Lecture (02)

The Microprocessor and Its Architecture

By:

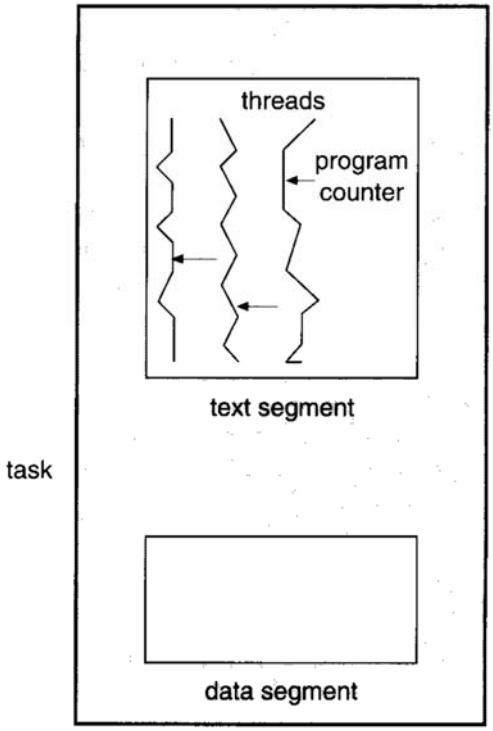
Dr. Ahmed ElShafee

FACULTY OF ENGINEERING
AHRAM CANADIAN UNIVERSITY

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

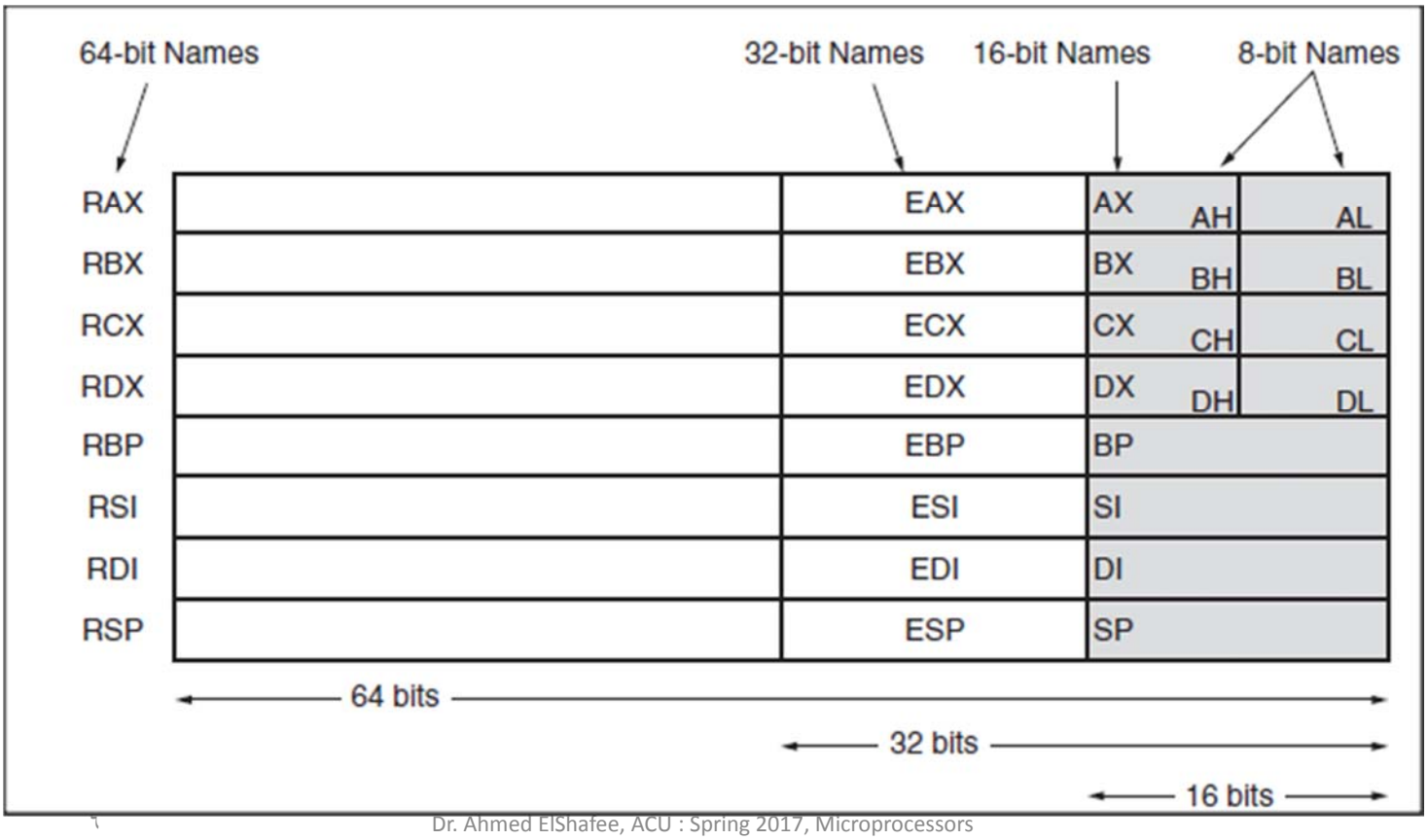


The Programming Model

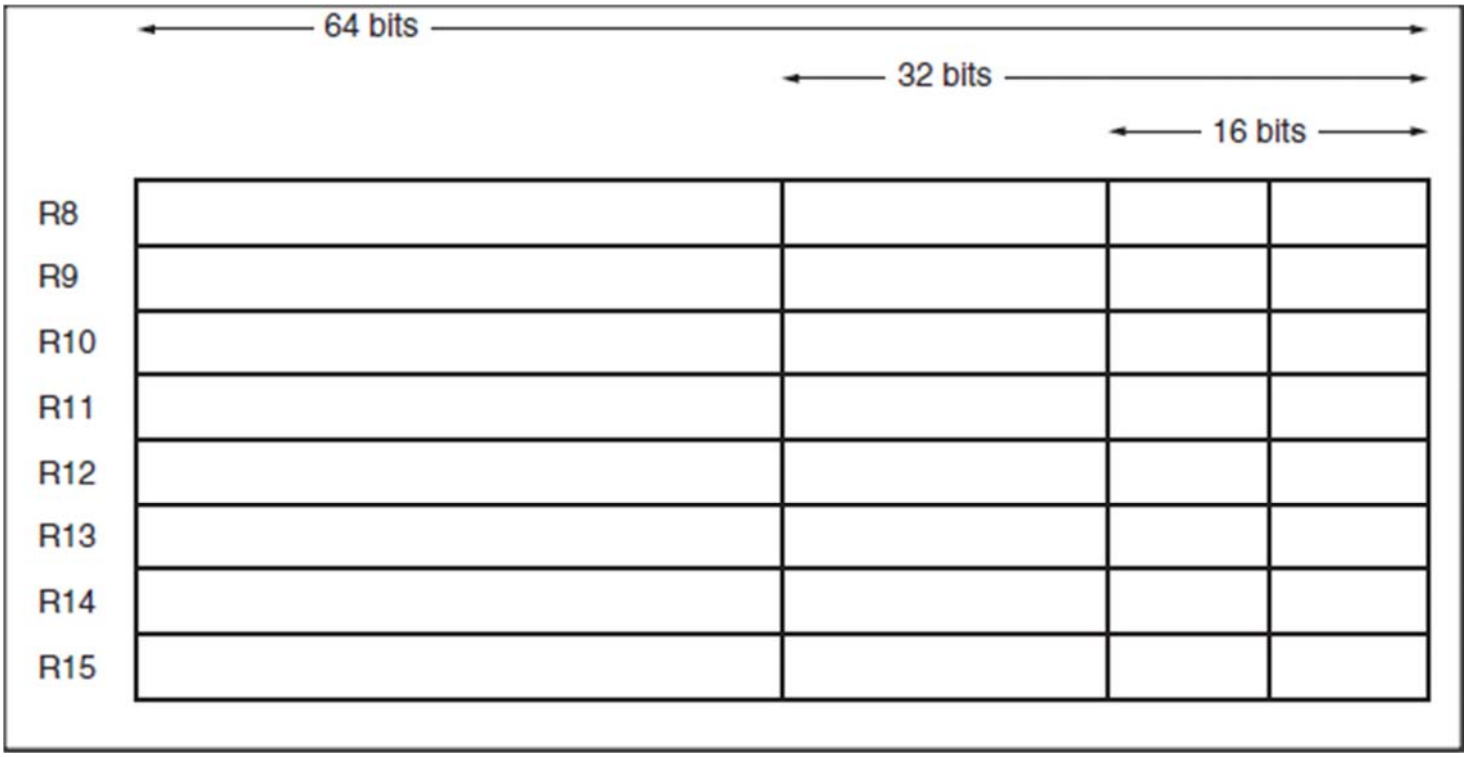
- 8086 through Core2 considered **program visible**.
  - registers are used during programming and are specified by the instructions
- Other registers considered to be **program invisible**.
  - not addressable directly during applications programming

- 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 (following) illustrates the programming model 8086 through Core2 microprocessor.
  - including the 64-bit extensions

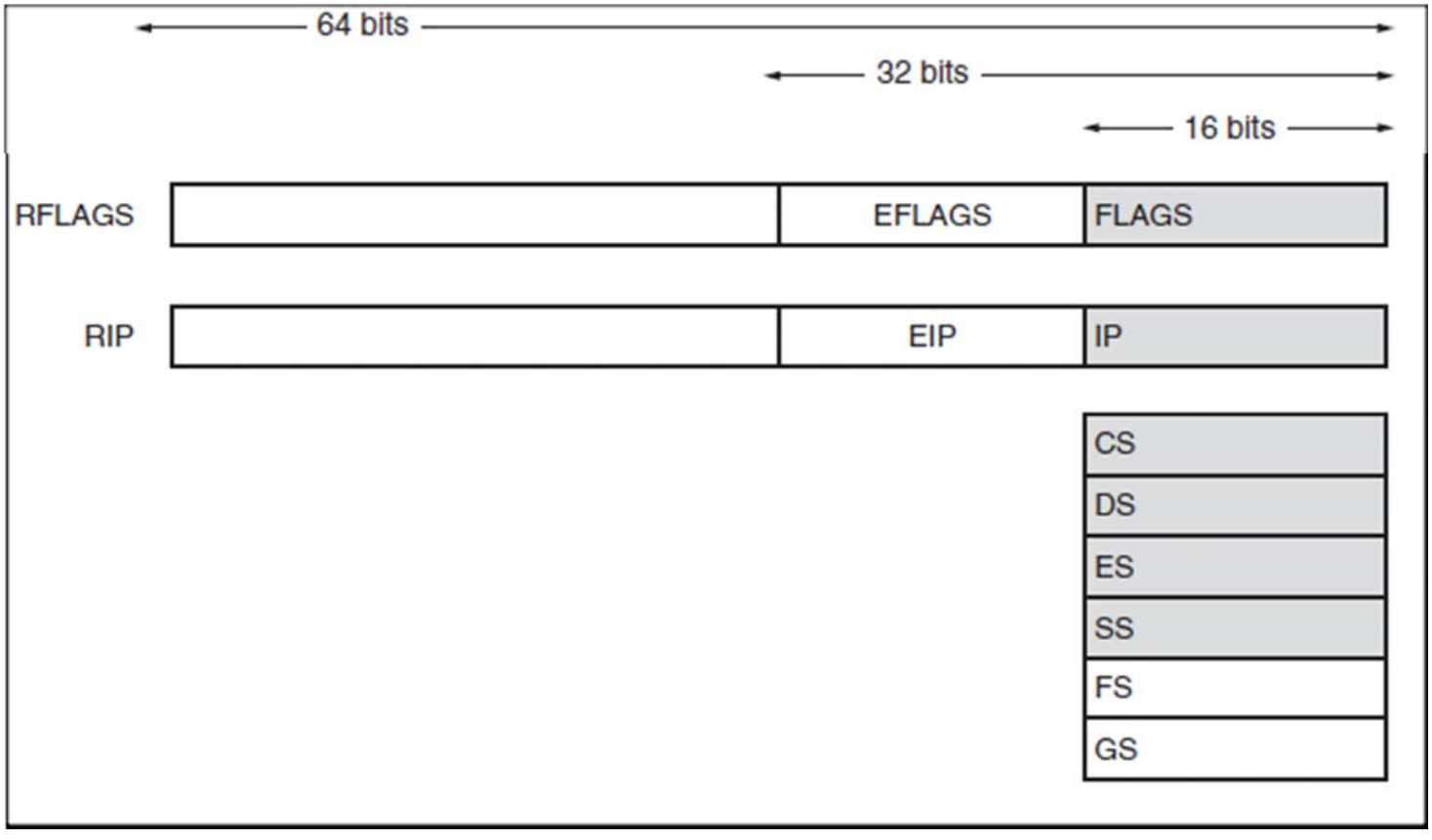
The programming model of the 8086 through the Core2 microprocessor including the 64-bit extensions.



The programming model of the 8086 through the Core2 microprocessor including the 64-bit extensions.



The programming model of the 8086 through the Core2 microprocessor including the 64-bit extensions.

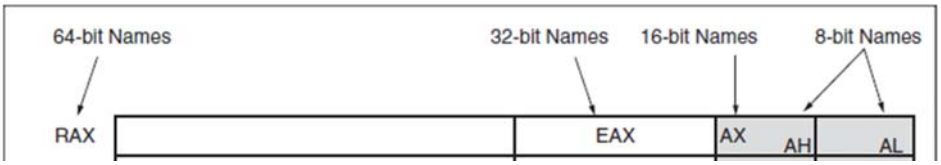


Multipurpose Registers

- **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).

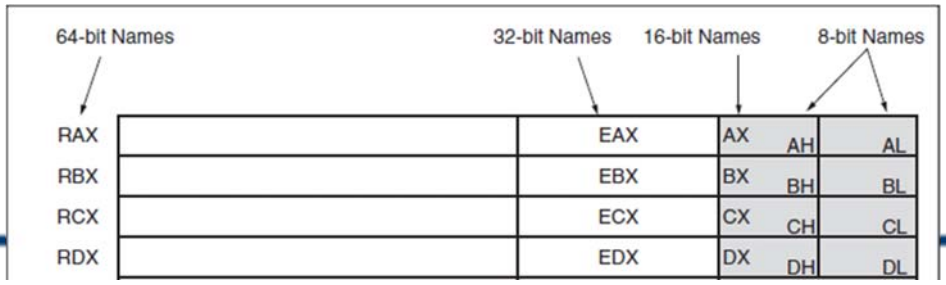
- The accumulator is used for instructions such as multiplication, division, and some of the adjustment instructions.

- Intel plans to expand the address bus to 52 bits to address 4P (2^{52}~10^{15} =peta) bytes of memory.

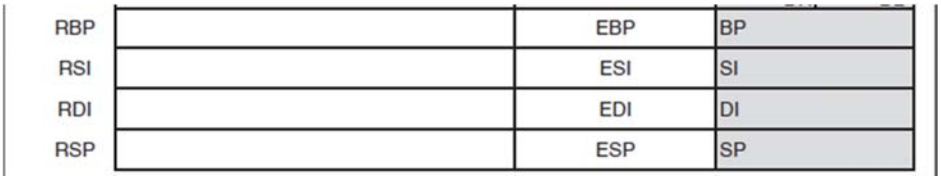


Address Space (Main Memory: RAM)

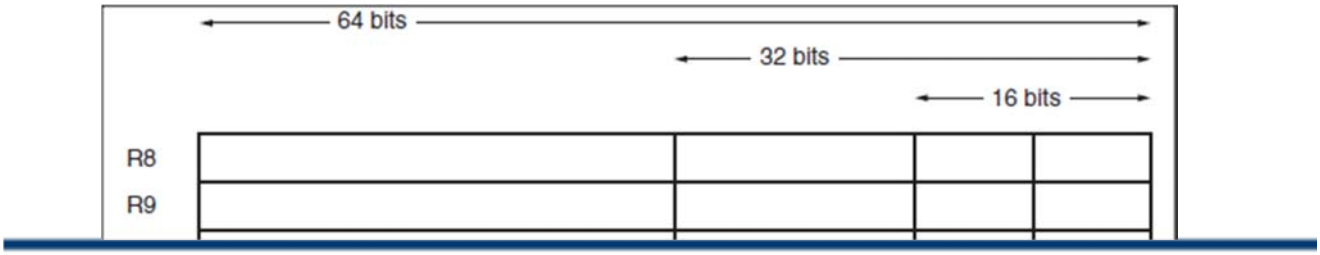
- Address bus:16 bit →Address Space:64 KBytes
- Address bus:20 bit →Address Space:1 MBytes
- Address bus:32 bit →Address Space:4 GBytes
- Address bus:34 bit →Address Space:16GBytes
- Address bus:36 bit →Address Space:64GBytes
- Address bus:38 bit →Address Space:256GBytes
- Address bus:52 bit →Address 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



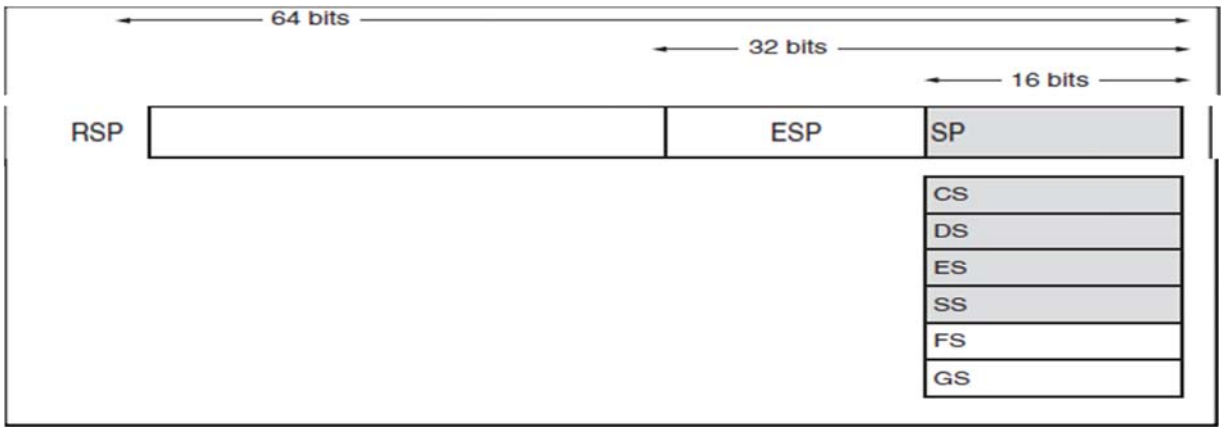
- **RBP,** as RBP, EBP, or BP.
  - points to a memory **(base pointer)** location for memory data transfers
- **RDI** addressable as RDI, EDI, or DI.
  - often addresses **(destination index)** string destination data for the string instructions
- **RSI** used as RSI, ESI, or SI.
  - the **(source index)**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



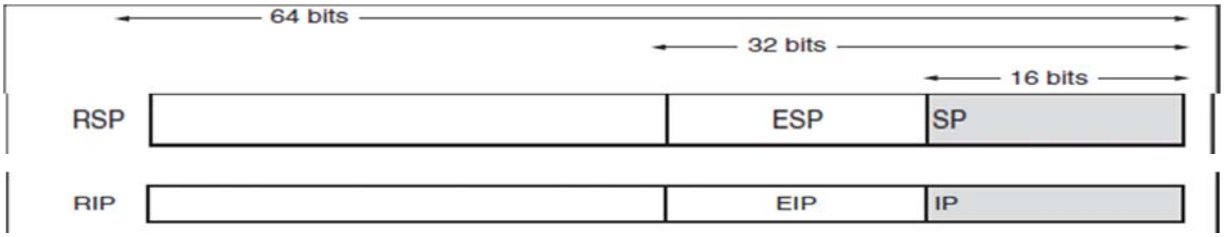
- **R8 - R15** found in the Pentium 4 and Core2 if 64-bit extensions are enabled.
  - data are addressed as 64-, 32-, 16-, or 8-bit sizes and are of general purpose
- Most applications will not use these registers until 64-bit processors are common.
  - the 8-bit portion is the rightmost 8-bit only
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

Special-Purpose Registers

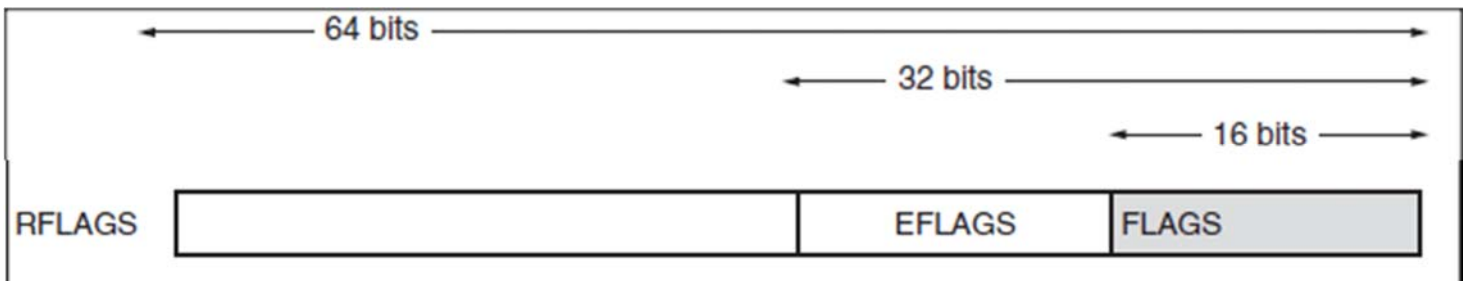
- Include RIP, RSP, and RFLAGS
  - segment registers include CS, DS, ES, SS, FS, and GS



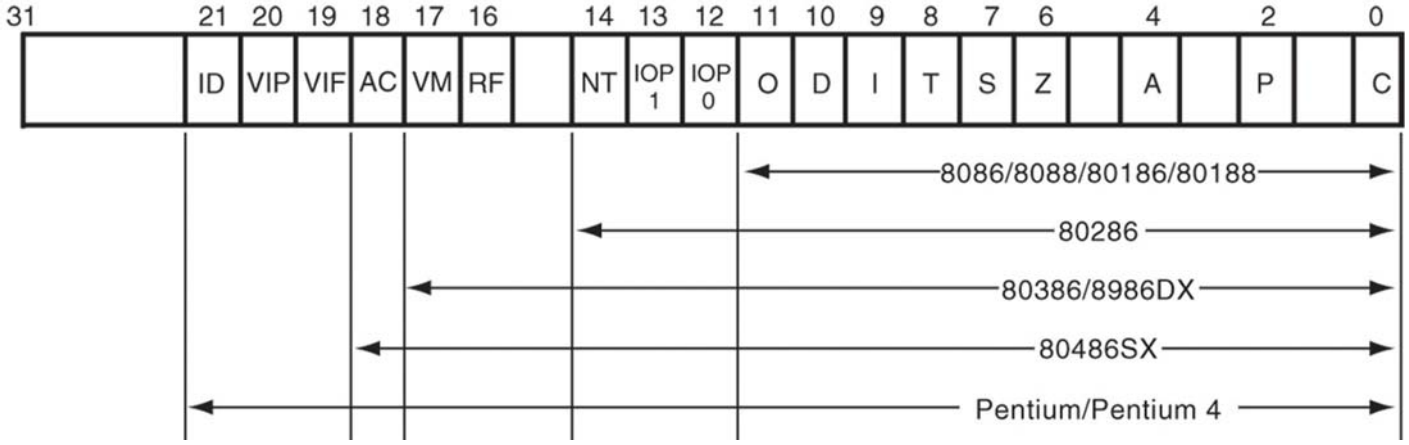
- **RIP** addresses the next instruction in a section of memory.
  - defined as **(instruction pointer)** a code segment
- **RSP** addresses an area of memory called the stack.
  - the **(stack pointer)** stores data through this pointer



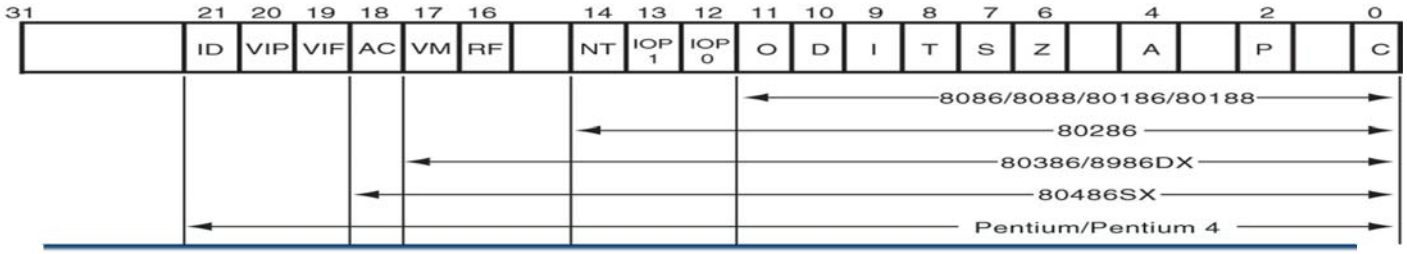
- **RFLAGS** indicate the condition of the microprocessor and control its operation.
- Flags are upward-compatible from the 8086/8088 through Core2 .
- The rightmost five and the overflow flag are changed by most arithmetic and logic operations.
  - although data transfers do not affect them



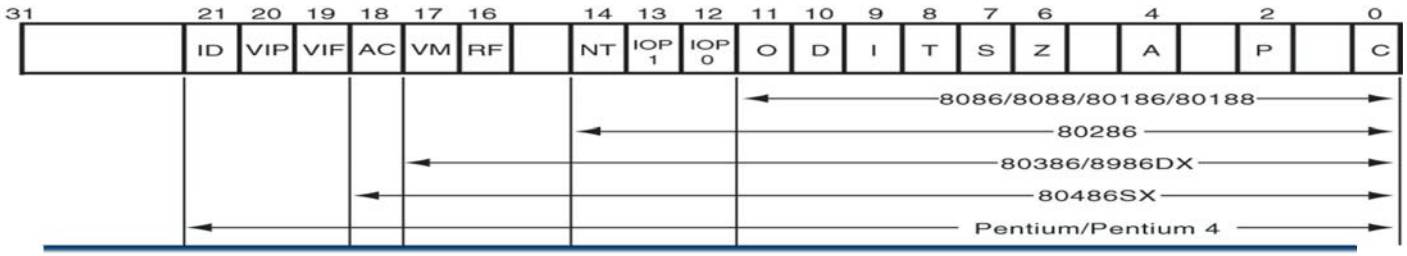
- The EFLAG and FLAG register counts for the entire 8086 and Pentium microprocessor family.



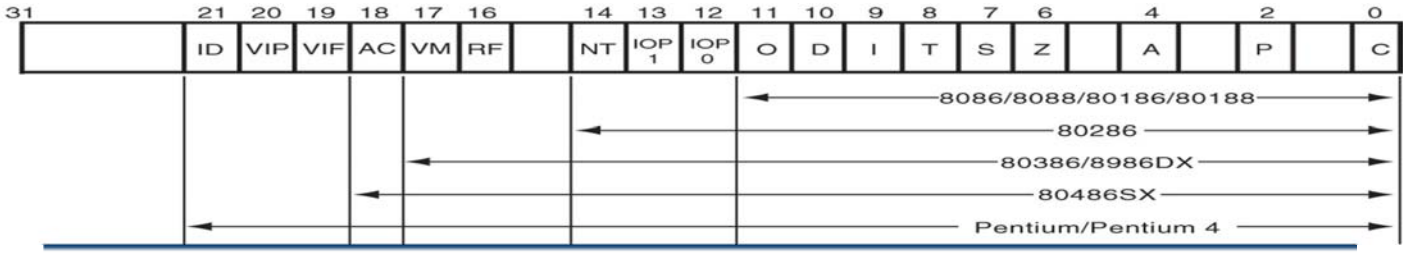
- Flags never change for any data transfer or program control operation.
- Some of the flags are also used to control features found in the microprocessor.



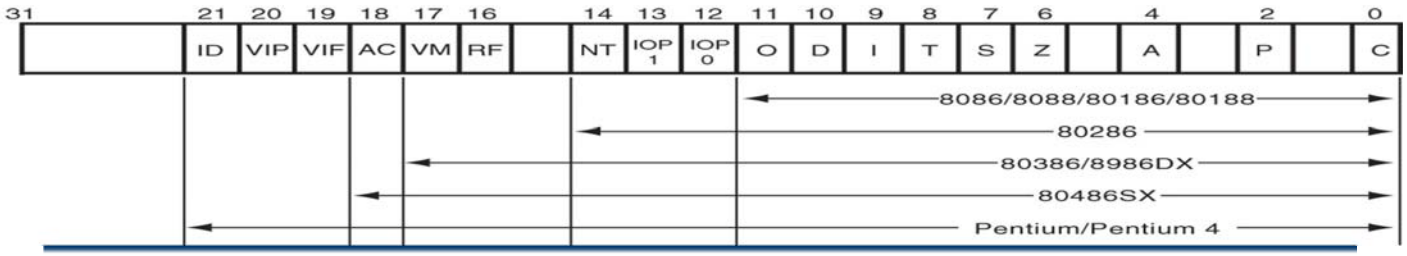
- Flag bits, with a brief description of function.
- **C (carry)** holds the carry after addition or borrow after subtraction.
  - also indicates error conditions
- **P (parity)** is the count of ones in a number expressed as even or odd. Logic 0 for odd parity; logic 1 for even parity.
  - if a number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



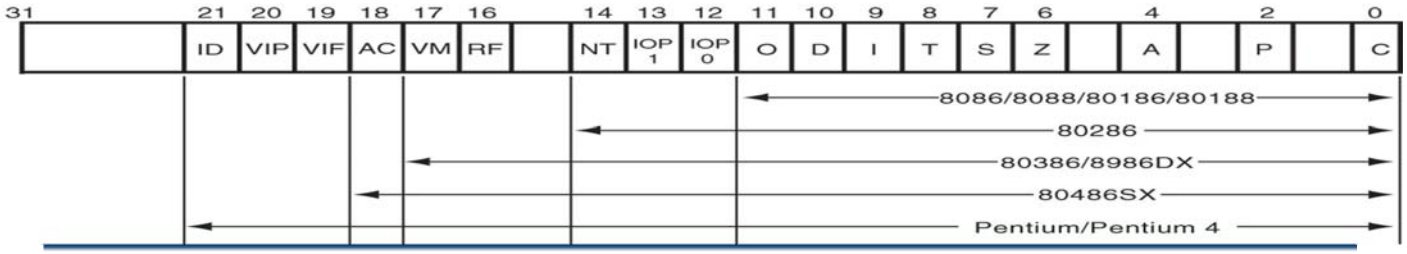
- **A (auxiliary carry)** holds the carry (half-carry) after addition or the borrow after subtraction between bit positions 3 and 4 of the result.
- **Z (zero)** shows that the result of an arithmetic or logic operation is zero.
- **S (sign)** flag holds the arithmetic sign of the result after an arithmetic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



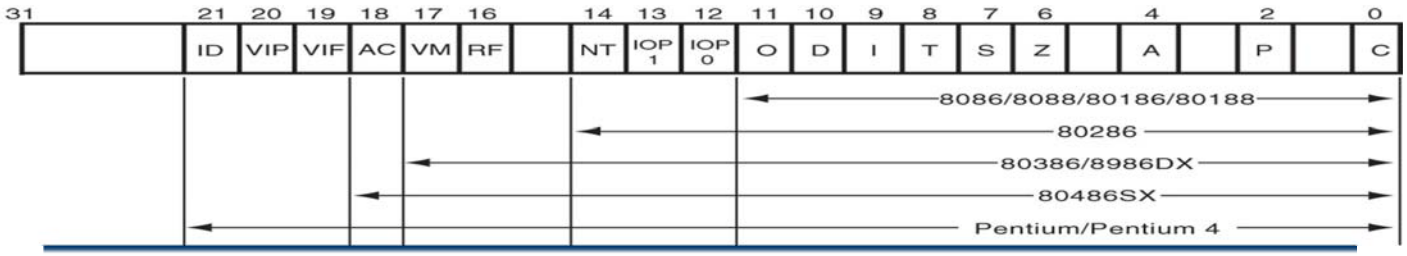
- **I (interrupt)** controls operation of the INTR (interrupt request) input pin.
- **D (direction)** selects increment or decrement mode for the DI and/or SI registers.
- **O (overflow)** occurs when signed numbers are added or subtracted.
  - an overflow indicates the result has exceeded the capacity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



- **IOPL** used in protected mode operation to select the privilege level for I/O devices.
- **NT (nested task)** flag indicates the current task is nested within another task in protected mode operation.
- **RF (resume)** used with debugging to control resumption of execution after the next instruction.
- **VM (virtual mode)** 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.

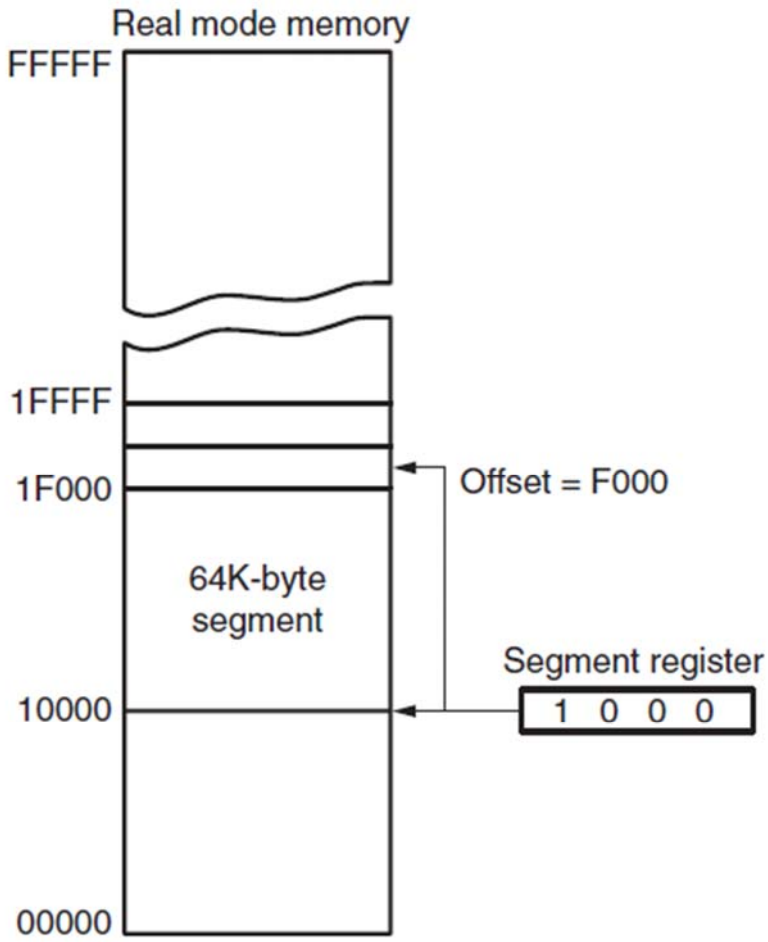
REAL MODE MEMORY ADDRESSING

- 80286 and above operate in either the real or protected mode.
- **Real mode operation** allows addressing of 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

Segments and Offsets

- 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
  - **offset address** selects any location within the 64K byte memory segment
- Figure shows how the **segment plus offset** addressing scheme selects a memory location.

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



- this shows a memory segment beginning at 10000H, ending at location IFFFFH
  - 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 is64K 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

- Once the beginning address is known, the **ending address** is found by adding FFFFH.
  - because a real mode segment of memory is64K 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	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.

- 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 shows a system that contains four memory segments.
  - a memory segment can touch or overlap if 64K bytes of memory are not required for a segment

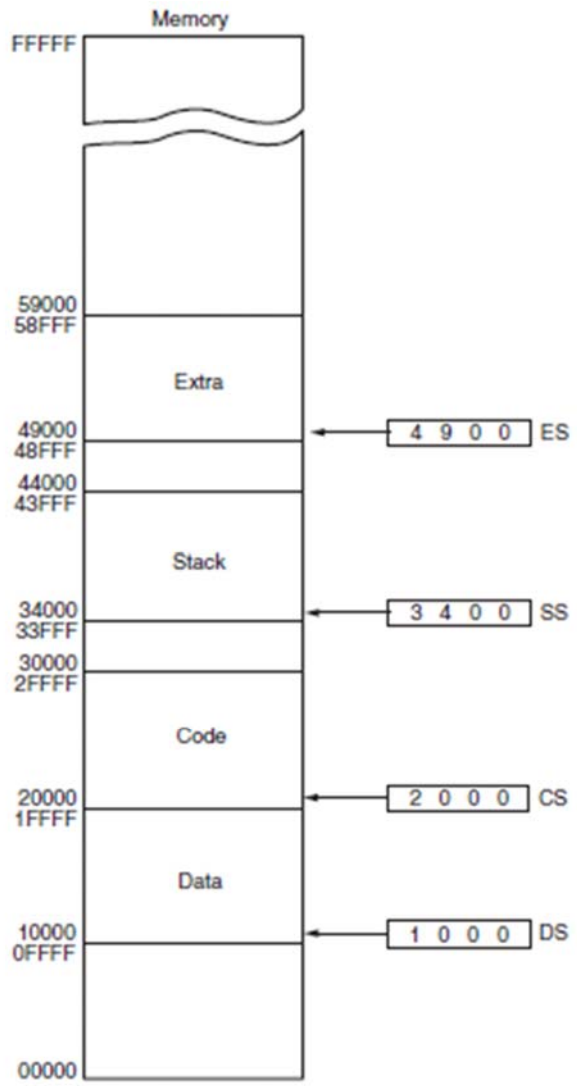
Default 16-bit segment and offset combinations.

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

Default 32-bit segment and offset combinations.

Segment	*Offset*	*Special Purpose*
CS	EIP	Instruction address
SS	ESP or EBP	Stack address
DS	EAX, EBX, ECX, EDX, ESI, EDI, an 8- or 32-bit number	Data address
ES	EDI for string instructions	String destination address
FS	No default	General address
GS	No default	General address

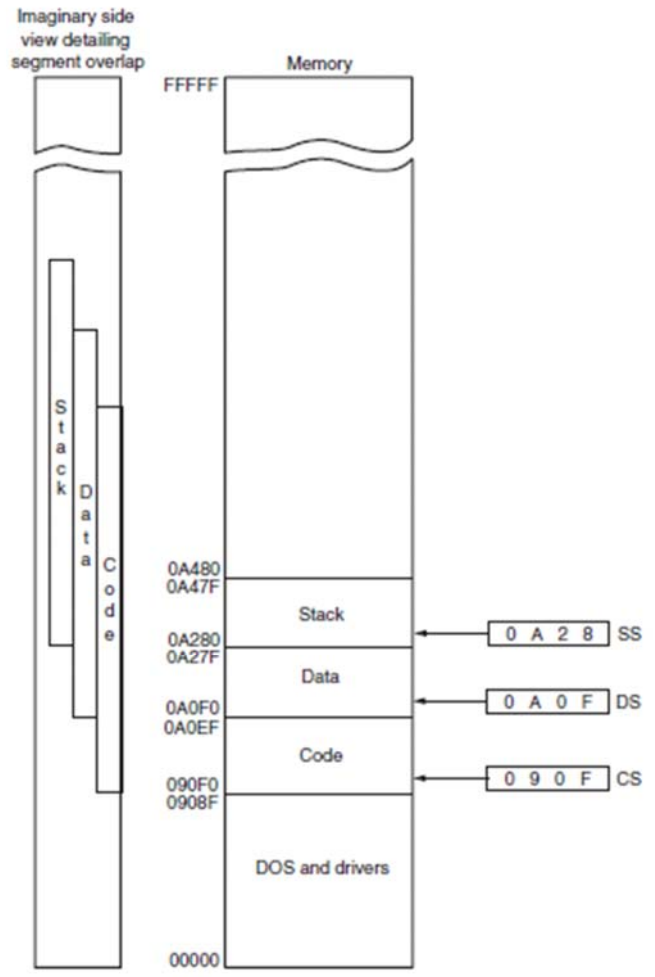
A memory system showing the placement of four memory segments.



-
  - 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

An application program containing a code, data, and stack segment loaded into a DOS system memory.

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.