York University

Dept. of Computer Science and Engineering

CSE4201 – Computer Architecture

HW 2

Problem 1

Consider the DAXPY loop (double-precision aX + Y), the following loop implements this loop for a vector of length 100.

```
foo:      L.D       F2,0(R1)   ;load X(i)
          MUL.D     F4,F2,F0   ;multiply by a
          L.D.      F6,0(R2)   ;load Y(i)
          ADD.D     F6,F4,F6   ; add ax+y
          S.D       F6,0(R2)   ; store y
          DADDUI    R1,R1,#8
          ADAAUI    R2,R2,#8
          DSGTUT    R3,R1,#800;test if done
          BEQZ      R3,foo
```

The pipeline function units are described as follows

FU	**Cycles in EX**	**Number of FU's**	**# of Reservation Stations**
Integer	1	1	5
FP adder	4	1	3
FP Multiplier	15	1	2

Assume the following

- Function units are not pipelined
- There is no forwarding between function units; results are communicated by the CDB
- The EX stage does both the effective address calculations and the memory access for load and stores (IF/ID/IS/EX/WB)
- Loads take 1 cycle
- The issue stage (IS) and write result (WB) stages each take 1 clock cycle.
- There are 5 load buffers and 5 store buffers.
- Assume that the BEQZ instruction takes 0 clock cycles.

a. Use the single issue Tomasulo's MIPS pipeline. Show for each instruction when it begins execution, and when it ends execution for the first three iterations of the loop.
b. Do the same if we have a 2-issue Tomasulo's and a fully pipelined functional units.

Problem 2

List all the dependences (output, anti, and true) in the following code fragment. Indicate whether the true dependences are loop carried or not.

```
for (i=2;i<100;i++){
     a[i]   = b[i]+a[i];       /* S1 */
     c[i-1] = a[i]+d[i];       /* S2 */
     a[i-1] = 2 * b[i];        /* S3 */
     b[i+1] = 2 * b[i];        /* S4 */
     }
```

Problem 3

Consider the code in Problem 1. Assuming the following latency in clock cycles.

I producing result	**I using result**	**Latency in clock cycles.**
FP ALU OP	FP ALU OP	3
FP ALU OP	Store	2
Load Double	FP ALU OP	1
Load Double	Store Double	0

a. Show how long it takes to do a single iteration *unscheduled* (same order as in the given code). Then show how long *scheduled*?
b. Assume a single issue pipeline; unroll the loop as many times as necessary to schedule it without any stalls. How long per iteration?
c. Assume a VLIW with instructions that contain 5 operations (2 mem operations. 2 FP, and 1 int). Unroll the loop the minimum number of times to schedule it without stalls (stalls here means a completely empty instruction).