. **Memory speed issue:** Memory speed issues are commonly solved using caches. A cache is a section of fast memory placed between the processor and slower memory. When the processor wants to read a location in main memory, that location is also copied into the cache. Subsequent references to that location can come from the cache, which	***4M*** ***Four design issues: 1M each***

		will return a result much more quickly than the main memory. Caches present one major problem to system designers and programmers, and that is the problem of coherency. When the processor writes a value to memory, the result goes into the cache instead of going directly to main memory. Therefore, special hardware (usually implemented as part of the processor) needs to write the information out to main memory before something else tries to read that location or before re-using that part of the cache for some different information. **3. Instruction Latency issue:** A poorly designed instruction set can cause a pipelined processor to stall frequently. Some of the more common problem areas are: Highly encoded instructions such as those used on CISC machines that require complex decoders. Those should be avoided. Variable-length instructions which require multiple references to memory to fetch in the entire instruction. Instructions which access main memory (instead of registers), since main memory can be slow. Complex instructions which require multiple clocks for execution (many floating-point operations, for example.)Instructions which need to read and write the same register. For example "ADD 5 to register 3" had to read register 3, add 5 to that value, then write 5 back to the same register (which may still be "busy" from the earlier read operation, causing the processor to stall until the register becomes available.) Dependence on single-point resources such as a condition code register. If one instruction sets the conditions in the condition code register and the following instruction tries to read those bits, the second instruction may have to stall until the first instruction's write completes. **4. Dependencies issues:** One problem that RISC programmers face is that the processor can be slowed down by a poor choice of instructions. Since each instruction takes some amount of time to store its result, and several instructions are being handled at the same time, later instructions may have to wait for the results of earlier instructions to be stored. However, a simple rearrangement of the instructions in a program (called Instruction Scheduling) can remove these performance limitations from RISC programs.	
6.	**a)** Ans.	**Attempt any TWO of the following:** **Describe the eight stage pipeling mechanism in floating point unit of Pentium.**	**16** ***8M***

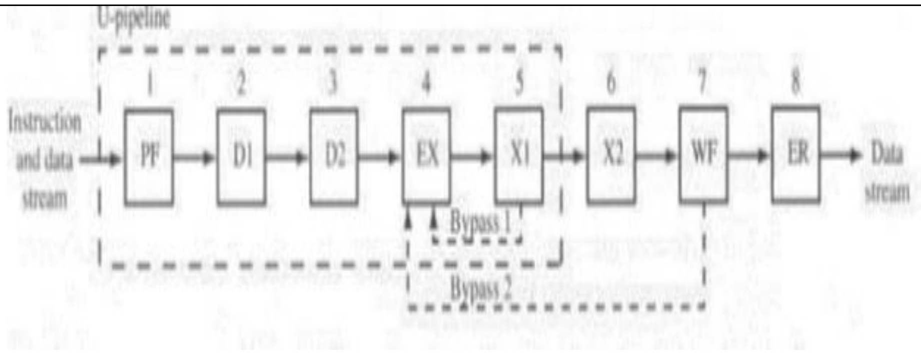


Diagram 4M

The pipelining stages in the floating point unit of Pentium are:

PF	Prefetch
D1	Instruction decode
D2	Address generation
EX	Memory and register read, floating-point data converted into memory format, memory write
X1	Floating-point execute, stage one. Memory data converted into floating-point format, write operand to floating-point register file, bypass 1 (send data back to EX stage)
X2	Floating-point execute stage two
WF	Round floating-point result and write to floating-point register file, bypass 2 (send data back to EX stage)
ER	Error reporting, update status word

Description 4M

Explanation eight stages:

1. Prefetch: In this stage FPU fetches instructions from instruction cache and also aligns the code appropriately.
2. D1 stage: In this FPU decodes the instructions and generate and control word.
3. D2 stage: It is required there control word from D1 stage is again decoded from final execution.
4. Operand fetch stage: In this stage FPU fetches operands either from floating point register file or from data cache.
5. X1 stage: First execution stage
6. X2 stage: Second execution stage
   In these two stages FPU reads the data from data cache and executes the floating point computation. There are eight general purposes floating point registers in FPU.
7. Write back stage (WB): In this stage FPU writes result to the floating

		point register file. 8. Error reporting file: In this stage FPU reports the internal status including errors which may require additional processing for completion of floating point execution.	
	b) Ans.	**Describe the loading sequence of MS-DOS in memory with neat sketch.** When the system is reset or started, the program execution begins at the address 0FFFF0H. The control is transferred to system test code, power on self-test (POST). Then the control is transferred to the ROM bootstrap routine, which reads the bootstrap from the first sector of the system startup disk into memory at some arbitrary address and transfers control to it. The disk bootstrap checks to see if the "boot" disk contains DOS by checking the first sector of the root directory for the file IO.SYS and MSDOS.SYS. If these are not found in the boot disk, the user gets a prompt for changing the disk. If the two files are found, the disk bootstrap reads the files into memory and transfers the control to IO.SYS The IO.SYS file consists of two separate modules. The first is the BIOS, which contains the linked set of resident device drivers for the console, auxiliary port, printer, clock devices and some hardware specific initialization code. Second module consists of system initialization program, which determines the RAM size in the PC. Then it loads the MSDOS.SYS program to its final memory location of the DOS Kernel program. The DOS Kernel initializes its tables and sets up its various work areas. It sets up the various interrupt vectors for the DOS interrupts20H-2FH pointing them to appropriate service routine. It then loads and executes the device drivers. Now it returns the control to system initialization program (SYSINIT). The SYSINIT calls MS-DOS file service to open the CONFIG.SYS file. It contains the list of additional device drivers that the user wants in his system. The required drivers are loaded into the memory, initialized by calls to their INIT modules, and linked into their device driver list. After SYSINIT calls the EXEC function to load the command interpreter (shell). Once the interpreter is loaded, it displays a prompt and waits for the user to enter the command. Top of RAM	*8M* *Description 4M*

		<table><tr><td>ROM Bootstrap routine</td></tr><tr><td>:</td></tr><tr><td>: Transparent part of Command.com</td></tr><tr><td>Transient program area</td></tr><tr><td>Resident part of Command.com</td></tr><tr><td></td></tr><tr><td>File Control Block</td></tr><tr><td>Disk Buffer Cache</td></tr><tr><td>DOS Kernel</td></tr><tr><td>BIOS</td></tr><tr><td></td></tr><tr><td>Interrupt Vector Table (00000H – 00400H)</td></tr></table>	*Diagram 4M*
	c) Ans.	**Draw and explain the internal architecture of 80386.** **The internal architecture of 80386 can be divided into 3 sections such as** 1. Central processing unit (CPU) 2. Memory management unit (MMU) 3. Bus interface unit (BIU) **The Central processing unit** consists of **Execution unit & Instruction unit** Instruction unit has Instruction pre-fetcher and instruction pre-decode unit The Instruction pre-fetcher fetches the 16 instruction bytes ahead of time and stores them into the 16 byte instruction pre-fetch queue(16 byte code).This speeds up the program execution process. The instruction pre-decode unit has the instruction decoder and 3 decoded instruction queue. The instruction decoder decodes 3 instructions ahead of time and stores them in the 3 decoded instruction queue. Execution unit has ALU and control unit. The control unit stores the control signals in the control ROM, which are generated at the time of decoding .The decode and sequencing unit decodes the control signals and sends the control signals sequentially to the ALU. ALU (arithmetic and logic unit): ALU performs all the arithmetic and	*8M* *Description of all 3 units :4M*

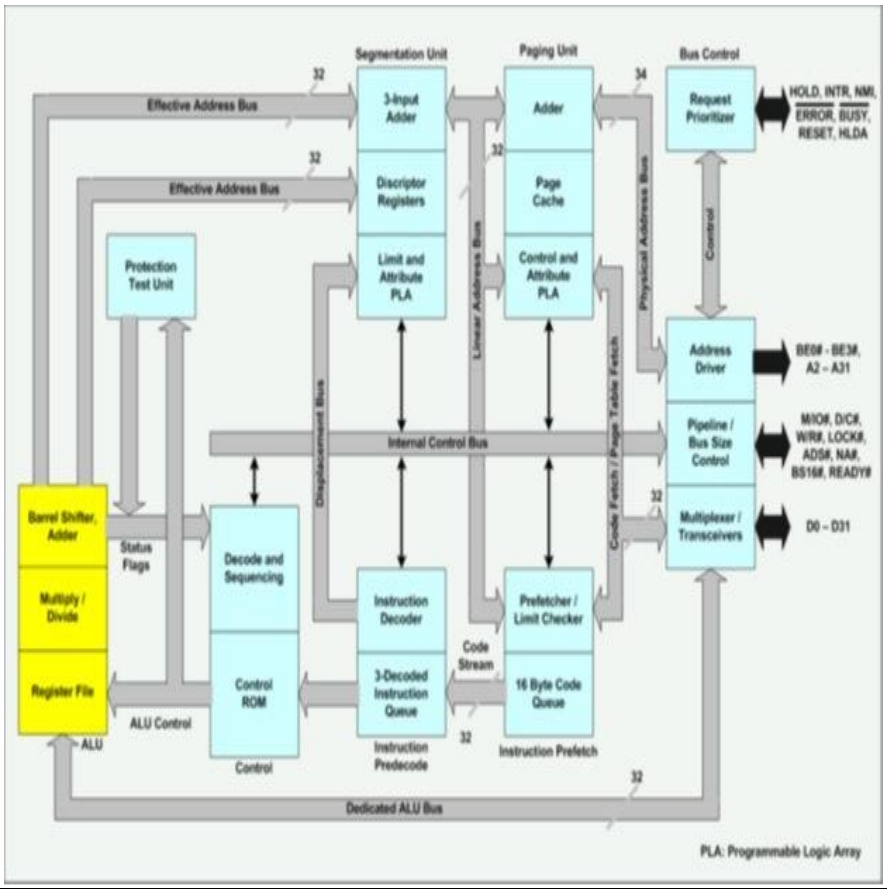
logical operations. It has a register file containing registers such as general purpose registers, control and flag registers, debug and test registers, special purpose registers etc. The barrel shifter is of 64 bits which can shift/rotate 64 bits at a time and hence can perform multiplication and divide operations within a microsecond.

The memory management unit has **segmentation unit** and **paging unit.**

The segmentation unit allows the use of two address components such as segment base address and offset address to calculate the physical address. It allows the size of the segment upto 4GB maximum. It provides the 4 level protection level mechanism for protecting and isolating the system's code and data from application programs and unauthorized access. This unit converts logical address spaces to the linear addresses. The Limit and Attribute PLA checks the segment limits and attributes at segment level to avoid invalid access to the code

The paging unit converts the linear addresses to the physical addresses. The control and attribute PLA checks the privileges at page level. Each of the pages maintain the paging information of the task. The paging unit organizes the physical memory in the terms of pages of 4KB each. This unit works under the control of segmentation unit i.e., each segment is further divided into pages. The virtual memory is also organized in the terms of segments and pages by the MMU.

The BIU has a bus control unit which has a request prioritizer which resolves the priorities of the various bus request operations. It also controls the access of the bus. The address drivers drives the bus (byte) enable signals BE0#-BE3# and the address signals A0-A31. The pipeline and bus size control unit handle the related control signals and supports the dynamic bus sizing feature. The data buffers (mux/ transceivers) interface the internal data bus with the system data bus.

			***Internal architecture of 80386 diagram : 4M***