Recitation J: Additional notes, Brandon Lum

1 %eax vs (%eax)

There was some confusion on the difference between %eax and (%eax) during recitation yesterday. Here are some examples to help clear up the concept.

%eax refers to the register %eax
(%eax) refers to the value in the register %eax

For example consider the current state of the machine:

```
%eax = 0xffffe008
%ebx = 0x0

Memory:
0xffffe008: 0xaabbccdd
```

- The instruction `movw %eax, %ebx` would result in `%ebx = 0xffffe008`.
- The instruction `movw (%eax), %ebx` would result in `%ebx = 0xaabbccdd`.

2 Cmp, Tests and Jumps

As you may have noticed, cmp and test are instructions used in conjunction with jumps to create conditionals/loops in assembly.

Each jump has an intuitive name, for example.

- je - jump equal
- jne - jump not equal
- jg - jump greater
- jge - jump greater than or equal

However, in some cases, it may not be intuitive. For example, in the case of bomb lab, many will see:

```
test %eax, %eax
je 0x44f80
```

At first glance, it may look as though we are testing %eax against itself, and that it should always be equal, but this is not true.

Do note that how the processor actually makes this decision is through the flags that are set. These namely are the Zero Flag (ZF), Sign Flag (SF), etc. (More details below in the appendix). You may view these and the registers in GDB with the command `info registers`.

2.1 Compare Example

`cmp x, y` calculates $y - x$ and sets the flag according to the result of $y - x$.

```
cmp 0x3, 0x1
je 0x44f80
```

We note that it first computes $1 - 3 = -2$ and sets the Zero Flag to 0 (because -2 is not 0), and sets the Sign Flag to 1 (since -2 has the sign bit set). We note that je jumps only when the zero flag is set, in this case it is not set, and thus it will NOT jump to 0x44f80.

```
cmp 0x1, 0x1
je 0x44f80
```

We note that it first computes $1 - 1 = 0$ and sets the Zero Flag to 1 (our result is 0), and sets the Sign Flag to 0 (since 0 has no sign bit set). We note that je jumps only when the zero flag is set, in this case it is set, and thus it will jump to 0x44f80.

2.2 Test Example

`test x, y` calculates `y & x` and sets the flag according to the result of `y & x`.

```
mov 0x1, %eax
test %eax, %eax
je 0x44f80
```

We note that it first computes 1&1 = 1 and sets the Zero Flag to 0 (because 1 is not 0), and sets the Sign Flag to 0 (since 1 does not have the sign bit set). We note that je jumps only when the zero flag is set, in this case it is not set, and thus it will NOT jump to 0x44f80.

```
mov 0x0, %eax
test %eax, %eax
je 0x44f80
```

We note that it first computes `0 & 0 = 0` and sets the Zero Flag to 1 (our result is 0), and sets the Sign Flag to 0 (since 0 has no sign bit set). We note that je jumps only when the zero flag is set, in this case it is set, and thus it will jump to 0x44f80.

3 Appendix

For a list of flags that jumps checks, please refer to the recitation slides, or here is an abbreviated list (referenced from recitation slides):

```
Carry (CF) - Arithmetic carry/ borrow
Parity (PF) - Odd or even number of bits set
Zero (ZF) - Result was zero
Sign (SF) - Most significant bit was set
Overflow (OF) - Result does not fit into the location

jmp *Operand Jump to specified locations
je/ jz Label Jump if equal/ zero (ZF)
jne/ jnz Label Jump if not equal/ nonzero (~ZF)
js Label Jump if negative (SF)
jns Label Jump if nonnegative (~SF)
jg/ jnle Label Jump if greater (signed) (~(SF ^ OF) & ~ZF)
jge/ jnl Label Jump if greater or equal (signed) (~(SF ^ OF))
jl/ jnge Label Jump if less (signed) (SF ^ OF)
jle/ jng Label Jump if less or equal (signed) ((SF ^ OF) | ZF)
ja/ jnbe Label Jump if above (unsigned) (~CF & ~ZF)
jae/ jnb Label Jump if above or equal (unsigned) (~CF)
jb/ jnae Label Jump if below (unsigned) (CF)
jbe/ jna label Jump if below or equal (unsigned) (CF | ZF)
```