Name _____

Computer Architecture
LSU EE 4720
Final Examination

10 May 2013, 12:30–14:30 CDT

Problem 1	_____	(20 pts)
Problem 2	_____	(15 pts)
Problem 3	_____	(20 pts)
Problem 4	_____	(20 pts)
Problem 5	_____	(25 pts)
Exam Total	_____	(100 pts)

Alias _____

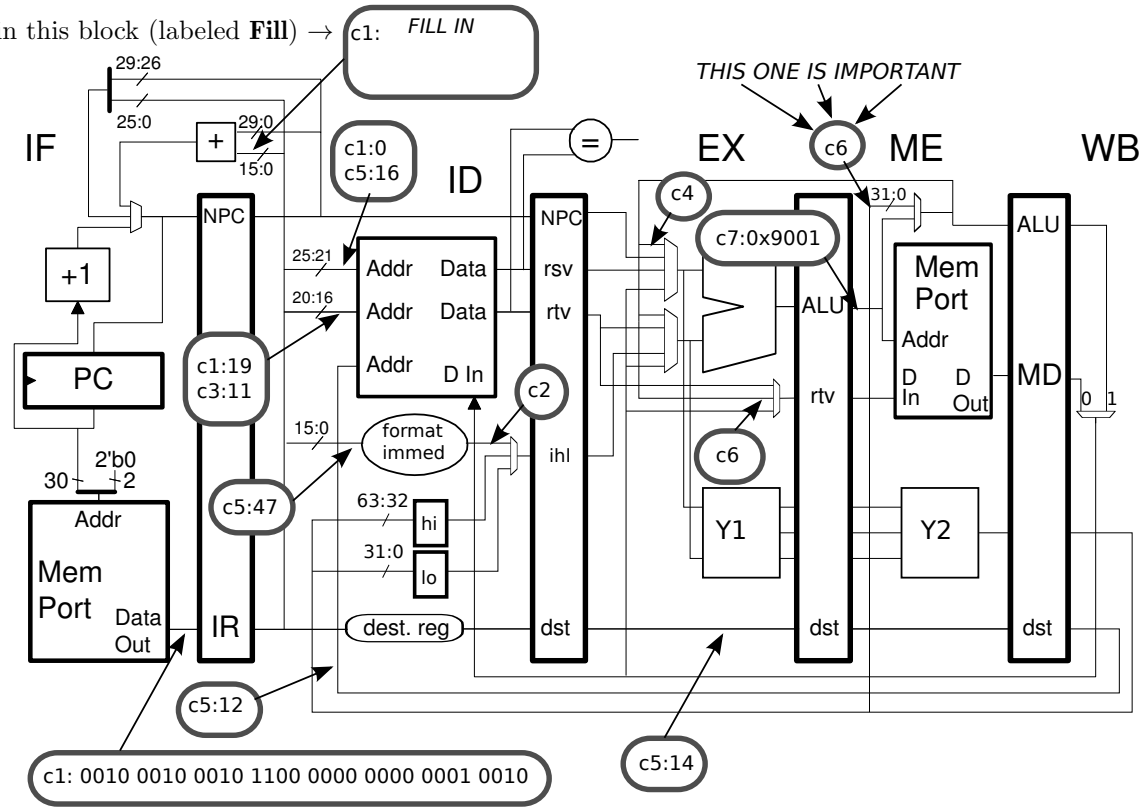
Good Luck!

Problem 1: (20 pts) The MIPS implementation below includes a two-stage integer multiply unit (`Y1` and `Y2`) for the `mult` instructions, similar to the MIPS implementation covered in Homework 4. Some wires are labeled with cycle numbers and values that will then be present. For example, c5:14 indicates that at cycle 5 the pointed-at wire will hold a 14. Other wires just have cycle numbers, indicating that they are used in that cycle. *Note that instruction addresses are provided.*

☐ Write a program consistent with these labels.

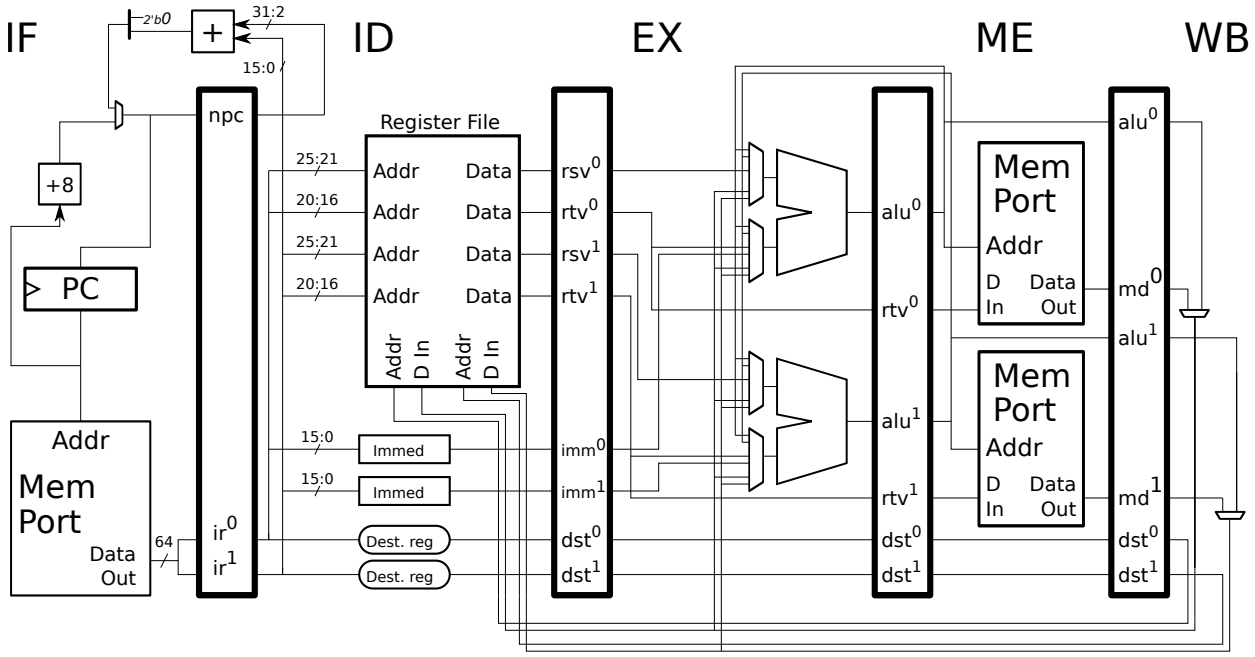
☐ All register numbers and immediate values can be determined.

☐ Fill in this block (labeled **Fill**) →



#	Cycle:	0	1	2	3	4	5	6	7	8
0x1000		IF	ID	EX	ME	WB				
0x1004			IF	ID	EX	ME	WB			
0x1104				IF	ID	EX	ME	WB		
0x1108					IF	ID	EX	ME	WB	
0x110c						IF	ID	EX	ME	WB
#	Cycle:	0	1	2	3	4	5	6	7	8

Problem 2: (15 pts) Illustrated below is a 2-way superscalar MIPS implementation. Design the hardware described below. You can use the following logic blocks (with appropriate inputs) in your solution: The output of logic block isALU is 1 if the instruction's result is computed by the integer ALU. The output of logic block rtSrc is 1 if the instruction uses the `rt` register as a source. The output of logic block isLoad is 1 if the instruction is a load.



(*a*) Design logic to generate a signal named `STALL`, which should be `1` when there is a true (also called data or flow) dependence between the two instructions in `ID`.

☐ Control logic to detect true dependence in `ID` and assign `STALL`.

(*b*) The code fragment below should generate a stall in our two-way superscalar implementation when the two instructions are in the same fetch group. However this particular arithmetic/load pair is a special case in which the stall is not necessary when the right bypass path(s) and control logic are provided. *Hint: Something about the* `lw` *makes it a special case.*

```
0x1000: add r1, r2, r3
0x1004: lw r4, 0(r1)
```

☐ Add the bypass path(s) needed so that the code executes without a stall.

☐ Add control logic to detect this special case and use it to suppress the stall signal from the first part.

Problem 3: (20 pts) Code producing the branch patterns shown below is to run on three systems, each with a different branch predictor. All systems use a 2^{14} entry BHT. One system has a bimodal predictor, one system has a local predictor with a *12-outcome local history*, and one system has a global predictor with a *12-outcome global history*.

(*a*) Branch behavior is shown below. The `r` outcomes for branch `B2` are random and are modeled by a Bernoulli random variable with $p = .25$ (taken probability is .25). Answer each question below, the answers should be for predictors that have already warmed up.

```
B1:  T   N   T   T   T   N    T   N   T   T   T   N    ...
B2:   T   N   T   r   T   N    T   N   T   r   T   N    ...
B3:    T   T   T   T   T   T    T   T   T   T   T   T   ...
```

☐ What is the accuracy of the bimodal predictor on branch `B1`?

☐ What is the approximate accuracy of the bimodal predictor on branch `B2`? ☐ Explain.

☐ What is the minimum local history size needed to predict `B1` with 100% accuracy? Ignore branch `B2` for this question.

☐ What is the accuracy of the local predictor on branch `B2`, ignoring the effect of `B1`? ☐ Explain.

☐ Describe a situation in which branch `B2` is mispredicted using the local predictor due to the effect of `B1`. The low bits of the address of `B1` and `B2` are different so there is no chance of a BHT collision. How frequently do such mispredictions occur? *Grading Note: The original exam and the Spring 2014 homework question did not include the statement about the low address bits.*

☐ How many possible GHR values will be present when predicting branch `B3`? It's okay to show patterns that include `r`'s, but indicate how many of each pattern there are.

Problem 3, continued: Recall that the local predictor uses a 12-outcome local history and a 2^{14}-entry BHT.

(*b*) Draw a diagram of the local predictor hardware used for making a prediction but omit the hardware for updating the predictor. The input to your hardware should be `PC`, and output should be a 1-bit quantity (`0` for not-taken, `1` for taken).

- Be sure to show the BHT and PHT.
- Assume that `PC` is the address of a branch. (That is, don't worry about detecting non-branch instructions.)
- Don't predict the target, just the direction (taken or not-taken).

☐ Local predictor hardware for predicting branch direction.

☐ Label wires with bit ranges (*e.g.* 31:26) or number of bits, as appropriate.

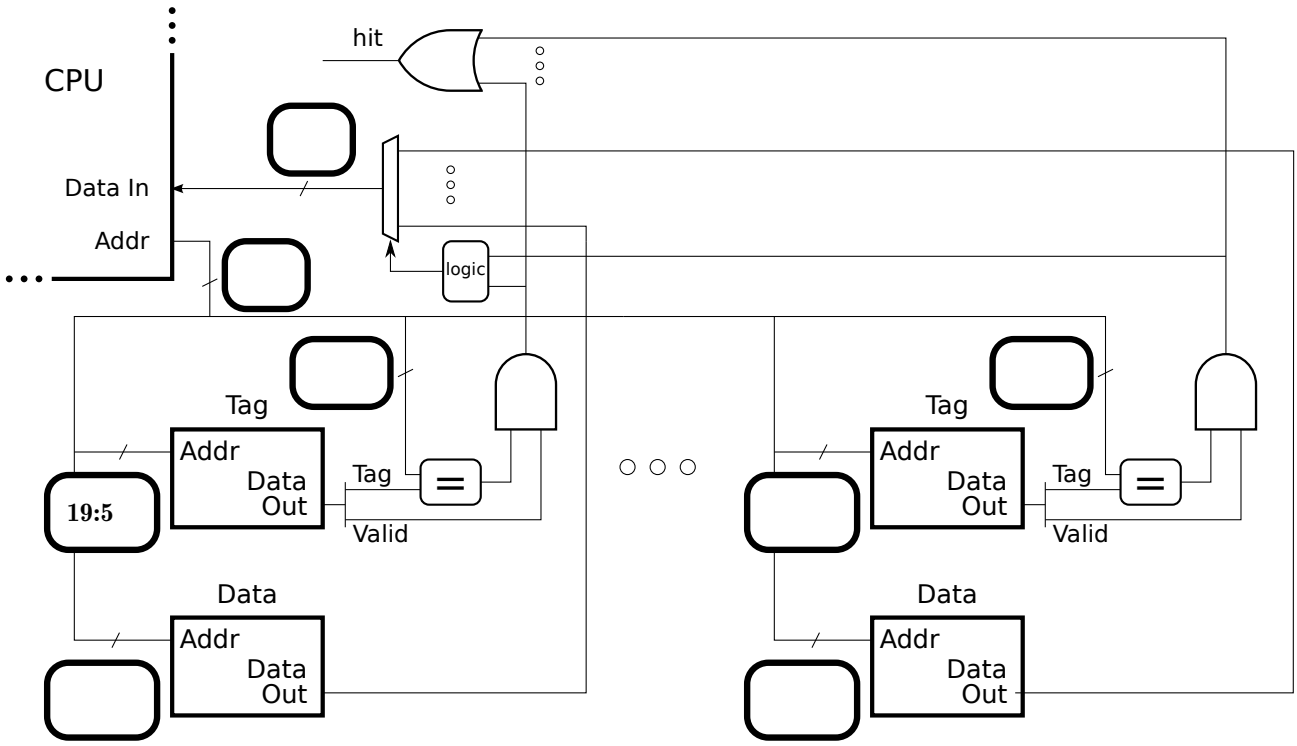
(*c*) How much memory is needed to implement the local predictor. Only include storage for predicting the branch direction, do not include storage for predicting the branch target.

☐ Storage needed to implement local predictor. ☐ Indicate unit.

Problem 4: (20 pts) The diagram below is for an 8-way set-associative cache. Hints about the cache are provided in the diagram and also in the address bit categorization just below the diagram.

(*a*) Answer the following, formulæ are fine as long as they consist of grade-time constants.

☐ Fill in the blanks in the diagram.



☐ Complete the address bit categorization. Label the sections appropriately. (Index, Offset, Tag.)

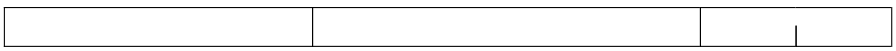
Address: 

☐ Cache Capacity (Indicate Unit!!):

☐ Memory Needed to Implement (Indicate Unit!!):

☐ Line Size (Indicate Unit!!):

☐ Show the bit categorization for a **fully associative** cache with the same capacity and line size.

Address: 

Problem 4, continued: The problems on this page are **not** necessarily based on the cache from the previous page. The code in the problems below run on a 4 MiB (2^{22} byte) 4-way set-associative cache with a line size of 256 bytes.

Each code fragment starts with the cache empty; consider only accesses to the arrays.

(*b*) Find the hit ratio executing the code below.

```
Complex sum = {0,0};
Complex *a = 0x2000000; // sizeof(Complex) == 16  Each element loaded with 1 load insn.
int i;
int ILIMIT = 1 << 11;    // = 2^11

for (i=0; i<ILIMIT; i++) sum += a[ i ];
```

☐ What is the hit ratio running the code above? Show formula and briefly justify.

(*c*) Find the smallest value for `ASIZE` that will minimize the hit ratio (make things as bad as they can get) of the code below.

```
struct Some_Struct {
  double val;        // sizeof(double) = 8
  double norm_val;
  double a[ASIZE];  };

  const int BSIZE = 1 << 10;
  Some_Struct *b;
  for ( int i = 0;  i < BSIZE;  i++ ) sum += b[i].val;
  for ( int i = 0;  i < BSIZE;  i++ ) b[i].norm_val = b[i].val / sum;
```

☐ Smallest value of `ASIZE` to minimize hit ratio:

Problem 5: (25 pts) Answer each question below.

(*a*) Indicate whether each feature below is a feature of an ISA or the feature of an implementation.

☐ Number of stages in pipeline. *Circle One*: ISA or Implementation

☐ Number of integer registers. *Circle One*: ISA or Implementation

☐ Number of bits in an integer register. *Circle One*: ISA or Implementation

☐ Whether an adjacent pair of instructions, such as `add r1,r2,r3; sub r4,r1,r2` will generate a stall. *Circle One*: ISA or Implementation

(*b*) Describe the problem with each of the following MIPS code fragments.

☐ Problem with fragment below:

```
addi r1, r2, 0x123456
```

☐ Problem with code execution on our FP pipeline.

```
add.s f1, f2, f3   IF ID A1 A2 A3 A4 WF
sub.s f4, f1, f5      IF ID A1 A2 A3 A4 WF
```

☐ Problem with code execution on our 5-stage pipeline.

```
lw r1, 2(r3)      IF ID EX ME WB
beq r1, r4 TARG      IF ID EX ME WB
xor r6, r7, r8          IF ID EX ME WB
TARG:
add r9, r10, r11           IF ID EX ME WB
```

(*c*) Consider the following two equal-cost design options:

A dual-core chip, each core is 4-way superscalar and dynamically scheduled.

A 16-core chip, each core is 2-way superscalar and statically scheduled.

Both have the same clock frequency.

☐ What is the peak execution rate of each chip, in IPC?

☐ Describe a situation in which the dual-core chip is a better choice than the 16-core chip.

☐ Describe a situation in which the 16-core chip is a better choice than the dual-core chip.

(*d*) Consider a $5n$-stage implementation created by splitting each stage of a 5-stage implementation into n pieces.

☐ How important is it to have a higher clock frequency in the $5n$-stage design (compared to the 5-stage design)? ☐ Explain.

(*e*) When an instruction raises an exception execution will switch to a trap handler.

☐ How is the trap handler address determined for MIPS?

☐ How is the trap handler address determined for SPARC?