

## (5 pts) Exercise 2-21

---

- a.) What is the MIPS assembly code for the following:  
do {  
    g = g + j;  
} while (g < h);  
Variables f to j are assigned to registers \$s0 to \$s4  
Use \$v0, \$v1 as temporaries if needed  
b.) Did your solution use any pseudo-instructions?

f \$s0  
g \$s1  
h \$s2  
i \$s3  
j \$s4

## (5 pts) Exercise 2-22

---

- a.) What is the MIPS assembly code for the following:  
do {  
    g = g + j;  
} while (g < 100);  
Variables f to j are assigned to registers \$s0 to \$s4  
Use \$v0, \$v1 as temporaries if needed  
b.) Did your solution use any pseudo-instructions?

f \$s0  
g \$s1  
h \$s2  
i \$s3  
j \$s4

## (5 pts) Exercise 2-23

---

- a.) What is the MIPS assembly code for the following:  
    while (g < i) {  
        g = g + j;  
    }  
    Variables f to j are assigned to registers \$s0 to \$s4  
    Use \$v0, \$v1 as temporaries if needed

f \$s0  
g \$s1  
h \$s2  
i \$s3  
j \$s4

b.) Did your solution use any pseudo-instructions?

## (10 pts) Exercise 2-24

---

- a.) What is the MIPS assembly code for the following:  
    while (g > i) {  
        g = g + 3;  
    }

f \$s0  
g \$s1  
h \$s2  
i \$s3  
j \$s4

Variables f to j are assigned to registers \$s0 to \$s4  
Use \$v0, \$v1 as temporaries if needed

b.) Did your solution use any pseudo-instructions?

## (15 pts) Exercise 2-26

---

- The MIPS translation of the C segment:

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

**is given on page 74 as follows:**

```
Loop:  sll $t1, $s3, 2 # temp reg t1 = 4 *i
      add $t1, $t1, $s6 # t1 = address of save[i]
      lw $t0, 0($t1) # temp reg t0 = save[i]
      bne $t0, $s5, Exit # goto Exit if save[i] != k
      add $s3, $s3, 1 # loop body: i++
      j Loop # repeat the loop
```

- **This code uses both a conditional branch and an unconditional jump each time through the loop. Only poor compilers would produce code with this loop overhead. Rewrite the assembly code so that it uses at most one branch or jump each time through the loop.**

## (20 pts) Exercise 2-27

---

```
        add $t0, $zero, $zero
loop:   beq $a1, $zero, finish
        add $t0, $t0, $a0
        sub $a1, $a1, 1
        j   loop
finish: addi $t0, $t0, 100
        add $v0, $t0, $zero
```

- (10 pts) Add comments to the MIPS code above. Assume that \$a0 and \$a1 are used for the input and both initially contain the integers a and b, respectively.
- (10 pts) In one sentence, what does this code compute (in terms of 'a' and 'b')?

## (20 pts) Exercise 2-28

---

```
        sll $a2, $a2, 2
        sll $a3, $a3, 2
        add $v0, $zero, $zero
        add $t0, $zero, $zero
outer:  add $t4, $a0, $t0
        lw  $t4, 0($t4)
        add $t1, $zero, $zero
inner:  add $t3, $a1, $t1
        lw  $t3, 0($t3)
        bne $t3, $t4, skip
        addi $v0, $v0, 1
skip    addi $t1, $t1, 4
        bne $t1, $a3, inner
        addi $t0, $t0, 4
        bne $t0, $a2, outer
```

- (10 pts) Add comments to the MIPS code above. This code processes two arrays and produces an important value in register \$v0. Assume that each array consists of 2500 words indexed 0 through 2499, that the base addresses of the arrays are stored in \$a0 and \$a1 respectively, and their sizes (2500) are stored in \$a2 and \$a3, respectively. In your comments, call the arrays Array1 and Array2.
- (10 pts) In one sentence, what does this code compute and store in \$v0?