# CMSC 313
# COMPUTER ORGANIZATION
# &
# ASSEMBLY LANGUAGE PROGRAMMING

LECTURE 04, SPRING 2013

# TOPICS TODAY

- Recap i386 Basic Architecture
- `toupper.asm`
- `gdb` debugger demo

# Recap i386 Basic Architecture

- Registers are storage units inside the CPU.
- Registers are much faster than memory.
- 8 General purpose registers in i386:
  - ◇ EAX, EBX, ECX, EDX, ESI, EDI, EBP, ESP
  - ◇ subparts of EAX, EBX, ECX and EDX have special names
- The instruction pointer (EIP) points to machine code to be executed.
- Typically, data moves from memory to registers, processed, moves from registers back to memory.
- Different addressing modes used.

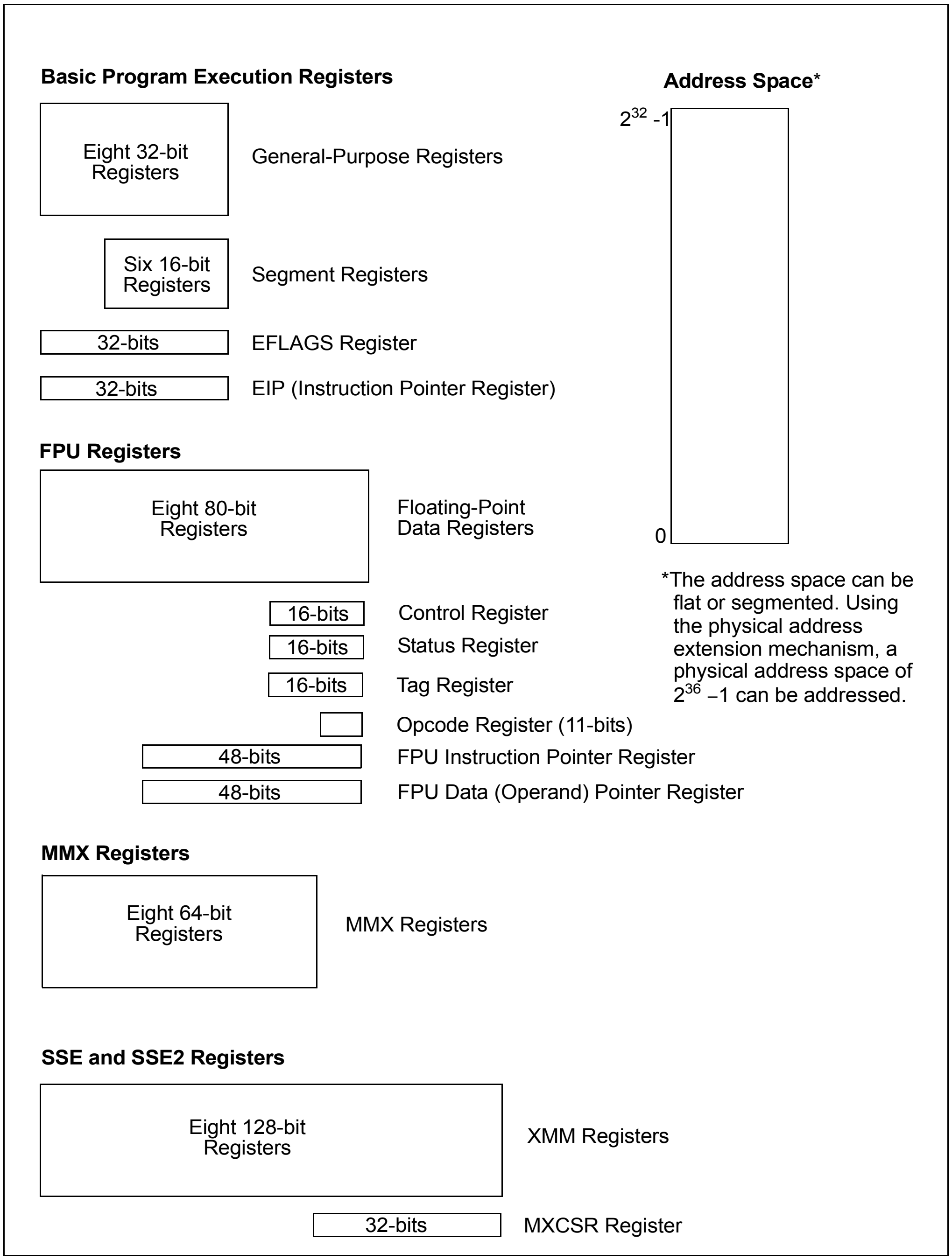
Basic Program Execution Registers
Eight 32-bit Registers
General-Purpose Registers
Six 16-bit Registers
Segment Registers
32-bits
EFLAGS Register
32-bits
EIP (Instruction Pointer Register)
FPU Registers
Eight 80-bit Registers
Floating-Point Data Registers
16-bits
Control Register
16-bits
Status Register
16-bits
Tag Register
Opcode Register (11-bits)
48-bits
FPU Instruction Pointer Register
48-bits
FPU Data (Operand) Pointer Register
MMX Registers
Eight 64-bit Registers
MMX Registers
SSE and SSE2 Registers
Eight 128-bit Registers
XMM Registers
32-bits
MXCSR Register
Address Space*
2^32 -1
0
*The address space can be flat or segmented. Using the physical address extension mechanism, a physical address space of 2^36 −1 can be addressed.

**Figure 3-1. IA-32 Basic Execution Environment**

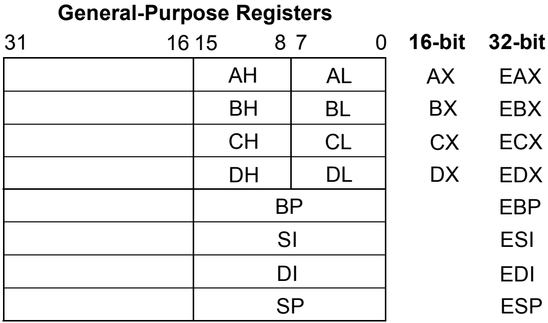

**Figure 3-4. Alternate General-Purpose Register Names**

# toupper.asm

- Prompt for user input.
- Use Linux system call to get user input.
- Scan each character of user input and convert all lower case characters to upper case.
- Use gdb to trace the program.

# THE GDB DEBUGGER

# Debugging Assembly Language Programs

- Cannot just put print statements everywhere.
- Use gdb to:
  - ◇ examine contents of registers
  - ◇ exmaine contents of memory
  - ◇ set breakpoints
  - ◇ single-step through program
- READ THE GDB SUMMARY ONLINE!

### Summary of `gdb` commands

| **Command** | **Example** | **Description** |
|---|---|---|
| run | | start program |
| quit | | quit out of gdb |
| cont | | continue execution after a break |
| break [addr] | break *_start+5 | sets a breakpoint |
| delete [n] | delete 4 | removes nth breakpoint |
| delete | | removes all breakpoints |
| info break | | lists all breakpoints |
| list _start | | list a few lines of the source code around _start |
| list 7 | | list 10 lines of the source code starting on line 7 |
| list 7, 20 | | list lines 7 thru 20 of the source code |
| stepi | | execute next instruction |
| stepi [n] | stepi 4 | execute next n instructions |
| nexti | | execute next instruction, stepping over function calls |
| nexti [n] | nexti 4 | execute next n instructions, stepping over function calls |
| where | | show where execution halted |
| disas [addr] | disas _start | disassemble instructions at given address |
| info registers | | dump contents of all registers |
| print/d [expr] | print/d $ecx | print expression in decimal |
| print/x [expr] | print/x $ecx | print expression in hex |
| print/t [expr] | print/t $ecx | print expression in binary |
| x/NFU [addr] | x/12xw &msg | Examine contents of memory in given format |
| display [expr] | display $eax | automatically print the expression each time the program is halted |
| info display | | show list of automatically displays |
| undisplay [n] | undisplay 1 | remove an automatic display |

# NEXT TIME

- i386 Instruction Set Overview
- i386 Basic Instructions
- Arithmetic Instructions
- EFLAGS Register
- Conditional Jump Instructions
- Using Jump Instructions