Chapter Objectives

1. Describe the function and purpose of each program-visible register in the 8086–80486 and Pentium–Pentium 4 microprocessors.
2. Detail the flag register and the purpose of each flag bit.
3. Describe how memory is accessed using real mode memory-addressing techniques.
4. Describe how memory is accessed using protected mode memory-addressing techniques.
5. Describe the program-invisible registers found within the 80286 through Pentium 4 microprocessors.
6. Detail the operation of the memory-paging mechanism.

Outline

2-1 Internal Microprocessor Architecture
2-2 Real Mode Memory Addressing
2-3 Introduction to Protected Mode Memory Addressing
2-4 Memory Paging
IA-32 Basic Environment

- IA-32
  - 32-bit internal for 80386 through Pentium 4
  - compatible with 8086 through 80286
  - FPU: floating point support after 80486
  - MMX: multimedia support after Pentium II
  - XMM: SIMD support after Pentium III
- Three basic types
  - General-purpose
  - special-purpose
  - segment

Programming Model

- 16 program visible registers
  - 7 multipurpose registers
    - EAX, EBX, ECX, EDX,
    - EBP, EDI, ESI
  - 3 special-purpose registers
    - EIP
    - ESP
    - EFLAGS
  - 6 segment registers
    - CS, DS, ES, SS, FS, GS
- program invisible registers
  - control and operate the protected memory system

Three Memory Management Models

- 8086/8088: real mode only
- real memory: first 1M

6 Segment Registers

hold 16-bit segment selections to three types of storage:
- code: CS
- data:
  - DS
  - ES (Extra)
  - FS, GS (available in 80386 or above)
- stack: SS
Special Use of 8 General-purpose Registers

- EAX: Accumulator for operands and results data
- EBX: Pointer to data in the DS segment
- ECX: Counter for string and loop operations
- EDX: I/O pointer
- ESI: Pointer to data in the segment pointed to by the DS register; source pointer for string operations
- EDI: Pointer to data (or destination) in the segment pointed to by the ES register; destination pointer for string operations
- ESP: Stack pointer (in the SS segment)
- EBP: Pointer to data on the stack (in the SS segment)

EFLAG Register

- Three types
  - status flag
  - control flag
  - system flag

EFLAG Register (cont.)

**Status Flags**
- **CF (bit 0)** Carry flag. Set if an arithmetic operation generates a carry or a borrow out of the most-significant bit of the result; cleared otherwise. This flag indicates an overflow condition for unsigned-integer arithmetic. It is also used in multiple-precision arithmetic.
- **PF (bit 2)** Parity flag. Set if the least-significant byte of the result contains an even number of 1 bits; cleared otherwise.
- **AF (bit 4)** Adjust flag. Set if an arithmetic operation generates a carry or a borrow out of bit 3 of the result; cleared otherwise. This flag is used in binary-coded decimal (BCD) arithmetic.
- **ZF (bit 6)** Zero flag. Set if the result is zero; cleared otherwise.
- **SF (bit 7)** Sign flag. Set equal to the most-significant bit of the result, which is the sign bit of a signed integer. (0 indicates a positive value and 1 indicates a negative value.)
- **OF (bit 11)** Overflow flag. Set if the integer result is too large a positive number or too small a negative number (excluding the sign-bit) to fit in the destination operand; cleared otherwise. This flag indicates an overflow condition for signed-integer (two’s complement) arithmetic.

**Control Flag**
- **DF (bit 10)** Direction flag. Controls string instructions (MOV, CMPS, SCAS, LODS, and STOS). Setting the DF flag causes the string instructions to auto-decrement. Clearing the DF flag causes the string instructions to auto-increment. The STD and CLD instructions set and clear the DF flag, respectively.

**System Flags**
- **IF (bit 9)** Interrupt enable flag. Controls the response of the processor to maskable interrupts requests. Set to respond to maskable interrupts; cleared to inhibit maskable interrupts.
- **TF (bit 8)** Trap flag. Set to enable single-step mode for debugging; clear to disable single-step mode.
- **IOPL (bits 12 and 13)** I/O privilege level field. Indicates the I/O privilege level of the currently running program or task. The current privilege level (CPL) of the currently running program or task must be less than or equal to the I/O privilege level to access the I/O address space. This field can only be modified by the POPF and IRET instructions when operating at a CPL of 0.
- **NT (bit 14)** Nested task flag. Controls the chaining of interrupted and called tasks. Set when the current task is linked to the previously executed task; cleared when the current task is not linked to another task.
- **RF (bit 16)** Resume flag. Controls the processor’s response to debug exceptions.
- **VM (bit 17)** Virtual-8086 mode flag. Set to enable virtual-8086 mode; clear to return to protected mode without virtual-8086 mode semantics.
- **AC (bit 18)** Alignment check flag. Set this flag and the AM bit in the CR0 register to enable alignment checking of memory references; clear the AC flag and/or the AM bit to disable alignment checking.
- **VIF (bit 19)** Virtual interrupt flag. Virtual image of the IF flag. Used in conjunction with the VIP flag. (0 use this flag and the VIP flag, the virtual mode extensions are enabled by setting the VM flag in control register CR4.)
- **VIP (bit 20)** Virtual interrupt pending flag. Set to indicate that an interrupt is pending; clear when no interrupt is pending. (Software sets and clears this flag; the processor only reads it.) Used in conjunction with the VIF flag.
- **ID (bit 21)** Identification flag. The ability of a program to set or clear this flag indicates support for the CPUID instruction.
2-2 Real Mode Memory Addressing

- Real mode operation
  - address only the first 1MB (2^20), even in Pentium 4
  - called real memory or conventional memory
  - compatibility
- Segments and Offsets
  - segment address: 16-bit selector defining the beginning address of any 64KB (2^16) memory segment
  - 16-byte boundary: paragraph
  - offset address: selecting any location within the 64KB (2^16) memory segment

<table>
<thead>
<tr>
<th>Segment Register</th>
<th>Starting Address</th>
<th>Ending Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000H</td>
<td>20000H</td>
<td>2FFFFFH</td>
</tr>
<tr>
<td>2001H</td>
<td>20010H</td>
<td>3000Fh</td>
</tr>
<tr>
<td>2100H</td>
<td>21000H</td>
<td>30FFH</td>
</tr>
<tr>
<td>AB800H</td>
<td>AB800H</td>
<td>BAFFFH</td>
</tr>
<tr>
<td>1234H</td>
<td>12340H</td>
<td>2233Fh</td>
</tr>
</tbody>
</table>

Segment and Offset Addressing

- Linear address of physical memory = segment address<<4 + offset address
- Real memory: 1MB
  - 00000H – FFFFFH
- High memory: (64K – 16) bytes
  - A20 address pin is enabled
  - Segment address = FFFFH
    0FFFF0H – 10FFEFH

Default Segment and Offset Registers

<table>
<thead>
<tr>
<th>Segment</th>
<th>Offset</th>
<th>Special Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS</td>
<td>IP</td>
<td>Instruction address</td>
</tr>
<tr>
<td>SS</td>
<td>SP or BP</td>
<td>Stack address</td>
</tr>
<tr>
<td>DS</td>
<td>BX, DI, SI, an 8-bit number, or a 16-bit number</td>
<td>Data address</td>
</tr>
<tr>
<td>ES</td>
<td>DI for string instructions</td>
<td>String destination address</td>
</tr>
</tbody>
</table>

TABLE 2-2 8086–80486 and Pentium–Pentium 4 default 16-bit segment and offset address combinations.

<table>
<thead>
<tr>
<th>Segment</th>
<th>Offset</th>
<th>Special Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS</td>
<td>EIP</td>
<td>Instruction address</td>
</tr>
<tr>
<td>SS</td>
<td>ESP and EBP</td>
<td>Stack address</td>
</tr>
<tr>
<td>DS</td>
<td>EAX, EBX, ECX, EDX, ESI, EDI, an 8-bit number, or a 32-bit number</td>
<td>Data address</td>
</tr>
<tr>
<td>ES</td>
<td>EDI for string instructions</td>
<td>String destination address</td>
</tr>
<tr>
<td>FS</td>
<td>No default</td>
<td>General address</td>
</tr>
<tr>
<td>GS</td>
<td>No default</td>
<td>General address</td>
</tr>
</tbody>
</table>

TABLE 2-3 80386 through the Pentium 4 default 32-bit segment and offset address combinations.

FIGURE 2–4 A memory system showing placement of four memory segments

- A memory segment can touch or even overlap
- A memory segment is like a window that can be moved over any area of memory to access data or code
- A program can have more than 4 or 6 segments, but can only access 4 or 6 segments at a time
17

- The program loader of DOS places the program in the TPA at the first available area of memory above the drivers and other TPA programs—indicated by a free-pointer—overlapped because the amount of data does not require 64KB of memory.

FIGURE 2–5 An application program containing a code, data, and stack segment loaded into a DOS system

2-3 Introduction to Protected Mode Memory Addressing

- Protected mode memory addressing
  - access above 1MB of memory (80286 and above)
  - offset address: still used to access information located within the memory segment
    - 80286: 16-bit
    - 80386 and above: 32-bit \(2^{32} = 4\)G
  - segment register: contains a selector that selects a descriptor from a descriptor table
    - descriptor: describing the memory segment’s location, length, and access rights
  - protected mode instructions are identical to real mode instructions; programs written to function in the real mode will function without change in the protected mode.

Segment and Offset Addressing Scheme

- complicated
- programs able to be relocated in the memory system
  - A relocatable program can be placed into any area of memory and executed without change
  - not all machines contain the same memory areas
    - segment and offset addressing vs. virtual memory addressing
- dynamic relocatable: the memory segment can be moved to any place in the memory system
  - changing only the contents of the segment registers
  - without changing any of the offset addresses

Selectors and Descriptors

- Two descriptor tables: 8192 descriptors each
  - global descriptor table (GDT)
    - system descriptors contain segment definitions that apply to all programs
    - GDT register (GDTR) contains the linear address of the base of the GDT
  - local descriptor table (LDT)
    - application descriptors are usually unique to an application
    - LDT register (LDTR) contains the linear address of the base of the LDT
- Descriptor
  - describing location, length, and access rights of the segment
  - 8 bytes in length, 8192 * 8 = 64KB per descriptor table
  - descriptor 0 is called null descriptor and may not be used
- Selector: located in the segment register
  - selects one of 8192 descriptors from one of two descriptor tables
  - 8192*2 memory segments described for each applications
**Descriptors Format**

- **base address**: starting location of memory segment
  - 80286: 24-bit (\(2^{24}=16\text{MB}\))
  - 80386 or above: 32-bit (\(2^{32}=4\text{GB}\))
  - begin at any location: no paragraph boundary limitation
- **segment limit**: last offset address in a segment
  - i.e. base=F00000H and limit=FFH: F00000 ~ F000FFH
  - 80286: 16-bit (\(2^{16}=64\text{KB}\))
  - 80386 or above: 20-bit (\(2^{20}=1\text{M}; 1\text{MB or 1M*4KB/page=4GB}\))

**FIGURE 2–6** The descriptor formats for 80286 and 80386 through Pentium 4 microprocessors

**Descriptors of 80386-P4**

- **G** (granularity) bit
  - (0) limit of 00000H to FFFFFH; (1) multiplied by 4K
- **AV** (available) bit: whether the segment is available
- **D** bit: how instructions access register/memory data
  - (0) 16-bit instruction; (1) 32-bit instruction

**EXAMPLE 2–1**

Base = Start = 10000000H
G = 0
End = Base + Limit = 10000000H + 001FFH = 10001FFH

**EXAMPLE 2–2**

Base = Start = 10000000H
G = 1
End = Base + Limit = 10000000H + 001FFFFH = 101FFFFH

**FIGURE 2–7** The access rights byte for the 80286 through Pentium 4 descriptor

**Format of Segment Register**

- **TI** (Table Index) bit: DGT or LDT
- **RPL** (Requested Privilege Levels) bits
  - 00 is the highest and 11 is the lowest
  - access is granted if the RPL matches or is higher in priority than the DPL set by the access rights byte
  - privilege violation is indicated if the privilege level is violated

**FIGURE 2–8** The contents of a segment register during protected mode operation of the 80286 through Pentium 4 microprocessors.
FIGURE 2-9 Using the DS register to select a descriptor from the global descriptor table

<table>
<thead>
<tr>
<th>Base (B11-B24)</th>
<th>Offset</th>
<th>Limit (L15-L16)</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access rights</td>
<td>Base (B23-B16)</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Base (B15-B0)</td>
<td>Limit (L15-L0)</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>0010 0000H</td>
<td>000FFH</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Base** = 0010 0000H
- **Limit** = 000FFH
- **Access rights** (92H=1001 0010)
  - P=1: valid
  - DPL=00: privilege level
  - S=1: code or data segment
  - E=0: data segment
  - ED=0: expand upward (data)
  - W=1: writable
  - A=0: not accessed yet
- **DS=0008H**
- **Selector=1**
- **TI=0: GDT**
- **RPL=00: privilege level**

The DS register accesses locations 100000H-1000FFH as a data segment.

Program-Invisible Registers

- **Segment registers**
  - CS
  - DS
  - ES
  - SS
  - FS
  - GS
- **Descriptor cache**
  - Base address
  - Limit
  - Access
- **Segment registers**
  - CS
  - DS
  - ES
  - SS
  - FS
  - GS
- **Descriptor cache**
  - Base address
  - Limit
  - Access

- **GDTR and IDTR** (interrupt descriptor table register) are initialized before using the protected mode
- **LDTR** is loaded with a selector within GDT
- **TR** (task register) holds a selector, which accesses a descriptor that defines a task
- **task switch in about 17μs**

2-4 Memory Paging

- **Memory paging mechanism**
  - 80386 and above
  - Any physical memory location can be assigned to any linear address
  - 4KB-page boundary (or 4MB-page boundary in Pentium)
- **Advantages**
  - A linear address is invisibly translated into a physical address
  - Allows memory to be placed into areas where no memory exists
    - EMM386.EXE: reassign extended memory, in 4K blocks, to the system memory between video BIOS and the system BIOS ROMS for upper memory blocks

Paging Control Registers (CR0-CR4)

- **CR4**: only for Pentium and above (support 4MB paging)
- **Page directory base address**: locates page directory at any 4KB boundary
  - The page directory contains 1024 directory entries of 4 bytes each
- **PCD (page-level cache disable)** and **PWT (page-level write transparent)**: control µP pins

FIGURE 2-10 The program-invisible register within the 80286–Pentium 4 microprocessors

- **descriptor cache associated with segment registers**
  - Not directly addressed by software
  - Loaded each time the segment register is changed
  - µP need not refer to GDT or LDT for each access

FIGURE 2–11 The control register structure of the microprocessor.

- **CR0**: Pag: 1 if paging is enabled
- **CR1**: Reserved
- **CR2**: Page fault linear address
- **CR3**: Page directory base address
  - PPD (page-level page disable) and PWT (page-level write transparent): control µP pins
- **CR4**: only for Pentium and above (support 4MB paging)
  - Page directory base address: locates page directory at any 4KB boundary
  - The page directory contains 1024 directory entries of 4 bytes each
Address Translation

A linear address is broken into 3 sections to access the page directory entry, page table entry, and page offset address.

Example: EMM386.EXE on DOS

Summary

Chapter 2 The Microprocessor and its Architecture

2-1 Internal Microprocessor Architecture

2-2 Real Mode Memory Addressing

2-3 Introduction to Protected Mode Memory Addressing

2-4 Memory Paging