CS 271 Lecture #3

Introduction to Intel IA-32 architecture

Preliminaries: **Metrics** (measurements)

- Speed (distance/time) is measured in electronic units:
  - K = 10^3, M = 10^6, G = 10^9, etc.
  - E.G., network speed of 8 Mbps means 8,000,000 bits per second
- Size in bits, Bytes is measured in binary units
  - Commonly used: K = 2^{10}, M = 2^{20}, G = 2^{30}, etc.
  - In this course, use: Ki = 2^{10}, Mi = 2^{20}, Gi = 2^{30}
  - E.G., disk size of 200 GiB means

    200 x 2^{30} Bytes = 214,748,364,800 Bytes

    = 1,717,986,918,400 bits
- Bytes and bits (abbreviations)
  - Use lower-case **b** for bits
  - Use upper-case **B** for Bytes
  - Example: 1 Mi**b** = 128 Ki**B**

Intel IA-32 architecture

- CISC
- Two modes of operation:
  - Protected
  - Real-address
- Two processors in one
  - integer unit
  - floating-point unit
  - Two processors can work in parallel (co-processors)
    - Separate instruction sets
    - Separate data registers
      - different configuration
    - Separate ALUs

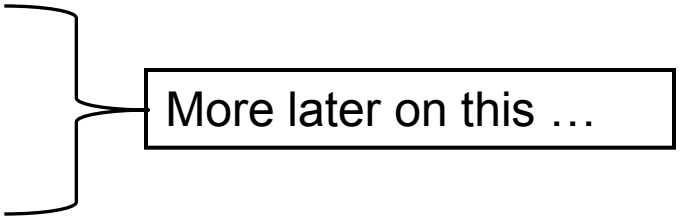
For now, we focus on the integer unit.

Intel IA-32 architecture

- Specific hardware implementations
  - registers
  - memory addressing scheme
- Specific instruction set and microprograms
- Specific assembly languages
  - MASM, NASM, TASM, etc,
- Specific operating systems
  - Windows, Linux, DOS, etc.

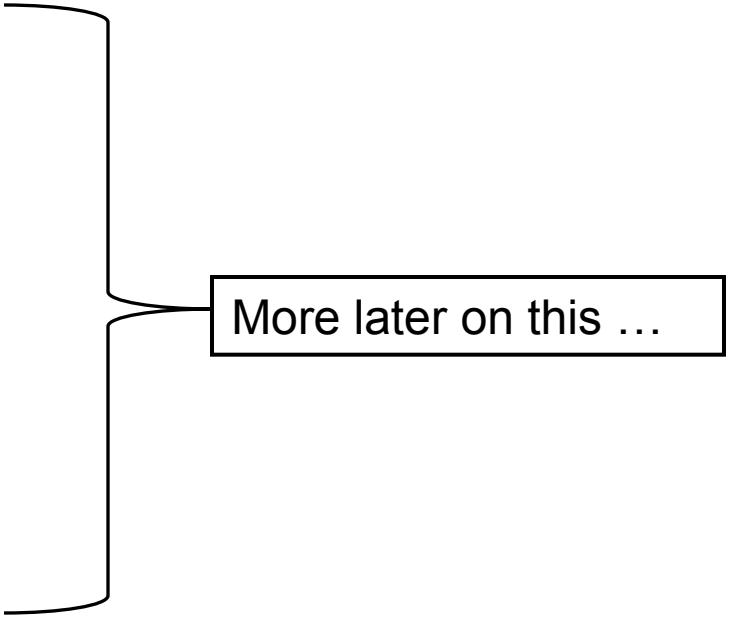
Intel IA-32 architecture

- Memory
  - Up to 4GiB
  - Byte-addressable
  - Little-endian
- 32-bit machine
  - Registers
  - Buses
  - ALU



Intel IA-32 architecture

- Byte is the smallest unit of data that can be manipulated directly in the IA-32 architecture.
- Operating system and instruction decoder determine how byte codes are interpreted
  - integer
  - character
  - floating-point
  - instruction
  - address
  - status bits



Integer Unit Registers

32-bit **general**-purpose registers

EAX
EBX
ECX
EDX

32-bit **multi**-purpose registers

EBP
ESP
ESI
EDI

32-bit **special**-purpose registers

EFL (status)
EIP (instruction pointer)
In protected mode, the Control Register, Instruction Register, MAR, and MDR are usually hidden.

16-bit **segment** registers

CS	ES
SS	FS
DS	GS

Integer unit registers

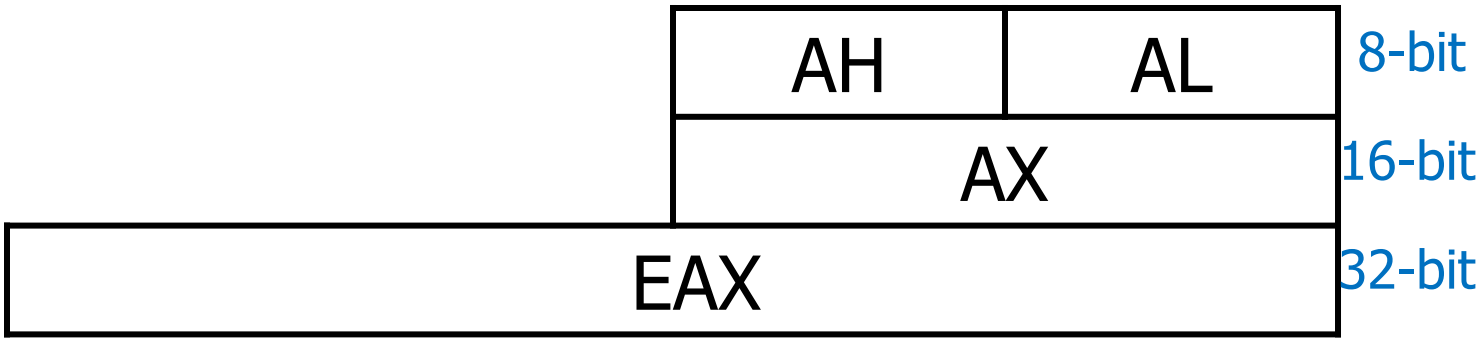
- Most of the 32-bit registers are visible during MASM debugging
  - The 32-bit “general” and “multi” registers may be manipulated directly
  - The 32-bit “special” registers are manipulated by the micro-programs that implement the instructions

Integer unit registers

- Some “general-purpose” and “multi-purpose” registers are used for special purposes:
  - EAX and EDX are automatically used by integer multiplication and division instructions
  - ECX is automatically used as a counter for some looping instructions
  - ESP is used for referencing the system stack
  - etc.

Integer unit registers

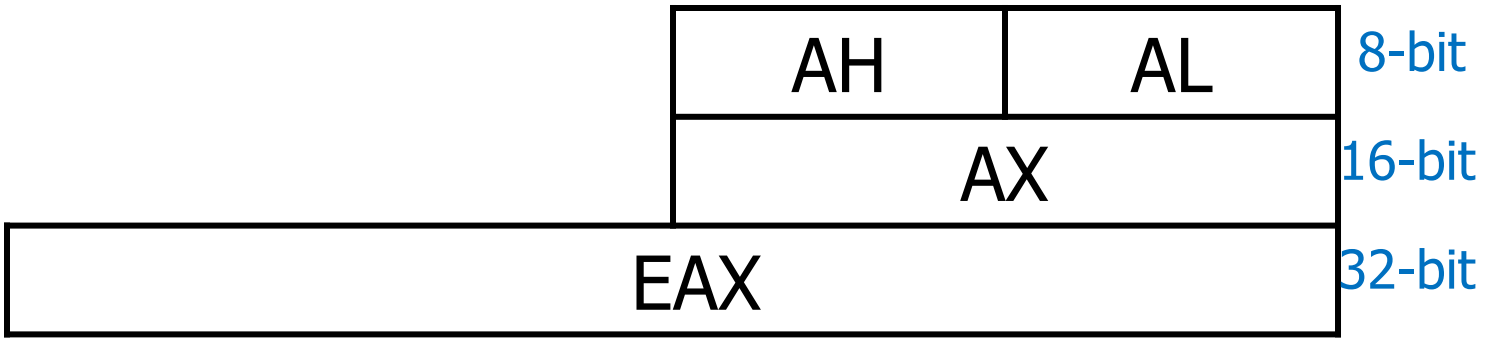
- Some 32-bit registers have 8-bit and 16-bit "sub-registers"
  - EAX, EBX, ECX, EDX
  - Example: Sub-registers of EAX
    - AX refers to the least-significant 16-bits of EAX
    - AL refers to the least-significant 8-bits of AX
    - AH refers to the most-significant 8-bits of AX



Integer unit registers

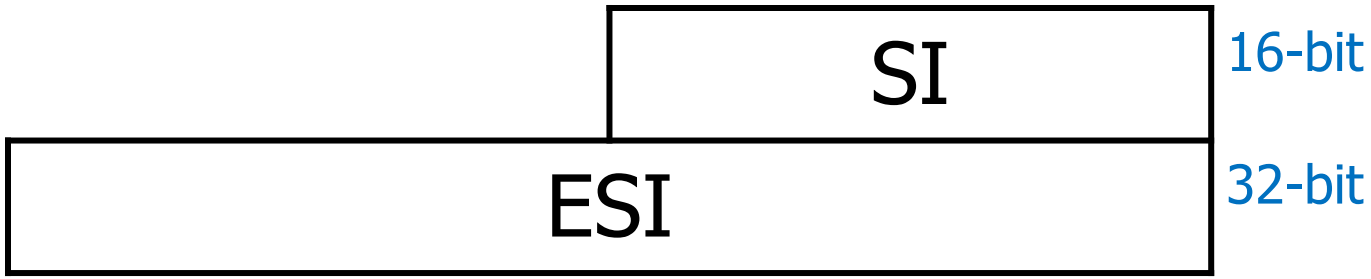
- Note: If you change a sub-register, the value in the entire register is changed.
- Example:
  - Suppose that EAX contains the electrical representation of **`67890`**.
  - We now give the instruction **`mov AL, 27`**
  - The new value in EAX is **`67867`**

If this doesn't look right, don't worry … yet.



Integer unit registers

- Some 32-bit registers have only 16-bit "sub-registers"
  - ESI, EDI, EBP, ESP
  - Example: Sub-registers of ESI
    - SI refers to the least-significant 16-bits of ESI



There's only one set of registers for the integer unit!

- Something like global variables
- Sometimes have to be saved and restored.

- Most register instructions (for now) reference EAX, EBX, ECX, and/or EDX