EE 460N Fall 2013
Aater Suleman, Instructor
Stephen Pruett, Abhishek Agarwal, Chirag Sakhuja, TAs
Final Exam
December 13, 2013

Name:_____

Problem 1 (25 points):_____

Problem 2 (10 points):_____

Problem 3 (10 points):_____

Problem 4 (10 points):_____

Problem 5 (20 points):_____

Problem 6 (20 points):_____

Problem 7 (30 points):_____

Total (125 points):_____

You have 3 hours to take this exam.
Note: Please be sure that your answers to all questions (and all supporting work that is required) are contained in the space provided.

Note: Please be sure your name is recorded on each sheet of the exam.

Please sign the following. I have not given nor received any unauthorized help on this exam.

Signature:_____

GOOD LUCK!

Name:_____

Problem 1 (25 points):

Part a (5 points): A [DMA] controller copies data from disk to memory without using the processor.

Part b (5 points): What is the greatest positive sub-normal number in an IEEE-like binary floating point format where sign, exponent, and mantissa are 1, 3, and 4 bits respectively. The value of bias is 3.

Please answer in binary floating-point format.

[0 000 1111]

Part c (5 points): How many core-to-core connections are required in an N-core system where coherence interconnect is a ring? (assume only one uni-directional ring)

[N−1]

Part d (5 points): Sun (now Oracle) built the Niagara chip where 8-processors shared the same Floating Point (FP) execution unit. To perform an FP operation, cores sent a request to this single FP unit over the interconnect and got a response back several cycles later. Knowing that this decision was made keeping bread-and-butter design in mind, what can you say about the software Sun expected to run on this chip. Answer in less than 10 words.

floating point will not be used as much

Part e (5 points): Assume a byte-addressable 4GB physical memory with N channels, 8 ranks/channel, and 8 chips/rank. The bus width is 64B and each chip is 32MB. What is N?

$$8 * 8 * 32MB * N = 4GB$$

[2]

Problem 2 (10 points):

Design the stall logic for an in-order 2-wide machine that uses a scoreboard. Assume that the two instructions that enter the Decode stage concurrently are called Instruction0 and Instruction1, where Instruction0 comes before Instruction1 in program order.

The stall logic has the following inputs and outputs:

Inputs:
1. The SR and DR (the source and destination registers) of the corresponding instruction. For example, Instruction0.DR is the destination register for Instruction0.
2. The Scoreboard bits. For example, Scoreboard[Instruction0.SR1] is the scoreboard bit for Instruction0's source register 1.

Outputs: Instruction0.stall that stalls Instruction0 only and Instruction1.stall that stalls Instruction1 only.

You may assume that every instruction has valid DR, SR1, and SR2 fields.
You may also assume that there are no stalls related to memory dependencies.

We have drawn the stall logic partially below. Your job is to complete it using **only** wires and one or more of the gates provided below (Equivalence, OR, AND). You may assume that the OR gates shown in the diagram have unlimited fan-in.

3 3	3 3	3 3
=	OR	AND
1	3	3
Equivalence	OR gate	AND gate

Instruction0.DR   Instruction1.SR1   Instruction1.SR2

3   3   3

=   =

Scoreboard[Instruction0.SR1]
Scoreboard[Instruction0.SR2]

Scoreboard[Instruction1.SR1]
Scoreboard[Instruction1.SR2]

Instruction0.stall   Instruction1.stall
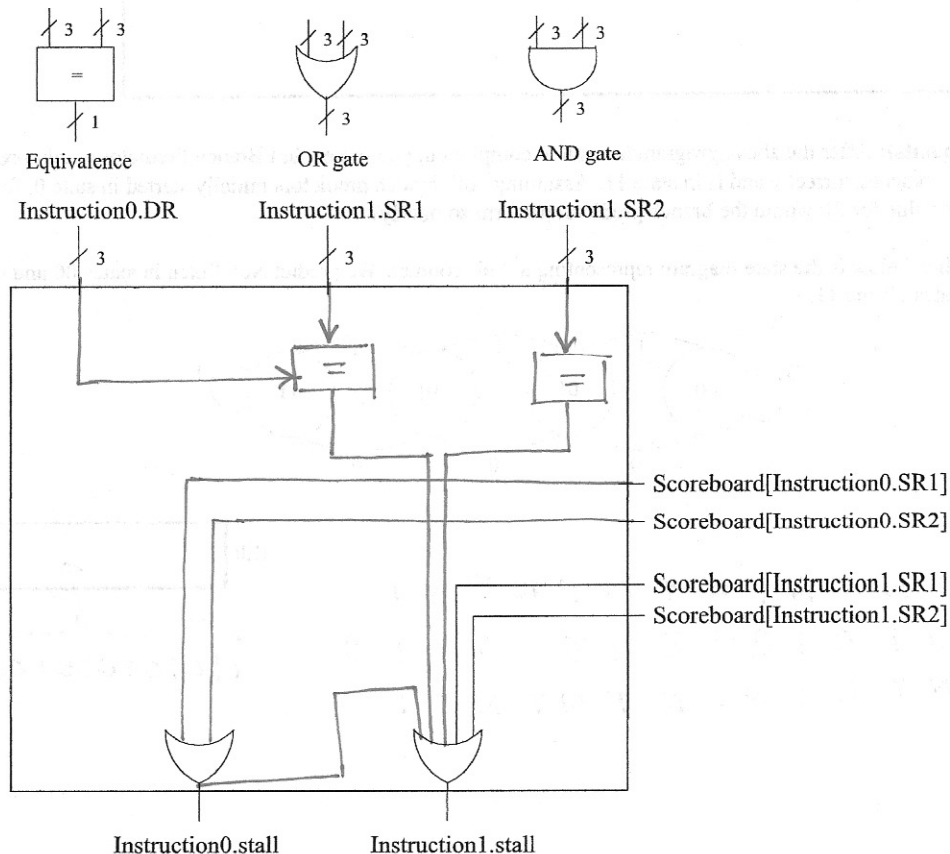
Name:_____

Problem 3 (10 points):

Suppose we have added <u>two</u> 2-bit counter branch predictors to the LC-3b. The program below runs on the modified LC-3b.

Note: Which counter to use for a branch is based on PC[1]. The branch predictor used by each branch is already specified below.

```
        .ORIG x3000
        AND R1, R1, #0
        ADD R0, R0, #0
A       BRn B              ; -- uses Branch Predictor 0
        ADD R1, R1, #1
B       ADD R0, R0, R0
        BRnp A             ; -- uses Branch Predictor 1
        HALT
```

Part a (3 points): Explain what the purpose of the program is. Answer in less than 20 words.

count # of 0s in more significant positions than
the least significant 1.

Part b (7 points): After the above program has run to completion, you notice that Branch Predictor 0 only predicted 1 out of 15 branches correctly and is in state 11. Assuming both branch predictors initially started in state 0, for what 16-bit input value for R0 would the branch predictor perform so poorly?

As a refresher, below is the state diagram representing a 2 bit counter. We predict Not-Taken in states 00 and 01 and Taken in states 10 and 11.
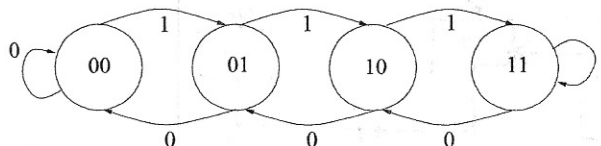
Predicted
↳N N T N T N T N T N T N T N T
 1 1 0 1 0 1 0 1 0 1 0 1 0 1 1 0
↗T T N T N T N T N T N T N T T
Actual

R0: [   ]
    ↑
 1 1 0 1 0 1 0 1 0 1 0 1 0 1 1 0

Name:_____

Problem 4 (10 points): Assume a system with byte-addressable memory, a 16-bit Virtual Address space, 14-bit Physical Address Space, page size of 4KB, and one level virtual to physical translation. What is the maximum size direct mapped cache that can be used in this system without introducing any cache consistency issues?

Note 1: Assume a Virtually Indexed Physically Tagged cache.
Note 2: The above provides the entire information required to solve this problem. Assume no restrictions on the Virtual Page Numbers.
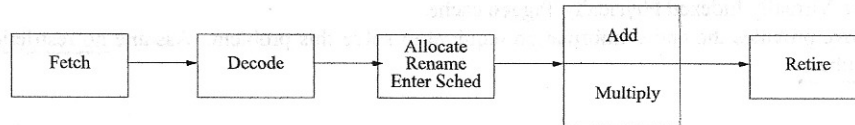
VIPT means index + BIB = page size

4 KB

Name:_____

Problem 5 (20 points):

Recall the core460N microarchitecture discussed in class. In this problem we use core460N with *one modification*: the width of the machine has been reduced from 3-wide to 1-wide. Below is a diagram of the modified core460N.

```
┌────────┐    ┌────────┐    ┌──────────────┐    ┌──────────┐    ┌────────┐
│ Fetch  │───▶│ Decode │───▶│  Allocate    │───▶│   Add    │───▶│ Retire │
│        │    │        │    │  Rename      │    ├──────────┤    │        │
└────────┘    └────────┘    │  Enter Sched │    │ Multiply │    └────────┘
                            └──────────────┘    └──────────┘
```

Notes:
1. There is one 1-cycle non-pipelined adder
2. There is one 4-cycle non-pipelined multiplier
3. Instructions fill in the Reservation stations in program order **from top to bottom**
4. There are only two reservation station entries per reservation station
5. Fetch, decode, allocate, and retire take one cycle each
6. An instruction's reservation station entry is freed *after it completes execution*.
7. The ROB, RAT, and RS are updated at the *end* of the execution stage
8. The Source 1 and Source 2 fields in the RS correspond to SR1 and SR2 in the instruction respectively
9. The ROB and RS are initially invalid.

Contents of the two Reservation Stations and Re-order buffer (ROB) at the **end of the 6th cycle** are shown below. The identities of the functional units have been replaced with a "?". You will need to identify them to solve this problem.

V	Source 1 TAG	VALUE	V	Source 2 TAG	VALUE
0	W	–	1	–	2
0	X	–	1	–	6

$+$?
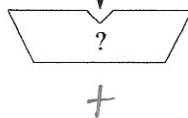
V	Source 1 TAG	VALUE	V	Source 2 TAG	VALUE
1	–	4	1	–	5
0	Y	–	1	–	6

$*$?
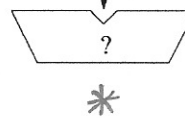
Tag	Ready	DR	Value
W	0	———	———
X	0	———	———
Y	0	———	———
Z	0	———	———

PROBLEM IS CONTINUED ON THE NEXT PAGE!

Name:_____

Problem 5 continued

Also shown below is the partial contents of the Register Alias Table (RAT) **after the 6th cycle**. After reset, four instructions (I1-I4) are executed.

Your job:

- Identify I1-I4.

- Identify the missing valid bits in the RAT by filling in the figures below.

	Opcode(+ OR *)	DR	SR1	SR2
I1	*	R0	R1	R4
I2	+	R0	R0	R5
I3	+	R2	R0	R2
I4	*	R3	R2	R3

Figure 1: Instruction Table

	VALID	TAG	VALUE
R0	0	X	1
R1	1	W	4
R2	0	Y	6
R3	0	Z	6
R4	1	X	5
R5	1	Y	2

Figure 2: Register Alias Table after the 6th cycle

Problem 6 (20 points):

Assume an x86-like virtual memory system where virtual addresses are 6 bits. The system page table is indexed using VA[5:4] and the user page table is indexed using VA[3:2]. When virtual address X is translated, there are no TLB hits or page faults, and the corresponding physical address is 0xA.

Each PTE is 8 bits and has the following fields:

7	6	5	4		2	1		0
V	D	R		Reserved			PFN	

Here are the complete contents of physical memory:

0x0	00000000
0x1	10000011
0x2	10000001
0x3	00000000
0x4	00000000
0x5	00000000
0x6	00000000
0x7	10000000
0x8	00000011
0x9	10000010
0xA	11111111
0xB	10101010
0xC	00000000
0xD	10000010
0xE	00000010
0xF	00000000

Part a (2 points): Calculate the number of frames in physical memory.

4

PROBLEM IS CONTINUED ON THE NEXT PAGE!

Problem 6 continued

PFN | BIB
xA: 10 | 10

Part b (8 points): Which one of these addresses could/couldn't contain the PTE of the page containing Virtual Address X? Why?

Addr	Value	Circle one	Why?
0x7	10000000	Y/**N**	PFN does not match
0x9	10000010	Y/**N**	Cannot be page table frame b/c xB isn't PTE
0xB	10101010	Y/**N**	Reserved bits are nonzero
0xD	10000010	**Y**/N	PFN is correct and valid is set
0xE	00000010	Y/**N**	Valid bit is not set

Part c (3 points): What is the SBR?

x 0

Part d (7 points): What is the Virtual Address X?

~~01 1010~~

0 1 0 1 10

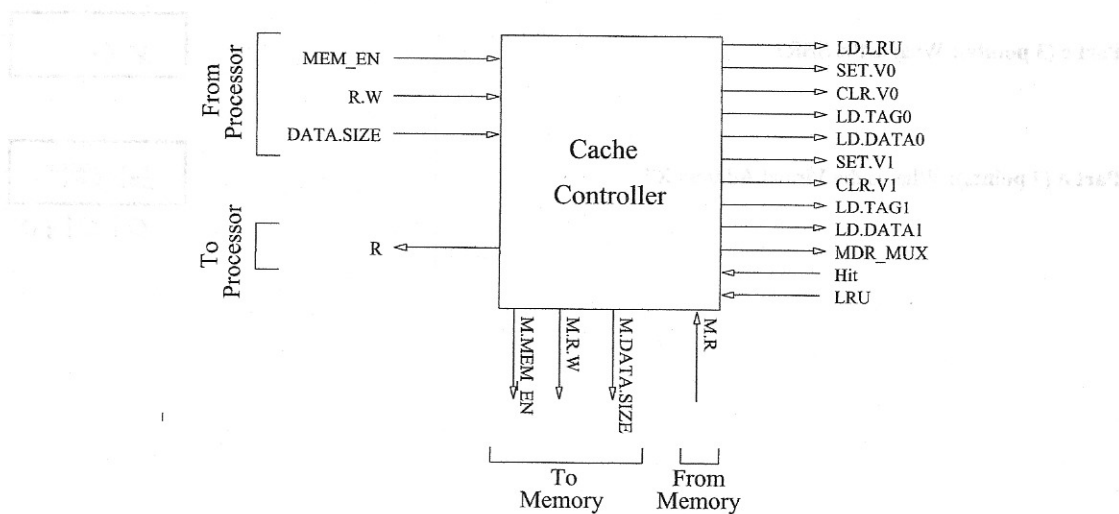Name:_____

Problem 7 (30 points):

Little Computers Inc. has decided to add a fully-associative 2-entry writethrough cache with a line size of 2B to the non-pipelined LC3b. Your job is to help us integrate this cache with the LC-3b. *For simplicity, you will implement ONLY the LDW instruction.*

The figure below shows the cache controller and its Inputs/Outputs. The controller interfaces with the processor, cache, and memory as follows:

Processor: To read data, the processor loads the address in MAR and sets the MEM_EN signal, R.W, and DATA.SIZE signals. When the access is complete, the controller loads the data in MDR and asserts the R signal.

Cache: To perform an access, the controller first accesses the Tag Store of the cache to check for a cache hit/miss. If there is a cache hit, it loads MDR with the data in the corresponding data store entry. If there is a cache miss, it accesses the memory.

Memory: To access memory, the controller asserts M.MEM_EN signal, M.R.W, and M.DATA.SIZE signals. When the data from memory is ready, it loads the data in MDR and the cache, updating both the tag and data store of the cache.
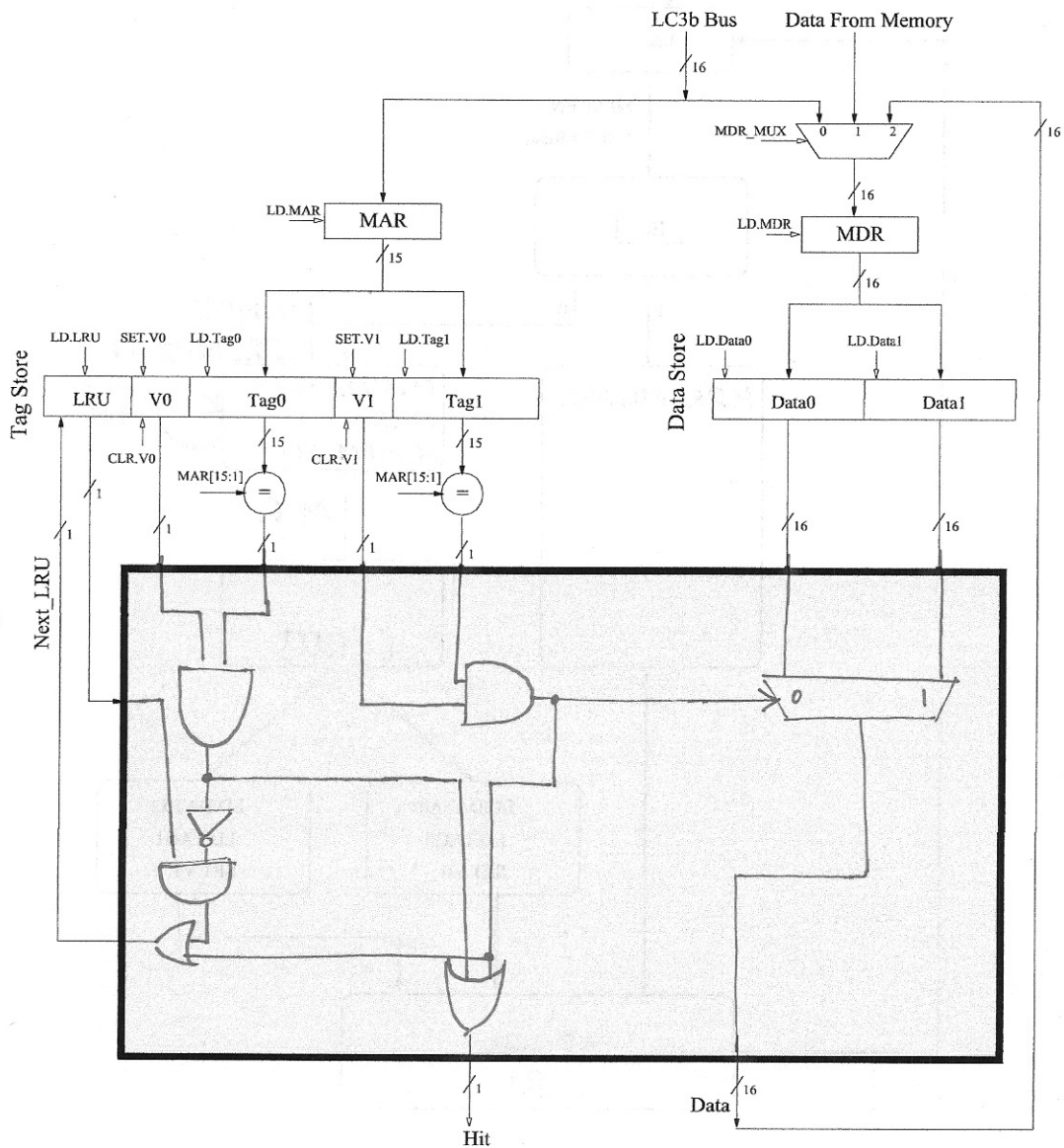
Name:_____

Problem 7 continued

Part a (15 points): Below are additions to the LC-3b data path to support loads from the cache. Draw the combinational logic that should reside in the shaded empty box. Control signals are generated by the cache controller shown above.

   Hint: Part a and Part b of this problem are interdependent. You may find it helpful to read and analyze Part b before you start working on Part a.

Problem 7 continued

Part b (15 points): Below is a partially complete state diagram of the cache controller. Complete this diagram by filling in the **missing states and any missing transitions.**

Note: Please follow the convention that signal names listed inside the state bubble are assumed high and all remainig signals are assumed low.

$\overline{\text{MEM\_EN}}$

Idle

MEM_EN
& R.W=Read

[Hit]

1   0

MDR_MUX ← 2

LD.MDR
MDR_MUX ← 1

M.DATA.SIZE
M.R.W
M.MEM_EN

$\overline{M.R}$

M.R

[LRU]

LD.DATA0
LD.TAG0
SET.V0

LD.DATA1
LD.TAG1
SET.V1

LD.LRU
R