x86 or
Oh No! Not Another Assembler

Jonathan Misurda
jmisurda@cs.pitt.edu

CISC vs. RISC

- x86 is the epitome of a Complex Instruction Set Computer
  - Hundreds of instructions
- **F2XM1 – Compute 2^x-1**
  - Computes the exponential value of 2 to the power of the source operand minus 1. The source operand is located in register ST(0) and the result is also stored in ST(0). The value of the source operand must lie in the range -1.0 to +1.0. If the source value is outside this range, the result is undefined.

32-Bit General Purpose Registers

- EAX – Accumulator
- EBX – Base
- ECX – Counter
- EDX –Data

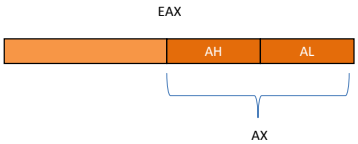
- ESI – String Source
- EDI – String Destination

Other 32-Bit Registers

- EIP – Instruction Pointer
- ESP – Stack Pointer
- EBP – Base or Frame Pointer

- EFLAGS – Flag register

Register Subfields



Hello World

```
 .file   "asm.c"
        .section        .rodata.str1.1,"aMS",@progbits,1
.LC0:
        .string "hello world!"
        .text
.globl main
        .type   main, @function
main:
        pushl   %ebp
        movl    %esp, %ebp
        subl    $8, %esp
        andl    $-16, %esp          ;1111 1111 1111 0000
        subl    $16, %esp
        movl    $.LC0, (%esp)
        call    puts
        movl    $0, %eax
        leave
        ret
        .size   main, .-main
        .section        .note.GNU-stack,"",@progbits
        .ident  "GCC: (GNU) 3.4.6 20060404 (Red Hat 3.4.6-8)"
```

AT&T Syntax

- `gcc` and `gas` use AT&T syntax:
  - Opcode appended by type
    - b – byte (8-bit)
    - w – word (16-bit)
    - l – long (32-bit)
    - q – quad (64-bit)
  - First operand is source
  - Second operand is destination
  - Memory dereferences are denoted by ( )

Intel Syntax

- Microsoft (MASM), Intel, NASM
  - Type sizes are spelled out
    - BYTE – 1 byte
    - WORD – 2 bytes
    - DWORD – 4 bytes (double word)
    - QWORD – 8 bytes (quad word)
  - First operand is destination
  - Second operand is source
  - Dereferences are denoted by [ ]

Intel Hello World

```
main:
        push    $ebp
        mov     $ebp, $esp
        sub     $esp, 8
        and     $esp, -16 ;1111 1111 1111 0000
        sub     $esp, 16
        mov     DWORD PTR [%esp], .LC0
        call    puts
        movl    $eax, 0
        leave
        ret
```

Stacks, Frames, and Calling Conventions

Jonathan Misurda
jmisurda@cs.pitt.edu

Stack

- Calling Convention
  - An agreement, usually created by a system's designers, on how function calls should be implemented
- Stack
  - A portion of memory managed in a last-in, first-out (LIFO) fashion
- Function Call
  - A control transfer to a segment of code that ends with a return to the point in code immediately after where the call was made (the *return address*)

Activation Records

- An object containing all the necessary data for a function
  - Values of parameters
  - Count of number of arguments
  - Return address
  - Return value
  - Value of $SP for Activation Record Below
- Also called a Frame

Temporary Storage

- Caller-Saved
  - A piece of data (e.g., a register) that must be explicitly saved if it needs to be preserved across a function call
- Callee-Saved
  - A piece of data (e.g., a register) that must be saved by a called function before it is modified, and restored to its original value before the function returns

MIPS Calling Convention

- First 4 arguments $a0-$a3
  - Remainder put on stack
- Return values $v0-$v1
- $t0-$t9 are caller-saved temporaries
- $s0-$s9 are callee-saved

x86 Calling Convention

- $EAX, $ECX, and $EDX are generally caller-saved
- Three registers are probably insufficient
  - Most registers are “spilled” onto the stack
- $EAX is the return value
- Everything else is on the stack

Hello World

```
 .file   "asm.c"
        .section        .rodata.str1.1,"aMS",@progbits,1
.LC0:
        .string "hello world!"
        .text
.globl main
        .type   main, @function
main:
        pushl   %ebp
        movl    %esp, %ebp
        subl    $8, %esp
        andl    $-16, %esp          ;1111 1111 1111 0000
        subl    $16, %esp
        movl    $.LC0, (%esp)
        call    puts
        movl    $0, %eax
        leave
        ret
        .size   main, .-main
        .section        .note.GNU-stack,"",@progbits
        .ident  "GCC: (GNU) 3.4.6 20060404 (Red Hat 3.4.6-8)"
```

Hello World Stack