Name: (as it would appear on official course roster)	
UCSB email address:	**@ucsb.edu**
Lab Section Time:	
Optional: name you wish to be called if different from above	
Optional: name of "homework buddy" (leaving this blank signifies "I worked alone")	

Assignment 02: Performance Measures and More MIPS Assembly Programming Refreshers

Assigned: *Friday, January 17th, 2020*
Due: *Wednesday, January 22nd, 2020*
Points: *70 (normalized to 100 in gradebook)*

- You may collaborate on this homework with AT MOST one person, an optional "homework buddy".
- MAY ONLY BE TURNED IN ON **GRADESCOPE as a PDF file** (see instructions in online lab01 description).
- There is NO MAKEUP for missed assignments.
- We are strict about enforcing the LATE POLICY for all assignments (see syllabus).

Only use the space provided for answers. Use clear and clean handwriting (or typing).
You will get penalized if you are asked to show your calculations and do not do so.
Use scientific notation where needed (e.g. instead of 1234000, use 1.234 x 10^6).

1. (6 pts) Assume a color display using 8 bits for each of the primary colors (red, green, blue) per pixel and a display frame size of 1920 × 1080.
   a. (3 pts) What is the minimum size in bytes of the frame buffer to store a single frame? Show your calculation.

   b. (3 pts) How long would it take, at minimum, for the frame to be sent over a network connection of 1 Gb/second? Show your calculation.

Name:

(as it would appear on official course roster)

2. (8 pts) Consider 3 different processors: P1, P2, and P3 executing the same instruction set. P1 has a 3.3 GHz clock rate and a CPI (cycles per instruction) of 1.5, P2 has a 2.7 GHz clock rate and a CPI of 1.2, and P3 has a 4.0 GHz clock rate and a CPI of 2.2.
   a. (4 pts) Which processor has the highest performance as measured in *instructions-per-second*? Show your calculation.

   b. (4 pts) If the processors each execute a program in 12 seconds, then find the number of cycles and the number of instructions. Show your calculation.

Name:

(as it would appear on official course roster)

3. (10 pts) Assume that a program requires the execution of $50x10^6$ FP (floating point) instructions, $110x10^6$ INT (integer) instructions, $80x10^6$ L/S (load/store) instructions, and $6x10^6$ branching instructions. These types of instructions have CPIs of 1, 1, 4, and 2, respectively. Assume that the processor has a 2 GHz clock rate.
   a. (5 pts) By how much must we improve the CPI of FP instructions if we want the program to run twice as fast? Show your calculations.

   b. (5 pts) By how much (as a percentage) is the execution time of the entire program improved if the CPI of the INT and the FP instructions is reduced by 40% and the CPI of L/S and of the branching instructions is reduced by 30%? Show your calculations.

Name:

(as it would appear on official course roster)

4. (14 pts) Consider two processors, P1 and P2. P1 has a clock rate of 4 GHz and an average CPI of 0.9. P2 has a clock rate of 3 GHz, an average CPI of 0.75. Both P1 and P2 are executing a program with 10^9 instructions.
   a. (7 pts) As discussed in class, a common fallacy is to use MIPS (millions of instructions per second) to compare the performance of two different processors, and to then consider that the processor with the largest MIPS has the largest performance. Check if this is true for P1 and P2 (show your calculations). Section 1.10 in the book may give you some insight.

   b. (7 pts) Another common performance figure is MFLOPS (millions of floating-point operations per second), defined as:
   MFLOPS = No. FP operations / (execution time $\times$ ***10^6)***, but this figure has the same problems as MIPS. Assume that 40% of the instructions executed on both P1 and P2 are floating-point instructions. Find the MFLOPS figures for the programs and show your calculations. Section 1.10 in the book may give you some insight.

Name:

(as it would appear on official course roster)

5. (12 pts) A pitfall cited in Section 1.10 and discussed in class is expecting to improve the overall performance of a computer by improving only one aspect of the computer. Consider a computer running a program that requires 250 seconds, with 70 sec. spent executing FP instructions, 85 sec. executed L/S instructions, and 40 sec. spent executing branch instructions (the remainder is spent executing other instruction types or idling).
    a. (6 pts) By how much is the total time reduced if the time for FP operations is reduced by 20%? Show your calculations.
    b. (6 pts) Can the total time can be reduced by 20% by reducing only the time for branch instructions? Why or why not (show your calculations)?

Name:
(as it would appear on official course roster)

6. (20 pts) Consider the following MIPS assembly program fragment which swaps values of two locations in memory (yes, similar to the one from **Lab01**).

```
swap:
      sll $t1, $a1, 2
      add $t1, $a0, $t1
      lw $t0, 0($t1)
      lw $t2, 4($t1)
      sw $t2, 0($t1)
      sw $t0, 4($t1)
```

Now consider the following C/C++ function definition:

```
void sort (int v[ ], int n)      // v[] = array, n = array size
{
   int i, j;
   for (i = 0; i < n; i += 1)
   {
         for (j = i - 1; j >= 0 && v[j] > v[j + 1]; j -= 1)
               swap(v, j);
   }
}
```

Write the sorting C++ code in MIPS assembly language – note that it uses the **swap** function. Remember that you can turn the **swap** instructions into a procedure (function) by adding a return instruction **jr $ra** inside that function and the routine calling instruction **jal swap** inside the code for **sort** (that calls the function **swap**). Organize your code and add whatever assembly instructions that are needed to get this to work, and then execute your code on **SPIM** with an array **v** of length 16 (you can select any set of integers for array **v**). Ensure that you use the stack per the MIPS calling convention. You can refer to and use the assembly program in Section 2.13 of the book (worth your while to look closely at that).

Hint: Organize your assembly code as follows –

```
.data
      # ...
.text
      swap:
      # ...
      sort:
      # ...
      main:
      # ...
```

Print/write neatly the entire MIPS assembly program on the next page(s).

Name:

(as it would appear on official course roster)

Question 6: Print/copy/write program here

Name:
(as it would appear on official course roster)

Question 6: Print/copy/write program here (if needed, or leave blank if not needed)