HomeWork_3

Name： 卫焱滨（Wei Yanbin）

SID：11710823

chapter 2

Exercise 2.19

$t0 = 0xAAAAAAAA, $t1 = 0x12345678

2.19.1

> The first statement "sll $t2, $t0, 4" , the value in $t2 is $t0 shifted left by 4 bits,1bytes.
>
> $t2 = 0xAAAAAAAA << 4 = 0xAAAAAAA0
>
> After first statement, $t2 = 0xAAAAAAA0
>
> The second statement "or $t2, $t2, $t1" ,compute $t0=0xAAAAAAA0 or $t1 = 0x12345678, the answer is :
>
> 0xAAAAAAA0 | 0x12345678
>
> = 10101010101010101010101010100000 | 00010010000110100010101100111000
>
> = 10111010101111101111111011111000
>
> = 0xBABEFEF8
>
> After second statement, $t2 = 0xBABEFEF8

2.19.2

> The first statement "sll $t2, $t0, 4" , the value in $t2 is $t0 shifted left by 4 bits,1bytes.
>
> $t2 = 0xAAAAAAAA << 4 = 0xAAAAAAA0

After first statement, $t2 = 0xAAAAAAA0

The second statement "andi $t2, $t2, −1" ,compute $t0=0xAAAAAAA0 and 0xFFFFFFFF(-1), the answer is :

0xAAAAAAA0 | 0xFFFFFFFF

= 10101010101010101010101010100000 & 11111111111111111111111111111111

= 10101010101010101010101010100000

= 0xAAAAAAA0

After second statement, $t2 = 0xAAAAAAA0

2.19.3

The first statement "srl $t2 $t0 3" , the value in $t2 is $t0 shifted right by 3 bits,1bytes.

$t2 = 0xAAAAAAAA >> 3 = 0x15555555

After first statement, $t2 = 0x15555555

The second statement "andi $t2, $t2, 0xFFEF" ,compute $t0=0x15555555 and 0xFFEF, the answer is :

0x15555555 | 0xFFEF

= 00010101010101010101010101010101 & 00000000000000001111111111101111

= 00000000000000000101010101000101

= 0x00005545

After second statement, $t2 = 0x00005545

Exercise 2.26

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

经分析，该代码的含义为若$t1<0, 则结束循环，否则，每次循环执行两次操作，$t1 = $t1 -1, $s2 = $s2 + 2直到$t1<0.

2.26.1

$t1 = 10 $s2 = 0

> $s2 = 0 + 2 × (10 - 0) = 20

2.26.2

```
i = 10;
while(i > 0){
    i = i - 1;
    B += 2;
}
```

2.26.3

> By analysis, because $t1 is initialized to the value N, we have N+1 loops totally.
>
> The first N loop has 5 instructions, and final one has 2 instructions
>
> So totally 5N+2 instructions.

Exercise 2.31

To achieve the goal to computing the fibonacci number, we use recuision code by mips as follows:

```
 fib: addi $sp, $sp, -12                  # make room on stack
 sw $ra, 8($sp)                          # push $ra
 sw $s0, 4($sp)                          # push $s0
 sw $a0, 0($sp)                          # push $a0 (N)
 bgt $a0, $zero, equ1                    # if n>0, turn to <equ1> to test (n==1?)
 li $v0, 0                              # else fib(0) = 0
 j return                               # turn to <return>

equ1: li $t0, 1
 bne $a0, $t0, all                       # if n>1, turn to <all>
 li $v0, 1                               # else fib(1) = 1
 j return                               # turn to <return>

all: subi $a0, $a0, 1                    # n-1
 jal fib                                # call fib(n-1)
 addi $s0, $v0, 0                       # add fib(n-1) to answer
 sub $a0, $a0,1                         # n-2
 jal fib                                # call fib(n-2)
 add $v0, $v0, $s0                      # add fib(n-2) to answer

return: lw $a0, 0($sp)                   # pop $a0
 lw $s0, 4($sp)                          # pop $s0
 lw $ra, 8($sp)                          # pop $ra
 addi $sp, $sp, 12                       # restore sp
 jr $ra                                 # return to the position of instruction
ra
```

> Instriuction numbers：

> fib(0)： 12 instructions
>
> fib(1)： 14 instructions
>
> fib(n)： 26 + 18n instructions （n >=2）