HY425: Computer Systems Architecture

Homework Problem Set 3
Assignment: Monday 30/11/2015
Due: Wednesday 9/12/2015 23:59:59

Instructions: Solve all problems in a .pdf file and send them via e-mail to Dimitris Giannopoulos (dgiannop@csd.uoc.gr). **Use the subject: HY425 - Homework 3**

Problem 1 (100 points)

The following code is known as the DAXPY loop (**D**ouble-precision **AX** **P**lus **Y**) from the BLAS package (**B**asic **L**inear **A**lgebra **S**ubprograms), where **x** and **y** are arrays of doubles and **a** is a double:

```
for ( i=0 ; i<N ; i++ ){
        y[i] = a * x[i] + y[i];
}
```

Assume that our compiler has generated the following RISC assembly code:
[note: R1 keeps x[] index , R2 keeps y[] index, R4 keeps x[N-1] index, F0 keeps a]

	Instruction	**Notes**
Loop:	LD F2, 0(R1)	Load **x[i]** into **F2**
	MULTD F4, F2, F0	Put **a*x[i]** into **F4**
	LD F6, 0(R2)	Load **y[i]** into **F6**
	ADDD F6, F4, F6	Put **a*x[i] + y[i]** into **F6**
	SD F6, 0(R2)	Store **F6** into **y[i]**
	ADDI R1, R1, 8	Increment **x** index (R1)
	ADDI R2, R2, 8	Increment **y** index (R2)
	SGT R3, R1, R4	Test if loop done
	BEQZ R3, Loop	Loop if not done
	NOP	Branch delay slot

Further assume the following latencies of a typical 5-stage in-order pipelined RISC processor (IF, ID, EX, MEM, WB) and that bypassing is applied whenever possible:

Operation(s)	Stage	Latency (cycles)
All Integer	EX	1
LD	MEM	2
SD	MEM	1
ADDD	EX	3
MULTD	EX	5

i) Show how the RISC processor would execute each loop iteration (indicate stalls) and calculate the total number of cycles required to run 120 iterations of the loop.

ii) Try to rearrange the instructions in order to reduce the number of stalls and then calculate the total number of cycles required to run 120 iterations of the loop. Compare the performance now with (i).

iii) Loop-unroll as many iterations needed, in order to reduce the number of stalls and then calculate the total number of cycles required to run 120 iterations of the loop. Compare the performance now with (i) and (ii).

iv) Apply the technique of software pipelining and then calculate the total number of cycles required to run 120 iterations of the loop. Compare the performance now with (i), (ii) and (iii). Do not forget the startup and cleanup code!

Now assume a VLIW processor that can issue two memory references, two FP operations, and one integer operation or branch in every clock cycle. Further assume the same operation latencies with the RISC processor above and that you have infinite registers.

v) Show how the code that you generated in (iii) would run in the VLIW processor and then calculate the total number of cycles required to run 120 iterations of the loop. Compare the performance now with (iii) and (iv).

vi) Show how the code that you generated in (iv) would run in the VLIW processor then calculate the total number of cycles required to run 120 iterations of the loop. Compare the performance now with (iii), (iv) and (v).

vii) Loop-unroll as many iterations needed, in order to reduce the number of stalls and keep the VLIW pipeline utilized, then calculate the total number of cycles required to run 120 iterations of the loop. Compare the performance now with (iii), (iv), (v) and (vi).