MICROPROCESSOR - 210254 (2019 Pattern, Semester IV)
Full Answers to University Exam Questions

Q1 a) General Descriptor Format

In 80386, a segment descriptor is 8 bytes long and gives information about a memory segment.

Descriptor Format:

```
63                    32 | 31                    0

-----------------------------------------------------------

| Base 31:24 | G | D/B | 0 | AVL | Limit 19:16 | P | DPL | S | Type |

|   Base 23:16   | Base 15:0      | Limit 15:0                  |

-----------------------------------------------------------
```

Fields:

- **Base**: 32-bit address of segment start
- **Limit**: 20-bit size of segment
- **G (Granularity)**: 1 = 4KB units, 0 = bytes
- **D/B**: Operand size, 1 = 32-bit, 0 = 16-bit
- **AVL**: Available for OS use
- **P (Present)**: 1 = Segment is present
- **DPL**: Descriptor Privilege Level (0-3)
- **S**: Descriptor type (0 = system, 1 = code/data)
- **Type**: Segment type details

Q1 b) LGDT, SGDT, SIDT Instructions

- **LGDT (Load Global Descriptor Table)**: Loads GDTR with base and limit of the GDT.
- **SGDT (Store Global Descriptor Table)**: Stores the current GDTR content to memory.
- **SIDT (Store Interrupt Descriptor Table)**: Stores the IDTR content to memory.

Q1 c) Segment Translation in 80386

1. Logical Address = Selector:Offset
2. Selector used to access descriptor in GDT or LDT
3. Descriptor base + offset = Linear Address

4. If paging is enabled, linear address is converted to physical address using page tables.

Q2 a) Address Translation in 80386

1. **Logical Address** (Selector:Offset)
2. Selector points to descriptor in GDT/LDT → Gives segment base
3. Add base + offset → Linear Address
4. If paging is enabled:
   - Linear address split into Page Dir Index (10), Page Table Index (10), Offset (12)
   - Traverse page directory and table → get physical address

Q2 b) System & Non-System Descriptors

System Descriptors:

- Task State Segment (TSS)
- Local Descriptor Table (LDT)
- Call Gate
- Interrupt Gate
- Trap Gate
- Task Gate

Non-System Descriptors:

- Code Segment Descriptor
- Data Segment Descriptor
- Stack Segment Descriptor

Q2 c) General Selector Format

```
15    3 2  1  0

-------------------

| Index   | TI | RPL |

-------------------
```

- **Index**: Descriptor index in table
- **TI**: Table Indicator (0 = GDT, 1 = LDT)
- **RPL**: Requested Privilege Level (0-3)

Q3 a) Protection Mechanism in Paging

- Uses Page Table Entries (PTE)
- Each page has R/W, U/S (user/supervisor) flags
- Controls user/kernel access and read/write access
- CR3 register holds base of page directory

Q3 b) CPL, EPL, IOPL

- **CPL (Current Privilege Level)**: Taken from code segment selector, 0 (high) to 3 (low)
- **EPL (Effective Privilege Level)**: max(CPL, RPL)
- **IOPL (I/O Privilege Level)**: Defined in EFLAGS, determines I/O access rights

Q3 c) Need for Protection

- Prevent unauthorized memory access
- Separate user and kernel modes
- Enable secure multitasking
- Avoid accidental corruption

Q4 a) Control Transfer using Call Gate

- Call Gate is a descriptor that enables transfer of control between segments with privilege checking
- CALL instruction uses gate descriptor
- Transfers control to a procedure in a higher privilege segment

Q4 b) Privileged Instructions

- Instructions that require CPL = 0
  Examples:
- LGDT, LLDT, LIDT
- LTR (Load Task Register)
- MOV to/from control registers (CRx)
- HLT (Halt)
- CLI/STI (Interrupt flag control)

Q4 c) Combining Segment and Page Protection

- Segment-level: controls access to memory regions
- Page-level: fine-grained access control per 4KB page
- Combined: Segment defines access domain, paging enforces more granular rules

Q5 a) V86 Task Structure & Protection

- V86 = Virtual 8086 mode (in protected mode)
- Each task has:
  - 16-bit addressing
  - Own page table
  - Separate I/O permissions
- Protection:
  - Page-level protection
  - IOPL and TSS control access

Q5 b) Task State Segment (TSS)

- Contains processor state for task:
  - ESP, SS for privilege levels
  - CR3, EIP, EFLAGS
  - General registers
  - I/O map base
- Used during task switching

Q5 c) Entering & Leaving Virtual Mode

- **Entering:**
  - Set VM flag in EFLAGS (via IRET)
  - Initialize segment registers
- **Leaving:**
  - Clear VM flag
  - Use task switch or interrupt return

Q6 a) TSS Descriptor

- Describes a TSS segment in GDT
- Provides base, limit, privilege
- Used for task switching and I/O control

Q6 b) Features of V86 Mode

- 8086 compatibility
- 1MB memory access
- Page-level protection
- Supports multitasking
- I/O permission control via TSS

Q6 c) Task Switching

Steps:

1. Save current task state to current TSS
2. Load new task's TSS via Task Gate or JMP/CALL
3. Load all processor state from new TSS
4. Resume new task

Q7 a) Handling Interrupts in Protected Mode

- IDT used instead of IVT
- Gate descriptor used:
  - Interrupt gate, trap gate, or task gate
- Steps:

1. Check DPL and access rights
2. Push flags, CS, EIP
3. Load new CS, EIP from gate descriptor
4. Clear IF for interrupt gates

Q7 b) Types of Exceptions in 80386

- **Faults**: Can be corrected (e.g., Page Fault)
- **Traps**: Reported after instruction (e.g., Debug)
- **Aborts**: Severe errors (e.g., Machine Check)

Q7 c) Microcontroller Architecture (8051 example)

- ALU
- Registers (A, B, SP, DPTR)
- RAM and ROM
- Timers
- Serial port
- I/O ports
- Interrupt control

Q8 a) IDT Descriptors

- Each entry is 8 bytes
- Contains:
  - Offset (Low & High)
  - Selector
  - Type (Interrupt, Trap, Task Gate)
  - DPL, P flag

Q8 b) Exceptions

- **Divide Error**: Divide by zero
- **Invalid Opcode**: Undefined instruction
- **Overflow**: Arithmetic overflow (INTO instruction)

Q8 c) Features of 8051 Microcontroller

- 8-bit CPU
- 4KB ROM, 128B RAM
- 4 I/O ports
- 2 Timers/Counters

- Serial Communication
- Interrupts (5 sources)
- Bit and Byte level operations