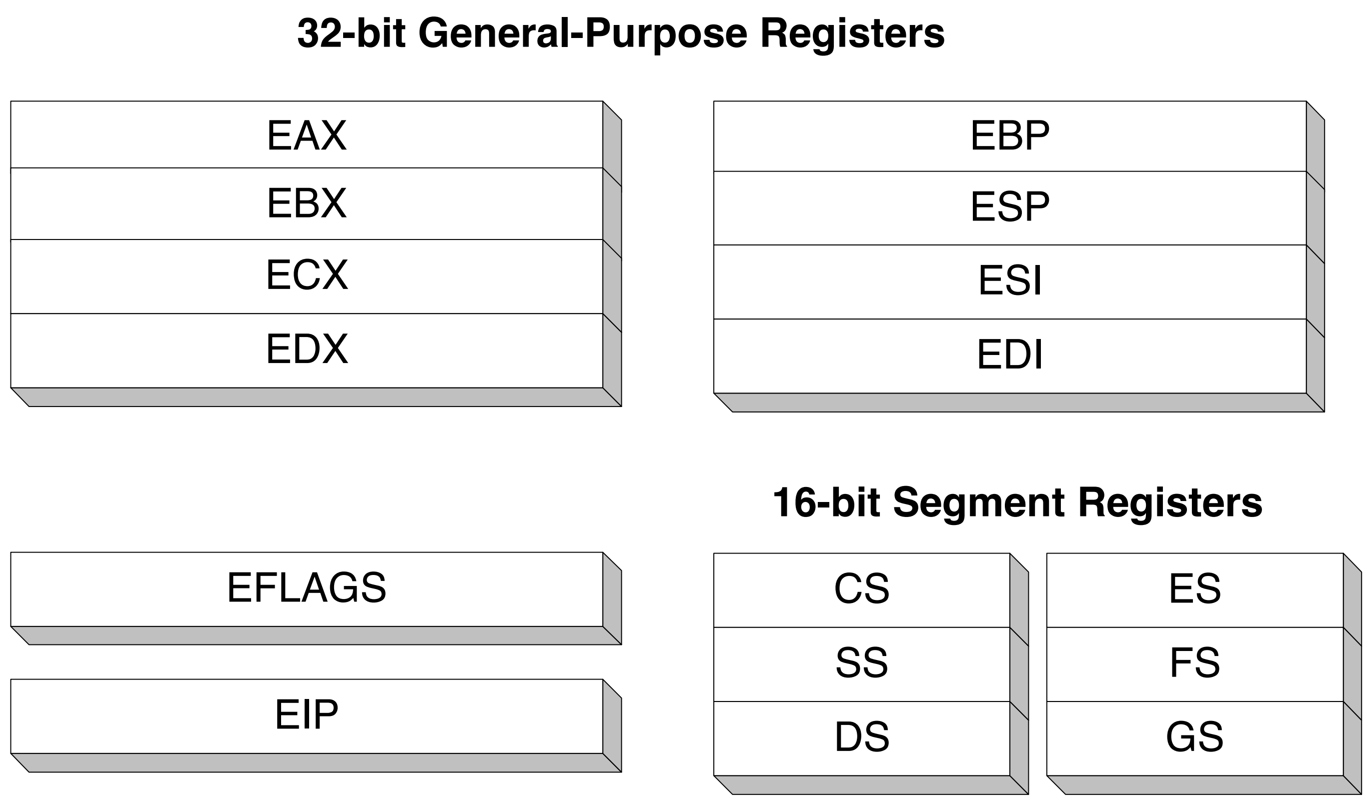
Computer Architecture CS-211

Spring 2017

Recitation #6

Registers Overview

- Named storage locations inside the CPU, optimized for speed



ASCII

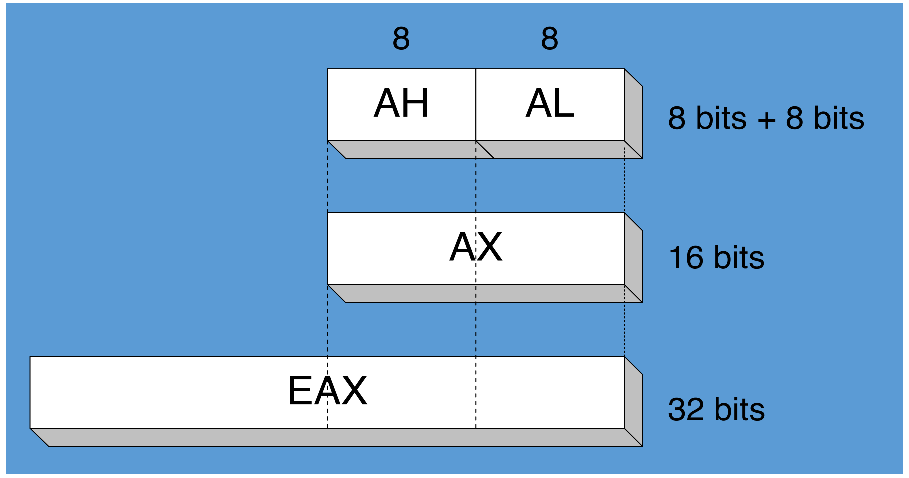
- Computers can only understand numbers
- an ASCII code is the numerical representation of a character such as 'a' or '@' or an action of some sort.
- http://www.asciitable.com

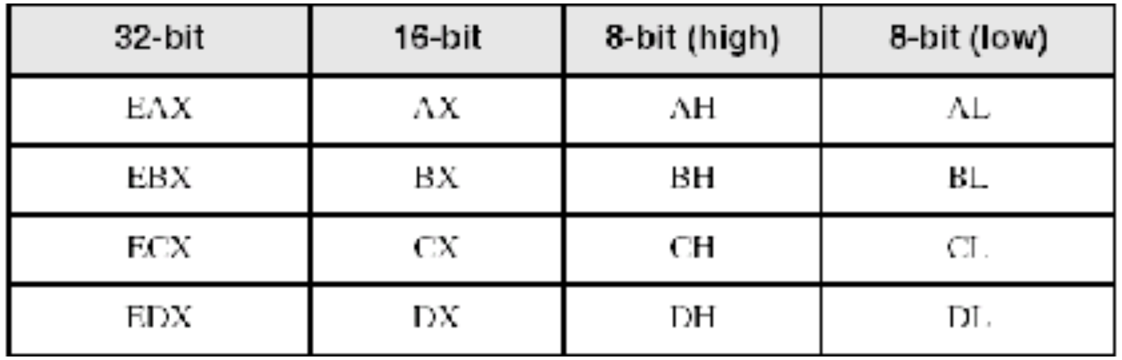
Data Registers

- AX is the primary accumulator
  - Used in most arithmetic instruction
- BX is the base register
  - Could be used in indexed addressing
- CX is the count register
  - Store the loop count in iterative operations
- DX is the data register
  - Used in input / output operations

Data Registers

Can use 8-bit, 16-bit, or 32-bit name





32-bit	16-bit	8-bit (high)	8-bit (low)
EAX	AX	AH	AL
EBX	BX	BH	BL
ECX	CX	CH	CL
EDX	DX	DH	DL

Pointer Registers

- ESP is stack pointer
    - It refers to be current position of data or address within the program stack
    - Changed by push, pop instructions
- EBP is frame pointer
    - Referencing the parameter variables passed to a subroutine
- EIP is instruction pointer
    - It stores the offset address of the next instruction to be executed

Stack Operation

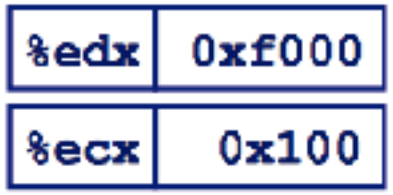
- By convention, %esp is used to maintain a stack in memory

  - %esp contains the address of top of stack
- Instructions to push (pop) content onto (off of) the stack
  - pushl %eax

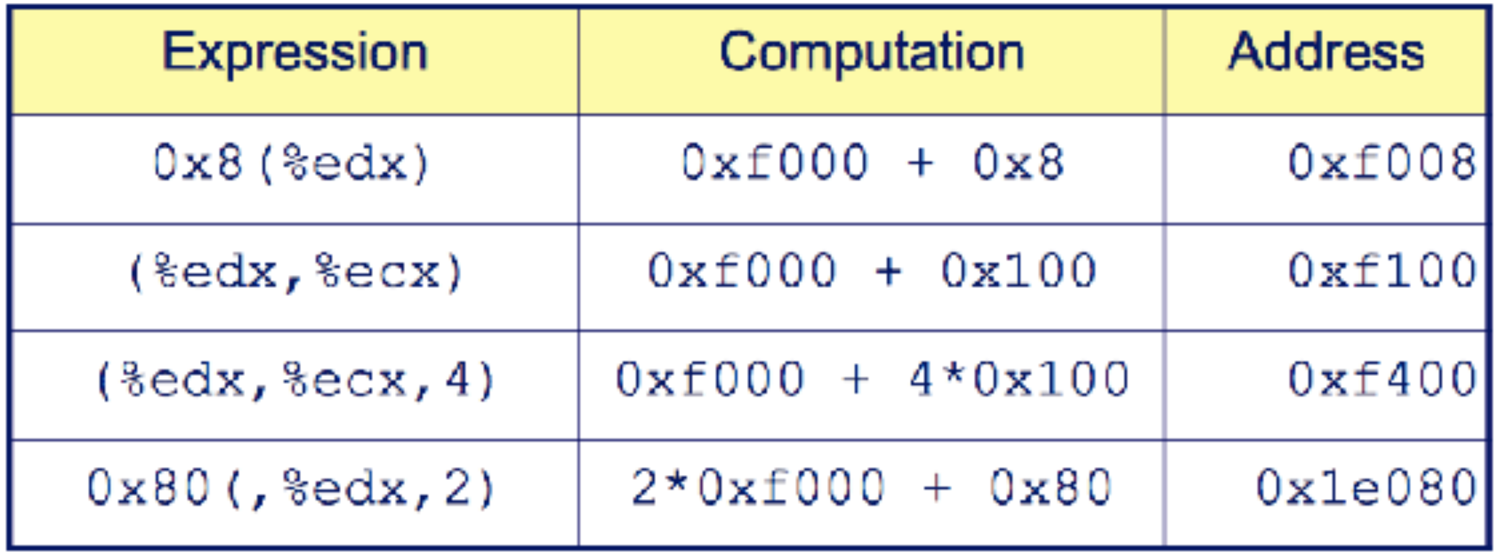
    esp = esp – 4 l Memory[esp] = eax
  - popl %ebx

    ebx = Memory[esp] l esp = esp + 4

Address computation



%edx	0xf000
%ecx	0x100



Expression	Computation	Address
0x8(%edx)	0xf000 + 0x8	0xf008
(%edx,%ecx)	0xf000 + 0x100	0xf100
(%edx,%ecx,4)	0xf000 + 4*0x100	0xf400
0x80(,%edx,2)	2*0xf000 + 0x80	0x1e080

Control Registers

- Overflow flag (OF)
    - Indicates the overflow of a high-order bit
- Carry flag (CF)
    - Contains the carry of 0 or 1 from high-order bit after arithmetic operation
    - Stores the last bit of a shift or rotate operation
- Sign flag (SF)
    - Shows the sign of the result of an arithmetic operation
    - Positive -> 0, Negative -> 1
- Zero Flag (ZF)

gdb

- A good tutorial for debugging assembly with gdb
  - https://www.csee.umbc.edu/~cpatel2/links/310/nasm/gdb_help.shtml
- gcc -m32 fib.c -g -o fib
- gdb fib
- r (arg1) (arg2) ..
- c (continue)
- layout asm
- b * address
- p /x $eax
- ni
- si
- info r
- x addr -> (x / x) address = shows the hex / (x/d) address = shows the decimal