Assignment 6: MIPS Pipeline - Part 2

This assignment is based on MIPS pipeline simulation using WinMIPS64 simulator. The simulator and its documentation/tutorial is available at http://indigo.ie/~mscott/. Since the simulator is based on windows, you will have to use something like Wine (https://www.winehq.org) to run it on ubuntu or macOS.

Submission Instructions: Make a directory named with your roll number. Write answer to each question (wherever asked) with appropriate numbering in a file named answers.txt and include it in your directory. For all the questions requiring a code, write the code to an appropriately named file (e.g., ques1a.s for question 1 part a) and include them in the directory. Finally submit a tar.gz i.e., [roll-no.].tar.gz (as follows) on moodle:

```
[roll-no.]
   |----answers.txt
   |----[all the code files and other files you think are relevant for demo]
```

This assignment is an extension of the last question of assignment 5 wherein you were asked to compute the sum of 12 FP numbers in the shortest possible time (minimum number of pipeline clock cycles). In that question, it was assumed that the summands reside initially in 12 FP registers whereas in this assignment, you will assume that the summands are instead in memory i.e., the data segment.

P1. Consider the following code snippet:

```
1       float  a[13],  sum=0.0;
2       //  Initialize  a  i.e. a[0] = 1.2,  a[1] = -7.87  . . .  etc.
3       for (int i = 0, i < 12, i++)
4               sum += a[i];
5       a[12] = sum;
```

Write the assembly equivalent of lines 3 through 5. You need to pay careful attention to the scheduling of instructions (and need to perform loop unrolling) in order to achieve the following goals:

(a) The primary goal is to consume as few pipeline clock cycles as possible so as to maximize the efficiency. Show the code and mention the total number of clock cycles consumed and the number of bubbles (stalls) appearing in the pipeline on execution of your code.

(b) The secondary goal is to use as few registers as possible. Given that you have maximized efficiency (from part (a)), i.e., used as few cycles as possible, what is the minimum number of registers required to accomplish this task? Show the optimized code and mention the minimum number of registers required such that the efficiency is same as that in part (a).

P2. This problem is a further extension to P1. Here you have to perform an addition of 499 FP numbers instead of 12. The above code snippet becomes the following in this case:

```
1       float  a[500],  sum=0.0;
2       //  Initialize  a  i.e. a[0] = 1.2,  a[1] = -7.87  . . .  etc.
3       for (int i = 0, i < 499, i++)
4               sum += a[i];
5       a[500] = sum;
```

Again answer the questions 1a and 1b w.r.t. the above code snippet.