Assembly Language – Control Flow

• Learning Objectives
  • Interpret test and cmp instructions
  • Explain the different flags in the x86 architecture, how they get set and how they get used
  • Follow a program’s control flow in assembly
Control Flow

• So far, all the programs we’ve looked at execute in address order – that is, we don’t have any conditional statements or calls to other functions.
• In this video we’ll focus on conditional statements and in the next we’ll dig into calling conventions.
• Getting started:
  • We’ve discussed ways to change various registers: the general registers, the stack pointer…
  • Now, control flow instructions create ways to change the value of the instruction pointer (eip).
Control Flow Overview

• Unconditional:
  • Directs the processor to execute at an address other than the next sequential address.

• Conditional:
  • Paired instructions:
    • Typically some sort of instruction that sets condition flags (an arithmetic, a logical, a cmp, or a test – most frequently a cmp or test)
    • A conditional jump instruction that changes the point of execution depending on the state of the condition flags.
  • Condition flags:
    • A set of bits stored in a special register called EFLAGS
Condition Flags

- **SF: Sign Flag**
  - The most recent operation yielded a negative value.
  - Equal to MSB of result; which indicates the sign of a two’s complement integer.
  - 0 means result was positive, 1 means negative.

- **CF: Carry Flag**
  - The most recent operation generated a carry bit out of the MSB.
  - Indicates overflow when performing unsigned integer arithmetic.

- **OF: Overflow Flag**
  - The most recent operation caused a 2’s complement overflow (either positive or negative).
  - Indicates an overflow when performing signed integer arithmetic.
  - Ignored during unsigned arithmetic.

- **ZF: Zero Flag**
  - The most recent operation yielded a zero.

- **Condition flags are set** *implicitly* by every arithmetic instruction
- **Condition flags are set** *explicitly* by comparison and test instructions
Comparison Instructions

• **cmpl** `src1, src2`
  • Compares value of `src1` and `src2`
  • `src1, src2` can be registers, immediate values, or contents of memory.
  • Computes `(src2 - src1)` without modifying either operand
    • like “subl src1, src2” without changing `src2`
  • But, sets the condition flags based on the result of the subtraction.

• **testl** `src1, src2`
  • Like `cmpl`, but computes `(src1 & src2)` instead of subtracting them.
Jump Instructions

• Use for signed or unsigned operations
  • JE/JZ: jump if equal/jump if 0 (ZF=1)
  • JNE/JNZ: jump if not equal/jump if not 0 (ZF=0)

• Use for signed operations
  • JL/JNGE: jump if less/jump if not greater or equal (SF != OF)
  • JLE/JNG: jump if less than or equal to/jump if not greater than (ZF=1 or SF != OF)
  • JG/JNLE: jump if greater than/jump if not less than or equal to (ZF=0 and SF=OF)
  • JGE/JNL: jump if greater than or equal to/jump not less than (SF=OF)

• A handy reference: http://unixwiz.net/techtips/x86-jumps.html
More Jump Instructions

• Use for unsigned operations
  • JB/JNAE/JC: jump if below/jump if not above or equal/jump if carry (CF=1)
  • JA/JNBE: jump if above/jump if not below or equal (CF=0 and ZF=0)