Exercise 2.6

The following problems deal with translating from C to MIPS. Assume that the variables f, g, h, i, and j are assigned to registers $s0, $s1, $s2, $s3, and $s4, respectively. Assume that the base address of the arrays A and B are in registers $s6 and $s7, respectively.

a.	`f = -g + h + B[1];`
b.	`f = A[B[g]+1];`

2.6.1 [10] <2.2, 2.3> For the C statements above, what is the corresponding MIPS assembly code?

2.6.2 [5] <2.2, 2.3> For the C statements above, how many MIPS assembly instructions are needed to perform the C statement?

2.6.3 [5] <2.2, 2.3> For the C statements above, how many registers are needed to carry out the C statement using MIPS assembly code?

The following problems deal with translating from MIPS to C. Assume that the variables f, g, h, i, and j are assigned to registers $s0, $s1, $s2, $s3, and $s4, respectively. Assume that the base address of the arrays A and B are in registers $s6 and $s7, respectively.

a.	`add $s0, $s0, $s1`
	`add $s0, $s3, $s2`
	`add $s0, $s0, $s3`

b.	`addi $s6, $s6, -20`
	`add  $s6, $s6, $s1`
	`lw   $s0, 8($s6)`

2.6.4 [5] <2.2, 2.3> For the MIPS assembly instructions above, what is the corresponding C statement?

2.6.5 [5] <2.2, 2.3> For the MIPS assembly above, assume that the registers $s0, $s1, $s2, $s3, contain the values 10, 20, 30, and 40, respectively. Also, assume that register $s6 contains the value 256, and that memory contains the following values:

Address	Value
256	100
260	200
264	300

Find the value of $s0 at the end of the assembly code.

2.6.6 [10] <2.3, 2.5> For each MIPS instruction, show the value of the op, rs, and rt fields. For I-type instructions, show the value of the immediate field, and for the R-type instructions, show the value of the rd field.

Exercise 2.12

In the following problems, the data table contains various modifications that could be made to the MIPS instruction set architecture. You will investigate the impact of these changes on the instruction format of the MIPS architecture.

a.	8 registers
b.	10 bit immediate constants

2.12.1 [5] <2.5> If the instruction set of the MIPS processor is modified, the instruction format must also be changed. For each of the suggested changes above, show the size of the bit fields of an R-type format instruction. What is the total number of bits needed for each instruction?

2.12.2 [5] <2.5> If the instruction set of the MIPS processor is modified, the instruction format must also be changed. For each of the suggested changes above, show the size of the bit fields of an I-type format instruction. What is the total number of bits needed for each instruction?

2.12.3 [5] <2.5, 2.10> Why could the suggested change in the table above decrease the size of a MIPS assembly program? Why could the suggested change in the table above increase the size of a MIPS assembly program?

In the following problems, the data table contains hexadecimal values. You will be asked to determine what MIPS instruction the value represents, and find the MIPS instruction format.

a.	0x01090010
b.	0x8D090012

2.12.4 [5] <2.5> For the entries above, what is the value of the number in decimal?

2.12.5 [5] <2.5> For the hexadecimal entries above, what instruction do they represent?

2.12.6 [5] <2.4, 2.5> What type (I-type, R-type) instruction do the binary entries above represent? What is the value of the op field and the rt field?

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<10; i++)
    a += b;
```

b.

```
while (a < 10){
    D[a] = b + a;
    a += 1;
}
```

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 value `a`, `b`, `i`, `j` are in registers `$s0`, `$s1`, `$t0`, `$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, 100
LOOP:   lw    $s1, 0($s0)
        add   $s2, $s2, $s1
        addi  $s0, $s0, 4
        subi  $t1, $t1, 1
        bne   $t1, $0, LOOP
```

b.

```
        addi  $t1, $s0, 400
LOOP:   lw    $s1, 0($s0)
        add   $s2, $s2, $s1
        lw    $s1, 4($s0)
        add   $s2, $s2, $s1
        addi  $s0, $s0, 8
        bne   $t1, $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 in MIPS assembly to reduce the number of MIPS instructions executed.

Exercise 2.24

Assume that the register `$t1` contains the address 0x1000 0000 and the register `$t2` contains the address 0x1000 0010.

a.	`lb $t0, 0($t1)` `sw $t0, 0($t2)`
b.	`lb $t0, 0($t1)` `sb $t0, 0($t2)`

2.24.1 [5] <2.9> Assume that the data (in hexadecimal) at address 0x1000 0000 is:

1000 0000	12	34	56	78

What value is stored at the address pointed to by register `$t2`? Assume that the memory location pointed to `$t2` is initialized to 0xFFFF FFFF.

2.24.2 [5] <2.9> Assume that the data (in hexadecimal) at address 0x1000 0000 is:

1000 0000	80	80	80	80

What value is stored at the address pointed to by register `$t2`? Assume that the memory location pointed to `$t2` is initialized to 0x0000 0000.

2.24.3 [5] <2.9> Assume that the data (in hexadecimal) at address 0x1000 0000 is:

1000 0000	11	00	00	FF

What value is stored at the address pointed to by register `$t2`? Assume that the memory location pointed to `$t2` is initialized to 0x5555 5555.

Exercise 2.27

In the following problems, you will be using exploring different addressing modes in the MIPS instruction set architecture. These different addressing modes are listed in the table below.

a.	Register Addressing
b.	PC-relative Addressing

2.27.1 [5] <2.10> In the table above are different addressing modes of the MIPS instruction set. Give an example MIPS instructions that shows the MIPS addressing mode.

2.27.2 [5] <2.10> For the instructions in 2.27.1, what is the instruction format type used for the given instruction?

2.27.3 [5] <2.10> List benefits and drawbacks of a particular MIPS addressing mode. Write MIPS code that shows these benefits and drawbacks.

In the following problems, you will be using the MIPS assembly code as listed below to explore the tradeoffs of the immediate field in the MIPS I-type instructions.

a.	0x00000000 0x00000004	`lui $s0, 100` `ori $s0, $s0, 40`
b.	0x00000100 0x00000104	`addi $t0, $0, 0x0000` `lw $t1, 0x4000($t0)`

2.27.4 [15] <2.10> For the MIPS statements above, show the bit-level instruction representation of each of the instructions in hexadecimal.

2.27.5 [10] <2.10> By reducing the size of the immediate fields of the I-type and J-type instructions, we can save on the number of bits needed to represent instructions. If the immediate field of I-type instructions were 8 bits and the immediate field of J-type instructions were 18 bits, rewrite the MIPS code above to reflect this change. Avoid using the `lui` instruction.

2.27.6 [5] <2.10> How many extra instructions are needed to execute your code in 2.27.5 MIPS statements in the table versus the code shown in the table above?

Exercise 2.30

Assembler pseudoinstructions are not a part of the MIPS instruction set, but often appear in MIPS programs. The table below contains some MIPS pseudoinstructions that ,when assembled, are translated to other MIPS assembly instructions.

a.	`move $t1, $t2`
b.	`beq $t1, small, LOOP`

2.30.1 [5] <2.12> For each pseudo instruction in the table above, produce a minimal sequence of actual MIPS instructions to accomplish the same thing. You may need to use temporary registers in some cases. In the table `large` refers to a number that requires 32 bits to represent and `small` to a number that can fit into 16 bits.

Exercise 3.5

For many reasons, we would like to design multipliers that require less time. Many different approaches have been taken to accomplish this goal. In the following table, A represents the bit width of an integer, and B represents the number of time units (tu) taken to perform a step of an operation.

	A (bit width)	B (time units)
a.	4	3 tu
b.	32	7 tu

3.5.1 [10] <3.3> Calculate the time necessary to perform a multiply using the approach given in Figures 3.4 and 3.5 if an integer is A bits wide and each step of the operation takes B time units. Assume that in step 1a an addition is always performed—either the multiplicand will be added, or a 0 will be. Also assume that the registers have already been initialized (you are just counting how long it takes to do the multiplication loop itself). If this is being done in hardware, the shifts of the multiplicand and multiplier can be done simultaneously. If this is being done in software, they will have to be done one after the other. Solve for each case.

3.5.2 [10] <3.3> Calculate the time necessary to perform a multiply using the approach described in the text (31 adders stacked vertically) if an integer is A bits wide and an adder takes B time units.

Exercise 3.6

In this exercise we will look at a couple of other ways to improve the performance of multiplication, based primarily on doing more shifts and fewer arithmetic operations. The following table shows pairs of hexadecimal numbers.

	A	B
a.	24	c9
b.	41	18

3.6.1 [20] <3.3> As discussed in the text, one possible performance enhancement is to do a shift and add instead of an actual multiplication. Since 9×6, for example, can be written $(2 \times 2 \times 2 + 1) \times 6$, we can calculate 9×6 by shifting 6 to the left three times and then adding 6 to that result. Show the best way to calculate $A \times B$ using shifts and adds/subtracts. Assume that A and B are 8-bit unsigned integers.

3.6.2 [20] <3.3> Show the best way to calculate $A \times B$ using shift and adds, if A and B are 8-bit signed integers stored in sign-magnitude format.