# y86ccmov.ys

<table>
<thead>
<tr>
<th>.pos 0x0</th>
<th>destination</th>
<th>flags*</th>
</tr>
</thead>
<tbody>
<tr>
<td>register</td>
<td>value</td>
<td>Z</td>
</tr>
<tr>
<td>irmovl</td>
<td>$1, %eax</td>
<td>%eax =</td>
</tr>
<tr>
<td>cmove</td>
<td>%eax,%ecx</td>
<td>%ecx =</td>
</tr>
<tr>
<td>irmovl</td>
<td>0, %ebx</td>
<td>%ebx =</td>
</tr>
<tr>
<td>addl</td>
<td>%eax, %eax</td>
<td>%eax =</td>
</tr>
<tr>
<td>cmovg</td>
<td>%eax, %ebx</td>
<td>%ebx =</td>
</tr>
<tr>
<td>andl</td>
<td>%ebx, %ebx</td>
<td>%ebx =</td>
</tr>
<tr>
<td>subl</td>
<td>%eax, %ecx</td>
<td>%ecx =</td>
</tr>
<tr>
<td>cmovg</td>
<td>%ecx, %edx</td>
<td>%edx =</td>
</tr>
<tr>
<td>irmovl</td>
<td>$0x7fffffff, %edx</td>
<td>%edx =</td>
</tr>
<tr>
<td>addl</td>
<td>%edx, %edx</td>
<td>%edx =</td>
</tr>
<tr>
<td>halt</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* remember, only OP instructions change the CC-bits

irmovl $128,%edx
irmovl $3,%ecx
rmmovl %ecx, 0(%edx)  # write to memory
irmovl $10,%ebx
mrmmovl 0(%edx), %eax  # read from memory
addl %ebx,%eax
halt

Changes to registers:
%eax: 0x00000000 0x000000000000003 0x00000000000000d
%ecx: 0x00000000 0x000000000000003
%edx: 0x00000000 0x000000000000080
%ebx: 0x00000000 0x00000000a

Changes to memory:
0x0080:0x00000000 0x00000003

What is the encoded (in hex) value of the first instruction given above? ➔ irmovl $128,%edx

0x 30 F2 80 00 00 00
1. Given the following code, answer the questions:

```
main:
    .pos 0x0
    irmovl $9, %edx
    6 ↦ offset/PC value = 0x0
    irmovl $21, %ebx
    6 ↦ offset/PC value = 0x6
    subl %edx, %ebx
    2 ↦ offset/PC value = 0xc
    irmovl $128, %esp
    6 ↦ offset/PC value = 0xe
    rmovl %esp, 100(%ebx)
    6 ↦ offset/PC value = 0x14
    pushl %edx
    2 ↦ offset/PC value = 0x1a
    popl %eax
    2 ↦ offset/PC value = 0x1c
    je done
    5 ↦ offset/PC value = 0x1e
    call proc
    5 ↦ offset/PC value = 0x23
done:
    halt
    1 ↦ offset/PC value = 0x28
proc:
    ret
    1 ↦ offset/PC value = 0x29
```

Stack (if needed) could start at 0x2a + room to manage stack info

What is the instruction encoding for the irmov $21, %ebx statement? _____________________0x30f315000000
What is the instruction encoding for the popl %eax statement? _________________________0xb00f
What is the value of %ebx after the program is executed? _____________________________0xc

2. Given the following C code, write the Y86 code:

```c
void main()
{
    // adds the numbers from 1 to 1000
    int sum = 0;
    int lim = 1000;
    for (num=1; num <= lim; num++)
    {
        sum += num;
    }
}
```

```
irmovl $0,%eax
# sum = 0
irmovl $1,%ebx
# num = 1
irmovl $1000,%ecx
# lim = 1000 (constant)
irmovl $1,%edx
# tmp = 1 (constant)
blue here too with "jl End" instead of jge Loop
Loop:
    addl %ebx,%eax
    # sum += num
    addl %edx,%ebx
    # num++
    rmovl %ecx, %esi
    # temp place for lim
    subl %ebx, %esi
    # lim-num
    jge Loop
    # loop again
End: halt
```

There are multiple ways to do this, however, will the values of sum, num and lim all be the same at the end of the loop? Also, compare with y86loop.ys in slides.