Intel IA-32 Architecture

80x86 Evolution

- 4004
  - 4-bit microprocessor.
  - 4KB main memory.
  - 45 instructions.
  - 50 KIPS
- 8008 (1971)
  - **8-bit** version of 4004.
  - **16KB** main memory.
  - 48 instructions.
- 8080 (1973)
  - 8-bit microprocessor.
  - **64KB** main memory.
  - 2 microseconds clock cycle time; 500,000 instructions/sec.
  - 10X faster than 8008.

80x86 Evolution

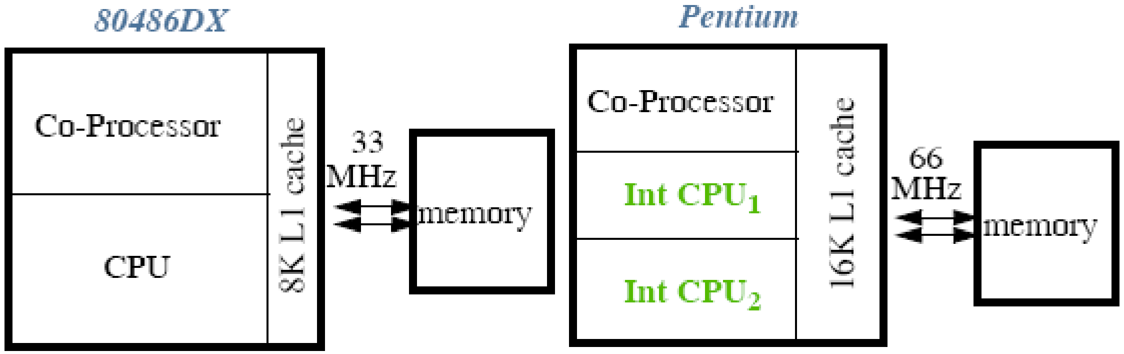
- 8085 (1977)
  - 8-bit microprocessor - upgraded version of the 8080.
  - 64KB main memory.
  - **1.3 microseconds** clock cycle time; 769,230 instructions/sec.
  - 246 instructions.
  - Intel sold 100 million copies of this 8-bit microprocessor.
- 8086 (1978) / 8088 (1979)
  - **16-bit** microprocessor.
  - **1MB** main memory.
  - 2.5 MIPS (400 ns).
  - 4- or 6-byte instruction cache.
  - Other improvements included more registers and additional instructions.
- 80286 (1982)
  - 16-bit microprocessor very similar in instruction set to the 8086.
  - **16MB** main memory.
  - 4.0 MIPS (250 ns/8MHz).

80x86 Evolution

- 80386 (1985)
  - **32-bit** microprocessor.
  - **4GB** main memory.
  - 12-33MHz.
  - Memory management unit added.
  - Variations: DX, EX, SL, SLC (cache) and SX.
    - 80386SX: 16MB through a 16-bit data bus and 24 bit address bus.
- 80486 (1989)
  - 32-bit microprocessor, 32-bit data bus and 32-bit address bus.
  - 4GB main memory.
  - 20-50MHz. Later at 66 and 100MHz
  - Incorporated an 80386-like microprocessor, 80387-like floating point coprocessor and an 8K byte cache on one package.
  - About half of the instructions executed in 1 clock instead of 2 on the 386.
  - Variations: SX, DX2, DX4.
    - DX2: Double clocked version:
    - 66MHz clock cycle time with memory transfers at 33MHz.

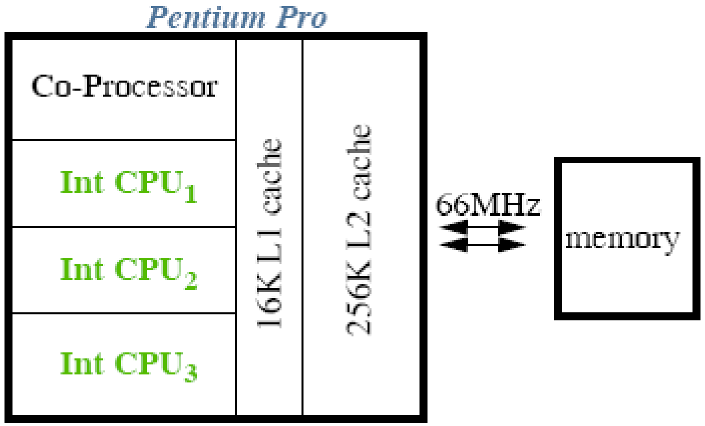
80x86 Evolution

- Pentium (1993)
  - 32-bit microprocessor, 64-bit data bus and 32-bit address bus.
  - 4GB main memory.
  - 60, 66, 90MHz.
    - 1-and-1/2 100MHz version.
    - Double clocked 120 and 133MHz versions.
    - Fastest version is the 233MHz (3-and-1/2 clocked version).
  - **16KB L1 cache** (split instruction/data: 8KB each).
  - Memory transfers at **66MHz** (instead of 33MHz).
  - Dual integer processors.



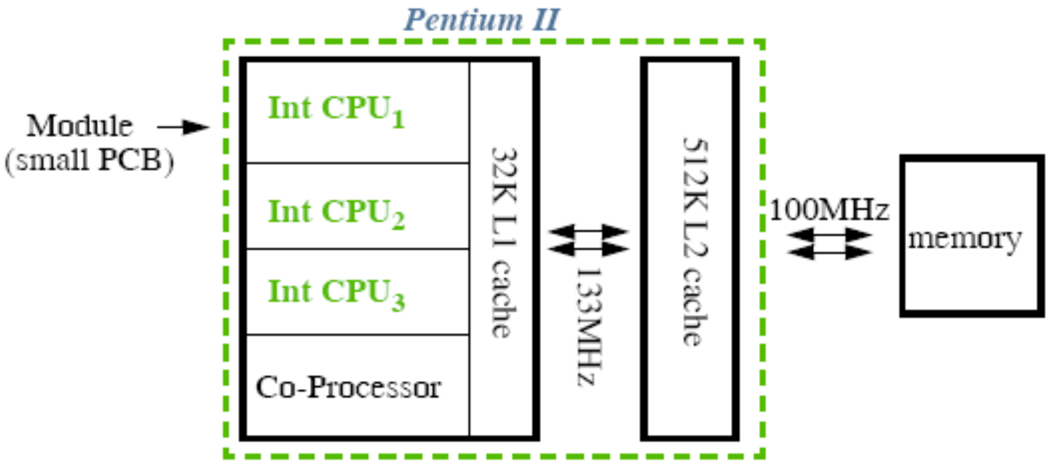
80x86 Evolution

- Pentium Pro (1995)
  - 32-bit microprocessor, 64-bit data bus and 36-bit address bus.
  - **64GB** main memory.
  - Starts at 150MHz.
  - 16KB L1 cache (split instruction/data: 8KB each).
  - **256KB L2 cache**
  - Memory transfers at 66MHz.
  - **3 integer processors**



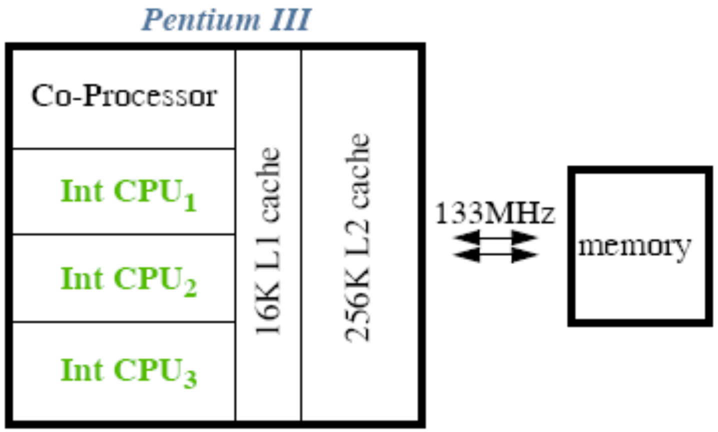
80x86 Evolution

- Pentium II (1997)
  - 32-bit microprocessor, 64-bit data bus and 36-bit address bus.
  - 64GB main memory.
  - Starts at 266MHz.
  - **32KB** split instruction/data L1 caches (16KB each).
  - **Module integrated 512KB L2 cache** (133MHz).
  - Memory transfers at 66MHz to 100MHz (1998).



80x86 Evolution

- Pentium III (1999)
  - 32-bit microprocessor, 64-bit data bus and 36-bit address bus.
  - 64GB main memory.
  - 800MHz and above.
  - 32KB split instruction/data L1 caches (16KB each).
  - **On-chip 256KB L2 cache**
  - Memory transfers 100MHz to 133MHz.



80x86 Evolution

- Pentium IV (2000~2006)
  - 1.4 to 1.9GHz and the latest at 3.20 GHz and 3.46GHz (Hyper-Threading)!
  - 1MB/512KB/256KB L2 cache.
  - 800 MHz (about 6.4GB/s)/533 MHz (4.3 GB/s)/ 400MHz (3.2 GB/s) system bus.
  - 1066 MHz front side bus just available.
  - Specialized for streaming video, game and DVD applications (144 new SIMD 128-bit instructions).
  - 0.13um, more than 55 million transistors, 60nm transistors.
  - Newer ones are in 90nm transistors, more than 125 million possible.

80x86 Evolution

- The Intel Xeon Processor (2001-2007)
- The Intel Pentium M Processor (2003-2006)
  - Support for Intel Architecture with Dynamic Execution
  - On-die, primary 32-KByte instruction cache and 32-KByte write-back data cache
  - On-die, second-level cache (up to 2 MByte) with Advanced Transfer Cache Architecture
  - Advanced Branch Prediction and Data Prefetch Logic
  - Support for MMX technology, Streaming SIMD instructions, and the SSE2 instruction set
  - A 400 or 533 MHz, Source-Synchronous Processor System Bus
  - Advanced power management using Enhanced Intel SpeedStep® technology
- The Intel Pentium Processor Extreme Edition (2005)
- The Intel Core Duo and Intel Core Solo Processors (2006-2007)
  - Intel® Smart Cache which allows for efficient data sharing between two processor cores
  - Improved decoding and SIMD execution
  - Intel® Dynamic Power Coordination and Enhanced Intel® Deeper Sleep to reduce power consumption
  - Intel® Advanced Thermal Manager which features digital thermal sensor interfaces
  - Support for power-optimized 667 MHz bus
- The Intel Xeon Processor 5100 Series and Intel Core 2 Processor Family (2006-Current)

80x86 Evolution

- The Intel® Xeon® Processor 5100, 5300 Series and Intel® Core™2 Processor Family (2006)
  - Intel® Wide Dynamic Execution to increase performance and execution throughput
  - Intel® Intelligent Power Capability to reduce power consumption
  - Intel® Advanced Smart Cache which allows for efficient data sharing between two processor cores
  - Intel® Smart Memory Access to increase data bandwidth and hide latency of memory accesses
- The Intel® Xeon® Processor 5200, 5400, 7400 Series and Intel® Core™2 Processor Family (2007)
  - A radix-16 divider, faster OS primitives further increases the performance of Intel® Wide Dynamic Execution.
  - Improves Intel® Advanced Smart Cache with Up to 50% larger level-two cache and up to 50% increase in wayset associativity.
  - A 128-bit shuffler engine significantly improves the performance of Intel® Advanced Digital Media Boost and SSE4.
- The Intel® Atom™ Processor Family (2008)
  - Intel® Hyper-Threading Technology
  - Deep Power Down Technology with Dynamic Cache Sizing
  - Support for instruction set extensions up to and including Supplemental Streaming SIMD Extensions 3 (SSSE3).
  - Support for Intel® Virtualization Technology
  - Support for Intel® 64 Architecture (excluding Intel Atom processor Z5xx Series)
- The Intel® Atom™ Processor Family Based on Silvermont Microarchitecture (2013)

80x86 Evolution

- The Intel® Core™i7 Processor Family (2008)
  - Intel® Turbo Boost Technology converts thermal headroom into higher performance.
  - Intel® HyperThreading Technology in conjunction with Quadcore to provide four cores and eight threads.
  - Dedicated power control unit to reduce active and idle power consumption.
  - Integrated memory controller on the processor supporting three channel of DDR3 memory.
  - 8 MB inclusive Intel® Smart Cache.
  - Support for SSE4.2 and SSE4.1 instruction sets.
  - Second generation Intel Virtualization Technology.
- The Intel® Xeon® Processor 7500 Series (2010)
  - Up to eight cores per physical processor package.
  - Up to 24 MB inclusive Intel®Smart Cache.
  - Provides Intel®Scalable Memory Interconnect (Intel®SMI) channels with Intel®7500 Scalable Memory Buffer to connect to system memory.

80x86 Evolution

- 2010 Intel® Core™ Processor Family (2010)
  - Deliver smart performance using Intel Hyper-Threading Technology plus Intel Turbo Boost Technology.
  - Enhanced Intel Smart Cache and integrated memory controller.
  - Intelligent power gating.
- The Intel® Xeon® Processor 5600 Series (2010)
  - Up to six cores per physical processor package.
  - Up to 12 MB enhanced Intel® Smart Cache.
  - Flexible Intel Virtualization Technologies across processor and I/O.
- The Second Generation Intel® Core™ Processor Family (2011)
  - Intel Turbo Boost Technology for Intel Core i5 and i7 processors
  - Intel Hyper-Threading Technology.
  - Enhanced Intel Smart Cache and integrated memory controller.

80x86 Evolution

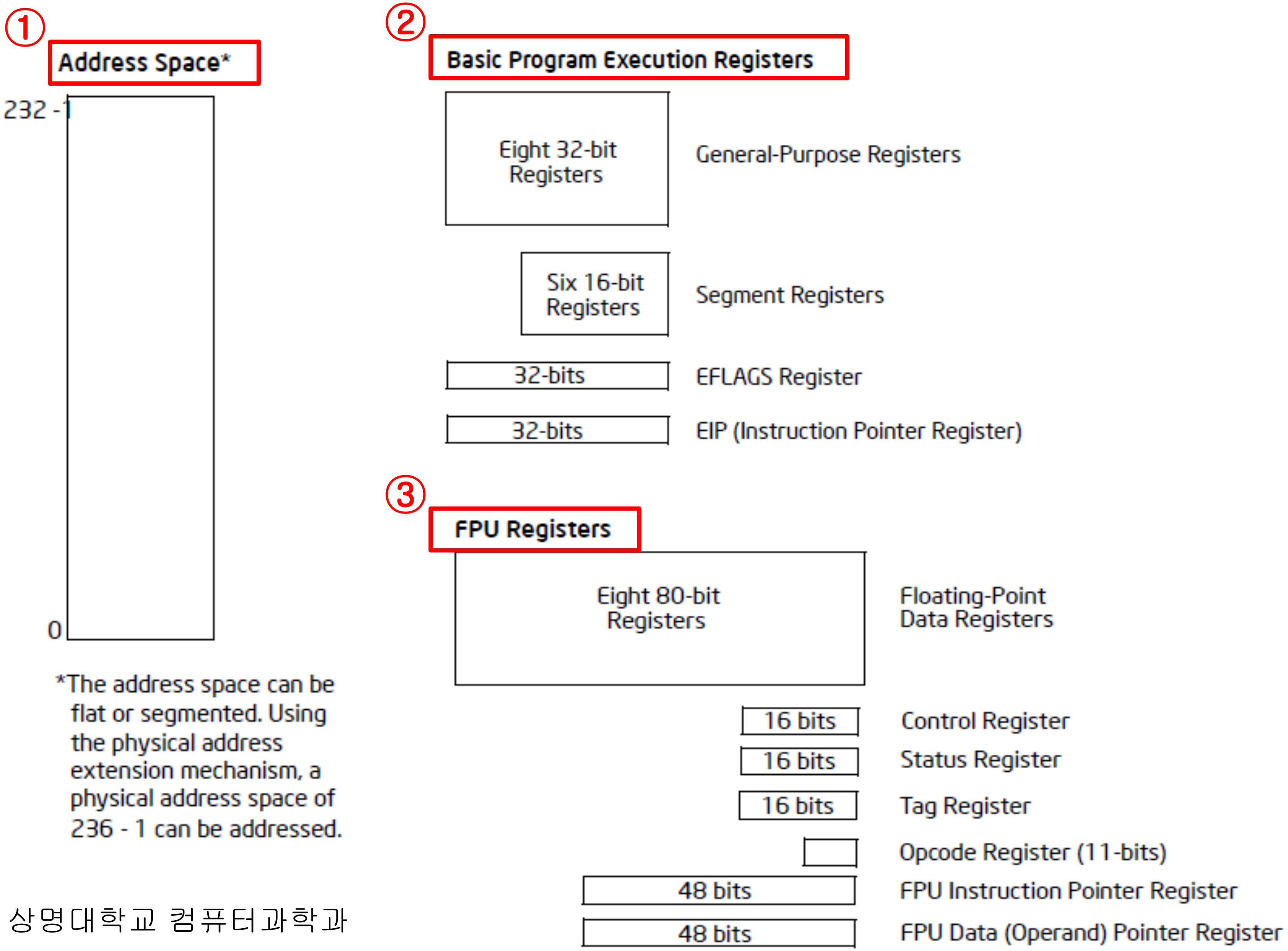
- The Third Generation Intel® Core™ Processor Family (2012)
- The Fourth Generation Intel® Core™ Processor Family (2013)

IA-32 Architecture

- **Computer architecture** is concerned with how the CPU acts and how it uses computer memory
- It includes *instruction set architecture, microarchitecture, system design* including all the other h/w components

- **IA-32 architecture** and **Intel 64 architecture** are the names of computer architecture starting from the Intel 8086 processor to the latest Intel Core 2 Duo and Intel Xeon processor 5100 series

IA-32 Basic Execution Environment



① Address Space*

232 -1

0

*The address space can be flat or segmented. Using the physical address extension mechanism, a physical address space of 236 - 1 can be addressed.

② Basic Program Execution Registers

Eight 32-bit Registers — General-Purpose Registers

Six 16-bit Registers — Segment Registers

32-bits — EFLAGS Register

32-bits — EIP (Instruction Pointer Register)

③ FPU Registers

Eight 80-bit Registers — Floating-Point Data Registers

16 bits — Control Register

16 bits — Status Register

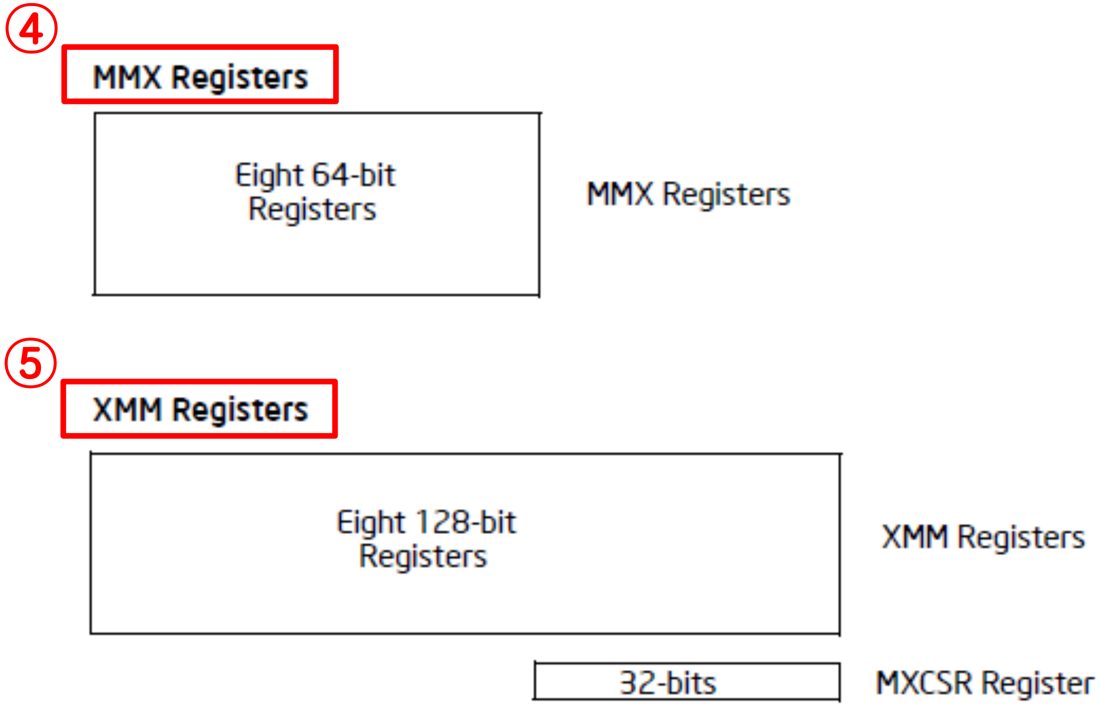
16 bits — Tag Register

Opcode Register (11-bits)

48 bits — FPU Instruction Pointer Register

48 bits — FPU Data (Operand) Pointer Register

IA-32 Basic Execution Environment



IA-32 Basic Execution Environment

- **Address space** — Any task or program running on an IA-32 processor can address a linear address space of up to 4 GB (2^{32} bytes) and a physical address space of up to 64 GB (2^{36} bytes).
- **Basic program execution registers** — The 8 general-purpose registers, the 6 segment registers, the *EFLAGS* register, and the *EIP* register comprise a basic execution environment to execute a set of general-purpose instructions. (integer arithmetic, program flow control, operate on bit and byte strings, and address memory)
- **Stack** — To support procedure or subroutine calls and the passing of parameters between procedures or subroutines, a stack and stack management resources are included in the execution environment

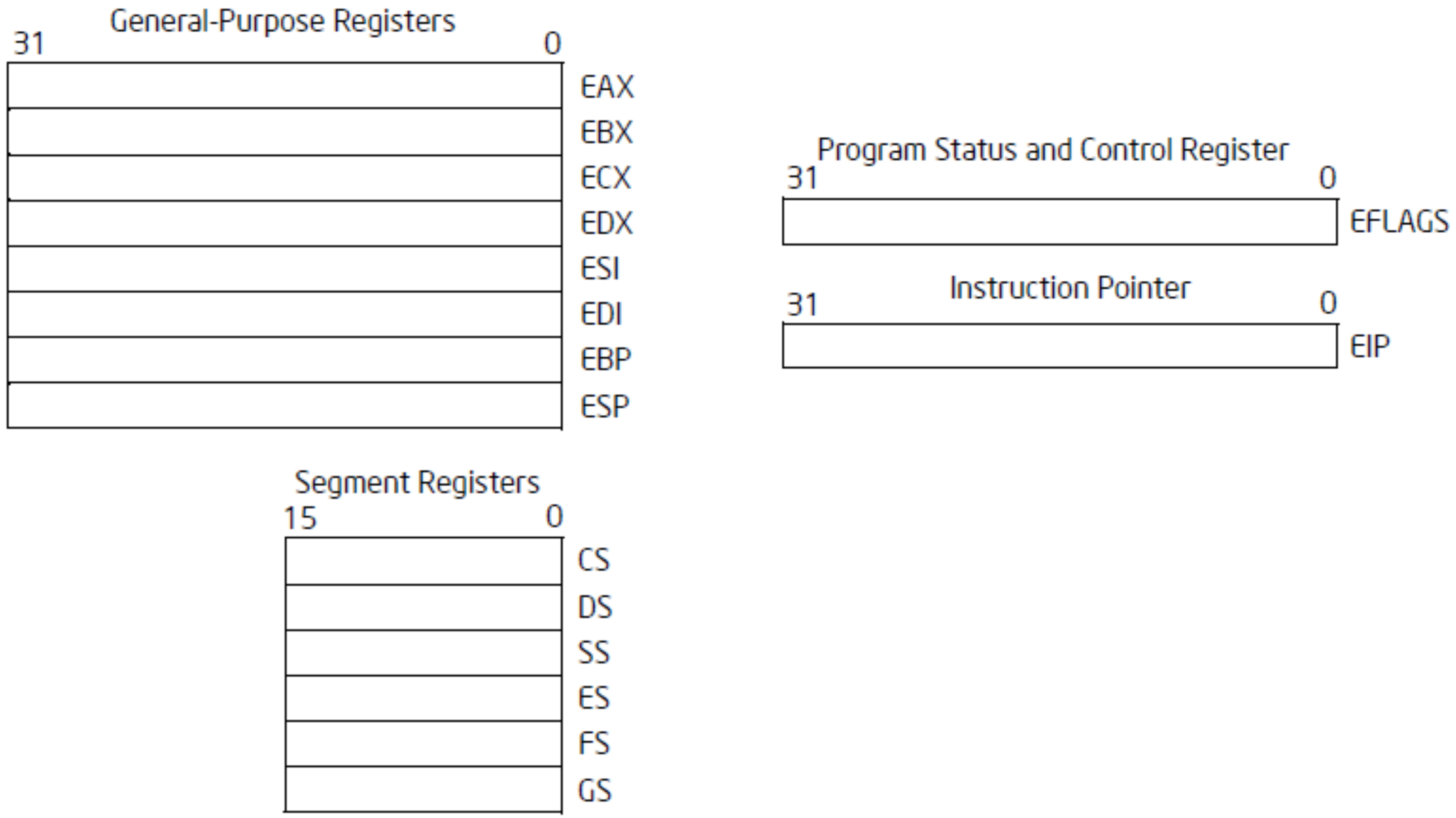
IA-32 Basic Execution Environment

- **x87 FPU registers** – The eight x87 FPU data registers, the x87 FPU control register, the status register, the x87 FPU instruction pointer register, the x87 FPU operand (data) pointer register, the x87 FPU tag register, and the x87 FPU opcode register for floating-point values
- **MMX registers** – The eight MMX registers support execution of single-instruction, multiple-data (SIMD) operations on 64-bit packed integers
- **XMM registers** – The eight XMM data registers and the MXCSR register support execution of SIMD operations on 128-bit packed integers.

Basic Program Execution Registers

- **General-purpose registers**. These eight registers are available for storing operands and pointers.
- **Segment registers**. These registers hold up to six segment selectors.
- ***EFLAGS* (program status and control) register**. The *EFLAGS* register report on the status of the program being executed and allows limited (application program level) control of the processor.
- ***EIP* (instruction pointer) register**. The *EIP* register contains a 32-bit pointer to the next instruction to be executed.

Basic Program Execution Registers



General Purpose Registers

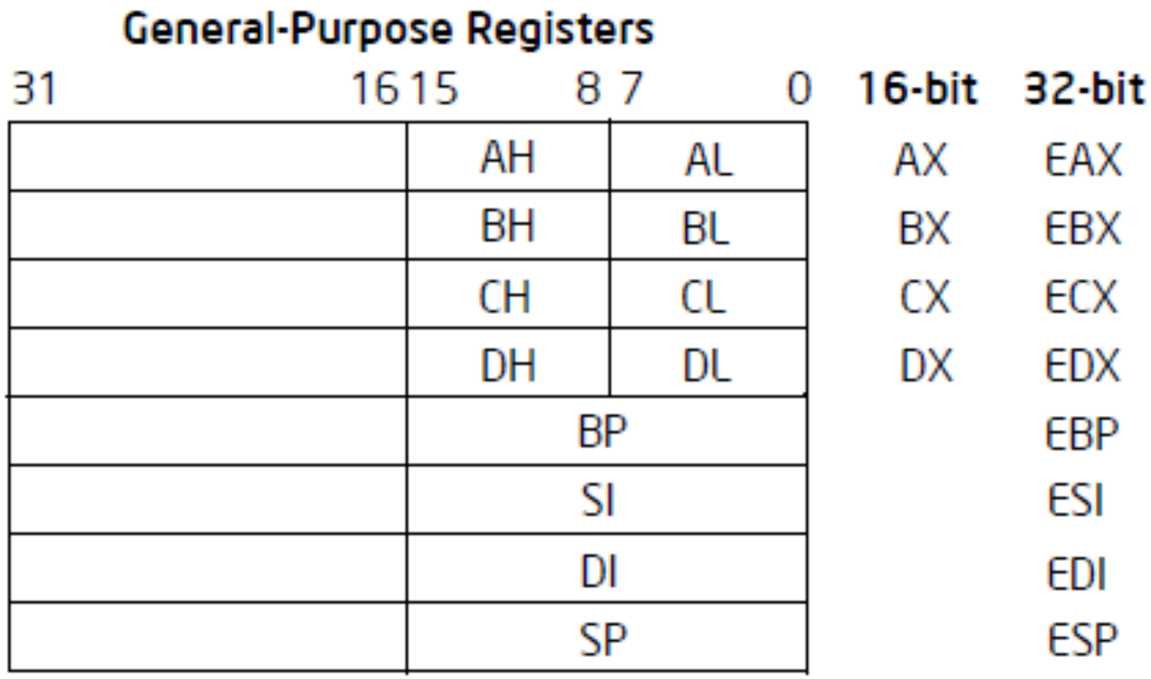
- The 32-bit general-purpose registers *EAX, EBX, ECX, EDX, ESI, EDI, EBP*, and *ESP* are provided for holding the following items:
  - Operands for logical and arithmetic operations
  - Operands for address calculations
  - Memory pointers
- The *ESP* register holds the stack pointer and as a general rule should not be used for another purpose.
- Many instructions assign specific registers to hold operands (eg. string instructions - *ECX, ESI,* and *EDI*)

- *EAX*: Accumulator: Referenced as *EAX, AX, AL* or *AH*
  - Used for *mult, div*, etc.
  - Used to hold an offset.

General Purpose Registers

- *EBX*: Base Index
  - Used to hold the offset of a data pointer.
  - Pointer to *DS* segment
- *ECX*: Count
  - Used to hold the count for some instructions, *REP* and *LOOP*.
  - Used to hold the offset of a data pointer.
- *EDX*: Data
  - Used to hold a portion of the result for *mult*, of the operand for *div*.
  - I/O pointer
- *ESI*: Source Index
- *EDI*: Destination Index
  - Holds the base source/destination pointer for string instructions.
- *ESP*: Stack pointer
- *EBP*: Base Pointer
  - Holds the base pointer for memory data transfers.
  - Pointer to data on the stack

Alternate General-Purpose Register Names



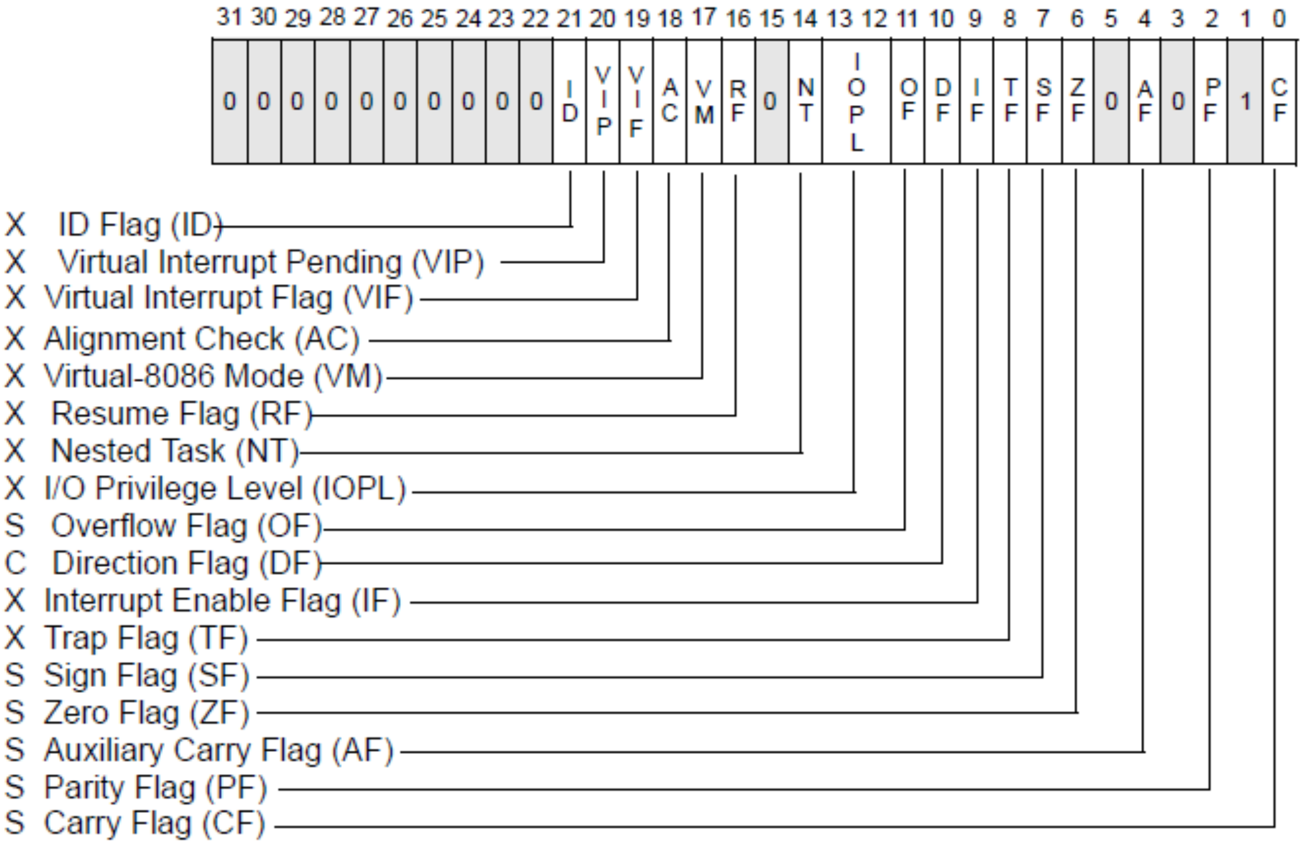
General-Purpose Registers

31 16	15 8	7 0	**16-bit**	**32-bit**
	AH	AL	AX	EAX
	BH	BL	BX	EBX
	CH	CL	CX	ECX
	DH	DL	DX	EDX
	BP			EBP
	SI			ESI
	DI			EDI
	SP			ESP

EFLAGS Register

- The 32-bit *EFLAGS* register contains a group of status flags, a control flag, and a group of system flags.
- The initial state of the *EFLAGS* register is 00000002H. Bits 1, 3, 5, 15, and 22 through 31 of this register are reserved.
- The following instructions can be used to move groups of flags to/from the procedure stack or the EAX register: *LAHF*, *SAHF*, *PUSHF*, *PUSHFD*, *POPF*, and *POPFD*
- When a call is made to an interrupt or exception handler procedure, the processor automatically saves the state of the *EFLAGS* registers on the procedure stack

EFLAGS Register



31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	ID	VIP	VIF	AC	VM	RF	0	NT	IOPL		OF	DF	IF	TF	SF	ZF	0	AF	0	PF	1	CF

X ID Flag (ID)
X Virtual Interrupt Pending (VIP)
X Virtual Interrupt Flag (VIF)
X Alignment Check (AC)
X Virtual-8086 Mode (VM)
X Resume Flag (RF)
X Nested Task (NT)
X I/O Privilege Level (IOPL)
S Overflow Flag (OF)
C Direction Flag (DF)
X Interrupt Enable Flag (IF)
X Trap Flag (TF)
S Sign Flag (SF)
S Zero Flag (ZF)
S Auxiliary Carry Flag (AF)
S Parity Flag (PF)
S Carry Flag (CF)

S Indicates a Status Flag
C Indicates a Control Flag
X Indicates a System Flag

▨ Reserved bit positions. DO NOT USE.
Always set to values previously read.

Status flags

- The status flags (bits 0, 2, 4, 6, 7, and 11) of the *EFLAGS* register indicate the results of arithmetic instructions, such as the *ADD*, *SUB*, *MUL*, and *DIV* instructions.

- ***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.
- ***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.)

Status flags

- ***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.

- Of these status flags, only the *CF* flag can be modified directly, using the *STC*, *CLC*, and *CMC* instructions. Also the bit instructions (*BT*, *BTS*, *BTR*, and *BTC*) copy a specified bit into the *CF* flag.
- The status flags allow a single arithmetic operation to produce results for three different data types: unsigned integers, signed integers, and BCD integers.
- The condition instructions *Jcc* (jump on condition code *cc*), *SETcc* (byte set on condition code *cc*), *LOOPcc*, and *CMOVcc* (conditional move) use one or more of the status flags as condition codes and test them for branch, set-byte, or end-loop conditions.

Direction flag

- The direction flag (*DF*, located in bit 10 of the *EFLAGS*) controls string instructions (*MOVS*, *CMPS*, *SCAS*, *LODS*, and *STOS*).
- Setting the **DF** flag causes the string instructions to auto-decrement (to process strings from high addresses to low addresses).
- Clearing the *DF* flag causes the string instructions to auto-increment (process strings from low addresses to high addresses).
- The *STD* and *CLD* instructions set and clear the *DF* flag, respectively.

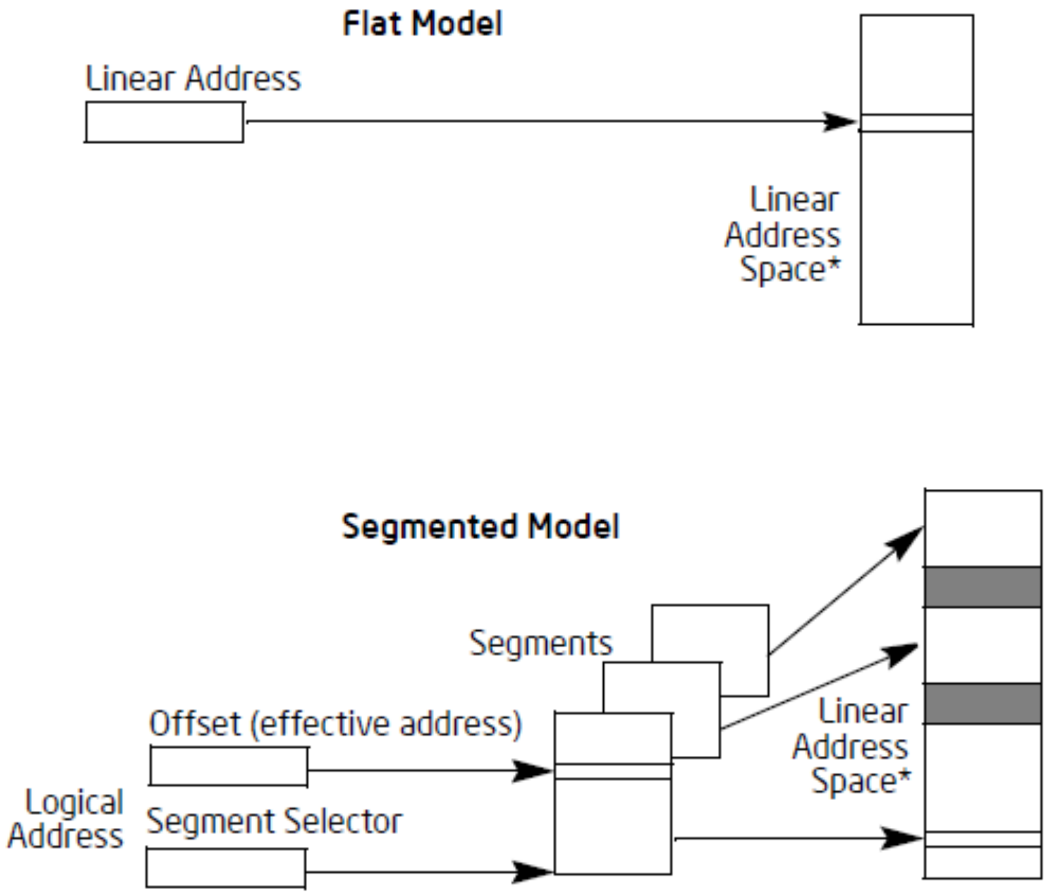
Instruction Pointer Register

- The instruction pointer (*EIP*) register contains the offset in the current code segment for the next instruction to be executed.
- It is advanced from one instruction boundary to the next in straight-line code or it is moved ahead or backwards by a number of instructions when executing *JMP*, J*cc*, *CALL*, *RET*, and *IRET* instructions.
- The *EIP* register cannot be accessed directly by software; it is controlled implicitly by control-transfer instructions (such as *JMP*, J*cc*, *CALL*, and *RET*), interrupts, and exceptions. The only way to read the *EIP* register is to execute a *CALL* instruction and then read the value of the return instruction pointer from the procedure stack.
- The *EIP* register can be loaded indirectly by modifying the value of a return instruction pointer on the procedure stack and executing a return instruction

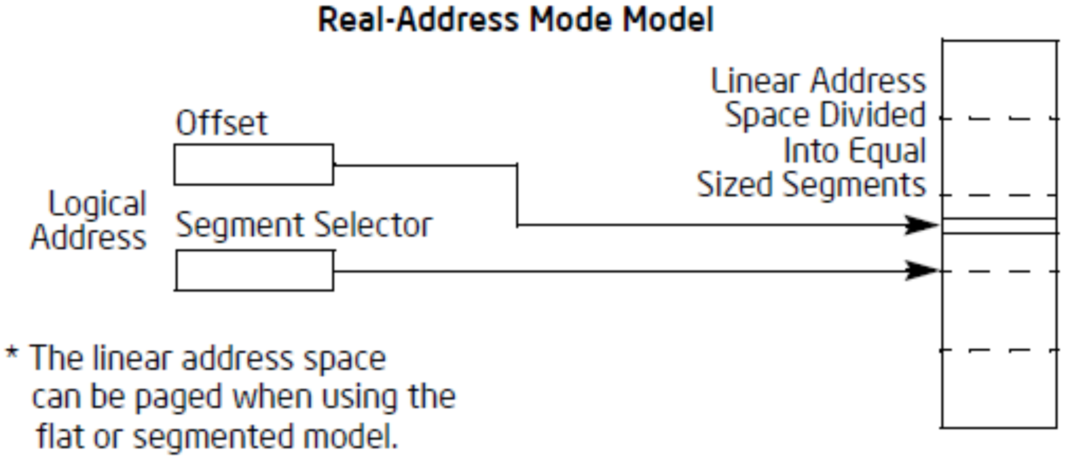
Modes of Operation

- The IA-32 architecture supports three basic operating modes. The operating mode determines which instructions and architectural features are accessible
  - Protected mode
  - Real-address mode
  - System management mode
- IA-32 memory models
  - Flat memory model
  - Segmented memory model
  - Real-address mode memory model

IA-32 Memory Models



IA-32 Memory Models


Real-Address Mode Model
Linear Address Space Divided Into Equal Sized Segments
Offset
Logical Address
Segment Selector
* The linear address space can be paged when using the flat or segmented model.

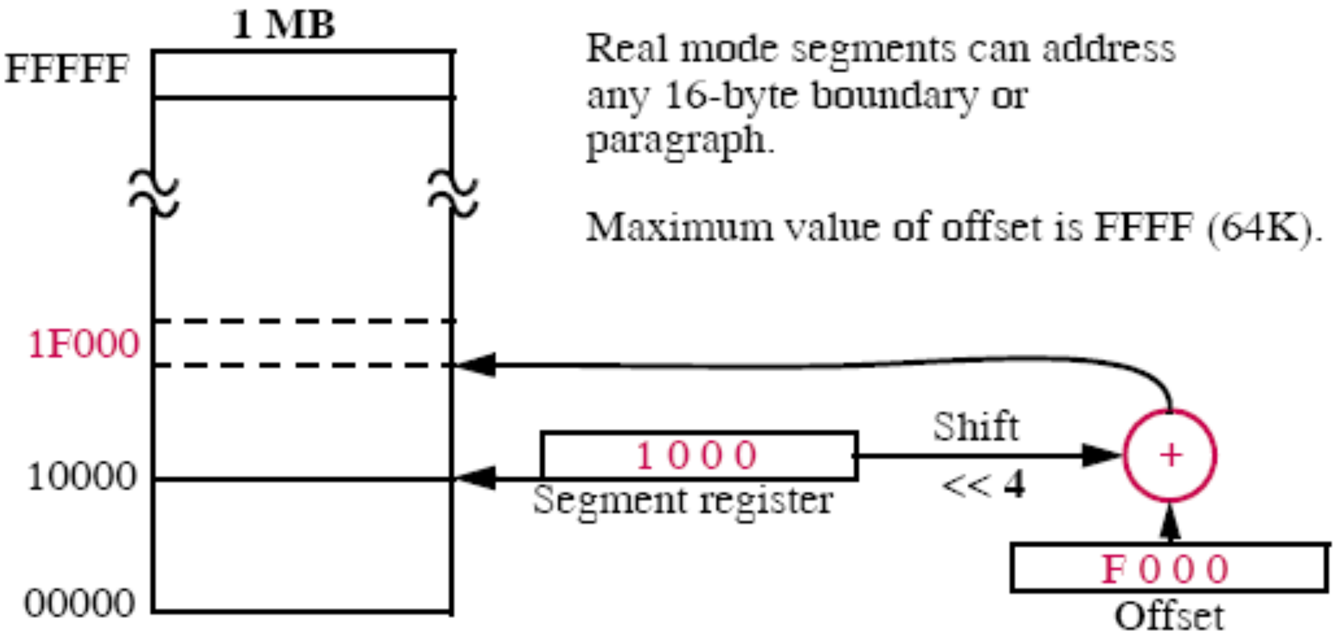
Segment Registers

- The segment registers, *CS*, *DS*, *SS*, *ES*, *FS*, and *GS* hold 16-bit segment selectors (A segment selector is a special pointer that identifies a segment in memory).

- *CS* (Code Segment):
  - In real mode, this specifies the start of a 64KB memory segment.
  - In protected mode, it selects a descriptor.
- *DS* (Data Segment):
  - Similar to the *CS* except this segment holds data.
- *ES* (Extra Segment):
  - Data segment used by some string instructions to hold destination data.
- *SS* (Stack Segment):
  - Similar to the *CS* except this segment holds the stack.
  - *ESP* and *EBP* hold offsets into this segment.
- *FS* and *GS*
  - Allows two additional memory segments to be defined.

Real Mode Memory Addressing

- Only mode available to the 8086 and 8088.
  - Allow the processor to address only the first 1MB of memory.
  - DOS requires real mode.
- *Segments and Offsets:*
  - Effective address = Segment address + an offset.



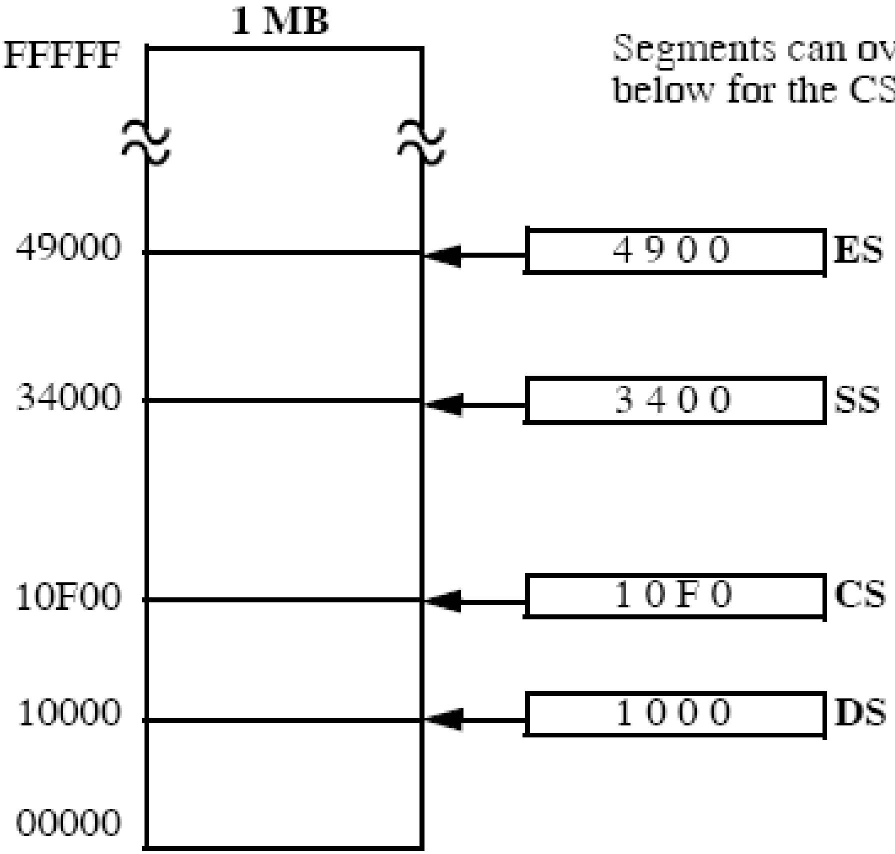
Real mode segments can address any 16-byte boundary or paragraph.

Maximum value of offset is FFFF (64K).

Real Mode Memory Addressing

- ***Segments and Offsets***:
  - Syntax is usually given as *seg_addr:offset*, e.g. *1000:F000* in the previous example to specify *1F000H*.
  - Implicit combinations of segment registers and offsets are defined for memory references.
  - For example, the code segment (*CS*) is always used with the instruction pointer (IP for real mode or EIP for protected mode).
- Examples
  - ***CS:IP***
  - ***SS:SP, SS:BP***
  - ***DS:AX, DS:BX, DS:CX, DS:DX, DS:DI, DS:SI, DS:8-bit_literal, DS:16-bit_literal***
  - ***ES:DI***
  - ***FS*** and ***GS*** have no default.

Real Mode Memory Addressing



Segments can overlap, as shown below for the CS and DS.