CMPT 295
Machine-Level Programming
Lecture 14 – Program Control - Loops
In x86-64 assembly, there are no conditional statements if/else can be implemented in x86-64 assembly using branching:
- `cmp*` instruction
- `jX` instructions (jump)
- `cmovX` instructions -> conditional move
- Condition codes: ZF, CF, SF, OF
- `cmp*` and `test*` instructions set condition codes
- CPU uses these condition codes to decide whether a
  - `jX` instruction (conditional jump) is to be executed
  - `cmovX` instruction (conditional move) is to be executed
- Examples – various ways of implementing ...
  - `int abs(int x)`
  - `int max(int x, int y)`
Today’s Menu

- Introduction
  - C program -> assembly code -> machine level code
- Assembly language basics: data, move operation
  - Memory addressing modes
- Operation leaq and Arithmetic & logical operations
- Conditional Statement – Condition Code + cmovX
- Loop
- Function/procedure call – Stack – Recursion
- Array
- Floating-point operations
While loop – “coding the false condition first”

**in C:**
```
while (x < y) {
    // stmts
}
```

**in assembly:**
```
loop:
    if cond false
goto done:
    stmts
    goto loop:
endloop:
    ret
```

Loop Pattern 1
While loop – “jump-to-middle”

in C:
while (x < y) {
    // stmts
}

in assembly:

loop:
    # stmts

endloop:
    ret

Loop Pattern 2

goto test:
goto loop:
stmts
test:
    if cond true
    goto loop:
done:
Do While loop – “jump-to-middle”

in C:

```c
do {
    stmts
} while (x < y);
```

in assembly:

```
loop:
    # stmts
 test:
    endloop:
    ret
```

Loop Pattern 2

goto test:
    loop:
    stmts
 test:
    if cond true
goto loop:
done:
For loop

In C:

for (i = 0; i < n; i++) {
    // stmts
}

while (i < n) {
    // stmts
}

In Assembly:

xorl %ecx, %ecx
loop:
    cmpl %edi, %ecx
    jge endloop
    ...  # stmts
    incl %ecx
    jmp loop
endloop:
    ret
Summary

- In x86-64 assembly, there are no iterative statements.
- To alter the execution flow, compiler generates code sequence that implements these iterative statements (while, do-while and for loops) using branching method:
  - `cmp*` instruction
  - `jx` instructions (jump)
- 2 loop patterns:
  - “coding the false condition first” -> while loops (hence for loops)
  - “jump-in-middle” -> while, do-while (hence for loops)
Next lecture

- Introduction
  - C program -> assembly code -> machine level code
  - Assembly language basics: data, move operation
  - Memory addressing modes
- Operation `lea` and Arithmetic & logical operations
- Conditional Statement – Condition Code + `cmovX`
- Loop
- Function/procedure call – Stack – Recursion
- Array
- Floating-point operations