1. Explain 80286 architecture. OR

List the four major processing units in an 80286 microprocessor and briefly describe the function of each.

Ans - The 80286 was designed for multi-user systems with multitasking applications, including communications and real-time process control.
- It had 134,000 transistors and consisted of four independent units: address unit, bus unit, instruction unit and execution unit.
- These were organized into a pipeline, significantly increasing performance.
- It was produced in a 68-pin package including PLCC (Plastic Leaded Chip Carrier), LCC (Leadless chip carrier) and PGA (Pin Grid Array) packages.
- The Intel 80286 had a 24-bit address bus and was able to address up to 16 MB of RAM, compared to 1 MB for its predecessor. However, cost and initial rarity of software using the memory above 1 MB meant that 80286 computers were rarely shipped with more than one megabyte of RAM.
- 80286 Architecture contains 4 separate processing units.
  (1) Bus Unit (BU)
  (2) Instruction Unit (IU)
  (3) Address Unit (AU)
  (4) Execution Unit (EU)
Bus Unit (BU):
It has address latches, data transceivers, bus interface and circuitry, instruction pre-fetcher, processor extension interface and 6 byte instruction queue.

Functions :
- To perform all memory and I/O read and write.
- To pre-fetch the instruction bytes.
- To control the transfer of data to and from processor extension devices like 80287 math co-processor.
- Whenever BU is not using the buses for the operation, it pre-fetches the instruction bytes and put them is a 6 byte pre-fetch queue.

Instruction Unit (IU):
It has 3 decoded instruction queue and instruction decoder.

Functions :
- It fully decodes up to three prefetched instructions and holds them in a queue.
- So that EU can access them.
- It helps the processor to speed up, as pipelining of instruction is done.

Execution Unit (EU):
It includes ALU, registers and the Control unit. Registers are general purpose, index, pointer, flag register and 16 –bit Machine Status Word (MSW).

Functions :
- To sequentially execute the instructions received from the instruction unit.
- ALU result is either stored in register bank or sent over the data bus.

Address Unit (AU):
It consists of segment registers, offset address and a physical address adder.

Functions :
- Compute the physical address that will be sent out to the memory or I/O by BU.
- 80286 operate in two different modes
  1. real address mode
  2. Protected virtual address mode.
- When used in Real address mode, AU computes the address with segment base and offset like 8086. Segment register are CS, DS, ES and SS hold base address. IP, BP, SI, DI , SP hold offset.
- Maximum physical space allowed in this mode is 1MB.
- When 80286 operate in protected mode, the address unit acts as MMU.
- All 24 address lines used and can access up to 16MB of physical memory.
- If descriptor table scheme is used it can address up to 1GB of virtual memory.

2. Explain Register Organization of 80286
Ans The 80286 CPU contains almost the same set of registers, as in 8086, namely
1. Eight 16-bit general purpose registers (AX, BX, CX, DX)
2. Four 16-bit segment registers (CS, SS, DS, ES)
3. Status and control registers (SP, BP, SI, DI)
4. Instruction Pointer (IP)
5. Two 16-bit register - FLAGS, MSW
6. Two 16-bit register - LDTR and TR
7. Two 48-bit register - GDTR and IDTR

![Figure: 80286 Register Set]

The flag register reflects the results of logical and arithmetic instructions.

<table>
<thead>
<tr>
<th>-</th>
<th>NT</th>
<th>IOPL</th>
<th>OF</th>
<th>DF</th>
<th>IF</th>
<th>TF</th>
<th>SF</th>
<th>ZF</th>
<th>-</th>
<th>AF</th>
<th>-</th>
<th>PF</th>
<th>-</th>
<th>CF</th>
</tr>
</thead>
</table>

![Figure: 80286 Flag Register]

- Flag register is of 32-bit, 15th bit undefined/reserved.
- System flags: reflect the current status of machine.
  i. IOPL – I/O Privilege Level flag: 2-bits are used in protected mode. It holds the privilege level from 0 to 3. ‘0’ assigns to highest privilege whereas ‘3’ assigns to lower privilege level.
  ii. NT: Nested Task flag: It is used in protected mode. Bit is set when one task invokes another task.

3. **Salient Features of 80286**

**Ans**
- The 80286 is the first member of the family of advanced microprocessors with memory management and protection abilities.
- The 80286 CPU, with its 24-bit address bus is able to address 16 Mbytes of physical memory. Various versions of 80286 are available that runs on 12.5 MHz, 10 MHz and 8 MHz clock frequencies.
- 80286 is upwardly compatible with 8086 in terms of instruction set.
- 80286 have two operating modes namely real address mode and virtual address mode.
- In real address mode, the 80286 can address up to 1Mb of physical memory address like 8086.
- In virtual address mode, it can address up to 16 Mb of physical memory address space and 1 GB of virtual memory address space.
- The instruction set of 80286 includes the instructions of 8086 and 80186.
- 80286 has some extra instructions to support operating system and memory management.
- The performance of 80286 is five times faster than the standard 8086.

4. **Explain MMU of 80286.**

**Ans** In advanced microprocessor, memory management becomes extremely important. Memory management is required due to the following two reasons.

1. **Limitation of Physical Memory**

   A microprocessor has limited number of address lines. Hence the physical memory addressability is limited. Increasing the number of address lines is not attractive as it makes the architecture and design complex without significant gain. Packaging becomes difficult and expensive. Memory Management Unit (MMU) solves this problem by translating the virtual memory address into the physical memory address. Virtual memory can be many times larger than the physical memory. Only the programs currently required are brought from the secondary storage such as a hard disk to, the physical memory (RAM) for execution.

2. **Need for Protection**

   In a multi-user operating system, there is a possibility that a user program can corrupt the operating system area or the area of some other user unless a protection mechanism is built. Hence each user should be protected from other users and the operating system should be protected from other user (task). The user (task) should be allowed to have a controlled access to the operating system resources. Hence various privilege levels are defined. For example, in a situation having 4 privilege levels, 0 is the highest privilege and 3 has the lowest privilege as shown in the figure. The figure shows a typical UNIX operating system layout. It has to be noted that the user is at the lowest privilege level, i.e., privilege level 3 and the operating system Kernel is at the highest privilege level, i.e., privilege level 0.

**Role of Memory Management Unit (MMU):**

- The virtual address space of a microprocessor may be many times larger than the actual physical address space.
- This is desirable as a microprocessor is supposed to store large programs and data which cannot be accommodated in the physical memory space.
- Usually programs and data are stored in a secondary storage such as a hard disk.
- The hard disk is in the virtual or logical address space but not in the physical address space.
- Faster memory such as RAM is used as the physical memory (Primary Storage).
- When a microprocessor is to execute a program, it checks whether the program is available in the physical memory (RAM).
- If the program is not available in the physical memory, it is brought from the secondary memory to the physical memory for execution.
- If available space is inadequate in the physical memory, some less important or unused program can be swapped back to the secondary memory to create space.

5. **Explain Operating Modes of 80286**

80286 works in two operating mode:

i. **Real Address Mode (Just act as a fast 8086).**
   - Instruction set is upwardly compatible
   - Because of extra pipelining and other circuit level improvements, in real address mode also, the 80286 operates at a much faster rate than 8086, although functionally they work in an identical fashion.
   - As in 8086, the physical memory is organized in terms of segments of 64Kbyte maximum size.
   - In the real mode the first 1Kbyte of memory starting from address 0000H to 003FFH is reserved for interrupt vector table.
   - The addresses from FFF0H to FFFFH are reserved for system initialization.
   - When the 80286 is reset, it always starts the execution in real address mode.
   - In real address mode, it initializes the IP and other registers of 80286.

![Real Address Mode Address Calculation](image-url)
ii. Protected Virtual Address Mode (PVAM)

- 80286 is the first processor to support the concepts of virtual memory and memory management.
- The concept of Virtual Memory is implemented using Physical memory that the CPU can directly access and secondary memory that is used as storage for data and program, which are stored in secondary memory initially.
- The complete virtual memory is mapped on to the 16Mbyte physical memory.
- If a program larger than 16Mbyte is stored on the hard disk and is to be executed, if it is fetched in terms of data or program segments of less than 16Mbyte in size into the program memory by swapping sequentially as per sequence of execution.
- The 80286 is able to address 1 GB ($2^{30}$ bytes) of virtual memory.
- 80286 uses the 16-bit content of a segment register as a selector to address a descriptor stored in the physical memory.
- The **descriptor** is a block of contiguous memory locations containing information of a segment, like segment base address, segment limit, segment type, privilege level, segment availability in physical memory descriptor type and segment.
- Hardware reset is the only way to come out of protected mode

![Physical Address Calculation in PVAM](image_url)
6. Explain Privilege level.

**Ans** There are four types of privilege levels

1. 00 - kernel level (highest privilege level)
2. 01 - OS services
3. 10 - OS extensions
4. 11 - Applications (lowest privilege level)

![Figure: Privilege Level]

- Each task assigned a privilege level, which indicates the priority or privilege of that task.
- It can only changed by transferring the control, using gate descriptors, to a new segment.
- A task executing at level 0, the most privileged level, can access all the data segment defined in GDT and LDT of the task.
- A task executing at level 3, the least privileged level, will have the most limited access to data and other descriptors.
- The use of rings allows for system software to restrict tasks from accessing data.
- In most environments, the operating system and some device drivers run in ring 0 and applications run in ring 3.

7. What is Descriptor table? What is its use? Differentiate between GDT and LDT.

**Ans**

- Descriptor is a identifier of a program segment or page.
- A segment cannot be accessed, if its descriptor does not exist in either LDT or GDT.
- Set of descriptor (descriptor table) arranged in a proper sequence describes the complete program.
- The descriptor is a block of contiguous memory location containing information of a segment, like
  i. Segment base address
  ii. Segment limit
iii. Segment type
iv. Privilege level – prevents unauthorized access
v. Segment availability in physical memory
vi. Descriptor type
vii. Segment use by another task

Figure: Protected Mode Addressing with Descriptor Table

Requirement of Descriptor Table:
- The descriptor describes the location, length, and access rights of the segment of memory.
- The selector, located in the segment register, selects one of descriptors from one of two tables of descriptors.

7. Explain LDT, GDT and IDT. Differentiate LDT and GDT

**Ans Global Descriptor Table (GDT):**
- The 80286 has a single Global Descriptor Table (GDT) which is shared between all tasks and addresses up to 512MB of the virtual address space.
- The **Global Descriptor Table** or GDT is a data structure used by Intel x86-family processors starting with the 80286 in order to define the characteristics of the various memory areas used during program execution, including the base address, the size and access privileges like execute-ability and write-ability.

**Local Descriptor Table (LDT):**
- Each task will have its own Local Descriptor Table (LDT) which is a private 512MB of address space.
- LDT is essential to implement separate address spaces for multiple processes.
- The operating system will switch the current LDT when scheduling a new process, using the LDT machine instruction.

**Descriptor Table (LDT):**
- IDT used to store interrupt gates and task gates.
- LIDT instruction is used to Load Interrupt Descriptor table.

Differentiate LDT and GDT:
- LDT is actually defined by a descriptor inside the GDT, while the GDT is directly defined by a linear address.
- The lack of symmetry between both tables is underlined by the fact that the current LDT can be automatically switched on certain events, notably if TSS-based multitasking is used, while this is not possible for the GDT.
- The LDT also cannot store certain privileged types of memory segments.
- The LDT is the sibling of the Global Descriptor Table (GDT) and similarly defines up to 8191 memory segments accessible to programs.
- LDT (and GDT) entries which point to identical memory areas are called aliases.
- Instruction to load GDT is LGDT (Load Global Descriptor Table) and instruction to load LDT is LLDT (Load Local Descriptor Table). Both are privileged instructions.

7. Processor Status Word in 80286
Ans There are four Processor Status Word in 80286:
1. PE - Protection Enable
   The PE bit is set to enable the Protected Mode. If PE is reset, the processor operates again in Real Mode.
2. MP - Monitor Processor Extension
   The MP bit is used in conjunction with the TS bit to determine if the WAIT opcode will generate a Coprocessor Not Available fault when TS = 1. When both MP = 1 and TS = 1, the WAIT opcode generates a trap. Otherwise, the WAIT opcode does not generate a trap. Note that TS is automatically set whenever a task switch operation is performed.
3. EM - Processor Extension Emulator
   The EMulate coprocessor bit is set to cause all coprocessor opcodes to generate a Coprocessor Not Available fault. It is reset to allow coprocessor opcodes to be executed on an actual Intel387 DX coprocessor. Note that the WAIT opcode is not affected by the EM bit setting.
4. TS – Task Switch
   TS is automatically set whenever a task switch operation is performed.

     ![Processor Status Word in 80286](image)

Figure: Processor Status Word in 80286

8. Explain 80286 Interrupt Handling
Ans
- Real addressing mode has 256 interrupts with types 0-255.
- Each interrupt takes 4 bytes, so we have to reserve 1KByte of memory for interrupt.
- In PVAM mode also we have 256 interrupts but it is not assigned a fixed memory.
- The Interrupt descriptor table can be anywhere in the physical memory.
- Base address of interrupt descriptor table is stored in interrupt descriptor table register (IDTR).
- The particular descriptor is accessed as follows:-

\[(\text{Interrupt Type} \times 8) + \text{IDTR} \rightarrow \text{Descriptor}\]

9. **Draw and explain Pin Diagram of 80286.**

**Ans**

The pin interface diagram of 80286 is as follows:-

Signals/pins are divide into four groups

1. Memory I/O interfaces signals
2. Interrupt interface signals
3. DMA interface signals
4. Co-processor interface signals

![80286 Interface Diagram](image)

- VCC is the system power supply connection for 10%, +5.0 V.
- VSS is the system ground connection.
- CLK is System Clock.
- RESET the companion reset output pin (goes high for a reset) connects to system peripherals to initialize them whenever the input goes low.
- NMI is a non-mask able interrupt input. It is positive edge-triggered and always active. When NMI is
activated, it uses interrupt vector 2.
- A0-A23: Unidirectional Address bus.
- D0- D15: Bidirectional Data bus.
- The **bus high enable** pin indicates (when a logic 0) that valid data are transferred through data bus.
- PEREQ and PEACK#: Processor extension/co-processor request and acknowledge.
- PEREQ input request the 80286 to perform data transfer for a processor extension/co-processor.
- Co-processor cannot transfer the data over the data bus by itself. Whenever it needs read or write the data from memory, it indicates 80286 to initiate the data transfer by making PEREQ signal high.
- PEACK# indicates co-processor that data transfer has started.
- Busy# indicates processor extension is busy with allotted job. Busy goes low indicates 80286 to suspend the execution and wait until busy becomes high.
- Error# signal indicates the error is detected by the co-processor. So when low, 80286 need to perform the processor extension interrupt while executing WAIT instruction.
- The **lock** pin is an output controlled by the LOCK prefix. If an instruction is prefixed with LOCK, the pin becomes a logic 0 for the duration of the locked instruction.
- S0 and S1 are bus cycle status output pin
- M/IO is memory I/O select pin
- COD/INTA# is code interrupt acknowledgement pin
- HOLD is bus hold input signal and HLDA/HOLDA is hold acknowledgement output signal

10. **Features of 80386**

**Ans**
- The 80386 microprocessor is an enhanced version of the 80286 microprocessor
- Memory-management unit is enhanced to provide memory paging.
- The 80386 also includes 32-bit extended registers and a 32-bit address and data bus. These extended registers include EAX, EBX, ECX, EDX, EBP, ESP, EDI, ESI, EIP and EFLAGS.
- The 80386 has a physical memory size of 4GBytes that can be addressed as a virtual memory with up to 64TBytes.
- The 80386 is operated in the pipelined mode, it sends the address of the next instruction or memory data to the memory system prior to completing the execution of the current instruction. This allows the memory system to begin fetching the next instruction or data before the current is completed. This increases access time.
- The instruction set of the 80386 is enhanced to include instructions that address the 32-bit extended register set.
- The 80386 memory manager is similar to the 80286, except the physical addresses generated by the MMU are 32 bits wide instead of 24-bits.
- The concept of paging is introduced in 80386
- 80386 support three operating modes:
  1. Real Mode (default)
  2. Protected Virtual Address Mode (PVAM)
  3. Virtual Mode
The memory management section of 80386 supports virtual memory, paging and four levels of protection.

- The 80386 includes special hardware for task switching.

### 11. Explain 80386 Architecture

**Ans**
- The internal architecture of the 80386 includes six functional units that operate in parallel. The parallel operation is called as pipeline processing.
- Fetching, decoding execution, memory management, and bus access for several instructions are performed simultaneously.
- The six functional units of the 80386 are:
  1. Bus Interface Unit
  2. Code Pre-fetch Unit
  3. Instruction Decoder Unit
  4. Execution Unit
  5. Segmentation Unit
  6. Paging Unit

![80386 Architecture Diagram]

- The **Bus Interface Unit** connects the 80386 with memory and I/O. Based on internal requests for fetching instructions and transferring data from the code pre-fetch unit, the 80386 generates the address, data and control signals for the current bus cycles.
- The **code pre-fetch unit** pre-fetches instructions when the bus interface unit is not executing the bus cycles. It then stores them in a 16-byte instruction queue for decoding by the instruction decoder.
The instruction decode unit translates instructions from the pre-fetch queue into micro-codes. The decoded instructions are then stored in an instruction queue (FIFO) for processing by the execution unit.

The execution unit processes the instructions from the instruction queue. It contains a control unit, a data unit and a protection test unit.

The control unit contains microcode and parallel hardware for fast multiply, divide and effective address calculation. The unit includes a 32-bit ALU, 8 general purpose registers and a 64-bit barrel shifter for performing multiple bit shifts in one clock. The data unit carries out data operations requested by the control unit.

The protection test unit checks for segmentation violations under the control of microcode.

The segmentation unit calculates and translates the logical address into linear addresses at the request of the execution unit.

The translated linear address is sent to the paging unit. Upon enabling the paging mechanism, the 80386 translates these linear addresses into physical addresses. If paging is not enabled, the physical address is identical to the linear address and no translation is necessary.

11. **Register organization of 80386**

**Ans** The Register organization of 80386 is as follows:

![Figure:80386 General Purpose, Index and Pointer Register](image)

**General Purpose Register**

- Registers EAX, EBX, ECX, EDX, EBP, EDI and ESI are regarded as general purpose or multipurpose
- EAX (ACCUMULATOR): The accumulator is used for instructions such as multiplication, division and some of the adjustment instructions. In 80386 and above, the EAX register may also hold the offset address of a location in memory system.

- EBX (BASE INDEX): This can hold the offset address of a location in the memory system in all version of the microprocessor. It the 80386 and above EBX also can address memory data.

- ECX (count): This acts as a counter for various instructions.

- EDX (data): EDX is a general-purpose registers that holds a part of the result for multiplication or part of the division. In the 80386 and above this register can also address memory data.

**Pointer and Index Register**

- EBP (Base Pointer): EBP points to a memory location in all version of the microprocessor for memory data transfers.

- ESP (Stack Pointer): ESP addresses an area of memory called the stack. The stack memory is a data LIFO data structure. The register is referred to as SP if used in 16 bit mode and ESP if referred to as a 32 bit register.

- EDI (Destination index): EDI often addresses string destination data for the string instruction. It also functions as either a 32-bit (EDI) or 16-bit (DI) general-purpose register.

- ESI (Source index): ESI can either be used as ESI or SI. It is often used to the address source string data for the string instructions. Like EDI ESI also functions as a general-purpose registers.

**Figure: 80386 Segment Register**

- **CS (Code):** The code segment is a section of memory that holds the code used by the microprocessor. The code segment registers defines the starting address of the section of memory holding code.

- **SS (Stack):** The stack segment defines the area of memory used for the stack. The stack entry point is determined by the stack segment and stack pointer registers. The BP registers also addresses data within the stack segment.

- **DS (Data)** – The data section contains most data used by a program. Data are accessed in the data segment by an offset address of the contests of other registers that hold the offset address.

- **ES (extra)** – The extra segment is used to hold information about string transfer and manipulation

- **FS and GS** – These are supplement segment registers available in the 80386 and above microprocessors to allow two additional memory segments for access by programs.
**EIP (Instruction Pointer):** EIP addresses the next instruction in a section of memory defined as a code segment. This register is IP (16bit) when microprocessor operates in the real mode and EIP (32 bits) when 80386 and above operate in protected mode.

![Figure:80386 Instruction Pointer and Flag Register](image)

**Flag Register:**
Indicates the condition of the microprocessor and controls its operations. Flag registers are also upward compatible since the 8086-80268 have 16bit registers and the 80386 and above have EGLAF register (32 bits)

![Figure: 80386 Flag Register](image)

- **IOPL (I/O Privilege level):** IOPL is used in protected mode operation to select the privilege level for I/O devices. If the current privilege level is higher or more trusted than the IOPL, I/O executed without hindrance. If the IOPL is lower than the current privilege level, an interrupt occurs, causing execution to suspend. Note that an IPOL is 00 is the highest or more trusted; if IOPL is 11, it’s the lowest or least trusted.
- **NT (Nested Task):** The nested task flag is used to indicate that the current task is nested within another task in protected mode operation. This flag is when the task I nested by software.
- **RF (Resume):** The resume flag is used with debugging to control the resumption of execution after the next instruction.
- **VM (Virtual Mode):** The VM flag bit selects virtual mode operation in a protected mode system.
- **Note:** All the other flag bit is having similar description as in 8086 flag register.

**System Address Register:**
Four memory management registers are used to specify the locations of data structures which control segmented memory management.

- GDTR (Global Descriptor Table Register) and IDTR (Interrupt Descriptor Table Register) be loaded with instructions which get a 6 byte data item from memory.
- LDTR (Local Descriptor Table Register) and TR (Task Register) can be loaded with instructions which take a 16-bit segment selector as an operand.

**Special 80386 Register**
- Control Register: Four Control Register (CR0-CR3)
- Debug Register: Eight Debug Register (DR0-DR7)
- Test Register: Two Test Register (TR6-TR7)

12. Explain Control Register in detail. OR
Describe the control registers of 80386 with proper meaning of each field.

**Ans**
- The 80386 has three 32 bit control registers CR0, CR2 and CR3 to hold global machine status independent of the executed task.
- Load and store instructions are available to access these registers
- Common tasks performed by control registers include interrupt control, switching the addressing mode, paging control, and coprocessor control.

**CR0:**
It 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. Resetting PE returns to real-
address mode.

- **MP (Math Present, bit 1)**
  MP controls the function of the WAIT instruction, which is used to coordinate a coprocessor.

- **EM (Emulation, bit 2)**
  EM indicates whether coprocessor functions are to be emulated.

- **TS (Task Switched, bit 3)**
  The processor sets TS with every task switch and tests TS when interpreting coprocessor instructions.

- **ET (Extension Type, bit 4)**
  ET indicates the type of coprocessor present in the system (80287 or 80387).

- **PG (Paging, bit 31)**
  PG indicates whether the processor uses page tables to translate linear addresses into physical addresses.

**CR2:**

- **CR2 is used for handling page faults when PG is set.** The processor stores in CR2 the linear address that triggers the fault.

- **Contains a value called Page Fault Linear Address (PFLA).** When a page fault occurs, the address the program attempted to access is stored in the CR2 register.

**CR3:**

- **It is used when virtual addressing is enabled; hence when the PG bit is set in CR0, CR3 enables the processor to translate linear addresses into physical addresses by locating the page directory and page tables for the current task.**

- **Typically, the upper 20 bits of CR3 become the page directory base register (PDBR), which stores the physical address of the first page directory entry.**

*Note: Reserved field shown in figure are kept reserved by manufacturer*

### 13. Explain Debug Register in detail.

**Ans** Debugging of 80386 allows **data access** breakpoints as well as **code execution** breakpoints.

- 80386 contains 6 debug registers to specify 4 breakpoints
  - Breakpoint Control options
  - Breakpoint Status
Linear Breakpoint Address Registers:
- The breakpoint addresses specified are 32-bit linear addresses
- While debugging, Intel 386 h/w continuously compares the linear breakpoint addresses in DR0-DR3 with the linear addresses generated by executing software.

Debug Control Register:

- \( \text{LEN}_i (i=0 - 3) \): Breakpoint Length Specification Bits:
  - 2 bit field for each breakpoint
  - Specifies length of breakpoint fields
  - The choices of data breakpoints are 1byte, 2bytes & 4bytes
  - For instruction execution breakpoint, the length is 1(beginning byte address)
Table: LEN<sub>i</sub> Encoding

- **RW<sub>i</sub>**<sub>(i=0 - 3)</sub>: Memory Access Qualifier Bit
  - 2 bit field for each breakpoint
  - Specifies the type of usage which must occur inorder to activate the associated breakpoint

Table: RW<sub>i</sub> Encoding

- **GD**: Global Debug Register Access Detect
  - Debug registers can only be accessed in real mode or at privilege level 0 in protected mode
  - GD bit, when set, provides extra protection against any Debug Register access even in Real Mode or at privilege level 0 in Protected Mode.
  - The GD bit, when set, causes an **exception 1** fault if an instruction attempts to read or write any Debug Register.
  - The GD bit is then automatically cleared when the exception 1 handler is invoked, allowing the exception 1 handler free access to the debug registers.

- **GE and LE bit**: Exact data breakpoint match, global and local
- **G<sub>i</sub>** and **L<sub>i</sub>**<sub>(i=0 - 3)</sub>: Breakpoint Enable, global and local
o If either G_i or L_i is set then the associated breakpoint is enabled.

**Debug Status Register:**
A Debug Status Register allows the exception 1 handler to easily determine why it was invoked. It can be invoked as a result of one of several events:

1) DR0 Breakpoint fault/trap.
2) DR1 Breakpoint fault/trap.
3) DR2 Breakpoint fault/trap.
4) DR3 Breakpoint fault/trap.
5) Single-step (TF) trap.
6) Task switch trap.
7) Fault due to attempted debug register access when GD = 1.

**Figure:** Debug Status Register

- B_i: Debug fault/trap due to breakpoint 0 - 3
  - Four breakpoint indicator flags, B0-B3, correspond one-to-one with the breakpoint registers in DR0-DR3.
  - A flag B_i is set when the condition described by DR_i, LEN_i, and RW_i occurs.
- BD: Debug fault due to attempted register access when GD bit is set
  - This bit is set if the exception 1 handler was invoked due to an instruction attempting to read or write to the debug registers when GD bit was set.
- BS: Debug trap due to single step
  - This bit is set if the exception 1 handler was invoked due to the TF bit in the flag register being set.
- BT: Debug trap due to task switch
  - This bit is set if the exception 1 handler was invoked due to a task switch occurring to a task having an Intel386.

14. Explain Test Register in detail.

**Ans**
- Two more test register are provided by 80386 for page cacheing.
- They are used to control the testing of Translation Look-aside Buffer (TLB) of Intel386.
- TR6 is the command test register
- TR7 is the data register which contains the data of Translation Look-aside buffer test.

**Figure:** Test Register

15. What are program-invisible registers?

**Ans**
- The global and local descriptor tables are found in the memory system.
- In order to access and specify the address of these tables, the program invisible registers used.
- The program invisible registers are not directly addressed by software so they are given name.
- The GDTR (global descriptor table register) and IDTR (interrupt descriptor table register) contain the base addresses of the descriptor table and its limit.
- The limit of each descriptor table is 16 bits because the maximum table length is 64 Kbytes.
- When the protected mode operation is desired, the address of the global descriptor table and its limit are loaded into the GDTR.

16. **Explain Real Modes of 80386**

**Ans**
- Default Mode
- After reset, the 80386 starts from the memory location FFFFFFF0 H under real address mode.
- In real address mode, 80386 works as a fast 8086 with 32 bit registers and data types.
- Real-address mode is in effect after a signal on the RESET pin. Even if the system is going to be used in protected mode, the start-up program will execute in real-address mode temporarily while initializing for protected mode.
- The addressing techniques, memory size, interrupt handling in this mode of 80386 are similar to the real addressing mode of 80286.
- In real address mode, the default operand size is 16 bit but 32 bit operands and addressing modes may be used with the help of override prefixed.
- Maximum physical memory = 1Mega byte (1MB)
- The only way to leave real-address mode is to switch to protected mode.

17. **Explain PVAM of 80386**

**Ans**
- Introduced in the 80386 processor.
- 32-bit address bus => access up to 232 bytes = 22. 230 B = 4 GB
- Base address => 32-bit value
- Offset =>16-bit or 32-bit value
- Linear address = base address + offset
- Linear address → physical address with paging
- In protected mode, the segment registers contain an index into a table of segment descriptors.
- Each segment descriptor contains the start address of the segment, to which the offset is added to generate the address.
- In addition, the segment descriptor contains memory protection information.
- This includes an offset limit and bits for write and read permission.
- This allows the processor to prevent memory accesses to certain data.
- Protected mode is accessed by placing a logic 1 into the PE bit of CR0
- This system contains one data segment descriptor and one code segment descriptor with each segment set to 4G bytes in length.
- PVAM mode support memory management, virtual memory, multitasking, protection, debugging, segmentation and paging.
18. **Explain Virtual Mode of 80386**

**Ans**
- In its protected mode of operation, 80386DX provides a virtual 8086 operating environment to execute the 8086 programs.
- The real mode can also be 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 enabled in virtual mode, but may be needed to run the 8086 programs which require more than 1Mbytes 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.
- 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.
19. Explain Segment Selector in 80386

Ans: In protected mode, the contents of segment register is called selectors shown below:

![Selector Diagram]

- **Descriptor Index** – Selects any one of the $2^{13}$ descriptors from a descriptor table.
- **TI** – Table Indicator
  - TI = 1 Local Descriptor Table
  - TI = 0 Global Descriptor Table
- **RPL** (Requested Privilege Level), 2-bit, refers to the privilege of that segment. They are used while protection checks to indicate whether access to segment is allowed or not allowed.

20. Explain 80386 Descriptor.

Ans: Need of Descriptor

- Segments are the memory area defined by programmer and can be code, stack and data segment.
- 80386 segments are also assigned to have attributes viz. privilege level, segment type, segment limit.
- So, it’s not possible to use a 16-bit segment register to represent all the info regarding a segment. Therefore, the solution is Descriptor.
- Segment sizes vary from 1 byte to 4 GB in 80386.

![80386 Descriptor Diagram]

Figure: 80386 Descriptor
### Offset | Bit | Description |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>A(Accessed)</td>
<td>80386 sets this bit to 1 whenever a memory reference to a segment defined by this descriptor is made.</td>
</tr>
<tr>
<td>41-43</td>
<td>TYPE</td>
<td>Describes the type of segment</td>
</tr>
</tbody>
</table>
| 44 | S(System) | 0 – Specifies system segment descriptor  
1 – Specifies non system segment descriptor  
E.g. code, stack or data |
| 45-46 | DPL (Descriptor Privilege level) | Indicates privilege level associated with memory space. 0 - most privileged 3 – least privileged |
| 47 | Present | If set 0 , indicates that the address range that is specified by the descriptor is temporarily not present. |
| 52 | U(User) | Undefined bit. |
| 53 | X | Reserved by Intel |
| 54 | D(Default Size) | 0 – Operands within this segments are assumed to be 16 bits.  
1 – Operands assumed to be 32 bits |
| 55 | G(Granularity bit) | 0 – Unit of limit field is 1 byte.  
1 – Unit of limit field is of 4096 bytes (or 1 page). |

- The Above figure is the format of the code/data descriptor; one descriptor is 64-bit long.
- As we can see, a descriptor actually includes a 32-bit base address and a 20-bit limit and some attributes, the 32-bit base address indicate where the segment starts, and the 20-bit limit indicates the length of the segment.
- However, a problem comes up, 20-bit limit can only represent \(2^{20} = 1MB\) memory, to access a 4GB memory space, descriptor uses G bit to indicate whether the limit use 4K or 1 byte for one unit, that means if G bit is set then we get \(2^{20} \times 4K = 4GB\) memory, if it is unset then we only use a memory space under 1MB.

21. **Explain Descriptor tables in 80386**

**Ans**
- The segment descriptors that we defined are grouped together and placed in a continuous memory location. The group arrangement called Descriptor Table.
- Each descriptor requires 8byte in order to store the info of a particular segment.
- Descriptor table can contain 8192(8K) descriptors at the max.
- The maximum length of a descriptor table is a 64Kbytes.
- The descriptor tables define all the segments used in the 80386 when it operates in the protected mode.
mode.
- There are three types of descriptor tables: the global descriptor table (GDT), the local descriptor table (LDT), and the interrupt descriptor table (IDT).
- The registers used by the 80386 to address these three tables are called the global descriptor table register (GDTR), the local descriptor table register (LDTR), and the interrupt descriptor table register (IDTR).
- These registers are loaded with the LGDT, LLDT, and LIDT instructions, respectively.
- The local and global descriptor tables hold up to 8192 entries each, and the interrupt descriptor table holds up to 256 entries.
- A descriptor is indexed from either the local or global descriptor table by the selector that appears in a segment register.
- Whenever a new selector is placed into one of the segment registers, the 80386 accesses one of the descriptor tables and automatically loads the descriptor into a program-invisible cache portion of the segment register.

**Global Descriptor Table (GDT):**
- This is the main table of descriptors.
- The same GDT can be used by all programs to refer to segment of memory.
- 80386 can have many LDT’s but only one GDT.

![Figure: Global Descriptor Table](image)

**Local Descriptor Table (LDT):**
- A multi tasking system is defined on a per task basis.
- The main purpose of an LDT would be combined with GDT in order to expand the total number of available descriptors.
- Generally, each task can have its own LDT and can also be shared with other task.

22. **Differentiate real mode interrupts and protected mode interrupts**

**Ans**
- The difference between real mode interrupts and protected mode interrupts is that the interrupt vector table is an IDT in the protected mode.
- The IDT still contains up to 256 interrupt levels, but each level is accessed through an interrupt
gate instead of an interrupt vector.
- Thus, interrupt type number 2 (INT 2) is located at IDT descriptor number 2 at 16 locations above the base address of the IDT.
- This also means that the first 1K byte of memory no longer contains interrupt vectors, as it did in the real mode.
- The IDT can be located at any location in the memory system.

23. Explain the basic concepts of 80386 paging system.
Ans
- Paging is one of the memory management techniques
- The paging mechanism allows any linear (logical) address, as it is generated by a program, to be placed into any physical memory page, as generated by the paging mechanism.
- A linear memory page is a page that is addressed with a selector and an offset in either the real or protected mode.
- A physical memory page is a page that exists at some actual physical memory location.
- For example, linear memory location 20000H could be mapped into physical memory location 30000H, or any other location, with the paging unit. This means that an instruction that accesses location 20000H actually accesses location 30000H. Each 80386 memory page is 4K bytes long.
- Paging allows the system software to be placed at any physical address with the paging mechanism.
- Three components are used in page address translation: the page directory, the page table, and the actual physical memory page.
- 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.
- The pages are just fixed size portions of the program module or data.
Advantage:
- The advantage of paging scheme is that the complete segment of a task need not be in the physical memory at any time.
- Only a few pages of the segments, which are required currently for the execution need to be available in the physical memory.
- Thus the memory requirement of the task is substantially reduced, relinquishing the available memory for other tasks.

24. Explain the Page Table and Page Directory Entry with example.
Ans
- Page Directory
- The page directory contains the location of up to 1024 page translation tables, which are each four bytes long.
- Each page translation table translates a logical address into a physical address.
- The page directory is stored in the memory and accessed by the page descriptor address register (CR3).
- Control register CR3 holds the base address of the page directory, which starts at any 4K-byte boundary in the memory system.
Each entry in the page directory translates the leftmost 10 bits of the memory address. This 10-bit portion of the linear address is used to locate different page tables for different page table entries.

**Page Table**
- The page table contains 1024 physical page addresses, accessed to translate a linear address into a physical address.

**Page Directory Entry**
- Total Page Directory Entries are 1024
- Each directory entry is of 4 byte

![Page Directory Entry](image)

**Page Table Entry**
- The page table entries contain the starting address of the page and the statistical information about the page.
- Total Entries are 1024
- Each page table entry is of 4 byte

![Page Table Entry](image)

- **D-bit**: Dirty bit is undefined for page table directory entries by the 80386 microprocessor and is provided for use by the operating system.
- **A-bit**: Access bit is set to a logic 1 whenever the microprocessor accesses the page directory entry.
- **R/W and Read/write and user/supervisor** are both used in the protection scheme. Both bits combine to develop paging priority level protection for level 3, the lowest user level.

<table>
<thead>
<tr>
<th>U/S</th>
<th>R/W</th>
<th>Access Level 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>None</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>None</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Read-only</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Write-only</td>
</tr>
</tbody>
</table>

- **P-bit**: Present bit, if logic 1 indicates that the entry can be used in address translation. If P = 0, the entry cannot be used for translation. When P = 0, the remaining bits of the entry can be used to
indicate the location of the page on the disk memory system.

**Difference between page directory and page table entry:**
- The main difference is that the page directory entry contains the physical address of a page table, while the page table entry contains the physical address of a 4K-byte physical page of memory.
- The other difference is the D (dirty bit), which has no function in the page directory entry, but indicates that a page has been written to in a page table entry.

25. **Explain Page Translation Mechanism in 80386**

Ans - A page frame is a 4K-byte unit of contiguous addresses of physical memory. Pages begin on byte boundaries and are fixed in size.
- A linear address refers indirectly to a physical address by specifying a page table, a page within that table, and an offset within that page.

![Figure: Format of Linear Address](image)

- The below figure of page translation shows how processor converts the DIR, PAGE, and OFFSET fields of a linear address into the physical address by consulting two levels of page tables.
- The addressing mechanism uses the DIR field as an index into a page directory, uses the PAGE field as an index into the page table determined by the page directory, and uses the OFFSET field to address a byte within the page determined by the page table.
- In the second phase of address transformation, the 80386 transforms a linear address into a physical address.
- This phase of address transformation implements the basic features needed for page-oriented virtual-memory systems and page-level protection.
- Page translation is in effect only when the PG bit of CR0 is set.

![Figure: Page Translation](image)
26. Explain Translation Look aside Buffer (TLB)

ANS

- 80386 supports demand paged virtual memory systems.
- Performance degrades if the processor access two levels of tables for every memory reference.
- To solve this problem, the Intel386 DX keeps a cache of the most recently accessed pages and this cache is called Translation Look aside Buffer (TLB).
- TLB has 4 sets of eight entries each.
- Each entry consists of a TAG and a DATA.
- Tags are 24 bit wide. They contain 20 upper bits of linear address, a valid bit (Validation of Entry) and three attribute bits (D,U/S and R/W).
- Data portion of each entry contains higher 20 bits of the Physical address.
- It automatically keeps the most commonly used Page Table Entries.
- TLB cache holds recent 32 entries of Page Table.
- 32-entry TLB coupled with a 4K page size results in the coverage of 128KB of memory addresses.

![Figure: Translation Look aside Buffer (TLB)](image)

27. Protection Mechanism in 80386

Ans

- The purpose of the protection features of the 80386 is to help detect and identify bugs.
- To help debug applications faster and make them more robust in production, the 80386 contains mechanisms to verify memory accesses and instruction execution for conformance to protection criteria.
- Protection in the 80386 has five aspects:
  1. Type checking
  2. Limit checking
  3. Restriction of addressable domain
4. Restriction of procedure entry points
5. Restriction of instruction set

- The protection hardware of the 80386 is an integral part of the memory management hardware.
- Protection applies both to segment translation and to page translation.
- Each reference to memory is checked by the hardware to verify that it satisfies the protection criteria.
- All these checks are made before the memory cycle is started; any violation prevents that cycle from starting and results in an exception.
- Since the checks are performed concurrently with address formation, there is no performance penalty.

Segment-Level Protection:
- All five aspects of protection apply to segment translation:
  1. Type checking:
     It is used to determine whether the current memory access (read/write) is allowed. For example, write operation is not allowed for the read only data segment.
  2. Limit checking:
     Limit checking uses the twenty limit bits stored in the segment descriptor to guarantee that addresses outside the range of the segment are not allowed. The limit field of a segment descriptor is used by the processor to prevent programs from addressing outside the segment.
  3. Restriction of addressable domain:
     The addressable domain of a task is a function of the task’s CPL (current privilege level). A task can access data segments of only same or lower (greater in magnitude) privilege levels.
  4. Restriction of procedure entry points:
     It is done through the use of Call gate. Call gates are used to control the transfer of execution between procedures of different privileged levels.
  5. Restriction of instruction set:
     The instructions that affect system data structures can only be executed when CPL is zero. If the CPU encounters one of these instructions when CPL is greater than zero, it signals a general protection exception.

- The segment is the unit of protection, and segment descriptors store protection parameters.
- Protection checks are performed automatically by the CPU when the selector of a segment descriptor is loaded into a segment register and with every segment access.
- Segment registers hold the protection parameters of the currently addressable segments.

Page-Level Protection
- Page-Table entries hold protection parameters (i.e. U/S and R/W bit)
- Two kinds of protection are related to pages:
  1. Restriction of addressable domain:
     The concept of privilege for pages is implemented by assigning each page to one of two
levels:
  i. Supervisor level (U/S=0) -- for the operating system and other systems software and related data.
  ii. User level (U/S=1) -- for applications procedures and data.

The current level (U or S) is related to CPL. If CPL is 0, 1, or 2, the processor is executing at supervisor level. If CPL is 3, the processor is executing at user level. When the processor is executing at supervisor level, all pages are addressable, but, when the processor is executing at user level, only pages that belong to the user level are addressable.

2. Type checking:
   At the level of page addressing, two types are defined:
   i. Read-only access (R/W=0)
   ii. Read/write access (R/W=1)

   When the processor is executing at supervisor level, all pages are both readable and writable. When the processor is executing at user level, only pages that belong to user level and are marked for read/write access are writable; pages that belong to supervisor level are neither readable nor writable from user level.

28. **Explain the concepts of Call gates with suitable example.**

   **Ans**
   - Call Gates are required at procedure entry point check.
   - Call gates allow a program to directly call system calls.
   - However, since system calls are often in a privileged ring, calling them directly is not allowed because of the ring protection. The 80386 protection mode uses a call-gate concept to allow this kind of transfer.
   - Call gates enable programs in a lower privileged ring to jump to designated places in a higher privileged ring.
   - It is used to control the transfer of execution between procedures of different privilege level.
80386 do allow a program to jump to a more privilege ring, but a program cannot jump to an arbitrary place, it must go through Call Gates, which basically define the entry points for the privileged code.

- Corresponding security checks will be conducted at those entry points to decide whether the invoking code has sufficient right.

- These security checks are enforced by operating systems.

- Like segment descriptors, call-gate entries (call-gate descriptors) are also stored in the GDT (or LDT) tables.

- Gates define an entry point of a procedure.

- Call-Gate Descriptor contains the following information:

  - Segment selector: Code segment to be accessed
  - Offset in segment: Entry point for a procedure in the specified code segment
  - DPL: Privilege level required for a caller trying to access the procedure
  - Parameter Count: if a stack switch occurs, it specifies the number of optional parameters to be
copied between stacks.

**How to use call gates?**
- Can be invoked through Call * or JMP * instructions.
- ‘*’ specifies the call gate entry in the GDT (or LDT) table
- From the table, the entry point of the procedure will be obtained.
- DPL of the gate descriptor allows the CPU to decide whether the invocator can enter the gate.

**Access Control Policy for Call Gates**
- CPL <= DPL of the call gate.
- For CALL: DPL of the code segment <= CPL (only calls to the more privileged code segment are allowed).
- For JMP: DPL of the code segment = CPL.

**29. Explain Access Control on Data and Code Access**

**Ans**

**Privilege Check for Data Access (Figure a):**
- Policy: CPL <= DPL of code segment.
- A subject can only access data objects with the same or lower privilege levels.

**Privilege Check for Control Transfer without Using a Gate (Figure b):**

**Policy**
- For non-conforming segment: transfer is allowed when CPL = DPL.
- For conforming segment: transfer is allowed when CPL >= DPL.
- RPL does not have much effect here.

**Why can’t we access code with a higher DPL (i.e., lower privilege)?**
- Possible reason 1: It is easy to jump (lower the CPL) to the code with higher DPL, but it is difficult to return back, because on returning, we jump from a lower privileged ring to a higher privileged ring. This violates the mandatory access control policy.
Possible reason 2:
Another reason is the data access. If a code A jumps to another code B at a lower privilege level, B cannot access A's data because the data are most likely in A's ring level.

Possible reason 3: Is there really a need to allow jumping from a higher privilege to a lower privilege?

Why can't we jump to code with a lower DPL (i.e., higher privilege)?
- For security reasons, we cannot do this.

How can we achieve jumping to lower DPL?
- Gates are designed for this purpose.

30. Multitasking in 80386

Ans
- Protected mode has the ability to support execution of multiple programs (called tasks) simultaneously.

![Multitasking in 80386](image)

Running multiple tasks simultaneously

To provide efficient, protected multitasking, the 80386 employs several special data structures.
- It does not, however, use special instructions to control multitasking; instead, it interprets ordinary control-transfer instructions differently when they refer to the special data structures.
- The registers and data structures that support multitasking are:
  i. Task state segment
  ii. Task state segment descriptor
  iii. Task register
  iv. Task gate descriptor

With these structures the 80386 can rapidly switch execution from one task to another, saving the context of the original task so that the task can be restarted later.

31. Explain Task State Segment and Task Descriptor.

Ans
- Task State Segment
  - During a task switch, the contents of all processors registers, as well as other information, are saved for the task being suspended and new information is loaded for the task.
  - This information is saved in a special memory structure called task state segment.
  - Task state segments may reside anywhere in the linear space.
Task Descriptor
- The BASE, LIMIT, and DPL fields and the G-bit and P-bit have functions similar to their counterparts in data-segment descriptors.
- A procedure that has access to a TSS descriptor can cause a task switch.
- TSS descriptors may reside only in the GDT

![Task Descriptor](image)

32. Explain Task Register
Ans - When multiple TSS descriptors exist in the GDT, the TSS currently in use is accessed through the use of task register.
- The task register (TR) identifies the currently executing task by pointing to the TSS.

![Task Register](image)

- 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 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.

33. Explain Task Gate and Task Gate Descriptor.
A task gate descriptor provides an indirect, protected reference to a TSS.

- The SELECTOR field of a task gate must refer to a TSS descriptor.
- The DPL field of a task gate controls the right to use the descriptor to cause a task switch.
- A procedure may not select a task gate descriptor unless the maximum of the selector's RPL and the CPL of the procedure is numerically less than or equal to the DPL of the descriptor.
- This constraint prevents untrusted procedures from causing a task switch.
- A procedure that has access to a task gate has the power to cause a task switch.
- Task gates may also reside in the IDT, making it possible for interrupts and exceptions to cause task switching. When interrupt or exception vectors to an IDT entry that contains a task gate, the 80386 switches to the indicated task.
- Thus, all tasks in the system can benefit from the protection afforded by isolation from interrupt tasks.

34. **Explain Context Switching.**

**Ans**

- Switching from one task to another is accomplished with four different ways:
  1. The current task jmps or CALLs a TSS descriptor.
  2. The current task jmps or CALLs a task gate.
  3. The current task executes a IRET.
  4. An interrupt or exception selects a task gate.

**When a context switch is called following tasks take place:**

- The new TSS descriptor or task gate must have sufficient privilege to allow a task switch.
- The DPL, CPL and RPL values are compared before any further processing takes place.
- The new TSS descriptor must have its present P bit set and have a valid limit field.
- The state of current task also called context is saved. This involves copying the contents of all processor register into the TSS for the current task.
- The task register is loaded with the selector of new TSS descriptor.
- The busy bit in the new TSS descriptor is set as TS bit in CR0 by CLTS (Clear task switched flag) instruction.

35. **Explain protection of I/O in protected mode of 80386.**

**Ans**

Two mechanisms provide protection for I/O functions:

1. The IOPL field in the EFLAGS register defines the right to use I/O-related instructions.
2. The I/O permission bit map of a 80386 TSS segment defines the right to use ports in the I/O address space.
- These mechanisms operate only in protected mode, including virtual 8086 mode; they do not operate in real mode.
- In real mode, there is no protection of the I/O space; any procedure can execute I/O instructions, and any I/O port can be addressed by the I/O instructions.

**I/O Privilege Level (IOPL):**
- Instructions that deal with I/O need to be restricted but also need to be executed by procedures executing at privilege levels other than zero.
- For this reason, the processor uses two bits of the flags register to store the I/O privilege level (IOPL).
- The IOPL defines the privilege level needed to execute I/O-related instructions.
- I/O instructions can be executed only if CPL <= IOPL

**I/O Permission Bit Map:**
- The I/O operations are allowed on a port by port basis via permission bits stored in the I/O permission bit map section of the task’s TSS.
- Each byte in the bit map stores permission bits for eight consecutive ports.

![Sample I/O permission Bit Map](image)

- The I/O permission map is a bit vector. The size of the map and its location in the TSS segment are variable.
- The processor locates the I/O permission map by means of the I/O map base field in the fixed portion of the TSS.
- The I/O map base field is 16 bits wide and contains the offset of the beginning of the I/O permission map.
- Because the I/O permission map is in the TSS segment, different tasks can have different maps.
- Thus, the operating system can allocate ports to a task by changing the I/O permission map in the task’s TSS.
36. Exceptions and Interrupts in 80386.

**Ans**
- Interrupts and exceptions are special kinds of control transfer; they work somewhat like unprogrammed CALLs.
- They alter the normal program flow to handle external events or to report errors or exceptional conditions.
- The difference between interrupts and exceptions is that interrupts are used to handle asynchronous events external to the processor, but exceptions handle conditions detected by the processor itself in the course of executing instructions.

**There are two sources for external interrupts and two sources for exceptions:**

1. **Interrupts**
   - Mask able interrupts, which are signalled via the INTR pin.
   - Non-mask able interrupts, which are signalled via the NMI (Non-Mask able Interrupt) pin.

2. **Exceptions**
   - Processor detected. These are further classified as faults, traps, and aborts.
   - Programmed. The instructions INTO, INT and BOUND can trigger exceptions. These instructions are often called "software interrupts", but the processor handles them as exceptions.

37. Explain IDT in detail.

**Ans**
- Real mode uses a 1 KB interrupt vector table and protected mode uses interrupt descriptor table IDT to support interrupt and exceptions.
- The IDT comprises 8-byte gate descriptors for task, trap or interrupt gates.
- It has maximum $2^8$ (256) descriptors.
- Like the GDT and LDTs, the IDT is an array of 8-byte descriptors.
IDT is 32-bit offset points to the first instruction in the handler’s code segment for trap and interrupt gates.

38. **Explain IDT Descriptors.**

**Ans** The IDT may contain any of three kinds of descriptor:

i. Task gates  
ii. Interrupt gates  
iii. Trap gates

The Figure below shows the format of task gates and 80386 interrupt gates and trap gates.
39. **Features of 80486**

**Ans**
- The 32-bit 80486 is the next evolutionary step up from the 80386.
- One of the most obvious feature included in a 80486 is a built in math coprocessor. This coprocessor is essentially the same as the 80387 processor used with a 80386, but being integrated on the chip allows it to execute math instructions about three times as fast as a 80386/387 combination.
- 80486 is an 8Kbyte code and data cache.
- To make room for the additional signals, the 80486 is packaged in a 168 pin, pin grid array package instead of the 132 pin PGA used for the 80386.
- Operates on 25MHz, 33 MHz, 50 MHz, 60 MHz, 66 MHz or 100MHz.
- It consists of parity generator/checker unit in order to implement parity detection and generation for memory reads and writes.
- Supports burst memory reads and writes to implement fast cache fills.
- Three mode of operation: real, protected and virtual 8086 mode.
- The 80486 microprocessor is a highly integrated device, containing well over 1.2 million transistors.

40. **Explain Architecture of 80486 Processor.**

**Ans** The architecture of the 80486DX is almost identical to the 80386. Added to the 80386 architecture inside the 80486DX is a math coprocessor and an 8K-byte level 1 cache memory.
BIU:
- BIU generates address, data and control signals for a bus cycle it is supported with an additional parity detection/generation for memory reads and writes.
- During memory write operation, the 486 generates even parity bit for each byte outputs these bits.
- These bits will be stored in a separate parity memory bank.
- During read operation, stored parity bits will be read from the parity memory.
- 80486 checks the parities of data bytes read and compare them with the DP0 – DP3 signals and generates parity check error, if it occurs.

Instruction Pre-fetch Unit:
- It pre-fetches the instruction bytes in advance and holds them in a 32-byte code queue.

Instruction Decoder:
- Decodes the instructions in the queue and passes the control and protection test unit.

Execution Unit:
- Executes the instruction with the help of Barrel Shifter, ALU and Register bank.

Segmentation Unit and Paging Unit:
- They are part of MMU (which manages virtual memory of system). Helpful in generation of Physical Address.
- Work same as they work in 80386.

Floating Point Unit:
- Responsible for performing the floating point operations.
- The 80486 has a built in math co-processor. It performs the floating point operations.
- It executes math instructions 3 times faster than the 386/387 combination.
Cache Unit:
- 8KB cache
- Additional high speed cache memory provides a way of improving overall system performance.
- It contains the recently used instructions, data or both.
- The main aim is that the microprocessor unit access code and data in the cache most of time, instead from the main memory.

41. **Explain EFLAG Register Of The 80486.**

**Ans** The extended flag register EFLAG is illustrated in the figure below:

![EFLAG Register Of The 80486](image)

- The only new flag bit is the **AC alignment check**, used to indicate that the microprocessor has accessed a word at an odd address or a double word boundary.