Homework 3

Practicing MIPS Assembly

Lecturer: Dr. Yisroel Mirsky

Objective:

To practice and enhance your skills in MIPS assembly programming focusing on arithmetic, memory, and control flow instructions. In this assignment, you should use the MARS MIPS Simulator to write, compile and execute your assembly code.

Submission Instructions:

- Submission is in groups of 1 only.
  Reservists may form groups of 2 (with or without another reservist).

- For each question, use the corresponding.asm file:
  - QS1.asm, QS2.asm, QS3.asm
  - In each file, you are allowed to write your code in the indicated areas (starting and ending with
    "# ----------Write Your Code Here----------"
  - Each file has some basic test testing the question partially. You are more than welcome to add more checks, but it won't be graded.
  - **The code to be graded is only the one written in the sections dedicated for it, as mentioned above. It Is important you follow the instructions for the gradings tests to work correctly.**
- Combine your asm files into a single zip file with no subdirectories (I.e. when you unzip the file, it unzips into 3 asm files named Qs1.asm, Qs2.asm, Qs3.asm)
- The zip file must be named ID.zip where "ID" is your תעודת זהות
  ID1_ID2.zip for those submitting in a group of two.

Grading:

Functionality (the code performs everything requested in the question): 60%

Proper coding (following the coding conventions we learned such as using the right registers in naming): 30%

Code clarity (the code is well organized and commented): 10%

Evaluation Criteria:

Your code will be evaluated based on the following:

- Correctness of the program.
- Clarity and organization of the code.
- Proper use of arithmetic and control flow instructions.
- Handling edge cases and input errors.

Question 1: Basic Arithmetic Operations (30 pnts)

Let A = 15 and B = 30.

Perform the following calculations:

- C = A + B
- D = A + 200 – 6
- E = B^2 (using add only)
- F = C + D + E - 2

The result of each operation above should be stored in main memory and not overwritten. At the end of the main function, the results should be stored in the $s0-$s3 registers. (C in $s0, D in $s1 and so on)

- Notice that A, B are already declared in $t0 and $t1. You can change the values but make sure you refer to A, B using these registers. If we change the initial value of $t0 we expect the effect of changing the initial value of A (if $t0 was changed to 10 then $s0 should store 30 at the end of the main function)

Skills Practiced:

- Basic arithmetic instructions
- Memory store and load operations

Question 2: Fibonacci Sequence Generator (30pnts)

Write a program to generate the first 10 Fibonacci numbers.

The Fibonacci sequence starts with 0 and 1. Each subsequent number is the sum of the two preceding ones.

Store each generated number into the array arr declared for you.

- At the end of the main function, you must store the array address in the $s0 register (we have implemented this for you). The array must contain your answers (the first element is 0 second is 1 and so on).
  We advise you run the Qs2 for that purpose. It will check that the two first elements in the array pointed by $s0 are correct. (This is of course not the full test to be that we will use)
- Notice that the first two elements are already declared in $t0 and $t1. You can change the values but make sure you refer to these registers. If we change the initial value of $t0 we expect the whole result array to be different
- Remember writing your code between the comments of the shape:
  “ # ---------- Write Your Code Here----------"

Skills Practiced:

- Arithmetic operations
- Memory manipulation with arrays
- Control flow with loops

Question 3: Array Element Finder (40 pnts)

Initialize the following array in using the `.data` section:

[-23, 2354, 34, 10, -3553, 4 234, 81, 90, 634, -27]

Write a program that finds and stores the following into separate memory locations:

- The maximum number in the array.
- The minimum number in the array.
- The sum of all numbers.
- The average of the numbers (you may round down to the nearest integer).

You must loop through the array only once.

The result of each operation above should be stored in the main memory and not overwritten.
You can use arr for this purpose but feel free to declare an array by yourself
At the end of the main function, the results should be stored in the $s0-$s3 registers. (Max in $s0, Min in $s1 and so on)

Skills Practiced:

- Memory manipulation with arrays
- Control flow with loops and conditionals
- Multiple arithmetic and comparison operations