Chapter 2-3

Exercise1:

[5] <§§2.3, 2.6, 2.9> Add comments to the following MIPS code and describe in one sentence what it computes. Assume that $a0 and $a1 are used for the input and both initially contain the integers *a* and *b*, respectively. Assume that $v0 is used for the output.

```
                  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
```

Exercise 2:

[12] <§§2.3, 2.6, 2.9> The following code fragment 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. Add comments to the code and describe in one sentence what this code does. Specifically, what will be returned in $v0?

```
            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
```

Exercise 2.16

For these problems, the table holds various binary values for register `$t0`. Given the value of `$t0`, you will be asked to evaluate the outcome of different branches.

a.	0010 0100 1001 0010 0100 1001 0010 0100_{two}
b.	0101 1111 1011 1110 0100 0000 0000 0000_{two}

2.16.1 [5] <2.7> Suppose that register `$t0` contains a value from above and `$t1` has the value

0011 1111 1111 1000 0000 0000 0000 0000_{two}

Note the result of executing these instructions on particular registers. What is the value of `$t2` after the following instructions?

```
      slt  $t2, $t0, $t1
      beq  $t2, $0, ELSE
      j    DONE
ELSE: addi $t2, $0, 2
DONE:
```

2.16.2 [5] <2.7> Suppose that register $t0 contains a value from the table above and is compared against the value `X`, as used in the MIPS instruction below. Note the format of the slti instruction. For what values of `X`, if any, will `$t2` be equal to 1?

```
slti $t2, $t0, X
```

2.16.3 [5] <2.7> Suppose the program counter (PC) is set to 0x0000 0020. Is it possible to use the jump MIPS assembly instruction to get set the PC to the address as shown in the data table above? Is it possible to use the branch-on-equal MIPS assembly instruction to get set the PC to the address as shown in the data table above?

For these problems, the table holds various binary values for register `$t0`. Given the value of `$t0`, you will be asked to evaluate the outcome of different branches.

a.	0x00101000
b.	0x80001000

2.16.4 [5] <2.7> Suppose that register `$t0` contains a value from above. What is the value of `$t2` after the following instructions?

```
      slt  $t2, $0,   $t0
      bne  $t2, $0,   ELSE
      j    DONE
ELSE: addi $t2, $t2,  2
DONE:
```

2.16.5 [5] <2.6, 2.7> Suppose that register `$t0` contains a value from above. What is the value of `$t2` after the following instructions?

```
sll $t0, $t0, 2
slt $t2, $t0, $0
```

2.16.6 [5] <2.7> Suppose the program counter (PC) is set to 0x2000 0000. Is it possible to use the jump (j) MIPS assembly instruction to get set the PC to the address as shown in the data table above? Is it possible to use the branch-on-equal (beq) MIPS assembly instruction to set the PC to the address as shown in the data table above? Note the format of the J-type instruction.

Exercise 2.17

For these problems, there are several instructions that are not included in the MIPS instruction set are shown.

a.	`subi $t2, $t3, 5`	`# R[rt] = R[rs] - SignExtImm`
b.	`rpt  $t2, loop`	`# if(R[rs]>0) R[rs]=R[rs]-1, PC=PC+4+BranchAddr`

2.17.1 [5] <2.7> The table above contains some instructions not included in the MIPS instruction set and the description of each instruction. Why are these instructions not included in the MIPS instruction set?

2.17.2 [5] <2.7> The table above contains some instructions not included in the MIPS instruction set and the description of each instruction. If these instructions were to be implemented in the MIPS instruction set, what is the most appropriate instruction format?

2.17.3 [5] <2.7> For each instruction in the table above, find the shortest sequence of MIPS instructions that performs the same operation.

For these problems, the table holds MIPS assembly code fragments. You will be asked to evaluate each of the code fragments, familiarizing you with the different MIPS branch instructions.

a.

```
LOOP:  addi $s2, $s2, 2
       subi $t1, $t1, 1
       bne  $t1, $0,  LOOP
DONE:
```

b.

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

2.17.4 [5] <2.7> For the loops written in MIPS assembly above, assume that the register `$t1` is initialized to the value 10. What is the value in register `$s2` assuming the `$s2` is initially zero?

2.17.5 [5] <2.7> For each of the loops above, write the equivalent C code routine. Assume that the registers `$s1`, `$s2`, `$t1`, and `$t2` are integers `A`, `B`, `i`, and `temp`, respectively.

2.17.6 [5] <2.7> For the loops written in MIPS assembly above, assume that the register `$t1` is initialized to the value N. How many MIPS instructions are executed?

Exercise 2.18

For these problems, the table holds some C code. You will be asked to evaluate these C code statements in MIPS assembly code.

a.	`for(i=0; i<a; i++)`
	`    a += b;`

b.	`for(i-0; i<a; i++)`
	`   for(j=0; j<b; j++)`
	`       D[4*j] = i + j;`

2.18.1 [5] <2.7> For the table above, draw a control-flow graph of the C code.

2.18.2 [5] <2.7> For the table above, translate the C code to MIPS assembly code. Use a minimum number of instructions. Assume that the values of `a`, `b`, `i`, and `j` are in registers `$s0`, `$s1`, `$t0`, and `$t1`, respectively. Also, assume that register `$s2` holds the base address of the array `D`.

2.18.3 [5] <2.7> How many MIPS instructions does it take to implement the C code? If the variables `a` and `b` are initialized to 10 and 1 and all elements of D are initially 0, what is the total number of MIPS instructions that is executed to complete the loop?

For these problems, the table holds MIPS assembly code fragments. You will be asked to evaluate each of the code fragments, familiarizing you with the different MIPS branch instructions.

a.	`      addi $t1, $0, 50`
	`LOOP: lw   $s1, 0($s0)`
	`      add  $s2, $s2, $s1`
	`      lw   $s1, 4($s0)`
	`      add  $s2, $s2, $s1`
	`      addi $s0, $s0, 8`
	`      subi $t1, $t1, 1`
	`      bne  $t1, $0, LOOP`

b.	`      addi $t1, $0, $0`
	`LOOP: lw   $s1, 0($s0)`
	`      add  $s2, $s2, $s1`
	`      addi $s0, $s0, 4`
	`      addi $t1, $t1, 1`
	`      slti $t2, $t1, 100`
	`      bne  $t2, $s0, LOOP`

2.18.4 [5] <2.7> What is the total number of MIPS instructions executed?

2.18.5 [5] <2.7> Translate the loops above into C. Assume that the C-level integer `i` is held in register `$t1`, `$s2` holds the C-level integer called `result`, and `$s0` holds the base address of the integer `MemArray`.

2.18.6 [5] <2.7> Rewrite the loop to reduce the number of MIPS instructions executed.