CS61C Summer 2012 Final

Your Name: _____ **SID:** _____

Your TA (Circle): Raphael Paul Brandon Sung Roa

Name of person to your LEFT: _____

Name of person to your RIGHT: _____

This exam is worth 92 points and will count for 25% of your course grade.

The exam contains 8 questions on 14 numbered pages, including the cover page. Put all answers on these pages; don't hand in stray pieces of paper.

Question 0: You will receive 1 point for properly filling out this page as well your login on every page of the exam.

Question	Points (Minutes)	Score
0	1 (0)	
1	25 (46)	
2	10 (22)	
3	7 (14)	
4	5 (12)	
5	10 (20)	
6	14 (26)	
7	10 (20)	
8	10 (20)	
Total	92 (180)	

All the work is my own. I had no prior knowledge of the exam contents nor will I share the contents with others in CS61C who have not taken it yet.

Signature: _____

Question 1: *Potpourri – hard to spell, nice to smell…* (25 points, 46 minutes)

a) (7 points) True/False:

T F Thanks to virtual memory, processes can share the physical memory space. Unfortunately, this means that writing off the end of an array might overwrite data of another process.

T F High reliability guarantees high availability.

T F PUE does not measure the power efficiency of Warehouse Scale Computer servers

T F Amdahl's law is restricting the speed increase predicted by Moore's Law

T F We can implement the atomic test-and-set operation in MIPS using `lw` and `sw`.

T F The CPI of superscalar processors can be less than one.

T F With a 4-entry TLB, 2 physical pages, and 4 virtual pages, the TLB will never be full.

---

b) (3 points) You are running a service that helps users meet others with similar tastes in food. For each pair of users with at least one food in common, we would like to count how many foods they have in common. We'll do so by chaining two MapReduce jobs together; in other words, the output of the first is the input to the second.

Your input is a list of key value pairs of the form `(user_id, list of foods)`. Given the first map and the second reduce, define in pseudocode `Reduce1` and `Map2`. The final output should be a list of `((user_id1, user_id2), count)` where `user_id1 < user_id2`.

```
Map1(key,value):
      For v in values:
            Emit(v,key)
Reduce1(key,values):
      ____________________________

      ____________________________

      ____________________________

      ____________________________
```

```
Map2(key,value):

      ____________________________
Reduce2(key, values):
      For each v in values:
            sum += v;
      Emit(key, sum)
```

---

c) (2 points) What's the correct data word given the following SEC Hamming code: `1110011`?

_____

Your friend thinks he has a better error correction/detection scheme: for m-bits of data, a copy of those m-bits will be stored. For example, if the data bits were 1110, the code word would be 11101110.

d) (1 point) What fraction of the code words is valid? Assume m-bit data words. _____

e) (1 point) What is the minimum Hamming distance between valid code words? _____

f) (1 point) Consider each of the following independently. Circle all that this scheme can accomplish.

i. Detect any single bit error.

ii. Detect any double bit errors.

iii. Detect any errors of m-bits or less.

iv. Correct any single bit error.

v. Correct any double bit errors.

vi. Correct any errors of m-bits or less.

There's a single disk failure in a disk array (you know which disk failed) and you want to read a single page. Assume that for block-striped arrays, the page is contained within a single block.

g) (2 point) What's the fewest number of disks you have to read from if the array were:

i. RAID 1 with 2 disks _____

ii. RAID 5 with 4 disks _____

h) (2 point) What's the greatest number of disks you have to read from if the array were:

i. RAID 4 with 4 disks (including parity) _____

ii. RAID 5 with 4 disks _____

Assume we have a 1 GHz processor using 4 KiB pages and an attached hard drive spinning at 6000 revolutions per minute, with a seek time of 3 ms, an average page throughput of 0.4 MB/s (remember, data rates use SI prefixes), and negligible controller overhead.

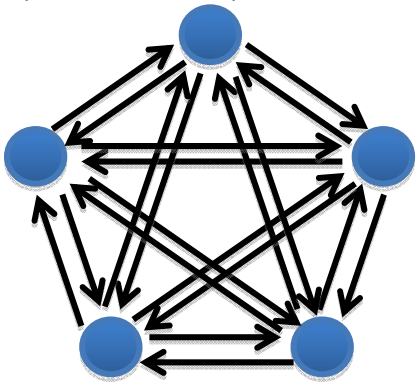
i) (2 point) What is the transfer time for a page?

_____

j) (2 points) Assume the processor polls the disk. Polling operation takes 100 clock cycles and the disk transfers data in 4 B chunks. What percentage of processor time is spent polling?

_____

k) (2 points) In a 5-state FSM with 4 arrows coming out of each state, as shown below, how many possible 1-bit output functions are there? Answer in IEC format.

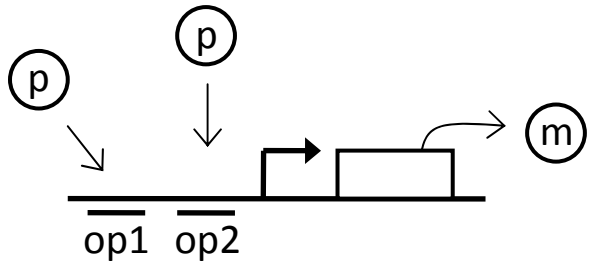


_____

Question 2: *Biological STATE-ment* (10 points, 22 minutes)

You didn't think you'd get through this test without seeing something bio-related, did you?

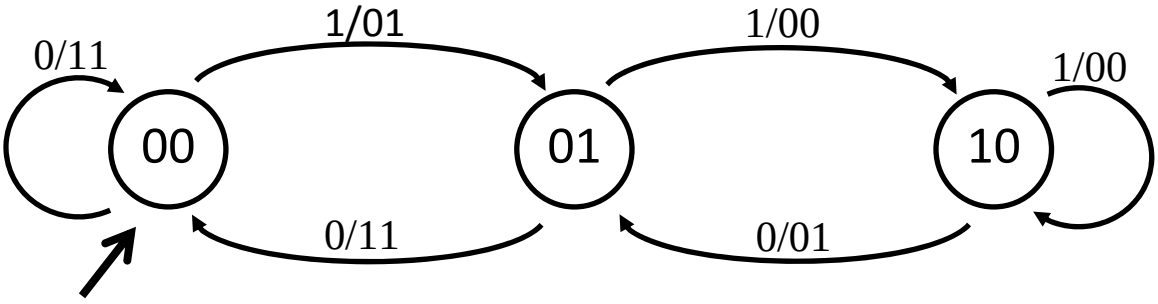
The following diagram shows a DNA construct known as a *promoter*. Proteins (p) interact with the promoter by binding and unbinding to its two operating sites (op1 and op2) and affecting how much mRNA (m) is being produced. A protein can only bind if there is an open operating site and a protein can only unbind if it was previously bound. Here we assume that p acts as a *repressor*, that is, the more p are bound to the promoter, the less m is produced.



The following FSM represents the behavior of this promoter under the assumption that a protein cannot bind to op2 unless there is already a protein bound to op1 and that a protein bound to op1 cannot unbind while a protein is bound to op2.

In = whether a protein attempts to bind (1) or a protein attempts to unbind (0)

Out (2 bits) = amount of mRNA produced when no proteins bound (3), 1 protein bound (1), or 2 proteins bound (0)



a) Provide more intuitive descriptions of the states:

State 00: _____

State 01: _____

State 10: _____

b) Fill in the blanks in the Truth Table below.

CS1	CS0	In	NS1	NS0	Out1	Out0
0	0	0	0			1
0	0	1	0			1
0	1	0	0			1
0	1	1	1			0
1	0	0	0	1	0	1
1	0	1	1	0	0	0
1	1	0	X	X	X	X
1	1	1	X	X	X	X

c) Provide the *most simplified* Boolean expression possible for `Out0`:

d) For the provided column `NS1`, draw out the combinational logic using the *fewest* number of gates possible, assuming the "don't cares" are taken to be 0 and 1 from top to bottom:

e) Provide one set of values for the "don't cares" in `NS1` that would *guarantee* that the FSM doesn't stay stuck in the 4th state even if it enters it by some glitch.

CS1	CS0	In	NS1
1	1	0	
1	1	1	

Question 3: *Mystery* (7 points, 14 minutes)

```
Mystery:
      la   $t0, L2
      lw   $t1, 8($t0)
      addi $t2, $0, 1
      sll  $t2, $t2, 16
      add  $t3, $0, $0
      add  $v0, $0, $0
      addi $t5, $0, 4
      sll  $t5, $t5, 16
L1:   beq  $t3, $t5, L3
      addu $t4, $t1, $t3
L2:   addu $t3, $t3, $t2
      sw   $t4, 8($t0)
      addu $v0, $v0, $a0
      j    L1
L3:   sw   $t1, 8($t0)
      jr   $ra
```

a) Which instruction gets modified during this function call?

b) How many times does the line at label L2 get executed?

c) Describe in a sentence or two what this function does.

Question 4: *MOESI On Through This Problem* (5 points, 12 minutes)

Fill out the empty fields corresponding to the state of the system.

Operation column: the fields can be ({P1,P2} {reads, writes} {Mem1,Mem2})

Processor {1,2} columns: MOESI state with corresponding block in cache, if applicable

Snoop bus signal(s): {Px.request(n), Px.send(n), Px.inv(n)} or a combination of them

Px.request(n) means processor Px is asking for data memory n

Px.inv(n) means processor Px invalidates all other copies of datablock n

Px.send(n) means Px is sending datablock n (in response to a request).

System characteristics: Write-back, write-allocate, invalidate others on write, Exclusive responds to read requests, 2 processors, 2-block memory, 1-block cache.

Operation	**Processor** 1	**Processor** 2	**Snoop Bus**
P1 reads Mem1	Exclusive(1)	Invalid	P1.request(1)
P2 reads Mem1	Owned(1)		P2.request(1) P1.send(1)
P2 write Mem1		Modified(1)	
	Modified(2)	Modified(1)	P1.request(2) P1.inv(2)
P2 read Mem2	Owned(2)	Shared(2)	
	Owned(2)	Exclusive(1)	P2.request(1)
	Invalid	Modified(2)	

Question 5: *This Problem Will BLOW Your Mine!* (10 points, 20 minutes)

One of your staff members loves the game of Minesweeper so much that he's going to build special hardware dedicated to playing this game. In Minesweeper, there is a rectangular grid of cells (the minefield), under which lie a finite number of mines. **Each cell contains either a mine or the number of mines in the adjacent EIGHT cells (touching on corners counts; 0 is shown as a blank).** The goal of the game is to uncover the whole field without clicking on/exploding a mine. See the images below:

Start of game: Blank field

Player clicked a mine and lost

Player uncovered field and won

Our goal is to design special hardware to facilitate high performance for setting up the game (generating the field).

a) We need to store the value of each cell in memory. How many bits are needed at minimum for each cell?

_____

b) Assume there is a helper function to randomly generate the locations of the mines. Name TWO SEPARATE REASONS why using an `int` array here is preferable to a linked list.

Reason 1: _____

Reason 2: _____

The procedure for generating the field is as follows. We start with an appropriately-sized array of all zeros. For each mine location, we add 1 to each adjacent cell and a relatively large number (say, 10) to the mine location itself. This way we know that any cell with value ≥ 10 contains a mine.

c) We are designing special hardware that acts like SIMD, but we will optimize the register size for Minesweeper. Given that an `int` is 32 bits, what size should our special "xmm" registers be?

_____

d) Assume our new special registers have been `typedef`-ed so that you can use the variable type **`xmm`** and that all the SIMD functions have been adjusted to work with the new register size. Only use the SIMD functions provided for you in the `#define` directives. Complete the following function to generate the minefield. **IGNORE EDGE CASES**.

```
#define load   _mm_loadu_ps
#define store  _mm_storeu_ps
#define add    _mm_add_ps

/* Field: m by n row-major array stored at mineField */
/* N mines stored in 2*N int array mineLocs (zero-indexed coordinates) */
void genField(int *mineLocs, int *mineField, int m, int n, int N) {
    int i, x, y;
    xmm reg1 = _mm_set_ps(1,1,1);
    xmm reg2 = _mm_set_ps(1,10,1);
    xmm reg3;
    for (i = 0; i < N; i++) {
        x = mineLocs[2*i];
        y = ____________________________________________________________;
        ____________________________________________________________
        ____________________________________________________________
        ____________________________________________________________
        ____________________________________________________________
        ____________________________________________________________
        ____________________________________________________________
        ____________________________________________________________
        ____________________________________________________________
        ____________________________________________________________
        ____________________________________________________________
    }
}
```

Question 6: *Watch Your (Time) Step!* (14 points, 26 minutes)

Consider the following difference equation (i.e. differential equation that is discretized in time):

```
x[n+1] - x[n] = -alpha*x[n]
x[n+1] = (1-alpha)*x[n]
```

Here we restrict ourselves to integer values of alpha and use the following integration function:

```
            # $a0 -> addr of array to store x[i] (assume x[0] is already set)
            # $a1 -> length of array/integration
            # $a2 -> alpha
      integrate:
1           beq   $a1,$0,exit
2           sub   $t0,$0,$a2   # $t0 = -alpha
3           addi  $t0,$t0,1    # $t0 = 1-alpha
4           lw    $t1,0($a0)   # $t1 = x[n]
5           mult  $t0,$t1
6           mflo  $t1          # $t1 = (1-alpha)*x[n]
7           sw    $t1,4($a0)   # x[n+1] = (1-alpha)*x[n]
8           addiu $a0,$a0,4
9           addiu $a1,$a1,-1
10          j     integrate
11    exit: jr    $ra
```

We are using a 5-stage MIPS pipelined datapath with separate I$ and D$ that can read and write to registers in a single cycle. Assume no other optimizations (no forwarding, etc.). The default behavior is to stall when necessary. Multiplication and branch checking are done during EX and the HI and LO registers are read during ID and written during WB.

a) As a reminder, T_c stands for "time between completions of instructions." Given the following datapath stage times, what is the ratio $T_{c,\text{single-cycle}}/T_{c,\text{pipelined}}$? _____

IF	ID	EX	MEM	WB
200 ps	100 ps	400 ps	200 ps	100 ps

b) Count the number of the different types of potential hazards found in the code above:

Structural: _____ Data: _____ Control: _____

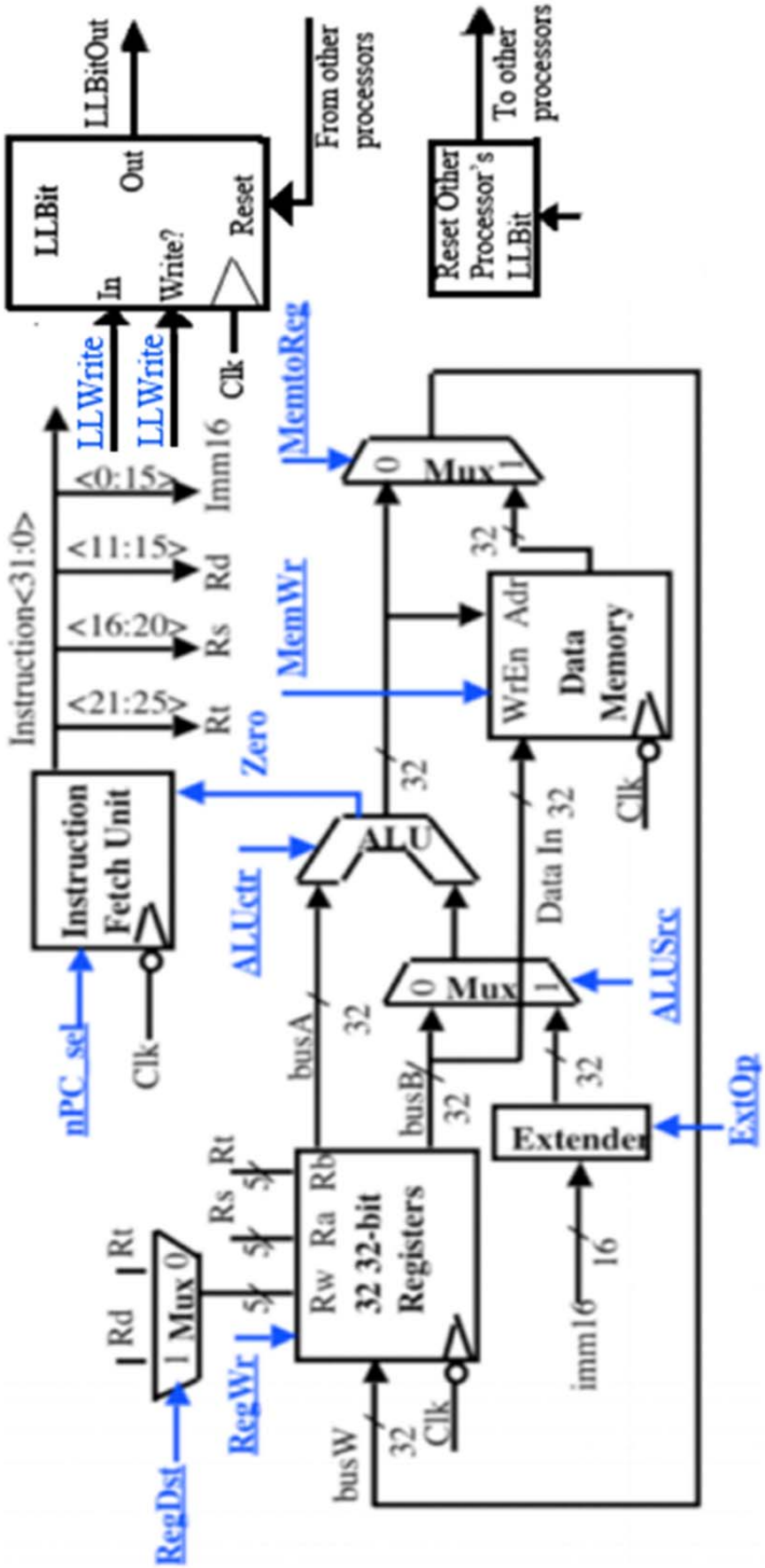
For the following questions, examine a SINGLE ITERATION of the loop (do not consider the `jr`).

c) With no optimizations, how many clock cycles does our 5-stage pipelined datapath take in one loop iteration (end your count exactly on completion of `j`)? _____

d) How many clock cycles LESS would be taken if we had FORWARDING? _____

e) Assuming that we introduce both FORWARDING and DELAY SLOTS, describe independent changes to `integrate` that will reduce the total clock cycles taken. Changes include replacing or moving instructions. You may not need all spaces given.

Instr	**Change**
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____


LLBit
Out
In
Write?
Reset
LLBitOut
From other processors
Reset Other Processor's LLBit
To other processors
LLWrite
LLWrite
Clk
MemtoReg
Imm16
<0:15>
<11:15>
Rd
<16:20>
Rs
<21:25>
Rt
Instruction<31:0>
MemWr
Zero
Instruction Fetch Unit
Clk
nPC_sel
ALUctr
ALU
32
WrEn Adr
Data Memory
Data In 32
Clk
busA
busB
32
Mux
ALUSrc
Extender
ExtOp
imm16
16
Rt
Rd
Mux
RegDst
RegWr
Rw Ra Rb
32 32-bit Registers
busW
32
Clk

Question 7: *Conditional Wiring* (10 points, 20 minutes)

Assume our MIPS machine is implemented as a single-cycle datapath. Your job is to provide new datapath and control elements that can implement **store-conditional** instruction (the implementation of load-link is provided for you). In particular, the load-link and store-conditional should be implemented as:

```
LL: R[rt] <- M[sign(imm) + R[rs]]
    LLbit <- 1

SC: if LLbit:
        M[sign(imm) + R[rs]] <- R[rt];
        Reset other processor's LLbit
    R[rt] <- {31 zeros, LLbit}
```

Modify the picture on the opposite page and **list** the changes you made here (you may not need all the given lines). You cannot modify the pre-existing datapath components besides the wires:

1. _____
2. _____
3. _____
4. _____
5. _____
6. _____
7. _____
8. _____

Now **state** what values of these control signals should be {0,1, X for "don't care," or any other intuitive names}, plus for any new signals you may have created.

	RegDst	**RegWr**	**nPC_sel**	**ExtOP**	**ALUSrc**	**ALUctr**
LL	rt	1	PC+4	sign	imm	add
SC						

	MemWr	**MemtoReg**	**LLWrite**			
LL	0	1	1			
SC						

Question 8: *It's Not My Fault* (10 points, 20 minutes)

Consider the following OpenMP snippet:

```
int values[size];
#pragma omp parallel
{
      int i = omp_get_thread_num();
      int n = omp_get_num_threads();
      for(int j = i * (size / n); j < (i + 1) * (size / n); j++)
      {
            values[j] = j;
      }
}
```

All cores share the same physical memory and we are running 2 threads. This is the sole process running. Each page is 1 KiB, and you have 2 pages of physical memory. The code snippet above starts at virtual address 0x400, and the values array starts at 0x800. The size, n, i, and j variables are all stored in registers. The functions omp_get_thread_num and omp_get_num_threads are stored in virtual addresses 0x440 to 0x480. The replacement policy for the page table is Least Recently Used.

At the start of the pragma omp parallel call the page table looks as follows:

Virtual Page Number	Valid	Dirty	Physical Page Number
0	0	0	0
1	1	0	0
2	1	1	1
3	0	0	1

a) How many page faults will occur if size = 0x080? _____

b) What is the minimum number of page faults that will occur if size = 0x200? _____

What is the maximum? _____

c) How could you reduce the maximum page faults for part (b)? Choose the valid options amongst the following:

___ increase virtual address space

___ increase physical address space

___ use SIMD instructions

___ change page table replacement policy to Random

___ decrease number of threads

___ increase number of threads

___ add a 4-entry TLB

PAGE PURPOSELY LEFT BLANK

(Any work on this page will not be graded)

BACK OF EXAM

(Any work on this page will not be graded)