# Intel Architecture 32-bit (IA-32) Registers and Instruction Execution Cycle

Muhammad Afzaal
m.afzaal@nu.edu.pk

# Book Chapter

- “Assembly Language for x86 processors”
- Author “Kip R. Irvine”
- 6th Edition
- Chapter 2
  - Section 2.1.2
  - Section 2.2

# Registers

- Very high speed memory units inside CPU
- Can be categorized into
  - General Purpose Registers
  - Segment Registers
  - Instruction Pointer
  - EFLAGS Register
    - Control Flags
    - Status Flags

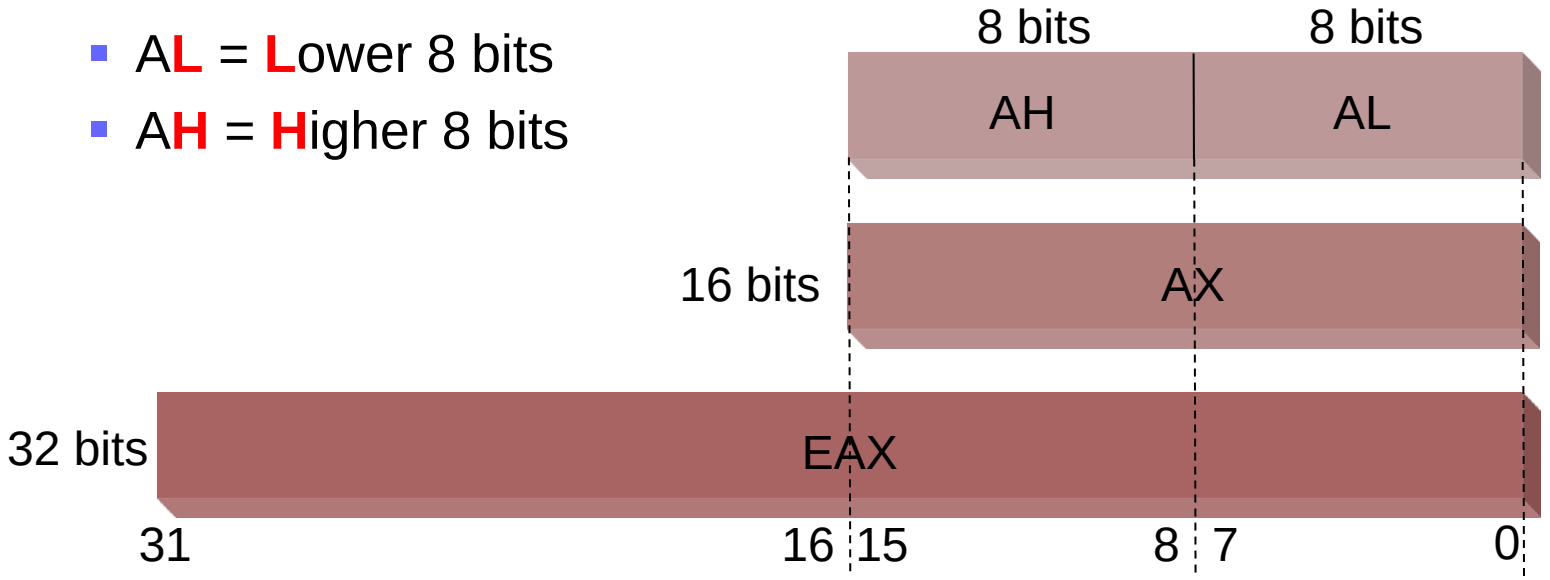
# General Purpose Registers

- Used primarily for arithmetic and data movement
- Eight 32-bit ***Extended*** general purpose registers

| Register | Description | Specialized Usage |
|---|---|---|
| E**A**X | **A**ccumulator | Used in *mul* and *div* instruction |
| E**B**X | **B**ase | |
| E**C**X | **C**ounter | Used in *LOOP* instruction |
| E**D**X | **D**ata | |
| E**BP** | **B**ase **P**ointer | Used to reference local variables on stack |
| E**SP** | **S**tack **P**ointer | Points at top of the stack, used by *PUSH* and *POP* |
| E**SI** | **S**ource **I**ndex | Used by high speed memory transfer instruction |
| E**DI** | **D**estination **I**ndex | Used by high speed memory transfer instruction |

# Accessing Parts of Registers (1/2)

- Portions of General Purpose Registers can also be accessed
  - Lower 16 bits of EAX are named AX
  - AX can further be divided into
    - A**L** = **L**ower 8 bits
    - A**H** = **H**igher 8 bits



# Accessing Parts of Registers (2/2)

- EAX, EBX, ECX, EDX can be accessed at word and byte level
- EBP, ESP, ESI, EDI can be accessed at word level

| 32-bit | 16-bit | 8-bit **H**igher | 8-bit **L**ower |
|---|---|---|---|
| EAX | AX | A**H** | A**L** |
| EBX | BX | B**H** | B**L** |
| ECX | CX | C**H** | C**L** |
| EDX | DX | D**H** | D**L** |

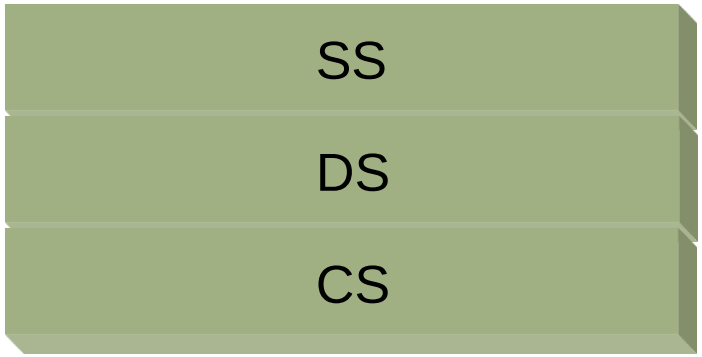
| 32-bit | 16-bit Lower |
|---|---|
| ESP | SP |
| EBP | BP |
| ESI | SI |
| EDI | DI |

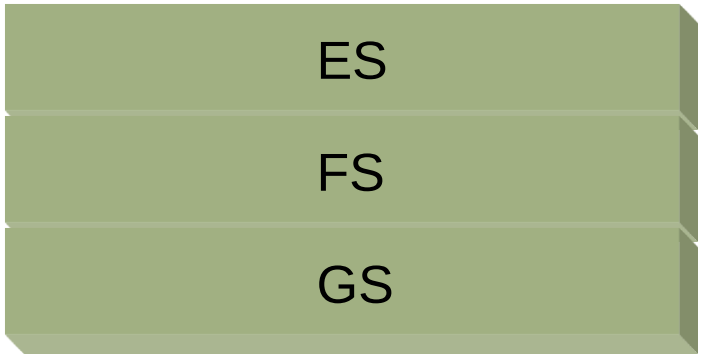
# Segment Registers (1/2)

- Six 16-bit segment registers
- Indicate base addresses of pre-assigned memory areas
- An assembly program usually contains 3 segments
  - Stack Segment: holds local function variables and function parameters
  - Data Segment: holds variables
  - Code Segment: holds program instructions

# Segment Registers (2/2)

- SS = Stack Segment
- DS = Data Segment
- CS = Code Segment
- 3 extra segment registers
  - ES, FS, GS (Extra Segment)



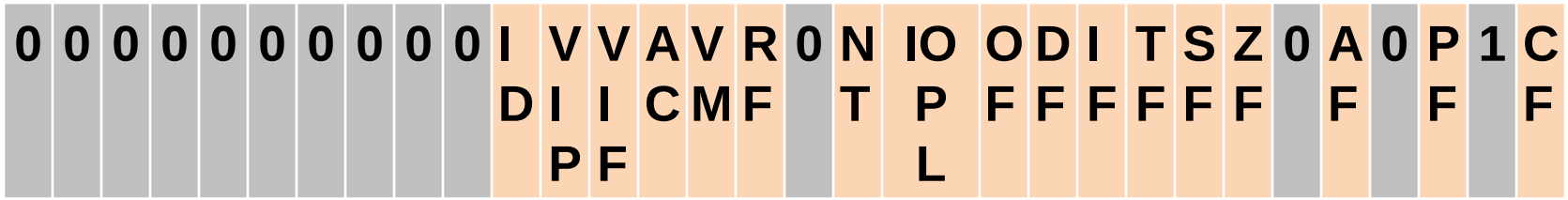



# Instruction Pointer

- Represented by EIP
- Also called Program Counter (PC)
- Contains the address of next instruction to be executed
- Its value changes with the execution of program
- Certain machine instructions influence the value of EIP

# EFLAGS

- EFLAGS register consists of individual binary bits
- Can be categorized into
  - Control Flags: control the operation of CPU
  - Status Flags: reflect the outcome of arithmetic and logical operations



# Status Flags

- Arithmetic and Logical operations set/reset them
- Status Flags are
  - Carry Flag (CF): set when unsigned arithmetic operation produces a carry
  - Overflow Flag (OF): sets when signed arithmetic result is out of range
  - Sign Flag (SF): sets when result of operation is –ve
  - Zero Flag (ZF): sets when result of operation is zero
  - Auxiliary Flag (AF): sets when there is a carry from bit 3 to bit 4
  - Parity Flag (PF): sets when parity is even

# Floating Point Unit (FPU)

- FPU performs high speed floating point operations
- Integrated into main processor chip from Intel486 onward
- Eight 80-bit floating point data registers
- All arranged as a stack

# Instruction Execution Cycle (1/2)

- A program is loaded into memory before execution
- Instruction Pointer (IP) contains the address of next instruction to be executed
- Instruction Queue contains the group of instructions about to be executed
- Three basic steps in execution of a machine instruction
  - Fetch
  - Decode
  - Execute
- Two more steps if the instruction uses an operand located in memory
  - Fetch Operand
  - Store Output

# Instruction Execution Cycle (2/2)

- Order of steps when instruction uses an operand located in memory
  - Instruction Fetch (IF)
  - Instruction  Decode (ID)
  - Operand Fetch from Memory (OF)
  - Instruction Execute (IE)
  - Write Back in Memory (WB)
- Instruction Execution Cycle (IEC) is the combination of all above operations

# Instruction Fetch

- Program Counter (PC) or IP holds address of next instruction to be executed
- Processor fetches instruction from memory pointed to be PC
- Increment PC to the next instruction
- Fetching operation concludes here

# Instruction Decode

- Control Unit (CU) determines the function of the instruction
- Instruction's input operands are passed to ALU
- Signals sent to ALU to perform the desired operation

# Operand Fetch from Memory

- Sometimes instructions may involve some operands located in memory
- In this case, data should be retrieved from memory
- The CU uses a memory *read* operation
- CU copies these operands into Instruction Registers which are invisible to user programs

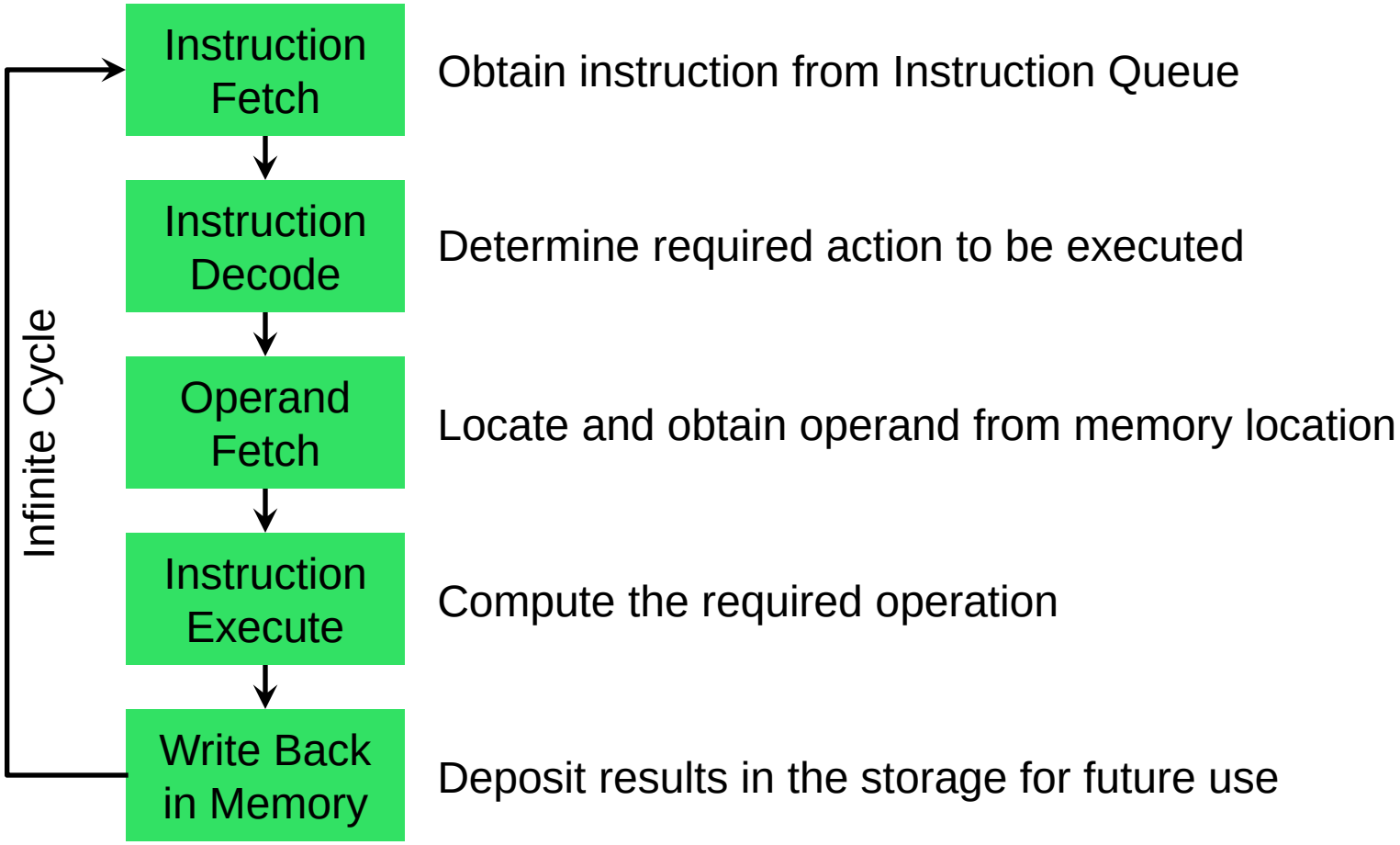
# Instruction Execute

- The instruction is ready for execution at this stage
- ALU executes the instruction
- ALU updates the status flags providing information about the result

# Write Back in Memory

- The output may be required to save in memory for future uses
- CU performs a *write* operation to save this output into memory

# IEC Graphical View



# Next Lecture

- Overview of Intel Microprocessors
- x86 Memory Management