LECTURE – OCTOBER 22, 2013

Overhead:

Midterms:
Average: 75
Median: 76
Standard Deviation: 9
*Highest score was a 93 which indicates the test was a little too hard. – Eddie Kohler

Problem Set 3:
Pset 3 is easier than Pset 2. Some people got 45x stdio on Pset2. “Well done, you deserve a Snickers.” – Eddie Kohler
By the end of today you should know everything you need to know to make progress on the problem set. Next week is virtual memory. Prepare yourselves.

Class:

Example:
f24.s: Returns max of a and b
    movl b, %ecx // moves global “b” into ecx
    movl a, %eax
    cmpl %ecx, %eax // compares %ecx with %eax.
    jge .LBB0_2 // jump if greater than or equal to
    movl %ecx, %eax
.LBB0_2:
    ret

*Note: use “info registers” in gdb to get info about registers in gdb

Condition Codes:
“Hidden” register st as a side effect of many arithmetic instructions.
Combines many flags
ZF – zero flag
OF – overflow flag
CF – carry flag
SF – sign flag

Comparison Instructions:
compl a, b => “subl a, b” without modifying b
testl a, b => “andl a, b” without modifying b

Conditional Jumps:
  jge means “jump if greater than or equal to” zero
  ja means “jump if above”
*Note: unixwiz.net/techtips/x86-jumps.html

  jge will occur if SF = OF, why?
  0111 – 1000 => 1111
  SF = 1 and OF = 1
Comparisons with signed integers:

0010 > 0001
1000 > 0111 (if unsigned)
1000 < 0111 (if signed)

More examples:

f25.s: This will return a if b == x, else will return b.

```assembly
movl b, %eax
cmp x, %eax
jne .LBB0_2 // jne if ZF is true
movl a, %eax
.LBB0_2: # %select.end
    ret
```

f26.s is the exact opposite of f25.s

```assembly
f28.s
    cmpl $0, a
    sete %al // puts the zero flag into %al,
    movzbl %al, %eax // move zero byte to long, sets all of %eax to zero except for %al
    ret
```

f29.c also compiles to the same assembly code:

```c
extern char* a;
int f(void) {
    return !a;
}
```

```assembly
f31.s
    movl x, %eax
    shll $2, %eax
    addl a, %eax
    ret
```

Possible c codes:

```c
char* a;
    return &a[4 * x];
int* a;
    return &a[x];
```
f32.s:
    movzbl a, %eax
    ret

c code:
    unsigned char a;

    int f(void) {
        return a;
    }

What if it was signed?
    char a;
    int f(void) {
        return a;
    }

Signed extensions:
    1111 ----ZBL-------> 00001111
    1111 ----SBL-------> 11111111

*Then f32.s would have to use movsbl instead of movzbl if it was signed

f33.s
    movl a, %eax
    movzbl (%eax), %eax // stores pointee of %eax into %eax
    // a is likely a pointer to a char because of movzbl
    ret

f34.s
    movl a, %eax
    movl a, %ecx
    movzbl (%ecx, %eax), %eax // look at general x86 assembly below!
    ret

General x86 assembly indirection
    off(base, index, size)
    dereference off + base + index * size

    off defaults to 0
    index defaults to 0
    size can be \{1, 2, 4, 8\} default size is 1

f37.s
    movl x, %eax
    movzwl (%eax), %eax // x is an unsigned short *
    ret
f38.s: sums together an array of ints
    xorl   %eax, %eax // set eax to 0
    movl   x, %ecx
    testl  %ecx, %ecx
    je .LBB0_3 // if(x == 0) return 0;
    xorl   %eax, %eax
    movl   a, %edx
    .align 16, 0x90
.LBB0_2:                                # =>This Inner Loop Header: Depth=1
    addl   (%edx), %eax     // eax += (%edx)
    addl   $4, %edx // (%edx += 1) b/c this is a pointer to an int
    decl   %ecx
    jne .LBB0_2
.LBB0_3:                                 # %._crit_edge
    ret

f39.s: returns the first parameter of a function
    movl 4(%esp), %eax
    ret