Disassembly of ipwr_for

```
0:  55    push %ebp
1:  89 e5  mov %esp,%ebp
3:  53     push %ebx
4:  8b 4d 08 mov 0x8(%ebp),%ecx
7:  8b 55 0c mov 0xc(%ebp),%edx
a:  b8 01 00 00 00 mov $0x1,%eax
f:  85 d2  test %edx,%edx
11: 74 14  je  27 <ipwr_for+0x27>
13: 89 c3  mov %eax,%ebx
15: 0f af d9 imul %ecx,%ebx
18: f6 c2 01 test $0x1,%dl
1b: 0f 45 c3 cmovne %ebx,%eax
1e: d1 ea  shr %edx
20: 74 05  je  27 <ipwr_for+0x27>
22: 0f af c9 imul %ecx,%ecx
25: eb ec  jmp 13 <ipwr_for+0x13>
27: 5b    pop %ebx
28: 5d    pop %ebp
29: c3    ret
```
### Conditional moves

```c
#define OP __________
int arith(int x) { return x OP 4; }
```

// What operation is OP? Fill in the comments to explain how the code works.
// x is in %edx... for example, what if x = 16? What if x = -8?

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Synonym</th>
<th>Move condition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmove</td>
<td>S, D</td>
<td>cmovz</td>
<td>ZF</td>
</tr>
<tr>
<td>cmovne</td>
<td>S, D</td>
<td>cmovnz</td>
<td>~ZF</td>
</tr>
<tr>
<td>cmovs</td>
<td>S, D</td>
<td></td>
<td>SF</td>
</tr>
<tr>
<td>cmovns</td>
<td>S, D</td>
<td></td>
<td>~SF</td>
</tr>
<tr>
<td>cmovg</td>
<td>S, D</td>
<td>cmovnle</td>
<td>~(SF ^ OF) &amp; ~ZF</td>
</tr>
<tr>
<td>cmovge</td>
<td>S, D</td>
<td>cmovnl</td>
<td>~(SF ^ OF)</td>
</tr>
<tr>
<td>cmovl</td>
<td>S, D</td>
<td>cmovnge</td>
<td>SF ^ OF</td>
</tr>
<tr>
<td>cmovle</td>
<td>S, D</td>
<td>cmovng</td>
<td>(SF ^ OF)</td>
</tr>
<tr>
<td>cmova</td>
<td>S, D</td>
<td>cmovnbe</td>
<td>~CF &amp; ~ZF</td>
</tr>
<tr>
<td>cmovae</td>
<td>S, D</td>
<td>cmovnb</td>
<td>~CF</td>
</tr>
<tr>
<td>cmovb</td>
<td>S, D</td>
<td>cmovnae</td>
<td>CF</td>
</tr>
<tr>
<td>cmovbe</td>
<td>S, D</td>
<td>cmovna</td>
<td>CF</td>
</tr>
</tbody>
</table>

**Answer:**
Divide is the OP
Add 3 because:
If x is negative,
it requires
biasing in order
to divide by 4
i.e.
$2^{k-1} = 3$
Since and $k = 2$
Switch Statements

Implementation options

- Series of conditionals
  - Good in few cases
  - Slow if many

- Jump table
  - Lookup branch target
  - Avoids conditionals
  - Possible when cases are small integer constants

- GCC
  - Picks one based on case structure

Usually should also specify “default:” case

```c
typedef enum
{ADD, MULT, MINUS, DIV, MOD, BAD}
op_type;

char unparse_symbol(op_type op)
{
    switch (op) {
    case ADD :
        return '+';
    case MULT:
        return '*';
    case MINUS:
        return '-';
    case DIV:
        return '/';
    case MOD:
        return '%';
    case BAD:
        return '?';
    }
}
```

switchasm.c
Jump table structure

**FYI:** Direct jump is an encoded target as part of the instruction

**Indirect jump** $\rightarrow$ *operand

- Operand is typically a register
  - *%eax* where reg is the target value; OR
  - *(%eax)* where jump target is read from memory

**Jump Table:**

An array where entry $i$ is the address of a code segment implementing the action the program should take when the switch index equals $i$.

Lookup branch target

Avoids conditionals

Possible when cases are small integer constants
Switch statement example

Symbolic Labels
- Labels of form `.LXX` translated into addresses by assembler

Table Structure
- Each target requires 4 bytes
- Base address at `.L57`

Jumping
- jmp `.L49`
- Jump target is denoted by label `.L49`
- jmp `*.L57(,%eax,4)`
- Start of jump table denoted by label `.L57`
- Register `%eax` holds `op`
- Must scale by factor of 4 to get offset into table
- Fetch target from effective Address `.L57 + op*4`

Branching Possibilities
```c
typedef enum
{ADD, MULT, MINUS, DIV, MOD, BAD}
  op_type;

char unparse_symbol(op_type op)
{
  switch (op) {
    ...
  }
}
```

Enumerated Values
- ADD 0
- MULT 1
- MINUS 2
- DIV 3
- MOD 4
- BAD 5
Sparse “switch” example

```c
/* Return x/111 if x is multiple
   && <= 999. -1 otherwise */
int div111(int x)
{
    switch(x) {
    case 0: return 0;
    case 111: return 1;
    case 222: return 2;
    case 333: return 3;
    case 444: return 4;
    case 555: return 5;
    case 666: return 6;
    case 777: return 7;
    case 888: return 8;
    case 999: return 9;
    default: return -1;
    }
}
```

Not practical to use
jump table
Would require 1000 entries
Obvious translation into
if-then-else would have
max. of 9 tests