Basic Architecture

Outline:

- Internal *programmer visible* architecture, e.g. registers

- Real Mode Addressing:

  Real Mode Memory: 00000H-FFFFFH (the first 1MB of main memory)

- Protected Mode Addressing:

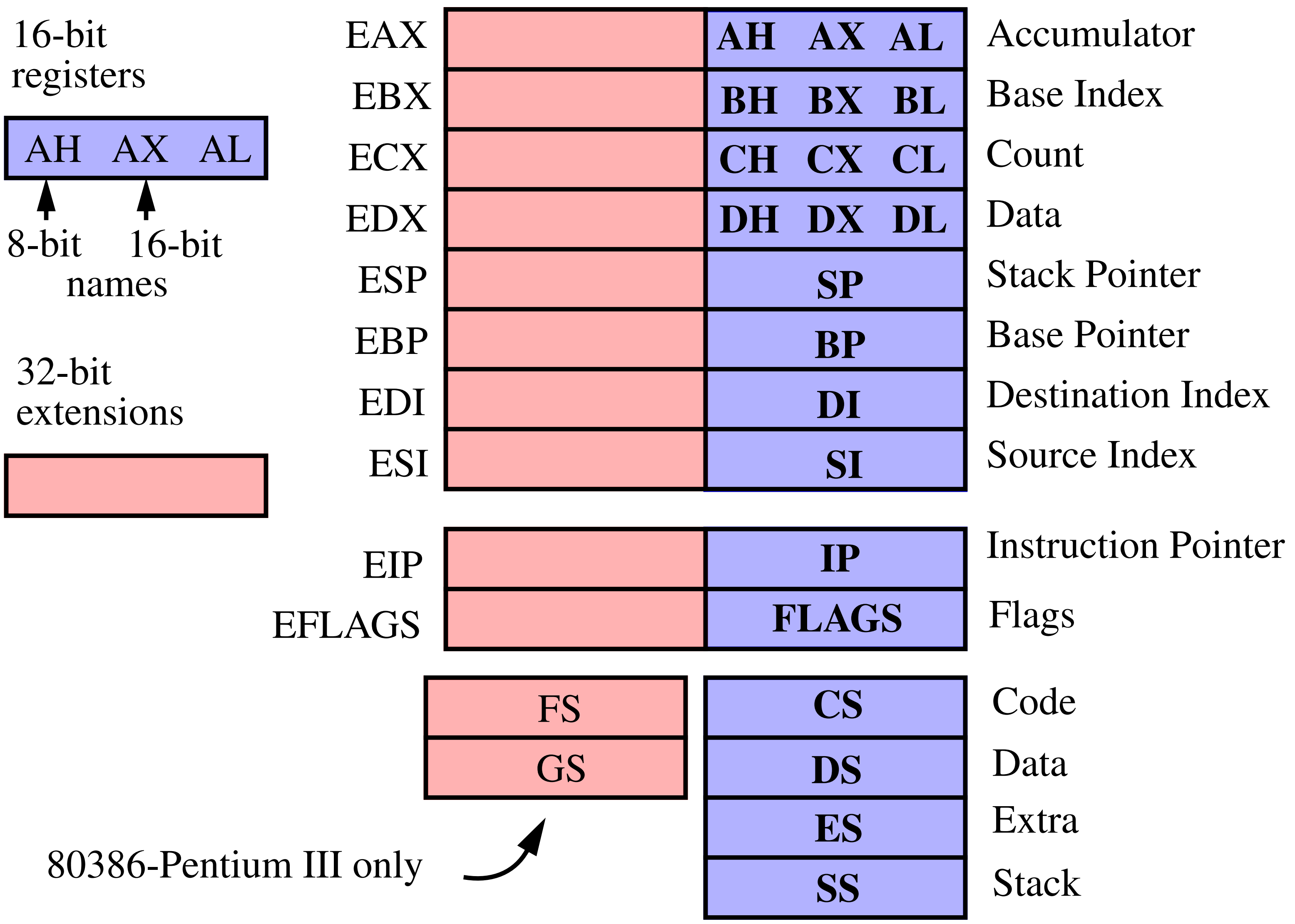
  All of memory (applicable to 80286 and later processors)

  Programmer **in***visible* registers to control and operate the protected memory system

- 80x86 Memory Paging

Programmer Visible Architecture

Programmer visible registers:



Programmer Visible Architecture

General Purpose Registers: The main functions are listed

- **EAX: Accumulator**: Referenced as EAX, AX, AL or AH
  - Used for mult, div, etc
  - Used to hold an offset
- **EBX: Base Index**:
  - Used to hold the offset of a data pointer
- **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
  - Used to hold the offset of a data pointer
- **EBP: Base Pointer**:
  - Holds the base pointer for memory data transfers
- **EDI: Destination Index**:
  - Holds the base destination pointer for string instructions
- **ESI: Source Index**:
  - Holds the base source pointer for string instructions

Programmer Visible Architecture

Special Purpose Registers:

- **EIP: Instruction Pointer**:

  Points to the next instruction in a code segment

  16-bits (IP) in real mode and 32-bits in protected mode
- **ESP: Stack Pointer**:

  Used by the stack, call and return instructions
- **EFLAGS**:

  Store the state of various conditions in the microprocessor

Programmer Visible Architecture

Special Purpose Registers:

EFLAGS Register:

31	21	20	19	18	17	16		14	13	12	11	10	9	8	7	6		4		2		0
	ID	VIP	VIF	AC	VM	RF		NT	IOP 1	IOP 0	O	D	I	T	S	Z		A		P		C

The rightmost 5 flag bits and overflow change after many of the arithmetic and logic instructions execute. Data transfer and control instructions never change the flags.

- **C (Carry)**:

  Holds the carry out after addition or the borrow after subtraction

  Also indicates error conditions

- **P (Parity)**:

  0 for odd number of bits and 1 for even

  Obsolete feature of the 80x86

- **A (Auxiliary Carry)**:

  Highly specialized flag used by DAA and DAS instructions after BCD addition or subtraction

Programmer Visible Architecture

Special Purpose Registers:

- **EFLAGS** (cont).
  - **Z (Zero)**:
    1 if the result of an arithmetic or logic instruction is 0
  - **S (Sign)**:
    1 if the sign of the result of an arith. or logic instruction is negative
  - **T (Trap)**:
    Trap enable. The microprocessor interrupts the flow of instructions on conditions indicated by the debug and control registers
  - **I (Interrupt)**:
    Controls the operation of the INTR (Interrupt request) pin. If 1, interrupts are enabled. Set by *STI* and *CLI* instructions.
  - **D (Direction)**:
    Selects with increment or decrement mode for the DI and/or SI registers during string instructions. If 1, registers are automatically decremented. Set by *STD* and *CLD* instructions.
  - **O (Overflow)**:
    Set for addition and subtraction instructions

Programmer Visible Architecture

Special Purpose Registers:

- **EFLAGS** (cont).

80286 and up:

- **IOPL (I/O privilege level)**:

  It holds the privilege level at which your code must be running in order to execute any I/O-related instructions. 00 is the highest.

- **NT (Nested Task)**:

  Set when one system task has invoked another through a CALL instruction in protected mode

80386 and up:

- **RF (Resume)**:

  Used with debugging to selectively mask some exceptions

- **VM (Virtual Mode)**:

  When 0, the CPU can operate in Protected mode, Virtual 8086 mode or Real mode. When set, the CPU is converted to a high speed 8086. This bit has enormous impact

Programmer Visible Architecture

Special Purpose Registers:

- **EFLAGS** (cont).

80486SX and up:

- **AC (Alignment Check)**:

  Specialized instruction for the 80486SX

Pentium and up:

- **VIF (Virtual Interrupt Flag)**:

  Copy of the interrupt flag bit
- **VIP (Virtual Interrupt Pending)**:

  Provides information about a virtual mode interrupt
- **ID (Identification)**:

  Supports the CPUID instruction, which provides version number and manufacturer information about the microprocessor

Programmer Visible Architecture

Segment Registers:

- **CS (Code Segment)**:
  - In real mode, this specifies the start of a 64KB memory segment
  - In protected mode, it selects a descriptor
  - The code segment is limited to 64KB in the 8086-80286 and 4 GB in the 386 and above
- **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**: 80386 and up.
  - Allows two additional memory segments to be defined

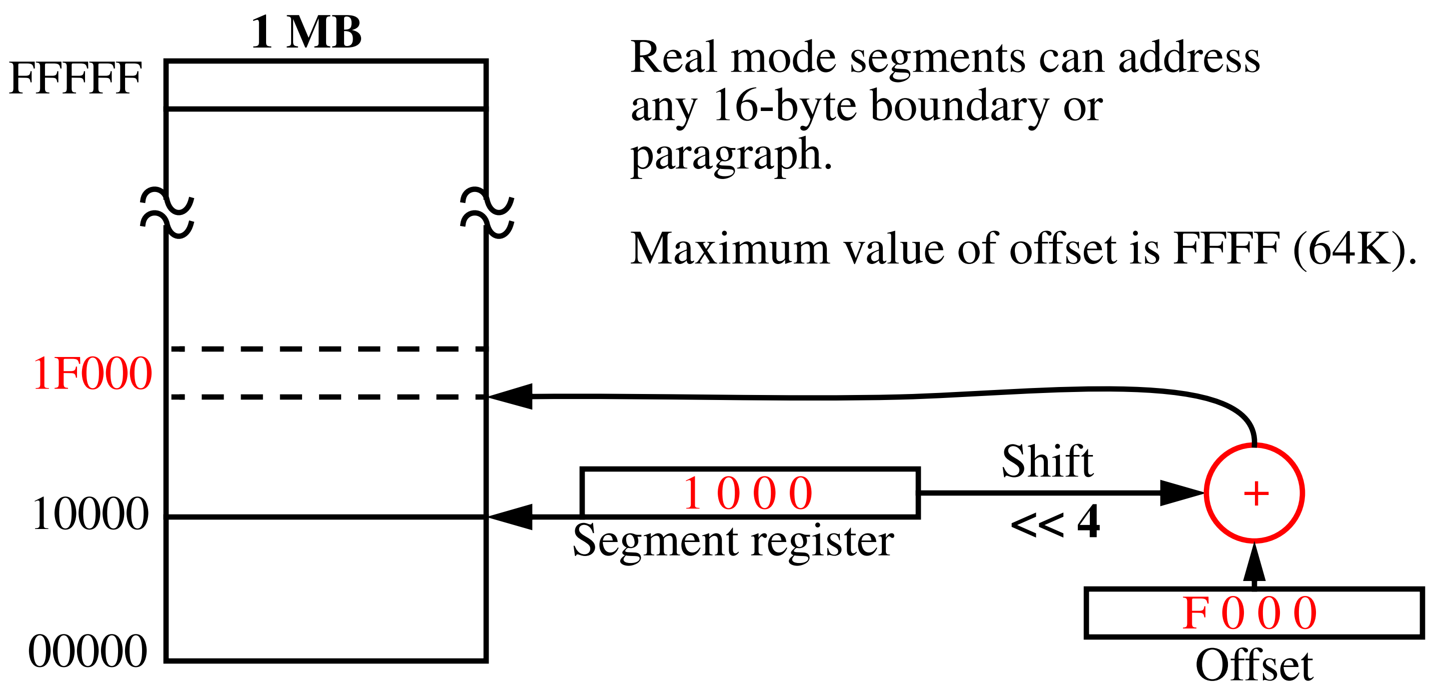
Real Mode Memory Addressing

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)

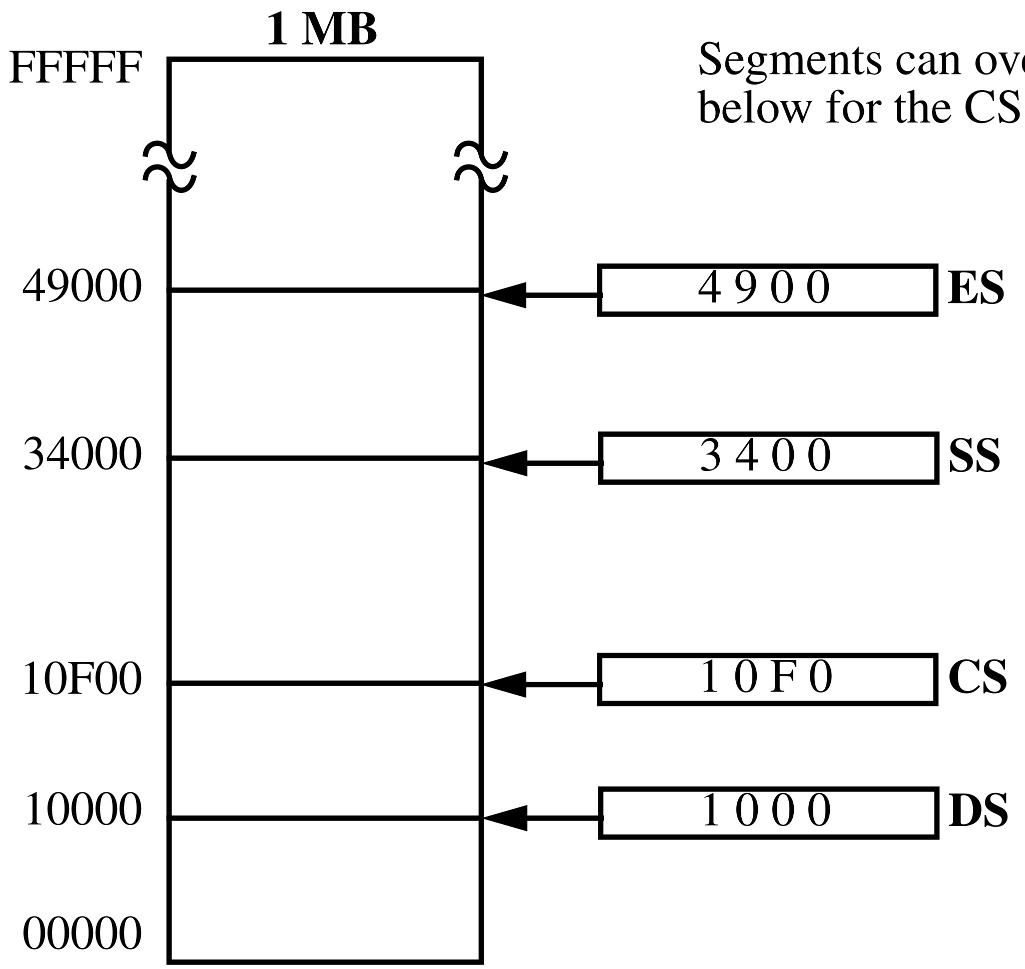
- **CS:EIP**
- **SS:ESP**, **SS:EBP**
- **DS:EAX**, **DS:EBX**, **DS:ECX**, **DS:EDX**, **DS:EDI**, **DS:ESI**, **DS:8-bit_literal**, **DS:32-bit_literal**
- **ES:EDI**
- **FS** and **GS** have no default.

It is illegal to place an offset larger than FFFF into the 80386 32-bit registers operating in Real Mode

Real Mode Memory Addressing

Segments and Offsets:

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



Segmented addressing allows relocation of data and code

OS can assign the segment addresses at run time