Assembly Language Programming IV

x86-64 Architecture

CSCI 2050U - Computer Architecture

Randy J. Fortier
@randy_fortier

OntarioTech UNIVERSITY

Outline

- Comparisons
- Unconditional jumps
- Conditional jumps
- Implementing conditionals
- Implementing loops

Branching and Jumping

CSCI 2050U - Computer Architecture

Branching/Jumping

- Branching (also called jumping) is when program flow does not simply flow to the next instruction
- A branch instruction may modify the `RIP` register
- Unconditional jumping
    - Always set the `RIP` register to the specified address
    - Basically, a `GOTO` statement
- Conditional jumping
    - Jump only when some condition is true
    - e.g. the value of some flag

Comparisons

- The `cmp` instruction is similar to a subtract, except that it doesn't modify its operands
  - Only the flags are modified

Flags

Flag	Meaning
Z (Zero)	Set when the result of an arithmetic operation is zero
O (Overflow)	Set when an arithmetic operation resulted in overflow
S (Sign)	Set when an arithmetic operation resulted in a negative result
C (Carry)	Set when an arithmetic operation resulted in carry

Conditional Jump Instructions

- These instructions jump when certain flags are set (or are not set)
  - Often by a `cmp` instruction

Intuition	Unsigned	Signed
`==`	`je`	`je`
`!=`	`jne`	`jne`
`<`	`jb`	`jl`
≤	`jbe`	`jle`
`>`	`ja`	`jg`
≥	`jae`	`jge`

Conditional Jump Instructions

- These instructions jump when certain flags are set in other ways:

Instruction	Flags
`jz`	`Z=1`
`jnz`	`Z=0`
`jc`	`C=1`
`jnc`	`C=0`
`jo`	`O=1`
`jno`	`O=0`
`js`	`S=1`
`jns`	`S=0`

Conditional Jump Instructions

- These instructions jump depending on the value of the rcx, (ecx, …) register
  - This register is often used as a loop counter

Instruction	Flags
`jcxz`	`RCX==0`
`jcxnz`	`RCX!=0`

Implementing Conditionals

- Conditional jumps make it easy to implement if/elseif/else statements

C++	Assembly

`int num = …;`
`if (num < 10) {`
`  // do something`
`}`
`// the rest of the code`
	`  mov rax, [num]`
	`  cmp rax, 10`
	`  jge skipCond`
	
	`  ; do something`
	
	`skipCond:`
	`  ; the rest of the code`

Implementing Loops

- Conditional jumps also make it easy to implement while, do/while, and for loops

C++

```cpp
int num = …;
while (num < 10) {
   // do something
   num++;
}
// the rest of the code
```

Assembly

```asm
   mov rax, [num]

loopStart:
   cmp rax, 10
   jge loopExit

   ; do something

   inc rax
   jmp loopStart

loopExit:
   ; the rest of the code
```

Wrap-Up

- Comparisons
- Unconditional jumps
- Conditional jumps
- Implementing conditionals
- Implementing loops

What is Next?

- Creating functions
  - Function definitions
  - Passing arguments
  - Returning values