Computer Science 3410, Cornell University
Spring 2016, Prof. Bracy

Prelim 1: 3 March 2016

Solutions

1. Write your name here and NetID at the top of each odd page or face -2 pt penalty. [0 pts]

2. Multiple Choice [8 pts]

   (a) [2 pts] **Pipelining.** Consider a non-pipelined processor with clock period C. If you divide the processor into N stages, your new clock period will be:

   A less than C/N
   B C/N
   C greater than C/N
   D C
   E N

   Correct Answer: **C**

   (b) [2 pts] **Simple CPU.** Which of the following statements about the Program Counter is false? There is only one correct answer.

   A The PC can change every cycle.
   B The PC is a register that has an address in it.
   C The PC is stored in one of the 32 general purpose MIPS registers.
   D The PC always ends in 00.
   E It isn't possible to perform an `addi` instruction that adds an immediate value to the PC.

   Correct Answer: **C**

   (c) [2 pts] **Endian-ness.** Which of the following instructions are endian-independent? (In other words, which of these instructions can be executed correctly without knowing the endian-ness of the machine?) There is only one correct answer.

   A Load byte
   B Load halfword
   C Load word
   D None of these instructions are endian-independent.
   E All of these instructions are endian-independent.

   Correct Answer: **A**

   (d) [2 pts] **MIPS.** Which of the following is **not** a way in which MIPS is ideal for pipelining?

A All MIPS instructions are 32 bits.
B MIPS only has 3 instruction formats.
C Memory can only be accessed by load and store instructions.
D Registers can only be accessed by R-type instructions.
E All of these are ways in which MIPS is ideal for pipelining.

Correct Answer: **D**

3. Number Systems [8 pts]

(a) [2 pts] Express the unsigned integer 10010011 in decimal (base 10).

$$10010011 = 128 + 16 + 2 + 1 = 147_{10}$$

(b) [2 pts] Express the sign magnitude number 10010011 in decimal (base 10).

$$10010011 = \text{-}\ (16 + 2 + 1) = -19_{10}$$

(c) [2 pts] Express the 2s complement number 10010011 in decimal (base 10).

$$10010011 = \text{-}128 + 16 + 2 + 1 = -109_{10}$$

(d) [1 pt] Express the unsigned binary number 10010011 in octal.

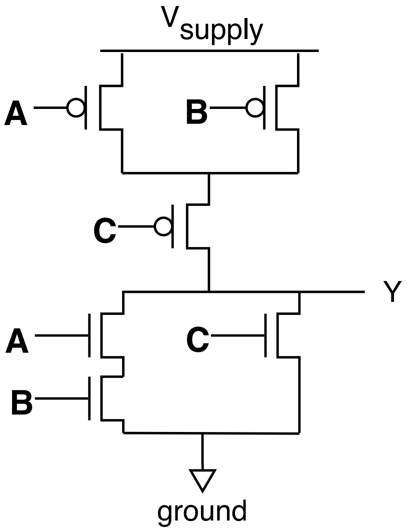
010 010 011 = o223

(e) [1 pt] Express the 2s complement number 10010011 in octal.

110 010 011 = o623

4. Transistors, Gates, and Storage [12 pts]

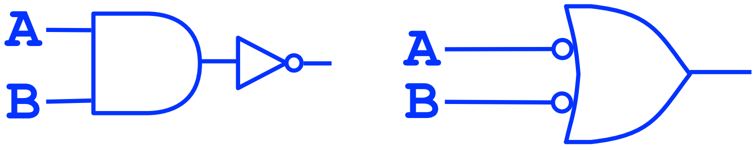
(a) [4 pts] Recall that a p-transistor connects source to drain when a negative/zero charge is applied to the gate. Complete the truth table for the following CMOS circuit with inputs A, B, and C and output Y.



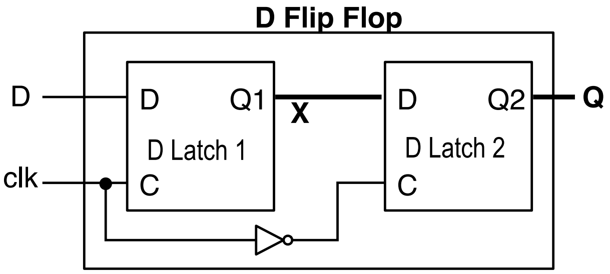
A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

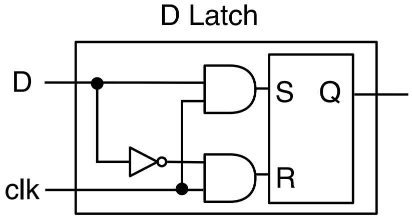
(b) [2 pts] DeMorgan's Laws are logical identities that can be used to simplify or manipulate a logical formula. One of them states: $\overline{(ab)} = \overline{a} + \overline{b}$

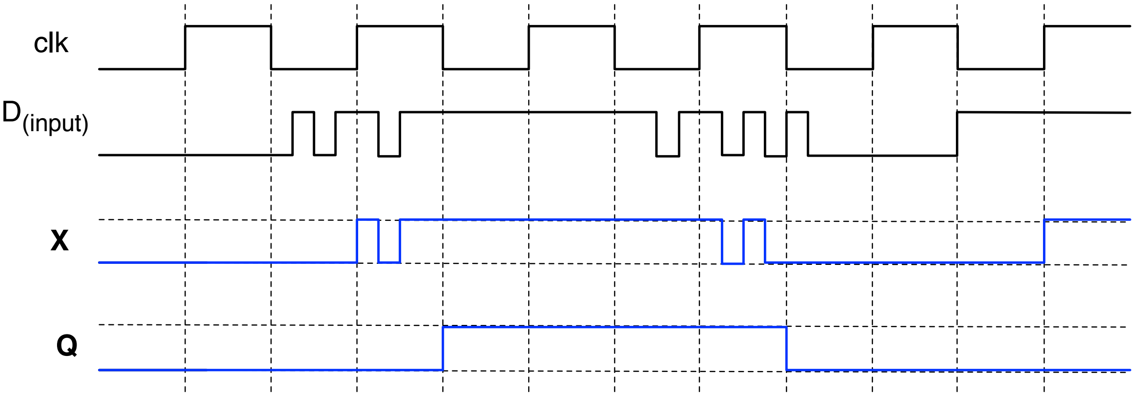
Draw the circuit version of both sides of this equation. You should draw two distinct (non-identical) circuits that have the same functional behavior. They should both have inputs A and B. They should not be connected to each other. You may use any standard logic gates.



(c) [6 pts] Complete the timing diagrams, indicating the values for X (the output of the first D Latch) and Q (the output of the D Flip Flop) in the following diagram. Both X and Q start low.





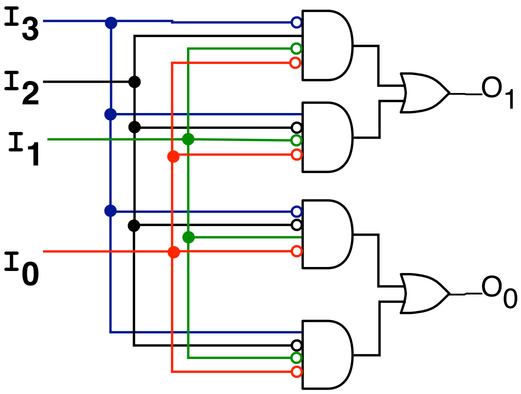


5. Components [8 pts]

Decoders convert a binary input value to a unary (one-hot) output. Another frequently used component is an **encoder** which does the opposite. It takes a unary input and converts it into a binary encoded output. An encoder answers the question “which input was hot?”. A 4-to-2 encoder’s input is in the form $I_3\ I_2\ I_1\ I_0$ and its output is in the form $O_1\ O_0$. If you input 0100 into the encoder, the output will be 10, which means “Input I_2 was hot”. You may assume that there are only ever 4 input combinations to the encoder: 1000, 0100, 0010, and 0001 and the encoder only needs to create the correct output for these 4 input combinations.

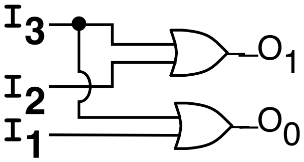
(a) [4 pts] Draw the circuit diagram for a 4:2 encoder using basic logic gates with any number of inputs. Use the inputs and outputs provided. The zero indexed bit is the *least significant bit*. You do not need to simplify your circuit. (That comes next).

I3	I2	I1	I0	01	O0
0	0	0	1	0	0
0	0	1	0	0	**1**
0	1	0	0	**1**	0
1	0	0	0	**1**	**1**



(b) [4 pts] Now simplify and re-draw your circuit using the method of your choice. If your circuit above is already simplified (and correct) then you will be awarded the points for this part with no further work required.

I3	I2	I1	I0	01	O0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1



6. Logic Minimization [15 pts]

For this question, you have only inverters, 2-input AND gates, and 2-input OR gates. Assume that these gates have a delay of 1, 2, and 3 ns respectively.

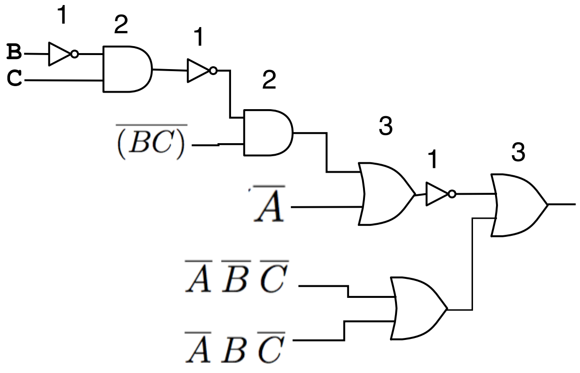
Consider the un-simplified circuit associated with the following logic:

$$\overline{A}\,\overline{B}\,\overline{C} \;+\; \overline{A}\,B\,\overline{C} + \overline{(\overline{A} + \overline{(BC)})\;\overline{(\overline{B}C)})}$$

(a) [3 pts] What is the maximum delay for this circuit? Assume an **unsimplified circuit** that has been laid out so as to minimize the critical path.

$$\overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,B\,\overline{C} + \overline{(\overline{A} + \overline{(BC)}\;\overline{(\overline{B}C)})}$$

Critical path includes the 3rd term:
1+ 2 + 1 + 2 + 3 + 1 + 3 = 13 ns


1
2
1
2
3
1
3
B
C
$\overline{(BC)}$
$\overline{A}$
$\overline{A}\,\overline{B}\,\overline{C}$
$\overline{A}\,B\,\overline{C}$

(b) [6 pts] Simplify the above equation using boolean logic. Apply only **one simplification per line** to receive full credit.

$$\overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,B\,\overline{C} + \overline{(\overline{A} + \overline{(BC)}\;\overline{(\overline{B}C)})} =$$

$A'B'C' + A'BC' + \overline{(A' + (BC)'(B'C)')}$

first simplify this mess

A((BC)'(B'C)')'
A((BC)+(B'C))
ABC + AB'C

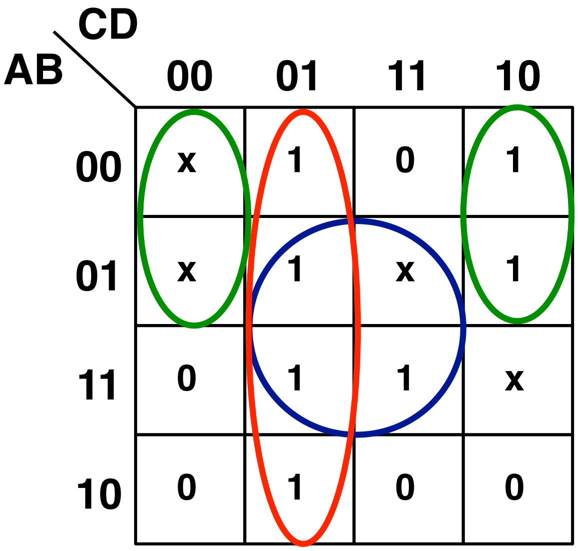
A'B'C' + A'BC' + ABC + AB'C
ABC + AB'C
→ AC(B + B')
→ **AC**

A'B'C' + A'BC'
A'C'(B + B') ←
A'C' ←

AC + A'C'

(c) [2 pts] What is the maximum delay for your simplified circuit?
1 inverter, 1 OR gate, 1 AND gate = 6 ns

(d) [4 pts] Simplify the function shown in the following Karnaugh map. Your equation must be as simple as possible in order to receive full credit.



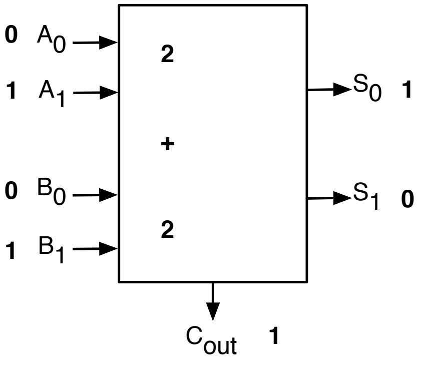
AB \ CD	00	01	11	10
00	x	1	0	1
01	x	1	x	1
11	0	1	1	x
10	0	1	0	0

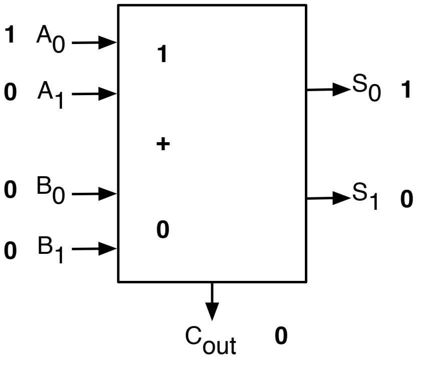
$A'D' + C'D + BD$

7. Arithmetic [12 pts]

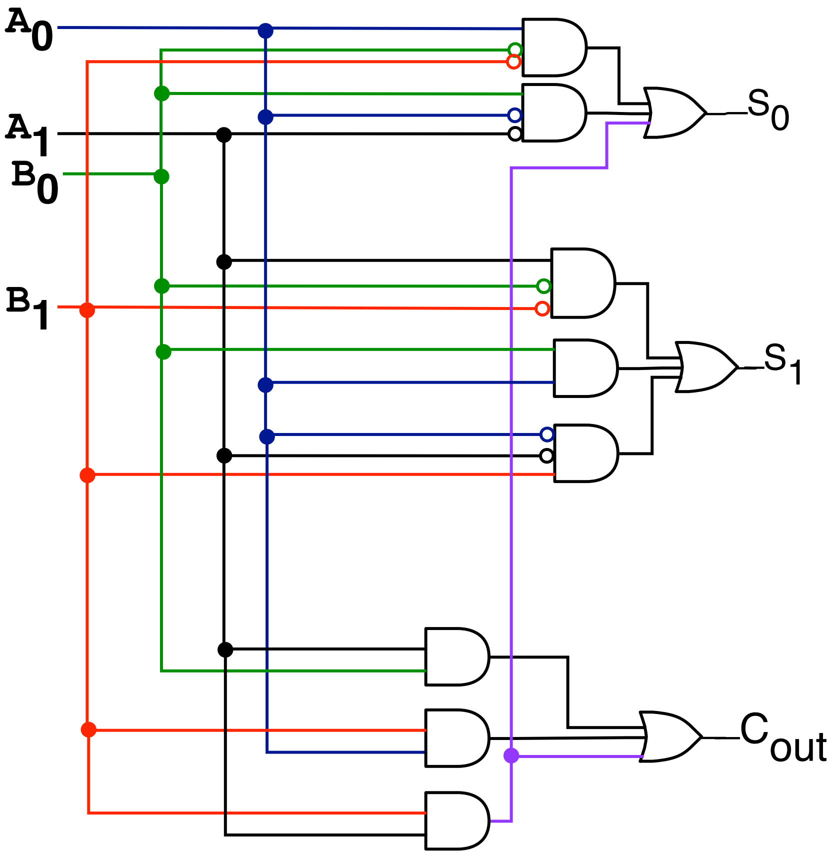
A ternary half-adder adds two **base 3** digits together, producing a sum digit and a carry-out bit. To represent ternary digits 0, 1, and 2, we will use bit pairs A_1A_0 with values 00, 01, and 10, respectively. Below are two examples of the ternary half-adder in action. On the left you see the inputs and outputs associated with adding 2+2. Each 2 is represented by the 2-bit pair 10. The sum, 4 cannot be expressed in a single ternary digit which can only express values 0, 1, and 2. The sum is 1 (represented as 01) and the carry-out is 1. On the right, you see 1 + 0.

Draw the internal circuit of the ternary half-adder. You may use any standard logic gates with any number of inputs. You do not need to minimize the circuit.





B_1	B_0	A_1	A_0	S_1	S_0	C
0	0	0	0	0	0	0
0	0	0	1	0	1	0
0	0	1	0	1	0	0
0	1	0	0	0	1	0
0	1	0	1	1	0	0
0	1	1	0	0	0	1
1	0	0	0	1	0	0
1	0	0	1	0	0	1
1	0	1	0	0	1	1



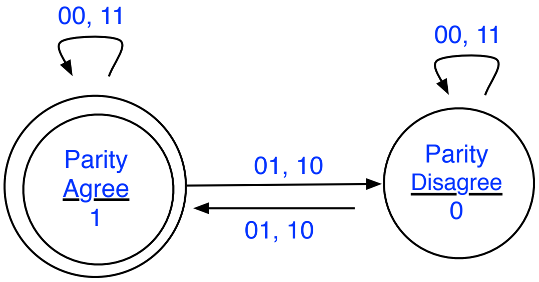
8. Finite State Machines [30 pts]

(a) [12 pts] You must design a finite state machine that reports parity [1] agreement between inputs A and B. Your FSM takes two inputs, A and B, and generates one output Z. The output Z should be 1 when A and B have both had an even number of 1's as inputs OR when A and B have both had an odd number of 1's as inputs. A sample series of inputs and the expected output is shown below. The first inputs and output combination appear on the far right of the sequence.

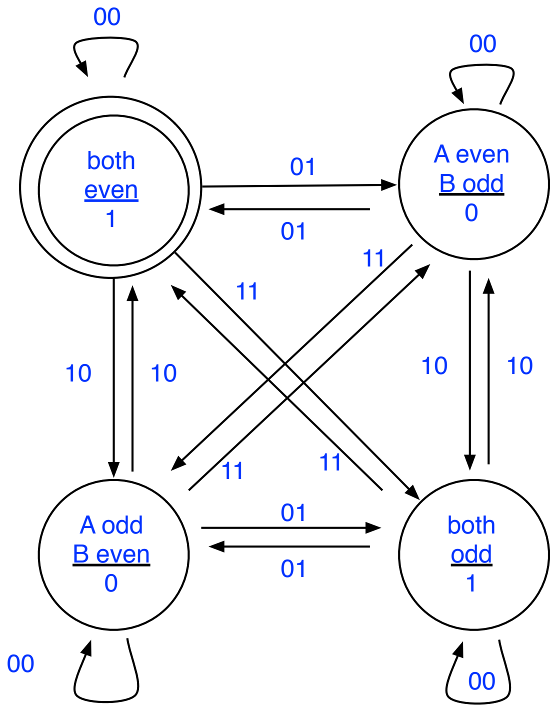
A = ...111101
B = ...010100
Z = ...011000

Draw a state transition diagram for this FSM. Remember to label your initial state with a double-circle. **Use a minimal number of states to receive full credit.**

[1] The parity of a number is whether it is even or odd.



2 States Solution:
This solution has
a minimal number of states
and will earn all 12 points.



3 States Solution:
Not shown, but it exists
and will earn at most 10 points.

4 States Solution:
This solution does not have
a minimal number of states
and will earn at most 10 points.

Consider the following State Transition Table:

Current State		Input	Next State		Output
S_1	S_0	I	S'_1	S'_0	O
0	0	0	1	0	0
0	0	1	1	0	1
0	1	0	1	0	0
0	1	1	1	1	1
1	0	0	0	0	0
1	0	1	0	1	0
1	1	1	1	0	0
1	1	0	1	0	0

(b) [2 pts] The state transition diagram associated with this table:

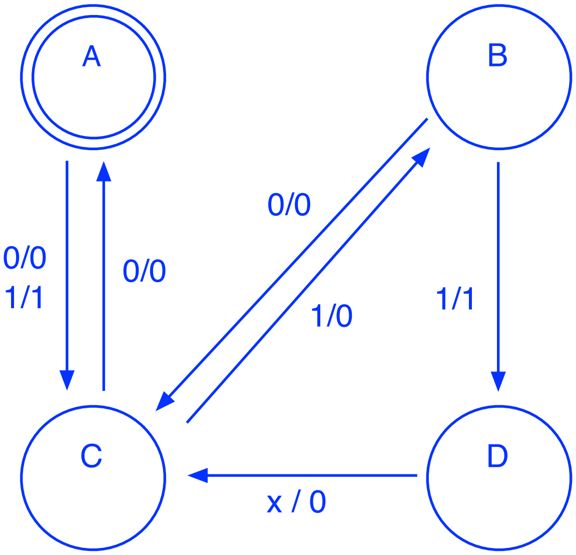
(a) Must be a Moore Machine.
(b) Must be a Mealy Machine. Your Answer: ________
(c) Could be either a Moore or a Mealy Machine.
(d) It is not possible to know based on the table provided.

Correct Answer: B.

(c) [1 pt] Explain why in *one* sentence.

The output is dependent on both the current state and the input (the first two rows of the table show this).

(d) [8 pts] Draw the state transition diagram associated with the table. Use the following state encoding **A = 00, B = 01, C = 10, D = 11**.



(e) [2 pts] A TA tells you to use a one-hot state encoding. Specifically, she tells you not to use the encoding **A = 00, B = 01, C = 10, D = 11** but instead **A = 0001, B = 0010, C = 0100, D = 1000**. Complete the State Transition Table given the new encoding:

Current State	Input	Next State	Output
S_3 S_2 S_1 S_0	I	S'_3 S'_2 S'_1 S'_0	O
0 0 0 1	0	0 1 0 0	0
0 0 0 1	1	0 1 0 0	1
0 0 1 0	0	0 1 0 0	0
0 0 1 0	1	1 0 0 0	1
0 1 0 0	0	0 0 0 1	0
0 1 0 0	1	0 0 1 0	0
1 0 0 0	1	0 1 0 0	0
1 0 0 0	0	0 1 0 0	0

(f) [4 pts] Provide a Boolean equation for the S'_2 Next State bit. Your equation should be minimized in order to receive full credit.

$S'_2 = S_0 + S_1\overline{I} + S_3$

(g) [1 pt] Typically, the combinational logic for next state calculations is simpler when the states are encoded in a one-hot encoding. This simplicity comes at a small hardware price, however. What is that price?

You need twice as many registers to save the state. (Before: 2 bits, After: 4 bits)

9. Assembly [8 pts]

BEFORE

	Register File
r0	0
r1	
r2	
r3	7
r4	
r5	
r6	
r7	100

fill these boxes
with **decimal** values

	Memory
11	
10	
09	
08	
07	x0d
06	x0c
05	x0b
04	x0a
03	
02	
01	
00	

fill these boxes
with **hex** values

Show the register and memory contents
after executing the following 4 instructions
on a little endian machine:

```
addi r3, r0, 3
sll r7, r3, 2
sw r3, 5(r0)
lw r5, 4(r0)
```

Only the part under this line will be graded. Show every known value.

AFTER

	Register File
r0	0
r1	
r2	
r3	3
r4	
r5	778
r6	
r7	12

	Memory
11	
10	
09	
08	x00
07	x00
06	x00
05	x03
04	x0a
03	
02	
01	
00	

0011 0000 1010
= 512 + 256 + 8 + 2
= 778

10. Performance [13 pts]

Consider a program with the following dynamic instruction counts being executing on a 1 GHz (1 cycle = 1 ns) processor with the following CPI for each instruction type:

- ALU: 40 million, 1 CPI
- Loads: 30 million, 4 CPI
- Stores: 20 million, 1 CPI
- Branches: 10 million, 2 CPI

(a) [2 pts] What is the CPI of this machine running this program?

.4 x 1 + .3 x 4 + .2 x 1 + .1 x 2=
.4 + 1.2 + .2 + .2 =
= 2

You decide that the load-store architecture of this processor is not efficient. You create a new instruction which can add two values found in memory and store the result back in memory. When you re-compile the program, the new instruction mix is:

- MemAdd: 10 million, 5 CPI
- ALU: 30 million, 1 CPI
- Loads: 10 million, 4 CPI
- Stores: 10 million, 1 CPI
- Branches: 10 million, 2 CPI

In other words, 10 million adds, 20 million loads, and 10 million stores have been replaced by 10 million `memAdd` instructions, which take 5 cycles. Supporting the new memAdd instruction slows down the clock; the frequency of the new processor is 500 MHz (0.5 GHz, 1 cycle = 2 ns).

(b) [5 pts] Which is faster: the original processor or the new processor? (2 pts) Justify your answer. Be concrete. (3 pts)

The original processor is faster.

The weighted CPI of the new processor is:
.1 x 5 + .3 x 1 + .1 x 4 + .1 x 1 + .1 x 2=
.5 + .3 + .4 + .1 + .2 = 1.5
But the CPI would need to be less than 1 to compensate for the the fact that the clock got 2x slower. Since this isn't the case, the original processor is faster.

A slightly slower but straightforward way to answer this question is to compare execution times.

(c) [6 pts] You want to compare the performance of processor A and processor B running program X. For each performance metric below, what must be true in order for this metric to be the correct metric?

To use metric: The following must apply:
(Check *all* that must apply.)

execution time

A & B have the same clock speed.
A & B have the same ISA.
XX none of the above.

MIPS

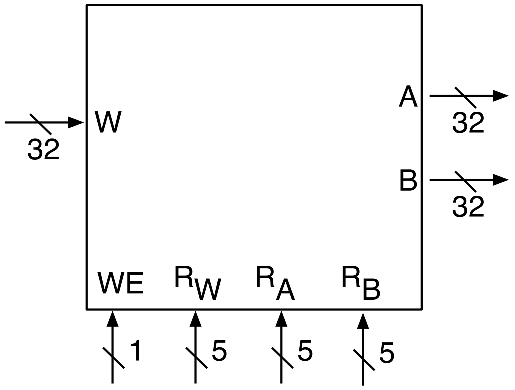
A & B have the same clock speed.
XX A & B have the same ISA.
none of the above.

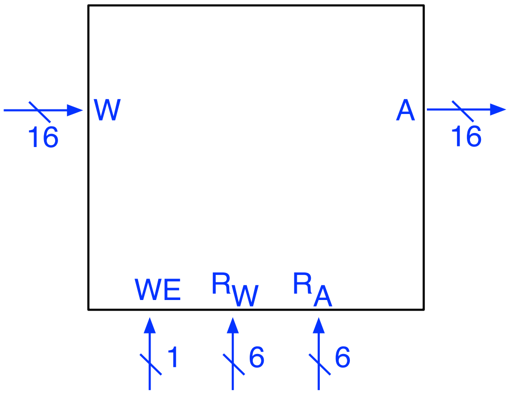
CPI

XX A & B have the same clock speed.
XX A & B have the same ISA.
none of the above.

11. Storage [6 pts]

Below on the left is the interface for the MIPS register file. It is a dual-read, single-write register file with 32 registers; each register holds a 32-bit word. Next to this interface, draw the interface for a new register file that is a single-read, single-write register file with 64 registers; each register holds a 16-bit word. Make your new interface as detailed as the original.


W
32
A
32
B
32
WE
R_W
R_A
R_B
1
5
5
5


W
16
A
16
WE
R_W
R_A
1
6
6