Condition codes: new bits in hidden %eflags register.

Some instructions set those bits based on comparisons:
  cmp, test
Other instructions change control flow (%eip) based on results:
  jmp family

INSTRUCTION: **cmpl B, A**

  computes A-B (but doesn't put result anywhere)

condition codes (incomplete):
  zero flag   : ZF=1 if (A-B) == 0 otherwise ZF=0
  signed flag : SF=1 if (A-B) < 0   otherwise SF=0

a-b>0

INSTRUCTION: **jmp** TARGET always changes %eip to TARGET

INSTRUCTION: **je** TARGET %eip=TARGET if ZF==1

INSTRUCTION: **jne** TARGET %eip=TARGET if ZF== 0

INSTRUCTION: **jg** TARGET %eip=TARGET if ~(SF^OF)

INSTRUCTION: **jge** TARGET %eip=TARGET if ~(SF^OF) | ZF

	SF	OF
	1	0
	0	1

a<b
a-b=-1

INSTRUCTION: **jl** TARGET %eip=TARGET if (a-b)<0 + no overflow } = SF^OF
(a-b)>0 + ∴overflow }

INSTRUCTION: **jle** TARGET %eip=TARGET if SF^OF | ZF

Problem #6

Assume value of a is in %eax, and value of b is in %ebx

Write x86 assembly code for:

```
if (a > b) {
    a++;
}
```

```
cmpl   %ebx , %eax
jle    DONT
inc    %eax   //   addl   $1   %eax
DONT:
```

Problem #7

Assume value of a is in %eax, and value of b is in %ebx

Write x86 assembly code for:

```
if (a > b) {
    a++;
} else {
    b = a;
}
```

```
    cmpl   %ebx , %eax   //  a - b ?
    jle    DONT
DO:
    addl   $1,  %eax
    jmp    END
DONT:
    movl   %eax, %ebx
END:
```

Problem #8

Assume value of a is in %eax, and value of b is in %ebx

Write x86 assembly code for:

```
while (b > 0) {
    a++;
    b--;
}
```

```
TOP:
    testl   %ebx, %ebx
    jle     BOT
    incl    %eax
    decl    %ebx
    jmp     TOP
BOT:
```