- Chapter 2：2.19, 2.26, 2.31
- Due on Mar. 19

11713020

张佳晨

2.19 Assume the following register contents:

```
$t0 = 0xAAAAAAAA, $t1 = 0x12345678
```

2.19.1 [5] <§2.6> For the register values shown above, what is the value of $t2 for the following sequence of instructions?

```
sll $t2, $t0, 4
or  $t2, $t2, $t1
```

sll $t2, $t0, 4   # $t2 = $t0 << 4 = 0xAAAAAAA0

or $t2, $t2, $t1   # $t2 = $t2 | $t1 = 0xAAAAAAA0 | 0x12345678

$t2 = 0xBABEFEF8

2.19.2 [5] <§2.6> For the register values shown above, what is the value of $t2 for the following sequence of instructions?

```
sll  $t2, $t0, 4
andi $t2, $t2, -1
```

sll $t2, $t0, 4   # $t2 = $t0 << 4 = 0xAAAAAAA0

addi $t2, $t2, -1   # $t2 = $t2 & 0xFFFFFFFF =

$t2 = 0XAAAAAAA0

2.19.3 [5] <§2.6> For the register values shown above, what is the value of $t2 for the following sequence of instructions?

```
srl  $t2, $t0, 3
andi $t2, $t2, 0xFFEF
```

srl $t2, $t0, 3   # $t2 = $t0 >> 3 = 0x15555555

andi $t2, $t2, 0xFFEF   # $t2 = $t2 & 0xFFEF =

$t2 = 0x00005545

2.26 Consider the following MIPS loop:

```
LOOP: slt  $t2, $0,  $t1
      beq  $t2, $0,  DONE
      subi $t1, $t1, 1
      addi $s2, $s2, 2
      j    LOOP
DONE:
```

2.26.1 [5] <§2.7> Assume that the register $t1 is initialized to the value 10. What is the value in register $s2 assuming $s2 is initially zero?

```
LOOP: slt  $t2, $0,  $t1      if($t1 > 0)
      beq  $t2, $0,  DONE
      subi $t1, $t1, 1        # $t1--
      addi $s2, $s2, 2        # $s2 = $s2+2
      j    LOOP
DONE:
```

So the final value in $s2 will be 20.

2.26.2 [5] <§2.7> For each of the loops above, write the equivalent C code routine. Assume that the registers $s1, $s2, $t1, and $t2 are integers A, B, i, and temp, respectively.

```
LOOP: slt  $t2, $0,  $t1
      beq  $t2, $0,  DONE
      subi $t1, $t1, 1
      addi $s2, $s2, 2
      j    LOOP
DONE:
```

```
while (true){
  temp = 0<i ? 1 : 0;
  if (temp == 0) break;
  i = i-1
  B = B+2
}.
```

2.26.3 [5] <§2.7> For the loops written in MIPS assembly above, assume that the register $t1 is initialized to the value N. How many MIPS instructions are executed?

when $t1 = 0, the code needs to run 2 instructions.

when $t1 > 0, for each loop, there are 5 instructions need to be executed.

So the summation of the MIPS instructions needing to be executed is $(5N+2)$.

2.31 [5] <§2.8> Implement the following C code in MIPS assembly. What is the total number of MIPS instructions needed to execute the function?

```
int fib(int n){
   if (n==0)
      return 0;
   else if (n == 1)
      return 1;
   else
      return fib(n-1) + fib(n-2);
```

```
.text
.global main
main:
li      $a0, 6       # $a0 = 5
jal     fib
add     $a0, $v0, $zero
li      $v0,    1
syscall
li      $v0,    10
syscall

fib:
    addi    $sp, $sp, -8        # malloc the space of 2 word    $sp = $sp + -8
    sw      $ra, 4($sp)         # save the return address onto the stack
    sw      $a0, 0($sp)         # save the argument n onto the stack

    slti    $t0, $a0, 2                 # test for n < 2
    beq     $t0, $zero, LargeThan_1      # if n >= 2 then goto L1
    addi    $sp, $sp, 8
    # n <= 1
    slti    $t0, $a0, 1                 # test for n < 1
    beq     $t0, $zero, Equal_1     # if n >= 1 then goto L1
    # n = 0
        addi    $v0, $v0, 0     # return 0
        jr      $ra                 # return to the callerl
    Equal_1:
        addi    $v0, $v0, 1     # return 1
        jr      $ra                 # return to the callerl
    LargeThan_1: # n >= 2
            # malloc the space of 2 word    $sp = $sp + -8
        addi    $a0, $a0, -1            # $a0 = n - 1
        addi    $sp, $sp, -4
        sw      $a0, 0($sp)

        jal     fib                 # jump to fact and save position to $ra

        lw      $a0, 0($sp)         # $a0 = n - 1

        addi    $a0, $a0, -1            # $a0 = n - 1
        addi    $sp, $sp,  4
        jal     fib             # jump to fact and save position to $ra

        add    $v1, $v0, $v1      # $v1 = fib(n-2)
        lw      $a0, 0($sp)
        lw      $ra, 4($sp)             # restore the return address
        addi    $sp, $sp, 8
        jr      $ra                     # return to the caller
```

Using instruction counter,

use count(n) to represent the number of instruction needed to execute for fib(n).

I got these data:

count(0) = 17

count(1) = 17

count(2) = 45

count(3) = 73

count(4) = 129

count(5) = 213

⋮

$$\text{count}(n) = \begin{cases} 17, & n=0 \text{ or } n=1 \\ 11 + \text{count}(n-2) + \text{count}(n-1), & n \geq 2. \end{cases}$$