(20 pts) Exercise 2-18: Pseudo-instructions

- **Below you will see several problems of the form:**
  ```
  li $t1, small              # $t1 = small
  ```
- **This is an example pseudo-instruction, with its meaning given as a comment. The instruction should load 'small' into $t1, where 'small' means a constant that fits within 16 bits. The constant is also called an 'immediate' – 'li' stands for 'load immediate'**
- **Your job is to write the real MIPS instruction (or sequence of instructions) that the compiler would produce for each given pseudo-instruction. For instance, the solution for the above pseudo-instruction is:**
  ```
  addi $t1, $zero, small
  ```
- **In the problems, 'small' and 'big' refer to some arbitary constants (numbers). A 'big' constant needs 32 bits to be represented. You will need some notation to talk about the upper (most significant) 16 bits and the lower (least significant) bits of this number. Use UPPER(big) to refer to the most significant bits and LOWER(big) to refer to the other bits.**
- **To make your answers simpler, you may make use of the 'li' psuedo-instruction where helpful (except when defining li itself). Do this – it will make your job easier!!**

- `clear $to          # $t0 = 0`
- `beq $t3, small, L # if ($t3 == small) go to L`
- `beq $t2, big, L   # if ($t2 == big) go to L`
- `li $t2, big       # $t2 = big`
- `bge $t5, $t4, L   # if ($t5 >= $t4) go to L`
- `lw $t0, big($t2)  # t0 = Memory[$t2+big]`

(5 pts) Exercise 2-21

- a.) What is the MIPS assembly code for the following:

```
do {
        g = g + j;
} while (g <h);
```

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? YES or NO

(5 pts) Exercise 2-22

- a.) What is the MIPS assembly code for the following:

```
do {
        g = g + j;
} while (g <100);
```

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? YES or NO

(5 pts) Exercise 2-23

- a.) What is the MIPS assembly code for the following:

```
while (g < i) {
        g = g + j;
}
```

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?   YES or NO

(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?   YES or NO

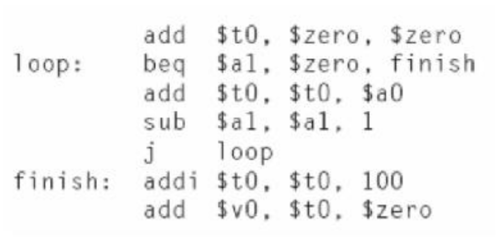
(15 pts) Exercise 2-26

- **The MIPS translation of the C segment:**
  while (save[i] == k) i = i + 1;
  is given on page 92-93 of the textbook as follows:

```
Loop:  sll $t1, $s3, 2  # temp reg t1 = i *4
   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 = i + 1
   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.**
- **You will have to think carefully about how to reorganize things to make this possible. Make sure that your final version still computes the same result as the original code!**

(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. You may assume ‘a’ and ‘b’ are both greater than zero.**
- **(10 pts) <u>In one simple sentence</u>, what does this code compute (in terms of ‘a’ and ‘b’)?**