Name ____

CS/COE 0447 Example Problems for Exam 4
Spring 2009

1. Suppose we are considering two different multi-cycle processors: a Steeler with a 1 GHz (1ns) clock rate and a Panther at 2 GHz (0.5ns). Both processors implement the same instruction set architecture, but the cycles per instruction type varies for the two implementations. You are trying to decide which machine to purchase and made some measurements of instruction frequency for each machine:

Instruction Class	Instruction Frequency	Steeler cycles for instruction class	Panther cycles for instruction class
Arithmetic	40%	3	7
Load/Store	25%	7	12
Branch	25%	3	8
Jump	10%	2	6

The column "cycles" gives the number of cycles for each instruction type on the processors.

(a) What is the average cycles per instruction for each machine?

*CPI Steeler = 0.4*3 + 0.25*7 + 0.25*3 + 0.1*2 = 3.9*

*CPI Panther = 0.4*7 + 0.25*12 + 0.25*8 + 0.1*6 = 8.4*

(b) Which machine is faster?

*CPU time = IC * CPI * Clock cycle length*

*CPU time Steeler = IC * 3.9 * 1ns = IC * 3.9ns*

*CPU time Panther = IC * 8.4 * 0.5ns = IC * 4.2ns*

IC on Steeler and Panther is the same value since they are the same instruction set.

thus, the Panther is the slower processor, despite having a faster clock rate.

(c) What is the speedup of the faster machine versus the slower machine?

*Speedup = CPU time Panther / CPU time Steeler = IC * 4.2ns / IC * 3.9ns = 1.08*

2. For this question, refer to the datapath diagram on slide 73 from the April 9 lecture for the multi-cycle implementation of MIPS. Indicate how the control signals are set on the given clock cycles for the instruction sequence:

Number	**Instructions**
1	add $1,$2,$3
2	lw $1, 0($10)
3	beq $5,$6,16

Name ______________________________

Assume the program counter is initially set to instruction number 1 and instruction number 1 is fetched at clock cycle 0. Show how control signals are set on only clock cycles 0 to 9 *(the solution shows all cycles for the three instructions)*.

Cycle	IorD	RegDst	MemToReg	RegWrite	ALUSrcA	ALUSrcB
0	0 (Instr)	X	X	0	0 (PC)	1 (4)
1	0 (Instr)	X	X	0	0 (PC+4)	3 (offset)
2	0	X	X	0	1 (A)	0 (B)
3	0	1 (Rd)	0 (ALUout)	1	X	X
4	0	X	X	0	0 (PC)	1 (4)
5	0	X	X	0	0	3
6	0	X	X	0	1 (A)	2 (offset)
7	1 (Data)	X	X	0	X	X
8	0	0 (Rt)	1 (MDR)	1	X	X
9	0	X	X	0	0	1
10	0	X	X	0	0	3
11	0	X	X	0	1 (A)	0 (B)

3. Suppose that CPU time is 1000 seconds, clock cycle time is 1 ns, and the CPI is 2. What is the instruction count?

*CPU time = Instruction count (IC) * Cycles per instruction * Clock cycle length*

*we have: 1000s = IC * 2 * 1ns and so, IC = 1000s / 2ns = 500 billion*

be careful of the units in this problem. 1 ns is 10^{-9} second

4. Suppose that the Eagles processor has a CPU time of 1100 seconds for program P and the Eagles processor is 1.1 times faster than the Steeler processor. What is the CPU time for P on the Steeler?

*CPU time for the Steeler is 1100s * 1.1 = 1210s*

5. You ran a program on the multi-cycle Incline processor and found the following information:

Instruction Class	Frequency	Number Cycles
ALU instructions	37%	2
Branches	18%	2
Loads	23%	4
Stores	22%	3

The "Number Cycles" column gives how many cycles an instruction of a particular type takes. What is the average cycles per instruction (CPI) for this processor?

*this is another CPI problem: CPI = 0.37*2 + 0.18*2 + 0.23*4 + 0.22*3 = 2.68 cycles*

6. Explain why the multicycle implementation of MIPS has to save between cycles, whereas the single cycle implementation does not.

because an instruction is executed over multiple cycles in multicycle implementation, certain state about an instruction has to be passed from one cycle to the next. for instance, the destination register number has to be propagated between all cycles until writeback is done. in the single cycle implementation, everything is done on one cycle and all information is available. note that the register file saves state between instructions (rather than between cycles for a single instruction in the multicycle case).

7. Let's add a bne instruction to the the single cycle implementation from class (see slide 59 from April 7). This implementation supports only the beq instruction. To support bne, we augment the control unit with a new signal, called BranchCondition. This signal is 0 when the branch is beq and is 1 when the branch is bne. The control unit still has the Branch signal. Explain any further modifications necessary to the datapath to support bne.

the BranchCondition signal can be used to invert the Zero output from the ALU (think about what circuit can do this!). no other changes are necessary because the Branch signal will still be set to 1 in both cases. the Zero output (possibly inverted) can be AND'ed with the Branch signal to derive the PCsrc control.

8. Now, let's add support for immediates to arithmetic instructions (e.g., addi) to the multi-cycle implementation (see slide 73 from April 9).

a) What information is needed from the control unit to support this instruction?

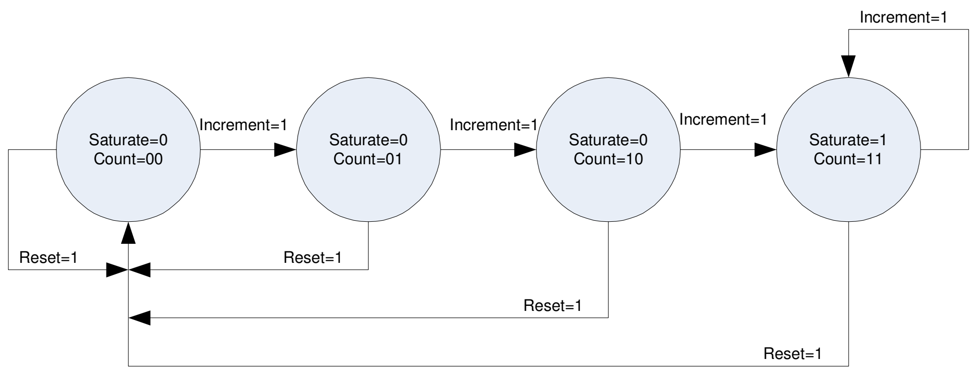
that the instruction is an immediate form (I-type).

b) Do any changes need to be made to the datapath itself? If so, explain what is needed.

the mux control for ALUSrcB has to be set to select the sign-extended value.

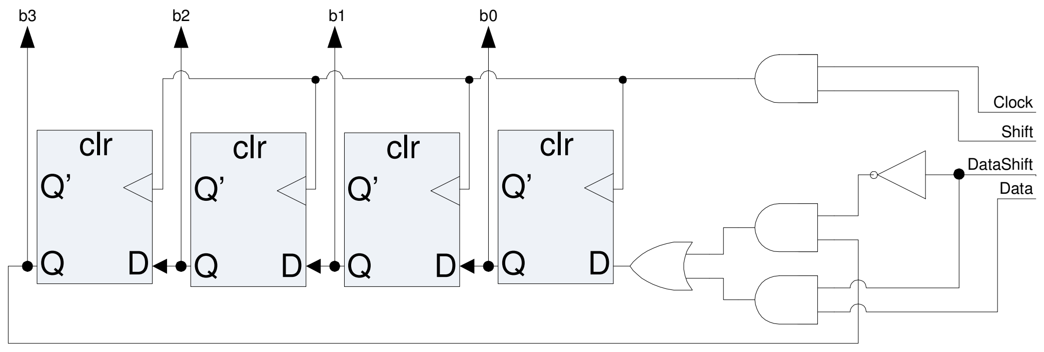
9. Consider a *saturating counter*. This counter does not "roll over" when it reaches the end of its range (e.g., a 4-bit counter with the value 1111b keeps this same value when 1 is added to it). Suppose the counter is incremented when the input signal Increment is 1. The counter is reset when the input signal Reset is 1. The counter has output signals for its bit values. In addition, it has an output signal to indicate saturation, called Saturate. Draw a finite state machine for this counter, assuming it is 2 bits (use circles for states and lines for state transitions, with appropriate labels to indicate signal values).

simply make each counter value to be a state: label the states with a possible counter value. a transition is made between successive states (i.e., 00 to 01) whenever Increment=1. the final state (11) sets Saturate=1. the state machine does not transition from state 11 until Reset=1 (back to state 00). all other states (00, 01, 10) have transitions to state 00 for Reset=1.



10. Now, consider a *left rotating register*. This register has three input signals: Data, DataShift, and Shift. When Shift=1, all bits are left shifted one position. DataShift controls what value is shifted into the least significant bit. When DataShift=1 and Shift=1, the value of Data is shifted into the least significant bit of the register. When DataShift=0 and Shift=1, the most significant bit of the register prior to shifting is put into the least significant bit. Draw the circuit for this rotating register, using D-flip flops and any necessary control logic.

the problem didn't specify the number of bits, so let's just assume it's four bits.



notice that the value shifted in is dependent on DataShift.

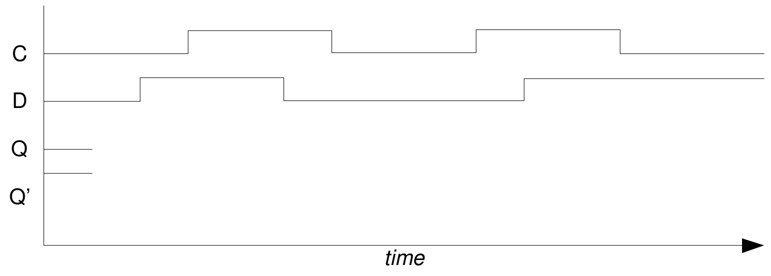
DataShift can be thought of as a "control signal" for a mux. it can set a 2:1 mux to select one of two values (Q from bit 3 or Data input) for D. 0 is Q (bit 3) and 1 is Data.

11. Consider an RS latch. The table below shows values for R, S, Q, and Q' at time *t*. Give the values of Q and Q' at time *t+1*.

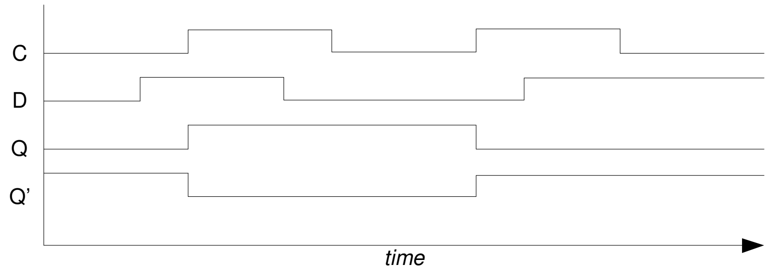
R	**S**	**Q**	**Q'**	**Q(*t+1*)**	**Q'(*t+1*)**
0	0	0	1	__0___	___1__
0	1	0	1	__1___	___0__
0	1	1	0	__1___	___0__
1	0	1	0	__0___	___1__

an easy way to remember this: RS=00=storage mode; RS=10=reset to 0; RS=01=set to 1.

12. Consider a D flip-flop. Assume this flip-flop is "rising-edge triggered". The figure below gives a signal timing diagram that shows waveforms for C, D, Q, and Q'. A waveform depicts a signal value over time. In the diagram, assume that Q has the initial value 0 and Q' has the initial value 1. Given C and D, draw the signal waveforms for Q and Q' in the remaining part of the diagram.



The D flip-flop tracks the input values on the rising clock edge (C).



Name ____

What happens for a D latch? (It tracks the inputs whenever C=1 -- not just on the edge.)