CS 250 Spring 2017 Homework 07
Due 11:58pm Friday, March 10, 2017
Submit your typewritten file in PDF format to Blackboard.

1. Text exercise 5.8. Assume that after the last instruction of the given code snippet is fetched, that the instructions fetched are denoted "???". Note that the pipeline of Figure 5.5 cannot read an operand that is written into the register unit on clock cycle N until clock cycle N+1. The comment to line 3 of the assembly code should read "# put 20 in register 9".
   Answer: Let your answer start as follows:

Fetch Clock Cycle	Instr.	Operands	Comment
1	loadi	r7, 10	# put 10 in register 7
2	loadi	r8, 15	# put 15 in register 8
3	loadi	r9, 20	# put 20 in register 5
4	nop		
5	nop		
6	nop		
7	addrr	r10, r7, r8	# add registers 7 8; put the result in register 10
8	movr	r12, r9	# copy register 9 to register 12
9	movr	r11, r7	# copy register 7 to register 11
10	nop		
11	nop		
12	nop		
13	nop		
14	addri	r14, r11, 27	# add 27 plus register 11; put the result
15	addrr	r13, r12, r11	# add registers 11 and 12; put the results
16	???		

2. Text exercise 6.4.
   In a jump operation the maximum branch distance is limited to (-2^{15}) to $(2^{15} - 1)$. In the provide exercise it specifies 'jump **40000**(r15)'. The 40000 is greater than $2^{15} - 1$ (which is 32767) therefore it is an invalid instruction and cannot be performed.

3. Text exercise 6.6.
   The circuit in figure 6.6 is not an infinite loop that runs wildly because first off it is limited by the floor and ceiling of the memory, therefore preventing it from running infinitely. It also does not run wildly as it is systematically using an adder and counter to increment the bits to navigate the memory. This is not random nor wild as this is a planned algorithm to navigate the memory.

4. Explain how CPU with two execution modes uses those modes to run an operating system and also execute code written by a user.
   A CPU usually has two execution modes, kernel mode and user mode. To run an operating system, the CPU will most definitely use the kernel mode, as this has direct and unrestricted access to underlying hardware and memory. This unrestricted access is very 'low-level' and any error can be catastrophic as it messes directly with hardware and memory addresses. Code written by a user will be executed most likely on the user mode. This user mode has no direct access to underlying hardware or memory addresses, instead they must utilize APIs and other methods to indirectly access these. This 'higher-level' access is less catastrophic than the kernel access and most errors/bugs are recoverable (contrary to kernel mode).

5. You are designing a microcoded version of the CPU in chapter 6 of our textbook. You have built a fully-functional circuit that includes all the components shown in Figure 6.9. What additional component do you need? To what should its input and its output be connected in Figure 6.9?
   You would need to add a microcontroller in order to work with the microcode. The input of this system would be a set of macro instructions that are implemented with microcode and executed by the CPU.

6. Assume that the IF (instruction fetch), ID (instruction decode), EX (instruction execute), MEM (data memory access, read or write), and WB (write-back to a register) stages of a 5-stage pipelined version of Figure 6.9 have propagation delays of 20, 30, 50, 50, and 10 nanoseconds (ns), or 10^{-9} seconds, respectively.
   a. What is the fastest clock rate for this pipeline?
      The fastest clock rate for this pipeline is limited by the slowest propagation delay, therefore it is 50 nanoseconds.

   b. Assume that the stage registers required for pipelining each have 3 ns of propagation delay to record an input. What was the propagation delay of the processor before pipelining?
      If the instructions were ran back to back, the total propagation delay for the instruction to complete would be 160 nanoseconds.

   c. Ideally, how much faster is this processor once it is pipelined?
      Ideally, this processor would be much faster due to the pipelining allowing instructions to be run sequentially and stacked on top of each other (assuming no dependencies) thus drastically reducing the time it takes for lists of instructions to complete.

7. Could the multiplexer controlled by the signal RegDst in Lecture 15, slide 52 be moved from the EX stage to the MEM stage without compromising the correct functioning of the pipeline?
   Yes.

Could this multiplexer be moved all the way to the WB stage and have correct operation of the pipeline be preserved?
No.

Why does the design in this slide place the multiplexer in the EX stage?
The design places the multiplexer in the EX stage because this allows it to be run one clock cycle sooner. It technically could be placed in the MEM stage as well however placing it in the EX stage (sooner than MEM) is beneficial to the runtime of the system.