Paging and Memory access in protected mode of 80386 Microprocessor

Presented By:
Alisha Bhusal
Anushila Basnet
Devendra Khadka

Paging

- Paging is one of the memory management techniques used for virtual memory multitasking operating system.

- It allows the use of virtual memory by translating virtual address into physical address.

- The advantage of paging scheme is that the complete segment of a task does not need to be in the physical memory at any time.

- Only a few pages of the segments, which are currently required for the execution need to be available in the physical memory.

Components of Paging in 80386:

1. **Page Directory:** The page directory contains 1024 entries, each pointing to a page table. It fits in one page (4 KB).
2. **Page Tables:** Each page table also contains 1024 entries, with each entry pointing to a page in physical memory.
3. **Pages:** The actual physical memory blocks, each typically 4 KB in size.

Benefits and Use Cases

- **Memory Isolation:** By using both segmentation and paging, protected mode can enforce memory isolation, preventing unauthorized access to memory.
- **Efficient Memory Usage:** Paging allows the physical memory to be used more efficiently by enabling non-contiguous allocation.
- **Security:** It enhances system security by ensuring that applications cannot interfere with the memory of the operating system or other applications.
- **System Stability:** Protected mode and paging contribute to system stability by preventing memory corruption and other faults.

Memory Access in Protected Mode

The 80386 uses a combination of segmentation and paging to manage memory efficiently and securely, ensuring that each program runs smoothly without interfering with others.

1. **Segment Registers and Selectors**

- Segment registers hold values called selectors.
- Selectors are like addresses that point to descriptions (called descriptors) of memory segments.
- These descriptors define the size (limit), starting point (base address), and access permissions for each segment of memory.

2. Calculating the Linear Address

- When a program needs to access memory, it provides an offset, which is an address within a segment.
- This offset is added to the segment's base address to get the linear address.
- Think of it as adding a street number (offset) to the beginning of a street (base address) to find a specific house (linear address).

3. Using the Linear Address

- If the paging system is off, the linear address is used directly as the physical address to access memory.
- If the paging system is on, the linear address goes through another step to get the physical address.

4. Paging Unit

- The paging unit is a special part of the processor that manages memory in pages.
- Paging is only active in Protected Mode.
- It translates the linear address into a physical address, allowing the use of virtual memory and providing additional protection.

Accessing Data in Protected Mode

Overview of Protected Mode

-In protected mode, the 80386 microprocessor has a 32-bit address bus, allowing it to address up to 4 GB of memory.
-Memory is organized into segments for code, data, and stack.
-Descriptor tables, such as the Global Descriptor Table (GDT) and Local Descriptor Table (LDT), are used to define the properties of these segments.

Enabling Protected Mode

1. **Initialize the GDT:** Define segment descriptors for the code, data, and stack segments.
2. **Load GDTR:** Use the LGDT instruction to load the GDT register with the address and size of the GDT.
3. **Set PE Bit:** Set the Protection Enable (PE) bit in the Control Register 0 (CR0) to switch the CPU to protected mode.
4. **Jump to Protected Mode:** Execute a far jump to clear the prefetch queue and officially enter protected mode.

Global Descriptor Table (GDT)

- The GDT is a table that holds segment descriptors, which define the properties of memory segments.

- Each segment descriptor contains:

~ A 32-bit base address, specifying the starting address of the segment.

~ A 20-bit segment limit, specifying the size of the segment.

~An 8-bit access byte, defining the segment type and access rights.

~ 4 flag bits for additional segment properties.

- Descriptors can define code segments, data segments, or system segments.

Local Descriptor Table (LDT)

- The LDT provides segmentation for individual tasks or programs.
- It has a similar structure to the GDT, containing segment descriptors.
- The ‘LLDT’ instruction is used to load the LDT register with the address and size of the LDT.

Accessing Data Segments

In the Intel 80386 microprocessor, accessing data segments involves several steps: -**Segment Registers:** The processor uses segment registers (DS, ES, FS, GS, SS) to point to the segment descriptors in the GDT or LDT.

-Segment Descriptors: Each segment register contains a selector that indexes into the GDT or LDT to find the segment descriptor. This descriptor provides the base address, limit, and access rights of the segment.

-Logical Address: To access data, a logical address is used, which consists of a segment selector (from a segment register) and an offset within the segment.

-Address Translation: The logical address is translated into a physical address using the base address from the segment descriptor and the offset.

-Access Control: The processor checks access permissions defined in the segment descriptor to ensure the operation is allowed. Thus, the segment register points to the segment descriptor, which provides the necessary information to translate the logical address into a physical address and enforce protection.

Handling Exceptions

- In protected mode, exceptions may occur due to illegal memory access or invalid instructions.
- Common exceptions include General Protection Fault (GPF), Page Fault, and Invalid Opcode.
- Handlers for these exceptions must be set up in the Interrupt Descriptor Table (IDT).
- The IDT contains pointers to interrupt and exception handlers, similar to how the GDT contains segment descriptors.