

# CMPE 310 Systems Design and Programming

## L16: Chapter 2 – The Microprocessor and Its Architecture

UMBC

AN HONORS UNIVERSITY IN MARYLAND

## L16 Objectives

- \* Describe the general aspects of a processor's "programming model"
- \* Describe the registers of the x386DX
- \* State the purpose of the code segment, data segment, stack segment, and extra segment
- \* Explain address generation

## The Programming Model

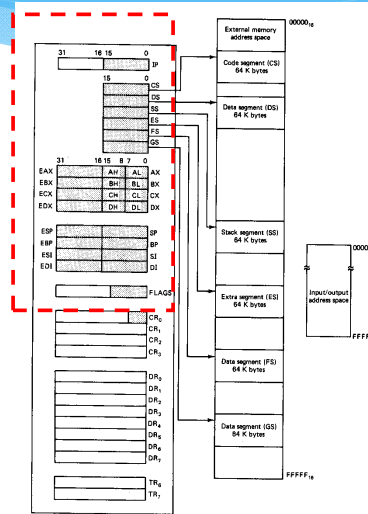
- \* Programmer's understanding the operation of the microcomputer from a program point of view
- \* Elements of the programming model
  - \* Register set
  - \* Memory address space
  - \* Input/output address space
  - \* Operations the processor can perform
- \* What the programmer must know about the microprocessor
  - \* Registers available within the device
    - \* Purpose, operating capabilities of each
  - \* Size, organization of memory and I/O address spaces
  - \* Types of data

## Intel Register Architecture

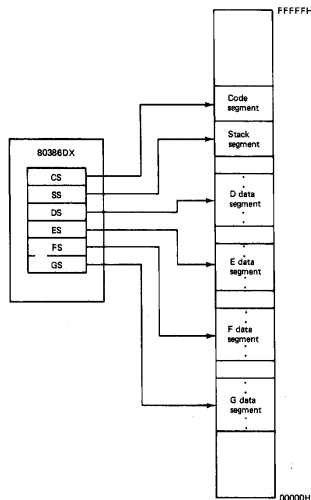
### Basic Architecture

- \* Internal **programmer visible** architecture
  - \* Registers are used during programming applications
- \* 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 invisible** registers to control and operate the protected memory system (not directly addressable)
- \* 80x86 Memory Paging.

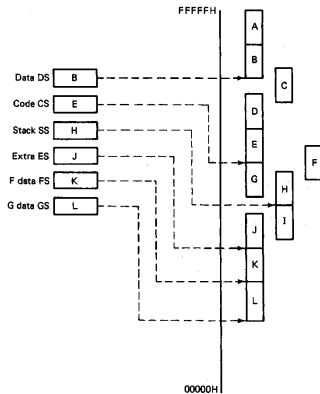
- \* Eight 32-bit registers
  - \* (4) Data registers- EAX, EBX, ECX, EDX, can be used as 32, 16 or 8bit
  - \* (2) Pointer registers- EBP, ESP
  - \* (2) Index registers- ESI, EDI
- \* Seven 16-bit registers
  - \* (1) Instruction pointer- IP
  - \* (6) Segment registers- CS, DS, SS, ES, FS, GS
- \* Flags (status) register-EFLAGS
- \* Control register- CRO



- \* Memory segmentation
  - Only subset of 8086 real-mode address space active
  - Each segment register points to lowest address of 64KB contiguous segment
  - Total active memory: 384 KB
    - 64 KB code segment (CS)
    - 64 KB stack segment (SS)
    - 256 KB over 4 data segments (DS, ES, FS, GS)
  - **Why are all the segment 64KB?**

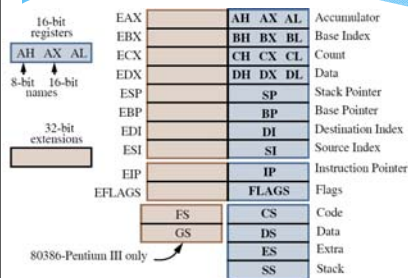


## User access, Restrictions, and Orientation



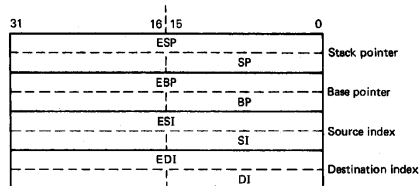
- Segment registers are user accessible
- Programmer can change values under software control
- Permits access to other parts of memory
- Segments must start on 16-byte boundary
  - Examples: 00000H, 00010H, 00020H
- Orientation of segments:
  - Contiguous—A&B or D,E&G
  - Disjointed—C&F
  - Overlapping—B&C, E&F, or F,G,&H

## General Purpose Data Registers



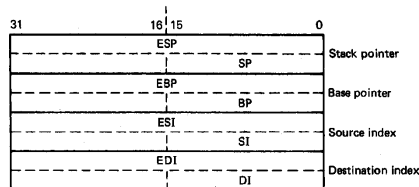
- \* Four general purpose data registers
  - \* Accumulator (A) register
  - \* Base (B) register
  - \* Count (C) register
  - \* Data (D) register
- \* Can hold 8-bit, 16-bit, or 32-bit data
  - \* AH/AL = high and low byte value
  - \* AX = word value
  - \* EAX = double word value
- \* General uses:
  - \* Hold data such as source or destination operands for most operations—ADD, AND, SHL
  - \* Hold address pointers for accessing memory
- \* Some also have dedicated special uses
  - \* C—count for loop,
  - \* B—table look-up translations, base address
  - \* D—indirect I/O and string I/O

## Pointer Registers



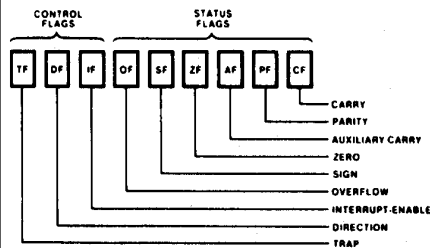
- \* Two pointer registers
  - \* Stack pointer register
    - \* ESP = 32-bit extended stack pointer
    - \* SP = 16-bit stack pointer
  - \* Base pointer register
    - \* EBP = 32-bit extended base pointer
    - \* BP = 16-bit base pointer
- \* Use to access information in stack segment of memory
  - \* SP/BP offsets from the current value of the stack segment base address
  - \* Select a specific storage location in the current 64K-byte stack segment
    - \* SS:SP—points to top of stack (TOS)
    - \* SS:BP—points to data in stack

## Index Registers



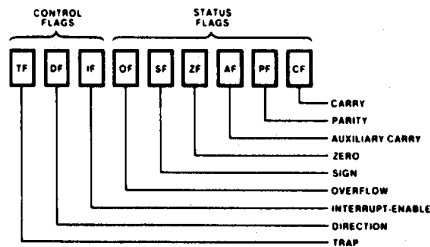
- \* Two index registers
  - \* Source index register
    - \* ESI = 32-bit source index register
    - \* SI = 16-bit source index register
  - \* Destination index registers
    - \* EDI = 32-bit destination index register
    - \* DI = 16-bit destination index register
- \* Used to access source and destination operands in data segment of memory
  - \* DS:SI—points to source operand in data segment
  - \* DS:DI—points to destination operand in data segment
  - \* Also used to access information in the extra segment (ES)

## Flags Register



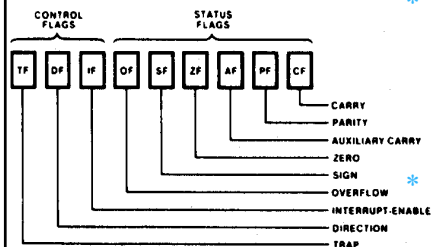
- \* 32-bit register holding single bit status and control information
- \* 9 active flags in real mode
- \* Two categories
  - \* **Status flags:** conditions resulting from instruction
    - \* Most instructions update status
    - \* Used as test conditions
  - \* **Control flags:** control processor functions
    - \* Used by software to turn on/off operating capabilities

## Status Flags



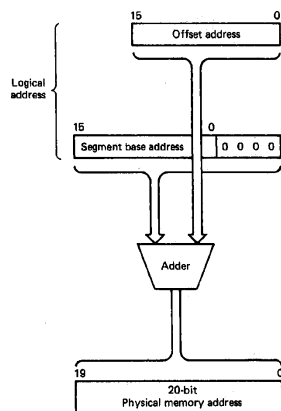
- \* **Carry flag (CF)**
  - \* 1 = carry-out or borrow-in from MSB of the result during the execution of an arithmetic instruction
  - \* 0 = no carry or borrow has occurred
- \* **Parity flag (PF)**
  - \* 1 = result produced has even parity
  - \* 0 = result produced has odd parity
- \* **Zero flag (ZF)**
  - \* 1 = result produced is zero
  - \* 0 = result produced is not zero
- \* **Sign bit (SF)**
  - \* 1 = result is negative
  - \* 0 = result is positive
- \* **Others**
  - \* Overflow flag (OF)
  - \* Auxiliary carry flag (AF)

## Control Flags



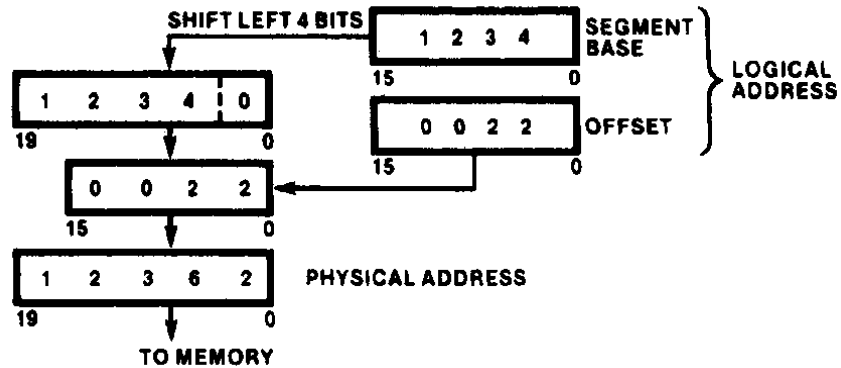
- \* Trap flag (TF)
  - \* 1/0 = turn on/off single-step mode
  - \* Mode useful for debugging
  - \* Employed by monitor to execute one instruction at a time (single step execution)
- \* Interrupt flag (IF)
  - \* Used to enable/disable external maskable interrupt requests
  - \* 1/0 = enable/disable external interrupts
- \* Direction flag (DF)
  - \* Used to determine the direction in which string operations occur
  - \* 1/0 = automatically decrement/increment string address—proceed from high address to low address

## Real Mode Memory Addressing

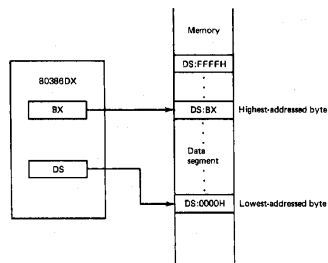


- \* Logical and Physical Addresses
- \* Logical address: real-mode architecture described by a segment base address and an offset
  - \* Segment base address (CS, DS, ES, SS, etc.) are 16 bit quantities
  - \* Offsets (IP, SI, DI, BX, DX, SP, BP, etc.) are 16bit
- \* Physical Address: actual address used for accessing memory
  - \* 20-bits in length
  - \* Formed by:
    - \* Shifting the value of the 16-bit segment base address left 4 bit positions
    - \* Filling the vacated four LSBs with 0s
    - \* Adding the 16-bit offset

## Generating a RM Memory Address



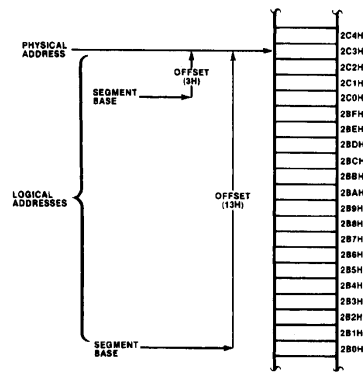
## Boundaries of a Segment



- \* Six active segments: CS, DS, ES, GS, FS, SS
- \* Each 64K-bytes in size → maximum of 384K-bytes of active memory
  - \* 64K-bytes for code
  - \* 64K-bytes for stack
  - \* 256K-bytes for data
- \* Starting address of a data segment  
DS:0H → lowest addressed byte
- \* Ending address of a data segment  
DS:FFFFH → highest addressed byte
- \* Address of an element of data in a data segment  
DS:BX → address of a byte, or word element of data in the data segment



# Logical and Physical Addresses



- \* Many different logical address can map to the same physical address

## Why?

- \* Said to be “aliases”
- \* Examples:
  - \* 2BH:13H → Physical address?
  - \* 2CH:3H → Physical address?

# Review

- \* Logical vs. physical addresses
- \* Logical address: base/offset register pair
  - \* i.e. CS:IP, DS:SI, SS:SP
- \* Physical address: actual address in memory
  - \* Calculate by shifting base left by 4 bits, adding offset
  - \* Example: CS = 0x1000 and IP = 0x1234
    - Physical addr. = 0x10000 + 0x1234 = 0x11234
- \* Can have multiple logical addresses mapping to same physical address: aliases

## Next time

- \* Stack Memory

