THE UNIVERSITY OF BRITISH COLUMBIA
Computer Science 313, Section 201
Quiz 1
February 10, 2017

Practice quiz 2, 45 minutes

Marks

1.	/12
2.	/10
3.	/7
4.	/7
Total	/36

12 1) Some short answer questions.

a) Briefly describe the functional difference between a pipeline register and a register like %rax.

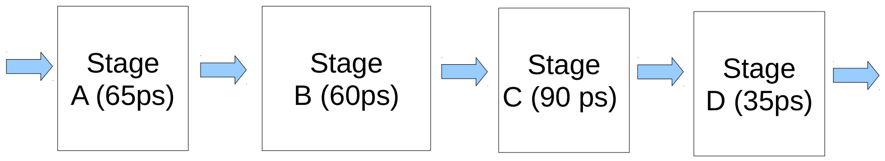
b) One of the branch prediction rules we discussed was one where forward jumps are not taken but backwards ones are. Why does such a rule make sense? (Think of how code for certain programming language structures might make use of this.)

c) When a conditional jump instruction is executed on a CPU that solves all hazards by stalling some number of bubbles get introduced into the instruction stream. So, when a conditional move is executed, on this same CPU, how many bubbles are introduced? Explain.

d) In our pipelined CPU we only care about casual dependencies and not output or alias (anti) dependencies. Carefully explain why.

10

2) Consider the following picture of a the various stages in a sequential CPU. Within each stage the amount of time to execute that stage is given in pico seconds (ps) (A ps is 10^{-12} seconds, that is there are 10^{12} ps in a second and a billion is 10^{9}) **For all parts of this question show your work and circle your answer. Make sure your final answer is a single number and circle it.**



a) When an instruction is executed on this processor what is its latency in ps?

b) What is the throughput in billions of instructions per second?

c) The plan is to create a pipelined CPU by inserting stage registers between each stage. The time to load the stage registers is 10ps. How long, in ps, does a clock tick need to be for this pipelined CPU?

d) For this pipelined CPU what is the latency, in ps, for an instruction?

e) What is the maximum throughput, in billions of instructions per second, of this pipelined CPU?

7 3) The table below contains some Y86-64 assembly code. You are to fill in all the open (none greyed cells.) In column (a) you are to list all the data hazards for the instruction in that row in the form of the line number followed by the dependency causing the hazard. For example if the irmovq instruction of row 3 depended upon rax in line 1 and rdi in line 2 you would write it as shown in the table. If there are no dependencies write none. In column (b) write the number of stalls this instruction is subject to. If the there is more than one dependency then indicate the maximum number of stalls. In column (c) follow the same rules as for column (b) and write the maximum number of stalls for that instruction for our pipelined CPU with data forwarding.

vv

			(a) Pipelined – no forwarding hazard/dependency	(b)Number of stalls no - forwarding	(c) Pipelined – forwarding CPU Stalls
1	xorq	%rax, %rax			
2	irmovq	data, %rcx			
3	irmovq	$80, %rdi	(eg. 1 rax, 3 rdi)	14	67
4	top: addq	%rdi, %rcx			
5	mrmovq	(%rcx), %rbx			
6	irmovq	$8, %rdx			
7	addq	%rbx, %rax			
8	subq	%rdx, %rdi			
9	jg	top			
10	call	done			

7 4) This question makes reference to the code in the previous question but concerns control hazards.

a) For the pipelined CPU with no forwarding and no branch prediction when the the jg instruction of line 9 is in the decode stage what instruction is in the fetch stage? If it is a bubble use the NOP instruction. Explain.

b) For the pipelined CPU **with branch prediction** (assume branches are taken) what instruction is in the **fetch** stage when the jg instruction is in the execute stage ? If it is a bubble use the NOP instruction. Explain.

c) Observe that instruction 6 (irmovq) doesn't really need to be in the loop. Move it so that it now executes between instructions 1 and 2. For the CPU with data forwarding and branch prediction will the change result in the overall running time of the supplied code increasing, decreasing, or staying the same? (Note that overall running time means the total number of clock ticks required to execute the code.) Explain.