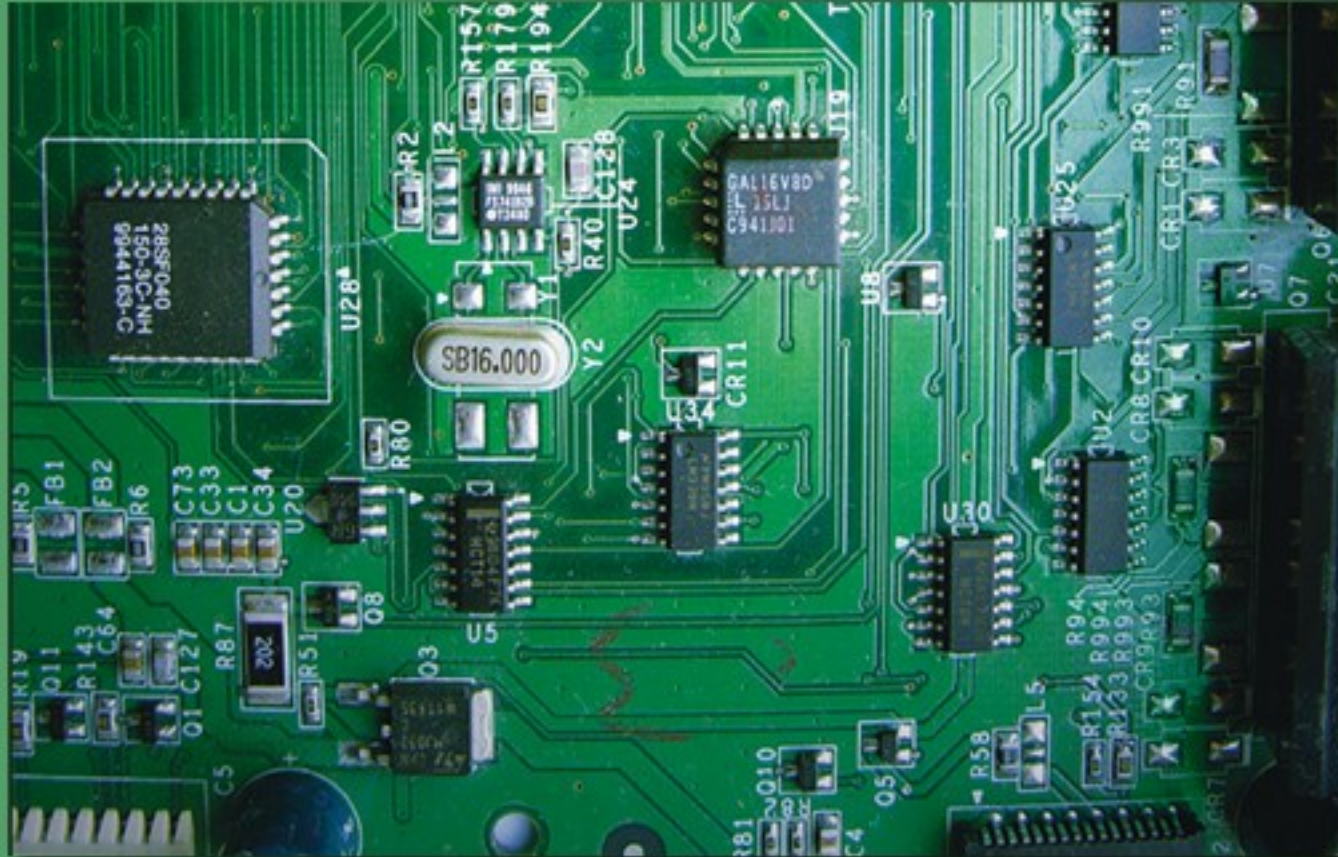


# The Intel Microprocessors

8086/8088, 80186/80188, 80286, 80386, 80486 Pentium, Pentium Pro Processor, Pentium II, Pentium 4, and Core2 with 64-bit Extensions

## Architecture, Programming, and Interfacing



EIGHTH EDITION

Barry B. Brey

PEARSON

## Chapter 2: The Microprocessor and its Architecture

# Introduction

- This chapter presents the microprocessor as a **programmable device** by first looking at its **internal programming model** and then how its memory space is addressed.
- The architecture of Intel microprocessors is presented, as are the ways that the family members address the memory system.
- Addressing modes for this powerful family of microprocessors are described for the **real**, **protected**, and **flat** modes of operation.

# Chapter Objectives

Upon completion of this chapter, you will be able to:

- Describe function and purpose of each **program-visible register** in the 8086-Core2 microprocessors, including 64-bit extensions.
- Detail the **flag register** and the purpose of each flag bit.
- Describe how memory is accessed using real mode **memory-addressing** techniques.

# Chapter Objectives

(*cont.*)

Upon completion of this chapter, you will be able to:

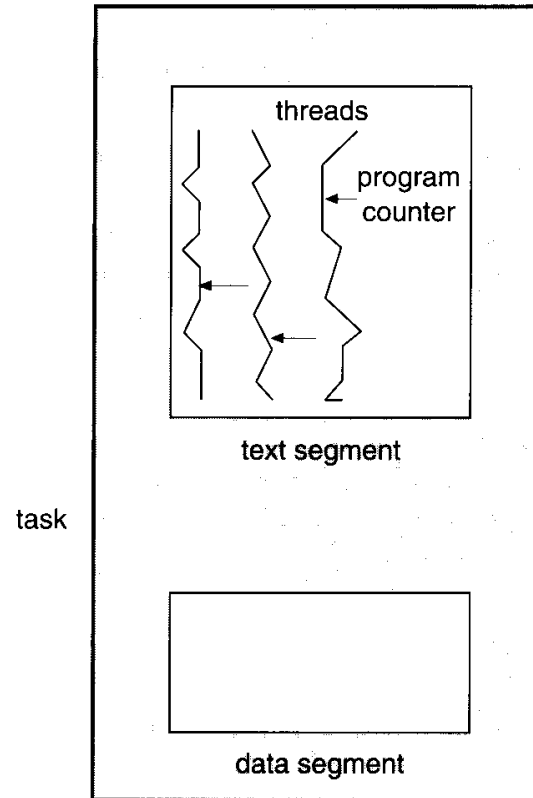
- Describe how memory is accessed using **protected mode** memory-addressing techniques.
- Describe how memory is accessed using the 64-bit **flat memory model**.
- Describe **program-invisible registers** found in the 80286 through Core2 microprocessors.
- Detail the operation of the **memory-paging** mechanism.

# 2-1 INTERNAL MICROPROCESSOR ARCHITECTURE

- Before a program is written or instruction investigated, internal configuration of the microprocessor must be known.
- In a multiple core microprocessor each core contains the same programming model.
- Each core runs a separate **task** or **thread** simultaneously.

A thread consists of a program counter, a register set, and a stack space.

A task shares with peer threads its code section, data section, and operating system resources



**Figure 4.8** Multiple threads within a task.

# The Programming Model

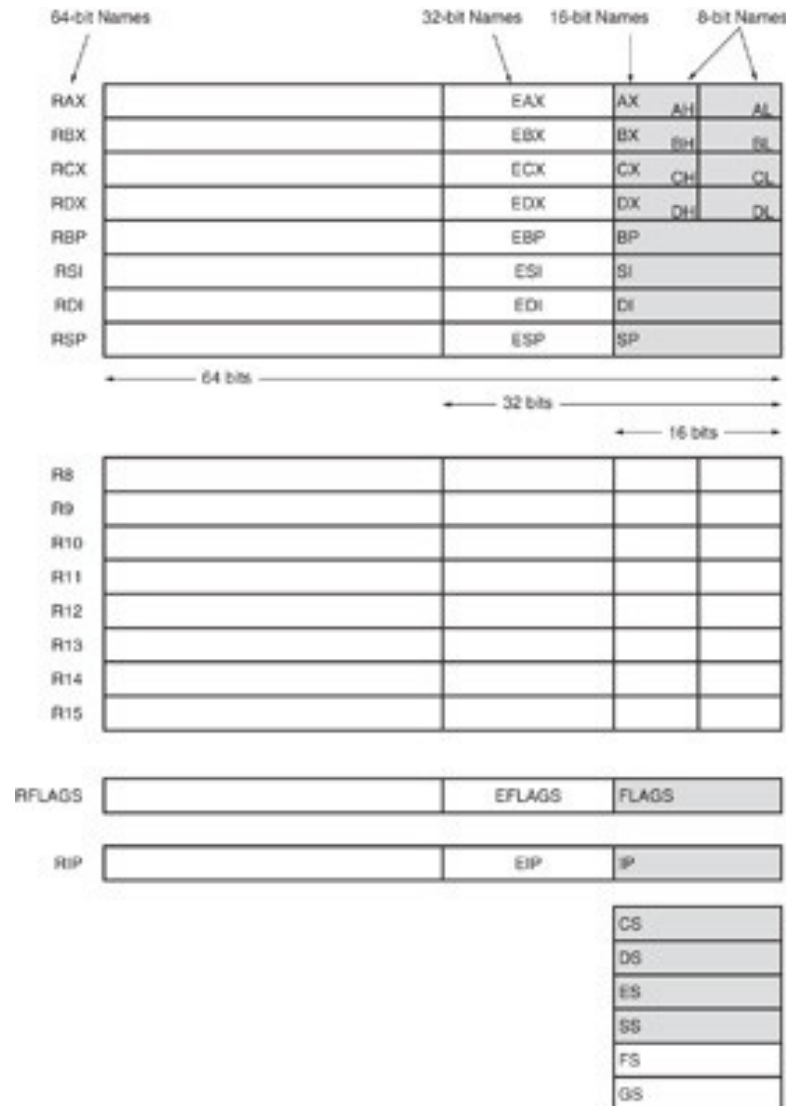
- 8086 through Core2 considered **program visible**.
  - registers are used during programming and are specified by the instructions
- Other registers considered to be **program invisible**.
  - not addressable directly during applications programming



- 80286 and above contain program-invisible registers to control and operate protected memory.
  - and other features of the microprocessor
- 80386 through Core2 microprocessors contain **full** 32-bit internal architectures.
- 8086 through the 80286 are fully upward-compatible to the 80386 through Core2.
- Figure 2–1 illustrates the programming model 8086 through Core2 microprocessor.
  - including the 64-bit extensions



**Figure 2-1** The programming model of the 8086 through the Core2 microprocessor including the **64-bit extensions**.



# Multipurpose Registers

- **RAX** - a **64-bit** register (RAX), a **32-bit** register (**accumulator**) (EAX), a **16-bit** register (AX), or as either of two **8-bit** registers (AH and AL).
- The accumulator is used for instructions such as multiplication, division, and some of the adjustment instructions.
- Intel plans to expand the **address bus** to **52 bits** to address 4P ( $2^{52} \sim 10^{15}$  =peta) bytes of memory.

# Address Space (Main Memory: RAM)

- Address bus:16 bit → Address Space:64 KBytes
- Address bus:20 bit → Address Space:1 MBytes
- Address bus:32 bit → Address Space:4 GBytes
- Address bus:34 bit → Address Space:16GBytes
- Address bus:36 bit → Address Space:64GBytes
- Address bus:38 bit → Address Space:256GBytes
- Address bus:52 bit → Address Space: $10^{15}$  Bytes

- **RBX**, addressable as RBX, EBX, BX, BH, BL.
  - BX register (**base index**) sometimes holds offset address of a location in the memory system in all versions of the microprocessor
- **RCX**, as RCX, ECX, CX, CH, or CL.
  - a (**count**) general-purpose register that also holds the count for various instructions
- **RDX**, as RDX, EDX, DX, DH, or DL.
  - a (**data**) general-purpose register
  - holds a part of the result from a multiplication or part of dividend before a division

- **RBP**, as RBP, EBP, or BP.
  - points to a memory (**base pointer**) location for memory data transfers
- **RDI** addressable as RDI, EDI, or DI.
  - often addresses (**destination index**) string destination data for the string instructions
- **RSI** used as RSI, ESI, or SI.
  - the (**source index**) register addresses source string data for the string instructions
  - like RDI, RSI also functions as a general-purpose register

- **R8 - R15** found in the Pentium 4 and Core2 if 64-bit extensions are enabled.
  - data are addressed as 64-, 32-, 16-, or 8-bit sizes and are of general purpose
- Most applications will not use these registers until 64-bit processors are common.
  - the 8-bit portion is the rightmost 8-bit only
  - bits 8 to 15 are not directly addressable as a byte

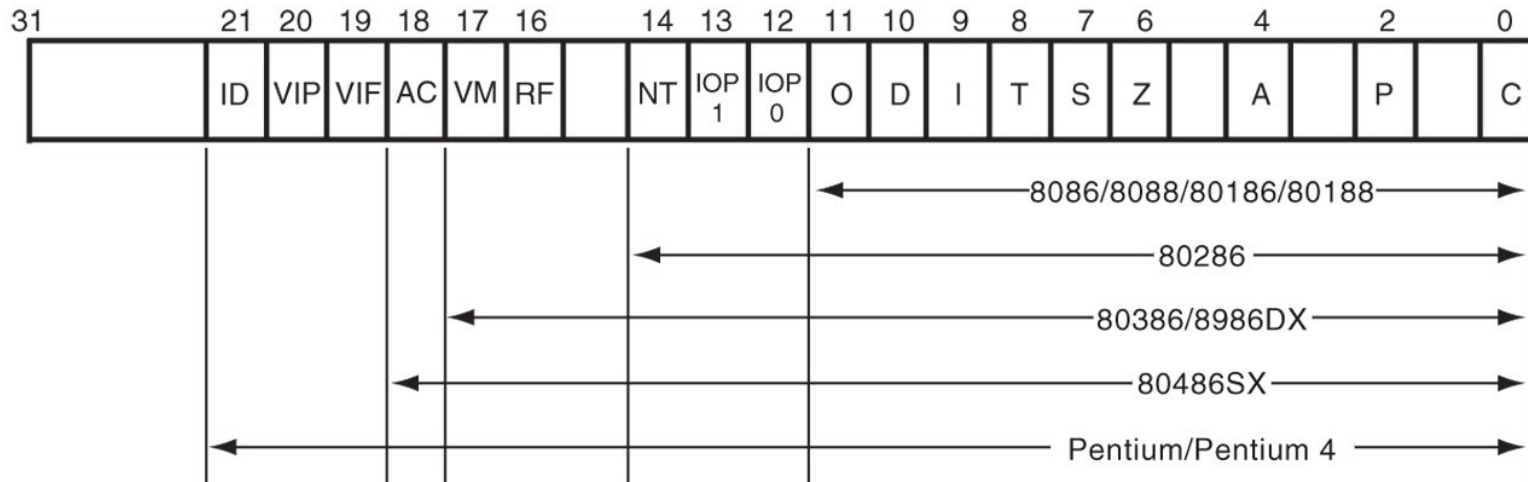
# ***Special-Purpose Registers***

- Include **RIP**, **RSP**, and **RFLAGS**
  - segment registers include CS, DS, ES, SS, FS, and GS
- **RIP** addresses the next instruction in a section of memory.
  - defined as (**i**nstruction **p**ointer) a code segment
- **RSP** addresses an area of memory called the stack.
  - the (**s**tack **p**ointer) stores data through this pointer



- **RFLAGS** indicate the **condition** of the microprocessor and **control** its operation.
- Figure 2–2 shows the flag registers of all versions of the microprocessor.
- Flags are **upward-compatible** from the 8086/8088 through Core2 .
- The rightmost five and the overflow flag are changed by most arithmetic and logic operations.
  - although data transfers do not affect them

**Figure 2-2** The EFLAGS and FLAG register counts for the entire 8086 and Pentium microprocessor family.



- Flags never change for any data transfer or program control operation.
- Some of the flags are also used to control features found in the microprocessor.

- Flag bits, with a brief description of function.
- **C (carry)** holds the carry after addition or borrow after subtraction.
  - also indicates error conditions
- **P (parity)** is the count of ones in a number expressed as even or odd. Logic 0 for odd parity; logic 1 for even parity.
  - if a number contains three binary one bits, it has odd parity
  - if a number contains no one bits, it has even parity

# List of Each Flag bit, with a brief description of function.

- **C (carry)** holds the carry after **addition** or borrow after **subtraction**.
  - also indicates error conditions
- **P (parity)** is the count of ones in a number expressed as even or odd. Logic 0 for odd parity; logic **1 for even parity**.
  - if a number contains three binary one bits, it has odd parity; If a number contains no one bits, it has even parity

- **A (auxiliary carry)** holds the carry (half-carry) after addition or the borrow after subtraction between bit **positions 3 and 4** of the result.
- **Z (zero)** shows that the **result** of an arithmetic or logic operation is zero.
- **S (sign)** flag holds the arithmetic sign of the **result** after an arithmetic or logic instruction executes.
- **T (trap)** The trap flag enables trapping through an on-chip debugging feature.

- **I (interrupt)** controls operation of the INTR (interrupt request) input pin.
- **D (direction)** selects **increment** or **decrement** mode for the DI and/or SI registers.
- **O (overflow)** occurs when signed numbers are added or subtracted.
  - an overflow indicates the result has exceeded the **capacity** of the machine

- **IOPL** used in protected mode operation to select the **p**rivilege **l**evel for I/O devices.
- **NT (n**ested **t**ask) flag indicates the current task is nested within another task in protected mode operation.
- **RF (r**esume) used with debugging to control resumption of execution after the next instruction.
- **VM (v**irtual **m**ode) flag bit selects virtual mode operation in a protected mode system.



- **AC, (alignment check)** flag bit activates if a word or doubleword is addressed on a non-word or non-doubleword boundary.
- **VIF** is a copy of the **interrupt flag** bit available to the Pentium 4—(**virtual interrupt**)
- **VIP (virtual)** provides information about a virtual mode interrupt for (**interrupt pending**) Pentium.
  - used in multitasking environments to provide virtual interrupt flags

- **ID (identification)** flag indicates that the Pentium microprocessors support the **CPUID** instruction.
  - CPUID instruction provides the system with information about the Pentium microprocessor

# ***Segment Registers***

- **Generate memory addresses** when combined with other registers in the microprocessor.
- Four or six **segment registers** in various versions of the microprocessor.
- A segment register functions differently in real mode than in protected mode.
- Following is a list of each segment register, along with its function in the system.

- **CS (code)** segment holds code (programs and procedures) used by the microprocessor.
- **DS (data)** contains most data used by a program.
  - Data are accessed by an offset address or contents of other registers that hold the offset address
- **ES (extra)** an additional data segment used by some instructions to hold destination data.

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

- **FS** and **GS** segments are **supplemental segment registers** available in 80386–Core2 microprocessors.
  - allow **two additional memory segments** for access by programs
- Windows uses these segments for **internal operations**, but no definition of their usage is available.

## 2-2 REAL MODE MEMORY ADDRESSING

- 80286 and above operate in either the real or protected mode.
  - *only 8086 and 8088 operate exclusively in the real mode*
  - *64-bit processors operates on long mode.*
- **Real mode operation** allows the microprocessor to address only the first 1M byte of memory space—even in Pentium 4 or Core2 microprocessor.
  - *the first 1M byte of memory is called the **real memory, conventional memory, or DOS memory** system*

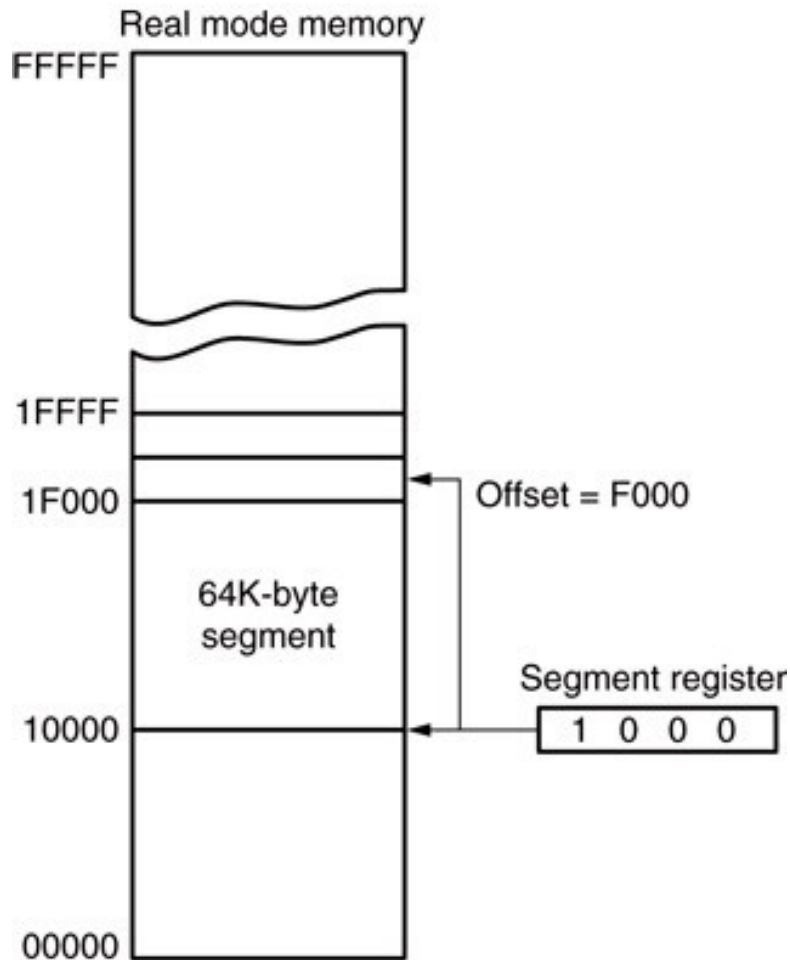


- The DOS requires microprocessor operates in real mode.
- Real mode operation allows application software written for 8086/8088 (which contains only 1M byte of memory) to function in 80286 and above.[upward compatibility]
- Each microprocessor begins operation in real mode by default whenever power is applied or the microprocessor is reset.
  - *Processors operating on 64-bit mode can not execute real mode application. **How DOS applications runs in such systems?***
    - *Using DOS emulators written for 64-bit mode.*

# Segments and Offsets

- A combination of a segment address and an offset address accesses a memory location in the real mode.
- All real mode memory addresses must consist of a segment address **plus** an offset address.
  - **segment address** *defines the beginning address of any 64K-byte memory segment.*
    - Segments in real mode always have a length of 64K bytes.
  - **offset address** *selects any location within the 64K byte memory segment*
- Figure 2–3 shows how the **segment plus offset** addressing scheme selects a memory location.

**Figure 2-3** The real mode memory-addressing scheme, using a segment address plus an offset.



- this shows a memory segment beginning at 10000H, ending at location 1FFFFH
  - 64K bytes in length
- also shows how an offset address, called a **displacement**, of F000H selects location 1F000H in the memory

- Once the beginning address is known, the **ending address** is found by adding FFFFH.
  - *because a **real mode** segment of memory is 64K in length*
- The offset address is always added to the segment starting address to locate the data.
- Segment and offset address is sometimes written as 1000:2000.
  - a segment address of 1000H; an offset of 2000H

# Example of real mode segment addresses

Segment Register	Starting Address	Ending Address
2000H		
2001H		
2100H		
AB00H		
1234H		

# Example of real mode segment addresses

Segment Register	Starting Address	Ending Address
2000H	20000H	
2001H	20010H	
2100H	21000H	
AB00H	AB000H	
1234H	12340H	

# Example of real mode segment addresses

Segment Register	Starting Address	Ending Address
2000H	20000H	2FFFFH
2001H	20010H	3000FH
2100H	21000H	30FFFH
AB00H	AB000H	BAFFFH
1234H	12340H	2233FH



# Default Segment and Offset Registers

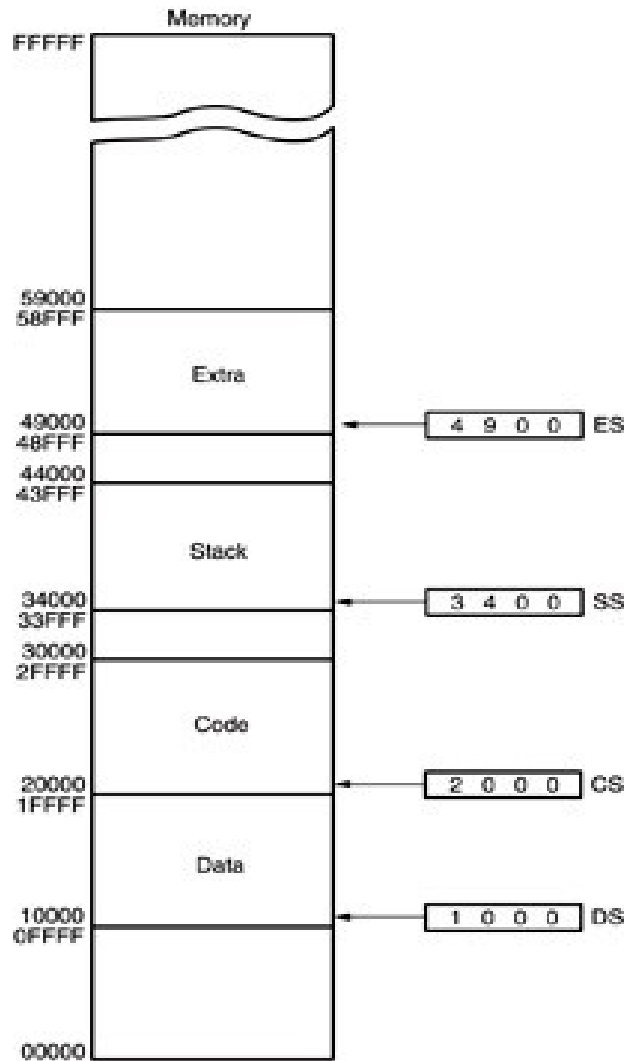
- The microprocessor has rules that apply to segments whenever memory is addressed.
  - these define the segment and offset register combination
- The **code segment** register defines the **start** of the code segment.
- The **instruction pointer** locates the next instruction within the code segment.
- The combination (CS:IP or CS:EIP) locates the next instruction executed by the microprocessor.
  - For example: if CS = 1400H and IP/EIP = 1200H, then microprocessor fetches the the next instruction from memory location 1400**0**H + 1200H or 15200H.

# Default 16-bit segment and offset combinations

Segment	Offset	Special Purpose
CS	IP	Instruction addresses
SS	SP or BP	Stack addresses
DS	BX, DI, SI, and 8-bit or 16-bit number	Data addresses
ES	DI for string instructions	String destination address

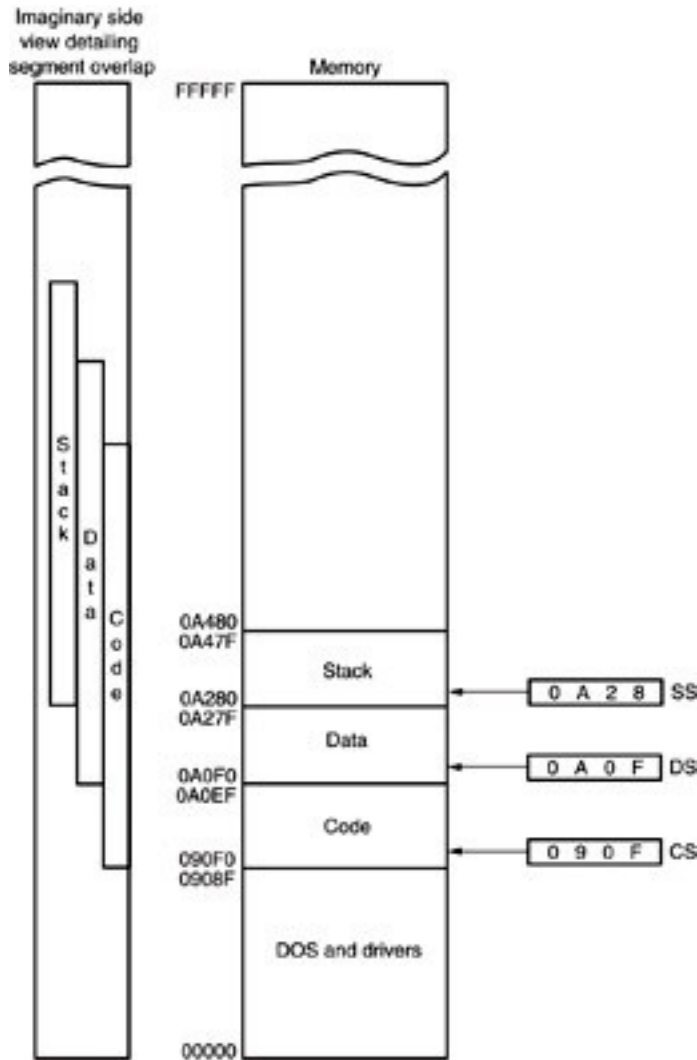
- Another of the default combinations is the **stack**.
  - stack data are referenced through the stack segment at the memory location addressed by either the stack pointer (SP/ESP) or the pointer (BP/EBP)
- Figure 2–4 shows a system that contains four memory segments.
  - *8086-80286 allow four memory segments, 80386-core2 allow six segments.*
  - *a memory segment can touch or overlap if 64K bytes of memory are not required for a segment*

**Figure 2-4** A memory system showing the placement of four memory segments.



- think of segments as Windows that can be moved over any area of memory to access data or code
- a program can have more than **four** or **six segments**,
  - but only access four or six segments at a time

**Figure 2-5** An application program containing a code, data, and stack segment loaded into a DOS system memory.



- a program placed in memory by DOS is loaded in the TPA at the first available area of memory above drivers and other TPA programs
- area is indicated by a **free-pointer** maintained by DOS
- program loading is handled automatically by the **program loader** within DOS

# TPA

- The transient program area (TPA) holds the DOS (disk operating system) operating system; other programs that control the computer system.

# Segment and Offset Addressing Scheme Allows **Relocation**

- Segment plus offset addressing allows DOS programs to be relocated in memory.
- A **relocatable program** is one that can be placed into any area of memory and executed without change.
- **Relocatable data** are data that can be placed in any area of memory and used without any change to the program.

- Because memory is addressed within a segment by an offset address, the **memory segment** can be **moved to any place** in the memory system without changing any of the offset addresses.
- Only the contents of the segment register must be changed to address the program in the new area of memory.
- Windows programs are written assuming that the first **2G** of memory are available for code and data.



# 2–3 INTRODUCTION TO PROTECTED MODE MEMORY ADDRESSING

- Allows access to data and programs located within & **above** the first 1M byte of memory.
- **Protected mode** is where Windows operates.
  - Protected mode addressing(80286 and above)
- The segment register contains a **selector** that selects a **descriptor** from a descriptor table.
  - *In real mode **SR** contains the segment address, which is used by microprocessor to access the memory.*
- The **descriptor** describes the memory **segment's location, length, and access rights.**

- There is **no segment address** in protected mode.
- **Instructions are identical** to real mode instructions, because the segment register and offset address still access memory.
- Most programs written to function in the real mode will **function without change** in the protected mode.
- In 80386 and above, the **offset address can be a 32-bit** number instead of 16-bit number in protected mode.
  - *A 32-bit offset address allows the microprocessor to access data within a segment that can be up to **4G bytes** in length.*
  - *Program that are written for the 32-bit protected mode executes in the 64-bit mode of the Pentium 4.*

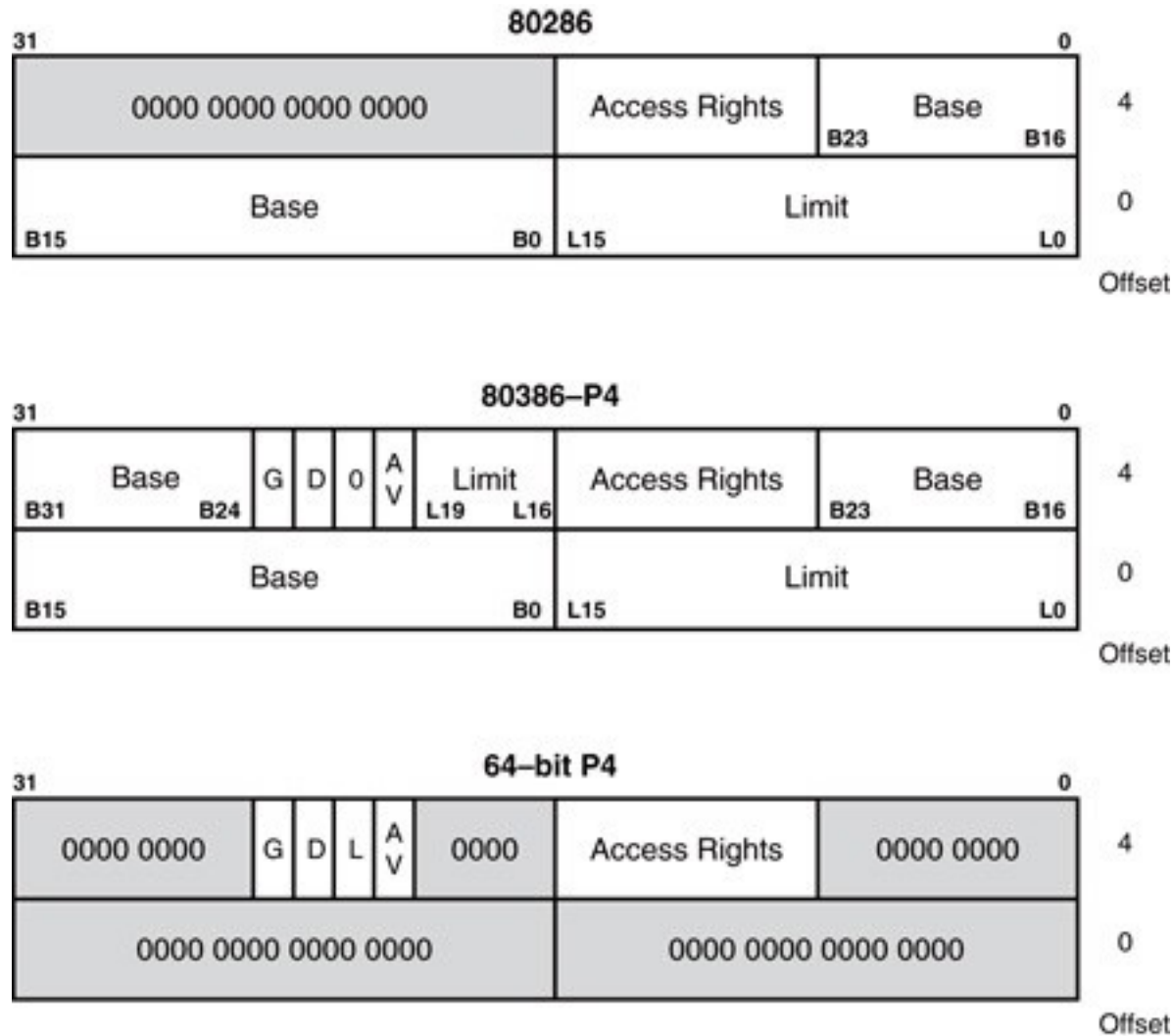
# Selectors and Descriptors

- The **selector** located in the segment register, selects one of **8192** descriptors from one of two tables of descriptor.
- The **descriptor** describes the **location**, **length**, and **access rights** of the segment of memory.
- Indirectly, the segment register still selects a memory segment, but not directly as in real mode.
- For example, in the the real mode if CS=0008H, code segment begins at location 00080H.
  - *In protected mode, this segment number can address any memory location in the system for the code segment.*

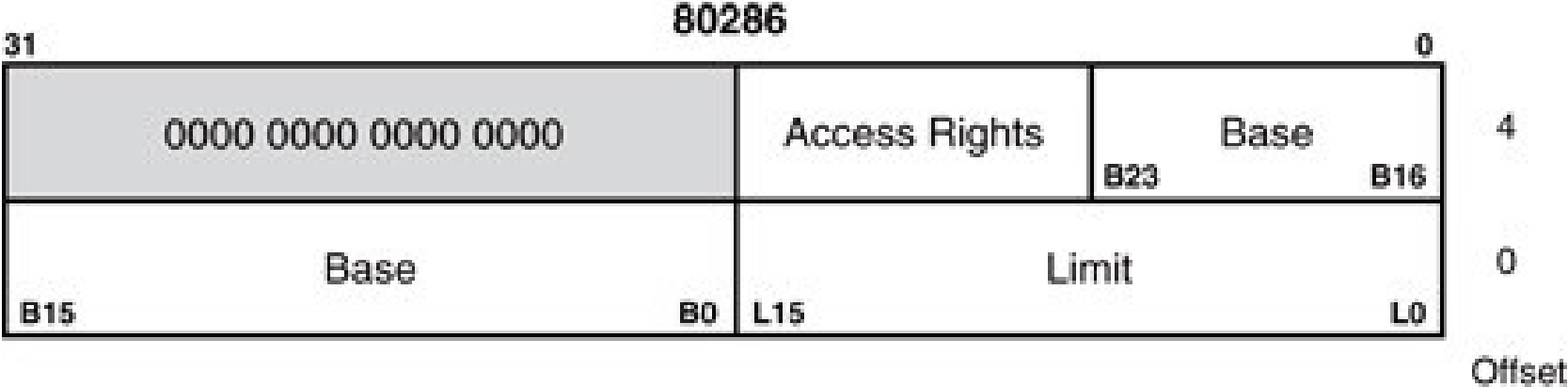
- There are **two** descriptor tables used with the segment registers
  - **Global descriptors** contain segment definitions that apply to all programs.
  - **Local descriptors** are usually unique to an application.
  - a global descriptor might be called a **system descriptor**, and local descriptor an **application descriptor**
- Each descriptor table contains 8192 descriptors
  - A total of 16,384 descriptors are available to an application at any time.

- Figure 2–6 shows the format of a descriptor for the 80286 through the Core2.
  - each descriptor is 8 bytes in length
  - global and local descriptor tables are a maximum of 64K bytes in length

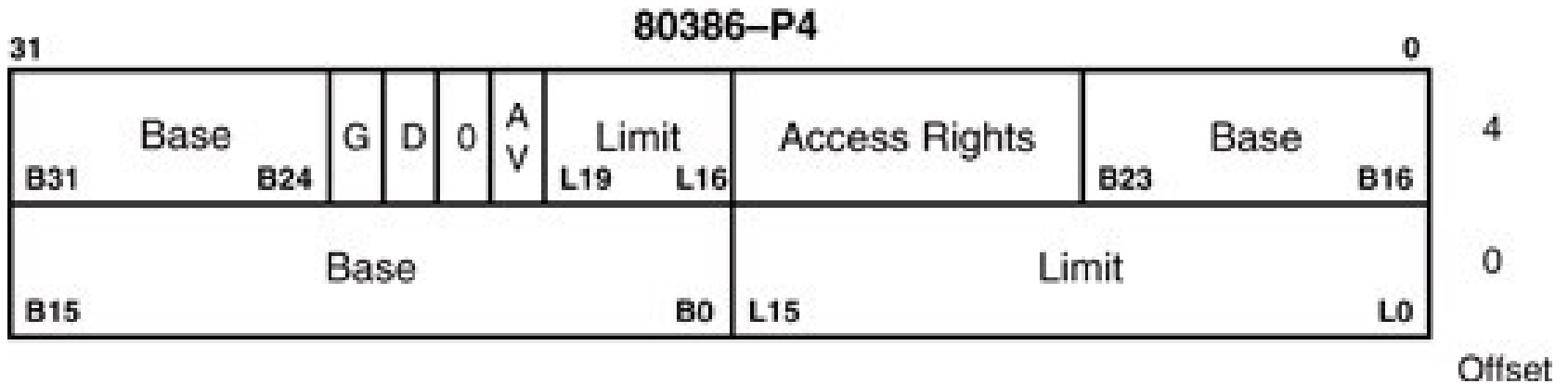
**Figure 2-6** The 80286 through Core2 **64-bit descriptors.**



**Figure 2-6** The 80286 through Core2 64-bit descriptors.

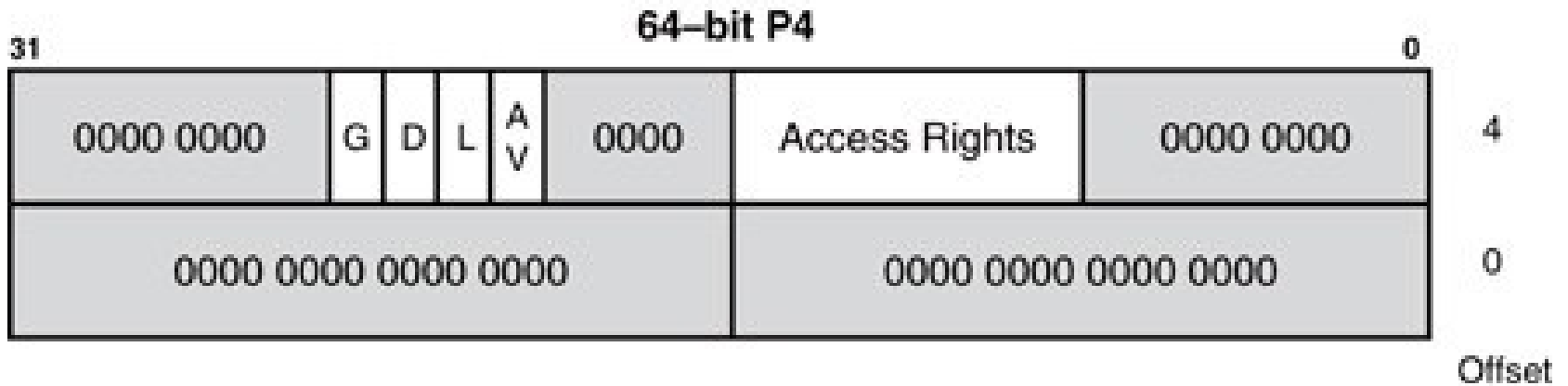


**Figure 2-6** The 80286 through Core2 64-bit descriptors.





**Figure 2-6** The 80286 through Core2 64-bit descriptors.



- The **base address** of the descriptor indicates the starting location of the memory segment.
  - For **80286**, the base address is a 24-bit address
    - segments may begin at **any address** in its 16M bytes of memory.
  - For **80386 and above**, the base address is a 32-bit address
    - Allows segments to begin at **any address** in its 4G bytes of memory.

- The **segment limit** contains the last offset address found in a segment.
  - *For example: if segment begins at memory location F00000H and ends at location F000FFH.*
  - For the **80286 microprocessor**
    - *Base address is: F00000H.*
    - *Limit is : 00FFH*
  - For **80386 and above**,
    - *Base address is: 00F00000H.*
    - *Limit is: 000FFH*
- Notice that the 80286 has a 16-bit limit and 80386 and through the Pentium 4 have a 20-bit limit.

- The G bit, or **granularity bit** (80386-Pentium 4)
- If G=0, the limit specifies a segment limit of 00000H to FFFFFH.
- If G=1, the value of limit is multiplied by 4K bytes(appendd with FFFH). The limit is then 00000FFFH to FFFFFFFFH.
  - allows a segment length of 4K to 4G bytes in steps of 4K bytes.
  - **32-bit offset address** allows segment lengths of 4G bytes
  - **16-bit offset address** allows segment lengths of 64K bytes.

- Calculate the segment start and end if base address is 10000000H, the limit is 001FFH, for both cases of G?

G	Start	End
0		
1		

- Calculate the segment start and end if base address is 10000000H, the limit is 001FFH, for both cases of G?

G	Base=Start	End=Base+Limit
0	10000000H	100001FFH
1	10000000H	101FFFFFFH

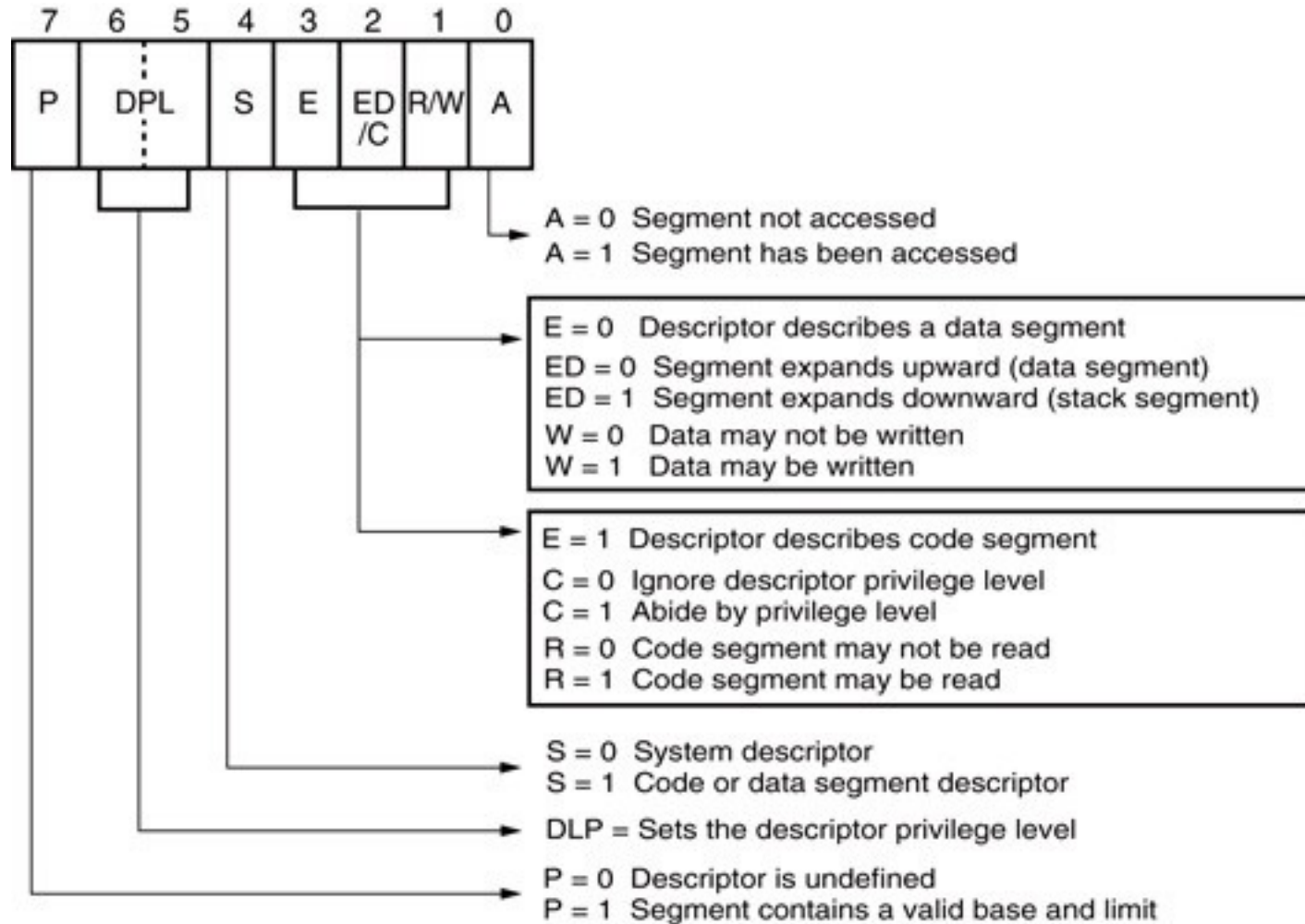
- Operating systems operate in a 16- or 32-bit environment.
- DOS uses a **16-bit** environment.
- Most Windows applications use a **32-bit** environment called **WIN32**.
- MSDOS/PCDOS & Windows 3.1 operating systems require 16-bit instruction mode.
- Instruction mode is accessible only in a protected mode system such as Windows XP → **Windows Vista → Windows 7 → Windows 8 → ???**.

- The L, AV, and D bits
- L bit is used in 64-bit descriptor
  - Selects 64-bit addresses in a P4 to core2 with 64-bit extension when  $L=1$  and 32-bit compatibility mode when  $L = 0$
- D bit indicates how the 80386-core2 instructions access register and memory data in the protected or real mode.
  - If  $D=0$  instructions are 16-bit instructions
  - If  $D=1$  instructions are 32-bit instructions
- AV bit in 80386 and above descriptor use by some Operating systems to indicate whether segment is available( $AV=1$ ) or not available( $AV=0$ )



- The **access rights byte** controls access to the protected mode segment.
  - describes segment function in the system and allows complete control over the segment
  - if the segment is a data segment, the direction of growth is specified
- If the segment grows **beyond** its **limit**, the operating system is interrupted, indicating a general **protection fault**.
- You can specify whether a data segment can be **written** or is **write-protected**.

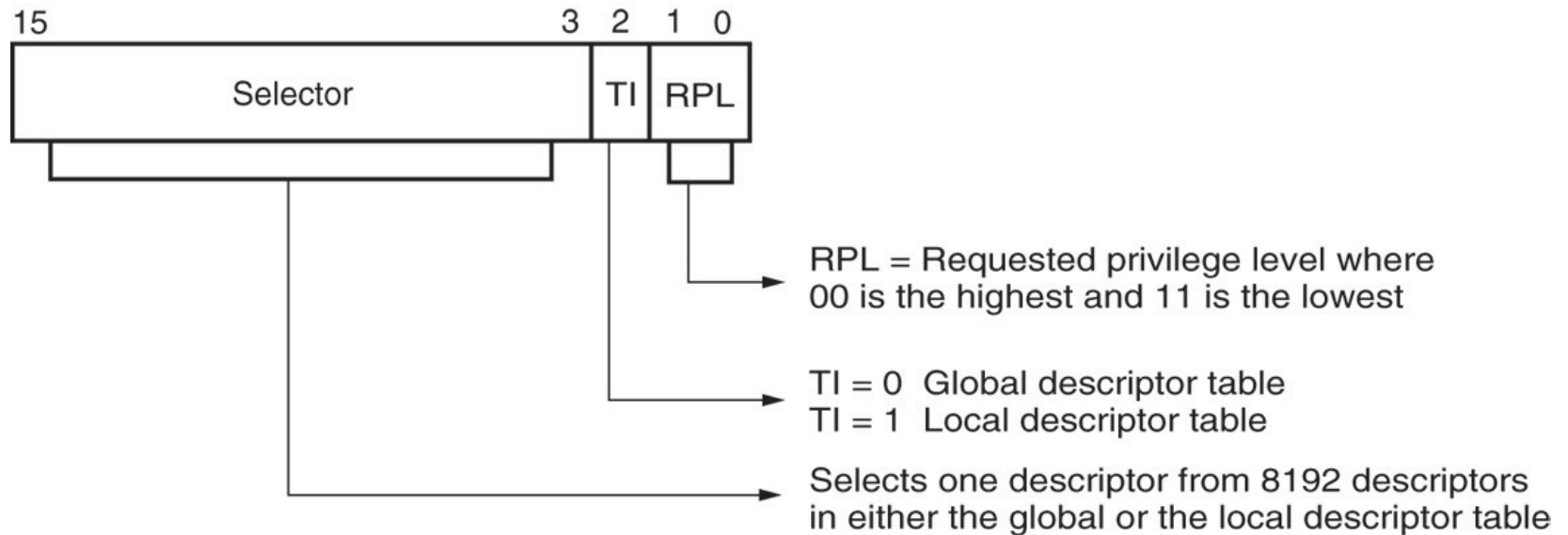
## Figure 2-7 The **access rights byte** for the 80286 through Core2 descriptor.



Note: Some of the letters used to describe the bits in the access rights bytes vary in Intel documentation.

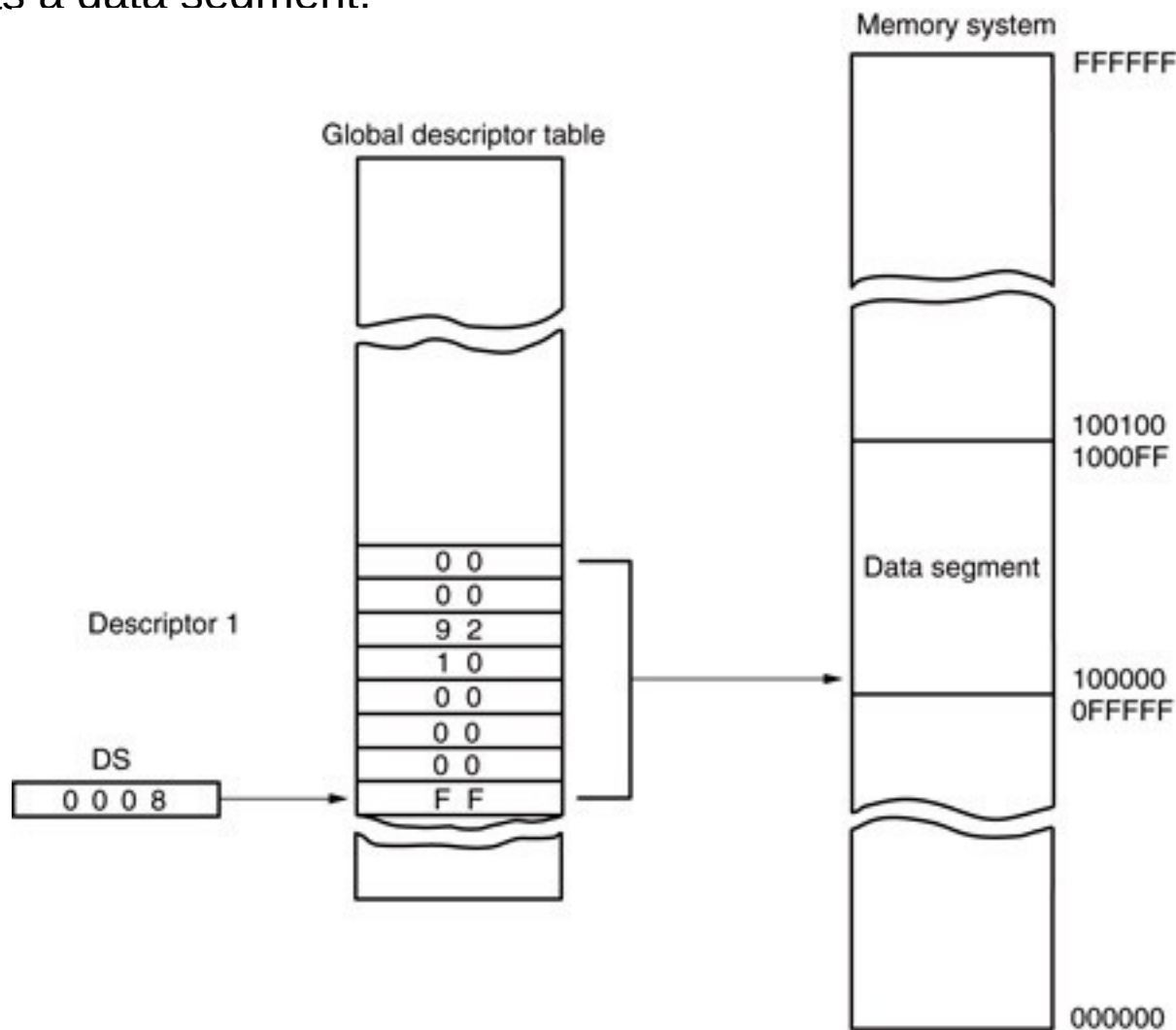
- Descriptors are chosen from the **descriptor table** by the segment register.
  - register contains a 13-bit selector field, a table selector bit, and requested privilege level field
- The **TI bit** selects either the global or the local descriptor table.
- **Requested Privilege Level** (RPL) requests the access privilege level of a memory segment.
  - If privilege levels are violated, system normally indicates an application or **privilege level violation**

**Figure 2-8** The contents of a **segment register** during **protected mode** operation of the 80286 through Core2 microprocessors.



- Figure 2–9 shows how the **segment register**, containing a selector, chooses a descriptor from the global descriptor table.
- The **entry** in the global descriptor table selects a segment in the memory system.
- Descriptor zero is called the **null descriptor**, must contain all zeros, and may not be used for accessing memory.

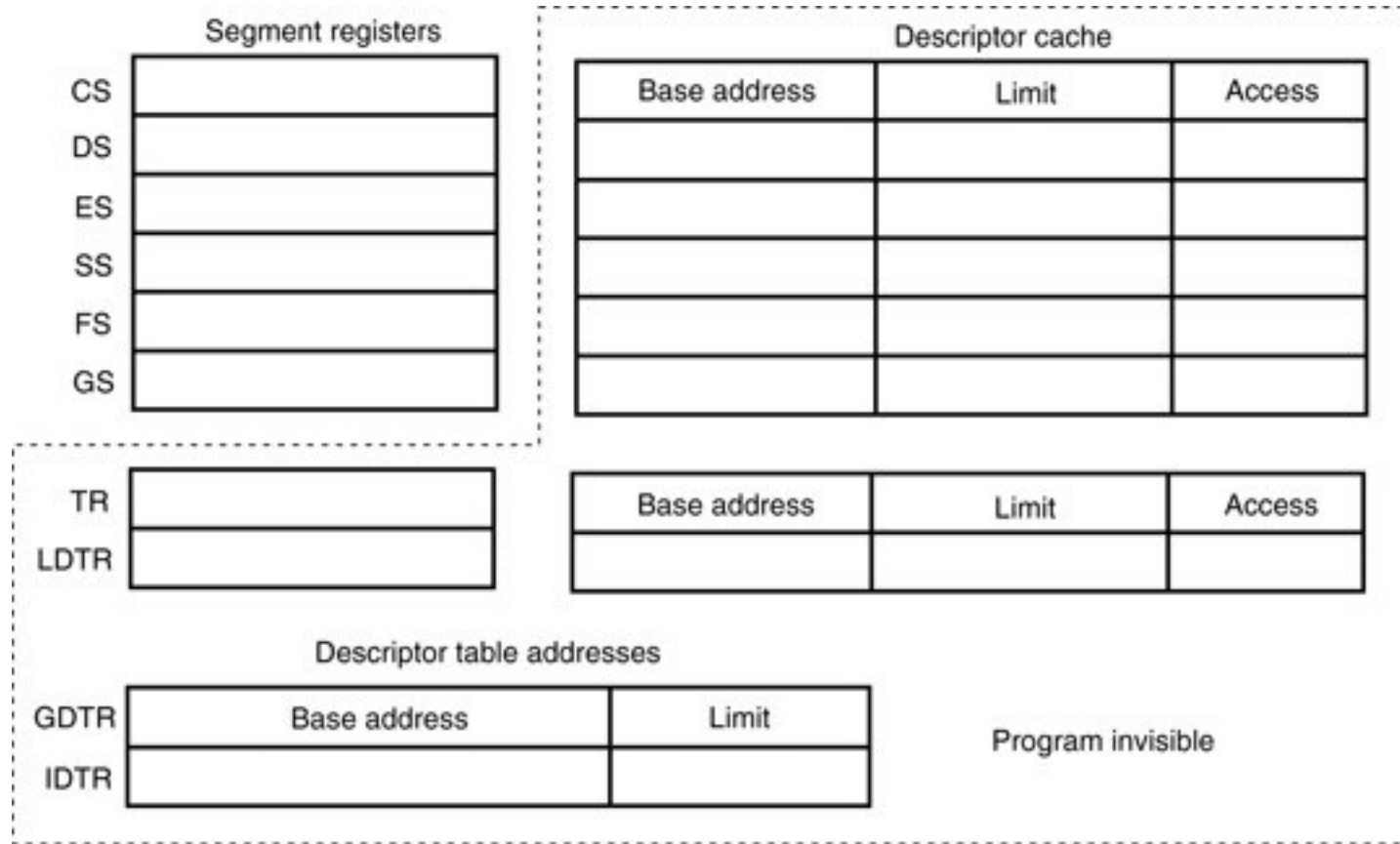
**Figure 2-9** Using the DS register to **select** a description from the global descriptor table. In this example, the DS register **accesses** memory locations 00100000H–001000FFH as a data segment.



# Program-Invisible Registers

- Global and local descriptor tables are found in the memory system.
- To access & specify the table addresses, 80286—Core2 contain **program-invisible registers**.
  - **not directly addressed by software**
- Each segment register contains a **program-invisible portion** used in the protected mode.
  - often called cache memory because cache is any memory that stores information

**Figure 2-10** The program-**invisible** register within the 80286–Core2 microprocessors.



**Notes:**

1. The 80286 does not contain FS and GS nor the program-invisible portions of these registers.
2. The 80286 contains a base address that is 24-bits and a limit that is 16-bits.
3. The 80386/80486/Pentium/Pentium Pro contain a base address that is 32-bits and a limit that is 20-bits.
4. The access rights are 8-bits in the 80286 and 12-bits in the 80386/80486/Pentium–Core2.



- When a new segment number is placed in a segment register, the microprocessor accesses a descriptor table and loads the descriptor into the program-invisible portion of the segment register.
  - held there and used to access the memory segment until the segment number is changed
- This allows the microprocessor to repeatedly access a memory segment **without referring** to the descriptor table.
  - hence the term **cache**

- The GDTR (**global descriptor table register**) and IDTR (**interrupt descriptor table register**) contain the **base address** of the descriptor table and its **limit**.
  - *when protected mode operation desired, address of the global descriptor table and its **limit** are loaded into the GDTR*
- The location of the local descriptor table is selected from the global descriptor table.
  - one of the global descriptors is set up to address the local descriptor table

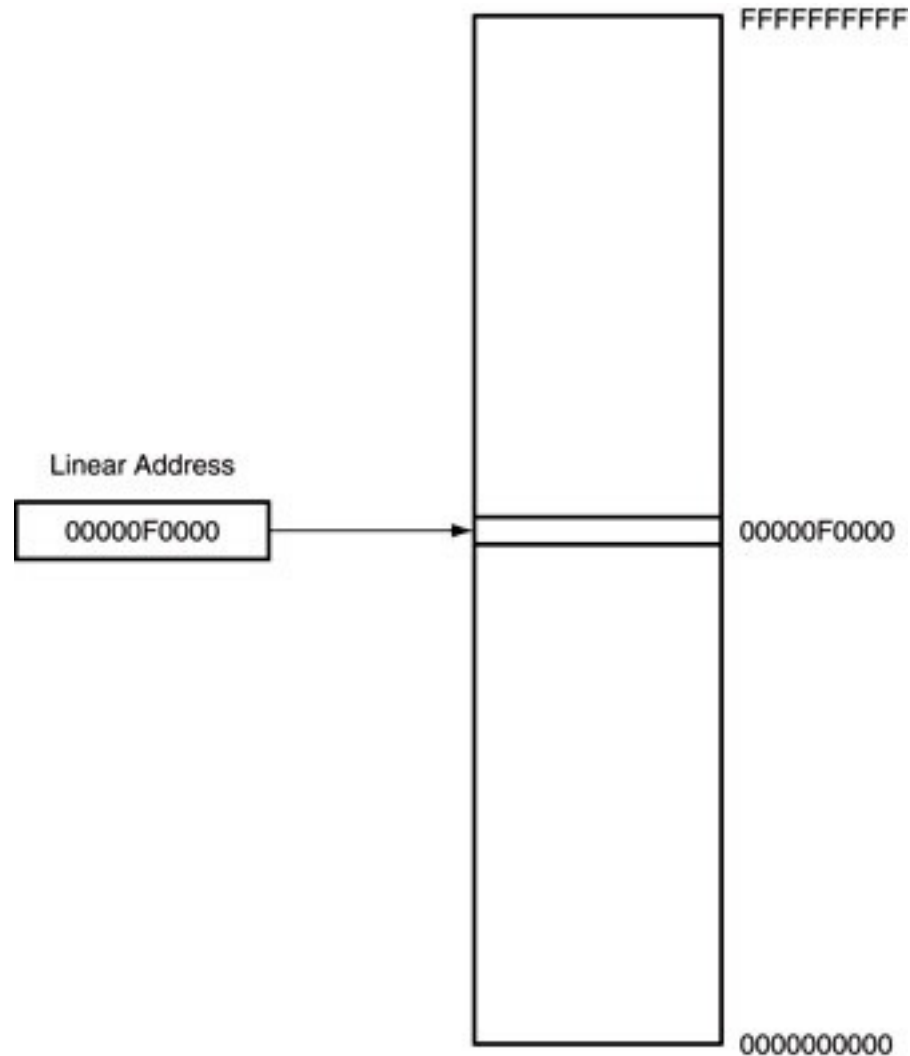
- To access the local descriptor table, the LDTR (**local descriptor table register**) is loaded with a selector.
  - **selector** accesses global descriptor table, & loads local descriptor table address, limit, & access rights into the cache portion of the LDTR
- The **TR** (**task register**) holds a selector, which accesses a descriptor that defines a task.
  - a task is most often a **procedure** or **application**
- Allows **multitasking** systems to switch tasks to another in a **simple** and **orderly** fashion.

# 2-5 Flat Mode Memory

- A *flat mode memory system* is one in which there is **no segmentation**.
  - does not use a segment register to address a location in the memory
- First byte address is at **00 0000 0000H**; the last location is at **FF FFFF FFFFH**.
  - address is **40-bits**
- The segment register still selects the privilege level of the software.

- Real mode system is *not* available if the processor operates in the 64-bit mode.
- **Protection** and **paging** are allowed in the 64-bit mode.
- The CS register is still used in the protected mode operation in the 64-bit mode.
- Most programs today are operated in the IA32 **compatible mode**.
  - current software operates properly, but **this will change in a few years** as memory becomes larger and most people have 64-bit computers

**Figure 2-15** The 64-bit **flat mode memory model**.



# SUMMARY

- The programming model of the 8086 through 80286 contains 8- and 16-bit registers.
- The programming model of the 80386 and above contains 8-, 16-, and 32-bit extended registers as well as two **additional** 16-bit segment registers: FS and GS.

# SUMMARY

(*cont.*)

- 8-bit registers are AH, AL, BH, BL, CH, CL, DH, and DL.
- 16-bit registers are AX, BX, CX, DX, SP, BP, DI, and SI.
- The segment registers are CS, DS, ES, SS, FS, and GS.
- 32-bit extended registers are EAX, EBX, ECX, EDX, ESP, EBP, EDI, and ESI.



# SUMMARY

(*cont.*)

- The 64-bit registers in a Pentium 4 with 64-bit extensions are RAX, RBX, RCX, RDX, RSP, RBP, RDI, RSI, and R8 through R15.
- In addition, the microprocessor contains an instruction pointer (IP/EIP/RIP) and flag register (FLAGS, EFLAGS, or RFLAGS).
- All real mode memory addresses are a combination of a **segment address** plus an **offset address**.

# SUMMARY

(*cont.*)

- The starting location of a segment is defined by the 16-bit number in the segment register that is appended with a hexadecimal zero at its rightmost end.
- The offset address is a **16-bit** number added to the **20-bit** segment address to form the real mode memory address.
- All instructions (code) are accessed by the combination of CS (**segment address**) plus IP or EIP (**offset address**).

# SUMMARY

(*cont.*)

- Data are normally referenced through a combination of the DS (data segment) and either an offset address or the contents of a register that contains the offset address.
- The 8086-Core2 use BX, DI, and SI as default offset registers for data if 16-bit registers are selected.
- The 80386 and above can use the 32-bit registers EAX, EBX, ECX, EDX, EDI, and ESI as default offset registers for data.

# SUMMARY

(cont.)

- Protected mode operation allows memory above the first 1M byte to be accessed by the 80286 through the Core2 microprocessors.
- This **extended memory system (XMS)** is accessed via a segment address plus an offset address, just as in the real mode.
- In the protected mode, the segment **starting address** is stored in a **descriptor** that is selected by the segment register.

# SUMMARY

(*cont.*)

- A protected mode **descriptor** contains a **base address**, **limit**, and **access rights** byte.
- The base address locates the starting address of the memory segment; the **limit** **defines the last location of the segment**.
- The access rights byte defines how the memory segment is accessed via a program.

# SUMMARY

(cont.)

- The 80286 microprocessor allows a memory segment to start at any of its **16M** bytes of memory using a **24-bit** base address.
- The 80386 and above allow a memory segment to begin at any of its **4G** bytes of memory using a **32-bit** base address.
- This allows an 80286 memory segment limit of 64K bytes, and an 80386 and above memory segment limit of either 1M bytes.

# SUMMARY

(*cont.*)

- The segment register contains three fields of information in the protected mode.
- The leftmost **13 bits** of the segment register address one of **8192 descriptors** from a descriptor table.
- The program-invisible registers are used by the 80286 and above to access the descriptor tables.

# SUMMARY

(*cont.*)

- Each segment register contains a cache portion that is used in protected mode to hold the base **address**, **limit**, and **access** rights acquired from a descriptor.
- The cache allows the microprocessor to access the memory segment without again referring to the descriptor table until the segment register's contents are changed.



# SUMMARY

(*cont.*)

- A memory page is 4K bytes in length. The linear address, as generated by a program, can be mapped to any physical address through the paging mechanism found within the 80386 through the Pentium 4.
- Memory paging is accomplished through control registers CR0 and CR3.
- The PG bit of CR0 enables paging, and the contents of CR3 addresses the page directory.

# SUMMARY

(cont.)

- The page directory contains up to **1024 page table addresses** that are used to access paging tables.
- The page table contains 1024 entries that locate the physical address of a 4K-byte memory page.
- The TLB (**translation look-aside buffer**) caches the 32 most recent page table translations.

# SUMMARY

- The **flat mode memory** contains **1T byte** of memory using a **40-bit** address.
- In the future, Intel plans to increase the address width to **52 bits** to access **4P** bytes of memory.
- The flat mode is only available in the Pentium 4 and Core2 that have their 64-bit extensions enabled.