University of California, Berkeley – College of Engineering

Department of Electrical Engineering and Computer Sciences

Fall 2005 Instructor: Dan Garcia 2005-12-17

☺ CS61C FINAL EXAM ☺

Last Name	
First Name	
Student ID Number	
Login	`cs61c-`
Login First Letter (please circle)	`a b c d e f g h i j k l m`

Login Second Letter (please circle)	`a b c d e f g h i j k l m`
	`n o p q r s t u v w x y z`

*The name of your **LAB** TA (please circle)*	Jeremy Michael Navtej Zhangxi
Name of the person to your Left	
Name of the person to your Right	
All the work is my own. I have no prior knowledge of the exam contents nor will I share the contents with others in CS61C who have not taken it yet. ***(please sign)***	

Instructions (Read Me!)

- This booklet contains 9 numbered pages including the cover page. Put all answers on these pages (feel free to use the back of any page for scratch work); don't hand in any stray pieces of paper.
- Please **turn off** all pagers, cell phones & beepers. Remove all hats & headphones. Place your backpacks, laptops and jackets at the front. Sit in *every other* seat. Nothing may be placed in the "no fly zone" spare seat/desk between students.
- Fill in the front of this page and put your name & login on every sheet of paper.
- You have 180 minutes to complete this exam. The exam is closed book, no computers, PDAs or calculators. You may use two pages (US Letter, front and back) of notes, plus the green reference sheet from COD 3/e.
- There may be partial credit for incomplete answers; write as much of the solution as you can**.** We will deduct points if your solution is far more complicated than necessary. When we provide a blank, please fit your answer within the space provided. "IEC format" refers to the mebi, tebi, etc prefixes. You have 3 hours...relax.
- **You must complete ALL THE QUESTIONS**, **regardless of your score on the midterm**. Clobbering only works from the Final to the Midterm, not vice versa.

Problem	**M1**	**M2**	**M3**	**Ms**	**F1**	**F2**	**F3**	**F4**	**F5**	**Fs**	**Total**
Minutes	20	20	20	60	24	24	24	24	24	120	180
Points	**10**	**10**	**10**	**30**	**18**	**18**	**18**	**18**	**18**	**90**	**120**
Score											

Name: ___________________________ Login: `cs61c-`____

Midterm Revisited

M1) "Doctor, our patient is *encoding*!" (10 pts, 20 min)

a) A Binary Coded Decimal (BCD) uses a dedicated nibble for each decimal digit, so a byte could represent all the numbers from 00-99. We will use our standard MIPS 32-bit word to encode a BCD. What is the ratio (to one significant figure, in decimal) of overall bit patterns to the ones that encode a valid BCD? (E.g., With a *single* decimal digit, it'd be 16/10 ≈ 2.) Show your work. Your answer should not be an expression, it should be a decimal number rounded to 1 significant figure.

____

b) Suppose we have a very small 4 pixel × 8 pixel grayscale video display where each pixel can independently be set to one of 4 shades of gray. How many unique images can possibly be displayed? Leave your answer in IEC form (e.g., 64 kibi images, 8 mebi images, etc).

____

c) If we were to try to compare two `float`s using our MIPS signed integer compare `slt`, when would we get an *incorrect* answer (i.e., describe in English the set of *all possible inputs* that generate incorrect answers)? Assume neither encodes a `NaN` or `±0`.

____

d) Put the corresponding letters for each 32-bit value in order from least to greatest. Hint: the question isn't asking you to write down what each one is, it only asks for the relative order!

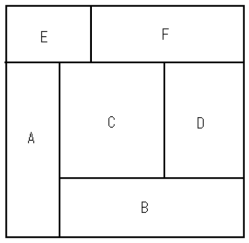
A. `0xF0000000` (IEEE float)
B. `0xF0000000` (2's complement)
C. `0xF0000000` (sign-magnitude)
D. `0xFFFFFFFF` (2's complement)
E. `0xFFFFFFFF` (1's complement)
F. `0xF1000000` (IEEE float)
G. `0x70000000` (IEEE float)
H. `0x7FFFFFFF` (2's complement)
I. `0x80000010` (IEEE float)

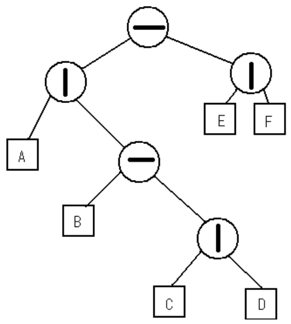
Least____Greatest

Name: ____________________________ Login: cs61c-____

M2) "A man, a *plan*, a canal…Panama" (10 pts, 20 min)

A slicing *plan* is a decomposition of a rectangle with horizontal and vertical sides using horizontal and vertical *cuts*. This type of decomposition can be represented as a binary tree, whose internal nodes are the *cuts* (horizontal or vertical), and whose external leaf nodes are the *rectangles*. The diagram to the right shows an example of how we would divide the space with five cuts into six rectangles, A-F.





```
#define CUT       0
#define RECTANGLE 1

typedef struct node {
  int type;        /* CUT or RECTANGLE */
  char label[16];  /* If a RECTANGLE, the name. If a CUT, "HORIZONTAL" or "VERTICAL". */
  struct node *L;  /* If type is a RECTANGLE, these may be any value! */
  struct node *R;  /* If type is a RECTANGLE, these may be any value! */
} slicenode_t;

1 slicenode_t masterPlan;
2 int main(int argc, char *argv[]) {
3   slicenode_t plans[2], *ptrs[2];
4   ptrs[1] = (slicenode_t *) malloc (sizeof(slicenode_t) * 10);
5   …
```

a) How many bytes would be used in the *static*, *stack* and *heap* areas as the result of each of these lines. *Treat each line independently!* E.g., For line 3, don't count the space allocated in line 1.

	static	***stack***	***heap***
Line 1			
Line 3			
Line 4			

b) Finally, assuming the entire plan is stored on the heap, finish the `Delete` function to *delete* the **full plan.** It should return *the number of slicenodes it freed* (e.g., Given the plan above, it'd return 11). When deleting, assume that the OS immediately fills any freed space with garbage, so you cannot access freed heap contents. You may assume `Delete` won't be called from the outside with NULL. You will lose points if your code is overly complicated. Our longest solution has only 7 semicolons.

```
int Delete (slicenode_t *plan) {

  if(                                        ) {



  } else {







  } /* Remember, Delete should RETURN THE NUMBER OF SLICENODES IT FREED */
}
```

Name: ___________________________ Login: `cs61c-`____

M3) "Fenry Hord invented the *disassembly* line…" (10 pts, 20 min)

a) Given the MIPS code below, write the equivalent C function below in the structure we've provided. Feel free to add comments to help your disassembly. To aid readability, you must use the variable names from our comments below in your C solution where appropriate.

```
foo:
addiu $sp, $sp, -12
sw    $a0, 0($sp)       # src
sw    $a1, 4($sp)       # size
sw    $ra, 8($sp)
move  $a0, $a1          #
addiu $a0, $a0, 1       #
jal   malloc            #
move  $t0, $v0          # dest
lw    $t1, 0($sp)       # src
lw    $t2, 4($sp)
addu  $t2, $t2, $t1     # end

foo_loop:
beq   $t2, $t1, foo_end #
lbu   $t4, 0($t1)       #
ori   $t4, $t4, 0x20    #
sb    $t4, 0($t0)       #
addiu $t0, $t0, 1       #
addiu $t1, $t1, 1       #
j     foo_loop

foo_end:
sb    $0, 0($t1)        #
lw    $ra, 8($sp)       #
addiu $sp, $sp, 12
jr    $ra
```

```
________  foo (________ src, ________ size) {



  for(___________; ____________; ____________) {



  }




}
```

b) If `src` contained *letters*, what is a more appropriate name for the subroutine `foo`? (i.e., what would "`jal foo`" do, from the point of view of the caller?) Hint: you might find the green sheet handy here.

c) What if we called `foo` from `printf` as so: `printf("…format string…", foo(source, size))`. *Why is this bad form?* Hint: think about what would happen if this were done many times.

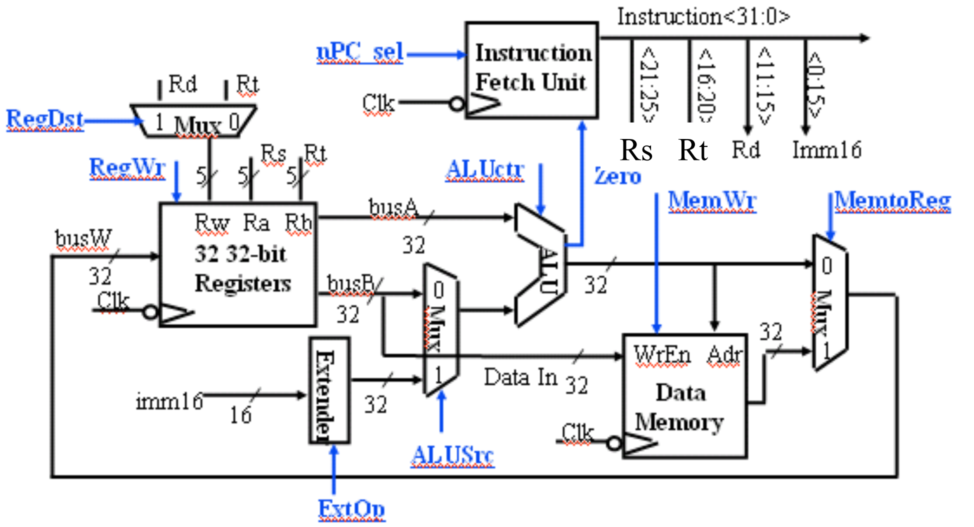
d) Let's say we removed the "**`sb $0, 0($t1)`**" instruction and then made the same call to `foo` from `printf` as in question (c) above: What are *all* the things that could happen?

Name: ______________________________ Login: cs61c-____

Post-Midterm Questions

F1) "Where's the `sofr` (sophomore, freshman) instr?" (18 pts, 24 min)

On the right is the single-cycle MIPS datapath presented during lecture. Your job is to modify the diagram to accommodate a new MIPS instruction. Your modification may use simple adders, shifters, mux chips, wires, and new control signals. If necessary, you may replace original labels.



Function calls in MIPS typically end with stack restoration and `jr $ra` as shown below. Because this happens often, we want to do this in *one* instruction instead of *two*. We'll design a new I-type instruction, `srjr` (stack restore, jump register), as follows:

```
addi  $sp, $sp, 16   ➔   srjr $ra $sp 16
jr    $ra
```

a) What is the RTL for `srjr` that will allow it to have the **widest range of stack restoration**? Hint: stack *restoration* always occurs in *non-negative word units*… The `jr` is done already.

____ **; PC = R[rs]**

b) What is the most `$sp` could change as a result of a `srjr` call (in Bytes)? Use IEC format.

__________________ **Bytes (plus or minus a byte).**

c) **Modify the picture above** and list your changes below. You may not need all the boxes. Please write them in "pipeline stage order" (i.e., changes affecting IF first, MEM next, etc)

(i)	
(ii)	
(iii)	
(iv)	
(v)	
(vi)	

d) We now want to set all the control lines appropriately. List what each signal should be (an intuitive name or {0, 1, x = don't care}). Include any *new* control signals you added.

RegDst	RegWr	nPC_sel	ExtOp	ALUSrc	ALUctr	MemWr	MemtoReg			

Name: ____________________________ Login: `cs61c-`____

F2) "*Cache*, money. Dollar bills, y'all." (18 pts, 24 min)

Given the following code for a MIPS machine with 8 byte blocks, an empty 128-entry *fully-associative* LRU L1 cache, 4 MiB `ARRAY_SIZE`, and `char A[]` starting at a block boundary (byte 0 of a block):

```
for (i = 0 ; i < (ARRAY_SIZE/STRETCH) ; i++) {                   /* # of STRETCHes    */
   for (j = 0 ; j < STRETCH ; j++)  sum     += A[i*STRETCH + j]; /* for each STRETCH */
   for (j = 0 ; j < STRETCH ; j++)  product *= A[i*STRETCH + j]; /* for each STRETCH */
}
```

a) What is the `T:I:O` bit breakup (assuming byte addressing)? ______:______:______

b) *Cache size* (data only, not tag and extra bits) in bytes? (Use IEC) __________________

c) What is the largest `STRETCH` that *minimizes cache misses*? (Use IEC) _________________

d) Given the `STRETCH` size from (c), what is the # *of cache misses*? (Use IEC) ________________

e) Given the `STRETCH` size from (c), if `A` ***does not*** *start* at a block boundary, *roughly* what is the ratio of the # of *cache misses* for this case to the number you calculated in question (d) above. (e.g., 8x, 1/16th) __________________

f) Can a 32-bit MIPS machine make use of more than 4 GiB of physical memory? Why or why not?

____

g) Would the performance of such a machine be any better than one with 4 GiB of physical memory? Why or why not?

____

h) In our familiar single-cycle CPU with interrupts turned off, 4 KiB pages and no other users logged in or other programs running (other than the OS, which is not doing anything interesting at present), we are executing lines of MIPS code and we come to two instructions:

```
and $t1, $t2, $t3
or  $t1, $t2, $t3
```

and we notice that the **`and`** completes in 1 cycle, but the **`or`** completes in 1,000,000 cycles. You realize you can now infer something about the *value of the PC* when the **`and`** was running...what is it? Be as explicit as possible.

____

Name: ___________________________ Login: `cs61c-____`

F3) "Folks in Alaska are experts in *Pipelining*…" (18 pts, 24 min)

Consider a processor with the following specification:

- Standard five (5) stage (F, D, E, M, W) pipeline.
- No forwarding.
- Stalls on ALL hazards.
- Non-delayed branches
- Branch comparison occurs during the second stage.
- Instructions are not fetched until branch comparison is done.
- Memory CAN be read/written on same clock cycle.
- The same register CAN be read & written on the same clock cycle (structural hazard).
- No out-of-order execution.

a) Count how many cycles will be needed to execute the code below and write out each instruction's progress through the pipeline by filling in the table below with pipeline stages (F, D, E, M, W).

```
[1]  add $a0, $a0, $t1
[2]  lw  $a1, 0($a0)
[3]  add $a1, $a1, $t1
[4]  sw  $a1, 0($t1)
[5]  add $t1, $t1, -1
[6]  bne $0, $0, end
[7]  add $t9, $t9, 1
```

Cycle →	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Inst 1	**F**																								
Inst 2																									
Inst 3																									
Inst 4																									
Inst 5																									
Inst 6																									
Inst 7																									

b) Considering the following two *changes*, fill in the table again:

- Our processor now forwards values
- Delayed branches

Cycle→	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Inst 1	**F**																								
Inst 2																									
Inst 3																									
Inst 4																									
Inst 5																									
Inst 6																									
Inst 7																									

Name: ____________________________ Login: cs61c-____

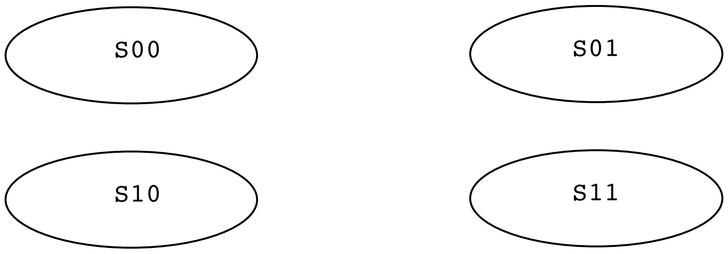
F4) Synchronous Digital *Circus* (18 pts, 24 min)

We are designing a circuit with a 1-bit input (`I(t)`) and a 2-bit output (`O(t)`), that will produce, at time `t`, the number of *zeros* in the set `{I(t-2), I(t-1), I(t)}`. As an example,

```
the input:                    I: 1 1 0 0 1 0 0 1 1 0 1 1 1 0 0 0
…will produce the output:     O: 0 0 1 2 2 2 2 2 1 1 1 1 0 1 2 3
```

a) Complete the FSM diagram below. Our states have been labeled `Sxy` indicating that the previous 2 bits,`{I(t-2), I(t-1)}` would be `{x, y}`. Fill in the truth table on the right. The previous state is encoded in (`P1`,`P0`), the next state is encoded in (`N1`,`N0`), and the output is encoded as (`O1`,`O0`). Make sure to indicate the value of the *output* on your state transitions.

P1	P0	I	O1	O0	N1	N0
0	0	0				
0	0	1				
0	1	0				
0	1	1				
1	0	0				
1	0	1				
1	1	0				
1	1	1				



b) Provide *fully reduced* (i.e., fewest gates to implement…you can use any *n*-input gates) Boolean expressions for the Output (`O1`,`O0`) and Next State (`N1`,`N0`) bits. If there is a name for any of the circuits, write it on the left. E.g., "The always-1", "3-input NAND", etc. A 2-input XOR has the symbol of "⊕".

Scratch space

__________________ O1 =

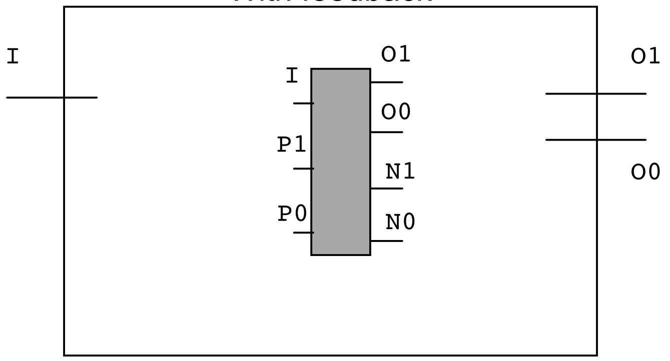
__________________ O0 =

__________________ N1 =

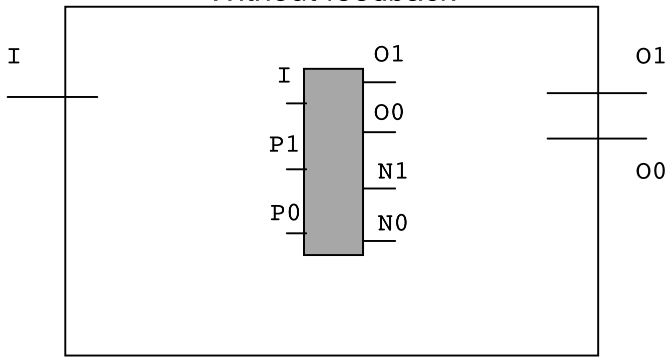
__________________ N0 =

c) Draw the overall circuit using the *fewest* gates possible with and without feedback below. You *may* add registers. "Feedback" means outputs are somehow fed back into inputs. Assume we've correctly implemented the answer to (b) as a black box in the middle.

With feedback



Without feedback



Name: ___________________________ Login: `cs61c-`____

F5) The "*Martha Stewart Potpourri*" Question… (18 pts, 24 min)

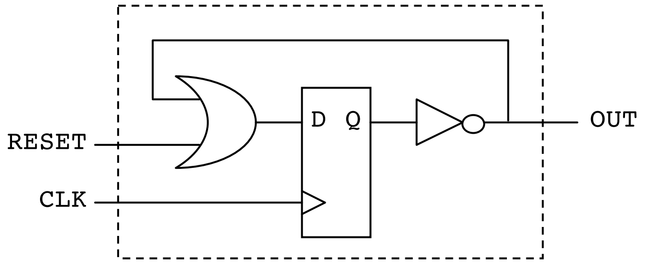
a) In lecture, we saw examples of "learning from failure" where we were shown history's top ten worst software bugs as chosen by Wired Magazine (e.g., the Therac-25 incident). Aside from Therac, what is another one of them and what is a lesson we should draw after seeing all ten?

____

b) Prof. Patterson said that in the 20th century, computer engineers optimized for performance and cost. Aside from *availability*, what does he suggest we optimize for in the 21st century?

____

"Much Ado About Not-ing":
Consider the circuit on the right with the following specifications:
$t_{inverter}$, t_{or}, $t_{clk\text{-}to\text{-}q}$, t_{setup}, t_{hold}, t_{clock}
(assume no delay on the wires):



c) If all other times are fixed, what is the valid range for $t_{inverter}$? Express it in terms of the variables listed above.

__________________ $\leq t_{inverter} \leq$ ____________________

Given the following instruction mix:

ALU	Load/Store	Branch
20%	30%	50%

and CPI_i for each instruction i:

Machine	Clock speed	ALU	Load/Store	Branch
A	2 GHz	2	2	2
B	4 GHz	1	1	$\boxed{}$

d) Given that machine A has a clock speed of 2 GHz and B has a clock speed of 4 GHz, how fast do branches for machine B need to be to achieve the same execution performance for this particular program? Write it in the box above.

e) (This question has nothing to do with MIPS) Assume we have enough bits to byte-address 16_{10} exbibytes. We want to define some number of the most-significant bits to encode $12_{10} \times 2^{10}$ things, and some number of the least-significant bits to encode $200{,}000{,}000_{10}$ things. How many things can we encode with the remaining bits? Use IEC format, like "16 kibithings", or "128 mebithings". Show your work on the right.

____

You need to design a 1 TebiByte disk, but you only have 4 platters, an outer radius of `sqrt`($30/\pi$) in and an inner radius of `sqrt`($22/\pi$) in.

f) What's the *bit density* you need (in B/in^2 units)? Use IEC format. ____________________

If you used 32 identical 1-TebiByte disks to build a single *virtual* RAID volume, how much space (in IEC format) would the user see if we used RAID level…

g) 0 ________________, 1 ________________, 3 ________________, 5 ________________