Anna Tang
3180300155

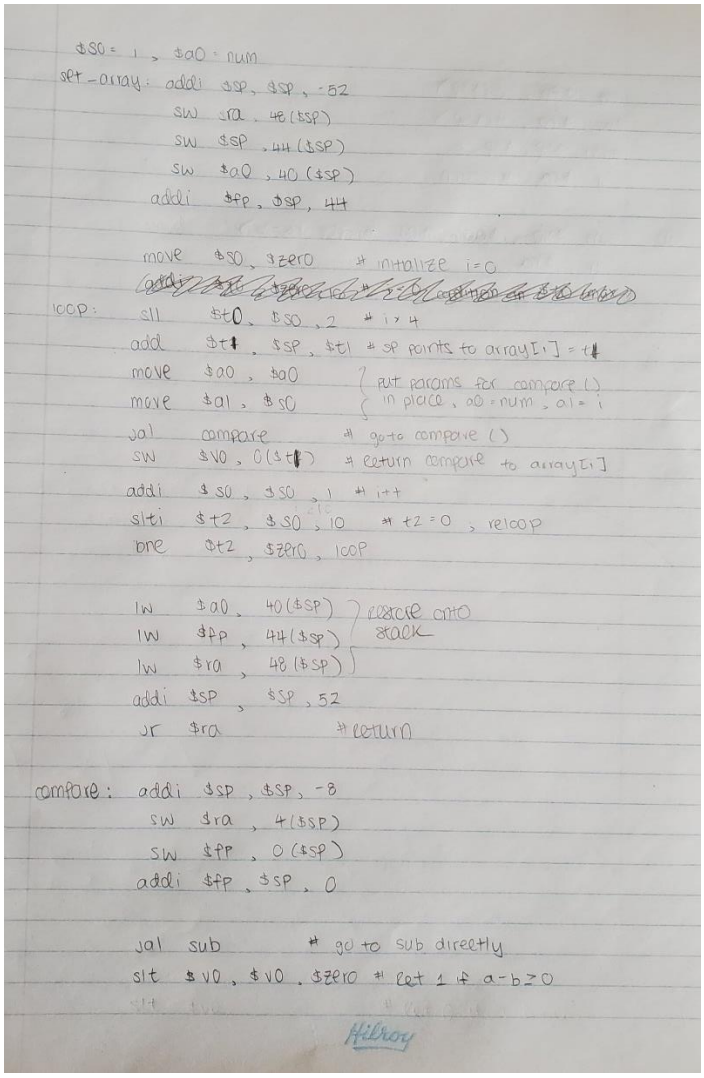
Computer Organization: Homework 2

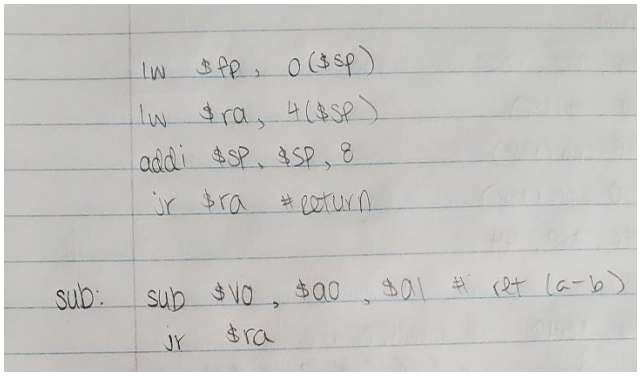
2.4) MIPS doesn't have a subtract immediate instruction because it already has an add immediate instruction (`addi`). One could simply add an negative number and that would suffice, making a `subi` instruction unnecessary.

2.6) To extract an arbitrary field from a 32-bit register from `$t3` to `$t0`, we can use the following instructions:

```
sll  $t0, $t3, 9
srl  $t0, $t0, 15
```

2.15)





2.29)

```
                add   $t0, $zero, $zero     # initialize t0 = 0
loop:           beq   $a1, $zero, finish    # exit loop when a1 = 0
                add   $t0, $t0, $a0         # t0 += a0 ; (a0 x a1)
                sub   $a1, $a1, 1           # a1 -- ; -1 for each iteration
                j     loop                  # reloop
finish:         addi  $t0, $t0, 100         # add 100 to t0 (a0 x a1 +100)
                add   $v0, $t0, $zero       # return t0
```

Using the provided parameters, this program calculates and returns (a0 x a1) + 100.

2.30)

```
                sll   $a2, $a2, 2  # i = 2500 x 4
                sll   $a3, $a3, 2  # j = 2500 x 4
                add   $v0, $zero, $zero  # v0 = 0
                add   $t0, $zero, $zero  # t0 = 0 (i)
outer:          add   $t4, $a0, $t0  # t4 = address of a0[i]

                lw    $t4, 0($t4)  # t4 = a1[i]
                add   $t1, $zero, $zero  # t1 = 0 (j)
inner:          add   $t3, $a1, $t1  # t3 = address of a1[j]
                lw    $t3, 0($t3)  # t3 = a1[j]
                bne   $t3, $t4, skip  # if a1[j] != a0[i], goto skip
                addi  $v0, $v0, 1  # v0 += 1
skip:           addi$ t1, $t1, 4  # j++
                bne   $t1, $a3, inner  # reloop if t1 != a3 (j != 2500 x 4)
                addi  $t0, $t0, 4  # i++
                bne   $t0, $a2, outer  # reloop if t0 != a2 (i != 2500 x 4)
```

This code snippit compares 2 arrays (a0 & a1), and returns the number of matching elements in $v0.

2.32)

b = 25 | a -> `ori  $t1  $t0  25`

2.34)

```
        addi $v0, $zero, 0  # Initialize count
loop:   lw   $v1, 0($a0)    # Read next word from source
        sw   $v1, 0($a1)    # Write to destination
        addi $a0, $a0, 4    # Advance pointer to next source
        addi $a1, $a1, 4    # Advance pointer to next destination

        beq  $v1, $zero, loop  # Loop if word copied != zero
```

-1

bne

addi $v0, $v0, 1 # increment count

2.37)

Pseudoinstructions	Purpose	MIPS Instructions
`move $t1, t2`	`$t1 = $t2`	`add $t1, $t2, $zero`
`clear $t0`	`$t0 = 0`	`add $t0, $zero, $zero`
`beq $t1, small, L`	`if ($t1 = small) go to L`	`li $at, small` `beq $t1, $at, L`
`beq $t2, big, L`	`if ($t2 = big) go to L`	`li $at, big` `beq $t2, $at, L`
`li $t1, small`	`$t1 = small`	`addi $t1, $zero,small`
`li $t2, big`	`$t2 = big`	`addi $t1, $zero, big`
`ble $t3, $t5, L`	`if ($t3 <= $t5) go to L`	`slt $at, $t5, $t3` `bne $at, $zero, L`
`bgt $t4, $t5, L`	`if ($t4 > $t5) go to L`	`slt $at, $t5, $t4` `bne $at, $zero, L`
`bge $t5, $t3, L`	`if ($t5 >= $t3) go to L`	`slt $at, $t5, $t3` `beq $at, $zero, L`
`addi $t0, $t2, big`	`$t0 = $t2 + big`	`li $at, big` `add $t0, $t2, $at`

`lw $t5, big($t2)`	`$t5 = memory[$t2 + big]`	`li $at, big`
		`add $at, $at, $t2`
		`add $t5, $t2, $at`

2.46)

while (save[i] == k) i += 1;

```
      lw   $t2 , 4($s6)   # t2 = size of save[]
loop: slt  $t0 , $s3 , $zero  # if i<0 , t0 = 1
      bne  $t0 , $zero , indexcutofBounds  # if t0 != 0 then error
      slt  $t0 , $t2 , $s3   # if i ≥ size , t0 = 0
      beq  $t0 , $zero , indexcutofBounds  # if t0 = 0 , error
      sll  $t1 , $s3 , 2
      add  $t1 , $t1 , $s6
      lw   $t0 , 0($t1)
      bne  $t0 , $s5 , Exit
      addi $s3 , $s3 , 1
      j loop
indexcutofBounds:
          ...
          ...
      Exit:
```

# of instructions:

(10 x 10) + 1 + 8 = 109

- (10 x 10): 10 iterations
- 1: lw
- 8: 11th run; assume save[i] != k

2.47)

for (i=0; i<=100; i++) a[i] = b[i] + c;

a0 = A s0 = C
a1 = B t0 = i

```
        addi $t0, $t0, $zero   # i=0
loop:   add  $t1, $a0, $t0     # t1 = address of a[i]
        add  $t2, $a1, $t0     # t2 = address of b[i]
        sw   $t2, 0($t1)       # a[i] = b[i]
        add  0($t1), $s0, $zero # a[i] += c
        addi $t0, $t0, 4       # i++
        bne  $t3, $zero, loop  # if i<101, reloop.
```

1 + (100 × 6) + 6 = 607 instructions

1 → addi; 100 → 100 iterations; 6 → if i > 100

# of data references = 101