CPE 631 Advanced Computer Systems Architecture: Homework #3

1 (25)	2 (25)	Total (50)

Q#1. (25 points) A Simple 5-stage Pipeline

Consider the following MIPS64 assembly language program that implements a simple C loop.

```
#
# for(i=99; i>=0, i=i-1) x[i] = x[i]+s;
# Aleksandar Milenkovic, milenkovic@computer.org
# CPE 631, January 2010

      .data
xarray: .word  0x0000000000000000,0x0000000000000001,0x0000000000000002,0x0000000000000003
        .word  0x0000000000000004,0x0000000000000005,0x0000000000000006,0x0000000000000007
        .word  0x0000000000000008,0x0000000000000009,0x000000000000000A,0x000000000000000B
        .word  0x000000000000000C,0x000000000000000D,0x000000000000000E,0x000000000000000F
        .word  0x0000000000000010,0x0000000000000011,0x0000000000000012,0x0000000000000013
        .word  0x0000000000000014,0x0000000000000015,0x0000000000000016,0x0000000000000017
        .word  0x0000000000000018,0x0000000000000019,0x000000000000001A,0x000000000000001B
        .word  0x000000000000001C,0x000000000000001D,0x000000000000001E,0x000000000000001F
        .word  0x0000000000000020,0x0000000000000021,0x0000000000000022,0x0000000000000023
        .word  0x0000000000000024,0x0000000000000025,0x0000000000000026,0x0000000000000027
        .word  0x0000000000000028,0x0000000000000029,0x000000000000002A,0x000000000000002B
        .word  0x000000000000002C,0x000000000000002D,0x000000000000002E,0x000000000000002F
        .word  0x0000000000000030,0x0000000000000031,0x0000000000000032,0x0000000000000033
        .word  0x0000000000000034,0x0000000000000035,0x0000000000000036,0x0000000000000037
        .word  0x0000000000000038,0x0000000000000039,0x000000000000003A,0x000000000000003B
        .word  0x000000000000003C,0x000000000000003D,0x000000000000003E,0x000000000000003F
        .word  0x0000000000000040,0x0000000000000041,0x0000000000000042,0x0000000000000043
        .word  0x0000000000000044,0x0000000000000045,0x0000000000000046,0x0000000000000047
        .word  0x0000000000000048,0x0000000000000049,0x000000000000004A,0x000000000000004B
        .word  0x000000000000004C,0x000000000000004D,0x000000000000004E,0x000000000000004F
        .word  0x0000000000000050,0x0000000000000051,0x0000000000000052,0x0000000000000053
        .word  0x0000000000000054,0x0000000000000055,0x0000000000000056,0x0000000000000057
        .word  0x0000000000000058,0x0000000000000059,0x000000000000005A,0x000000000000005B
        .word  0x000000000000005C,0x000000000000005D,0x000000000000005E,0x000000000000005F
        .word  0x0000000000000060,0x0000000000000061,0x0000000000000062,0x0000000000000063

len:   .word 100
s:     .word 5


        .text
        dadd $t0,$zero,$zero         # $t0 = 0
        ld $t1,len($zero)            # $t1 = len (array size)
        dsll $t2,$t1,3               # $t2 = len*8 (800)
        ld $t3,s($zero)              # $t3 = s

ml:     daddui $t2,$t2,-8
        ld $t4,xarray($t2)           # $t4 = x[i]
        dadd  $t4,$t4,$t3            # x[i] = x[i] + s
        sd $t4,xarray($t2)
        daddui $t1,$t1,-1
        bne $t1,$zero,ml
        nop
out:    halt
```

A. (7 points) Show the timing of one loop iteration (in steady state) assuming the MIPS pipeline without any forwarding hardware. Give the total number of clock cycles to execute the program, a single loop (in the steady state), and the number of stalls (for each category). Use WinMIPS64 simulator to support your answers.

B. (8 points) Show the timing of one loop iteration (in steady state) assuming the MIPS pipeline with forwarding hardware. Give the total number of clock cycles to execute the program, a single loop (in the steady state), and the number of stalls (for each category). Use WinMIPS64 simulator to support your answers.

C. (10 points) Assuming MIPS pipeline with delayed branch and forwarding hardware, schedule instructions (move them around preserving program semantics) in the loop including branch delay slot. You may reorder instructions and modify individual instruction operands, but do not undertake other loop transformations. Show a pipeline diagram and give the total number of clock cycles to execute the program, a single loop (in the steady state), and the number of stalls (for each category). Use WinMIPS64 simulator to support your answers.

Q#2. (25 points) Consider the following code fragment in C:

```
double x[100] = {0.0, 0.1, ...., 9.9};     // 100 elements
double y[100] = {0.0, 1.0, .... 99.0};     // 100 elements
double z[100] = {0.0, 0.0, .... 0.0};      // 100 elements

double a = 10.0;

for (i=99; i>=0; i--) {
      z[i] = a*x[i] + y[i];
      y[i] = a*y[i];
}
```

A. (15 points) Write a MIPS64 assembly language program that implements the code fragment assuming no optimizations (no code scheduling is used to eliminate stalls). Verify its correctness using WinMIPS64 and discuss its performance. Assume the default configuration of the MIPS64 (7 clock cycles for FP multiplication, 4 clock cycles for FP addition, with data forwarding and branch delay slot). Illustrate the pipeline for a single loop iteration and clearly mark the stalls and what caused them.

B. (10 points) Schedule the code inside the loop to reduce the number of stalls. Illustrate the pipeline for a single loop iteration (in the steady state).