Chapter 2: Exercises

1. Given the following C program fragment:

```
x = 5;
y = x - 2;
a = x * 4;
b = y * 2;
z = (x + a) - (y + b);
```

Convert it into MIPS assembly code.

2. Given the C assignment statement:

```
x = y + a[5];
```

Represent this assignment in MIPS assembly, and then show the machine code corresponding to the MIPS instructions.

3. Distinguish between big-endian and little-endian memory alignment strategies. Which strategy does MIPS use?

4. Given the following C code fragment:

```
if (x < 0)
   h = 0;
else
   h = x;
```

Write the equivalent MIPS assembly code for the above C program.

5. Describe the basic steps that occur when a procedure (function) in a program is executed.

6. Write a MIPS assembly program for a main program that has four parameters a, b, c, and d stored respectively in registers $s0, $s1, $s2, and $s3. The main program passes these four parameters to a procedure called proc_exa, which computes the result of the expression (a + b) – (c + d) and

returns the result in register $s6. The computation inside the procedure proc_exa uses temporary registers $t1, $t2, and $s0.

7. Write a MIPS assembly program that prints 'Hello world' on the screen.

8. Write a MIPS assembly program that reads a person's name and prints 'Hello <name>' on the screen.

9. Write a MIPS assembly program that takes the length and width of a rectangle as input, computes its perimeter and area, and displays the results on the screen.

10. Write a MIPS assembly program that takes two positive integers n and k as input and computes C(n, k) = n! / (k! * (n-k)!). Display the result on the screen.

11. Write a MIPS assembly program that finds the largest value in an array of 10 integers.

12. Write a MIPS assembly program that computes the average value of the elements in an array of 10 floating-point numbers.

13. Given the following C function:

```
int fact(int n){
   if (n < 1)
      return(1);
   else
      return(n * fact(n - 1));
}
```

Write the corresponding MIPS assembly code for the fact function, knowing that the parameter n is stored in register $a0. The compiler begins the function with its label and saves the registers $ra and $a0 onto the top of the stack.