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

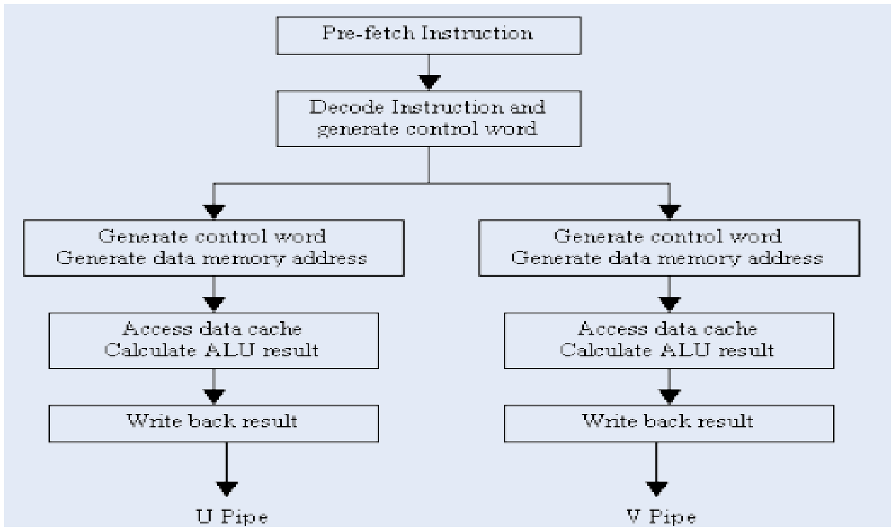
MODEL ANSWER

SUMMER – 2018 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 of the following:** **List any four salient features of 80386.** 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	**(3x4=12)** **4M** ***Any 4 features 1M each***

		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 DR0-DR7 for hardware debugging and control. 14. It has a 32 bit E-flag register. 15. It supports the dynamic bus sizing by which the 80386 can be interfaced to 16 bits devices effectively. And also supports the 8bits, 16 bits and 32 bits operands. 16. It operates on 20 MHz and 33 MHz frequency.	
	ii) **Ans.**	**Explain super scalar execution stages of Pentium with neat diagram.** ***(Note: Relevant topic's shorter description shall be considered)***  Pre-fetch Instruction Decode Instruction and generate control word Generate control word Generate data memory address Generate control word Generate data memory address Access data cache Calculate ALU result Access data cache Calculate ALU result Write back result Write back result U Pipe V Pipe The superscalar execution stages of Pentium are as shown in the fig above. There are total 5 stages in this. **The first stage of the pipe-line is Prefetch (PF)** stage in which instructions a re-prefetched from the on chip instruction cache or memory. Because the Pentium processor has separate caches for	**4M** ***Diagram 2M*** ***Description 2M***

		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-line. The D1 stage is followed by **Decode 2 (D2)** in which the address of memory resident operands is 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.	
	iii) **Ans.**	**State any four advantages of RISC processor.** ***(Note: Relevant topic's shorter description shall be considered)*** **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. 3. Several extra functionalities such as MMUs, floating point	**4M** ***Any four advantag ed 1M each***

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.

iv) **State any four differences between .com and .exe program.** **4M**

Ans.

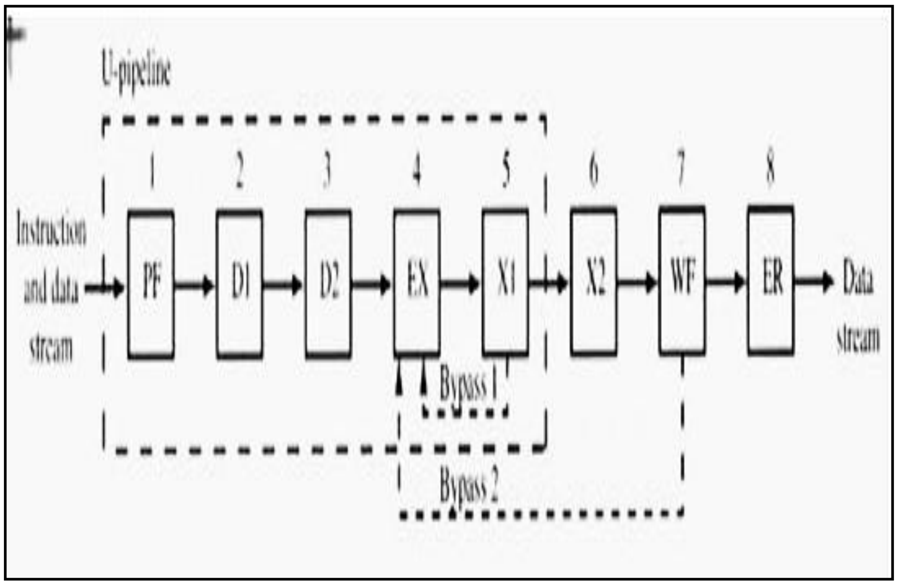
Sr. No	.COM programs	.EXE Programs
1.	.COM file does not contain any header	.EXE file contains header
2.	.COM file cannot contain relocation items.	.EXE file may contain relocation items.
3.	Maximum size is 64k minus 256 bytes. For PSP and 2 bytes for stack.	No limit on size; Can be of any size
4.	Entry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

Any four differences 1M each

MODEL ANSWER

SUMMER – 2018 EXAMINATION

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

1.	**b)** **i)** **Ans.**	**Attempt any one of the following:** **Describe the eight stage pipeline mechanism in floating point unit of Pentium processor.**  U-pipeline 1 2 3 4 5 6 7 8 Instruction and data stream PF D1 D2 EX X1 X2 WF ER Data stream Bypass 1 Bypass 2 **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 **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.	**(1x6=6)** **6M** ***Diagram 3M*** ***Stages and description 3M***

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.

ii) **Draw and explain internal architecture of 80386.** **6M**

(Note: Relevant topic's shorter description shall be considered)

Ans.

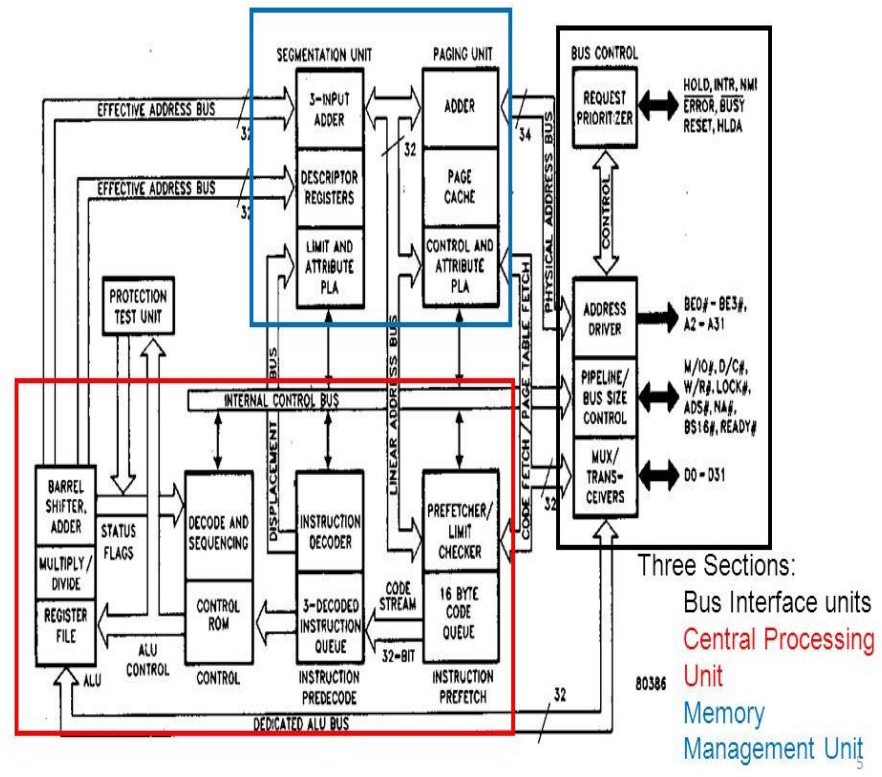


Fig. Architecture of 80386

Diagram of internal architecture of 80386 3M

		The internal architecture of 80386 can be divided into 3 sections as shown in the above figure such as : 1. Central processing unit (CPU) 2. Memory management unit(MMU) 3. Bus interface unit(BIU) **1. The Central processing unit** consists of a. Execution unit & b. Instruction unit **a. Instruction unit has Instruction prefetcher and instruction pre-decode unit :** The Instruction prefetcher fetches the 16 instruction bytes ahead of time and stores them into the 16 byte instruction prefetch 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 queues. **b. 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. **2. The memory management unit** It has segmentation unit and paging unit. **a. segmentation unit** : • The segmentation unit allows the use of two address components such as segment base address and offset address to calculate the physical address.	***Description 3M***

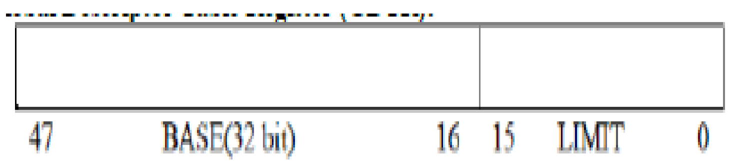
- It allows the size of the segment upto 4GB maximum.
- It provides the 4 level protection level mechanisms for protecting and isolating the system's code and data from application programs and unauthorized access.
- This unit converts logical address spaces to the linear addresses.
- The Limit and Attribute PLA checks the segment limits and attributes at segment level to avoid invalid access to the code.

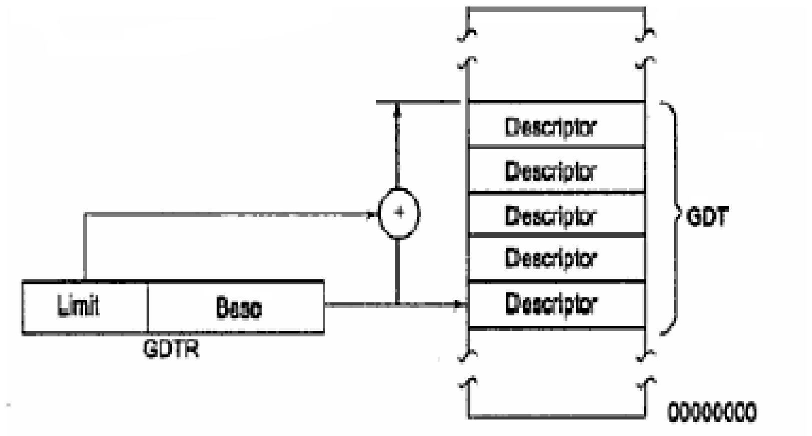
b. Paging unit:

- 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 maintains 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.

3. Bus Interface Unit :

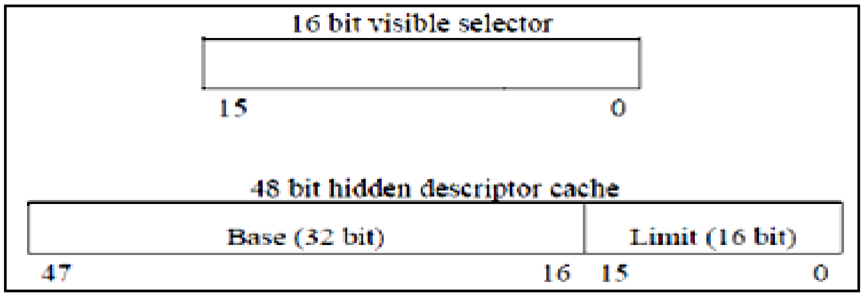
- BIU is responsible for interfacing the microprocessor with the system with the help of all buses.
- Control of all buses 1.e. address bus, data bus, control bus is in the hands of BIU.
- The BIU has a bus control unit which has a request prioritizer which resolves the priorities of the various bus request operations. It also controls the access of the bus.
- The address drivers drives the bus(byte) enable signals BE0#-BE3# and the address signals A0-A31.
- The pipeline and bus size control unit handle the related control signals and supports the dynamic bus sizing feature. The pipeline unit supports the instruction pipelining feature in 80386. Fetching other instruction while the other is in execution is known as instruction pipelining.
- The data buffers (mux / transceivers) interface the internal

		data bus with the system data bus. **4. Protection Test Unit:** This unit performs the built in self test for the 80386 microprocessor along with the other tests.	
2.	**a)** **Ans.**	**Attempt any FOUR of the following:** **Explain any two system address registers of micro-processor 80386 with neat diagram.** ***(Note: Relevant topic's shorter description can be considered)*** **Systems Address Registers:** **Memory-Management Registers** Four registers of the 80386 locate the data structures that control segmented memory management: • **GDTR** (Global Descriptor Table Register), • **LDTR** (Local Descriptor Table Register): These registers point to the segment descriptor tables GDT and LDT. • **IDTR** (Interrupt Descriptor Table Register): This register points to a table of entry points for interrupt handlers (the IDT). • **TR** (Task Register): This register points to the information needed by the processor to define the current task. **Global Descriptor Table Register (GDTR):** 	**(4x4=16)** **4M** ***Any 2 System address registers, diagram and explanation 2M each***



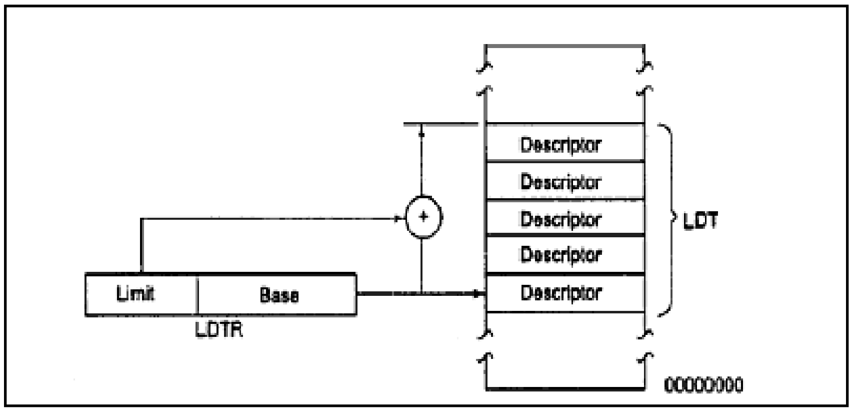
The Global Descriptor Table Register (GDTR) is a dedicated 48-bit (6 byte) register used to record the base and size of a system's global descriptor table (GDT). Thus, two of these bytes define the size of the GDT, and four bytes define its base address in physical memory. LIMIT is the size of the GDT, and BASE is the starting address. LIMIT is 1 less than the length of the table, so if LIMIT has the value 15, then the GDT is 16 bytes long. To load the GDTR, the instruction LGDT is used.

Local Descriptor Table Register (LDTR):



The visible component of the LDTR is a 16-bit "selector" field. The format of these 16 bits is same as a segment selector in a virtual address pointer. Thus, it contains a 13-bit INDEX field, a 1-bit TI field, and a 2-bit RPL field. The TI "table indicator" bit must be zero, indicating a reference to the GDT (i.e., to global address space). The

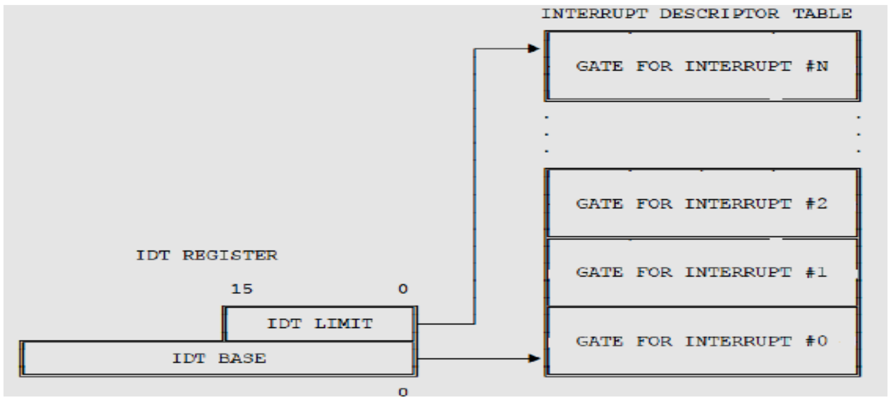
INDEX field consequently provides an index to a particular entry within the GDT. This entry, in turn, must be an LDT descriptor. In this way, the visible "selector" field of the LDTR, by selecting an LDT descriptor, uniquely designates a particular LDT in the system. The dedicated, protected instructions LLDT and SLDT are reserved for loading and storing, respectively, the visible selector component of the LDTR register.



The Local Descriptor Table Register (LDTR) is a dedicated 48-bit register that contains, at any given moment, the base and size of the local descriptor table (LDT) associated with the currently executing task. Unlike GDTR, the LDTR register contains both a "visible" and a "hidden" component. Only the visible component is accessible, while the hidden component remains truly inaccessible to application programs.

Interrupt Descriptor Table
The IDT may reside anywhere in physical memory. The processor locates the IDT by means of the IDT register (IDTR). IDT register contains a 32-bit base and a 16-bit limit. The instructions LIDT and SIDT operate on the IDTR. Both instructions have one explicit operand: the address in memory of a 6-byte area. Figure below shows the format of this area. LIDT (Load IDT register) loads the IDT register with the linear base address and limit values contained in the memory operand. This instruction can be executed only when the CPL is zero. SIDT (Store IDT register) copies the base and limit

value stored in IDTR to a memory location. This instruction can be executed at any privilege level.



The task register (TR) identifies the currently executing task by pointing to the TSS. The task register has both a "visible" portion (i.e., can be read and changed by instructions) and an "invisible" portion (maintained by the processor to correspond to the visible portion; cannot be read by any instruction). The selector in the visible portion selects a TSS descriptor in the GDT. The processor uses the invisible portion to cache the base and limit values from the TSS descriptor. Holding the base and limit in a cache register makes execution of the task more efficient, because the processor does not need to repeatedly fetch these values from memory when it references the TSS of the current task.

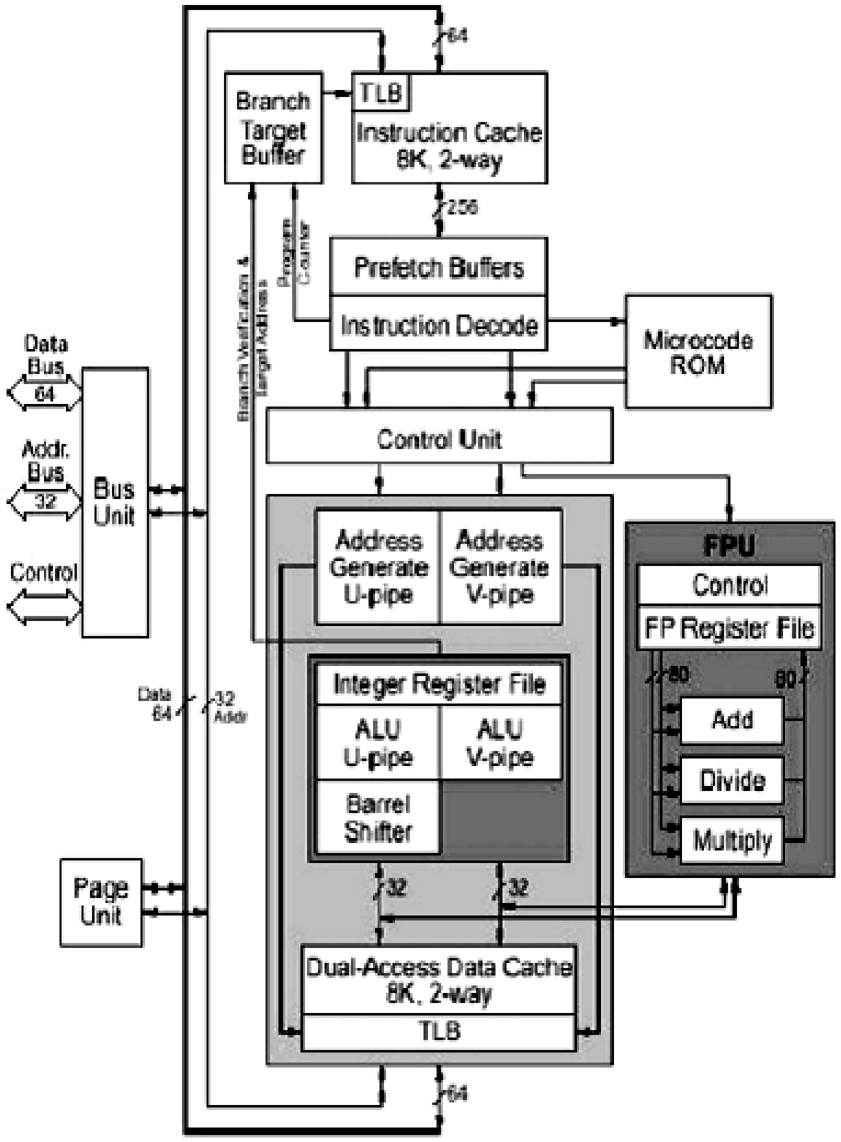
The instructions LTR and STR are used to modify and read the visible portion of the task register. Both instructions take one operand, a 16-bit selector located in memory or in a general register.

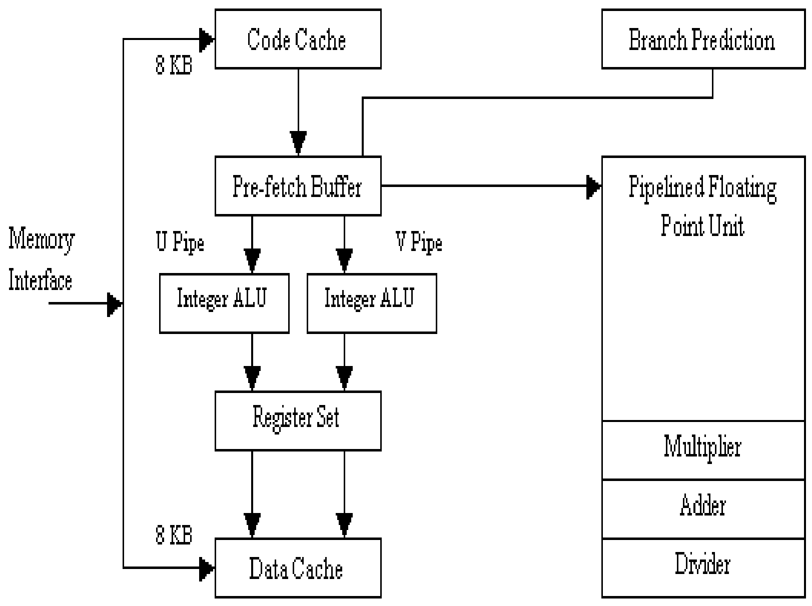
LTR (Load task register) loads the visible portion of the task register with the selector operand, which must select a TSS descriptor in the GDT. LTR also loads the invisible portion with information from the TSS descriptor selected by the operand. LTR is a privileged instruction; it may be executed only when CPL is zero. LTR is generally used during system initialization to give an initial value to the task register; thereafter, the contents of TR are changed by task switch operations. STR (Store task register) stores the visible portion of the task register in a general register or memory word. STR is not privileged.

MODEL ANSWER

SUMMER – 2018 EXAMINATION

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

	b) Ans.	**Draw the block diagram of Pentium-System architecture.**  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 **OR**	**4M** *Block diagram of Pentium 4M*

		 Code Cache 8 KB Branch Prediction Pre-fetch Buffer Pipelined Floating Point Unit Memory Interface U Pipe V Pipe Integer ALU Integer ALU Register Set Multiplier Adder Divider 8 KB Data Cache	
	c) **Ans.**	**What is MMX? State its benefits.** ***(Note: Relevant topic's shorter description shall be considered)*** **MMX :** Intel's MMX media enhancement technology is a major extension of the Intel Architecture that makes PCs into richer multimedia and communications platforms. This technology introduces 57 instructions oriented to highly parallel operations with multimedia and communications data types. These instructions use a technique known as SIMD (Single Instruction, Multiple Data) to deliver better performance for multimedia and communications computation. Intel processors that provide MMX technology support are fully compatible with previous generations of the Intel Architecture and the installed base of software. To further improve performance, the Pentium II processor, like the Pentium processor with MMX technology, can execute 2 Intel MMX instructions at a time. **Benefits of MMX technology :** 1. 57 new microprocessor instructions are In MMX technology are capable to handle video, audio, and graphical data more efficiently. Programs can use MMX instructions without	**4M** ***MMX descripti on 2M***

		changing to a new mode or operating-system visible state. 2. MMX technology incorporates New 64-bit integer data type (Quad word), 4 new MMX data types for audio video signal processing. 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.	***Any 2 Benefits 1M each***
	d) **Ans.**	**State any four features of SUN Ultra SPARC.** 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 and executed by the FPU unit. The source and data results are stored in 32 register files. Majority of the floating point instructions have a throughput of one cycle and a latency of three cycles. Although the single precision (32 bit) or double precision	**4M** ***Any 4 features 1M each***

	e) Ans.	**State the function of INT 17H. Give any two examples.** **INT 17h: INT 17H** is used for Printer Services. It is a BIOS Operation interrupt to Print data and test the printer status. Parameters used by this function are : AX and DX INT 17h controls the parallel printer interfaces on the IBM PC. INT 17h provides three sub functions, specified by the value in the AH register. These sub functions are: **0**-Print the character in the AL register. **1**-Initialize the printer. **2**-Return the printer status. Like the serial port services, the printer port services allow you to specify which of the three printers installed in the system you wish to use (LPT1:, LPT2:, or LPT3:). The value in the dx register (0..2) specifies which printer port is to be used. One final note- under DOS it's possible to redirect all printer output to a serial port. This is quite useful if you're using a serial printer. The BIOS printer services only talk to parallel printer adapters. If you need to send data to a serial printer using BIOS, you'll have to use INT 14H to transmit the data through a serial port. **1. AH=0: Print a Character:** If ah is zero and when you call INT 17H, then the BIOS will print the character in the AL register. Exactly how the character code in the al register is treated is entirely up to the printer device you're using. Most printers, however, respect the printable ASCII characterset and a few control characters as well. Many printers will also printall the symbols in the IBM/ASCII character set (including European, line drawing, and other special symbols). Most printers treat control characters (especially ESC sequences) in completely different manners. Therefore, if you intend to print something other than standard ASCII characters, be forewarned that your software may not work on printers other than the brand you're developing your software on. Upon return from the INT 17H sub function zero routine, the AH register contains the current status. The values actually returned are described in the section on sub function number two. **2. AH=1: Initialize Printer** Executing this call sends an electrical impulse to the printer telling it to initialize itself. On return, the AH register contains the printer	**4M** ***Function of INT 17h 2M*** ***Any 2 examples 1M each***

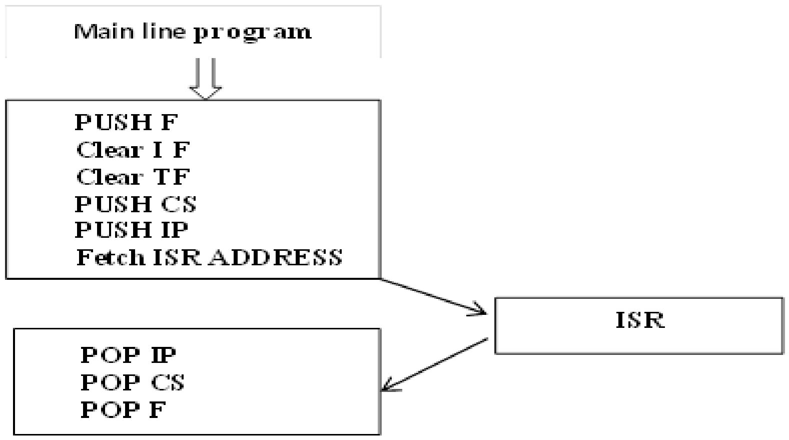
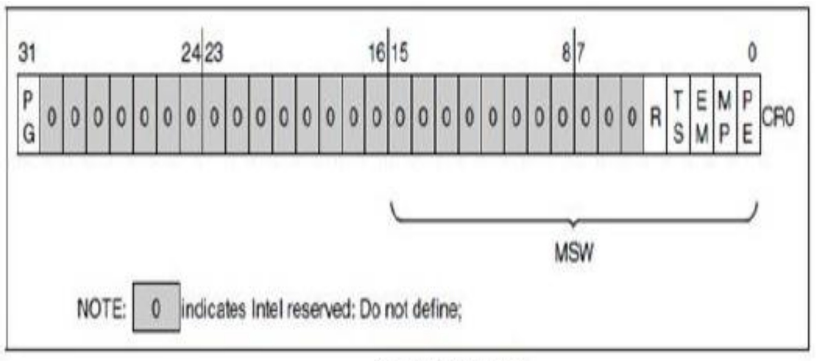
status as per function number two.

3. AH=2: Return Printer Status
This function call checks the printer status and returns it in the AH register. The values returned are:

AH: Bit	Meaning
7	1 = Printer busy, 0 = printer not busy
6	1 = Acknowledge from printer
5	1 = Out of paper signal
4	1 = Printer selected
3	1 = I/O error
2	Not used
1	Not used
0	Time out error

4. Acknowledge from printer is, essentially, a redundant signal (since printer busy/not busy gives you the same information). As long as the printer is busy, it will not accept additional data. Therefore, calling the print character function (AH=0) will result in a delay. The out of paper signal is asserted whenever the printer detects that it is out of paper. This signal is not implemented on many printer adapters. On such adapters it is always programmed to a logic zero (even if the printer is out of paper). Therefore, seeing a zero in this bit position doesn't always guarantee that there is paper in the machine. Seeing a one here, however, definitely means that your printer is out of paper.
The printer selected bit contains a one as long as the printer is on-line. If the user takes the printer off-line, then this bit will be cleared.
The I/O error bit contains a one if some general I/O error has occurred.
The time out error bit contains a one if the BIOS routine waited for an extended period of time for the printer to become "not busy" yet the printer remained busy.
Note that certain peripheral devices (other than printers) also interface to the parallel port, often in addition to a parallel printer.
Some of these devices use the error/status signal lines to return data to the PC. The software controlling such devices often takes over the

		INT 17H routine (via a technique we'll talk about later on) and always returns a "no error" status or "time out error" status if an error occurs on the printing device. Therefore, you should take care not to depend too heavily on these signals changing when you make the INT 17H BIOS calls.	
	f) **Ans.**	**Explain interrupt processing sequence of X86 microprocessor.** Interrupt processing sequence is as given below: When INT n instruction is executed: 1. The processor pushes flag register on stack then the contents of CS and IP register on stack 2. It clears two flags TF (trap flag) and IE (Interrupt enable flag). 3. Number of interrupt is used to find correct address of ISR in the IVT. 4. Interrupt number (is called as interrupt type) is used to find out the correct address of ISR in the IVT. 5. The interrupt number is multiplied by 4 to get the address with the IVT that contains the addresses of ISR. 6. ISR ADDRESS = Interrupt type x 4 7. All addresses are 4 bytes long. The Interrupt vector address is then filled in CS and IP register. Finally CPU control is transferred to new address. 8. It decrements stack pointer by 2 & push flag register on stack. 9. It clears the interrupt request by clearing interrupt flag. 10. It also reset trap flag in flag register. 11. Decrement stack pointer by 2 & store code segment in it. 12. Decrement stack pointer by 2 & pushes IP in it. 13. If fetches the ISR & jumps on it. After the completion of ISR, it decodes the instruction IRET &retrieves the main program address & status of flag register.	**4M** ***Explanation of interrupt processing 4M***

		 Main line program PUSH F Clear I F Clear TF PUSH CS PUSH IP Fetch ISR ADDRESS ISR POP IP POP CS POP F	
3.	a) Ans.	**Attempt any FOUR of the following:** **Describe enabling and disabling of paging in 80386 with neat diagram.** Enabling and disabling the paging in 80386 is done with the help of CR0 register. the layout of CR0 is as shown in the figure below:  31 24 23 16 15 8 7 0 PG 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 R TS EM MP PE CR0 MSW NOTE: 0 indicates Intel reserved: Do not define; Control Register 0 • By setting the PG bit (Most significant Bit or bit 31) of CR0 to 1, paging can be enabled. • This bit cannot be set until the processor is in protected mode. • When PG is set, the PDBR (Page Directory Base Register – CR3) should already be initialized with a physical address that points to a valid page directory.	(4x4=16) 4M *Diagram 2M* *Description 2M*

		• To enable paging, use MOV instruction to set most significant bit of CR0, • *E.g.* MOV CR0, EAX • After this bit is set, page translation takes effect on the next instruction. Paging can be disabled by making the PG bit of CR0 as zero.	
	b) **Ans.**	**List any four important 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. 2. It supports burst read and burst write back cycles 3. It supports pipelining 4. It has a separate Instruction cache. 5. Pentium processor has 8 KB of dedicated instruction cache 6. It has Two Integer execution units, one Floating point execution unit 7. It has a Dual instruction pipeline 8. It has 256 lines between instruction cache and prefetch buffers; allows 32 bytes to be transferred from cache to buffer 9. It has a separate Data cache. 10. It has a 8 KB dedicated data cache gives data to execution units 11. It has 32 byte lines. 12. Pentium processor has Two parallel integer execution units 13. It allows the execution of two instructions to be executed simultaneously in a single processor clock. 14. It has a Floating point unit for Faster internal operations. 15. It has a Local advanced programmable interrupt controller, it Speeds-up upto 5 times for common operations including add, multiply and load, than 80486. 16. It has a Branch Prediction Logic. 17. To reduce the time required for a branch caused by internal delays 18. When a branch instruction is encountered, microprocessor begins prefetch instruction at the branch address. 19. It has Data Integrity and Error Detection logic. 20. Has significant error detection and data integrity capability. 21. Data parity checking is done on byte – byte basis.	**4M** ***Any 4 features of Pentium processor 1M each***

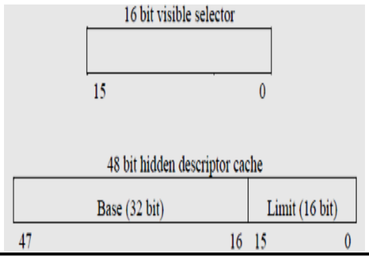
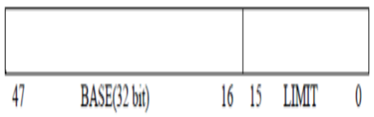
22. Address parity checking and internal parity checking features are added.
23. It has a Dual Integer Processor which allows execution of two instructions per clock cycle

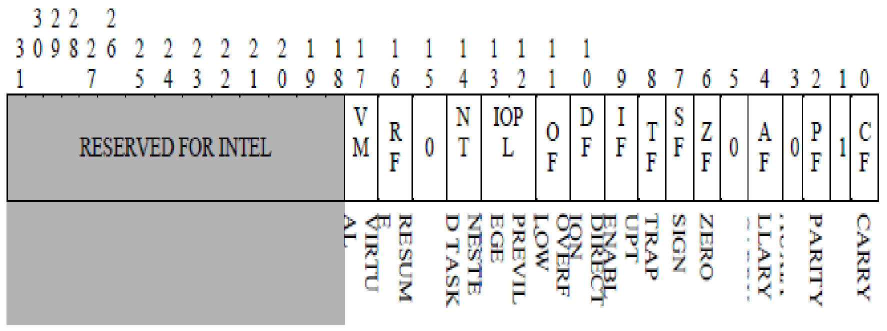
c) **Distinguish between LDTR and GDTR (Any four points).** **4M**

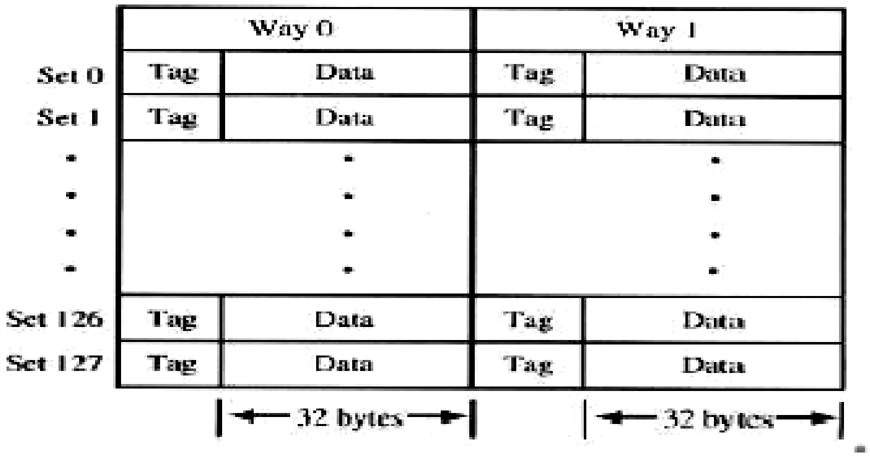
Ans.

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

4 Valid differences 1M each

		 16 bit visible selector 15 0 48 bit hidden descriptor cache Base (32 bit) Limit (16 bit) 47 16 15 0 	
	d) Ans.	**State the instruction latency in RISC processor designing.** **Instruction Latency:** 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.	**4M** ***Instruction latency description 4M***
	e) Ans.	**Describe divide by zero error and single step interrupts of X86 processors.** **1. Divide by zero:** • This interrupt is caused by the instructions such as DIV or IDIV.	**4M**

		• 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.	***Divide by zero interrupt 2M*** ***Single step interrupt 2M***
4.	**a)** **i)** **Ans.**	**Attempt any three of the following:** **Explain the flag register format of 80386 with suitable figure.** ***(Note: The description of all flags is not expected)***  31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 RESERVED FOR INTEL VM RF 0 NT IOPL OF DF IF TF SF ZF 0 AF 0 PF 1 CF VIRTUAL RESUME NESTED TASK PREVILEGE OVERFLOW DIRECTION ENABLE TRAP SIGN ZERO AUXILLARY PARITY CARRY **Flag register Organization:** The flag register in 80386 is 32 bits registers. It is also called as E-flags. The active flags in the flag register of 386 are as given below : 1. Carry Flag 2. Parity Flag 3. Auxiliary Carry Flag 4. Zero Flag 5. Sign Flag	**(3x4=12)** **4M** ***Flag register format of 80386 diagram 2M*** ***Description 2M***

		6. Trap Flag 7. Interrupt Enable Flag 8. Direction Flag 9. Overflow Flag 10. IO Privilege Level Flag 11. Nested Task Flag 12. Virtual Mode Flag 13. Resume Flag	
	ii) **Ans.**	**Explain separate code and data cache of Pentium micro processor.** **Separate 8K B instruction and Data Cache :** The following figure shows the organization of instruction and data cache.  Way 0 Way 1 Set 0 Set 1 Set 126 Set 127 Tag Data 32 bytes 32 bytes The Pentium processor has 2 separate 8KB data and code Caches. But they need more bandwidth than the unified cache. Both the caches have TLB's associated with them. The TLBs are used to covert the linear addresses to the respective physical addresses. As the data cache stores only 8KB data and code cache stores only instructions, the lookup process speed for Pentium increases. **Advantages of separate instruction and data caches :** 1. Separate code and data cache memories effectively and efficiently executes the branch prediction. 2. Simultaneous cache look up is achieved by Pentium processor	**4M** ***Separate code and instruction cache 4M***

		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.	
	iii) **Ans.**	**Explain the concept of pipelining in RISC processor.** 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.	**4M** ***Concept of pipelining in RISC 4M***
	iv) **Ans.**	**State the functions of following interrupts:** **1) INT 10H: Function 06H and 07H** **2) INT 21H: Function 01H and 02H.** **1) INT 10H': Function 06H and 07H:** **Function 06H**: this function is used to scroll the screen in upward direction. The parameters used are : Ah=06h AL,=number of lines ,00=default for whole screen BH=color attributes of the screen CH= starting row CL=starting column	**4M** ***Function 06h 1M***

DH=ending row
DL=ending column
for example
MOV AH,06H
MOV AL,10
MOV BH,61H
MOV CX,1010
MOV DX,2030
INT 10H
Will scroll the screen 10 line up starting from row column 10. 10 and ending row upto 20 and column number 30.

Function 07H: this function is used to scroll the screen in downward direction.
The parameters used are :
AH=07h
AL =number of lines ,00=default for whole screen
BH=color attributes of the screen
CH= starting row
CL=starting column
DH=ending row
DL=ending column
for example
MOV AH,06H
MOV AL,10
MOV BH,61H
MOV CX,2030
MOV DX,1010
INT 10H
Will scroll the screen 10 lines down starting from row column 20 30 and ending row upto 10 and column number 10.

Function 07h 1M

2) INT 21 H:
Function 1- Character input with echo

Action:	Reads a character from the standard input device and echoes it to the standard output device. If no character is ready it waits until one is available. I/O can be re-directed, but prevents detection of OEF.
On entry:	AH = 01h

Function 01h1M

Returns:	AL = 8 bit data input
Notes:	Equivalent to CP/M BDOS call 01h, except that if the character is CTRL-C an INT 23H is performed. Syntax : MOV AH,01H INT 21H Will accept a single alphanumeric /character input and will store its ASCII Code into AL register .Also the character input will be echoed on the screen.

Function 2 - Character output

Action:	Outputs a character to the standard output device. I/O can be re-directed, but prevents detection of 'disc full'.

On entry:	AH = 02h
	DL = 8 bit data (usually ASCII character)

Returns:	Nothing
Syntax	Mov ah,02h Mov dl, “*” Int 21h Will display * on the standard output device i.e. Monitor.

Function 02h 1M

4. b) **Attempt any one of the following:** **(1x6=6)**

i) **Describe Pentium-Il processor with respect to cache memory, memory banking and input/ output system.** **6M**

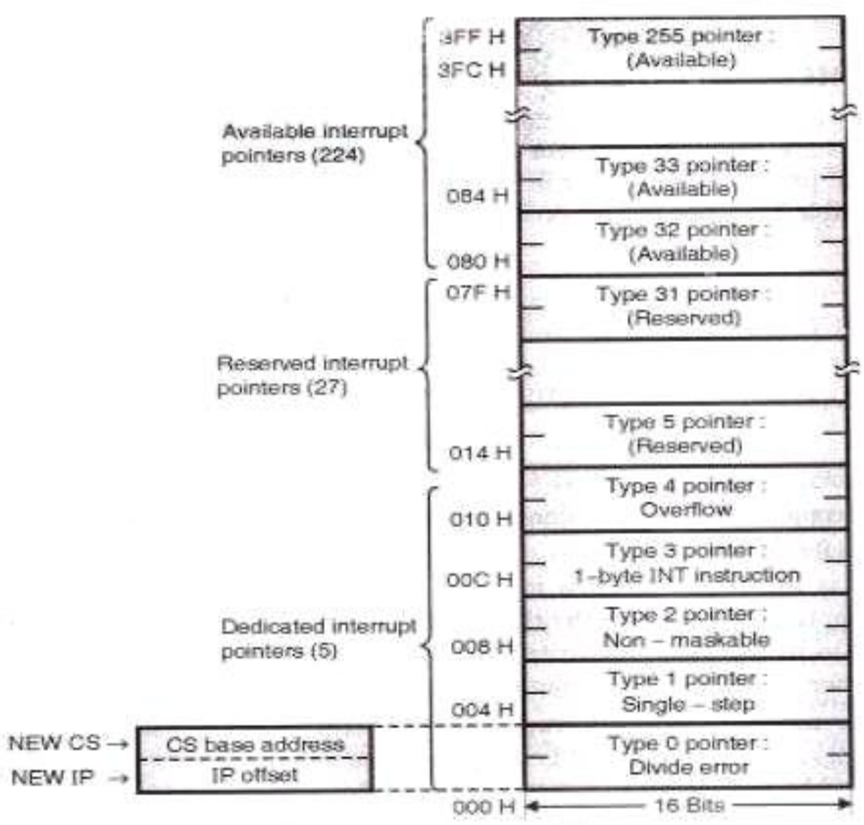
(Note: Any other relevant description shall be considered).

Ans. **Cache memory and memory banking in Pentium II processor:**
The Pentium II processor has 32 KB of non-blocking L1 cache, which is divided into a 16K instruction cache and a 16K data cache. Each of these caches runs at the processor frequency and provides fast access to heavily used data. The Pentium II processor has a 512K L2 cache which is unified for code and data, and is non-blocking. There is a dedicated 64-bit bus to facilitate higher data transfer rates between the processor and the L2 cache.

Cache memory 2M, Memory banking 2M and IO 2M

Input /output system:
Write Combining: ⇒ The Write Combining technology of the P6 architecture can be used to achieve very high graphics I/O

		performance. This feature combines multiple writes to a region of memory (for example, a video controller's frame buffer) declared as WC type into a single-burst write operation. This is well suited for the bus, which is optimized for burst transfers. The combining also leads to burst writes of cache line sizes. These writes are further combined by the chipset, leading to high throughput for graphics I/O. The result is enhanced multimedia performance, more realistic full-motion video, and realistic, fast graphics performance.	
	ii) **Ans.**	**Explain interrupt vector table of X86 processor with neat diagram.** ***(Note: Any other relevant description should be given marks)*** 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:	**6M** ***Description 3M***

		 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	***Input Vector Table diagram 3M***
5.	**a)** **Ans.**	**Attempt any four of the following:** **Describe the function of following pins of 80386:** **i) $\overline{BE_0} - \overline{BE_3}$ ii) $D/\overline{C}$** **iii) $\overline{LOCK}$ iv) $\overline{BUSY}$** **i) $\overline{BE0}$-$\overline{BE3}$ (Bus/byte enable signal):** The 32-bit Data bus supported by 80386 and the memory system of 80386 can be viewed as a 4-byte wide memory access mechanism. The four byte enable lines, , may be used for enabling these four banks. Using these four enable signal lines, the CPU may transfer 1 byte/2bytes/3bytes or 4bytes of data	**(4x4=16)** **4M** ***Each description 1M***

simultaneously.

Byte Enable Signal	Data bus Signal
BE_0	D_0-D_7 (Byte 0- least significant)
BE_1	D_8- D_{15} (Byte 1)
BE_2	D_{16}- D_{23} (Byte 2)
BE_3	D_{24}- D_{31} (Byte 3- Most Significant)

2. $D/\overline{C}$: The data/control output pin distinguishes between a data transfer cycle from a machine control cycle.

3. $\overline{LOCK}$: The output pin enables the CPU to prevent the other bus masters (like coprocessor) from gaining the control of the system bus.

4. $\overline{BUSY}$: It signals a busy condition from a processor extension. When asserted this input indicates the coprocessor is still executing an instruction and is not yet able to accept another.

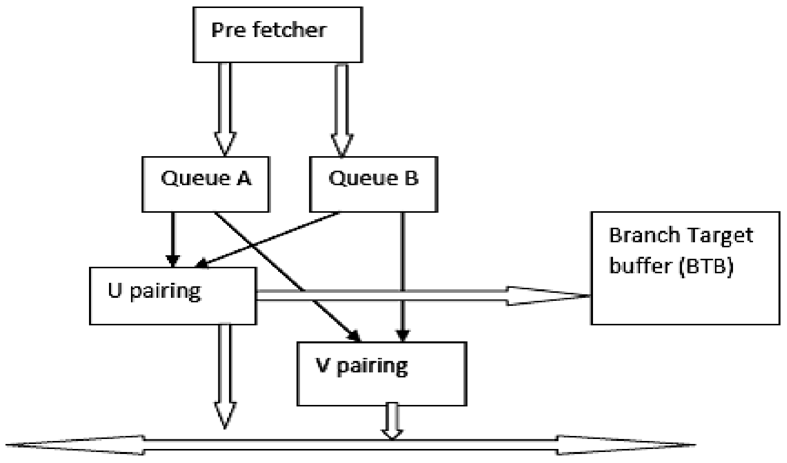
b) Ans. **Explain function of branch prediction unit in Pentium processor.** **4M**

Branch Prediction:

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

The branch instructions occur frequently while running any application. These instructions change the normal sequential control flow of the program and may stall the pipelined execution in the Pentium system. Branches may be of two types: Conditional branch and unconditional branch. In case of conditional branch, the CPU has to wait till the execution stage to determine whether the condition is met or not. ***Explanation 2M***

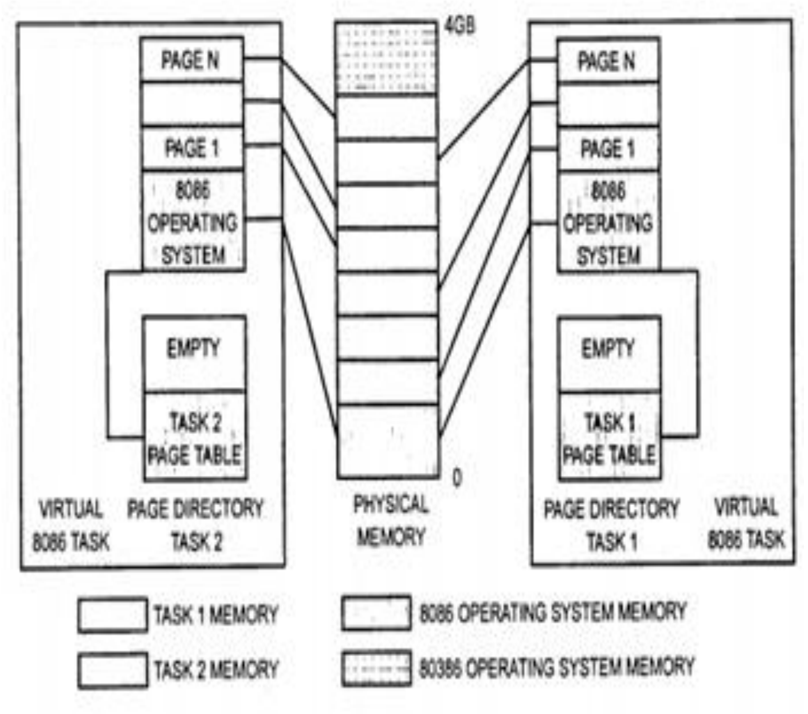
The Pentium processor makes the dynamic branch prediction using a Branch Target Buffer (BTB). To efficiently predict branches, the Pentium uses two prefetch buffers. One buffer prefetches code in linear fashion, while the other prefetches instructions based on address in the branch target buffer. As a result the needed code is

		prefetched before it is required for execution.The Pentium processors prediction algorithm not only forecast the simple branch choices but also supports more complex branch prediction for example, within nested loops. This is achieved by storing multiple branch address in the branch prediction buffer. The design of the branch target buffer allows 256 addresses to be stored and thus the prediction algorithm can forecast up to 256 branches **Branch Prediction Logic:-**  Pre fetcher Queue A Queue B Branch Target buffer (BTB) U pairing V pairing	***Diagram 2M***
	c) **Ans.**	**State features of Pentium-III processor. (any four).** ***(Note: Any other significant features to be given marks)*** **The features of Pentium III processor are:** 1. Pentium III processor has 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. 2. PIII is incorporated with MMX technology. 3. Dynamic execution, micro-architecture incorporates unique combination of multiple branch prediction, data flow analysis and speculative execution. 4. It Supports power management capabilities like System management mode and	**4M** ***Any 4 features 1M each***

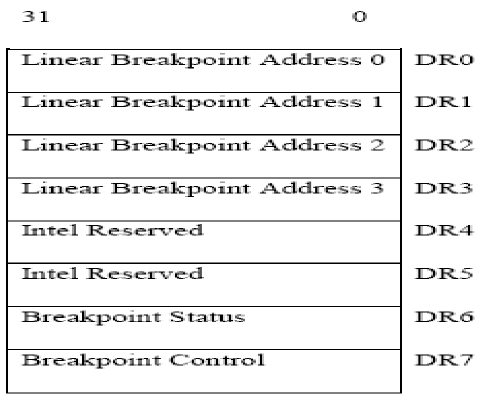
		5. The Pentium III processor has Multiple low power states. 6. Pentium III is optimized for 32 bits applications running on advanced 32 bits OS. 7. It has 32KB L1 cache divided as 16KB instruction cache and 16KB data cache. 8. It has Quad quad word wide ie. 256 bits cache data bus , ways set associative cache 9. It provides improved cache hit rate. 10. It supports Multiprocessor system. 11. It Works on 1.0 GHz, 850, 800, 750, 700, 650 MHZ.	
	d) **Ans.**	**What do you mean by 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. 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 overlaps. The overlap consists of 8 registers in CPU. Notice that the organizations of the windows are supposed to be circular and not linear; meaning that the last window overlaps	**4M** ***Explanation 4M***

		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.	
	e) **Ans.**	**List hardware interrupts of X86 processors. Explain in brief about overflow interrupt.** **Hardware Interrupts** (External Interrupts): If the signal for the processor is from external device or hardware is called hardware interrupts. Example: from keyboard we will press the key to do some action this pressing of key in keyboard will generate a signal which is given to the processor to do action, such interrupts are called hardware interrupts. Hardware interrupts can be classified into two types they are: 1. Maskable Interrupt: The hardware interrupts which can be delayed when a much highest priority interrupt has occurred to the processor. 2. Non Maskable Interrupt: The hardware which cannot be delayed and should process by the processor immediately. The Intel microprocessors **support hardware interrupts** through: Two pins that allow interrupt requests, INTR and NMI One pin that acknowledge, INTA, the interrupt requested on INTR. **INTR and NMI:** **INTR** is a maskable hardware interrupt. The interrupt can be enabled/disabled using STI/CLI instructions or using more complicated method of updating the FLAGS register with the help of the POPF instruction. **NMI** is a non-maskable interrupt. Interrupt is processed in the same way as the INTR interrupt. Interrupt type of the NMI is 2, i.e. the address of the NMI processing routine is stored in location 0008h. This interrupt has higher priority than the maskable interrupt. – ***Ex:* NMI, INTR.** **Overflow**: This trap occurs when the processor encounters an INTO instruction and the OF (overflow) flag is set. Since signed arithmetic and	**4M** ***List 1M*** ***Explanation2M*** ***Overflow 1M***

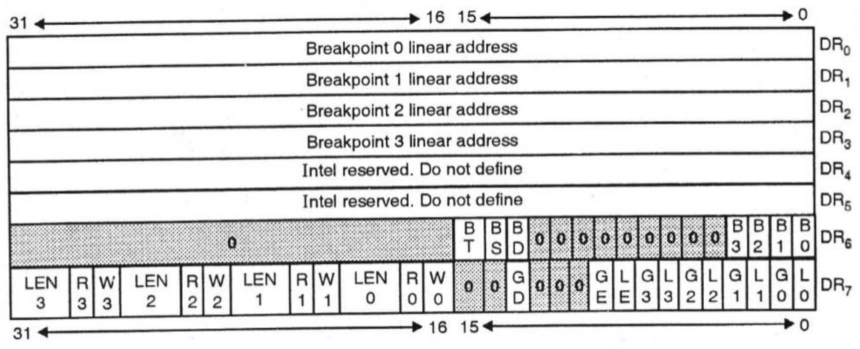
		unsigned arithmetic both use the same arithmetic instructions, the processor cannot determine which is intended and therefore does not cause overflow exceptions automatically. Instead it merely sets OF when the results, if interpreted assigned numbers, would be out of range. When doing arithmetic on signed operands, careful programmers and compilers either test OF directly or use the INTO instruction.	
	f) **Ans.**	**Draw and explain virtual 8086 mode in 80386.** 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 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	**4M** *Explanation 2M*

		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.  4GB PAGE N PAGE 1 8086 OPERATING SYSTEM EMPTY TASK 2 PAGE TABLE VIRTUAL 8086 TASK PAGE DIRECTORY TASK 2 PHYSICAL MEMORY 0 TASK 1 PAGE TABLE PAGE DIRECTORY TASK 1 TASK 1 MEMORY TASK 2 MEMORY 8086 OPERATING SYSTEM MEMORY 80386 OPERATING SYSTEM MEMORY	*Diagram 2M*
6.	**a)** **Ans.**	**Attempt any four of the following:** **Describe debug and test register of 80386 microprocessor.**	**(4x4=16)** **4M**

Debug register:


31
0
Linear Breakpoint Address 0
Linear Breakpoint Address 1
Linear Breakpoint Address 2
Linear Breakpoint Address 3
Intel Reserved
Intel Reserved
Breakpoint Status
Breakpoint Control
DR0
DR1
DR2
DR3
DR4
DR5
DR6
DR7

OR

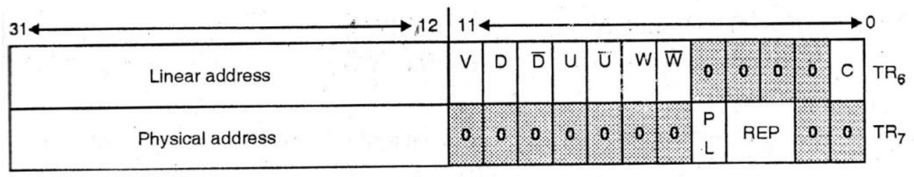
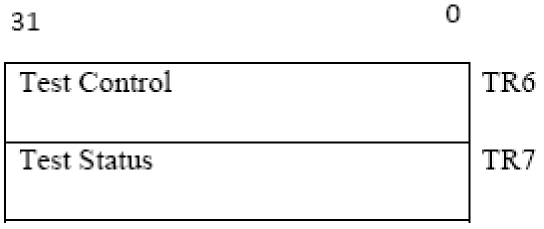

31
16 15
0
Breakpoint 0 linear address
Breakpoint 1 linear address
Breakpoint 2 linear address
Breakpoint 3 linear address
Intel reserved. Do not define
Intel reserved. Do not define
0
BT BS BD 0 0 0 0 0 0 0 0 0 B3 B2 B1 B0
LEN 3 R3 W3 LEN 2 R2 W2 LEN 1 R1 W1 LEN 0 R0 W0 0 0 GD 0 0 0 GE LE G3 L3 G2 L2 G1 L1 G0 L0
DR0
DR1
DR2
DR3
DR4
DR5
DR6
DR7

Debug Registers 1M for diagram

There are eight debug registers DR0 to DR7 for hardware debugging. The DR0 to DR3 are used to store program controllable breakpoint addresses. The DR4 and DR5 are not used and are reserved by Intel. The DR6 and DR7 are used to hold the breakpoint status and breakpoint control information respectively.

Test registers of 80386: The 80386 has two test registers for page caching. The registers are TR6 – Test Control and TR7 – Test Status. TR6 & TR7 are used for translation look aside buffer (TLB). TLB holds page table address translation to reduce the no. of memory required for page table translation.

1M for description

		 31 12 11 0 Linear address V D D̄ U Ū W W̄ 0 0 0 0 C TR6 Physical address 0 0 0 0 0 0 0 PL REP 0 0 TR7 The test registers are used to perform the confidence checking on the paging. TR6 is the TLB testing command register. By writing into this register, you can either initiate a write directly into the TLB or perform a mock TLB lookup. TR7 is the TLB testing data register. When a program is performing writes, the entry to be stored is contained in this register, along with cache set information. 	***1M for description*** ***1M for diagram***
	b) **Ans.**	**Describe the features of Pentium MMX (any four).** **Features of Pentium MMX:** 1.In Pentium there are eight general purpose floating point registers in a floating point unit. 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	**4M** ***Any 4 features 1M each***

		floating point instructions. 8. Although content switching between multimedia program execution and floating point execution is permissible. It is not recommended. **The MMX technology supports the following four data types.** 1. Packed bytes-In this data types, eight bytes can be packed into one 64 bit quantity. 2. Packed word-Four words can be packed into 64 bit. 3. Packed double word-Two double words can be packed into 64 bit 4. One quadword-One single64 bit quantity.	
	c) Ans.	**List and explain design issues of RISC processor (any two).** **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 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	**4M** ***List design issues 2M*** ***Explain any two 1M each***

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.

<u>Drawbacks of register windowing </u>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.

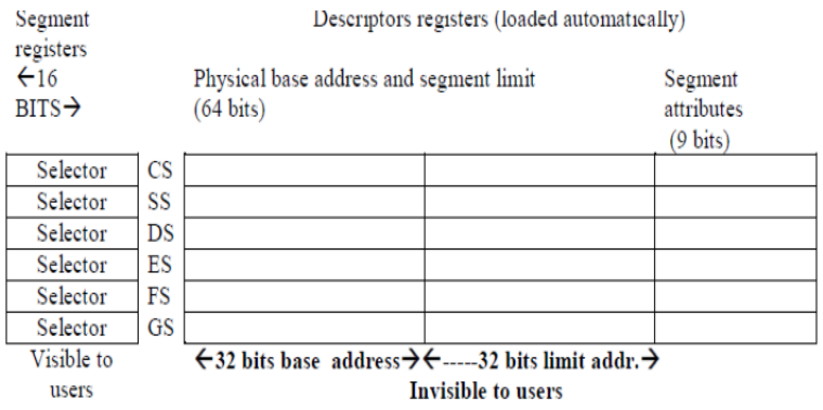
2. Pipelining in RISC: A RISC processor pipeline operates in much the same way, although the stages in the pipeline are different. While different processors have different numbers of steps, they are basically variations of these five, used in the MIPS R3000 processor:
1. Fetch instructions from memory
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.	
	d) **Ans.**	**List any four features of Pentium-proprocessor.** **Features of Pentium-proprocessor:** 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)	**4M** ***Any 4 features 1M each***

		voltage inputs. Separate 16-Kbyte, 4-way set-associative code and data caches, each with improved fully associative TLBs 10) Pool of four write buffers used by both execution pipelines 11) Enhanced branch prediction algorithm 12) New Fetch pipeline stage between Prefetch and Instruction Decode.	
	e) Ans.	**Explain the concept of segment descriptor cache register of 80386 with neat diagram.** **Segment descriptor cache registers:** - These registers are not available for the users. - These registers are associated with the segments and the segment registers in 80386 i.e. CS,DS,ES,SS,FS,GS - Every segment descriptor cache register is 72 bits long. -Every segment descriptor cache register holds a. 32 bits segment base address b. 32 bits segment limit c. Other required segment attributes. -When a selector is loaded, its associated segment descriptor cache register is automatically get loaded with the values from descriptor table. Either from LDT or GDT. -In the real mode, only the base address is updated directly by shifting the selector values 4 bits to the left. -In the protected mode, the base address, limit and all attributes are loaded.  Segment registers ←16 BITS→ Descriptors registers (loaded automatically) Physical base address and segment limit (64 bits) Segment attributes (9 bits) Selector CS Selector SS Selector DS Selector ES Selector FS Selector GS Visible to users ←32 bits base address→←-----32 bits limit addr.→ Invisible to users	4M *Explanation 2M* *Diagram 2M*