Module 5

2.20 srl $t2, $t0, 10 # move bits 16-11 down to the very right, Bits we want at the start)
sll $t2, $t2, 26 # shift $t2 26 bits to the left, bits 0-25 will be zeros
sll $t1, $t1, 6 # cut off $t1 bits 26-31
srl $t1, $t1, 6 # shift $t1 6 bits to the right, so last 6 bits are zeros
add $t1, $t1, $t2 # replaces the zeroes in $t1 in bits 26-31 with the 6 bits in $t2

2.25.1 I format to hold an address and two registers for comparison.

2.25.2
```
LOOP: slt  $t1, $zero, $t2  # if 0 < $t2, $t1 = 1 else $t1 = 0
      beq  $t3, $zero, EXIT
      addi $t1, $t1, -1     # decrement, no subi

      j LOOP  # if R[rs] > 0, go to LOOP

EXIT:
```

$s2 = 20

Loop	$t1	$s2
1	9	2
2	8	4
3	7	6
4	6	8
5	5	10
6	4	12
7	3	14
8	2	16
9	1	18
10	0	20
11	fail, go to DONE	

2.26.1

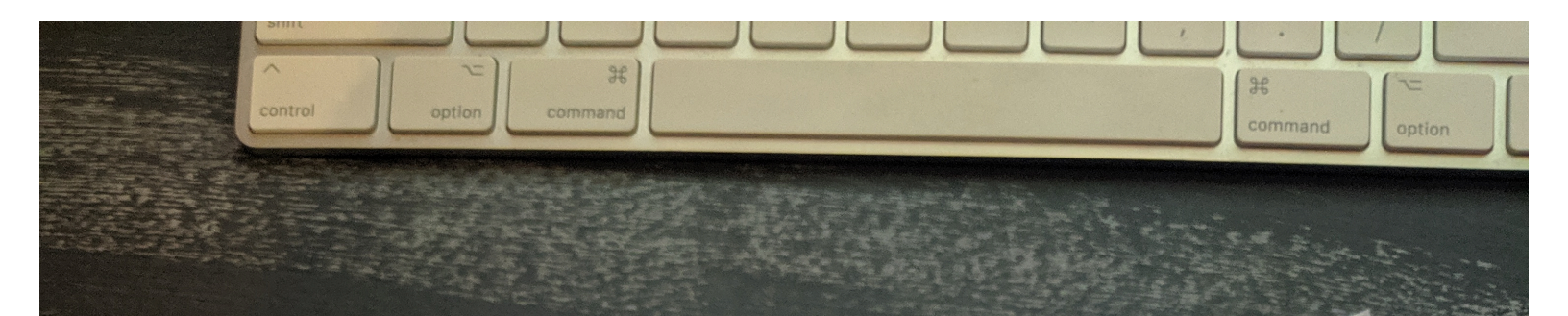
2.26.2
```
while (i > 0) {
    i = i - 1
    B = B + 2
}
```

2.26.3 if N = 0 = $t1 then only 2 instructions are executed.

```
LOOP slt $t2, $0, $t1   <- first
     beq $t2, $0, DONE  <- second.
```

If N = 1, 7 instructions
N = 2, 12.

So Number of instructions = ($t1 or N · 5) + 2



Additional

1) abs $s4, $s1 # absolute value

```
bgez   $s1, LOAD
nor    $s4, $s1, $zero   # NOR will invert $s1
addi   $s4, $s4, 1
LOAD : add  $s4, $s1, $zero
```

useful instructions

bgez : A-GO, branch on greater than equal zero
if rs >= 0, branch

nor : rd, rs, rt
put logical NOR into rd.

basically find the two's complement

2) rol $t7, $t3, 8 , rotate left

```
sll $t0, $t3, 8
srl $t1, $t3, 24
or  $t7, $t0, $t1
```

rol: bits off the left and move onto the right
so rol of 2 of 011010 → 101001

31-23 are shifted out
31-23 become 7-0 and the new 31-7 are all zeros

3) ld $t5, 0($t8)

ld rdest, address
ld : load 64 bit quantity at address in rdest and rdest+1

lui : load upper immediate
loads into upper 16 bits
ori : insert into lower 16 bits

```
sll $t6, $t8, 32
slr $t6, $t7, 32
lw  $t1, 0($t6)
slr $t3, $t8, 32
lw  $t0, 0($t3)
```

get lower 32 bits of $t8 and store in $t1

get upper 32 bits of $t8 and store in $t0

```
lw $t5, 0 ($t0)
lw $t6, 0 ($t1)
```

$t0 has the upper immediate

$t1 has the lower immediate