WINTER – 2018 EXAMINATION
MODEL ANSWER

Subject: Advanced Microprocessor **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.

<table>
<tr><th>Q. No.</th><th>Sub Q.N.</th><th>Answer</th><th>Marking Scheme</th></tr>
<tr><td>1.</td><td>a)
Ans.</td><td>Attempt any five of the following.
Distinguish between Hardware Interrupt and Software Interrupt.

<table>
<tr><th>Sr. No.</th><th>Hardware Interrupt</th><th>Software Interrupt</th></tr>
<tr><td>1.</td><td>Used to handle asynchronous events.</td><td>Used to handle synchronous events</td></tr>
<tr><td>2.</td><td>Requested by external device.</td><td>Requested by microprocessor itself through program.</td></tr>
<tr><td>3.</td><td>After execution of these interrupts PC is not incremented.</td><td>After execution of these interrupts
PC is incremented.</td></tr>
<tr><td>4.</td><td>Some hardware interrupts are maskable.</td><td>All software interrupts are non-maskable.</td></tr>
<tr><td>5.</td><td>Lower priority than software interrupts.</td><td>Higher priority than hardware interrupts.</td></tr>
<tr><td>6.</td><td>Improves throughput of the</td><td>Does not improve</td></tr>
</table>
</td><td>20
4M

Any four points 1M each</td></tr>
</table>

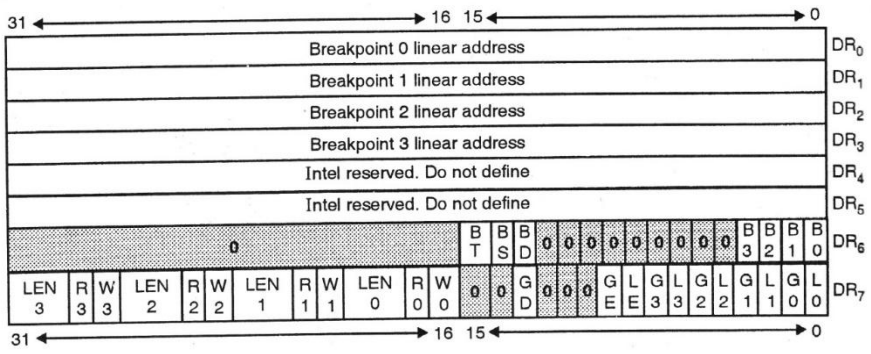
			system.	throughput of the system.
			7. Affect interrupt control logic.	Does not affect interrupt control logic.
			8. Microprocessor executes interrupt acknowledge cycle to acknowledge this interrupt.	Microprocessor does not executes interrupt acknowledge cycle to acknowledge this interrupt and only normal machine cycle is executed.
	b) Ans.	**List any eight features of 80386 processor.** **The salient features of 80386 are as follows:** 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 2^{32} 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 an integrated memory management unit which supports the virtual memory and four levels of protections. 8. It has an 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. 11. It supports instruction pipelining with the help of 16 bytes instruction prefetch queue. 12. It has eight, 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	**4M** ***Any eight features $^{1/2}$M each***

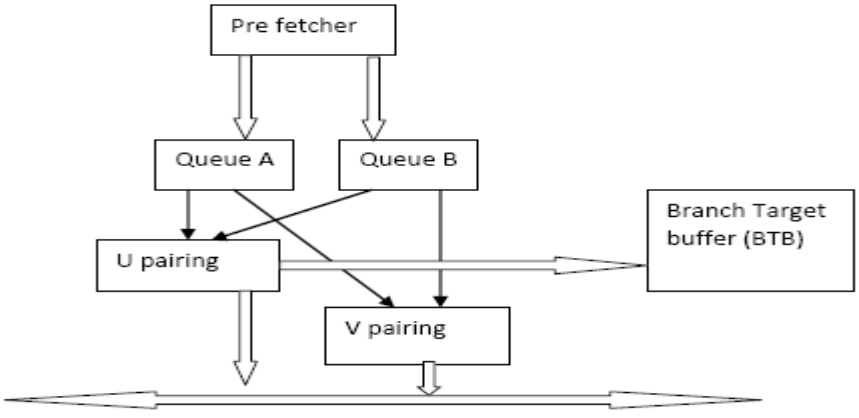
		control. 14. It has a 32 bit E-flag register. 15. It supports the dynamic bus sizing by which the 80386 can be interfaced to 16 bits devices effectively. And also supports the 8bits, 16 bits and 32 bits operands. 16. It operates on 20 MHz and 33 MHz frequency.	
	c) Ans.	**Explain the basic features of 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	**4M** ***Any four features 1M for each***

		Load and Store architecture, implying that all the memory accesses takes place using Load and Store type operations.	
	d) Ans.	**Enlist any eight salient features of Pentium processor.** **Features of Pentium processor:** 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). 10. On chip APIC (Advanced Programmable Interrupt Controller). 11. 3.3v operation. 12. Pentium address of a higher clock speed of 66MHZ 13. The cache structure has two caches one 815x8 cache is designed as an instruction cache & the other 815x8 is data cache. 14. The Build InSelf Test (BIST) allows the Pentium to be tested when power is first applied to the system. A BIST power up reset activates IUIT & then deactivate the reset pin. 15. The paging unit allows 4M byte pages. This is accomplished by using the page directory to address 1024 pages that each contains 4M byte of memory.	**4M** ***Any eight features of Pentium processor $^{1/2}$M each***
	e) 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:	**4M** ***Description of any four DOS interrupts 1M each***

		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 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:	

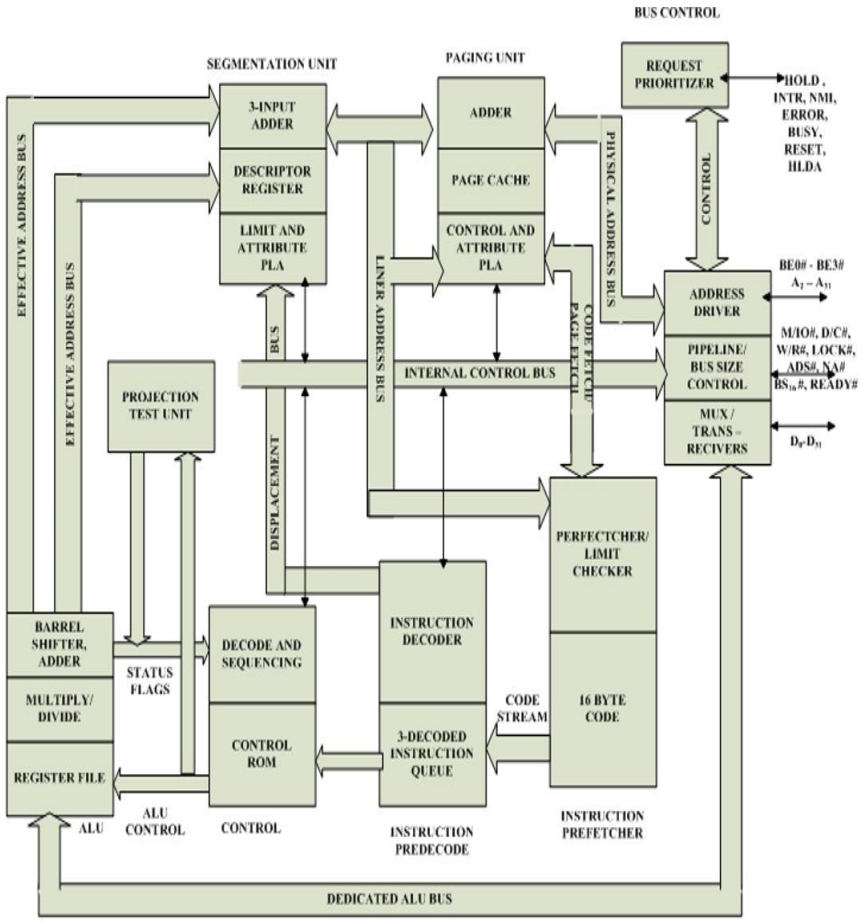
		ASCIIZ filename DS: DX - zero terminated full paths. *Syntax*: mov ah, 41h; delete file int 21h; transfer to DOS **7) 56H: to rename the file** This functions renames the given file with new name specified by ES: DI Registers to be used before calling the function using INT 21H : DS: DX address of ASCIIZ 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	
	f) Ans.	**Describe debug register of 80386 microprocessor.** Fig shows the sets of debug and test registers:  **<u>Debug Registers:</u>** The first four debug registers contain 32 bit linear breakpoint addresses. (A linear address is a 32-bit address generated by a microprocessor instruction that may or may not be same as the physical address.) The breakpoint address, which may locate an instruction or data are constantly compared with the address generated by the program. If a match occurs, the 80386 will cause a type 1 interrupt [TRAP] to occur, if directed by debug registers DR6 and DR7. This feature is much expanded version of the basic trapping or tracing allowed with the earlier processors through the type 1 interrupt. The breakpoint addresses are very useful in debugging faulty software. The control bits in DR6 and DR7 are defined as follows: **a. <u>BT:</u>** If set to 1, the debug interrupt was caused by a task switch.	**4M** ***Diagram for debugr egister 2M*** ***Description 2M***

		b. BS: If set the debug interrupt was caused by the TF bit in the flag register. **c. BD:** If set, the debug interrupt was caused by an attempt to read the debug register with the GD bit set. The GD bit protects access to the debug registers. **d. B3-B0:** Indicates which of the four debug breakpoint addresses caused the debug interrupt. **e. LEN:** Each of the four length fields pertains to each of the four breakpoint addresses stored in DR0-DR3. These bits further define the size of the access at the breakpoint address as 00(byte), 01(word), or 11(double word). **f. RW:** Each of the four read/write fields pertains to each of four breakpoint addresses stored in the DR0-DR3. The RW field selects the cause of action that enabled a breakpoint address as 00 (instruction access), 01(data write) and 11(data read and write). **g. GD:** If set, GD prevents any read or write of a debug interrupt by generating the debug interrupt. This bit is automatically cleared during the debug interrupt so that the debug registers can be read or changed if needed. **h. GE:** If set, selects a local breakpoint addresses for any of the four breakpoint address registers. **i. LE:** If set, selects a local breakpoint address for any of the four breakpoint address.	
	g) Ans.	**Explain branch prediction in Pentium.** 	**4M** ***Correct Diagram 2M***

		Branch Prediction Logic:- 1. Branch prediction is the technique to predict more likely set of instructions to be executed and pre-fetch to make them available to the pipeline as and when they are referred. 2. The Pentium employs a branch target buffer which is an associative memory used to improve the performance if it takes the branch instruction. 3. Branch instructions occur more frequently that changes the normal sequential control flow of the program execution and may stall the pipeline execution in the Pentium system. 4. If the branching is conditional then the CPU has to wait till the execution stage determines whether the condition is satisfied or not. If condition satisfies then the branching will take place. Pentium designer implemented a branch prediction technique algorithm to speed up of the instruction execution. 5. A 256 entry branch target buffer in the Pentium holds branch target addresses for previously executed branches. The branch target is four ways set associative memory. 6. Whenever a branch is taken, the CPU enters the branch instruction address and also the destination address in the branch instruction. 7. If there is hit that is there exist such entries, then the CPU use the history to decide whether the branch will be taken or not. 8. If the CPU based on its previous history decides to take the branch, it fetches the instruction from the target address and decodes them. 9. If the prediction is correct, the process continue else the CPU flushes the pipeline and fetches from the correct target address.	***Relevant explanation 2M***
2.	**a) Ans.**	**Attempt any two of the following.** **Draw and explain Internal architecture of 80386 processor.** **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	**16** **8M**

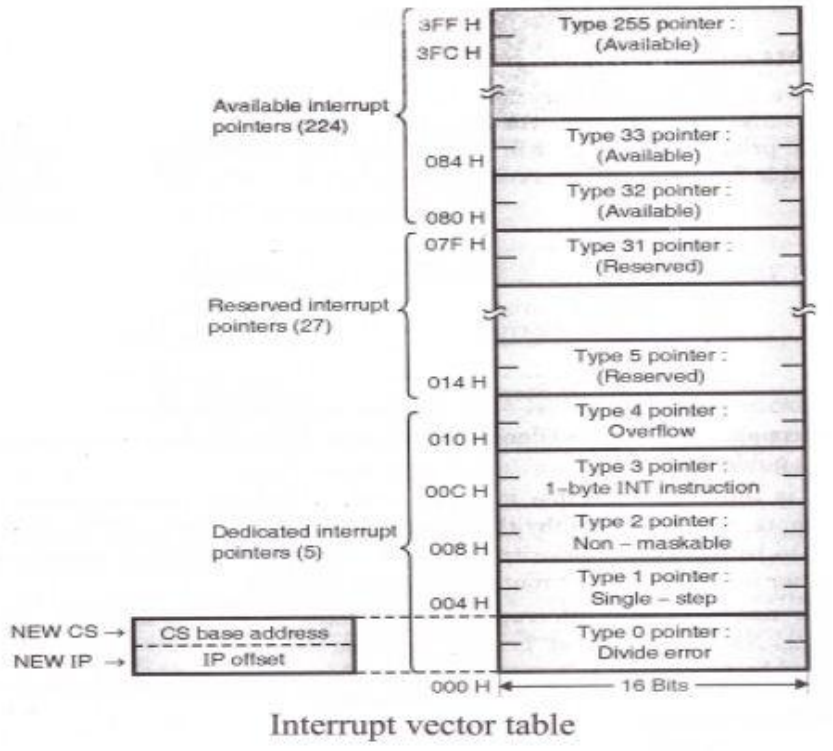
		unit Instruction unit has Instruction pre-fetcher and instruction pre-decode unit. The Instruction pre-fetcher fetches the 16 instruction bytes ahead of time and stores them into the 16 byte instruction pre-fetch queue(16 byte code).This speeds up the program execution process. The instruction pre-decode 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 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	***Description of all 3 units 4M***

		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 processor diagram 4M***
	b) Ans.	**Explain any four floating point exceptions in Pentium processor.** The Pentium provides 6 floating point exceptions: 1. Invalid operation 2. Divide by zero 3. De-normalized operand 4. Numeric overflow	**8M**

		5. Numeric underflow 6. Inexact result **1. Invalid operation** The floating point invalid exception occurs in response to two general types of operations : • Stack overflow or underflow • 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 underflow 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 underflow 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 bit 2 of the control word. The FDIV, FDIVP, FDIVR, FDIVRP, FIDIV, FIDIVR instructions and the other instructions that perform division internally can report the 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	***Any four floating point exceptions in Pentium processor 2M each***

		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 represent able in the destination format.	
	c) **Ans.**	**Explain any four DOS functions of INT 21 H with suitable example.** **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	**8M** ***Any four DOS functions of INT 21 H 2M each***

		= 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 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.	
3.		**Attempt any four of the following:**	**16**

	a) **Ans.**	**Draw and explain Interrupt Vector Table (IVT) of X86 processor.**  Interrupt vector table The interrupt vector table is a collection of 4 bytes addresses which resides in the 1KB memory. It tells the processor where it should jump to execute the associated Interrupt Service Routine. There are total 256 interrupts types. The IVT is 1KB long located in memory from 00000H to 003FFH. Each entry of 4 bytes is composes 2 bytes for CS and 2 bytes for IP. Out of 256 interrupts: 1. The lowest five types are dedicated interrupts such as ❖ Type 0: Divide by zero interrupt (INT0) ❖ Type 1: Single step interrupt (INT1) ❖ Type 2: Non-maskable interrupt (INT2) ❖ Type 3: Break point interrupt (INT3) ❖ Type 4: Overflow interrupt (INT4) 2. The next 27 interrupt types from Type 5 to Type31 are reserved	**4M** ***Diagram of Interrupt Vector Table (IVT) of X86 processor 2M*** ***Explanation2M***

		by Intel for use in future microprocessor. 3. The upper 224 interrupt types from Type 32 to Type 255 are available for you to use hardware or software or software interrupts. When the 8086 responds to an interrupt, it automatically goes to the specified location in the interrupts pointer table to get the starting address of the interrupt service procedure. In this table the new value of the instruction pointer is put as the low word of the pointer and the new value for code segment register is put in as the high word of the pointer.	
	b) Ans.	**Describe the 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 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 Indian 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	**4M** ***Any four features of Sun ULTRA SPARC-1M each***
	c) Ans.	**Draw and explain the format of CR_0 register of 80386.** **CR_0 Register of 80386:**	**4M**

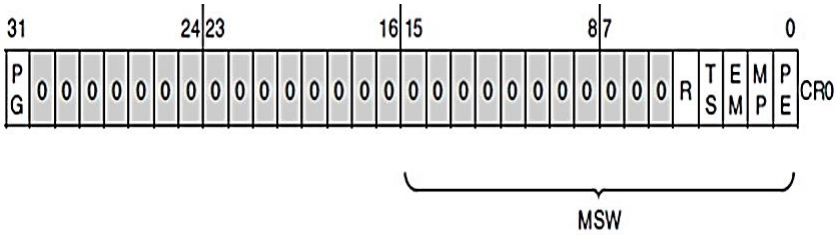


Diagram of CR_0 register of 80386 2M

- **CR_0** contains system control flags, which control or indicate conditions that apply to the system as a whole, not to an individual task.
- **PE** (Protection Enable, bit 0)

Setting PE causes the processor to begin executing in protected mode. This can be cleared by resetting the microprocessor. This can be set only in real mode.

Explanation 2M

- **MP** (Monitor processor extension/Coprocessor or Math Present, bit 1)

If this bit is set to 1, it allows the Wait instruction to generate a processor extension absent exception i.e. exception number 7.In short when this bit is set to 1 it indicates the absence of coprocessor (processor extension) if its not present and permits the emulation of the processor extension by the CPU.

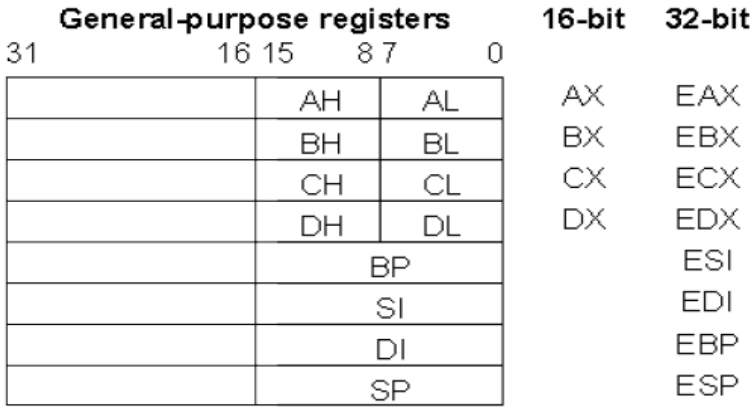
- **EM** (Emulate, bit 2)

If this bit is set to 1, it allows the generation of exception 7 (processor extension not present) and will permit the emulation of the processor extension by the CPU.(If this bit is set and the processor extension is absent it will allow the CPU to work as a coprocessor)

- **TS** (Task Switched, bit 3)

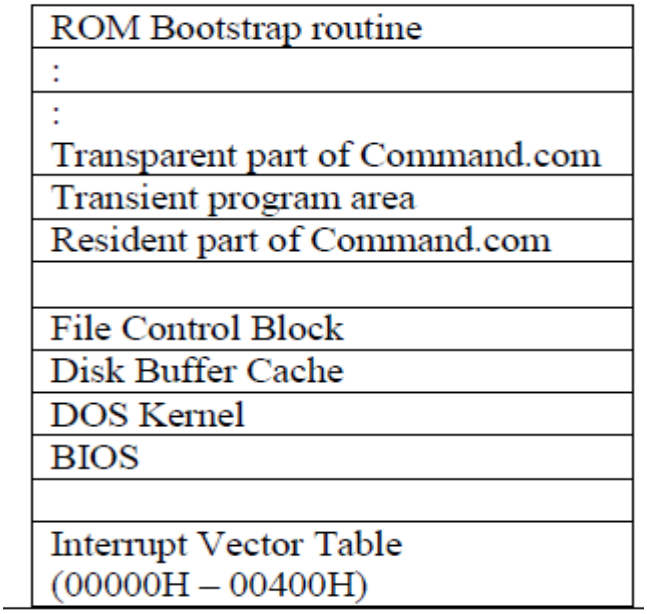
The TS bit of CR0 helps to determine when the context of the coprocessor does not match that of the task being executed by the 80286 CPU. The 80386 sets TS each time it performs a task switch (Whether triggered by software or by hardware interrupt). If, when interpreting one of the ESC instructions, the CPU finds TS already set, it causes exception 7. The WAIT instruction also causes

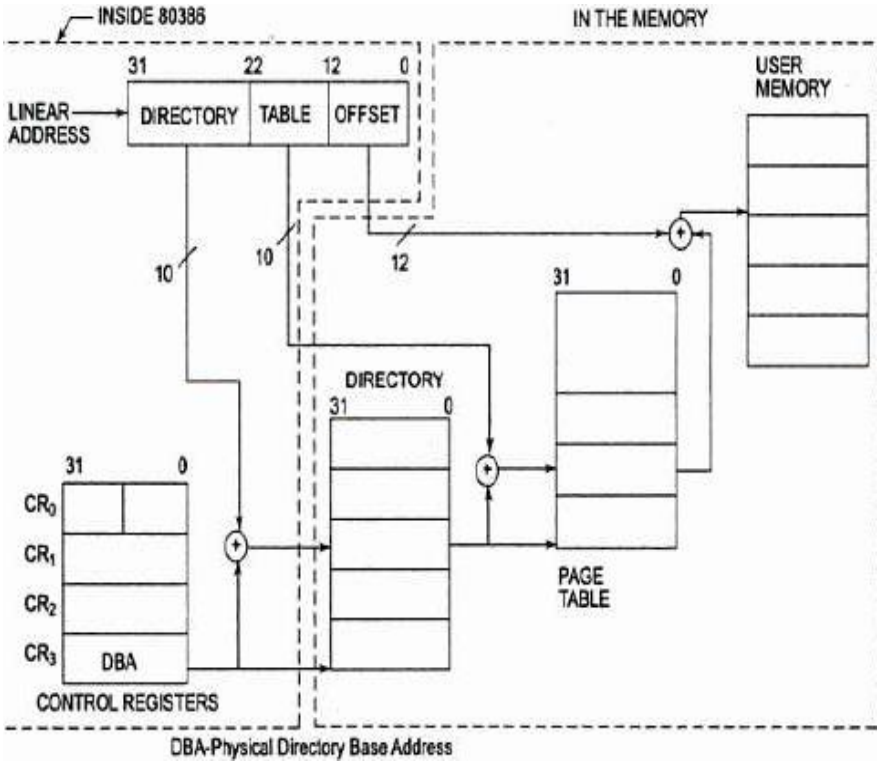
		exception 7 if both TS and MP are set. Operating systems can use this exception to switch the context of the coprocessor to correspond to the current task. • **ET** (Extension Type, bit 4) ET indicates the type of coprocessor present in the system (ET=0 -> 80287 or ET=1 ->80387) • **PG** (Paging, bit 31) PG indicates whether the processor uses page tables to translate linear addresses into physical addresses.	
	d) **Ans.**	**Describe the General purpose register of Pentium processor.** ***(Note: Diagram is optional)*** There are three types of registers: general-purpose data registers, segment registers, and status and control registers. The eight 32-bit general-purpose data registers are used to hold operands for logical and arithmetic operations, operands for address calculations and memory pointers. The following shows what they are used for: • EAX-Accumulator for operands and results data. • EBX-Pointer to data in the DS segment. • ECX-Counter for string and loop operations. • EDX-I/O pointer. • ESI-Pointer to data in the segment pointed to by the DS register; source pointer for string operations. • EDI-Pointer to data (or destination) in the segment pointed to by the ES register; destination pointer for string operations. • ESP-Stack pointer (in the SS segment). • EBP-Pointer to data on the stack (in the SS segment). The following figure shows the lower 16 bits of the general-purpose registers can be used with the names AX, BX, CX, DX, BP, SP, SI, and DI (the names for the corresponding 32-bit ones have a prefix "E" for "extended"). Each of the lower two bytes of the EAX, EBX, ECX, and EDX registers can be referenced by the names AH, BH, CH, and DH (high bytes) and AL, BL, CL, and DL (low bytes).	**4M** ***General Purpose register of Pentium Processor explanation 4M***



	e) Ans.	**Explain the hybrid architecture of RISC and CISC processors.** **Hybrid architecture:** State of the art processor technology has changed significantly since RISC chips were first introduced. Because a number of advancements are used by both RISC and CISC processors, the lines between the two architectures have begun to blur. In fact, 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. **OR** Processor designers where using two opposing camps some used CISC designs due to its low burden on compiler developers and wide availability of existing software while others used RISC designs due to its simplicity and efficiency. Most CISC processors are based on hybrid CISC- RISC architecture. Such architectures uses a decoder to convert a CISC instructions into RISC instructions before execution which are processed by RISC core which performance quickly and enhances performance due to branch prediction and pipelining. This have only been possible in	**4M** ***Explanation of hybrid architecture of RISC and CISC processors 4M***

		RISC design as fixed length instructions are required for such features to work. The Hybrid architecture processors are compatible with software develop for CISC predecessors yet they perform competitively against processors based on RISC design but CISC –RISC hybrids continue to consume lot of power and are not used for mobile and embedded applications RISC processors have become more CISC like by supporting more functions and more instructions than old CISC designs. Due to which application can run much faster this includes multimedia applications like telecommunications, encoding or decoding, image conversions and video processing.	
4.	**a)** **Ans.**	**Attempt any two of the following:** **Describe the loading sequence of MS-DOS in memory with the neat diagram.** 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	**16** **8M** ***Explanation 4M***

		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. **Diagram Of Loading Of Ms-Dos In Memory:** Top of RAM 	***Loading sequence of DOS in memory diagram 4M***
	b) **Ans.**	**What is paging? Explain concept of paging in 80386.** ***(Note : Any other relevant diagram of paging can be given marks)*** Paging is one of the memory management techniques used for virtual memory multitasking operating system. The segmentation scheme may divide the physical memory into a variable size segments but the paging divides the memory into a fixed size pages. The segments are supposed to be the logical segments of the program, but the pages do not have any logical relation with the program. The pages are just fixed size portions of the program module or data.	**8M** ***Paging Concept 2M***



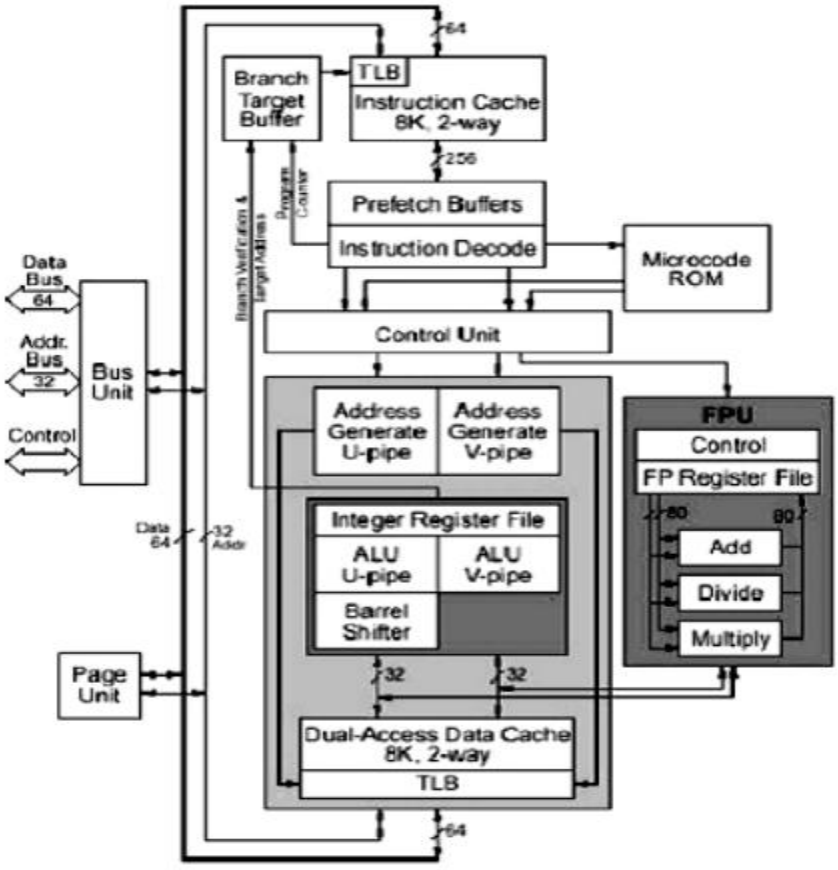
Paging diagram 3M

The paging unit operates under the control of segmentation unit. The paging unit if enabled converts linear addresses into physical address, in protected mode.

•**Paging Operation**: Paging is one of the memory management techniques used for virtual memory multitasking operating system.
•The segmentation scheme may divide the physical memory into a variable size segments but the paging divides the memory into a fixed size pages.
•The segments are supposed to be the logical segments of the program, but the pages do not have any logical relation with the program.
•The pages are just fixed size portions of the program module or data.

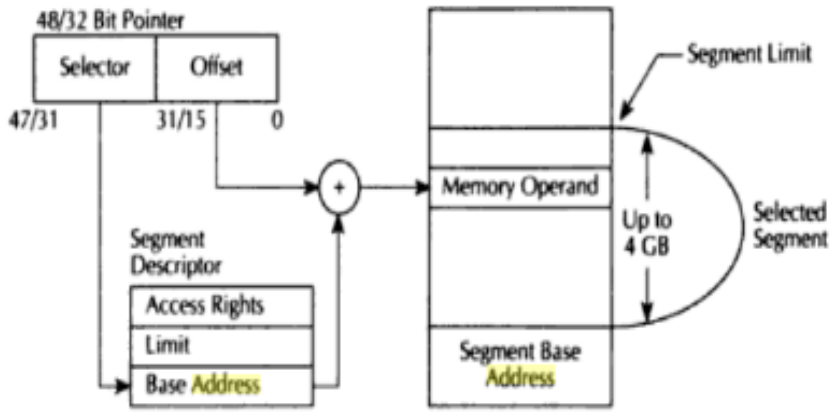
Explanation 3M

		•The advantage of paging scheme is that the complete segment of a task need not be in the physical memory at any time. •Only a few pages of the segments, which are required currently for the execution need to be available in the physical memory. Thus the memory requirement of the task is substantially reduced, relinquishing the available memory for other tasks. •Whenever the other pages of task are required for execution, they may be fetched from the secondary storage. •The previous page which are executed, need not be available in the memory, and hence the space occupied by them may be relinquished for other tasks. •Thus paging mechanism provides an effective technique to manage the physical memory for multitasking systems. •**Paging Unit:** The paging unit of 80386 uses a two level table mechanism to convert a linear address provided by segmentation unit into physical addresses. The paging unit converts the complete map of a task into pages, each of size 4K. The task is further handled in terms of its page, rather than segments. The paging unit handles every task in terms of three components namely page directory, page tables and page itself. •**Paging Descriptor Base Register:** The control register CR2 is used to store the 32-bit linear address at which the previous page fault was detected. The CR3 is used as page directory physical base address register, to store the physical starting address of the page directory. The lower 12 bit of the CR3 are always zero to ensure the page size aligned directory. A move operation to CR3 automatically loads the page table entry caches and a task switch operation, to load CR0 suitably. •**Page Directory:** This is at the most 4Kbytes in size. Each directory entry is of 4 bytes, thus a total of 1024 entries are allowed in a directory. The upper 10 bits of the linear address are used as an index to the corresponding page directory entry. The page directory entries point to page tables.	

		•**Page Tables:** Each page table is of 4Kbytes in size and many contain a maximum of 1024 entries. The page table entries contain the starting address of the page and the statistical information about the page. •The upper 20 bit page frame address is combined with the lower 12 bit of the linear address. The address bits A12- A21 are used to select the 1024 page table entries. The page table can be shared between the tasks.	
	c) Ans.	**Draw and explain Architecture of Pentium processor.** **Architecture of Pentium:**  Architecture of Pentium is as shown in the above diagram.	**8M** ***Architecture of Pentium processor diagram 4M***

		The most important enhancements over the 486 are the separate instruction and data caches, the dual integer pipelines (the U-pipeline and the V-pipeline, as Intel calls them), branch prediction using the branch target buffer (BTB), the pipelined floating-point unit, and the 64-bit external data bus. Even-parity checking is implemented for the data bus and the internal RAM arrays (caches and TLBs). As for new functions, there are only a few; nearly all the enhancements in Pentium are included to improve performance, and there are only a handful of new instructions. Pentium is the first high performance micro-processor to include a system management model like those found on power-miserly processors for notebooks and other battery-based applications; Intel is holding to its promise to include SMM on all new CPUs. The integer data path is in the middle, while the floating- point data path is on the side opposite the data cache. In contrast to other superscalar designs, such as Super SPARC, Pentium's integer data path is actually bigger than its FP data path. This is an indication of the extra logic associated with complex instruction support. Intel estimates about 30% of the transistors were devoted to compatibility with the x86 architecture. Much of this overhead is probably in the microcode ROM, instruction decode and control unit, and the adders in the two address generators, but there are other effects of the complex instruction set. For example, the higher frequency of memory references in x86 programs compared to RISC code led to the implementation of the dual-ac. Register set The purpose of the Register is to hold temporary results, and control the execution of the program. General-purpose registers in Pentium are EAX, ECX, EDX, EBX, ESP, EBP,ESI, or EDI. The 32-bit registers are named with prefix E, EAX, etc, and the least 16 bits 0-15 of these registers can be accessed with names such as AX, SI Similarly the lower eight bits (0-7) can be accessed with names such as AL & BL. The higher eight bits (8-15) with names such as AH & BH. The instruction pointer EAP known as program counter(PC) in 8-bit microprocessor, is a 32-bit register to handle 32bit memory addresses, and the lower 16 bit segment IP is used for 16bits memory address. The flag register is a 32-bit register, however 14-bits are being used at present for 13 different tasks; these flags are upward compatible with those of the 8086 and 80286.	***Explanation 4M***

		The comparison of the available flags in 16-bit and 32-bit microprocessor is may provide some clues related to capabilities of these processors. The 8086 has 9 flags, the 80286 has 11 flags, and the 80286 has 13 flags. All of these flag registers include 6 flags related to data conditions (sign, zero, carry, auxiliary, carry, overflow, and parity) and three flags related to machine operations.(interrupts, Single-step and Strings). The 80286 has two additional: I/O Privilege and Nested Task. The I/O Privilege uses two bits in protected mode to determine which I/O instructions can be used, and the nested task is used to show a link between two tasks.	
5.		**Attempt any four of the following:**	**16**
	a)	**Describe address generation PVAM mode of 80386 with neat diagram.**	**4M**
	Ans.	**Address calculation in protected mode:** Address generation in PVAM: In PVAM there are two components. A 16-bit selector which determines the linear base address of a segment and the base address is added to a 32-bit effective address to form a 32 bit linear address. The linear address is used as the 32-bit physical address or if paging is enabled the paging mechanism maps the 32-bit linear address into a 32 physical address. The selector is used to specify an index into an OS defined table that contains the 32-bit base address of given segment. The physical address is formed by adding the base address obtained from the table to the offset. Paging provides additional memory management that operates only in PVAM. It provides a mean of managing large segments of memory. The paging mechanism translates the protected linear address from segmentation unit into a physical address. **Diagram:**	***Explanation 2M***

		 Protected mode addressing	***Address generation PVAM of 80386 diagram 2M***
	b) Ans.	**Explain any four features of Intel MMX.** **List of features of Intel MMX technology:** 1. 57 new microprocessor instructions have been added that are designed to handle video, audio, and graphical data more efficiently. Programs can use MMX instructions without changing to a newmode or operating-system visible state. 2. New 64-bit integer data type (Quadword).(4 new MMX data types) 3. A new process, Single Instruction Multiple Data (SIMD), makes it possible for one instruction to perform the same operation on multiple data items. 4. The memory cache on the microprocessor has increased to 32 KB, meaning fewer accesses to memory that is off the microprocessor. 5. 8,64 bits wide MMX technology registers have are added to support the Multimedia .	**4M** ***Any four features 1M each***
	c) Ans.	**Write Advantages of the RISC processor.** **Advantages of RISC processor are as follows:** 1. RISC instructions are simple in nature and hence can be hardwired, while CISC architectures nay have to use microprogramming in order to implement microprogramming. 2. A set of simple instructions results in reduced complexity of the control unit and the data path. As a consequence the processor can work at a higher clock frequency and yields greater speed.	**4M**

		3. Several extra functionalities such as MMUs, floating point arithmetic units can also be placed on the same chip. 4. Smaller chips allow the semiconductor manufacturer to place more parts on a single silicon wafer, which can lower the cost of the processor's chip. 5. High level language compilers produce more efficient codes in a RISC processor than CISC, because they tend to use the smaller set of instructions in a RISC computer. 6. Shorter design cycle : a new RISC processor can be designed, developed and tested more quickly since they are simple than CISC processors. 7. Application programmers who use the microprocessor's instructions will find it easier to develop a code with the smaller and optimized instruction set. 8. The loading and decoding of the instructions in a RISC processor is simple and fast and it is not needed to wait until the length of the instruction is known in order to start decoding the following one. Decoding is simplified as op-code and address fields are located in the same location for all instructions.	***Any four advantages 1M each***
	d) **Ans.**	**Differentiate between 80386 and Pentium Processor (any 8 points).** ***(Note: Any other relevant differences shall be given marks).***	**4M**

Sr. No.	**80386**	**Pentium Processor**
1	Data Bus is of 32 bits	Data Bus is of 128 bits
2	Address Bus is 32 bits	Address Bus is 64 bits
3	Register size is 32 bits	Register size is 32/64 bits
4	Floating point pipeline is Not available	8 stages floating point pipeline is available
5	supports instruction pipelining of 16bytes	superscalar with 5 pipeline stages (u and v integer pipeline)
6	Works in 1.,Real mode,2.,Protected mode, Virtual mode	Works in 1., Protected mode,2. Real-Address mode
7	Only integer registers are	Separate floating point

Any eight differences of 80386 and Pentium Processor $^{1/2}$M each

			available.	registers are available.
		8	No branch prediction logic .	Separate branch prediction logic is available
		9	No support to cache memory	It supports separate data and code cache memory.
		10	No such feature is available.	It has a functional redundancy check
	e) **Ans.**	**Write difference between real mode and protected mode.** ***(Note: Any other relevant differences shall be given marks).***		**4M**

Sr. No.	Real Mode	Protected mode
1.	In real mode the 80386 microprocessor works as Fast 8086 processor	In protected mode the 80386 microprocessor shows its all the enhanced features.
2.	Data bits only D0-D15 are available	All the data bits from D0-D31 are available.
3.	Address bits A0-A19 are available	All the address bits A0-A31 are available
4.	Physical memory size up to 1MB is available	Physical memory size up to 4GB is available
5.	Virtual memory is not available in this mode.	Virtual memory available is upto 64TB.
6.	Register size is only 8/16 bits.	8/16/32 bits registers can be used in this mode.

Any four differences 1M each

6.	**a)** **Ans.**	**Attempt any two of the following:** **Draw the format of flag register and explain it in detail.** **Diagram of format of flag register:**	**16** **8M**

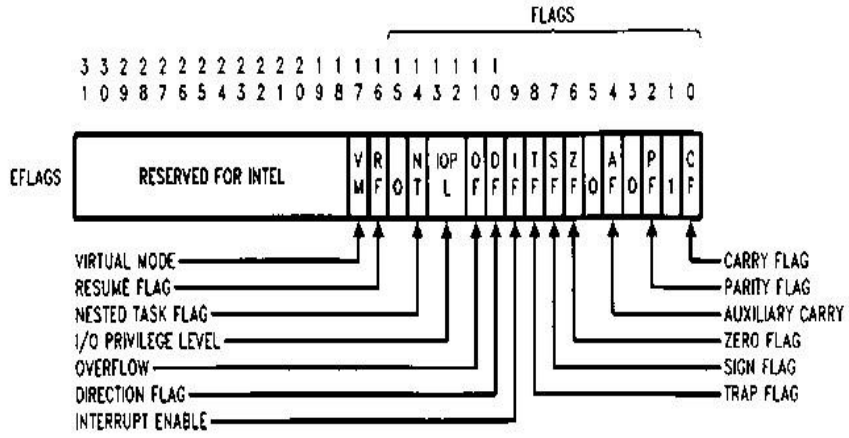


Diagram of format of flag register 4M

The flags in the flag register are :

VM (Virtual 8086 Mode, bit 17)

The VM bit provides Virtual 8086 Mode within Protected Mode. If set while the 80386 is in Protected Mode, the 80386 will switch to Virtual 8086 operation, handling segment loads as the 8086 does, but generating exception 13 faults on privileged opcodes.

RF (Resume Flag, bit 16)

The RF flag is used in conjunction with the debug register breakpoints. It is checked at instruction boundaries before breakpoint processing. When RF is set, it causes any debug fault to be ignored on the next instruction. RF is then automatically reset at the successful completion of every instruction.

Explanation 4M

NT (Nested Task, bit 14)

This flag applies to Protected Mode. NT is set to indicate that the execution of this task is nested within another task. If set, it indicates that the current nested task's Task State Segment (TSS) has a valid back link to the previous task's TSS. This bit is set or reset by control transfers to other tasks.

IOPL (Input/Output Privilege Level, bits 12-13)

This two-bit field applies to Protected Mode. IOPL indicates the numerically maximum CPL (current privilege level) value permitted to execute I/O instructions without generating an exception 13 fault

		or consulting the I/O Permission Bitmap. It also indicates the maximum CPL value allowing alteration of the IF (INTR Enable Flag) bit when new values are popped into the EFLAG register. **OF (Overflow Flag, bit 11)** OF is set if the operation resulted in a signed overflow. Signed overflow occurs when the operation resulted in carry/borrow into the sign bit (high-order bit) of the result but did not result in a carry/borrow out of the high order bit, or vice-versa. For 8/16/32 bit operations, OF is set according to overflow at bit 7/15/31, respectively. **DF (Direction Flag, bit 10)** DF defines whether ESI and/or EDI registers post decrement or post increment during the string instructions. Post increment occurs if DF is reset. Post decrement occurs if DF is set. **IF (INTR Enable Flag, bit 9)** The IF flag, when set, allows recognition of external interrupts signaled on the INTR pin. When IF is reset, external interrupts signaled on the INTR are not recognized. IOPL indicates the maximum CPL value allowing alteration of the IF bit when new values are popped into EFLAGS or FLAGS. **TF (Trap Enable Flag, bit 8)** TF controls the generation of exception 1trap when single-stepping through code. When TF is set, the 80386 generates an exception 1 trap after the next instruction is executed. When TF is reset, exception 1 traps occur only as a function of the breakpoint addresses loaded into debug registers DR0±DR3. **SF (Sign Flag, bit 7)** SF is set if the high-order bit of the result is set, it is reset otherwise. For 8-, 16-, 32-bit operations, SF reflects the state of bit 7, 15, 31 respectively. **ZF (Zero Flag, bit 6)** ZF is set if all bits of the result are 0. Otherwise it is reset.	

		AF (Auxiliary Carry Flag, bit 4) The Auxiliary Flag is used to simplify the addition and subtraction of packed BCD quantities. AF is set if the operation resulted in a carry out of bit 3 (addition) or a borrow into bit 3 (subtraction). Otherwise AF is reset. AF is affected by carry out of, or borrow into bit 3 only, regardless of overall operand length: 8, 16 or 32 bits. **PF (Parity Flags, bit 2)** PF is set if the low-order eight bits of the operation contains an even number of ``1's" (even parity). PF is reset if the low-order eight bits have odd parity. PF is a function of only the low-order eight bits, regardless of operand size. **CF (Carry Flag, bit 0)** CF is set if the operation resulted in a carry out of (addition), or a borrow into (subtraction) the high-order bit. Otherwise CF is reset. For 8-, 16- or 32-bit operations, CF is set according to carry/borrow at bit 7, 15 or 31, respectively.	
	b) **Ans.**	**What are the design issues of RISC processor.** **Design issues of RISC processor are as follow:** 1. Register Window 2. Memory speed issue 3. Instruction Latency issue 4. Dependencies issues **1. Register Window:** 1. 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). 2. 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. 3. 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. 4. Most RISC CPUs have some global registers which are always accessible. The remaining registers are windowed so that only a	**8M** ***Explanation of each design issue 2M***

		subset of the registers are accessible at any specific time. 5. To understand how register windows work, we consider the windowing scheme used by the Sun SPARC processor. 6. 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. 7. Of these 32 registers, 8 are global registers that are always accessible. The remaining 24 registers are contained in the register window. 8. The register window overlap. The overlap consists of 8 registers in SPARC CPU. Notice that the organization of the windows are supposed to be circular and not linear; meaning that the last window overlaps with the first window. 9. 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 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 reusing 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. **OR** **Design issues of RISC processor are:** 1. Register Window. 2. Pipelining in RISC 3. Single cycle instruction execution in RISC: 4. Dependencies: **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 are 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 organization 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. • The RISC CPU must keep track of which window is active and which windows contain valid data. A window pointer register contains the value of the window that is currently active. A window mask register contains 1 bit per window and denotes which windows contains valid data. • Register windows provide their greatest benefit when the CPU calls a subroutine. During the calling process, the register window is moved down 1 window position. In the SPARC CPU, if window 1 is active and the CPU calls a subroutine, the processor activates window 2 by updating the window pointer and window mask registers. The CPU can pass parameters to the subroutine via the registers that overlap both windows instead of memory. This saves a lot of time when accessing data. The CPU can use the same registers to return results to the calling routine. Drawbacks of register windowing is that on interactions with the system, the registers need to be flushed to the stack, necessitating the long sequence of writes to memory of data that is often mostly garbage. It opposes the multitasking workloads and by considering compilers with poor optimization. **2. Pipelining in RISC:** A RISC processor pipeline operates in much the same way, although the stages in the pipeline are different. While different processors have different numbers of steps, they are	

		basically variations of these five, used in the MIPS R3000 processor: 1. Fetch instructions from memory 2. Read registers and decode the instruction 3. Execute the instruction or calculate an address 4. Access an operand in data memory 5. Write the result into a register The length of the pipeline is dependent on the length of the longest step. Because RISC instructions are simpler than those used in pre RISC processors (now called CISC, or Complex Instruction Set Computer), they are more conducive to pipelining. While CISC instructions varied in length, RISC instructions are all the same length and can be fetched in a single operation. Ideally, each of the stages in a RISC processor pipeline should take 1 clock cycle so that the processor finishes an execution of every instruction in same time. Pipeline Problems In practice, however, RISC processors operate at more than one cycle per instruction. The processor might occasionally stall a result of data dependencies and branch instructions. A data dependency occurs when an instruction depends on the results of a previous instruction. A particular instruction might need data in a register which has not yet been stored since that is the job of a preceding instruction which has not yet reached that step in the pipeline. Branch instructions are those that tell the processor to make a decision about what the next instruction to be executed should be based on the results of another instruction. Branch instructions can be troublesome in a pipeline if a branch is conditional on the results of an instruction which has not yet finished its path through the pipeline. **3. Single cycle instruction execution in RISC:** RISC designers are concerned primarily with creating the fastest chip possible, and so they use a number of techniques, including pipelining. Pipelining is a design technique where the computer's hardware processes more than one instruction at a time, and doesn't wait for one instruction to complete before starting the next. The four stages in our typical CISC machine are fetch, decode, execute, and write. These same stages exist in a RISC machine, but the stages are executed in parallel. As soon as one stage completes, it passes on the result to the next stage and then begins working on another instruction. The performance of a pipelined system depends on the time it takes only for any one stage	

		to be completed-not on the total time for all stages as with non-pipelined designs. In an typical pipelined RISC design, each instruction takes 1 clock cycle for each stage, so the processor can accept 1 new instruction per clock. Pipelining doesn't improve the latency of instructions (each instruction still requires the same amount of time to complete), but it does improve the overall throughput. As with CISC computers, the ideal is not always achieved. Sometimes pipelined instructions take more than one clock to complete a stage. When that happens, the processor has to stall and not accept new instructions until the slow instruction has moved on to the next stage. Since the processor is sitting idle when stalled, both the designers and programmers of RISC systems make a conscious effort to avoid stalls. To do this, designers employ several techniques. **4. Dependencies:** 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. One common optimization involves "common sub-expression elimination." A compiler which encounters the commands: B = 10 * (A / 3); C = (A/ 3) / 4; might calculate (A/3) first, put that result into a temporary variable, and then use the temporary variable in later calculations. Another optimization involves "loop unrolling." Instead of executing a sequence of instruction inside a loop, the compiler may replicate the instructions multiple times. This eliminates the overhead of calculating and testing the loop control variable. Compilers also perform function in lining, where a call to a small subroutine is replaced by the code of the subroutine itself. This gets rid of the overhead of a call/return sequence.	
	c) **Ans.**	**Describe the eight stage pipelining mechanism in floating point unit of Pentium.** **Eight stage pipelining mechanism in floating point unit of Pentium:**	**8M**

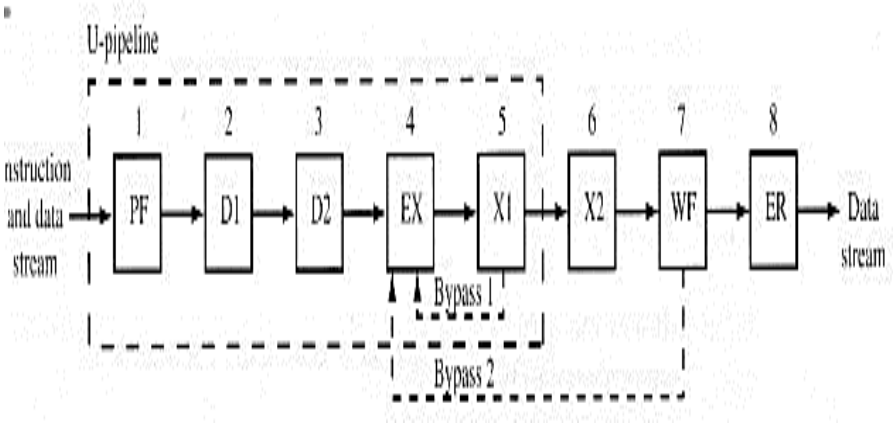


Diagram 4M

Floating Point Pipeline : Floating Point Unit stages in Pentium Processor : The floating point pipeline has **8 stages** as follows:

1. Prefetch (PF) :
- Instructions are prefetched from the on-chip instruction cache

2. Instruction Decode (D1):
- Two parallel decoders attempt to decode and issue the next two sequential instructions
- It decodes the instruction to generate a control word
- A single control word causes direct execution of an instruction
- Complex instructions require micro-coded control sequencing

3. Address Generate (D2):
- Decodes the control word
- Address of memory resident operands are calculated

Explanation 4M

4. Memory and Register Read (Execution Stage) (EX):
- Register read, memory read or memory write performed as required by the instruction to access an operand.

5. Floating Point Execution Stage 1 (X1):
- Information from register or memory is written into FP register.
- Data is converted to floating point format before being loaded into the floating point unit.

6. Floating Point Execution Stage 2 (X2):
- Floating point operation performed within floating point unit.

		7. Write FP Result (WF): • Floating point results are rounded and the result is written to the target floating point register. **8. Error Reporting(ER)** • If an error is detected, an error reporting stage is entered where the error is reported and • FPU status word is updated.	