HW#3 Conner Wulf

1) Compare:

```
    addi  $sp, $sp, -4
    sw    $ra, 0($sp)
    jal   sub
    li    $t0, 0
    bltz  $v0, exit
    li    $t0, 1
Exit:
    move  $v0, $t0
    lw    $ra, 0($sp)
    addi  $sp, $sp, 4
    jr    $ra
sub:
    sub   $v0 $a0, $a1
    jr    $ra
```

Stack contents

Compare function called: $sp = 0x7ffffff8

return address of compare: 0x7fffffff8

2) fib_ iter:

```
        bne   $a2, $0, else
        move  $v0, $a1
        jr    $ra
else:
    addiu  $sp, $sp, -4
     sw    $ra, 0($sp)
     move  $t0, $a0
     addu  $a0, $a0, $a1
     move  $a1, $t0
     addiu $a2, $a2, -1
     Jal   fib_iter
     Lw    $ra, 0($sp)
   addiu   $sp, $sp, 4
     jr    $ra
```

Stack contents (assume fib_iter is called with n=3

the stack would look like this

n=3	0x7ffffff8
n=2	0x7ffffff4
n=1	0x 7ffffff0
n=0	0x7fffffec

3)a)f: addiu $sp, $sp, -8
sw $ra, 0($sp)
sw $a2, 4($sp)
Jal func
move $a0, $v0
lw $a1, 4($sp)
Jal func
lw $ra, 0($sp)
addiu $sp, $sp, 8
Jr $ra

B) $Ra is equal to the return address and $sp has the same value as $s3 that they had when function f was invoked

4)A)f: addiu $sp, $sp, -12
sw $ra, 0($sp)
sw $a1, 4($sp)
sw $a2, 8($sp)
Jal func
lw $a0 4($sp)
lw $a1 8($sp)
sw $v0 4($sp)
Jal func.
lw $t0, 4($sp)
addu $v0, $t0, $v0
lw $ra, 0($sp)
addiu $sp, $sp, 12
JR $ra

4B) This is the same as 3B above

5) funct_Convert:

```
        li $t6, 0x[redacted]30
        Li $t7, 0x[redacted]39
        li $V0 , 0
       move $t0 , $a0
        lb $t1 , ($t0)
Loop:
        blt $t1 , $t6, set
        bgt $t1 , $t7 , set
       subu $t1 , $t1 , $t6
       mul $v0, $v0 , 10
       add $ v0 , $ v0 , $t1
       addiu $t0, $t0 , 1
        lb $t 1 , ($t0)
       bne $t1 , $0 , Loop
        jr $ra
set:
        li $v0 , -1
        jr $ra
```

6) Hexi_func:

```
        li   $t4  0x[redacted] 41
        Li   $t5  0x[redacted] 46
        Li   $t6  0x[redacted] 30
        Li   $t7, 0x[redacted] 39
        li   $v0, 0
        move $t0, $a0
        lb   $t1, ($t0)
Loop:
        blt $t1 , $t6 , set
        bgt $ t1 , $t7, set_Hex
        sub a $t1 , $t1 , $t6
        j    compute
Set Hex:
        blt  $tl , $ t4 set
        bgt  $tl , $ t5  set
        add.u  $t 1 , $t 1 ,  -55
        sll    $v0 , $ v0 , 4
        add    $ v0 , $v 0 , $t 1
        l b    $t1 , ($t0)
        bne  $t1 , $0 , Loop
        Jr    $ra
Set:
        l: $v0, -1
        Jr   $ra
```