$100,000 1100 $101 1100

$80 0000 0000 1.1111 1.1111

and $151, $40, $50

$51 = $40 AND $50
$30,000,000$ 1100 1011 1100

$80,000,000,000$

OR: $\pm 1, \pm 10, \pm 80$

OR: $\pm 1, \pm 10, 0 \times 0000$
add $80, $0, 4 \times 80 = 0 + 4
add $51, $0, 1 \times 51 = 0 + 1
511 \times 51, \times 51, 2 \times 51 = 51 \times 4
beg $80, $51, target
add $51, $51, 1
sub $51, $51, $80
label $51, $81, $50
label:
# Python

```python
int sum = 0
for (i = 1; i < 257; i = i + 8)
    sum = sum + i
```

---

```assembly
.data
    sum: .word 8
    i: .word 7

.data for loop
    la $50, sum
    lw $50, 0($50)
    addi $50, $0, 0
    # i -> $51
    li $51, 1
    li $t0, 257
loop: bge $51, $t0, done
    add $50, $50, $51
    sll $51, $51, 1
    # loop
done: # after for loop
    la $t0, sum
    ssw $50, 0($t0)
    # saved sum
```

- For each number `i` from 1 to 257, add it to the sum.
- The loop continues as long as `i` is less than 257.
- The sum is saved in memory using the `sw` instruction.

---

September 20, 2017
while

int pow=1;  int n=0;
while (pow != (128)) {
    pow = pow * 2;
    n = n + 1;
}

# $50 <= pow   n >= $51
addi $50, $0, 1   # $50, 1
move $51, $0   # $51 <- t0
$1 $40, 128

loop: beq $50, $40, done
sll $60, $60, 1
addi $51, $51, 1
done: j loop
Switch (amount)

Case 20:
  fee = 2
  break
Case 50:
  fee = 3
  break
default:
  fee = 7

amount = $50  fee = $6
li $40, 20
bne $50, $40, case50
li $51, 2
j done

Case 50:
li $40, 50
bne $50, $40, case50
li $51, 3
j done

default:
li $51, 7
done