Name _____

Computer Architecture
EE 4720
Final Examination

4 May 2015, 10:00–12:00 CDT

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

Alias _____

Good Luck!

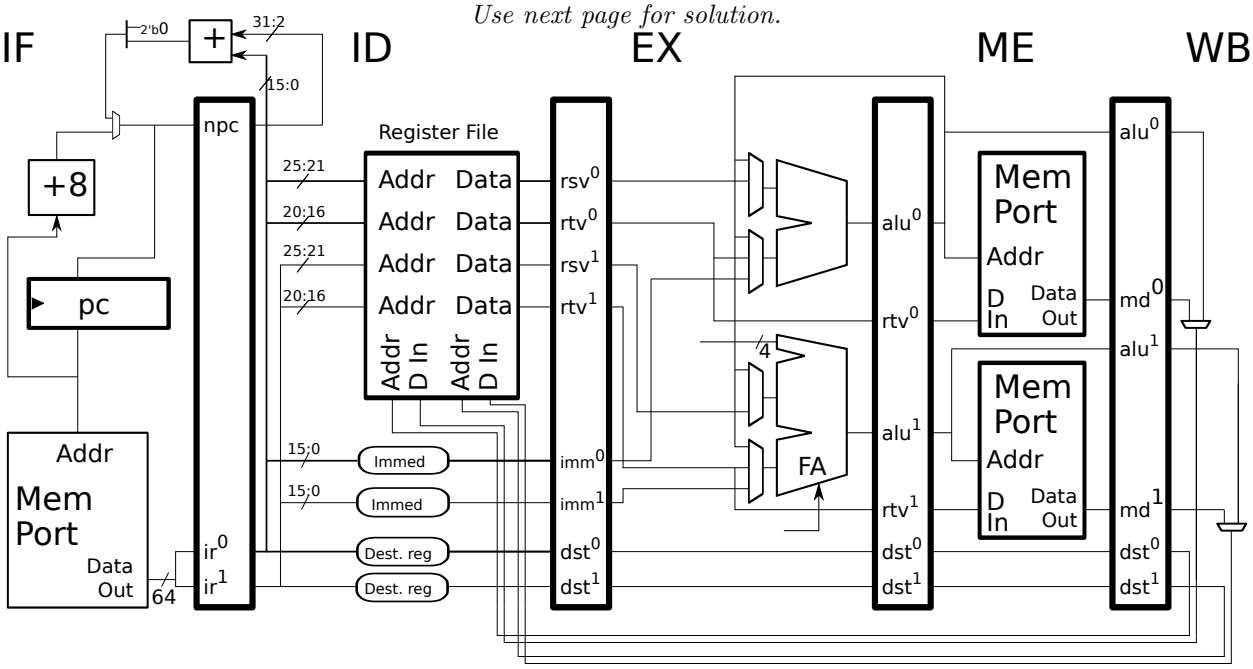
Problem 1: (20 pts) The instruction sequence below performs the calculation `r2+3+r5` and requires two MIPS instructions to do so. If these two instructions were in the same fetch group in a two-way superscalar processor then the second one would stall, this is shown in the execution below.

```
addi r1, r2, 3   IF ID EX ME WB
add  r4, r1, r5  IF ID -> EX ME WB
```

An *FA-ALU* has two regular inputs, and a third input reserved for small quantities, here for values < 16, and it can produce a result in the same time as an ordinary two-input ALU. An FA-ALU can be used to execute instruction pairs like the one above without a stall, call such an execution a *fused add*.

Illustrated below is a 2-way superscalar implementation with an FA-ALU. Most bypass paths have been removed to provide space for the solution to this problem. In addition to a 4-bit input for the third operand, the FA-ALU has a control input, `FA`, which should be set to `1` when the third input is to be used.

Use next page for solution.



Use next page for solution.

(*a*) Add datapath connections to this implementation so that the FA-ALU can be used for fused adds. *Hint: The datapath changes are not just for that new FA-ALU input.*

☐ Add datapath so that the correct operands are delivered to the FA-ALU for fused adds.

☐ Non-fusable sequences must still execute correctly. (Don't break existing functionality.)

☐ Make reasonable cost and clock frequency tradeoffs.

(*b*) Add control logic.

☐ Design logic to generate the `FA` signal for the FA-ALU and deliver it in the correct cycle.

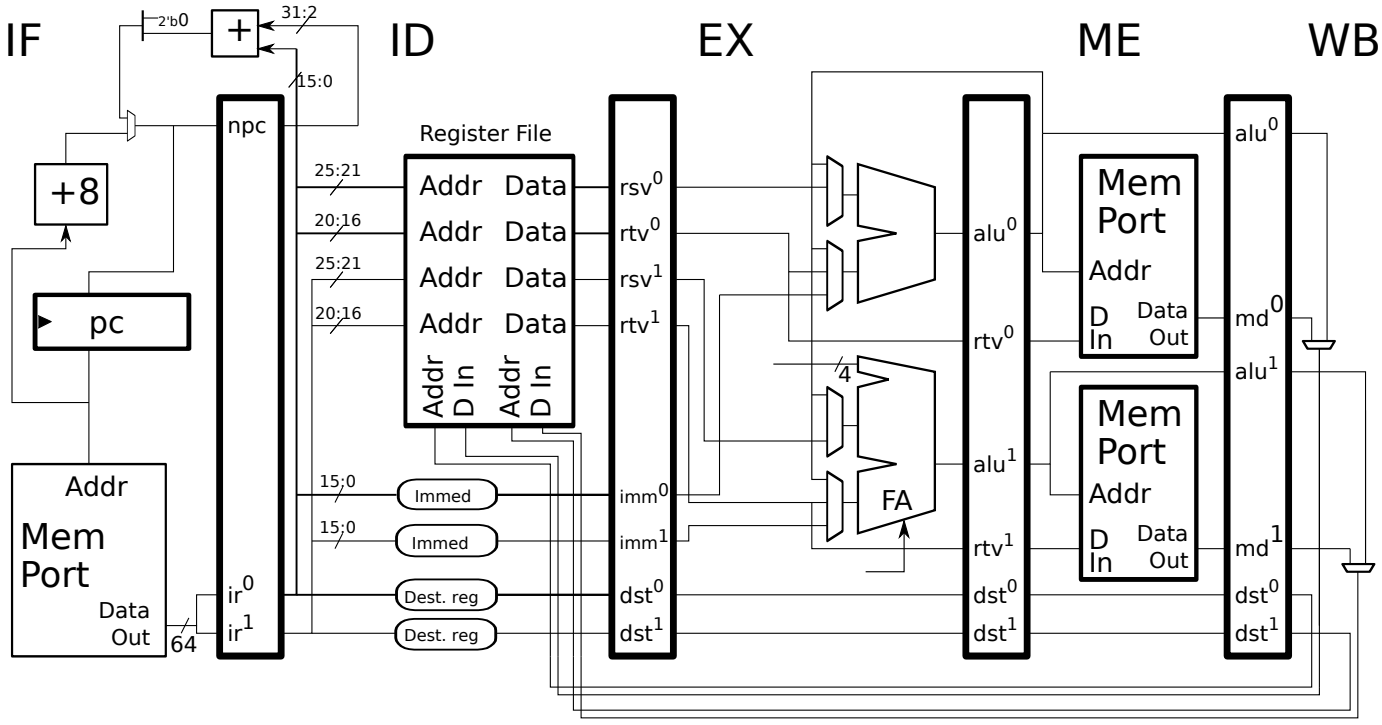
☐ Consider ☐ instruction type, ☐ dependencies, and ☐ operand size.

☐ Add control signals for the datapath added in the previous part.

☐ Make reasonable cost and clock frequency tradeoffs.

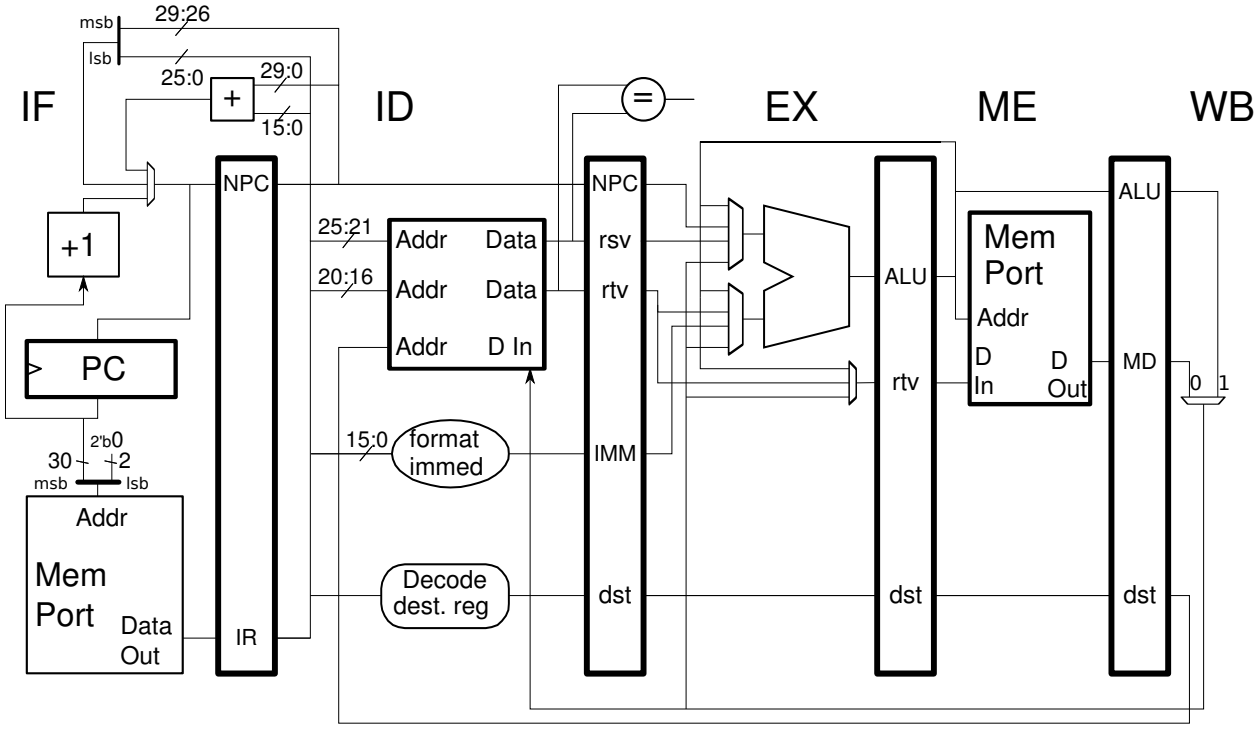
Problem 1, continued: You can use logic blocks such as isADD and isADDI in your solution.

```
addi r1, r2, 3    # Sample fusable sequence.
add  r4, r1, r5
```



Problem 2: (20 pts) Show the execution of the code fragments below on the illustrated MIPS implementations. All branches are taken.

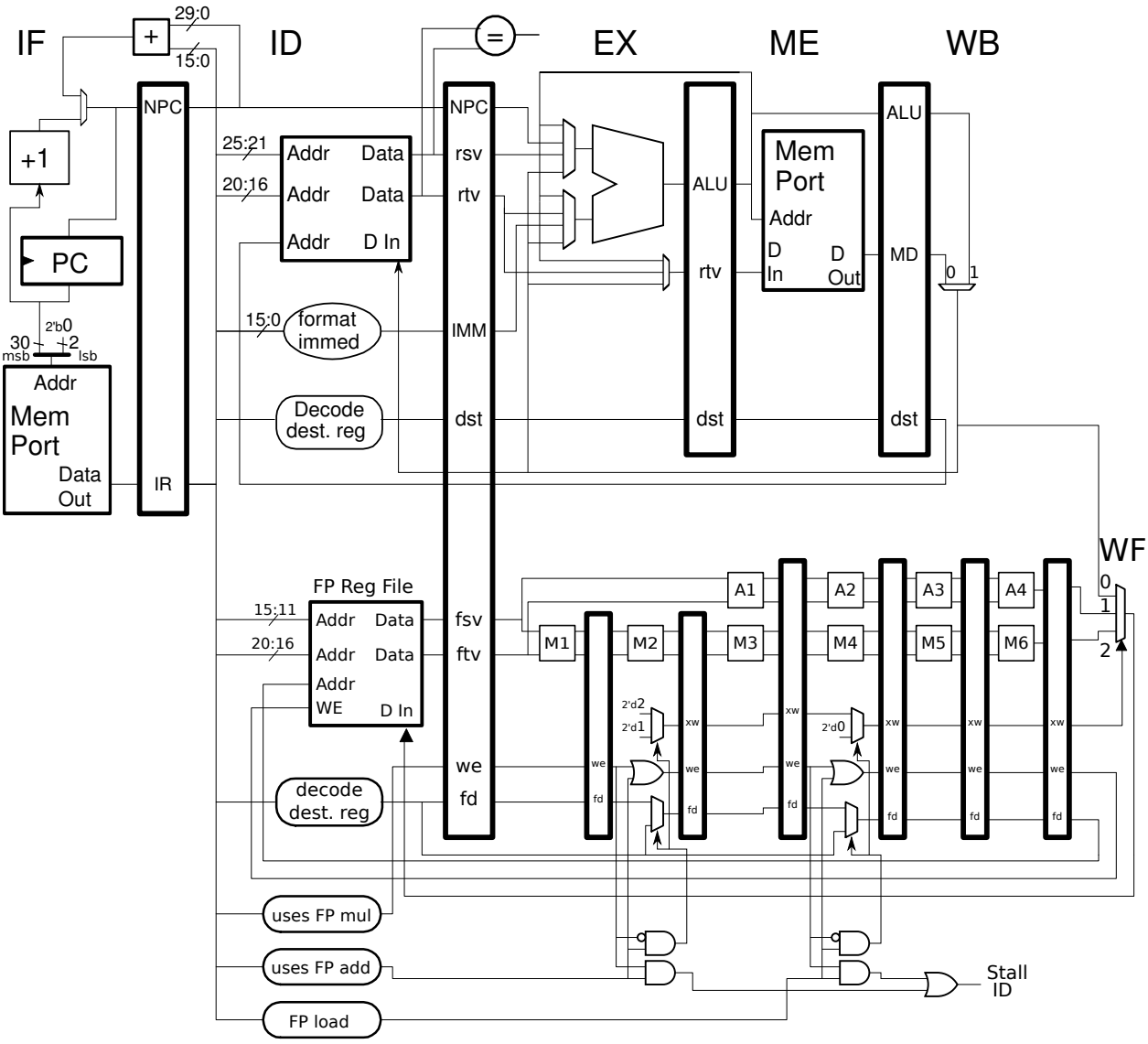
(*a*) Show the execution of the code below on the following implementation. The branch is taken.



☐ Show execution. ☐ Doublecheck for dependencies.

```
 add r4, r2, r3

 lw r6, 8(r4)

 sub r1, r6, r5

 beq r7, r1 TARG

 and r8, r7, r10

 or  r11, r12, r13

 xor r14, r11, r8

TARG:
 sw r1, 0(r2)
```

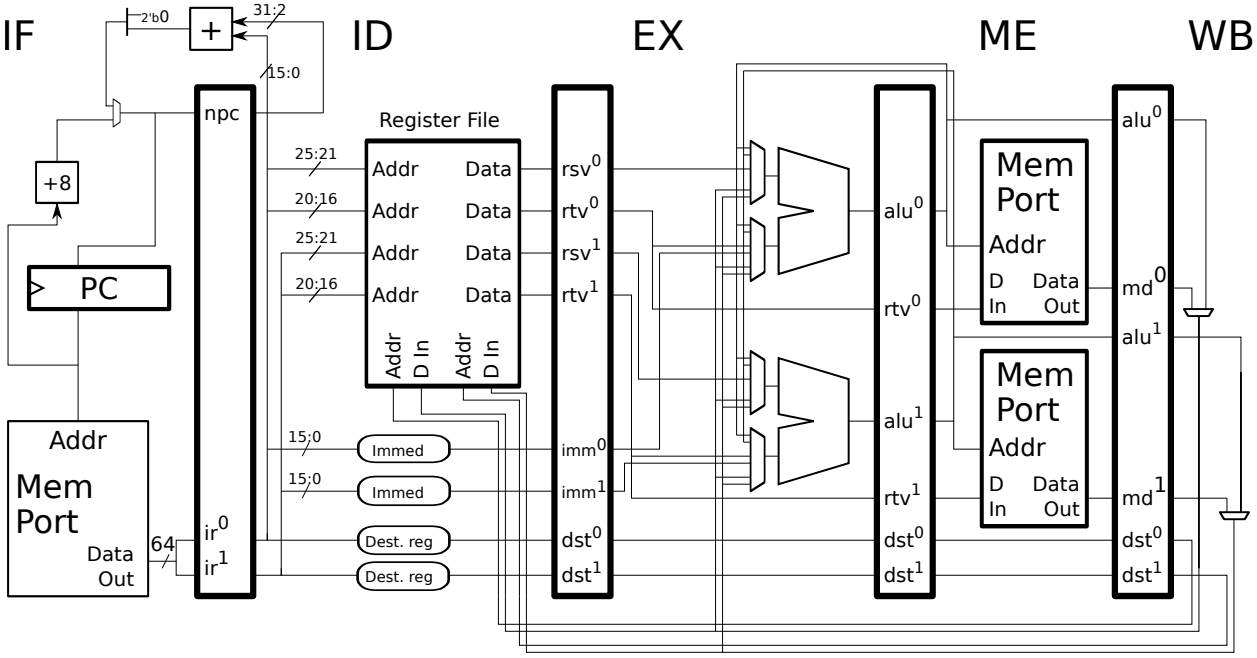
(*b*) Show the execution of the MIPS FP code below on the following implementation. Assume that **there is** a bypass from `WF` to `M1` and `A1`.



☐ Show code execution. ☐ Doublecheck for dependencies.

```
add.s  f6, f2, f3

add.s  f1, f7, f8

lwc1   f9, 0(r10)

mul.s  f4, f1, f9
```

(*c*) Show the execution of the MIPS code below on the following 2-way superscalar implementation. The branch is taken. Fetch groups in this implementation must be 8-byte-aligned.



☐ Show code execution. ☐ Show squashed instructions with an `x`.

☐ Doublecheck for dependencies. ☐ Account for aligned fetch groups.

```
START: Instruction address of add is 0x1000

 add r4, r2, r3

 lw r6, 8(r4)

 sub r1, r6, r5

 beq r7, r15 TARG

 and r8, r7, r10

 or  r11, r12, r13

 xor r14, r11, r8



 sll r16, r17, 8

TARG: Instruction address of sw is 0x2004

 sw r1, 0(r2)

 lui r15, 0x1234
```

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^{30} 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. Branch `B2` consists of a pattern in which there will be either 2, 4, or 6 `T` outcomes followed by exactly four `N` outcomes. After the fourth `N` the probability of exactly two `T`s is $\frac{1}{3}$ and the probability of exactly four `T`s is $\frac{1}{3}$.

Answer each question below, the answers should be for predictors that have already warmed up.

```
B1:  N    T    T    N    N    N      N    T    T    N    N    N   ...
B2:   T{2,4,6}  N    N    N    N      T{2,4,6}  N    N    N    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 accuracy of the bimodal predictor on branch `B2`? ☐ Explain.

☐ What is the minimum local history size needed to predict `B1` with 100% accuracy?

☐ What is the accuracy of the local predictor on branch `B2`, after warmup. ☐ Explain.

☐ What is the minimum GHR size to predict `B1` with about 100% accuracy. ☐ Explain, and state any assumption about `B2`'s behavior.

Problem 3, continued:

In this part consider the same predictors as on the previous page, except this time the BHT has 2^{14} entries.

Branch `B1` below is perfectly biased not taken. It executes r_1 times over a short time interval, followed by a big time gap before its next bunch of r_1 executions. Branch `B2` behaves similarly, except that it is biased taken and executes in bunches of length r_2 executions.

```
0xee4720  B1:   N..N r1                          N..N r1
# ...
          B2:             T..T r2                          T..T r2
```

(*b*) Choose an address for branch `B2` that will result in a BHT collision with branch `B1`. (Branch `B1` is at address `0xee4720`.)

☐ Address for `B2` that results in a collision.

(*c*) Assume that branch `B1` and `B2` collide in the BHT. For what positive values of r_1 and r_2 will this collision be most harmful? For what range of positive values of r_1 and r_2 will collisions be least harmful?

☐ Worst values of r_1 and r_2 for collision. ☐ Explain.

☐ Favorable range of positive values of r_1 and r_2 for collision. ☐ Explain.

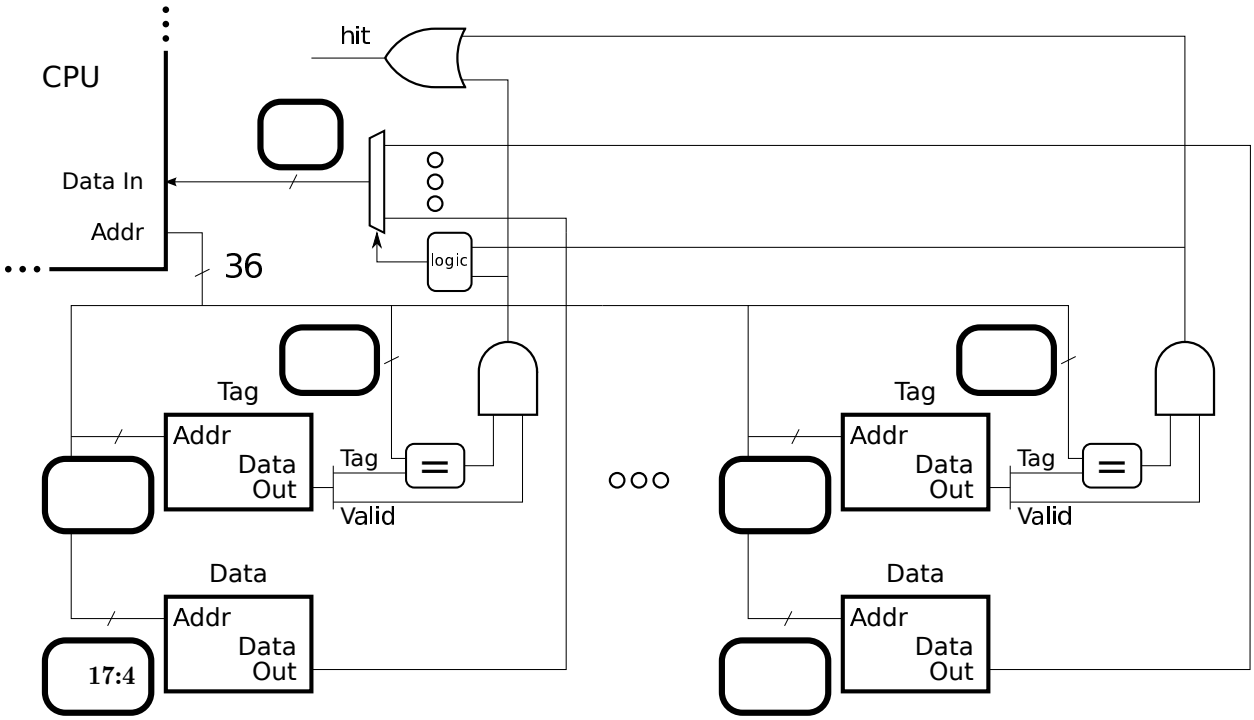
(*d*) A tag can be placed in the BHT to detect collisions. Which branch will this help?

☐ Branch that would benefit from tag: `B1` or `B2`. ☐ Explain.

Problem 4: (20 pts) The diagram below is for a 4 MiB (2^{22} B) set-associative cache with a line size of 128 bytes. Hints about the cache are provided in 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:

0

☐ Associativity:

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

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

Address:

Problem 4, continued: The problems on this page are **not** 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 64 bytes. Each code fragment starts with the cache empty; consider only accesses to the arrays.

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

```
int sum = 0;
half *a = 0x2000000; // sizeof(half) == 2
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*) The code appearing below is the same as the code from the 2014 Final Exam and 2015 Homework 6, except that two possibilities for `Some_Struct` are shown, Plan A, and Plan B. The caches too are identical except that here the line size is 64 bytes whereas in the prior problems it was 128 bytes. The value of `BSIZE` has been chosen to the largest value for which the hit ratio of the second loop will be 100% when using the Plan B struct. *Note: In the original exam* `norm_val` *was omitted from the first loop.*

```
struct Some_Struct { // Plan A
  double val;        // sizeof(double) = 8
  double norm_val;
  double a1[14];   };

struct Some_Struct {  // Plan B
  double val;         // sizeof(double) = 8
  double a1[7];
  double norm_val;
  double a2[7];    };

  const int BSIZE = 1 << 15;
  Some_Struct *b;
  for ( int i = 0;  i < BSIZE;  i++ ) sum += b[i].val + b[i].norm_val;  // First loop.
  for ( int i = 0;  i < BSIZE;  i++ ) b[i].norm_val = b[i].val / sum;   // Second loop.
```

☐ Explain why the Plan A struct will result in better performance on the first loop than the Plan B struct.

☐ Suppose that the value of `BSIZE` is to be chosen based on the Plan A struct, to a maximum value for which the second loop enjoys a 100% hit ratio. Explain why that value is the same as for Plan B.

Problem 5: (20 pts) Answer each question below.

(*a*) Which kind of implementations were RISC ISAs originally designed to simplify? Explain how a RISC ISA feature achieves this goal.

☐ Type of implementation RISC designed to simplify.

☐ Choose a feature and ☐ explain how it achieves the goal.

(*b*) What is the difference between a trap instruction and an instruction that raises an exception? Give an example of how each is used in a system with bug-free code.

☐ Difference between trap and exception.

☐ Example of what trap instruction used for.

☐ Example of what instruction exception used for.

Problem 5, continued:

(*c*) Consider a $5n$-stage scalar implementation and an n-way superscalar implementation, both derived from our 5-stage design with the goal of improving performance by as much as $n\times$. As n increases the clock frequency of the $5n$-stage implementation increases but the frequency of the n-way superscalar implementation decreases. Explain why. Assume that both types of system are well designed.

☐ Clock increases in the $5n$-stage system because ...

☐ Clock decreases in the n-way system because ...

(*d*) A company is preparing a run of the SPEC benchmarks for their new system. While trying to tweak compiler flags to get the best performance they discover a new optimization technique and implement it in their compiler. They re-test with that compiler and disclose the test results. When is the use of the modified compiler allowed under SPEC rules? When isn't that allowed under SPEC rules?

☐ Modified compiler allowed provided that ...

☐ This is a beneficial because ...

☐ Modified compiler isn't allowed if ...

☐ The modified compiler is not beneficial because ...