Prelim 1, Version 1
Computer Science 3410, Cornell University
Spring 2017, Professor Bracy 28 March 2017

- Write your NetID at the top of each physical page of the exam.
- On the **scantron** form, please **write and bubble in** your Last Name, First Name, and your ID Number (this is *entirely numeric*; it is *not* your NetID).
- **Complete the above steps or face -2 pt penalty.**
- Please turn off and stow away all electronic devices. You may not use them for any reason for the entirety of the exam. Do not bring them with you if you leave the room temporarily.
- This is a closed book and notes examination. You may use the 3-sided reference provided.
- To receive partial credit you must show your work. If you believe a question is open to interpretation, then please ask us about it! Please state any assumptions you make.
- Questions to be answered using the scantron form are clearly marked. For example **(Scantron-10)** should be answered by bubbling in question 10 on the scantron form.
- Scantron questions have **only one** correct answer. Multiple answers will receive 0 points.
- There are 6 problems and 15 pages. Make sure you have the whole exam. You may use the back-side of each page as scratch paper, but it will *not* be graded.
- You have **120 minutes** to complete 116 points. Use your time accordingly.

Problem	Topic	Points	Score
1	Multiple Choice	30	
2	Processor Design	22	
3	Simplify your Life!	15	
4	A Number from 0 to 9	20	
5	Finite Soda Machine	9	
6	Who you gonna call?	20	
Total		116	

Your Name _____

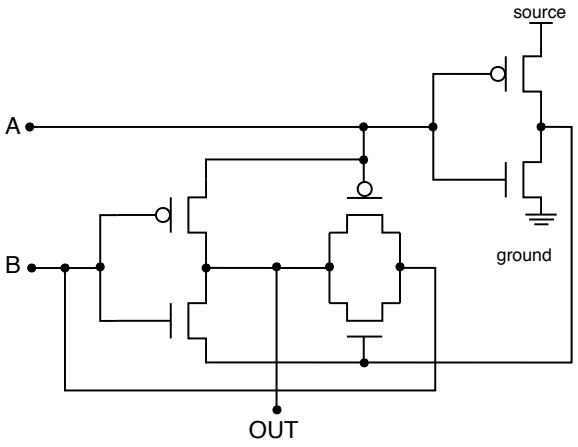
Your NetID _____

NetID: _____

Answer these questions on the scantron form. THIS PAGE WILL NOT BE GRADED.

1. Multiple Choice [30 pts]

(Scantron-1) [2pts] CMOS. What gate is implemented by this CMOS schematic?



A. AND
B. NAND
C. NOR
D. XOR
E. XNOR

(Scantron-2) [2pts] Encoding Options. You are given a 4-digit binary number and told that it is encoded as either unsigned binary, sign-magnitude or signed 2's complement. Which of the following would allow you to know *for certain* what the encoding is, for any value of the number. You are given...

A. the value of the number.
B. the result of adding 1 to the number (in the unknown encoding).
C. the result of subtracting 1 from the number (in the unknown encoding).
D. the 8-bit sign extended version of the number (in the unknown encoding).
E. None of these will work for any value of the 4-digit number.

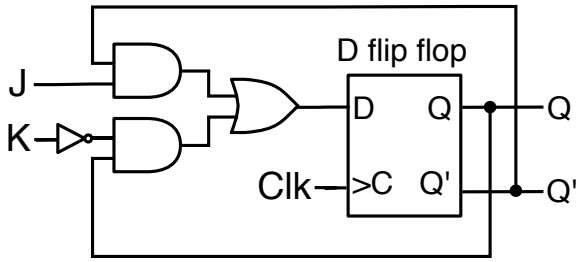
(Scantron-3) [2pts] Fractional Binary Encoding is a binary encoding with a decimal point. The columns to the right of the decimal point continue in the pattern of decreasing powers of 2: the column immediately to the right of the point is the 2^{-1} column (value 1/2), next to that is the 2^{-2} column (value 1/4). Given your understanding of two's complement, what is the value of the 4-bit **two's complement fractionally binary encoded** number `10.01`:

A. -2.25
B. -1.75
C. 1.50
D. 1.75
E. 2.25

NetID: _____

Answer these questions on the scantron form. THIS PAGE WILL NOT BE GRADED.

No Kidding, Flip Flops. [8 pts] This is a component known as JK Flip Flop:



A JK Flip Flop is a 2-input variation on the standard D Flip Flop, which only takes 1 input. (Both components also take a clock.) The value of the output, Q, is updated on the rising edge of the clock. Q(t) is the output value of the JK Flip Flop at time t.

Inspect the JK Flip Flop and define its behavior by completing all four rows of the truth table. Each row is a separate scantron question. For each row of the truth table, you may choose one of the answers (A-E) on the left:

	J	K	Q(t+1)
(Scantron-4) [2pts]	0	0	
(Scantron-5) [2pts]	0	1	
(Scantron-6) [2pts]	1	0	
(Scantron-7) [2pts]	1	1	

A. 0
B. 1
C. Q(t)
D. Q'(t)
E. Clk

NetID: _____

Answer these questions on the scantron form. THIS PAGE WILL NOT BE GRADED.

(Scantron-8) [2pts] Little Endian. Suppose register `r1` has the value `0x01234567` and `r2` has `0x89ABCDEF`. What byte value is located at memory address **3** after the following store instructions are executed on this **little** endian machine? Assume that memory is byte addressed, memory accesses need not be word-aligned, and memory has been initialized to all 0's.

```
sw r1 0(r0)
lw r3 3(r0)
sw r2 0(r3)
```

A. ab

B. cd

C. 01

D. 67

E. None of the above

(Scantron-9) [2pts] Big Endian. Suppose register `r1` has the value `0x01234567` and `r2` has `0x89ABCDEF`. What byte value is located at memory address **3** after the following store instructions are executed on this **big** endian machine? Assume that memory is byte addressed, memory accesses need not be word-aligned, and memory has been initialized to all 0's.

```
sw r1 0(r0)
lb r3 3(r0)
sh r2 0(r3)
```

A. 00

B. 01

C. 67

D. ab

E. None of the above

NetID: _____

Answer these questions on the scantron form. THIS PAGE WILL NOT BE GRADED.

(Scantron-10) [2pts] ISA Design. Which is **not** true about a CISC ISA? A CISC ISA...

A. ...is harder for a compiler to optimize than a RISC ISA.

B. ...yields executables with fewer dynamic instructions than a RISC ISA.

C. ...supports more instruction formats than RISC ISA.

D. ...is currently the most successful ISA in the low-power / embedded market.

E. ...better supports hand assembling than a RISC ISA.

(Scantron-11) [2pts] Conditional Instructions. As discussed in class, Arm v7 supports *conditional instructions*, which are always fetched and decoded, but are executed conditioned upon the value in the 4-bit condition register. For example, `SUBGT Ri, Ri, Rj` will perform a subtraction only if the greater-than bit in the condition register is a 1. Conditional instructions enable one to transform code that once had an `if (test) {if-block} else {else-block}` structure into straight-line code (without branches). What statement about conditional instructions is **false**?
Conditional instructions...

A. ... complicate the implementation of the processor.

B. ... remove control dependences between instructions.

C. ... are particularly beneficial when the if-block and else-block are quite long.

D. ... are particularly beneficial when the `if(test)` is taken with a 50% likelihood.

E. ... are not supported in Arm v8.

(Scantron-12) [2pts] Errors. Which of the following errors could not plausibly be found by the suggested stage in the compilation process of a C program?

A. Compiler: finds multiple initializations of the same global variable

B. Linker: finds multiple initializations of the same global variable

C. Preprocessor: can't find definition of a `#define`

D. Linker: can't find definition of a `#define`

E. Assembler: cannot locate a target label

NetID: _____

Answer these questions on the scantron form. THIS PAGE WILL NOT BE GRADED.

(Scantron-13) [2pts] Compilation Process. Which of the following steps must be performed *each time* a program is run?

A. Assembling

B. Loading

C. Compiling

D. Linking

E. A-D must *all* be performed each time a program is run.

(Scantron-14) [2pts] C Basics. What value is printed in the last line of `main`?

```
void set1(int val1, int *val2) {
   *val2 = val1 + 6;
}

void set2(int val1, int *val2) {
   val1 = *val2 + 3;
}

int main() {
   int x = 5;
   int *y = &x;
   set1(x, y);
   set2(x, y);
   printf(%d\n, x);
}
```

A. 5

B. 8

C. 11

D. 14

E. It depends on the address of x.

(Scantron-15) [2pts] Function Calls. Which of the following statements is *true*?

A. The compiler can decide whether a caller-saved register needs to be saved by the caller because it can see whether the caller will need the value later.

B. The compiler can decide whether a callee-saved register needs to be saved by the callee because it can see whether the caller will need the value later.

C. Because the stack grows and shrinks as functions are called and returned, there is always enough room on the stack for the next function to be called.

D. Malloc allocates space for new data structures on the stack.

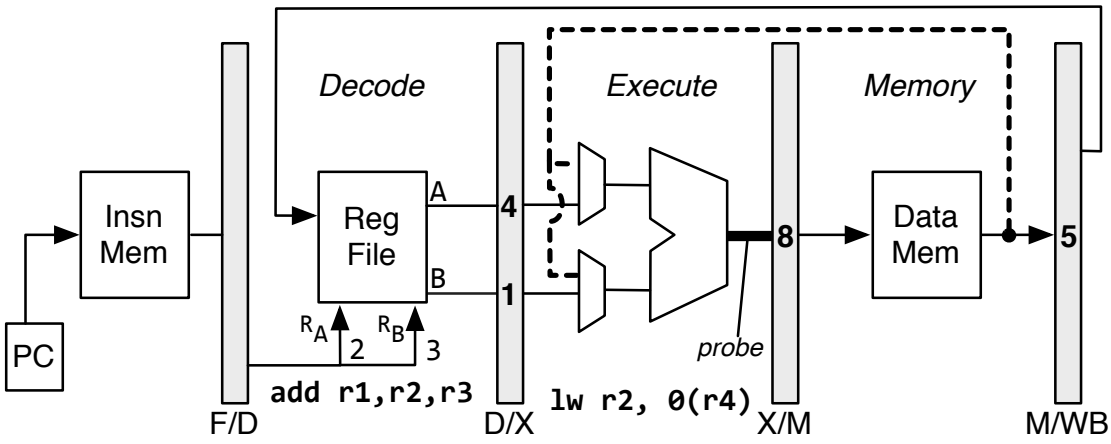
E. No function can execute correctly without its own stack frame.

NetID: _____

Answer these questions on the scantron form. THIS PAGE WILL NOT BE GRADED.

2. Processor Design [22 pts] Consider a **Baseline** Processor that stalls in response to a load-use hazard. Now consider 3 alternate design options which would yield processors A, B, and C.

Option A: Forward the result of the load (in MEM) to its use (in EX). An approximate implementation of *Processor A* (showing only a few high-level changes that would be required) is shown below with a dashed line:



During cycle n-1, **add** is in Decode and **lw** is in Execute.
At the end of cycle n-1, the **bold** values are stored into the pipeline registers shown.

(Scantron-16) [2pts] Load-Use Bypass. Forwarding values directly from Memory to the Execute stage as shown above...

A. Removes the need for the *use* in the load-use dependency to stall.
B. Adds one too many possible inputs to the ALU.
C. Will cause the pipeline register to have the wrong value.
D. Halves the frequency of the processor.
E. Both A and D

(Scantron-17) [2pts] Load-Use Bypass Values. Suppose that the value 6 is stored in memory at address **8** and the value **5** was read from memory at cycle $n-1$. If you put a probe on the thick wire exiting the ALU during cycle n, in which `add` is in EX and `lw` is in MEM. What possible values could you see on the wire *in order* during cycle n?

A. 8,6,7
B. 8,5,7
C. 8,5,10
D. 5,7
E. 5,10

Answer these questions on the scantron form. THIS PAGE WILL NOT BE GRADED.

Option B: Introduce a **load delay slot**. Just as a *control* delay slot frees the processor from having to decide whether to zap and flush the instruction immediately after a control instruction, the *load* delay slot frees the processor from having to decide whether to stall the instruction immediately following a load.

The original **Baseline Processor** would stall in response to a load-use hazard, which happened for 1/3 of loads. The benchmark is 30% loads, 60% integer operations, and 10% branches which can be executed in 3, 1, and 2 cycles, respectively.

Processor B is the processor that has a load delay slot. **Processor B** is running the same benchmark modified to account for the **load delay slot** by adding a `NOP` after each load instruction. You should consider this NOP as a 1 cycle stall which all load instructions experience (and not a new instruction added to the benchmark).

(Scantron-18) [2pts] Original CPI. What is the CPI of the Baseline Processor?

A. 1.8

B. 1.7

C. 0.7

D. 1.0

E. 1.6

(Scantron-19) [2pts] Modified CPI. What is the CPI of Processor B?

A. 1.8

B. 1.7

C. 2.0

D. 2.1

E. 1.9

For each of the following statements, is the statement:

A. True

B. False

C. Cannot say with the given information.

(Scantron-20) [2pts] Option B changes the ISA.

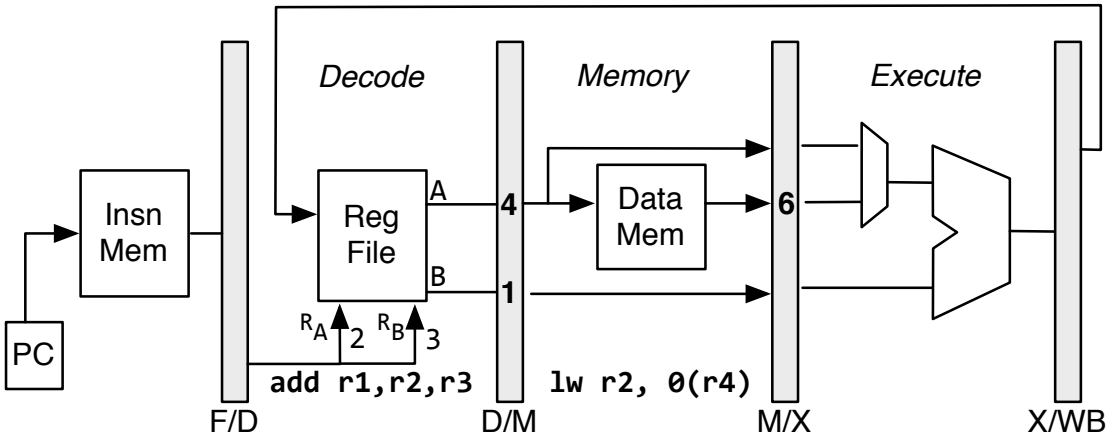
(Scantron-21) [2pts] Option B supports a CPU with a faster clock and fewer transistors.

(Scantron-22) [2pts] To preserve the semantics of the original benchmark, the compiler must insert NOPs after every load.

NetID: _____

Answer these questions on the scantron form. THIS PAGE WILL NOT BE GRADED.

Option C: Swap the execute and memory stages in the pipeline. An approximate implementation of *Processor C* (showing only a few high-level changes that would be required) is shown below:



During cycle n-1, **add** is in Decode and **lw** is in Memory.
At the end of cycle n-1, the **bold** values are stored into the pipeline registers shown.

For each of the following statements, is the statement:

A. True

B. False

C. Cannot say with the given information.

(Scantron-23) [2pts] Option C changes the ISA.

(Scantron-24) [2pts] Option C removes the need for the *use* in the load-use dependency to stall.

(Scantron-25) [2pts] Option C introduces a new reason for the pipeline to stall, but that can be avoided by updating the forwarding/bypassing logic of the processor.

(Scantron-26) [2pts] Option C changes the dynamic instruction count of the Processor C with respect to the original Baseline.

3. Simplify your Life! [15 pts]

   Consider the following Boolean representation of a combinational circuit:

   $\overline{(a + \bar{b})} + \bar{a}\bar{b}c + a\bar{b}\bar{c}$

   (a) [3 pts] What is the minimal hardware cost of this circuit in its current (unsimplified) form?

   _____ inverters
   _____ 2-input AND gates
   _____ 2-input OR gates

   (b) [6 pts] Use Boolean Algebra to simplify the above equation. Your goal is to reduce the hardware cost as much as possible. You do not need to say which rule you are using, but you should only use one rule per line in order to get full credit for a correct answer.

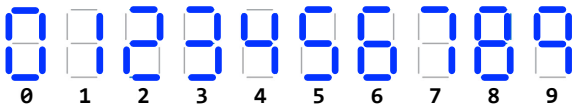
(c) [6 pts] Using the K-Map provided, what is the most simplified function it represents? Your answer should be in the form of a Boolean equation.

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

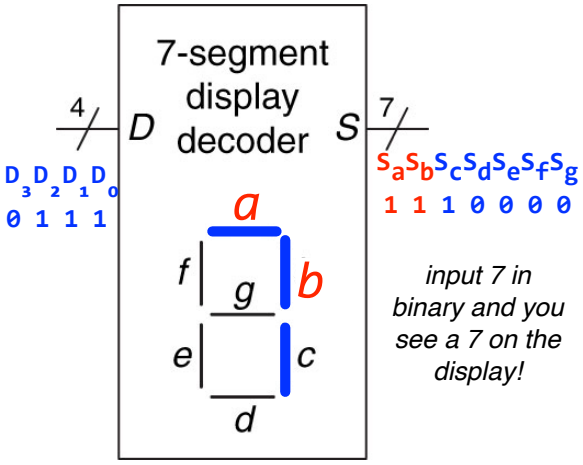
NetID: _____

4. A Number from 0 to 9 [20 pts] (10 pts correctness, 8 pts minimization, 2 pts clarity)

A **7-segment display decoder** takes a 4-bit data input, the binary number $D_3D_2D_1D_0$, and produces seven outputs that control a display which shows a single digit from 0 to 9:



The seven outputs are segments S_a through S_g, as shown to the right. **On the next page**, draw the circuit that defines just outputs S_a and S_b. Illegal input values (10−15) should produce a blank readout. You may use only INVERTERS and 2-input AND and OR gates.



You may use this page for your work; only your circuit (on the next page) will be graded.

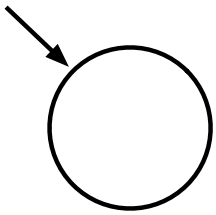

D3
Sa
D2
Sb
D1
D0

5. Finite Soda Machine [9 pts]

Design a soda machine dispenser for Gates Hall. Sodas are subsidized by the ACSU, so they cost only 25 cents. The machine accepts dimes and quarters. *At most one coin* is inserted on each cycle. Only after enough coins have been inserted, it dispenses the soda and returns any necessary change. The FSM input is a 2-bit signal, QD; the Q indicates whether a Quarter was inserted, D indicates whether a Dime was inserted. The output is a 4-bit signal SQDN (Dispense**Soda**, Return**Quarter** (25 cents), Return**Dime** (10 cents), Return**Nickel** (5 cents)). When the FSM reaches 25 cents, it asserts Dispense**Soda** and returns change (only if change is due). Then it should be ready to start accepting coins for another soda. For example, the *hypothetical* transition **10/1001** means that someone inserted a quarter and in response they received a soda and a nickel.

Complication: the machine can return at most 1 of each coin type when it returns change. (It can return 1 nickel, but not 2 nickels in a particular cycle.) If ever more than 1 coin type is due (for example, 2 nickels), the machine returns one of the coins and applies the second coin towards the purchase of the next soda. It should only employ this workaround when it absolutely has to and you should return as much change as possible.

Design **a Mealy Machine state transition diagram** for the soda machine. The start state has been drawn for you.



6. Who you gonna call? [20 pts] Consider the following function in C:

```
void f2(int n) {
    int x;
    int y = n * 2;
    int z = n + 1;
    f3(&x, n);            // 'JAL f3' is the last line of your answer
    return y + x + z + n; // you do NOT need to write the MIPS for this line
}
```

Function `f1` calls function `f2` which calls function `f3`, using the 3410 Calling Conventions.

Write the MIPS assembly for `f2`, beginning with the prologue (assume that `f1`'s `JAL f2` was just executed by the processor) and **ending with** `JAL f3`.

Further requirements: Store `y` in `t1`. Store `z` in `s1`. You decide where to store `x`.

To be eligible for partial credit, a line must have a comment, explaining what the line accomplishes.