Name Partial Solution (Just 5a, 5b)

Formatted For 2-Sided Printing

Staple This Side

Computer Architecture
LSU EE 4720
Final Examination

Thursday, 8 May 2025 7:30-9:30 CDT

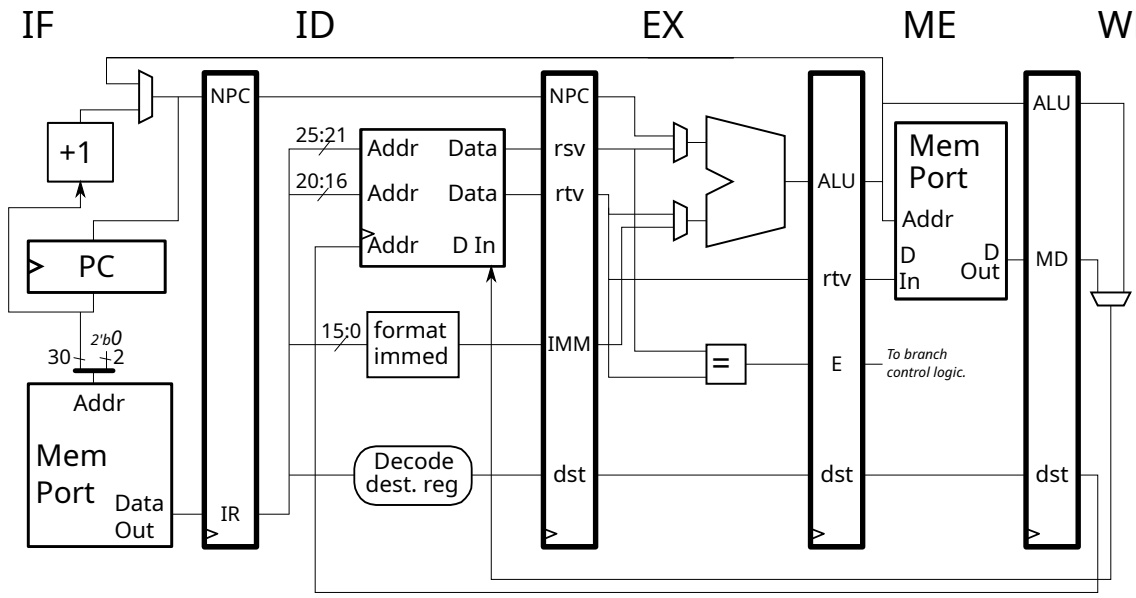
Problem 1	_____	(20 pts)
Problem 2	_____	(20 pts)
Problem 3	_____	(20 pts)
Problem 4	_____	(15 pts)
Problem 5	_____	(25 pts)
Exam Total	_____	(100 pts)

Alias Loose Bit

Good Luck! Thank you for your effort in EE 4720!

Staple This Side

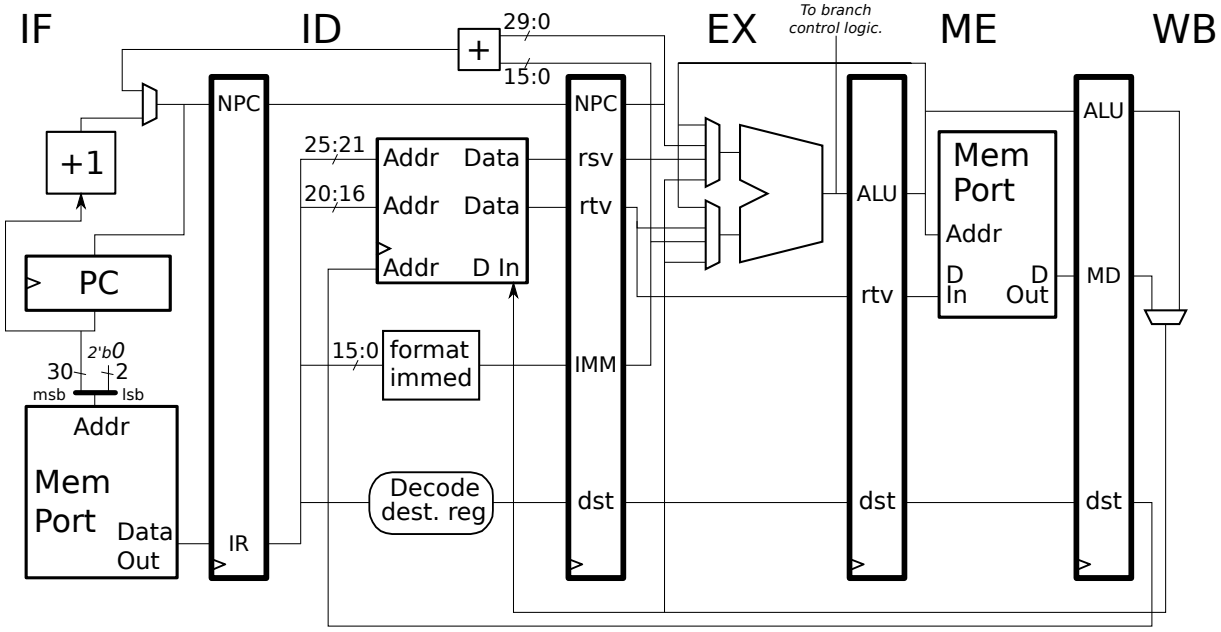
Problem 1: (20 pts) Appearing below are MIPS implementations and code fragments. Show execution (a pipeline execution diagram) of the code on the accompanying implementations.



☐ Show execution of the code below on the implementation above ▷ with the branch taken. ☐ Check for dependencies, including ☐ those for the branch.

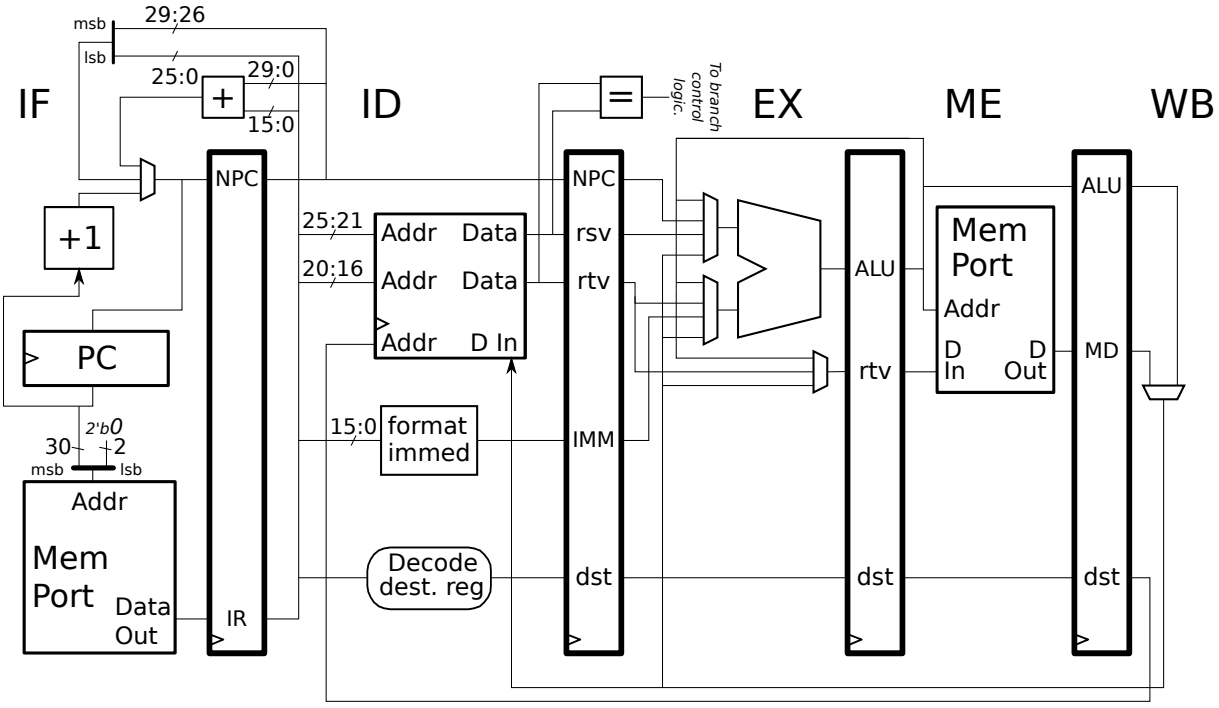
```
# Cycle

lw r2, 4(r5)

slt r1, r2, r3

bne r1, r0  SKIP

lw r4, 0(r5)

ori r7, r4, 0xaa

addi r10, r11, 12

sub r11, r12, r14

SKIP:
andi r6, r4, 0xf0

addi r5, r5, 4
```


Staple This Side



☐ Show execution of the code below on the implementation above (debuting in this exam) ▷ with the branch taken. ☐ Check for dependencies, including ☐ those for the branch.

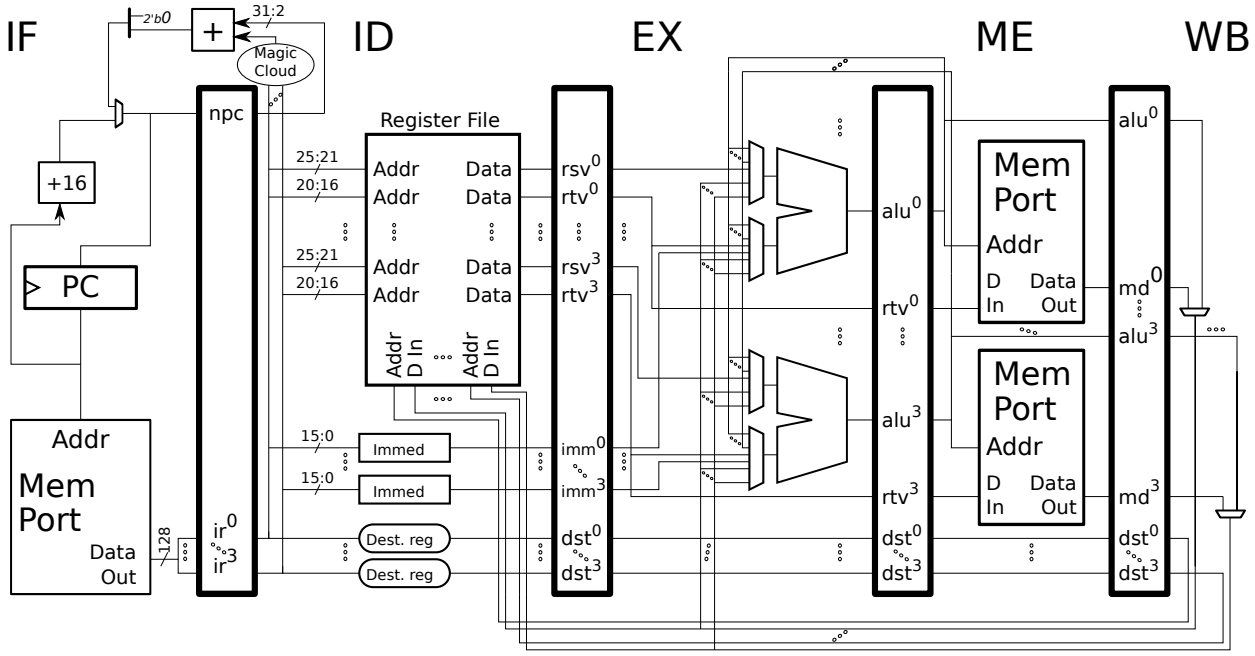
```
# Cycle

lw r2, 4(r5)

slt r1, r2, r3

bne r1, r0  SKIP

lw r4, 0(r5)

ori r7, r4, 0xaa

addi r10, r11, 12

sub r11, r12, r14

SKIP:
andi r6, r4, 0xf0

addi r5, r5, 4
```



☐ Show execution of the code below on the implementation above ▷ with the branch taken. ☐ Check for dependencies, including ☐ those for the branch.

```
# Cycle

lw r2, 4(r5)

slt r1, r2, r3

bne r1, r0  SKIP

lw r4, 0(r5)

ori r7, r4, 0xaa

addi r10, r11, 12

sub r11, r12, r14

SKIP:
andi r6, r4, 0xf0

addi r5, r5, 4
```


Staple This Side



☐ Show execution of the code below with ▷ the branch taken on the ▷ 4-way superscalar implementation above in which ▷ fetch groups are not aligned. ▷ Use course assumptions about instruction ordering and branch handling. ☐ Use an `x` to show where instructions gets squashed, ☐ don't omit `IF` of instructions that will be squashed. ☐ Check for dependencies. ☐ The code below is different than the previous parts.

```
# Cycle

lw r2, 4(r5)

sub r1, r2, r3

bne r9, r0  SKIP

lw r4, 0(r5)

ori r7, r8, 0xaa

SKIP:
and r6, r4, r7

addi r5, r5, 4
```

Problem 2: (20 pts) Recall that in Homework 5 Problem 2b the FP add hop paths from the 2024 final exam were to be replaced by bypass paths. Those bypass paths are shown in green on the facing page, which shows a slightly cleaned up solution.

(*a*) The bypass paths shown are correct, however the `WF.wa` pipeline latch output (on the right side) is unconnected and the `Addr` and `WE` inputs to the FP register file are wrong.

☐ Modify the hardware so that the `WE` (write enable) input to FP Reg File is correct. ☐ Don't forget about the unconnected `WF.wa`.

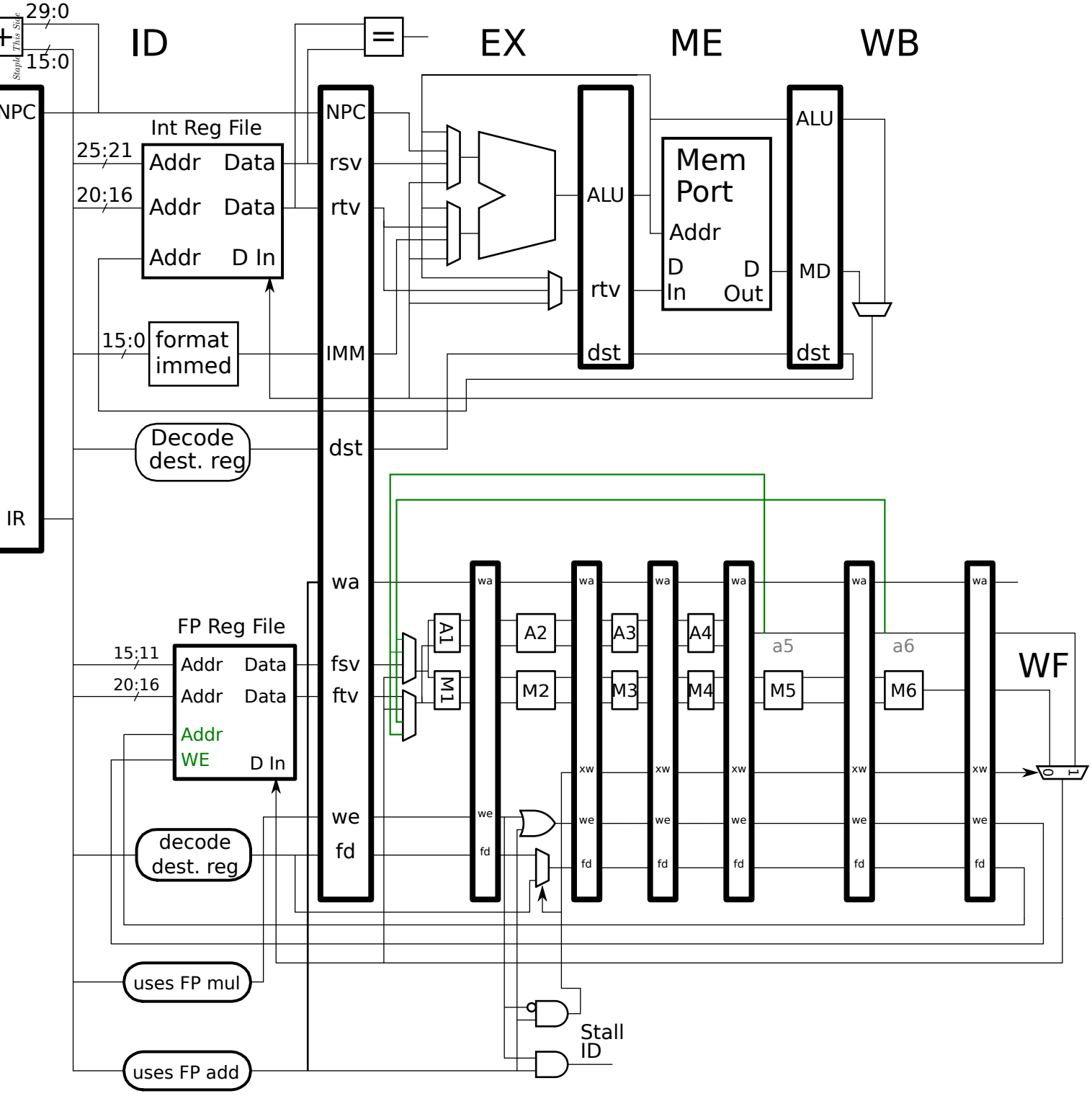
☐ Modify the hardware so that the `Addr` (destination register number) input to FP Reg File is correct. ☐ Think about deleting hardware rather than adding new hardware (other than reconnecting wires).

☐ Modify the hardware so that the select signal for the `WF` mux is correct. ☐ Think about cost, remove unneeded hardware.

☐ *Don't overdo it.* A solution to this part consists mostly of crossing things out and reconnecting things. Only a few gates need to be added.

Staple This Side

The IF stage is not shown to make space.



(*b*) Those expecting a bypass mux control logic problem have not been forgotten! On the facing page is the homework solution but with a part covered, and with parts of the integer pipeline off the page to make space. Assume that the values in the `wa`, `fd`, and `we` pipeline latches in stages `M3` and later are correct. Also assume that the instruction in `ID` does not need to stall. *Stall logic might be asked for in a 2026 homework assignment based on this problem.* Some control logic for the lower FP bypass mux is shown. The mux is blue to stand out.

☐ The wire labeled `SFTV` is the mux select signal. Provide a path for it to the lower bypass mux feeding `M1` and `A1`. ▷ Assume that the critical path passes through `M1` and `A1`.

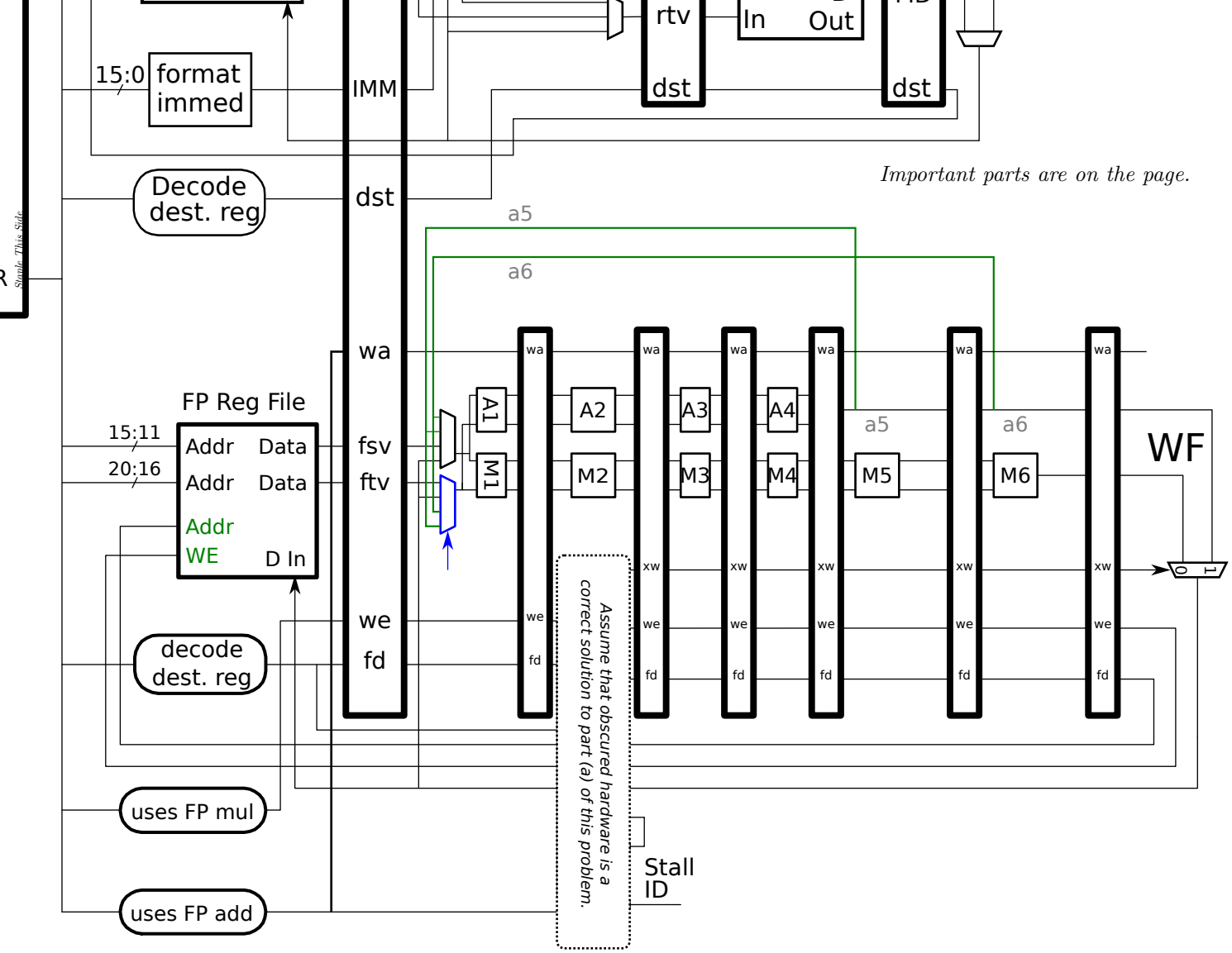
☐ Complete the hardware to compute `SFTV`. ☐ Note that `wa` and `we` should be used and their values can be based on your part (a) solution.

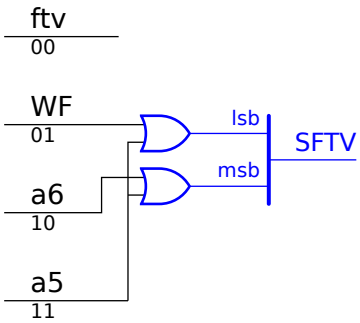
☐ Explain why a $\boxed{=}$ and not a $\boxed{='}$ should be used in the solution.

Use the execution diagram below to help keep track of where instructions might be when designing control logic.

```
# Cycle          0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15
mul.s F1, f2, f3 IF ID M1 M2 M3 M4 M5 M6 WF
add.s F7, f4, f5    IF ID A1 A2 A3 A4 a5 a6 WF
add.s F8, f6, f6       IF ID A1 A2 A3 A4 a5 a6 WF
add r1, r2, r3            IF ID EX ME WB
sub r1, r2, r3               IF ID EX ME WB
add.s f12, f11, F7              IF ID A1 A2 A3 A4 a5 a6 WF
add.s f10, f11, F1                 IF ID A1 A2 A3 A4 a5 a6 WF
add.s f14, f11, F8                    IF ID A1 A2 A3 A4 a5 a6 WF
# Cycle          0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15
```


Staple This Side


rtv
dst
In
Out
dst
15:0
format immed
IMM
Decode dest. reg
dst
a5
a6
wa
FP Reg File
15:11
20:16
Addr
Data
Addr
Data
Addr
WE
D In
fsv
ftv
we
fd
decode dest. reg
uses FP mul
uses FP add
A1
M1
A2
M2
A3
M3
A4
M4
M5
M6
a5
a6
WF
xw
we
fd
Assume that obscured hardware is a correct solution to part (a) of this problem.
Stall ID


ftv
00
WF
01
a6
10
a5
11
lsb
msb
SFTV

Problem 3: (20 pts) Answer the following branch prediction questions.

(*a*) Code producing the branch patterns shown below is to run on several systems, each with a different branch predictor. *Commas are used* to emphasize the repeating patterns. One system has a bimodal predictor, and the others use local predictors. Answer each question below, the answers should be for predictors that have already warmed up.

```
B1:  T  N  T  T  N  T  N  T,  T  N  T  T  N  T  N  T,  T  N  T  T  N  T  N  T,

B2:   N  N  T, N  N  T,  N  N  T, N  N  T,  N  N  T, N  N  T, ...
```

☐ What is the accuracy of the bimodal predictor on branch `B1`? ☐ Be sure to base the accuracy on a warmed-up 2-bit counter and repeating pattern.

For help in solving the local predictor problems the 8 possible B1 pattern rotations and the three possible B2 rotations are shown below. The table to the right shows them sorted to facilitate computing local history accuracies.

Unsorted		Sorted	
TnTTnTnTT	B1	TTnTTnTnT	B1
nTTnTnTTn	B1	TTnTnTTnT	B1
TTnTnTTnT	B1	TnTTnTTnT	B1
TnTnTTnTT	B1	TnTTnTnTT	B1
nTnTTnTTn	B1	TnTnTTnTT	B1
TnTTnTTnT	B1	TnnTnnTnn	b2
nTTnTTnTn	B1	nTTnTTnTn	B1
TTnTTnTnT	B1	nTTnTnTTn	B1
nnTnnTnnT	b2	nTnTTnTTn	B1
nTnnTnnTn	b2	nTnnTnnTn	b2
TnnTnnTnn	b2	nnTnnTnnT	b2

☐ What is the accuracy of an 8-outcome local history predictor on B1 ignoring B2.

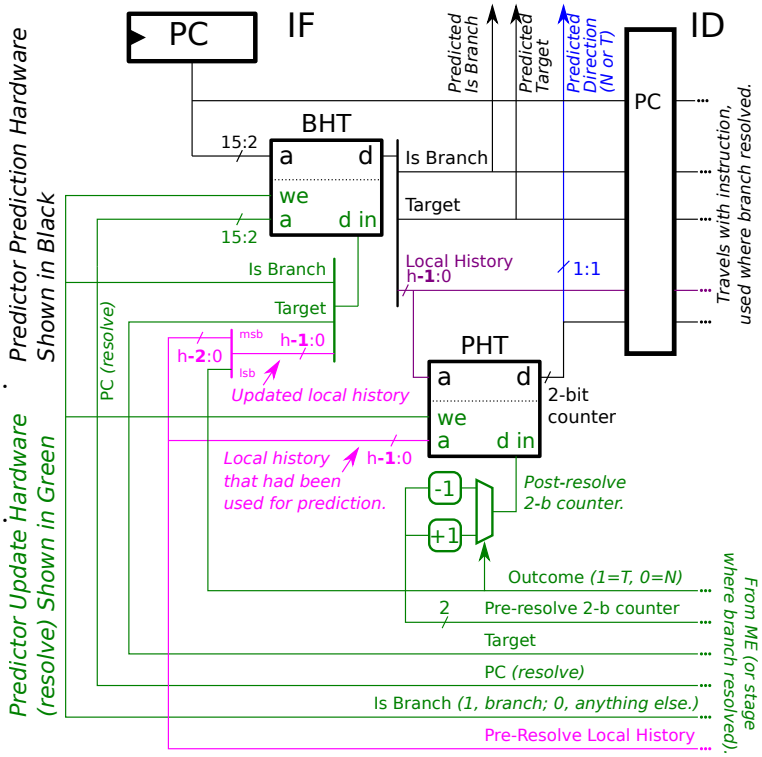
☐ What is the accuracy of a 3-outcome local history predictor on B1 ignoring B2.

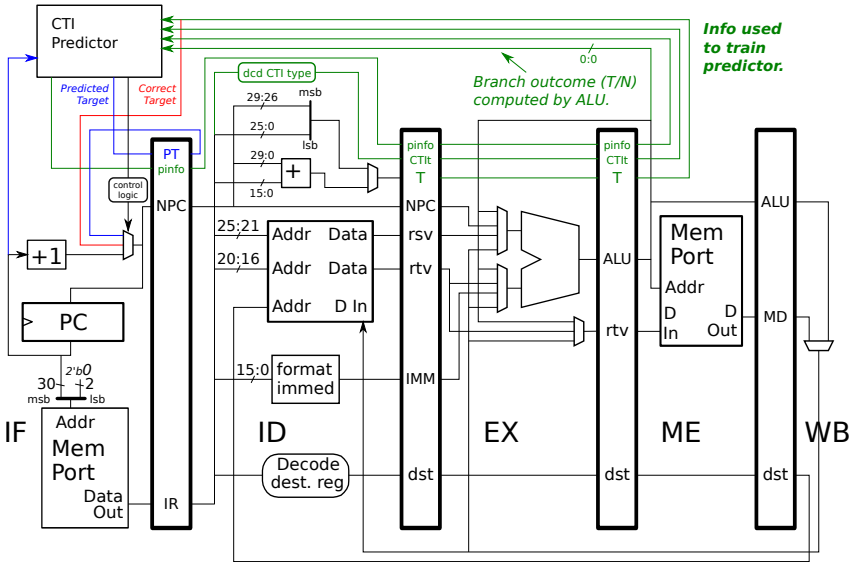
☐ What is the accuracy of a 3-outcome local history predictor on B1 **taking into account** B2. ☐ Note that the B2 pattern repeats faster than the B1 pattern.

Staple This Side

(*b*) The diagrams show a local predictor (to the right) and how a predictor might be integrated into the MIPS pipeline (below). Because only bits 15:2 of an instruction address are used in the BHT lookup it is possible to predict one branch, say at address `0x10024` in the code fragment below, using the BHT entry for a different branch, say at `0x30024`.

```
0x10024 beq r1, r2, TARGa   T T T...

# Far away.

0x30024 beq r3, r4, TARGb   T T T...
```



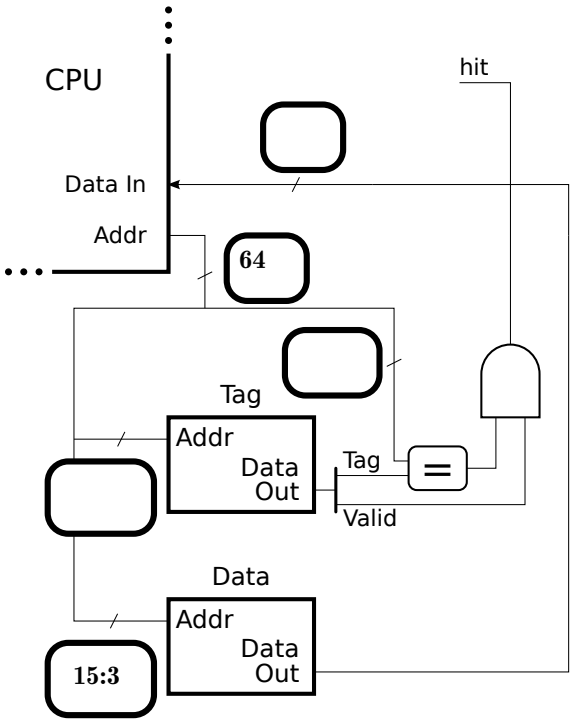


☐ Suppose both branches are always taken and that they are correctly predicted taken. Why would there still be a misprediction when predicting `0x10024` that can result in the wrong target being reached?

☐ Add hardware to detect the problem, the output should be labeled `MISPRED`. ☐ The hardware is very simple.

Problem 4: (15 pts) The diagram below is for a direct-mapped cache with a 64 byte (2^6 B) line size. The character size is a byte.
(*a*) Answer the following, formulæ are fine as long as they consist of grade-time constants.

☐ Fill in the blanks in the diagram.



☐ Show the number of bits or bit ranges in the unfilled boxes above.

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

Address: 

☐ Cache Capacity, in Bytes (how much data can it cache).

Staple This Side

Staple This Side

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 bytes (characters). The code fragment starts with the cache cold (empty); consider only accesses to the array. Of course, $2^8 = 256$.

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

```
int64_t sum = 0;
int64_t *a = 0x2000000; // sizeof(int64_t) == 8
int ILIMIT = 1 << 14;    // = 2^14

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

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

(*c*) The program from the previous part should run faster on a system with longer line size.

☐ In general, what is the disadvantage of a longer line size?

☐ Describe the characteristics of a program that runs better with shorter line sizes.

Staple This Side

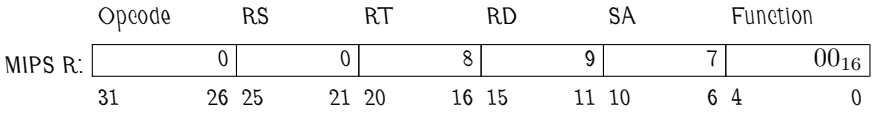
Problem 5: (25 pts) Answer each question below.

(*a*) Show the encoding of the MIPS instructions below.

☑ Show encoding. ☑ Label fields, and show specific values where possible. ☑ Pay attention to the format that this particular instruction uses.

```
sll r9, r8, 7
```

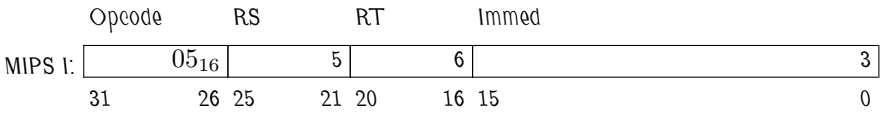
Solution appears below. Those who might have drawn a blank on the bit positions could look at one of the MIPS implementations. One thing that's not obvious from the MIPS implementations is that `sll` is a Type-R instruction despite the fact that the shift amount, 7 here, is an immediate. Another non-obvious thing is that the register source is encoded in the `rt` field and that the `rs` field must be set to zero. To receive full credit on the exam all fields except Function would need to have correct values. Those solving this as a homework would need to show all fields. In the Function field it would be okay to write "sll" or to note that the numeric value was just a wild guess. On an exam no points would be deducted for using the `rs` field for the source register.



☑ Show encoding of `bne`. ☑ Label fields, and show numeric values where possible, ☑ including the field encoding the branch target.

```
 bne r5, r6, SKIP     # Show encoding of this instruction only.
 sw r4, 5(r6)
 lui r8, 0x5678
 addi r8, r8, 0x1234
SKIP:
 add.s f1, f2, f3
```

Solution appears below. To receive full credit on the exam all fields except Opcode would need to have correct values. Those solving this as a homework would have to show all fields. In the Opcode field it would be okay to write "bne or to note that the numeric value was just a wild guess. Note that the `Immed` value is the number of instructions to skip, starting with zero at the delay slot (`sw` here).



(*b*) The code below uses MIPS pseudo instruction `li` (load immediate) several times.

```
li r1, 0x12345678
li r2, 0x990000
li r3, 0x8
```

☑ Rewrite the code using real MIPS instructions.

```
# SOLUTION
lui r1, 0x1234      # Two instructions needed to
ori r1, r1, 0x5678  # put 0x12345678 into a register.

lui r2, 0x99        # Just one instruction needed since 16 LSB are zero.

addi r3, r0, 0x8    # One instruction needed.
```


Staple This Side

Staple This Side

(*c*) The organization SPEC decides the programs that go into the very popular and influential SPECcpu benchmarks and how they should be run. The goal is to measure performance on a set of programs that typical users run.

Suppose there are rumors that company E has successfully bribed SPEC to omit benchmarks from the next SPECcpu suite that make company E's products look bad in comparison to other companies' products.

☐ How might ordinary users gauge whether these rumors are true based upon how SPEC is organized?

(*d*) One reason the VAX CALLS instruction was not suitable for a RISC ISA was because it might need to write several registers to memory.

☐ Why does the need to write several registers to memory make CALLS unsuitable for a RISC ISA?

☐ Why is it not a problem for a CISC ISA?

(*e*) The cost of an n-way superscalar implementation at some point will be proportional to n^2.

☐ What will contribute most to the n^2 cost?

(*f*) Provide two important reasons that an n-way superscalar RISC implementation will not have n times the instruction throughput (IPC) of a scalar RISC implementation on typical code.

☐ Two important reasons it does not have $n\times$ the instruction throughput on typical code.

Staple This Side