Arithmetic: R-type

000000	01000	01001	01010	00000	100000
b_{31-26}	b_{25-21}	b_{20-16}	b_{15-11}	b_{10-6}	b_{5-0}
opcode	\$rs	\$rt	\$rd	shamt	function

`add, sub`

```
add $rd, $rs, $rt    # R[d] <- R[s] + R[t]
sub $rd, $rs, $rt    # R[d] <- R[s] - R[t]
```

R-type: 3 registers

add or subtract 2C values

note that there is nothing to indicate whether the registers used
actually contain values of the proper type
unlike high-level languages, there is no type;
the bitstrings are simply used as 2C

opcode: 0

function add: 32
sub: 34

`addu, subu`

Unsigned add or subtract

Ignore overflow

opcode: 0

function add: 33
sub: 35

Arithmetic: I-type

001000	01000	01001	01010 00000 100001
b_{31-26}	b_{25-21}	b_{20-16}	b_{15-0}
opcode	$rs	$rt	immediate

addi

```
addi $rt, $rs, immed    # R[t] <- R[s] + immed
```

I-type

add value given in the instruction to contents of a register

how many bits in the value?

sign bit extended

note $rt is destination

what about subi?

opcode: 8

addiu

unsigned add, without overflow

opcode: 9

How would we increment the value of a register by a constant?

Arithmetic: summary

	R-type	I-type
Add unsigned	addu	addiu
Add signed	add	addi
Subtract unsigned	subu	
Subtract signed	sub	

Also multiply and divide (later)