Microprocessors Architecture

Microprocessor Architecture: Registers

- ***Basic Architecture***
- ***Outline***

  Before a program is written the internal configuration of the µp must be known.

  Program-visible architecture of the 8086 through P4 will be detailed. Program visible architecture, e.g. registers *.*

  The adressing mode of this family of µps are described for both the real and protected modes.

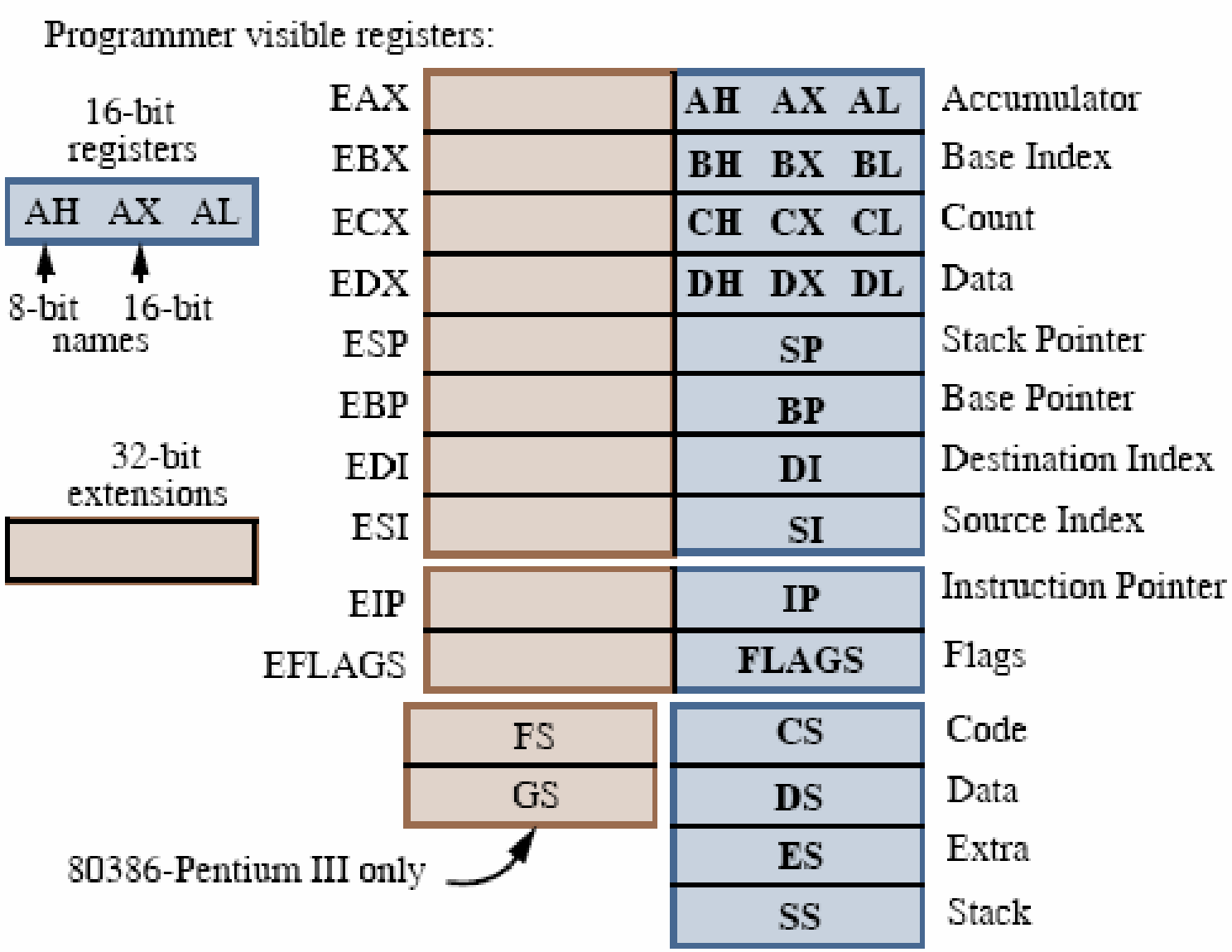
  - Real Mode Addressing:

    Real Mode Memory: 00000H-FFFFFH (the first 1MB of main memory).
  - Protected Mode Addressing: *(covered later)*

    All of memory location (applicable to 80286 and later processors).

    Programmer ***in****visible* registers to control and operate the protected memory system.
  - 80x86 Memory Paging (covered later).

- The programming model of 8086 through the P4 : program visible since its registers are used during application programming and specified by the instructions.
- Some registers can not!
  - Program invisible , 80286 above
  - Used to control and operate the protected memory system
- 8086-8088-80286 contain 16-bit internal architectures
- The 80386-P4 contain 32 bit internal architecture
- Some registers are general purpose regs., some registers are special purpose regs.


Programmer visible registers:
16-bit registers
AH AX AL
8-bit names
16-bit names
32-bit extensions
EAX
EBX
ECX
EDX
ESP
EBP
EDI
ESI
EIP
EFLAGS
AH AX AL
BH BX BL
CH CX CL
DH DX DL
SP
BP
DI
SI
IP
FLAGS
Accumulator
Base Index
Count
Data
Stack Pointer
Base Pointer
Destination Index
Source Index
Instruction Pointer
Flags
FS
GS
80386-Pentium III only
CS
DS
ES
SS
Code
Data
Extra
Stack

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 adress of a location in the memory system.

- ***EBX: Base Index:***

Used to adress memory data.
(for BX) Used to hold an offset adress of a location in the memory system.

- ***ECX: Count:***

Used to hold the count for some instructions, shift, rotate, LOOP.
Used to adress memory data.

- ***EDX: Data:***

Used to hold a portion of the result for mult, of the operand for div.
Used to adress memory data.

- ***EBP: Base Pointer:***

Points to memory location for memory data transfers.

- ***EDI: Destination Index:***

Addresses destination string data for string instructions.

- ***ESI: Source Index:***

Addresses source string data for string instructions.

Special Purpose Registers:EIP,ESP,EFLAGS,and segment regs. CS;DS;ES;SS;FS;GS

- ***EIP: Instruction Pointer:***

  Points to the next instruction in a code segment.

  16-bits (IP) in real mode and 32-bits (EIP) in protected mode.

- ***ESP: Stack Pointer:***

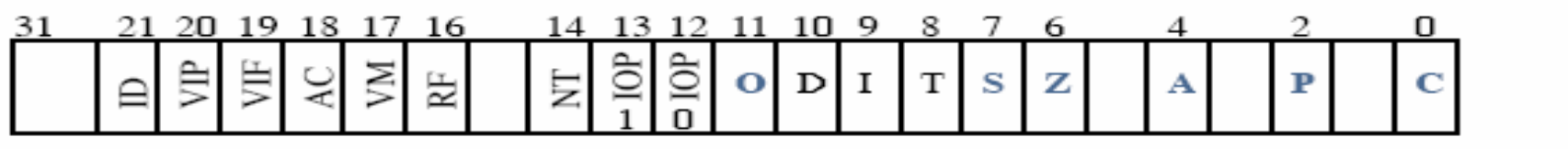
  Used by the stack. Stack stores data through this pointer.(covered later)

- ***EFLAGS:***

  Store conditions of the microprocessor and control its operation.

Special Purpose Registers:

EFLAGS Register:



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 parity and 1 for even.(1110:odd parity ; 1111000:even parity)

- ***A (Auxiliary Carry):***

Holds the half carry after addition or the borrow after subtraction between 3. bit and 4. bit of the result.

- ***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) input pin. If 1, interrupts are

enabled. Set by *STI (set I flag)* and *CLI (clear I flag)* instructions.

- ***D (Direction):***

Selects with increment or decrement mode for the DI(destination index) and/or SI registers during string instructions. If 1, registers are automatically decremented. Set by *STD(set direction)* and *CLD (clear direction)* instructions.

- ***O (Overflow):***

Set for addition and subtraction instructions.Overflow occurs when signed numbers are added or subtracted.

80286 and up:

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

It holds the privilege level for I/O devices. Used in the protected mode operation. 00 is the highest.

- ***NT (Nested Task):***

indicates that the current task is nested within another task in protected Mode operation. This flag is set when the task is nested by software.

80386 and up:

- ***RF (Resume)****:*

Used with debugging to control resumption of execution after the next instr.

- ***VM (Virtual Mode)****:*

The VM flag selects virtual mode operation in a protected mode operation. This allows multiple DOS memory partitions to coexist in the memory system.

80486SX and up:

- ***AC (Alignment Check)****:*

Specialized instruction for the 80486SX. If a word or doubleword is adressed on a non word or non double word boundary this bit is activated.

Pentium and up:

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

Copy of the interrupt flag bit.

- ***VIP (Virtual Interrupt Pending):***

Provides information about a virtual mode interrupt. This is used in multitasking environments to provide the system with VIF and interrupt pending information.

- ***ID (Identification):***

provides the system with information about the microprocessor such as version number and manufacturer information.

Segment Registers:

- ***CS (Code Segment):***

The code segment is a section of memory that holds code. The code segment reg. defines starting adress of the code segment. In real mode, this specifies the start of a 64KB memory segment.

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)****:*

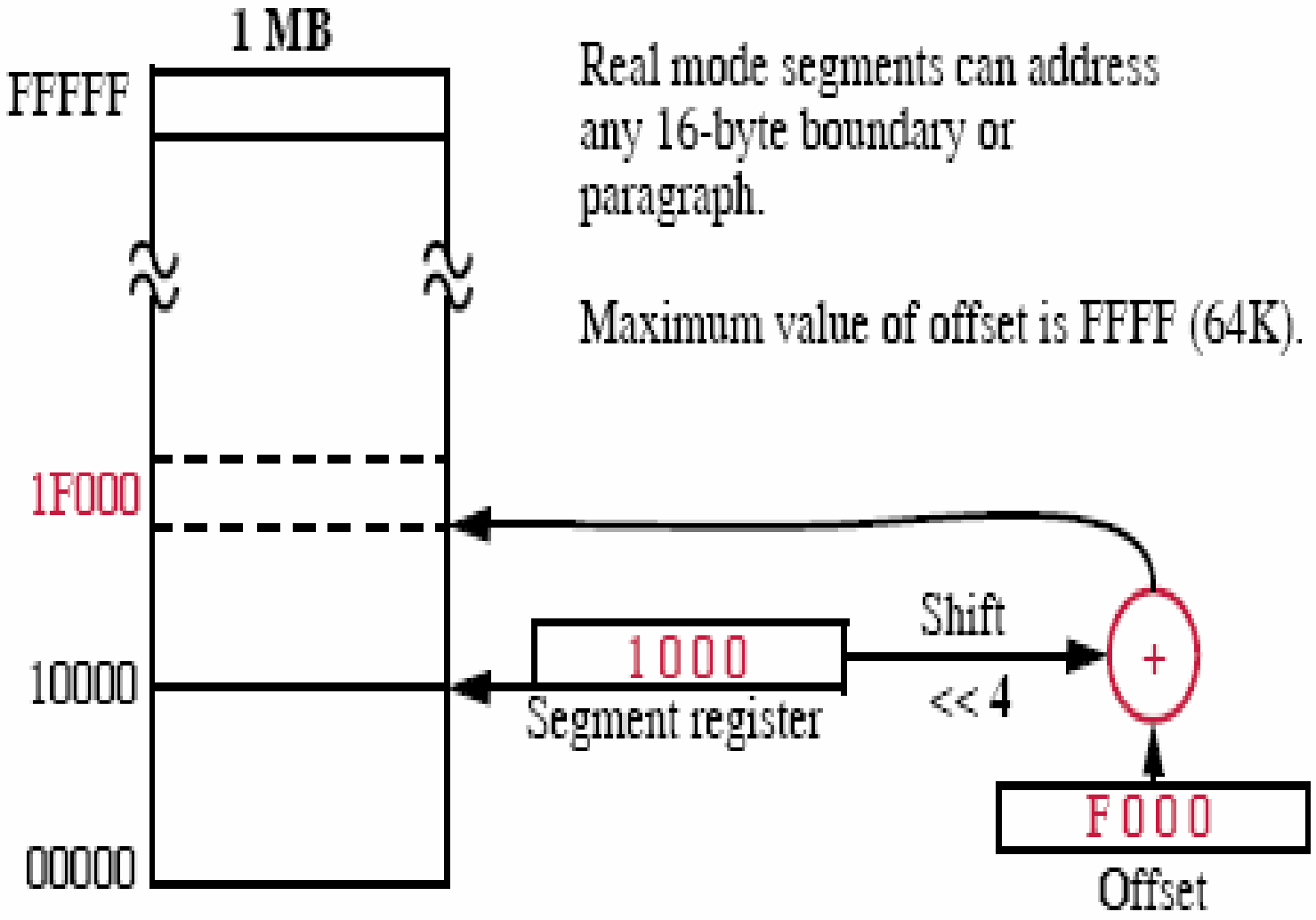
Similar to the CS except this segment holds the stack.

ESP and EBP hold offsets into this segment.

- ***FS and GS****:* 80386 and up.

Allows two additional memory segments to be defined.

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


1 MB
FFFFF
1F000
10000
00000
Real mode segments can address any 16-byte boundary or paragraph.
Maximum value of offset is FFFF (64K).
1000
Segment register
Shift
<< 4
+
F000
Offset

***Segments and Offsets**:*

- Syntax is usually given as *seg_addr:offset*, e.g. *1000:F000* in the previous example to specify *1F000H*.
- The microp.has some rules that apply to segments whenever memory is addressed:
- 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: if cs=1400h and IP=1200H next instr in the location 14000h+1200h=15200h

 SS:ESP, SS:EBP: stack data referenced through ESP and EBP

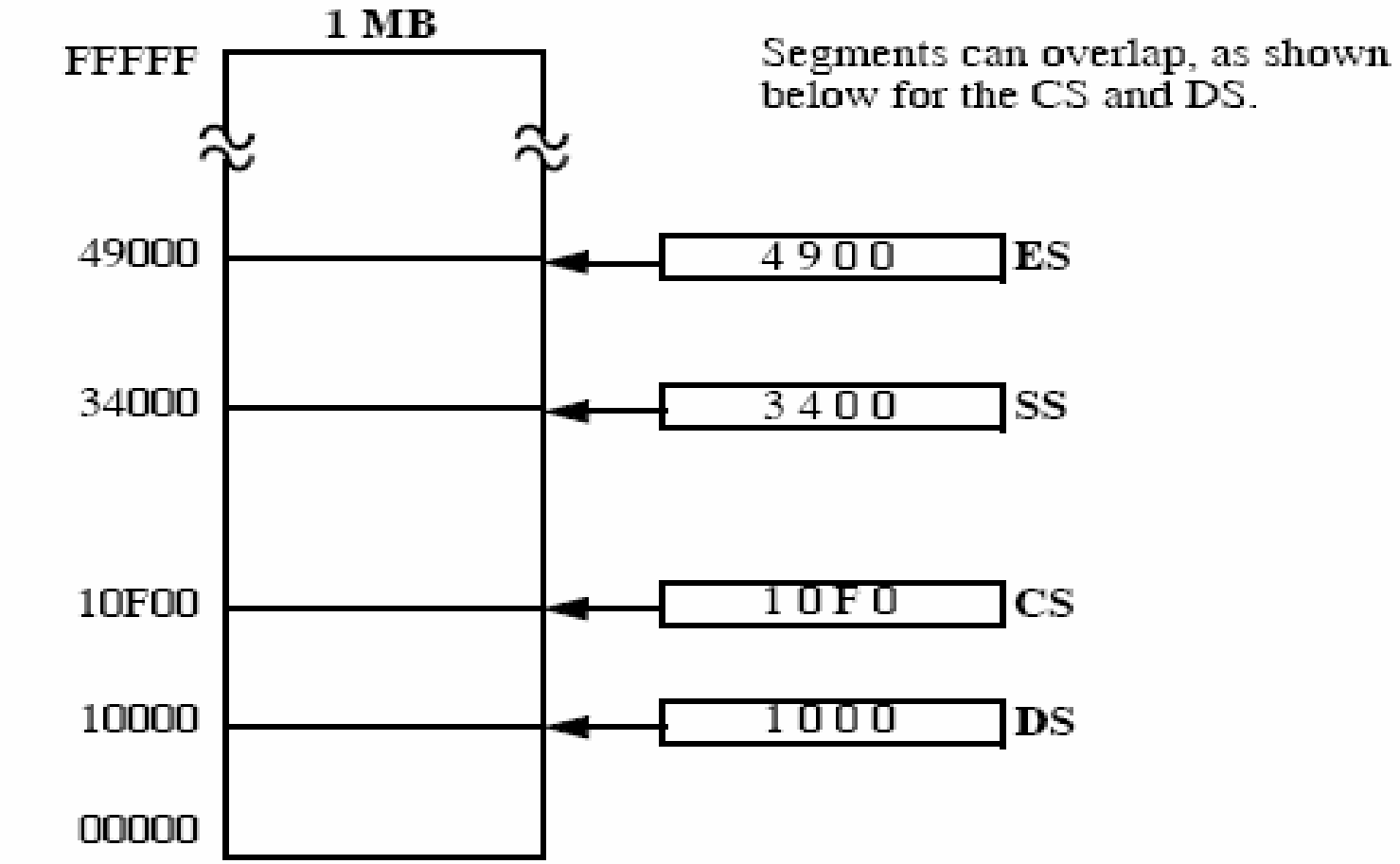
İf ss=2000h and bp=3000h then the memory location addressed 23000h.

 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.



Segmented addressing allows relocation of data and code.
OS can assign the segment addresses at run time.