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

MODEL ANSWER

WINTER - 2017 EXAMINATION

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.

Q. No.	Sub Q.N.	Answer	Marking Scheme
1.	**A)** **i)** **Ans.**	**Attempt any three.** **List salient features of 80386 (4 points).** **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 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 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	**12** **4M** ***Any 4 features 1M each***

MODEL ANSWER

WINTER - 2017 EXAMINATION

Subject: Advanced Microprocessor **Subject Code: 17627**

		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 8,32 bit General Purpose bits registers to store the data and address at the time of programming. 13. It has 8 debug registers DR_0-DR_7 for hardware debugging and control. 14. It has a 32 bit Eflag 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.	
	ii) **Ans.**	**List salient features of pentium processor.** **Features of Pentium processor:** **1. Pentium processor has 64 bit data bus** **8 bytes** of data information can be transferred to and from memory in a single bus cycle with the help of 64 bits data lines. It supports **burst read and burst write back cycles** It supports **pipelining** **2. It has a separate Instruction cache** Pentium processor has **8 KB** of dedicated instruction cache It has **Two Integer execution units, one Floating point execution unit** It has a **Dual** instruction pipeline It has **256 lines** between instruction cache and prefetch buffers; allows 32 bytes to be transferred from cache to buffer **3. It has a separate Data cache** It has a **8 KB** dedicated **data ca**che gives data to execution units It has **32 byte lines.** **4. Pentium processor has Two parallel integer execution units** It Allows the execution of **two instructions to be executed simultaneously** in a single processor clock	**4M** ***Any 4 features 1M each***

		5. It has a Floating point unit for Faster internal operations **6. It has a Local advanced programmable interrupt controller , it Speeds up upto 5 times** for common operations including add, multiply and load, than 80486 **7. It has a Branch Prediction Logic** To reduce the time required for a branch caused by **internal delays** When a branch instruction is encountered, microprocessor begins prefetch instruction at the branch address **8. It has a Data Integrity and Error Detection logic** Has significant error detection and data integrity capability Data parity checking is done on byte – byte basis **Address parity checking** and **internal parity checking** features are added **9. It has a Dual Integer Processor** which allows execution of **two instructions per clock cycle** **10. It has a Functional redundancy check** To provide maximum error detection of the processor and interface to the processor. A **second processor 'checker'** is used to execute in lock step with the 'master' processor It checks the master's output and compares the value with the internal computed values. An error signal is generated in case of mismatch **11. It has a Superscalar architecture,** which has Three execution units One execution unit executes floating point instructions The other two (U pipe and V pipe) execute integer instructions	
	iii) Ans.	**State features of RISC processor.** **Features of RISC processor:** 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	**4M** ***Any 4 features 1M each***

		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.	
	iv) **Ans.**	**Describe the function of the following pins of 80386.** **1) $\overline{BS_{16}}$ 2) $\overline{READY}$** **3) PEREQ 4) $\overline{B_0}$ - $\overline{B_3}$** 1) **BS16(#) :** BUS SIZE 16# :active low **input signal** : This input pin allows the interfacing of 16 bit devices with the 32 bits wide data bus of 386. It selects 32 bits data bus D0-D31 if its 1 and 16 bits data bus D0-D15 if its 0. Dynamic bus sizing is supported by 386 with the help of this pin. Asserting this input will disable the BE02# and BE3# signals and will enable only 16 bit data transfer operations. **2) READY(#) :** Ready is a active low input signal for 80386. When the external peripherals make this signal low it indicates that the external peripherals are not able to cope up with the speed of the processor and hence the processor has to wait for some time. This active low input puts the processor in wait state. 3) **PEREQ :** processor extension request active high input signal:	**4M** ***Function of pins 1M each***

It is an input signal for 80386 from the processor extension i.e. coprocessor.

When the processor extension requires the data operand transfer it requests the processor by asserting this signal high and issuing it to the processor. It indicates the request to the 80386 to perform a data operand transfer for a processor extension.

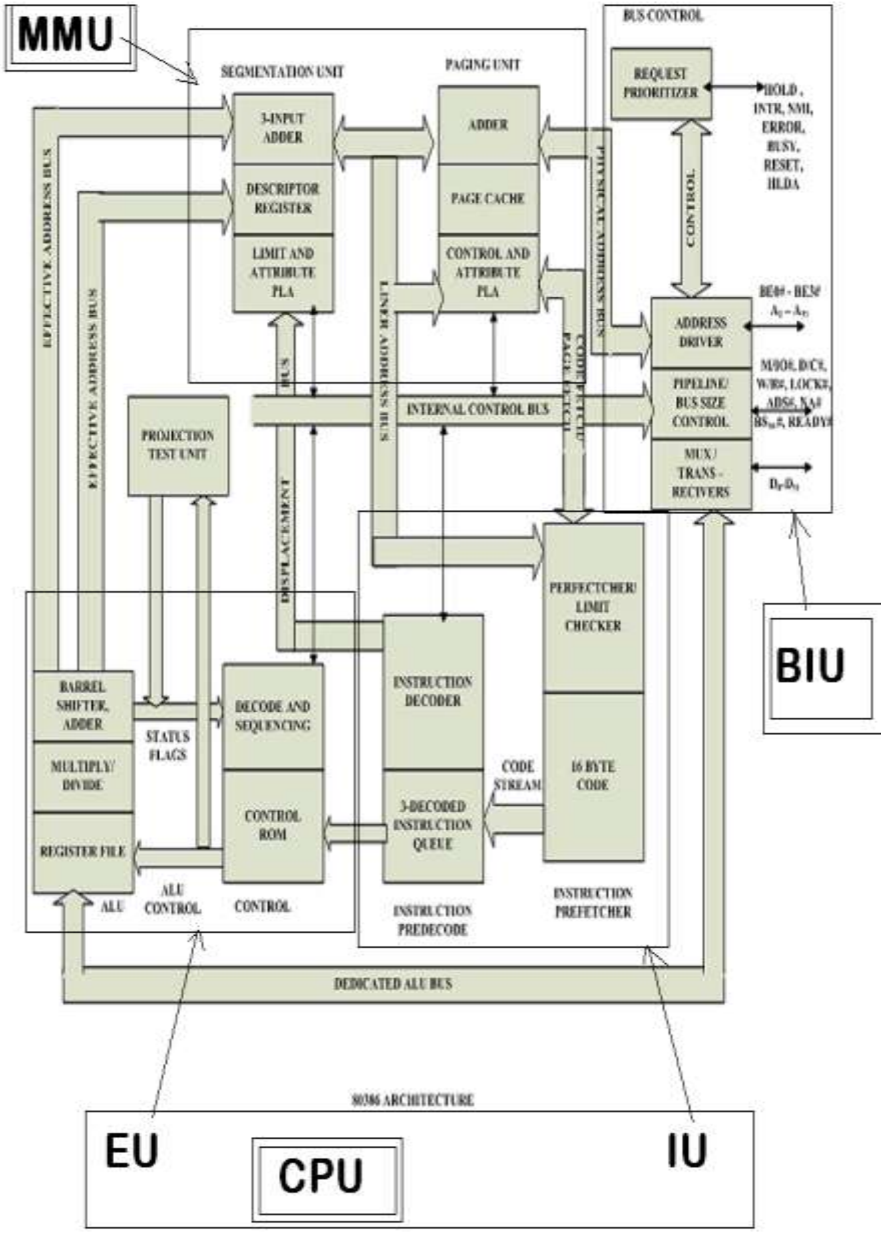
4) **B0(#)-B3(#) :** BUS SIZE 16# :active low **input signal** : This input pin allows the interfacing of 16 bit devices with the 32 bits wide data bus of 386. It selects 32 bits data bus D0-D31 if its 1 and 16 bits data bus D0-D15 if its 0. Dynamic bus sizing is supported by 386 with the help of this pin. Asserting this input will disable the BE02# and BE3# signals and will enable only 16 bit data transfer operations.

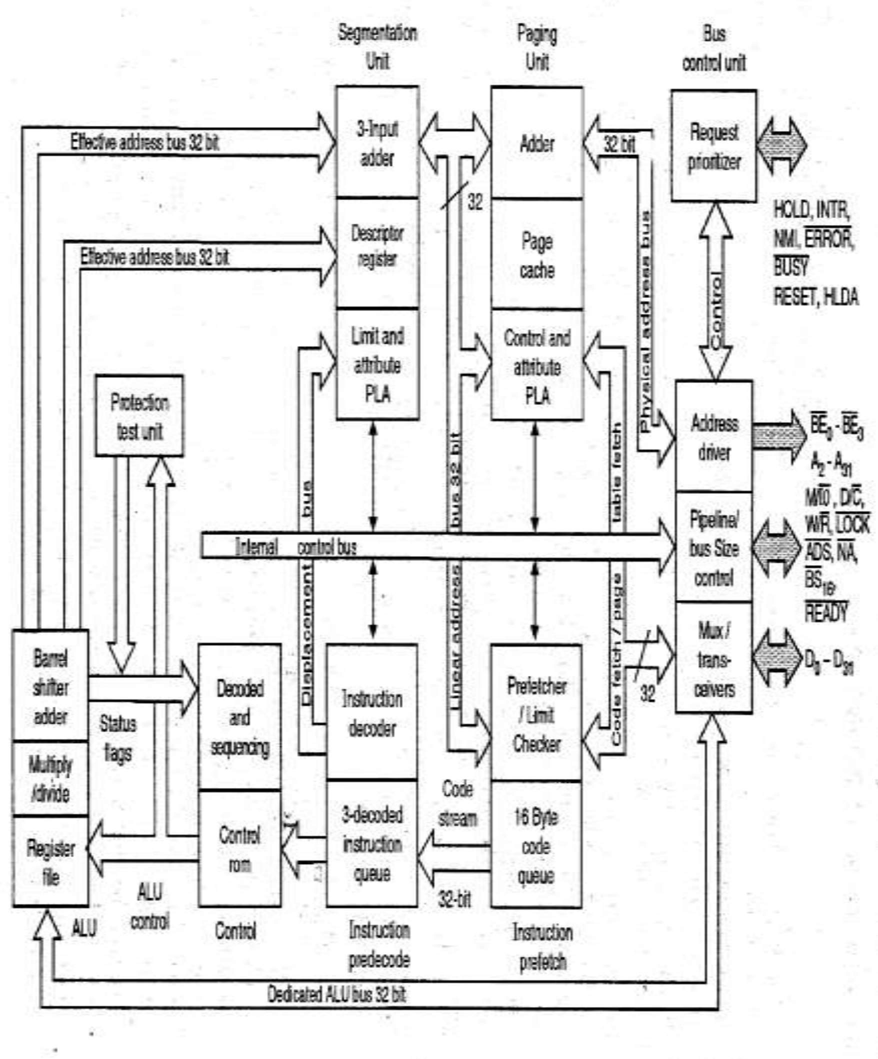
BE0#	D0-D7
BE1#	D8-D15
BE2#	D16-D23
BE3#	D24-D31

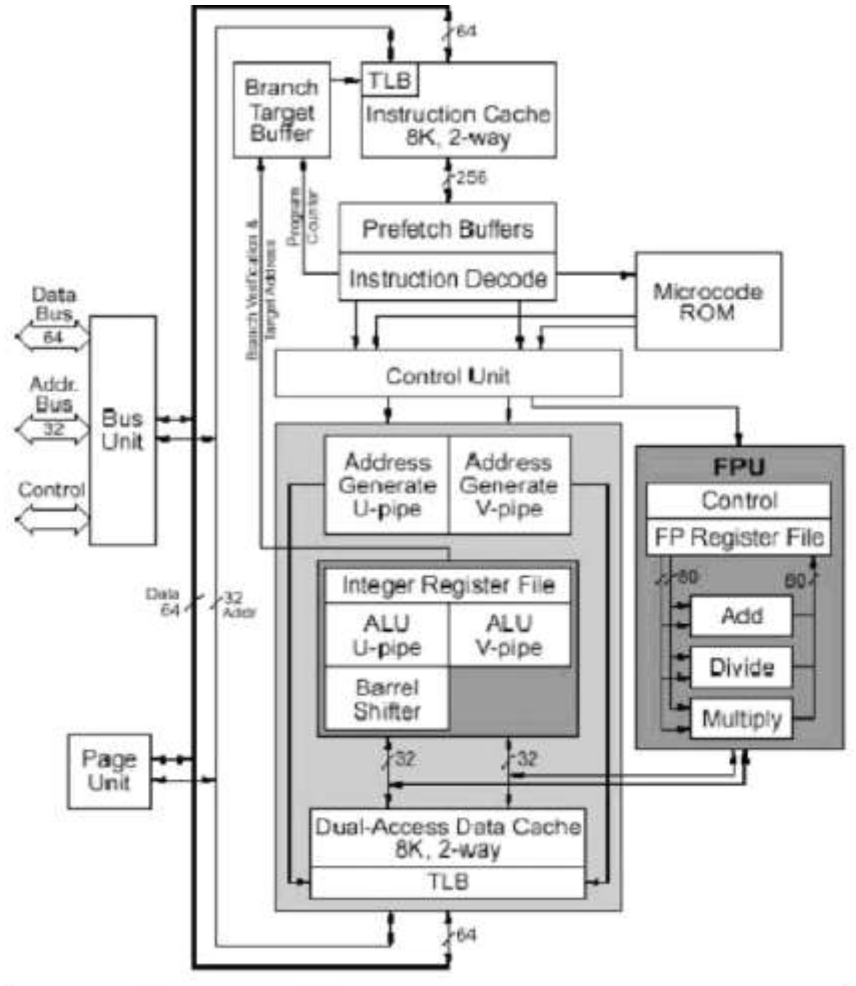
1. **(B)** **Attempt any one.** **6**

i) **Draw the neat labelled architecture of 80386.** **6M**

Ans.

		 MMU SEGMENTATION UNIT PAGING UNIT BUS CONTROL 3-INPUT ADDER DESCRIPTOR REGISTER LIMIT AND ATTRIBUTE PLA ADDER PAGE CACHE CONTROL AND ATTRIBUTE PLA REQUEST PRIORITIZER HOLD, INTR, NMI, ERROR, BUSY, RESET, HLDA CONTROL PHYSICAL ADDRESS BUS ADDRESS DRIVER BE0# - BE3# $A_2 - A_{31}$ PIPELINE/ BUS SIZE CONTROL M/IO#, D/C#, W/R#, LOCK#, ADS#, NA#, BS_{16}#, READY# MUX / TRANS - RECIVERS D_0-D_{31} EFFECTIVE ADDRESS BUS EFFECTIVE ADDRESS BUS BUS LINER ADDRESS BUS CODE FETCH/ PAGE FETCH INTERNAL CONTROL BUS PROJECTION TEST UNIT DISPLACEMENT PERFECTCHER/ LIMIT CHECKER BIU BARREL SHIFTER, ADDER MULTIPLY/ DIVIDE REGISTER FILE STATUS FLAGS DECODE AND SEQUENCING CONTROL ROM INSTRUCTION DECODER 3-DECODED INSTRUCTION QUEUE CODE STREAM 16 BYTE CODE ALU ALU CONTROL CONTROL INSTRUCTION PREDECODE INSTRUCTION PREFETCHER DEDICATED ALU BUS 80386 ARCHITECTURE EU CPU IU **OR**	***Neat labelled architecture of 80386 6M***

		 Segmentation Unit Paging Unit Bus control unit 3-Input adder Adder Request prioritizer Effective address bus 32 bit 32 bit Descriptor register Page cache 32 HOLD, INTR, NMI, ERROR, BUSY RESET, HLDA Effective address bus 32 bit Limit and attribute PLA Control and attribute PLA Physical address bus Control Protection test unit Address driver $\overline{BE_0}$ - $\overline{BE_3}$ A_2 - A_{31} $M/\overline{IO}$, $D/\overline{C}$, $W/\overline{R}$, $\overline{LOCK}$ $\overline{ADS}$, $\overline{NA}$, $\overline{BS_{16}}$, $\overline{READY}$ Displacement bus Linear address bus 32 bit Code fetch / page table fetch Internal control bus Pipeline/ bus Size control Barrel shifter adder Status flags Decoded and sequencing Instruction decoder Prefetcher / Limit Checker Mux / trans-ceivers D_0 - D_{31} 32 Multiply /divide Register file Control rom 3-decoded instruction queue Code stream 16 Byte code queue 32-bit ALU ALU control Control Instruction predecode Instruction prefetch Dedicated ALU bus 32 bit	
	ii) **Ans.**	**Describe the pentium CPU architecture with neat sketch.** ***(Note: Any other relevant description shall be considered).*** **Pentium architecture:**	**6M**

		 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 Data 64 32 Addr Integer Register File ALU U-pipe ALU V-pipe Barrel Shifter 32 32 Page Unit Dual-Access Data Cache 8K, 2-way TLB 64 **Architecture of Pentium is as shown in the above diagram.** 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 mode	***Diagram 3M*** ***Description 3M***

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.

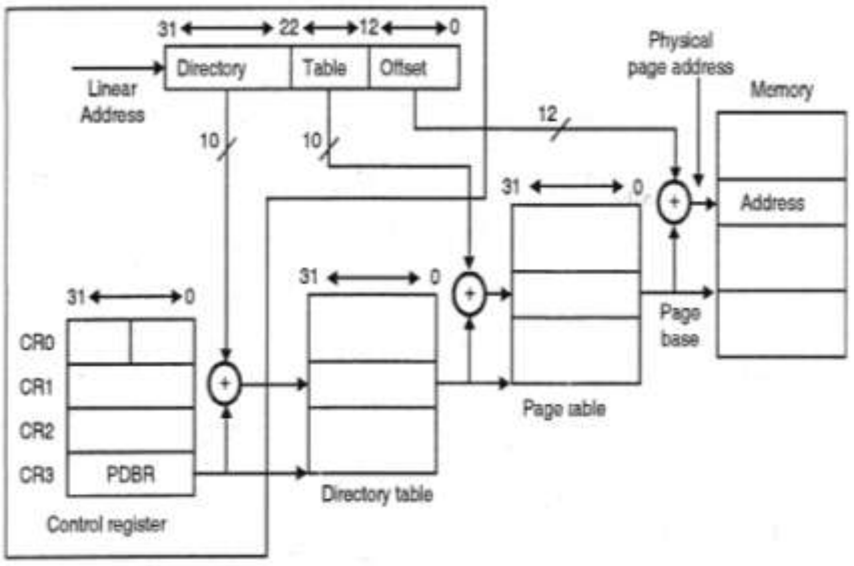
The integer data path is in the middle, while the floating- point data path is on the side opposite the data cache. In contrast to other superscalar designs, such as Super SPARC, Pentium's integer data path is actually bigger than its FP data path. This is an indication of the extra logic associated 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 32-bit memory addresses, and the lower 16 bit segment IP is used for 16-bi 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. 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.

		The processor also includes control registers and system address registers, debug and test registers for system and debugging operations.	
2.		**Attempt any four.**	**16**
	1)	**Describe the concept of paging mechanism in 80386.**	**4M**
	Ans.	**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. Thus paging mechanism provides an effective technique to manage the physical memory for multitasking systems. Two level paging scheme  31 22 12 0 Directory Table Offset Linear Address Physical page address Memory 10 10 12 31 0 Address Page base Page table Directory table CR0 CR1 CR2 CR3 PDBR Control register • **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.	***Description 2M*** ***Diagram 2M***

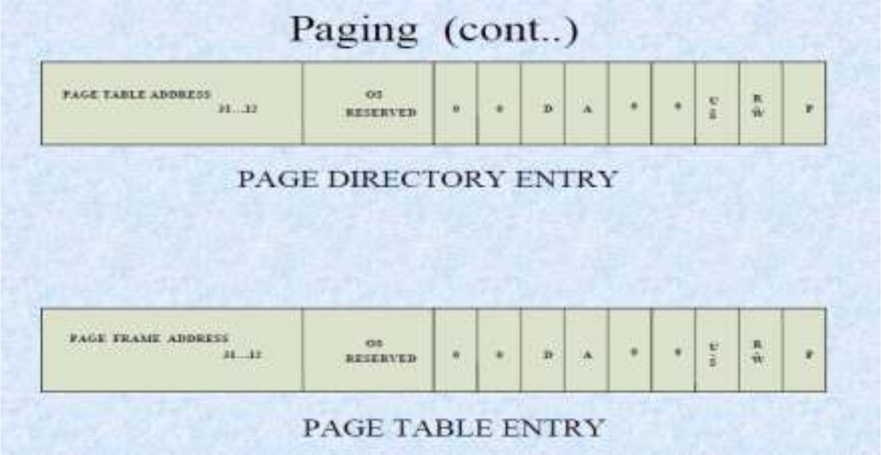
- The paging unit converts the complete map of a task into pages, each of size 4K. The task is further handled in terms of its page, rather than segments.
- The paging unit handles every task in terms of three components namely page directory, page tables and page itself.

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.

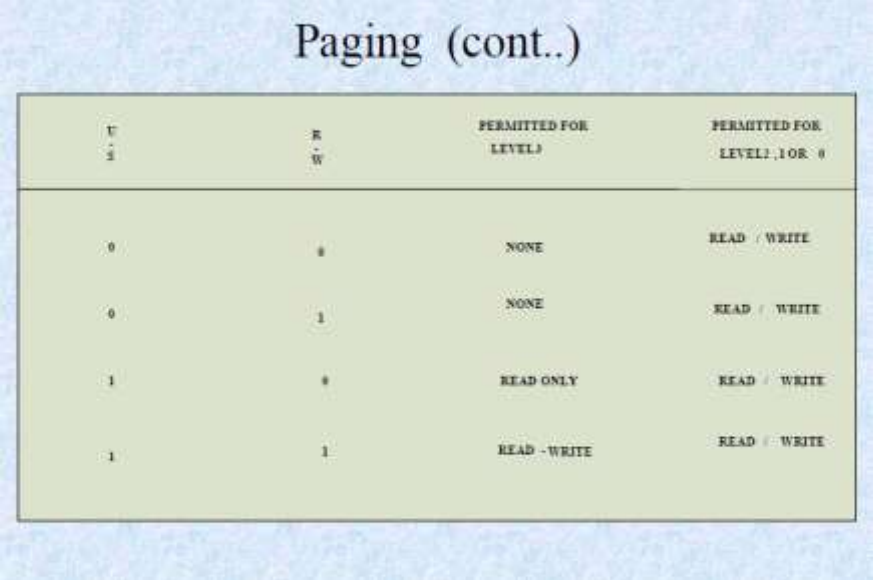
- The CR 3 is used as page directory physical base address register, to store the physical starting address of the page directory.
- The lower 12 bit of the CR3 are always zero to ensure the page size aligned directory. A move operation to CR 3 automatically loads the page table entry caches and a task switch operation, to load CR 0 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.



Paging (cont..)



Paging (cont..)

U-S	R-W	PERMITTED FOR LEVEL3	PERMITTED FOR LEVEL2 ,1 OR 0
0	0	NONE	READ / WRITE
0	1	NONE	READ / WRITE
1	0	READ ONLY	READ / WRITE
1	1	READ -WRITE	READ / WRITE

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.

- The P bit of the above entries indicates, if the entry can be used in address translation.
- If P=1, the entry can be used in address translation, otherwise it cannot be used.
- The P bit of the currently executed page is always high.
- The accessed bit A is set by 80386 before any access to the page. If A=1, the page is accessed, else unaccessed.

The D bit (Dirty bit) is set before a write operation to the page is carried out. The D-bit is undefined for page director entries.

- The OS reserved bits are defined by the operating system software.
- The User / Supervisor (U/S) bit and read/write bit are used to provide protection. These bits are decoded to provide protection under the 4 level protection model.
- The level 0 is supposed to have the highest privilege, while the level 3 is supposed to have the least privilege.
- This protection provide by the paging unit is transparent to the segmentation unit.

	2) **Ans.**	**State any four advantages of RISC processor.** **Advantages of RISC:** 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. 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.**	**4M** ***Any 4 advantages 1M each***
	3) **Ans.**	**Describe the five stage pipeline mechanism.** Five stage mechanism diagram of Pentium processor is as shown below :	**4M**

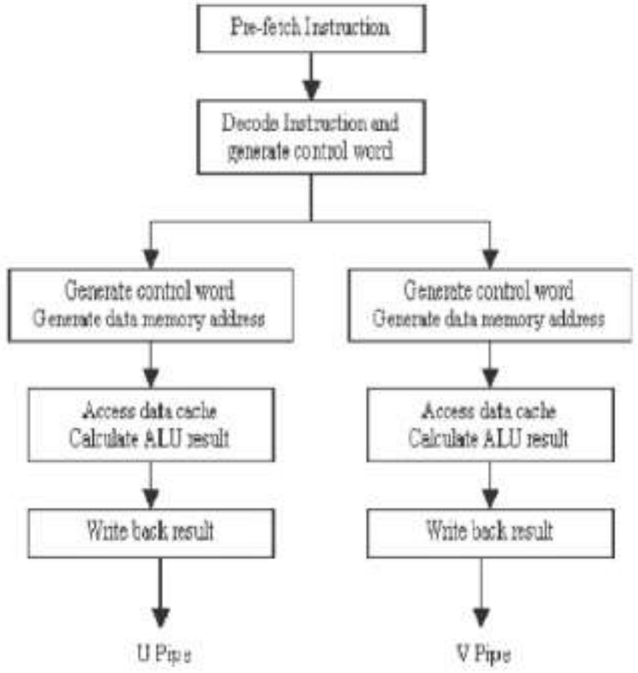


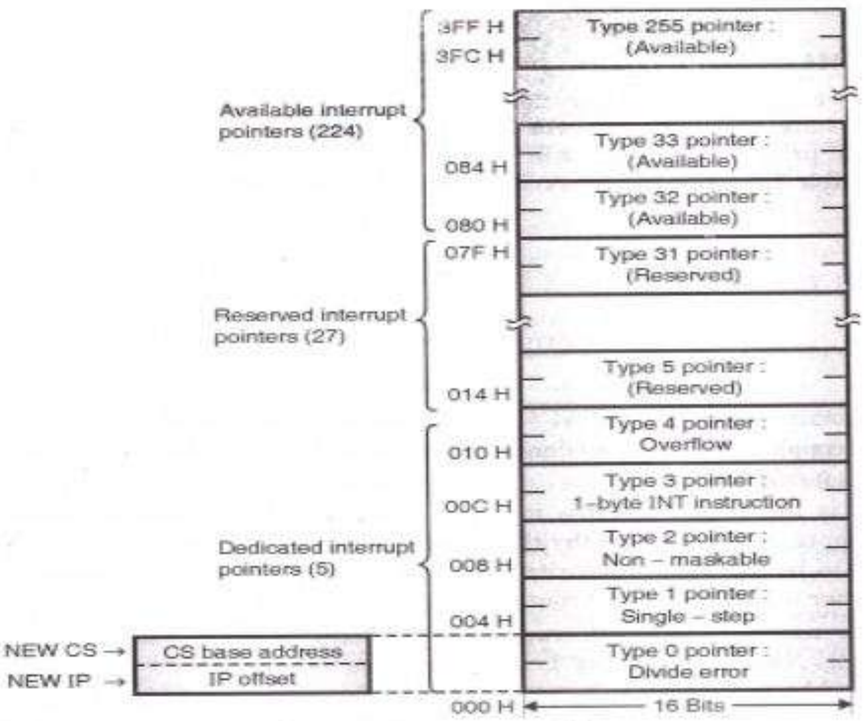
Diagram 2M

The 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 prefetch buffer to prefetch instructions sequentially, while the other prefetches according to the branch target buffer predictions. The prefetch buffers alternate their prefetch paths.

The next pipe-line stage is Decode1 (D1) in which two parallel 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 opcode map (0Fh prefix) in a single clock in either pipe-

Description 2M

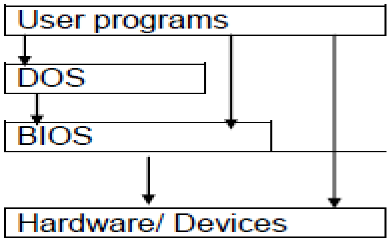
		line. The D1 stage is followed by **Decode 2 (D2)** in which the address of memory resident operands are calculated. The **Execute (EX) stage** of the pipe line for both ALU operations and for data cache access; therefore those instructions specifying both an ALU operation and a data cache access will 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 stage is **Writeback (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.	
	4) **Ans.**	**Draw and explain 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. In the IVT some of the vectors are predefined such as vector 0 as been chosen to handle divide by zero errors, vector 1 to implement single step operation, Vector 2 for NMI, Vector 3 to implement break point when troubleshooting an new program and so on. The vectors 32 to 255 are unused and are free for users. The fig shows the interrupt vector table:	**4M** ***Description 2M***

		 3FF H 3FC H Type 255 pointer : (Available) Available interrupt pointers (224) 084 H Type 33 pointer : (Available) 080 H Type 32 pointer : (Available) 07F H Type 31 pointer : (Reserved) Reserved interrupt pointers (27) 014 H Type 5 pointer : (Reserved) 010 H Type 4 pointer : Overflow 00C H Type 3 pointer : 1-byte INT instruction Dedicated interrupt pointers (5) 008 H Type 2 pointer : Non – maskable 004 H Type 1 pointer : Single – step NEW CS → CS base address NEW IP → IP offset Type 0 pointer : Divide error 000 H 16 Bits Interrupt vector table	***Interrupt vector table diagram 2M***
	5) **Ans.**	**Explain the different types of interrupt in X86.** Basically there are 2 types of interrupts available for X86 microprocessor: 1. Hardware Interrupts 2. Software interrupts 3. Conditional Interrupts **1. Hardware interrupts:** Hardware interrupts occur as the result of an external event and are classified into 2 types : maskable and Non maskable interrupts the hardware interrupts are given to the processor via external pins. **Maskable interrupts:** These are the most common way used by the X86 processor to respond to asynchronous external hardware events. Hardware interrupts occurs when the INTR is pulled high and the Interrupt flag bit is enabled. The processor only responds to interrupts between instructions. When an interrupt occurs the processor reads	**4M** ***Listing and explanation of interrupts 4M***

the 8 bit vector code of interrupt supplied by hardware which identifies the source of interrupt (one of the 224 user defined interrupts.) The IF bit in the flag register is reset when as interrupt is being serviced. This effectively disables servicing additional interrupts during an Interrupt service routine. To allow nesting of interrupts this IF bit can be set explicitly by interrupt handler. When an IRET instruction is executed the original state of IF is restored.

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.

2. Software Interrupts: These are generated directly by an executing program. These types of interrupts are also called as exceptions. INT or INTO instructions initiate interrupt processing when they are executed. Exceptions are classified as faults, traps and aborts depending on the way they are reported, and whether or not restart of the instruction causing the exception is supported.
2.1 **Faults**: these are the exceptions which are detected and serviced before the execution of faulting instructions.
A fault would occur in a virtual memory system when the processor referenced page or a segment which was not present. The OS would fetch the page or segment from disk, and then the X86 processor would restart the instruction
2.2 **Traps**: are the exceptions that are reported immediately after the execution of instruction which caused the problem. User defined interrupts are traps.
2.3 **Aborts**: these are the exceptions which do not permit the precise

		location of the instruction causing to be determined. These are used to report severe errors, such as a hardware error or illegal values in the system tables. **3. Conditional Interrupts**: Third source of interrupt is conditional interrupts, which are mainly caused due to some error condition generated in 8086 by the execution of an instruction. For example INTO-divide by zero interrupt. Program execution will automatically be interrupted if your attempt to divide an operated by zero.	
	6) **Ans.**	**Explain with neat diagram DOS-BIOS interface.** **DOS and BIOS Interface :**  Figure shows the DOS-BIOS interface. 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** ***Diagram 2M*** ***Description 2M***
3.	**1)** **Ans.**	**Attempt any four.** **Explain pipeline RISC.** **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	**16** **4M**

		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.	***Explanation 4M***
	2) **Ans.**	**Describe the virutal 8086 mode in 80386 with neat sketch of memory mapping.** **Virtual 8086 Mode:** In its protected mode of operation, 80386DX provides a virtual 8086 operating environment to execute the 8086 programs. The real mode can also used to execute the 8086 programs along with the capabilities of 80386, like protection and a few additional instructions. Once the 80386 enters the protected mode from the real mode, it cannot return back to the real mode without a reset operation. Thus, 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. 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. The 80386 supports multiprogramming, hence more than one programmer may be use the CPU at a time. 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	**4M** ***Description 2M***

4Gbytes physical memory. The virtual mode allows the multiprogramming of 8086 applications.

• The virtual 8086 mode executes all the programs at privilege level 3.Any of the other programmers may deny access to the virtual mode programs or data.

However, the real mode programs are executed at the highest privilege level, i.e. level 0.

• The virtual mode may be entered using an IRET instruction at CPL=0 or a task switch at any CPL, executing any task whose TSS is having a flag image with VM flag set to 1.

• The IRET instruction may be used to set the VM flag and consequently enter the virtual mode.

The PUSHF and POPF instructions are unable to read or set the VM bit, as they do not access it.

Even in the virtual mode, all the interrupts and exceptions are handled by the protected mode interrupt handler.

To return to the protected mode from the virtual mode, any interrupt or execution may be used.

As a part of interrupt service routine, the VM bit may be reset to zero to pull back the 80386 into protected mode.

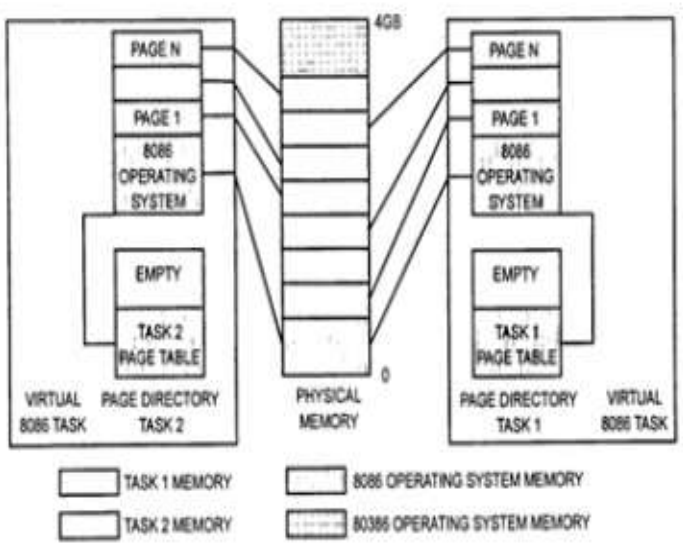
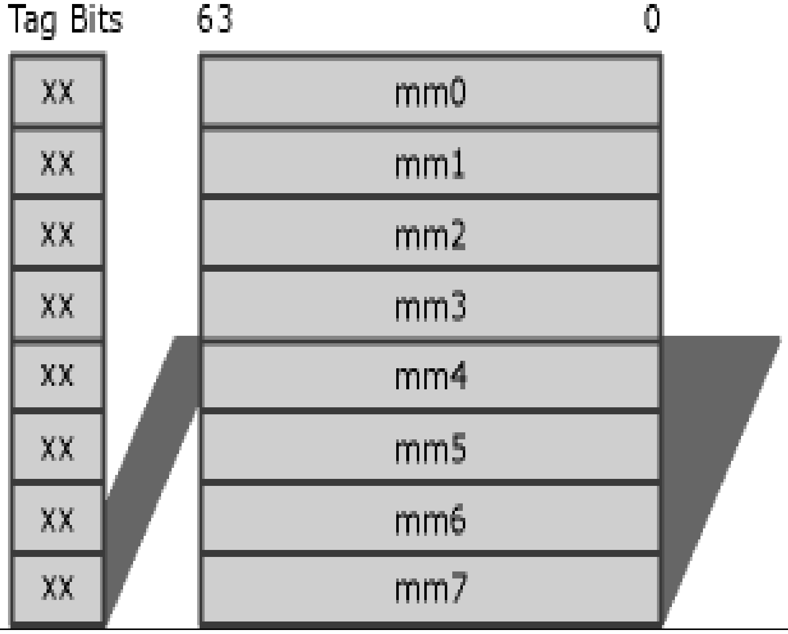


Diagram 2M

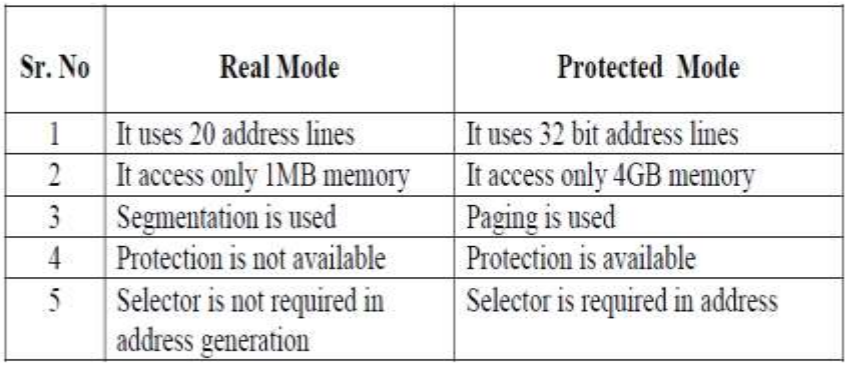
	3) **Ans.**	**Explain design issues of RISC processor.** • **Register Window:** 1. The reduced hardware requirements of RISC processors leave additional space available on the chip for the system designer. RISC CPUs generally use this space to include a large number of registers (> 100 occasionally). 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 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 alsothe 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. • **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.	**4M** ***Any two design issues 2M each***

		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. • **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. • **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.	
	4) **Ans.**	**Explain MMX architecture with register set.** 1. In Pentium there are eight general purpose floating point registers in a floating point unit.	**4M**

		2. Each of these eight registers are 80-bit wide for floating point operations, 64 bits are used for mantissa and rest of 16 bit for exponent. 3. Intel MMX instructions use these floating point registers as MMX registers and used only 64 bit mantissa portion of these registers to store MMX operands. 4. Thus MMX programmers virtually get new MMX registers each of 64bits. 5. It is possible to use same set of registers as floating point registers and MMX register in the same program; it is preferable not to use them concurrently. 6. After a sequence of MMX instruction is executed, these registers should be cleared by an instruction 'EMMS' which implies empty MMX stack. 7. The floating point users should use same instruction after executing floating point instructions. 8. Although content switching between multimedia program execution and floating point execution is permissible. It is not recommended. 9. It is advisable that multimedia program developers should partition MMX instruction into separate library routine.	***MMX registers set explanation 2M***
		 Tag Bits 63 0 XX XX XX XX XX XX XX XX mm0 mm1 mm2 mm3 mm4 mm5 mm6 mm7	***Diagram 2M***

5) Ans.

Difference between real mode and PVAM mode. **4M**



Sr. No	Real Mode	Protected Mode
1	It uses 20 address lines	It uses 32 bit address lines
2	It access only 1MB memory	It access only 4GB memory
3	Segmentation is used	Paging is used
4	Protection is not available	Protection is available
5	Selector is not required in address generation	Selector is required in address

Any 4 differences 1M Each

6) Ans.

Describe any two operational functions of DOS interrupts. **4M**

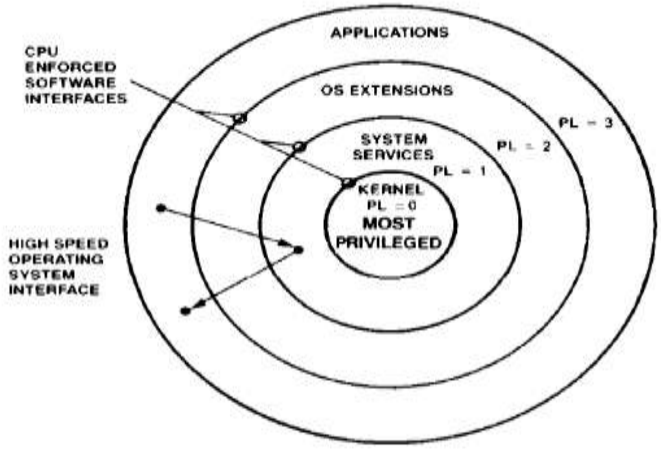
1) 3CH : to create file
This function creates a file with indicated attributes and opens the file
Registers to be used before calling the function using INT 21H:
CX=File Attribute DS: DX - full file path (zero terminated) – an ASCIIZ String file descriptor;
a start variable in data segment loaded to DX
Example: 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
Example: 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
Example: mov ah, 3Eh; function 3Eh - close a file
int 21h; transfer to DOS

Any two operational functions 2M each

		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 *Example:* mov ah,3Fh; function 3Fh – read the file int 21h; transfer to DOS	
4.	**A)** **i)** **Ans.**	**Attempt any three.** **State the feature of pentium III processor.** **Pentium III processor features:** 1.Featured with SSE instruction set (STREAMING SIMD EXTENSIONS) SSE contains 70 new instructions, most of which work on single precision floating point data. SIMD instructions can greatly increase performance when exactly the same operations are to be performed on multiple data objects. Typical applications are digital signal processing and graphics processing. 2.512KB full speed on chip L2 cache with ecc(error correcting code) for high performance. workstations/ servers, can work on windows 98, WINDOWS NT, 2000, LINUX OS. 3. PIII is also incorporated with MMX technology. 4.Dynamic execution, micro-architecture incorporates unique combination of multiple branch prediction, data flow analysis and speculative execution. 5.Supports power management capabilities like System management mode and Multiple low power states. 6.PIII is optimized for 32 bits applications running on advanced 32 bits os. 7.It has 32KB L1 cache divided as 16KB instruction cache and 16KB data cache 8.Quad quad word wide ie. 256 bits cache data bus, ways set associative cache provides improved cache hit rate. 9. It supports Multiprocessor system. 10. It Works on 1.0ghz,850,800,750,700,650 MHZ.	**12** **4M** ***Any 4 features of Pentium III processor 1M each***
	ii) **Ans.**	**Describe four level protection in 80386.** 80386 DX has four levels of protection which isolate and protect user programs from each other and the operating system. •It offers an additional type of protection on a page basis, when	**4M** ***Description 2M***

		paging is enabled(using U/S and R/W fields) •The four level hierarchical privilege system is illustrated as follows:  CPU ENFORCED SOFTWARE INTERFACES APPLICATIONS OS EXTENSIONS SYSTEM SERVICES PL = 3 PL = 2 PL = 1 KERNEL PL = 0 MOST PRIVILEGED HIGH SPEED OPERATING SYSTEM INTERFACE • The privilege levels (PL) are numbered 0 through 3. Level 0 is the most privileged or trusted level. Level 3 is the least privileged level.	***Four level protection diagram 2M***
	iii) **Ans.**	**Define maskable and non-maskable interrupt of X86.** **Maskable interrupts:** These are the most common way used by the X86 processor to respond to asynchronous external hardware events. Hardware interrupts occurs when the INTR is pulled high and the Interrupt flag bit is enabled. The processor only responds to interrupts between instructions. When an interrupt occurs the processor reads the 8 bit vector code of interrupt supplied by hardware which identifies the source of interrupt (one of the 224 user defined interrupts.) The IF bit in the flag register is reset when as interrupt is being serviced. This effectively disables servicing additional interrupts during an Interrupt service routine. To allow nesting of interrupts this IF bit can be set explicitly by interrupt handler. When an IRET instruction is executed the original state of IF is restored. **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	**4M** ***Maksable interrupt 2M*** ***Non-maskable interrupts 2M***

		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.	
	iv) **Ans.**	**Describe any two dedicated interrupts.** **1. Divide by zero:** This interrupt is caused by the instructions such as DIV or IDIV. Type 0 interrupt is generated when such divide by zero error occurs in the system. When the ISR for this interrupt executes it expects the user to get the divide by zero error corrected. Since it is type 0, its vector address is 00000H to 00003H. 2. **Single step**: When a trap flag in the flag register is set, the processor generates a type 1 interrupt after the execution of every instruction. This interrupt is used for debugging a newly written program. This interrupt causes the display of the contents of flag register and other registers for the user. The ISR vector address of the single step interrupt is 00004H to 00007H. **3. NMI:** NMI is a non-maskable interrupt which is compulsorily serviced by the processor. It is always executed under the catastrophic (unavoidable) circumstances. One of such event is disastrous power failure. The processor tends to reset abnormally with the occurrence of this interrupt and so does not store the contents of flags and registers anywhere. In the event of such a power failure, NMI ISR should store the contents of each processor register in the NVRAM. These values can be reloaded when the power comes back. Type of this interrupt is 2 and hence is stored from the location 00008H to 0000BH. **4. Breakpoint:** This is a type 3 interrupt. It is used for debugging purpose. A program being debugged will have the first byte of one of its instructions replaced by the code for breakpoint. When the processor gets this instruction, then processor generates type 2 interrupt. The ISR associated with breakpoint is similar to trap ISR and should be	**4M** ***Any two dedicated interrupts explanation 2M each***

		capable of displaying the contents of processor registers and also the address at which the breakpoint occurred. Before the ISR exits, it will replace the breakpoint byte with the original first byte of the instruction. The ISR for this breakpoint is stored in memory locations on 0000CH to 0000FH. **5. Overflow:** This is a type 4 interrupt which is caused when INTO instruction is executed with the overflow flag set. The ISR vector address for overflow is stored in memory locations from 00010H to 00013H. **6. INTR:** When the INTR is made high, it causes the processor to perform two INTA# cycles. The first low going pulse is used to indicate to other device that the processor is beginning with the INTA# cycle. The second low going pulse indicates that the interrupt number should be placed on the lower byte of the processors data bus. The8259 PIC is used to respond to the 8086's interrupt acknowledge cycle. The INTR type may be from 00 to FFH. The INTR can be masked by using the IF flag.(Interrupt enable flag).	
4.	**B)** **i)** **Ans.**	**Attempt any one.** **Draw the MSW of 80386 and describe function of each in detail.** • 80386 has four control registers, CR0, CR1, CR2 and CR3, which are 32 bits each. • They are used to hold the machine status of a global system. • To access the control registers load and store instructions are available. • LMSW and SMSW instructions are used to access the CR0.	**6** **6M**

- The lower order 16 bits of CR0 are also called as Machine Status Word.

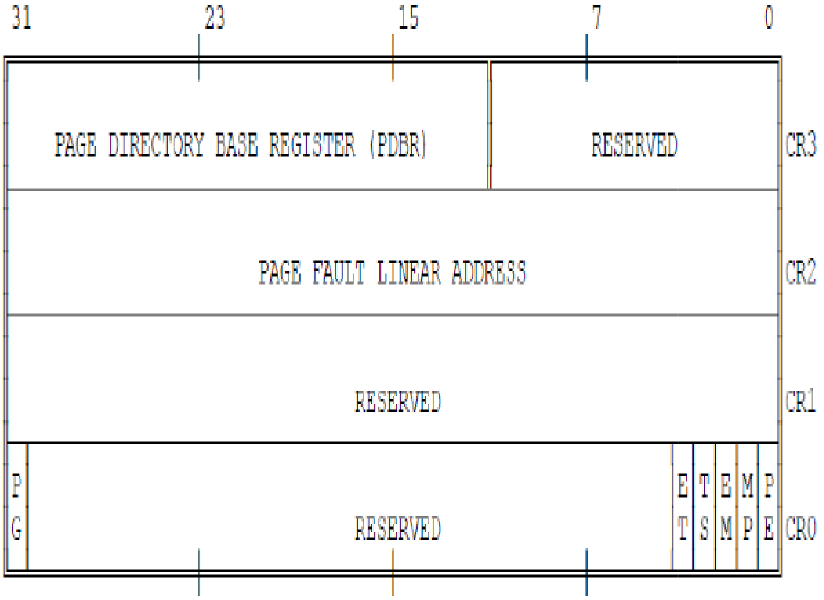

31
23
15
7
0
PAGE DIRECTORY BASE REGISTER (PDBR)
RESERVED
CR3
PAGE FAULT LINEAR ADDRESS
CR2
RESERVED
CR1
PG
RESERVED
ET
TS
EM
MP
PE
CR0

Diagram of MSW 3M

- **CR0** 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.

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

Description 3M

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.

- **CR2 : control register 2:**

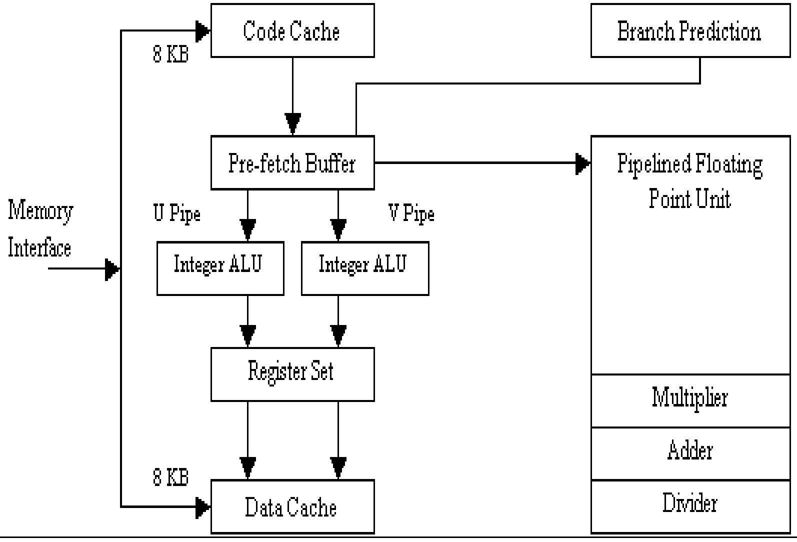
it is used for handling page faults when PG is set(i.e. paging is enabled). The processor stores the linear address that triggers the fault (i.e. page fault linear address).

- **CR3** is used when PG is set. CR3 enables the processor to locate the page table directory for the current task.

ii) Ans.

State diagram of branch prediction logic. **6M**

The diagram shows the branch prediction logic in Pentium processor:



Correct diagram 6M

5. **Attempt any four.** **16**

1) **Explain memory organization with neat diagram of address translation.** **4M**

Ans. In a virtual memory system, the program memory is divided into fixed sized pages and allocated in fixed sized physical memory frames. The pages do not have to be contiguous in memory. A page table keeps track of where each page is located in physical memory. This allows the operating system to load a program of any size into any available frames. Only the currently used pages need to be loaded. Unused pages can remain on disk until they are referenced. This allows many large programs to be executed on a relatively small memory system. A resident flag in the page table indicates whether or not the page is in memory. The page table also includes several other flags to keep track of memory usage. A use flag is set whenever the page is referenced. A dirty bit is set whenever the page is changed to inform the operating system that the page in memory is different than the page on disk.

Description 2M

There are several virtual memory parameters set:

Maximum Virtual Address space	The size of a program address is determined by the maximum size of the virtual address space. The number of bits in a virtual address is the log base 2 of this value.
Maximum Physical Address space	The amount of real memory that the system can support determined the number of bits needed to address the physical memory. The size of a physical address is log base 2 of this value.
Size of a page	This is the size of a virtual memory page and a physical memory frame. It is always a power of 2.

The addresses that appear in programs are the virtual addresses or program addresses. For every memory access, either to fetch an instruction or data, the CPU must translate the virtual address to a real physical address. A virtual memory address can be considered to be composed of two parts: a page number and an offset into the page.

The page number determines which page contains the information and the offset specifies which byte within the page. The size of the offset field is the log base 2 of the size of a page.

Consider an example system with:

16MB Maximum Virtual Address space (24 bits)

8MB Maximum Physical Address space (23 bits)

1024byte Page size (10 bits)

The virtual addresses can be represented as

13 bits	10 bits
page number	offset

To convert a virtual address into a physical address, the CPU uses the page number as an index into the page table. If the page is resident, the physical frame address in the page table is concatenated in front of the offset to create the physical address.

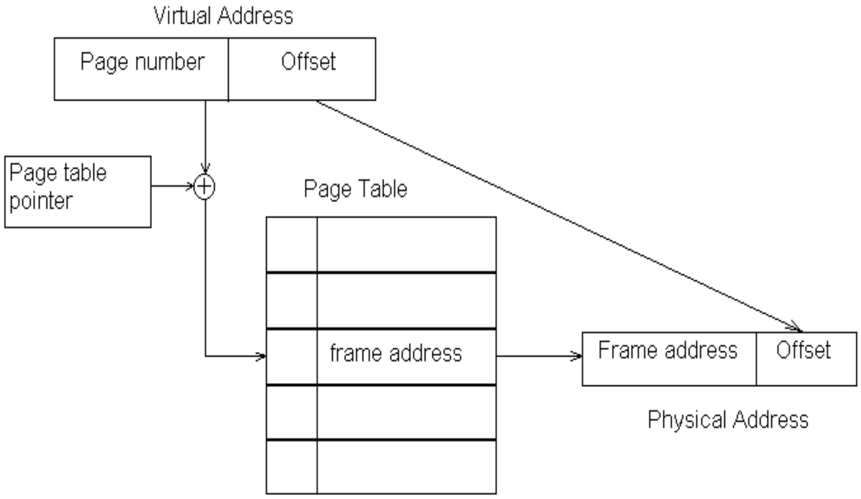


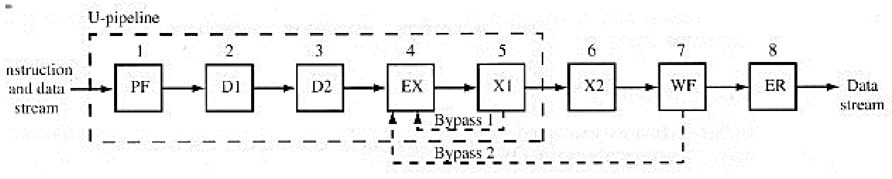
Diagram 2M

2) Ans. **State any four features of SUN-Ultra SPARC.** **4M**

The 64 bits Ultra SPARC architecture has following features:

1. It has 14 stages non-stalling pipeline

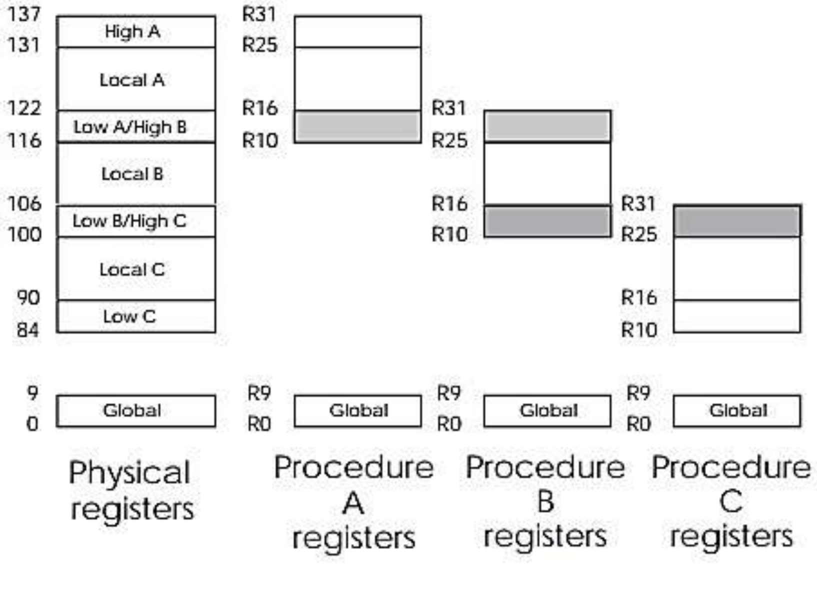
2. It has 6 execution units including two for integer, two for floating

		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 ata 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	***Any 4 features 1M each***
	3) **Ans.**	**Describe the eight stage pipeline mechanism in floating point unit of pentium processor.** The floating point pipeline has 8 stages as follows:  U-pipeline 1 2 3 4 5 6 7 8 nstruction and data stream PF D1 D2 EX X1 X2 WF ER Data stream Bypass 1 Bypass 2 **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 checks whether the instructions can be paired – 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 **4.Memory and Register Read (Execution Stage) (EX):** - **Register read or memory read** 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	**4M** ***Diagram 1M*** ***Description 3M***

MODEL ANSWER

WINTER - 2017 EXAMINATION

Subject: Advanced Microprocessor **Subject Code: 17627**

		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**	
	4) **Ans.**	**Explain register windowing in RISC processor.** **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.  137 131 High A Local A 122 116 Low A/High B Local B 106 100 Low B/High C Local C 90 84 Low C 9 0 Global R31 R25 R16 R10 R9 R0 Physical registers Procedure A registers Procedure B registers Procedure C registers *Register windowing* 3. The RISC processor may not be able to access all the registers it	**4M** ***Description 2M*** ***Diagram 2M***

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 subset of the registers are accessible at any specific time.
5. To understand how register windows work, 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 overlaps. 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.
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.

5)
Ans.

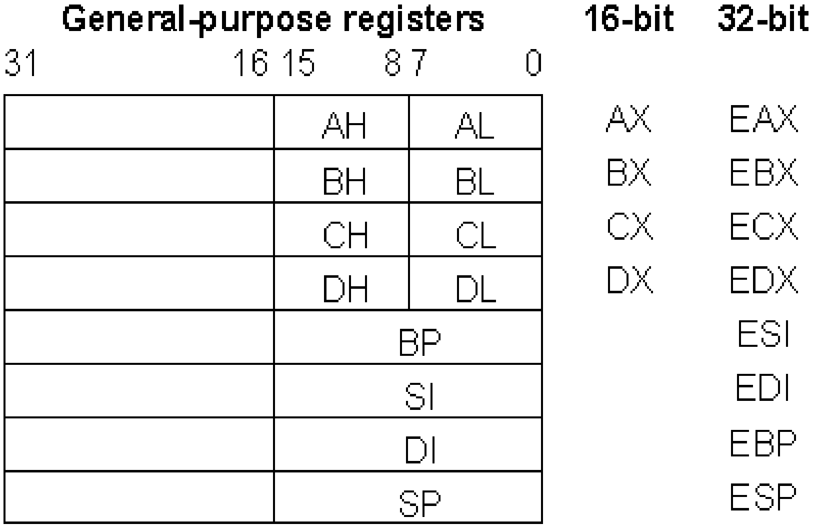
Describe the general purpose register of pentium. **4M**

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).

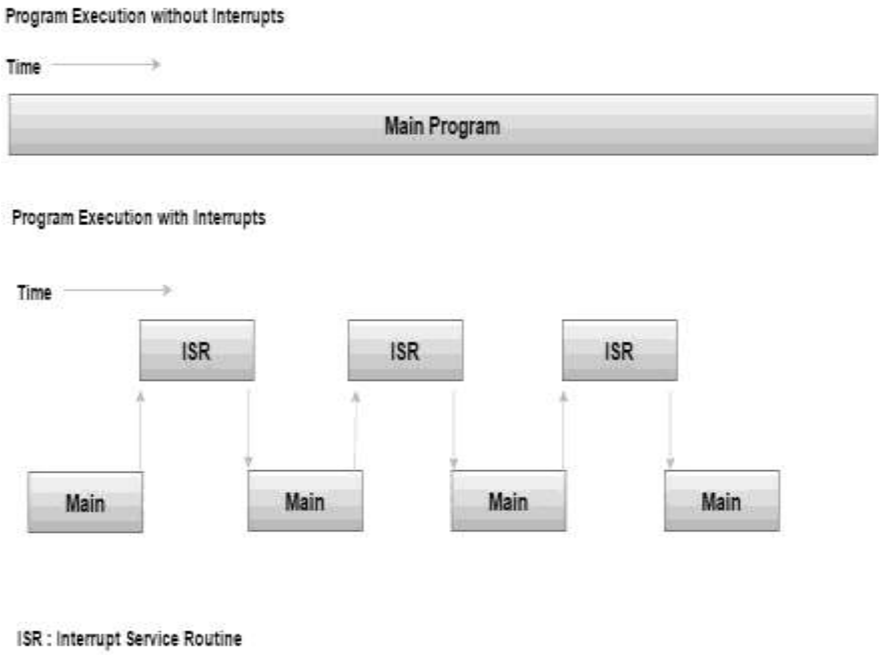
Description 2M

		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).  General-purpose registers 16-bit 32-bit 31 16 15 8 7 0 AH AL BH BL CH CL DH DL BP SI DI SP AX BX CX DX EAX EBX ECX EDX ESI EDI EBP ESP	***Diagram 2M***
	6) **Ans.**	**Describe interrupt services.** Interrupt means event, which invites attention of the processor on occurrence of some action at hardware or software interrupt instruction event. **Hardware Interrupt** A hardware interrupt is an electronic alerting signal sent to the processor from an external device, like a disk controller or an external peripheral. For example, when we press a key on the keyboard or move the mouse, they trigger hardware interrupts which cause the processor to read the keystroke or mouse position. **Software Interrupt** A software interrupt is caused either by an exceptional condition or a special instruction in the instruction set which causes an interrupt when it is executed by the processor. For example, if the processor's	**4M** ***Description 4M***

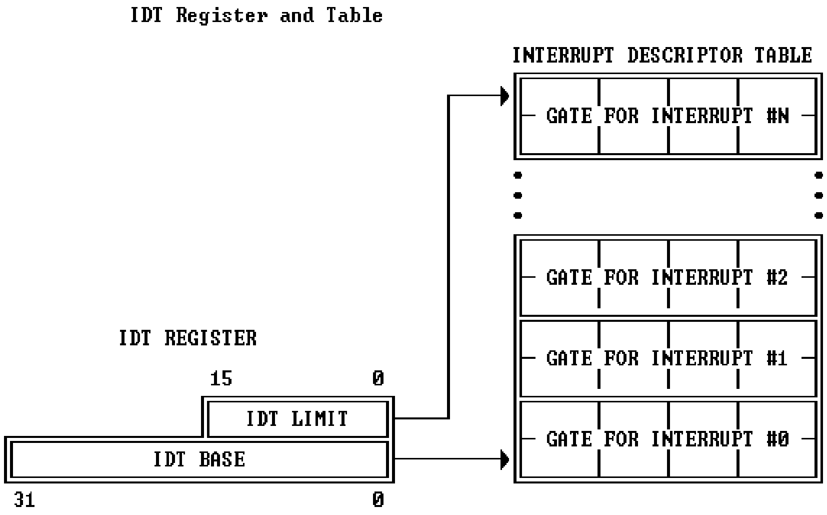
arithmetic logic unit runs a command to divide a number by zero, to cause a divide-by-zero exception, thus causing the computer to abandon the calculation or display an error message. Software interrupt instructions work similar to subroutine calls.

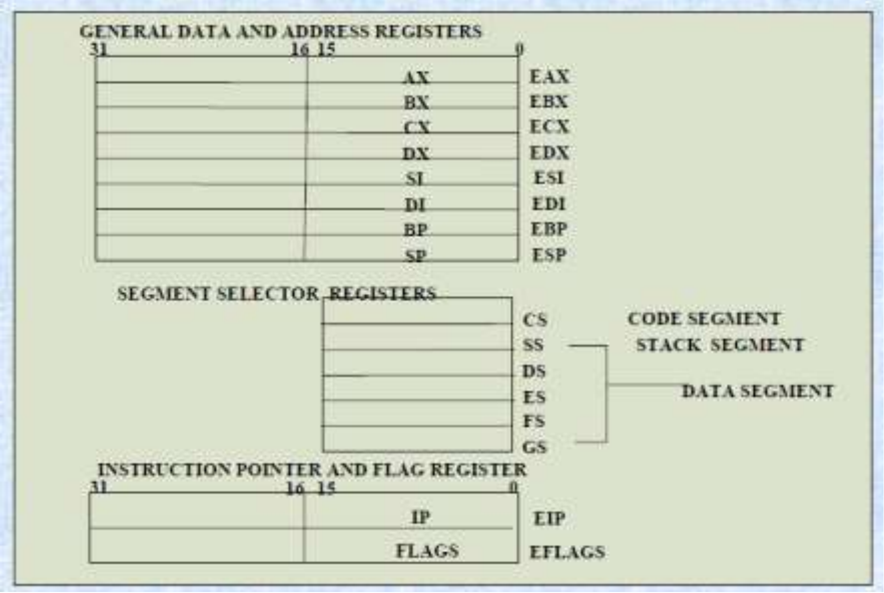
For every interrupt, there must be an interrupt service routine (ISR), or **interrupt handler**. When an interrupt occurs, the microcontroller runs the interrupt service routine. For every interrupt, there is a fixed location in memory that holds the address of its interrupt service routine, ISR. The table of memory locations set aside to hold the addresses of ISRs is called as the Interrupt Vector Table
In response to the interrupt, the routine or program, which is running presently interrupts and an interrupt service routine (ISR) executes.

Processor executes the program, called interrupt service routine or signal handler or trap handler or exception handler or device driver, related to input or output from the port or device or related to a device function on an interrupt and does not wait and look for the input ready or output completion or device-status ready or set.


Program Execution without Interrupts
Time
Main Program
Program Execution with Interrupts
Time
ISR
ISR
ISR
Main
Main
Main
Main
ISR : Interrupt Service Routine

6.	**1)** **Ans.**	**Attempt any four.** **List any four file handling function of INT 21H. Describe the functions with their syntax and usages.** **INT 21H:** DOS provides INT 21h, is called the DOS function dispatcher and supports functions such as: read from the keyboard, write to the screen, write to the printer, read and write to disk files, etc. INT 21h must be told which function is being requested. *Eg.* • **Function 01h** – used to read the character from standard input device. • **INT 21h Functions 02h and 06h:** Write Character to Standard Output Write the letter 'A' to standard output: mov ah,02h mov dl,’A’ int 21h • **INT 21h Function 40h:** Write a block of data(array of byts) to a File or Device Input: BX = file or device handle (console = 1), CX= number of bytes to write, DS:DX = address of array Returns : AX = number of bytes written • **INT 21h Function 3Fh:** Read from file or device -Reads a block of bytes. -Can be interrupted by Ctrl-Break (^C)	**16** **4M** ***Each function 1M***
	2) **Ans.**	**Draw and Describe interrupt descriptor table of 80386.** The **Interrupt Descriptor Table** (**IDT**) is a data structure used by the x86 architecture to implement an interrupt vector table. The IDT is used by the processor to determine the correct response to interrupts and exceptions. Use of the IDT is triggered by three types of events: hardware interrupts, software interrupts, and processor exceptions, which together are referred to as "interrupts". The IDT consists of 256 interrupt vectors–the first 32 (0-31 or 00-1F) of which are reserved for processor exception	**4M** ***Description 2M***

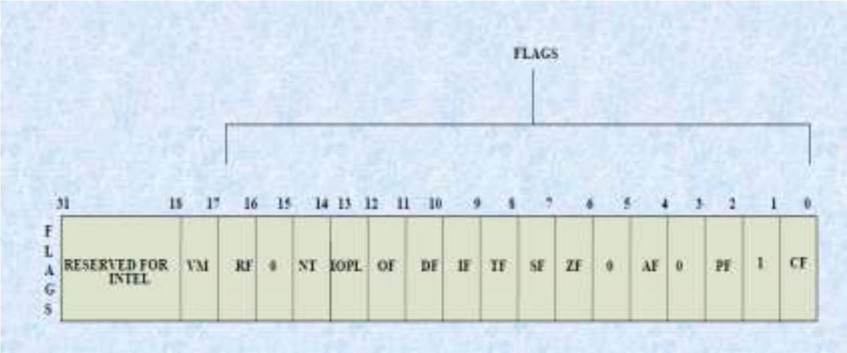
		The interrupt descriptor table (IDT) associates each interrupt or exception identifier with a descriptor for the instructions that service the associated event. Like the GDT and LDTs, the IDT is an array of 8-byte descriptors. Unlike the GDT and LDTs, the first entry of the IDT may contain a descriptor. To form an index into the IDT, the processor multiplies the interrupt or exception identifier by eight. Because there are only 256 identifiers, the IDT need not contain more than 256 descriptors. It can contain fewer than 256 entries; entries are required only for interrupt identifiers that are actually used.  IDT Register and Table INTERRUPT DESCRIPTOR TABLE GATE FOR INTERRUPT #N GATE FOR INTERRUPT #2 GATE FOR INTERRUPT #1 GATE FOR INTERRUPT #0 IDT REGISTER 15 0 IDT LIMIT IDT BASE 31 0	***Diagram 2M***
	3) **Ans.**	**State the instruction latency in RISC processor designing.** A poorly designed instruction set can cause a pipelined processor to stall frequently 1) Highly encoded instructions, such as those used in CISC machine need complex decoders, they should be avoided. 2) Variable length instructions require multiple references to memory to fetch the entire instruction should not be considered for inclusion. 3) Instruction which access main memory, instead of register are slow in execution since main memory is comparatively slow. 4) 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	**4M** ***Relevant description 4M***

		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.) 5) 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) **Ans.**	**Explain the register organization of 80386 microprocessor.** The 80386 has eight 32 - bit general purpose registers which may be used as either 8 bit or 16 bit registers. • A 32 - bit register known as an extended register, is represented by the register name with prefix E. • Example : A 32 bit register corresponding to AX is EAX, similarly BX is EBX etc. • The 16 bit registers BP, SP, SI and DI in 8086 are now available with their extended size of 32 bit and are names as EBP,ESP,ESI and EDI. • AX represents the lower 16 bit of the 32 bit register EAX. • BP, SP, SI, DI represents the lower 16 bit of their 32 bit counterparts, and can be used as independent 16 bit registers.  GENERAL DATA AND ADDRESS REGISTERS 31 16 15 0 AX BX CX DX SI DI BP SP EAX EBX ECX EDX ESI EDI EBP ESP SEGMENT SELECTOR REGISTERS CS SS DS ES FS GS CODE SEGMENT STACK SEGMENT DATA SEGMENT INSTRUCTION POINTER AND FLAG REGISTER 31 16 15 0 IP FLAGS EIP EFLAGS	**4M** ***Description 2M*** ***Diagram 2M***

The six segment registers available in 80386 are CS, SS, DS, ES, FS and GS.

- The CS and SS are the code and the stack segment registers respectively, while DS, ES, FS, GS are 4 data segment registers.
- A 16 bit instruction pointer IP is available along with 32 bit counterpart EIP.

• ***Flag Register of 80386***: The Flag register of 80386 is a 32 bit register. Out of the 32 bits, Intel has reserved bits D18 to D31, D5 and D3, while D1 is always set at 1.Two extra new flags are added to the 80286 flag to derive the flag register of 80386.They are VM and RF flags.



VM - *Virtual Mode Flag*: If this flag is set, the 80386 enters the virtual 8086 mode within the protection mode. This is to be set only when the 80386 is in protected mode.

RF- Resume Flag: This flag is used with the debug register breakpoints. It is checked at the starting of every instruction cycle.

Segment Descriptor Registers: This registers are not available for programmers, rather they are internally used to store the descriptor information, like attributes, limit and base addresses of segments.

Control Registers: The 80386 has three 32 bit control registers CR1, CR2 and CR3 to hold global machine status independent of the executed task. Load and store instructions are available to access these registers.

• ***System Address Registers***: Four special registers are defined to refer to the descriptor tables supported by 80386.

The 80386 supports four types of descriptor table, viz. Global descriptor table (GDT), interrupt descriptor table (IDT), local descriptor table (LDT) and task state segment descriptor (TSS).

<table>
<tr><td></td><td></td><td><b><i>Debug and Test Registers</i></b>: Intel has provide a set of 8 debug registers for hardware debugging. Out of these eight registers DR0 to DR7, two registers DR4 and DR5 are Intel reserved.</td><td></td></tr>
<tr><td></td><td><b>5)</b>
<b>Ans.</b></td><td><b>Differentiate between .COM and .EXE programs.</b>
<table>
<tr><th>.COM programs</th><th>.EXE programs</th></tr>
<tr><td>.COM file does not contain any header</td><td>.EXE file contains header</td></tr>
<tr><td>.COM file cannot contain relocation items.</td><td>.EXE file may contain relocation items.</td></tr>
<tr><td>Maximum size is 64k minus 256 bytes.For PSP and 2 bytes for stack</td><td>No limit on size; Can be of any size</td></tr>
<tr><td>Size of file is exact size of program.</td><td>Size of file is size of program plus header
(Multiple of 256 bytes</td></tr>
<tr><td>Stack size is 64K minus 256 bytes for PSP and size of executable data and code.</td><td>Stack size is defined in a program with STACK directive.</td></tr>
<tr><td>Entry point is PSP:0100</td><td>Entry point is defined by END directive.</td></tr>
</table></td><td><b>4M</b>
<b><i>Any 4 differen ce 1M each</i></b></td></tr>
<tr><td></td><td><b>6)</b>
<b>Ans.</b></td><td><b>Write the advantages of separate code and data cache available in Pentium.</b>
<b>Advantages of separate instruction and data caches:</b>
1. Separate code and data cache memories effectively and efficiently execute 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.</td><td><b>4M</b>
<b><i>Any 4 advanta ges 1M Each</i></b></td></tr>
</table>