Fall 2010 CS61C Final

Your Name: _____

Your TA: Andrew Michael Conor Charles

Login: cs61c-___

This exam is worth 186 points, or about 20% of your total course grade. The exam contains 10 questions.

This booklet contains 14 numbered pages including the cover page, plus the 2 pages for the green card. Put all answers on these pages, please; don't hand in stray pieces of paper.

Question	*Points (Minutes)*	*Score*
1. Justified	**30 points (15 minutes)**	
2. Virtuosity	**14 (7)**	
3. Mo' Cache, Mo' Problems	**24 (12)**	
4. My Secret Cache	**34 (17)**	
5. Heaven's Gate	**8 (4)**	
6. Off the Map	**20 (10)**	
7. Pay It Forward	**12 (6)**	
8. Bigger, Stronger, Faster	**10 (5)**	
9. Altered States	**10 (5)**	
10. Three's Company	**24 (12)**	
Total	**186 points (93 minutes)**	

1. **Justified**. True or False with Justification

Circle whether each statement in questions a) to g) is True or False, and ***give a one sentence justification***. We reserve the right to dock points for long-winded answers. Be concise! If you feel like you can give a clear justification in just one word, please do!

a. Designing a processor with dynamic multiple issue places extra burden on the compiler author to ensure the correctness of emitted code.
TRUE / FALSE
Justification:

b. As a result of hitting the Power Wall, Moore's Law no longer holds.
TRUE / FALSE
Justification:

c. Increasing the depth of a pipeline tends to increase performance primarily by decreasing the amount of time it takes on average to execute a particular instruction.
TRUE / FALSE
Justification:

d. In a single-cycle MIPS CPU during the execution of a jump instruction, the adder block inside the ALU always computes the sum of its inputs.
TRUE / FALSE
Justification:

e. After enough time, a circuit composed of combinational logic without feedback will always show the same output for a given set of inputs.
TRUE / FALSE
Justification:

f. For pipelined execution, if multiple exceptions happen in the same clock cycle for different instructions, the hardware should set the EPC to address of the earliest issued instruction.
TRUE / FALSE
Justification:

g. To get good performance from future microprocessors in the next 5 to 10 years, SIMD via successors to SSE4 instructions could be as important to use as MIMD via more cores.
TRUE / FALSE
Justification:

h. A particular memory access results in a page fault but not a protection error. Which of the following statements about this access is ***always*** true? Circle the answer.

Always true / Not always true There was a TLB miss.
Always true / Not always true The cache(s) will miss.
Always true / Not always true Space for the required page is allocated somewhere on disk.

i. Which of the following could be enabled by adding more ALUs to a processor design? (Check all that apply.)
____ More structural hazards
____ More issue slots
____ Wider SIMD width
____ Larger CPI

j. Multiple Choice. (Pick one.) Virtual machines are used in Cloud Computer by Amazon to
I. Protect users from each other
II. Offer more price points
III. Simplify introduction of new and faster hardware
A. None of the above
B. I only
C. II only
D. III only
E. I and II
F. II and III
G. I and III
H. I, II, and III

2. **Virtuosity**
   A computer has a 32-bit virtual address with ***8 KB pages*** and a 16 KB cache with 32 Byte blocks that is 2-way set associative

   IV. Show how the 32-bit address is divided into the virtual page number and page offset, showing the number of bits in each field:

   31 0

   V. Show how the 32-bit address could be divided into the cache tag, cache index, and block offset, showing the number of bits in each field:

   31 0

   VI. Given these layouts, must the lookup in the TLB to translate from virtual address to physical address be done ***sequentially*** before access the cache?
   Why or why not?

3. Mo' Cache, Mo' Problems

a) A naive hardware developer designs the cache protocol for a 2 CPU system, each CPU with a 2 block direct mapped cache, 1 byte blocks. Memory addresses are 32 bits (byte addressed). The cache policies are Allocate on write, and Write through.

Assume that both caches start out with all blocks invalid. Which of the following access patterns, ***executed independently from one another***, will always yield correctly updated results? Assume that each Read/Write finishes completely during its time period.

	CPU	Time 1	Time 2	Time 3	Correct?
i.	1	Write 0x0	---	Read 0x0	
	2	---	Write 0x0	---	
ii.	1	Write 0x0	---	Read 0x1	
	2	---	Write 0x1	---	
iii.	1	Write 0x0	---	Read 0x0	
	2	---	Write 0x2	---	
iv.	1	Read 0x0	---	Read 0x2	
	2	---	Write 0x2	---	
v.	1	Read 0x0	---	Read 0x0	
	2	---	Write 0x0	---	

b) What feature could you add to this cache protocol to make it work in all cases? Be specific; your revision should describe a particular behavior.

4. My Secret Cache

The average memory access time (AMAT) is:

AMAT = Hit $Time_{L1}$ + Miss $Rate_{L1}$ x Miss $Penalty_{L1}$

(a) Write down the AMAT equation for a three level cache (use HT for Hit Time, MR for Miss Rate, and MP for Miss Penalty):

(b) Given the following specification:

For every 1000 CPU-to-memory references
- 40 will miss in L1$;
- 20 will miss in L2$;
- 10 will miss in L3$;

L1$ hits in 1 clock cycle;
L2$ hits in 10 clock cycles;
L3$ hits in 100 clock cycles;
Main memory access is 400 clock cycles;
There are 1.25 memory references per instruction; and
The ideal CPI is 1.

Answer the following questions:

(i) What is the local miss rate in the L2$?

(ii) What is the global miss rate in the L2$?

(iii) What is the local miss rate in the L3$?

(iv) What is the global miss rate in the L3$?

(v) What is the AMAT with all three levels of cache?

(vi) What is the AMAT for a *two-level* cache *without L3$?*

(vii) What is the AMAT for a ***one-level*** cache ***without L2$ and L3***$?

(viii) What is the average memory stalls per reference in the system of question (v)?

(ix) What is the average memory stalls per reference in the system of question (vi)?

(x) What is the average memory stalls per reference in the system of question (vii)?

(xi) What is the average memory stalls per instruction in the system of question (v)?

(xii) What is the average memory stalls per instruction in the system of question (vi)?

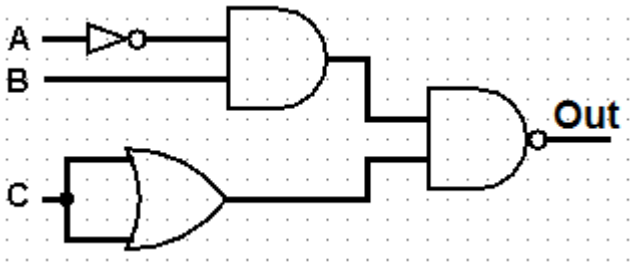
(xiii) What is the average memory stalls per instruction in the system of question (vii)?

(xiv) What is the performance of the system of question (vii) versus that of question (v)? (Long-division challenged can give just the ratio.)

(xv) What is the performance of the system of question (vi) versus that of question (v)? (Long-division challenged can give just the ratio.)

5. Heaven's Gate

What is the truth table for the following circuit?
(It might be simpler to first write the equation)



A	B	C	Out
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

6. Off the Map

Given:

```
typedef struct llnode {
     struct llnode *next;
     int contents;
} llnode;
```

Implement `map` in C. `map` should take a pointer to an `llnode` and a `map_fun`, which is just a pointer to a function taking one int argument and returning int. It should return a pointer to a **new** linked list, where the contents of each of the new nodes is equal to the value returned by the map function called on the contents of the corresponding old node. Don't forget to fill in the return value correctly (pointers to function are tricky so we've done the arguments for you). Assume `llnode` lists are terminated by a `NULL` pointer. You should **not** modify any part of the list identified by `head`. You may assume `map_fun` has been properly declared by a `typedef` somewhere else.

```
_______________ map(llnode *head, map_fun f)
{
  if( ____________________________________ ) {

    return ____________________;
  }

  ____________________________________________________________;

  ____________________________________________________________;

  ____________________________________________________________;

  return ____________________________________________________;
}
```

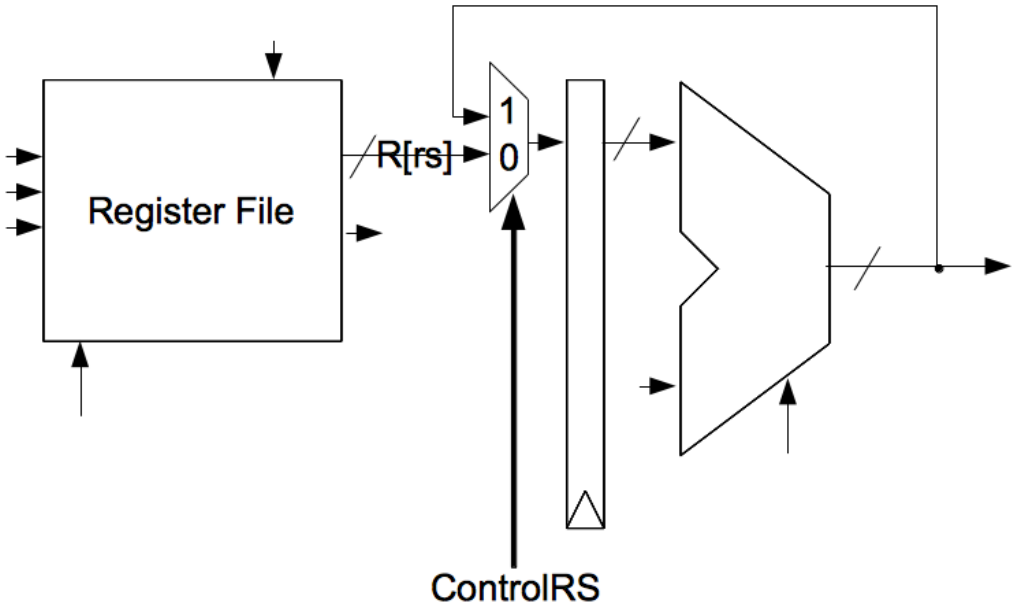
(Note that to call a function pointer, you just use it like any other function. So, in the code template given above, saying

```
int result = f(3);
```

would correctly call the function pointed to by `f` with the value 3 and put the return value into `result`.)

7. Pay It Forward

Consider the *excerpt* below of a 5-stage pipelined MIPS datapath.



a. Consider the following sequence of instructions

```
[A] srl $zero, $zero, 0
[B] addu $t0, $t1, $t2
[C] addu $t0, $t0, $t2
[D] lw $s0, 0($t3)
[E] subu $t3, $s0, $t0
```

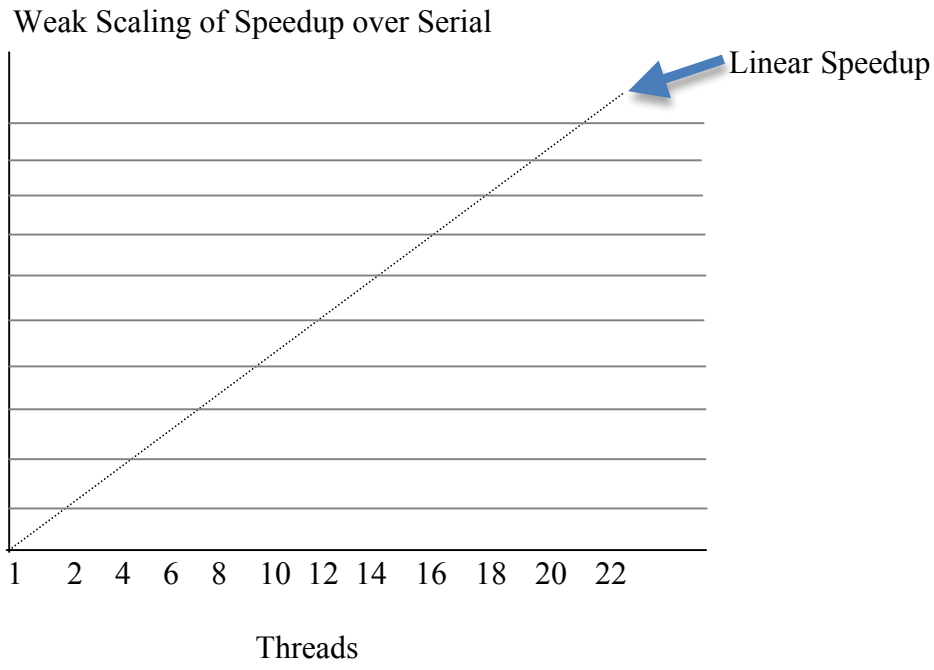
During which of these instructions' decode stages in the sequence above should ControlRS be 1 to avoid pipeline stalls? Use the labels [A], [B], …

b. Which fields of which instructions from part a does the control logic need to compute the value of ControlRS?

8. Bigger, Stronger, Faster:

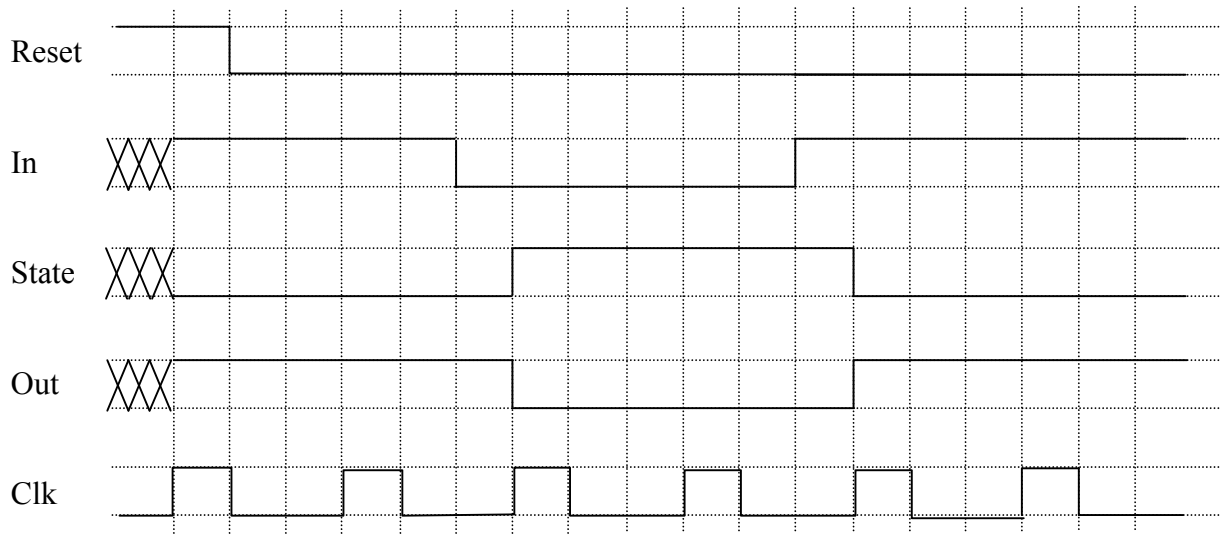
Suppose that you are running an algorithm for various problem sizes, and have obtained the data below. Sketch a weak scaling plot of parallel code performance that shows speedup over the serial implementation. ***Be sure to label the Y-axis.***

Problem Size	Gflop/s (serial)	Threads	Gflop/s (parallel)
100	5	1	5
200	5	2	10
400	5	4	19
600	5	6	25
800	5	8	35
1000	5	10	36
1200	5	12	37
1400	5	14	37
1600	5	16	38

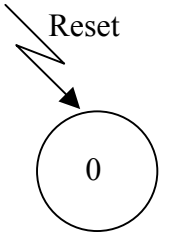


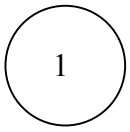
9. Altered States

Given the following timing chart, and assuming 0 set-up, hold, and propagation times, as well as a positive edge-triggered synchronous system,



Fill-in the transitions and outputs of the following State Diagram so it has the identical behavior to the timing diagram above:



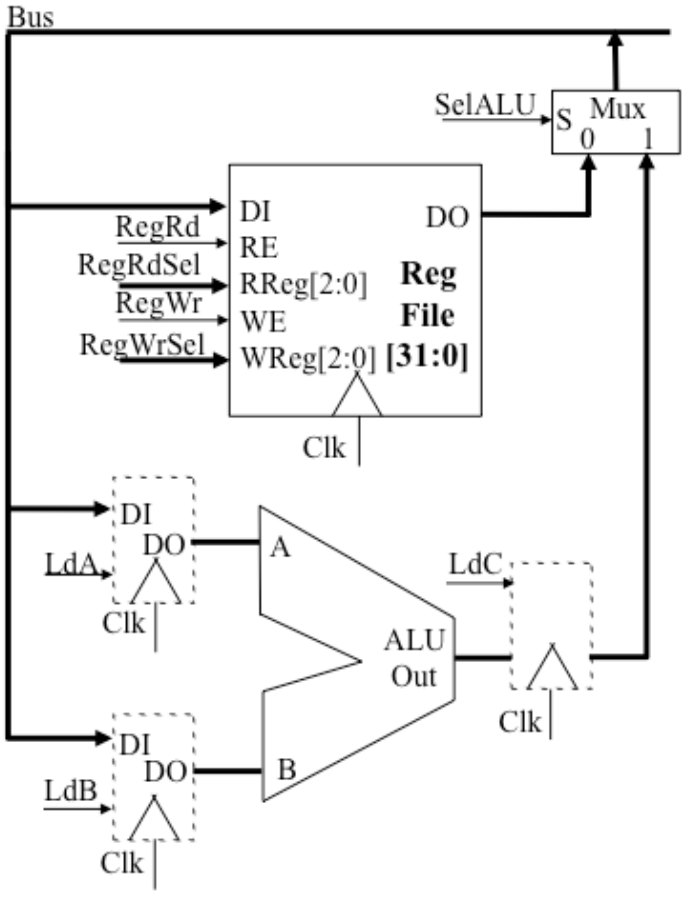


10. Three's Company

Consider the following datapath with an Arithmetic Logic Unit (ALU) and an eight-register register file organized around a single bus. The ALU is to apply add, subtract, and so on operations to its two input operands to generate an output result. The register file has an asynchronous read and a synchronous write. That is, as soon as the Read Enable (RE) is asserted, the register file selects the indicated 32-bit register and presents its value on the Data Out (DO). On the other hand, the Write Enable (WE) is sampled only on the rising edge of the clock, and only writes the indicated register from the Data In (DI) lines on the same edge that WE is asserted. The ALU and Register File share the Bus via a 32-bit wide 2:1 multiplexer. When SelALU is set to 1, the ALU path is connected to the Bus. Otherwise, the Register File path is connected to the Bus.

The datapath must support three-address instructions of the form $R_z \leftarrow R_x$ <op> R_y.

To make use of a single bus architecture, the ALU can be surrounded by one, two, or three 32-bit temporary registers, labeled A, B, and C, as shown below (the temporary registers are shown as dotted lines – the correct solution requires at least one and possibly all three of the registers):



Using the fewest of the A/B/C registers and possible clock cycles, what is the fewest number of each to implement the register transfer for the instructions of the three-address type (circle one for each):

Registers	1	2	3
Clock Cycles	1	2	3

For your answer, on the previous page cross out the registers you don't need, and fill-in the outline of the registers that you do. For each clock cycle that you need according to your answer above, write in the space below the control signals that must be asserted to implement the register transfers for the three-address instructions:

Clock Cycle 1:

Clock Cycle 2:

Clock Cycle 3: