Name _____ *Formatted For 2-Sided Printing*

Computer Architecture
EE 4720
Final Examination

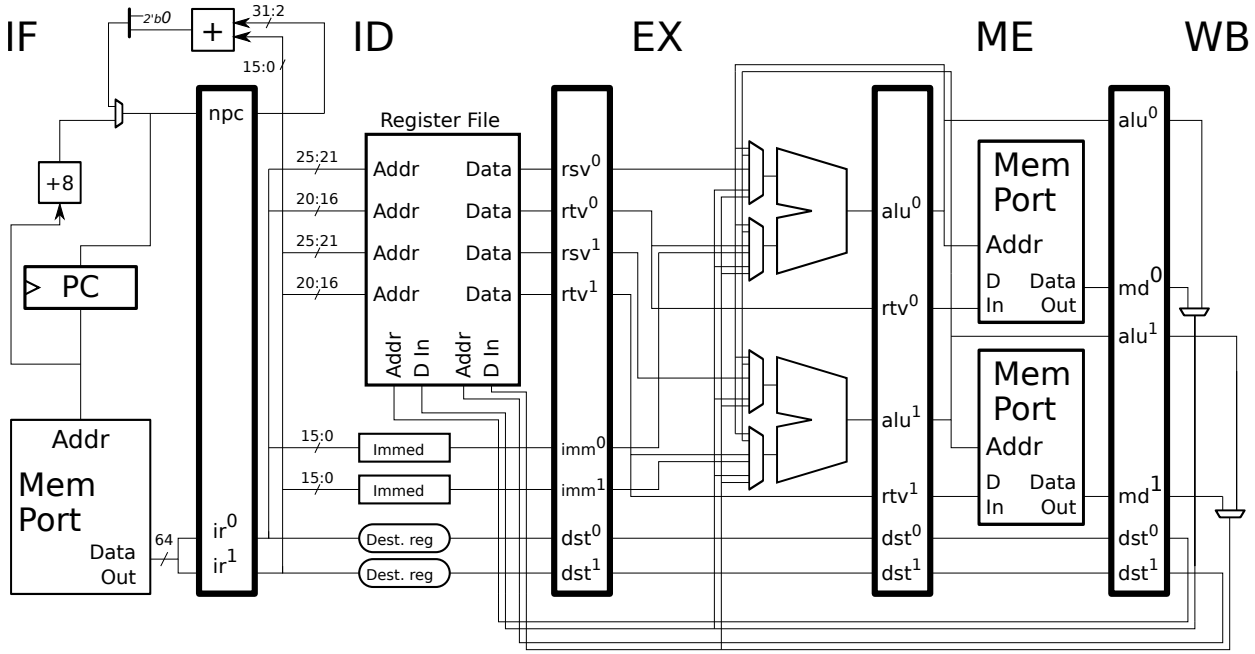
2 May 2018, 15:00–17:00 CDT

Problem 1	_____	(15 pts)
Problem 2	_____	(10 pts)
Problem 3	_____	(20 pts)
Problem 4	_____	(16 pts)
Problem 5	_____	(10 pts)
Problem 6	_____	(8 pts)
Problem 7	_____	(21 pts)
Exam Total	_____	(100 pts)

Alias _____

Good Luck!

Problem 1: (15 pts) Appearing below is the 2-way superscalar implementation used in class. As we usually assume, fetch groups are aligned and stalls must keep instructions within a stage in order.



(*a*) Show the execution of the code below on the implementation above.

☐ Show execution. ☐ Check for dependencies!!

```
LINE1: # Address of the first lw insn below is 0x1000

lw r1, 0(r2)

lw r3, 0(r1)

lw r4, 4(r1)

lw r5, 8(r1)

sw r5, 12(r1)

sw r4, 16(r1)
```

(*b*) Show the execution of the code fragment below on the illustrated implementation.

☐ Show execution and ☐ check for dependencies here too.

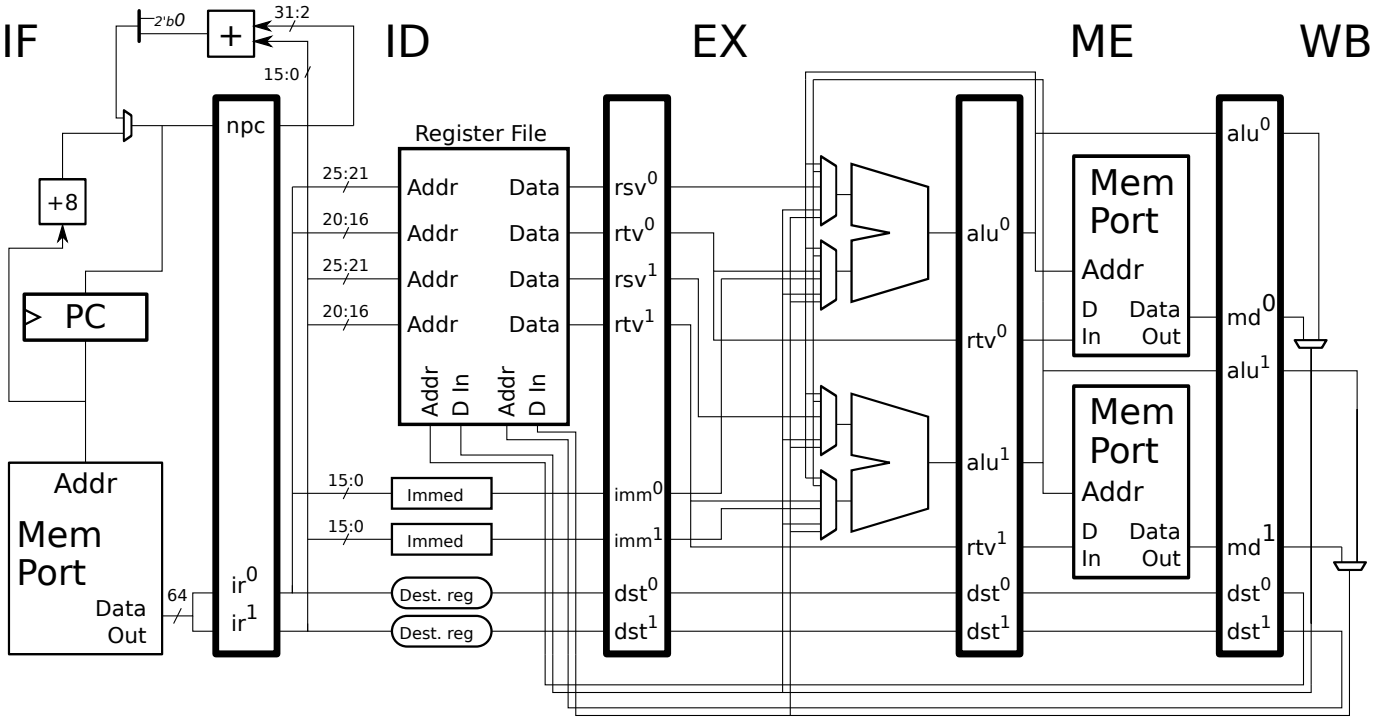
☐ Don't overlook the fact that the branch is taken.

☐ Pay attention to fetch groups and the aligned fetch restriction.

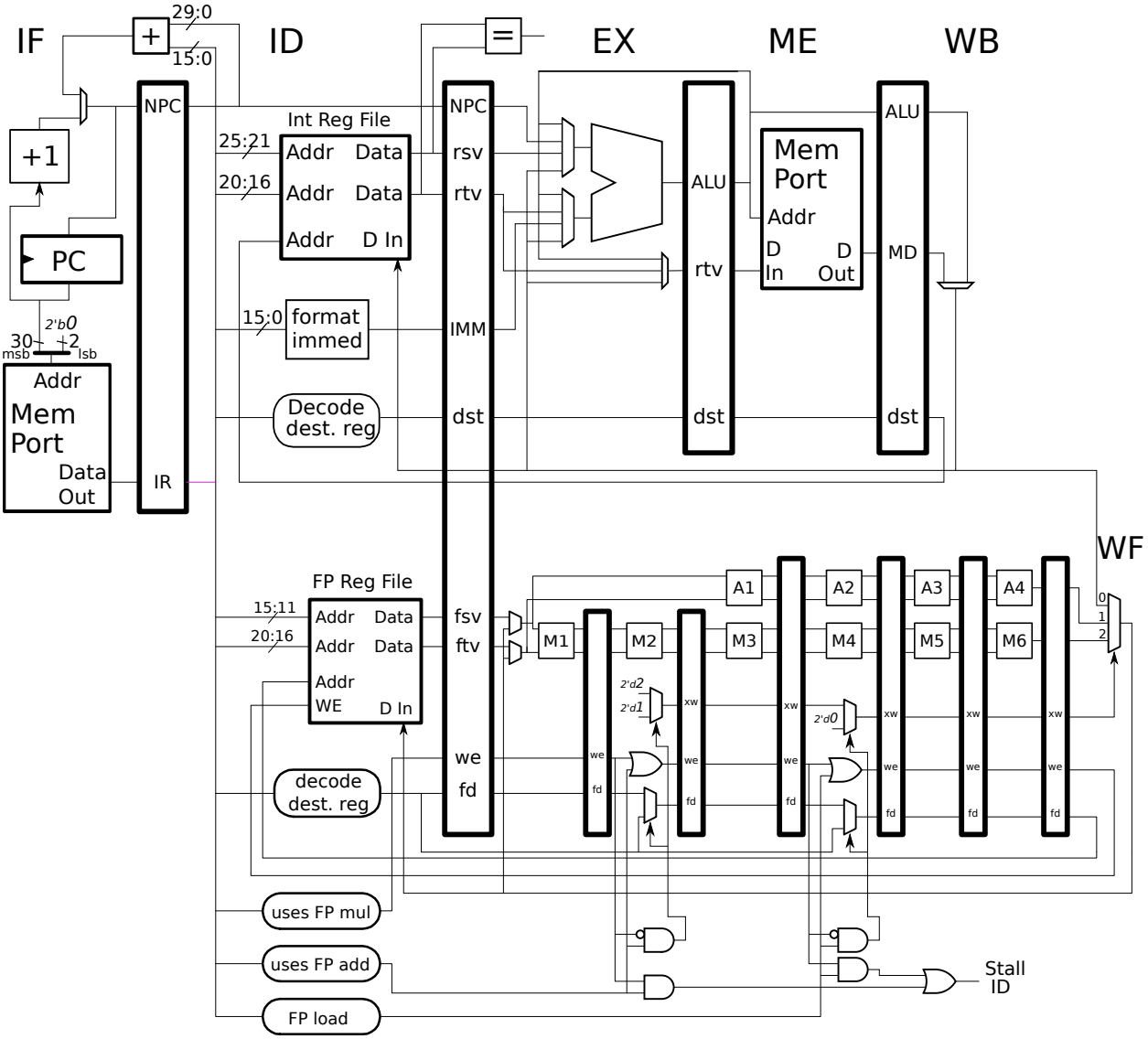
```
 # Address of beq is 0x1004
 beq r1, r1, SKIP1
 lw r2, 0(r3)
 sw r4, 0(r3)
 addi r3, r3, 4
SKIP1:  # Address of andi r2 is 0x2008
 andi r2, r2, 0xfff
 andi r6, r2, 0xff0
 add r7, r2, r6
 sub r8, r2, r6
```

(*c*) Appearing below is again our 2-way superscalar MIPS. Notice that the branch hardware shown can only provide the target for a branch in slot 1. Add hardware for providing the branch target of a branch in slot 0. **Do not** add hardware for checking the branch condition. **Do not** add control logic.

- [ ] Add hardware for a slot-0 branch.
- [ ] **Pay attention to cost**.
- [ ] Be sure the hardware computes the correct target address.


IF
ID
EX
ME
WB
2'b0
+
31:2
15:0
npc
+8
PC
Addr
Mem
Port
Data
Out
64
ir0
ir1
Register File
25:21
20:16
25:21
20:16
Addr
Data
Addr
D In
15:0
Immed
Dest. reg
rsv0
rtv0
rsv1
rtv1
imm0
imm1
dst0
dst1
alu0
rtv0
alu1
rtv1
dst0
dst1
Mem
Port
Addr
D
In
Data
Out
alu0
md0
alu1
md1
dst0
dst1

Problem 2: (10 pts) Appearing below is the execution of a bit more than two iterations of a loop on the illustrated MIPS implementation. The execution shows the use of a two-stage FP compare unit, `C1-C2`, by the `c.lt.s` instruction, but the unit isn't shown.




```
LOOP:  #          0  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 26 27 28 29 30 31 32 33 34 35 36
mul.s f1, f1, f2  IF ID M1 M2 M3 M4 M5 M6 WF
c.lt.s f1, f3        IF ID -------------> C1 C2 WF
bc1f LOOP               IF -------------> ID ----> EX ME WB
add.s f1, f1, f4                          IF ----> ID A1 A2 A3 A4 WF
LOOP:  #          0  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 26 27 28 29 30 31 32 33 34 35 36
mul.s f1, f1, f2                             IF ID -------> M1 M2 M3 M4 M5 M6 WF
c.lt.s f1, f3                                   IF -------> ID -------------> C1 C2 WF
bc1f LOOP                                                   IF -------------> ID ----> EX ME WB
add.s f1, f1, f4                                                              IF ----> ID A1 A2 A3 A4 WF
LOOP:  #          0  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 26 27 28 29 30 31 32 33 34 35 36
mul.s f1, f1, f2                                                                    IF ID -------> M1 M2 M3 M4 M5 M6 WF
```

(*a*) Compute the CPI of the execution of the loop above for a large number of iterations.

☐ Compute the CPI.

☐ Clearly show how the time for an iteration was determined, perhaps using the pipeline diagram.

(*b*) Reschedule the instructions to reduce the time needed to execute a large number of iterations of the loop. Add a `nop` if that helps. A correct solution will still have many stalls.

☐ Re-schedule to improve performance.

☐ Don't change what the loop is computing.

Problem 3: (20 pts) The MIPS implementation on the next page shows the two stages of the comparison units, `C1` and `C2`, but they are not connected to anything. The illustration also shows an `FCC` register that will hold the floating-point condition code value computed by compare instructions such as `c.lt.s`. Connect the comparison units and the `FCC` register so that they operate correctly and as described by the check items below. Notice that logic to detect FP branch instructions and FP compare instructions has been added to the `ID` stage near the bottom.

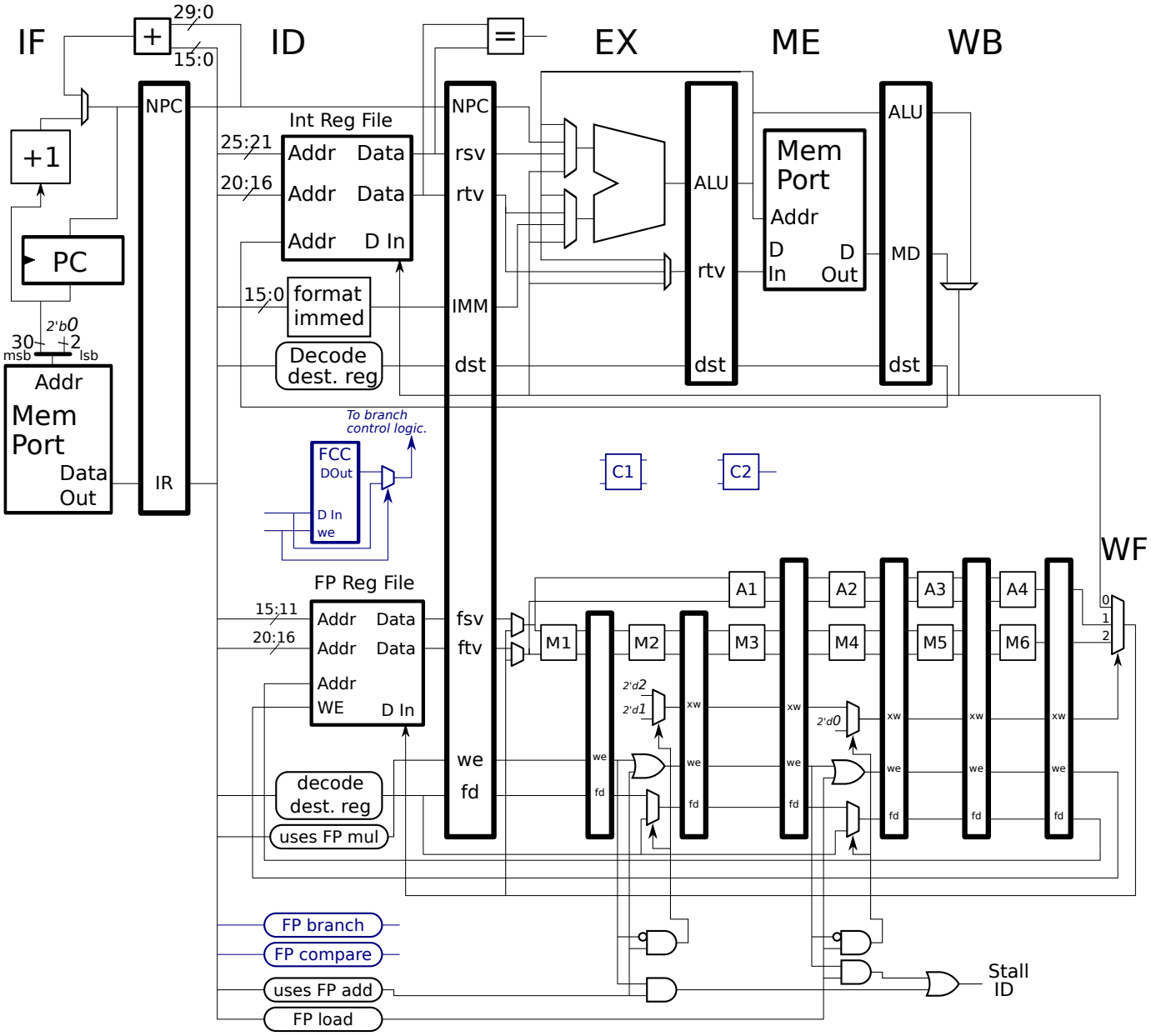
```
LOOP:     # Cycle  0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16
mul.s f1, f1, f2   IF ID M1 M2 M3 M4 M5 M6 WF
c.lt.s f1, f3         IF ID -------------> C1 C2 WF
bc1f LOOP                IF -------------> ID ----> EX ME WB
add.s f1, f1, f4                           IF ----> ID A1 A2 A3 A4 WF
          # Cycle  0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16
```

☐ Provide connections to `C1`, `C2`, and the two `FCC` inputs so that the code above executes as shown.

☐ Modify the control logic so that a compare does not arrive in `WF` in the same cycle as other FP instructions. (This is despite the fact that compares do not write the FP register file.)

☐ Modify the control logic so that the `Stall ID` signal is asserted for dependencies from compare to branches, such as occurs above with the `bc1f`.

☐ As always, pay attention to cost and performance and ☐ don't break existing functionality.


IF
ID
EX
ME
WB
WF
29:0
15:0
+
=
+1
PC
NPC
IR
Addr
Mem
Port
Data
Out
msb
lsb
30
2
2'b0
Int Reg File
25:21
20:16
Addr
Data
D In
15:0
format
immed
Decode
dest. reg
rsv
rtv
IMM
dst
ALU
MD
D
In
Out
To branch
control logic.
FCC
DOut
D In
we
C1
C2
FP Reg File
15:11
20:16
WE
fsv
ftv
fd
decode
dest. reg
uses FP mul
FP branch
FP compare
uses FP add
FP load
Stall
ID
M1
M2
M3
M4
M5
M6
A1
A2
A3
A4
xw
2'd2
2'd1
2'd0

Problem 4: (16 pts) Answer the following branch prediction questions.

(*a*) Code producing the branch patterns shown below is to run on two systems, each with a different branch predictor. All systems use a 2^{12} entry BHT. One system has a bimodal predictor and one system has a local predictor with an 8-outcome local history.

Branch `B2` starts with a random outcome, then repeats that same outcome two more times, followed by another random outcome followed by two more repeats of that. The random outcome is `T` with probability .3 and is independent of other outcomes. The following are possible `B2` outcome sequences: `TTT NNN NNN TTT TTT`. Note that the number of consecutive `T`'s or `N`'s must be a multiple of 3 and so the following **is not** a possible sequence of outcomes for `B2`: `TT NN T NNNN T`.

Answer each question below, the answers should be for predictors that have already warmed up. Show work or provide brief explanations.

```
B1:  T  T  N  T  T  T  N  N   T  T  N  T  T  T  N  N  ...

B2:   r  r  r  q  q  q  s  s  s ...
```

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

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

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

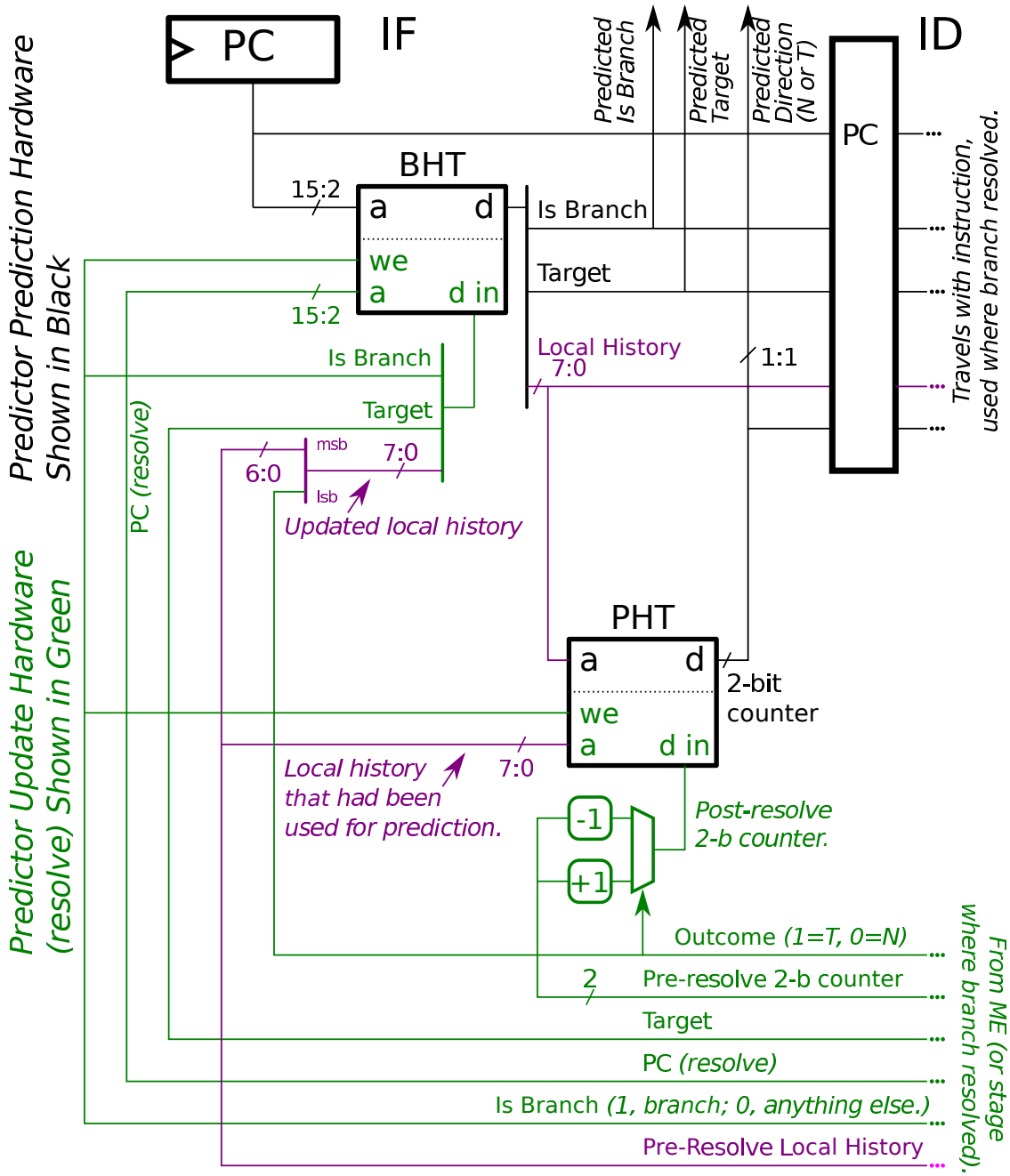
☐ What is the accuracy of the local predictor on branch `B2`?

(*b*) Appearing below is a diagram of a local predictor, showing in detail the logic for predicting the instruction in `IF` and for updating the predictor for the resolving branch. Modify the diagram so that it is a global predictor with an 8-outcome global history.

☐ Modify so that it is a global predictor.

☐ Remove hardware that's no longer needed.

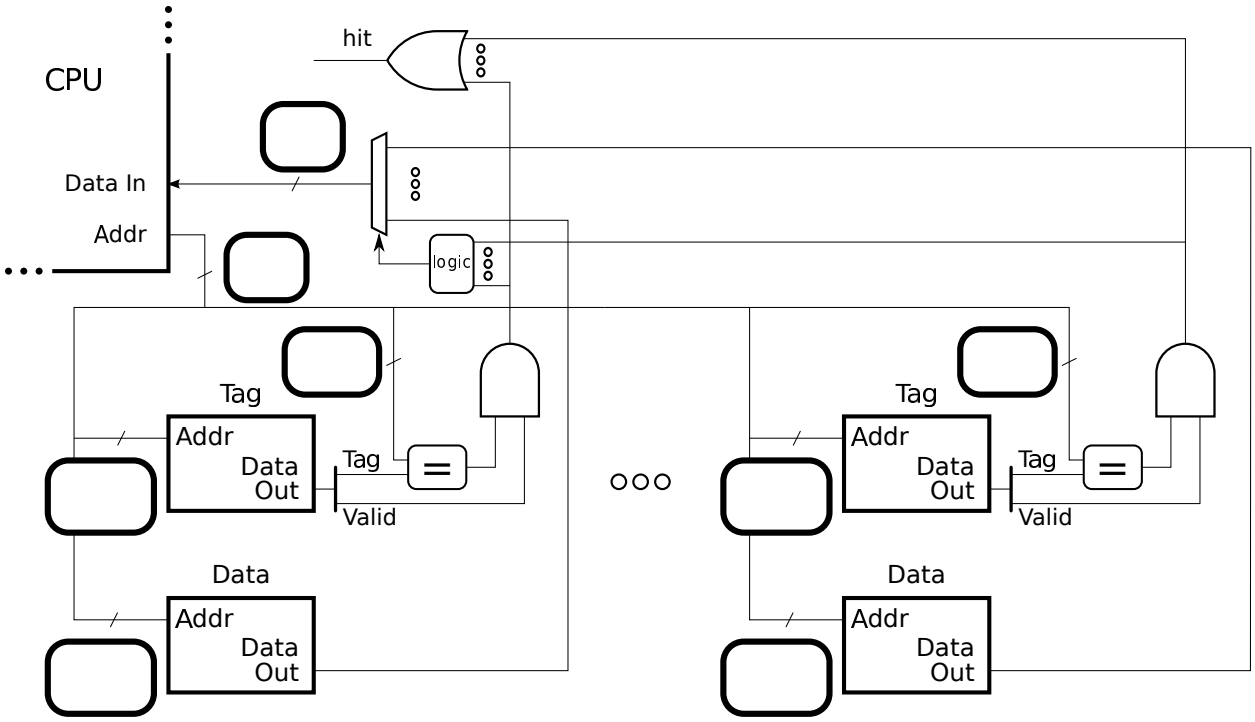
☐ Be sure to show the GHR (global history register).



Problem 5: (10 pts) The diagram below is for a 64 MiB, 4-way set-associative cache with a line size of 256 B, a bus width (w) of 8 B, for a 64 b address space. Helpful facts: $64\,\text{MiB} = 64 \times 2^{20}\,\text{B} = 2^{26}\,\text{B}$ and $256 = 2^8$.

(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 below. Label the sections appropriately. (Index, Offset, Tag.)

Address: ☐ ☐ ☐

0

☐ Memory Needed to Implement ☐ Indicate Unit!!:

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

Address: ☐ ☐ ☐

The problem on this page is **not** based on the cache from Part a. The code in the problem belows run on a cache with a line size of 256 B (which is 2^8 B). Each code fragment starts with the cache empty; consider only accesses to the arrays.

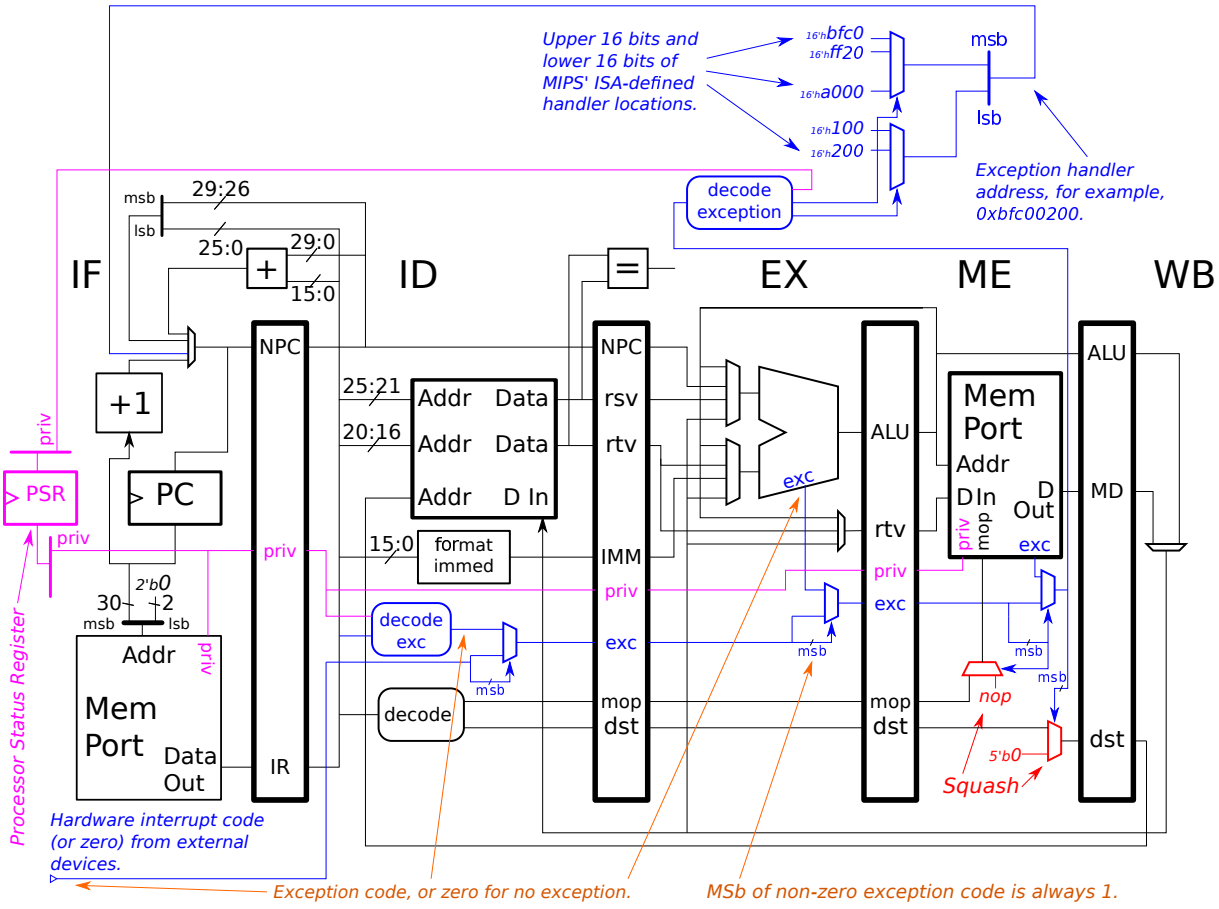
(*b*) Find the hit ratio executing the code below.

```
int sum = 0;
short *a = 0x2000000; // sizeof(short) == 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.

Problem 6: (8 pts) Appearing below is a MIPS implementation that includes hardware for interrupts (hardware interrupts, exceptions, and traps). An exception code, `exc`, is collected and passed down the pipeline. Its value indicates the type of hardware interrupt, exception, or trap that has been encountered, a value of 0 indicates no interrupt of any kind.



(*a*) Notice that the decode exc logic in the `ID` stage examines the opcode of the instruction in `ID` as well as the value of `ID.priv`. *Hint:* `priv` *is an abbreviation for privileged.* Some opcode values raise exceptions only when `ID.priv` is zero, others raise exceptions whether or not `ID.priv` is zero.

☐ Describe an instruction that raises an exception in `ID` only if `ID.priv` is zero.

☐ Why is it important that such an instruction raise an exception?

☐ Describe an instruction that raises an exception in `ID` whether or not `ID.priv` is zero.

(*b*) The illustrated hardware squashes the faulting instruction in `ME`, but no hardware is shown to squash any instructions that may be in the stages before `ME` nor for the stage after `ME`. That hardware may have been omitted for simplicity (the same reason that control logic is omitted) or because it is not needed.

☐ To implement precise exceptions should the instructions in the stages before `ME` be squashed? ☐ Explain in terms of the handler and what would go wrong if instructions were not treated the right way.

☐ To implement precise exceptions should the instructions in the stage after `ME` be squashed? ☐ Explain in terms of the handler and what would go wrong if instructions were not treated the right way.

Problem 7: (21 pts) Answer each question below.

(*a*) Show the encoding of the following MIPS instructions. Write the instruction name in opcode or func field values that cannot be determined.

```
0x1000:  beq r10, r11,  TARG
0x1004:  add r7, r8, r9

TARG:
0x1034:  lw r12, 14(r15)
```

☐ Encoding for `beq` from code fragment above. ☐ Pay attention to the branch target.

☐ Encoding for `add` from code fragment above:

☐ Encoding for `lw` from code fragment above:

(*b*) MIPS has one kind of memory addressing for all load and store instructions, such as in `lw r1, 2(r3)` where the immediate, 2, is added to the value in `r3`. A CISC ISA might have two versions of the load, `lw r1, (r3)`, which lacks an immediate (the immediate would be zero in MIPS), and `lwi r1, 2(r3)` for when an immediate is needed.

☐ What would be the benefit for the CISC ISA of having the no-immediate version of the `lw`?

☐ Why would MIPS and other RISC ISAs not realize the same benefit?

(*c*) A design team is considering removing a bypass connection to the ALU and adding a bypass connection to the branch resolve unit. This won't change the cost, they hope it will improve performance. "Simulation of this design change shows that performance drops by 5%," a sad-faced engineer announces. "We forgot to talk to the compiler people!," another excitedly points out, splashing hope and excitement around the room.

☐ What should they ask the compiler people?

(*d*) Someone preparing the SPECcpu benchmarks for their company's next product (currently under development) decides to replace one of the benchmark programs with an improved version, one which better reflects that company's customers. *Note: the emphasis below was added after the original exam, as was the phrase "not to market."*

☐ Should the company use the SPECcpu benchmarks in this way **to develop (but not to market)** their product?

With the substituted benchmark program the SPEC scores are higher (better). The company decides to release these higher SPEC results without mentioning the substitution.

☐ How does the design of SPECcpu make it likely that they will get caught?

(*e*) Compiler optimization is more important for a supercalar implementation than a scalar implementation.

☐ Optimization is more important for superscalar than scalar because:

☐ The important optimization is: