Assignment 1: Instruction opcodes

The following table gives the full set of instructions for our simulator, with corresponding opcodes:

Assembly Name	Instruction	Opcode
subl	subl	0
addl	addl_reg_reg	1
addl	addl_imm_reg	2
imull	imull	3
shrl	shrl	4
movl	movl_reg_reg	5
movl	movl_deref_reg	6
movl	movl_reg_deref	7
movl	movl_imm_reg	8
cmpl	cmpl	9
je	je	10
jl	jl	11
jle	jle	12
jge	jge	13
jbe	jbe	14
jmp	jmp	15
call	call	16
ret	ret	17
pushl	pushl	18
popl	popl	19
printr	printr	20
readr	readr	21

Notice that two assembly instructions have multiple different forms, *addl* and *movl*. The various forms of these instructions have the same name in the assembly file, but they have different opcodes based on the inputs/outputs they use.

For example, this instruction moves a register into a register (opcode 5):

```
movl    %eax, %ebx
```

This instruction moves an immediate into a register (opcode 8):

```
movl    $1, %eax
```

This instruction moves a register into memory at the address pointed to by %eax + 0 (opcode 7):

```
movl    %ebx, 0(%eax)
```

Notice that not all instructions use two registers and not all instructions use an immediate. For such instructions, the unused components have a value of 0. Thus, all instructions can be decoded using the same logic, and you do not need to write any special cases based on the opcode — simply extract the bits into the associated fields in an *instruction_t* struct.