

## Homework 3

Assume we have following address binding table and value of registers :

| Address | Value | Register | Value |
|---------|-------|----------|-------|
| 0x100   | 0x10  | %eax     | 0x10  |
| 0x110   | 0x11  | %ebx     | 0x100 |
| 0x120   | 0x12  |          |       |
| .....   | ..... |          |       |
| 0x190   | 0x19  |          |       |
| 0x200   | 0x20  |          |       |

## Addressing

Please fill in the table below

| Operand         | Value |
|-----------------|-------|
| %ebx            | 0x100 |
| \$0x150         | 0x150 |
| 0x170           | 0x17  |
| (%ebx)          | 0x10  |
| (%ebx,%eax)     | 0x11  |
| 0x30(%ebx)      | 0x13  |
| 80(%ebx,%eax,2) | 0x17  |

## Instructions

Suppose registers and bound values will be reset as above after each instruction. Please fill in the table below: (Write all if there are more than one destinations and None if there is no destination)

| Instruction           | Destination | Value |
|-----------------------|-------------|-------|
| addl %eax,%ebx        | %ebx        | 0x110 |
| subl %eax,(%ebx)      | 0x100       | 0     |
| leal 0x50(%eax), %edx | %edx        | 0x60  |
| movzbl %al, %ebx      | %ebx        | 0x10  |
| movsbl %bh, %ecx      | %ecx        | 0x01  |

## Flags

Assume the initial value of the flags is 0. Fill the table below

| Instruction     | OF | SF | ZF | CF |
|-----------------|----|----|----|----|
| leal(%eax),%ebx | 0  | 0  | 0  | 0  |
| subl %ebx, %eax | 0  | 1  | 0  | 1  |
| xorl %eax, %eax | 0  | 0  | 1  | 0  |
| test %eax, %ebx | 0  | 0  | 1  | 0  |

## Assembly

Translate the following assembly into C codes. You can name local variables represented by `-12(%ebp)`, `-8(%ebp)`...or `a,b,c`... freely as you like. The beginning of C codes is given.

```
push    %ebp
movl    %esp,%ebp
subl    $0x10, %esp
movl    $0x3,-0xc(%ebp)
movl    $0x2,-0x8(%ebp)
movl    $0x1,-0x4(%ebp)
jmp     .L1
.L2:
movl    -0x4(%ebp),%eax
movl    %eax,-0x10(%ebp)
movl    -0x8(%ebp),%eax
movl    %eax,-0x4(%ebp)
movl    -0x10(%ebp),%eax
addl    %eax,-0x8(%ebp)
addl    $0x1,-0xc(%ebp)
.L1:
cmpl    $0x5,-0xc(%ebp)
jle     .L2
movl    -0x8(%ebp), %eax
leave
ret
```

```
int -0xc(%ebp) = 3;           int i = 3;
int -0x8(%ebp) = 2;          or  int b = 2;
.....
```

The order of local variables is not important.

```
int a = 1;
int b = 2;
int i;
for(i=3; i<= 5; i++) {
    int tmp = a;
    a = b;
    b += tmp;
}

return b;
```