

Name _____

Computer Architecture
LSU EE 4720
Midterm Examination
Wednesday, 30 March 2022 9:30-10:20 CDT

Alias _____

Problem 1 _____ (30 pts)

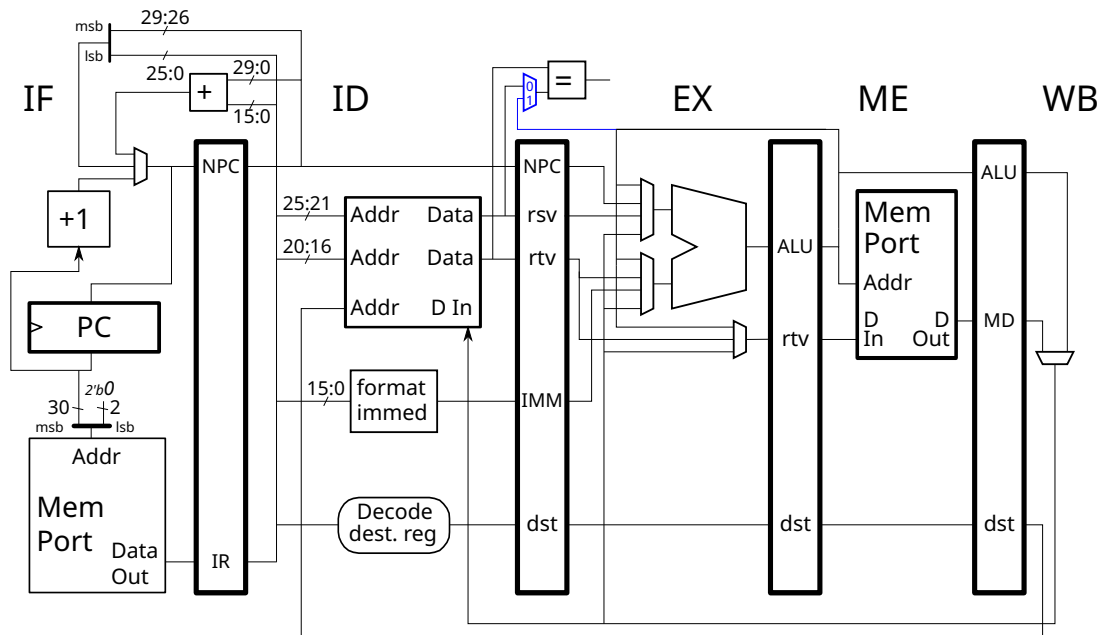
Problem 2 _____ (30 pts)

Problem 3 _____ (40 pts)

Exam Total _____ (100 pts)

Good Luck!

Problem 1: [30 pts] The code fragment below is to execute on the illustrated implementation. Show its execution and compute the instruction throughput (IPC) for a large number of iterations. (Note: `sh` is store half.)



- Show execution of code below.
- Mark each input to the `rtv` mux (in EX) and by the branch comparison (blue) mux used by the code below.
- Compute instruction throughput (IPC) for a large number of iterations.

```
lw r1, 0(r2)
```

LOOP:

```
addi r2, r2, 4
```

```
sh r1, -2(r2)
```

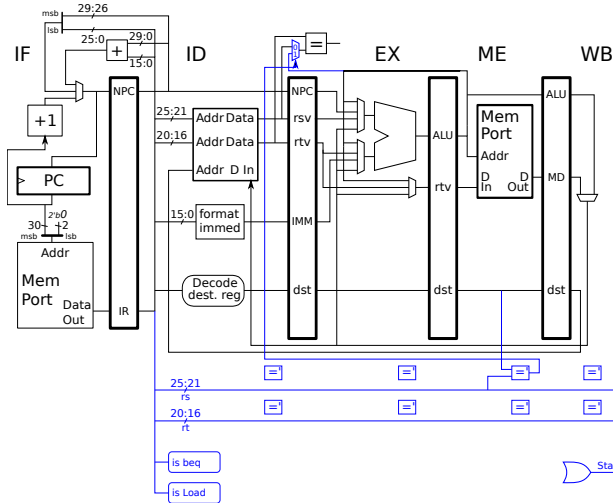
```
lw r3, -4(r2)
```

```
bne r3, r1, LOOP
```

```
lw r1, 0(r2)
```

Problem 2: [30 pts] Appearing below (and larger on the next page) is a MIPS implementation based on the solution to Homework 4 Problem 2, in which control logic for a branch bypass was designed. The diagram includes a **Stall** signal in the lower right. Add control logic to set the stall signal to 1 when a **beq** needs to stall due to a dependence that can't be bypassed.

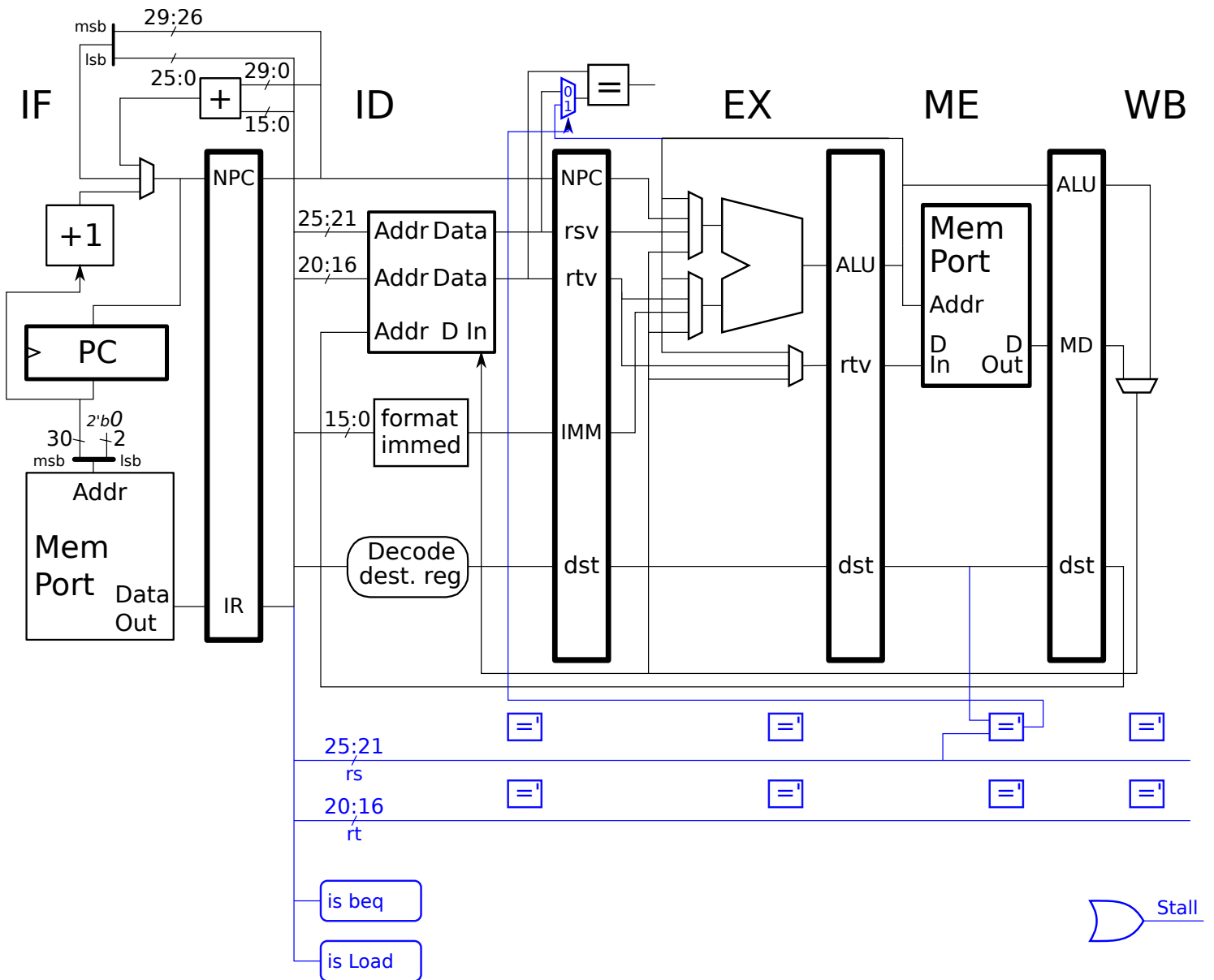
Appearing below are some code fragments. Complete executions are shown for the first two, in the others the executions are incomplete. The control logic should work with these code fragments. It may be helpful to complete the executions.



Use next page for solution.

# Cycle	0	1	2	3	4	5	6	7	
addi r1, r2, 3	IF	ID	EX	ME	WB				
beq r1, r4, TARG		IF	ID	->	EX	ME	WB		
nop			IF	->	ID	EX	ME	WB	
# Cycle	0	1	2	3	4	5	6	7	8
addi r1, r2, 3	IF	ID	EX	ME	WB				
beq r4, r1, TARG		IF	ID	----->	EX	ME	WB		
nop			IF	----->	ID	EX	ME	WB	
# Cycle	0	1	2	3	4	5	6	7	8
lw r1, 0(r2)	IF	ID	EX	ME	WB	# Note: Intentionally incomplete.			
beq r1, r4, TARG		IF	ID						
nop			IF						
# Cycle	0	1	2	3	4	5	6	7	8
lw r1, 0(r2)	IF	ID	EX	ME	WB	# Note: Intentionally incomplete.			
beq r4, r1, TARG		IF	ID						
nop			IF						
# Cycle	0	1	2	3	4	5	6	7	8
lw r9, 0(r2)	IF	ID	EX	ME	WB	# Note: Intentionally incomplete.			
beq r1, r4, TARG		IF	ID						
nop			IF						

- Design control logic to generate the stalls for a beq. Show connections to the input of the OR gate on the lower right.
- Make sure that the logic handles the cases above and for similar situations.
- Use as many or as few comparison units, [=], as you need.



Problem 3: [40 pts] Answer each question below.

(a) The MIPS code below loads, stores, and loads again. The two sets of tables further below show the contents of memory before and after the code executes. Numbers in the table are hexadecimal. The code runs on a big-endian system.

```
# Initially r2 = 0x1200
LOOP:
lw r1, 0(r2)
sb r1, 1(r2)
lw r3, 0(r2)
bne r1, r3, LOOP
addi r2, r2, 4
```

Modify the *After* column so that it shows the contents of memory after the code executes.

Before		After	
Memory Address	Memory Contents	Memory Address	Memory Contents
0x1200	0xa0	0x1200	0xa0
0x1201	0xa1	0x1201	0xa1
0x1202	0xa2	0x1202	0xa2
0x1203	0xa3	0x1203	0xa3
0x1204	0xa4	0x1204	0xa4
0x1205	0xa5	0x1205	0xa5
0x1206	0xa6	0x1206	0xa6
0x1207	0xa7	0x1207	0xa7

Modify **one row** in the *Before* column below so that the code above executes just one iteration.

Before		After	
Memory Address	Memory Contents	Memory Address	Memory Contents
0x1200	0xa0	0x1200	0xa0
0x1201	0xa1	0x1201	0xa1
0x1202	0xa2	0x1202	0xa2
0x1203	0xa3	0x1203	0xa3
0x1204	0xa4	0x1204	0xa4
0x1205	0xa5	0x1205	0xa5
0x1206	0xa6	0x1206	0xa6
0x1207	0xa7	0x1207	0xa7

(b) Show the encoding of each MIPS instruction below. (That is, show the layout of the 32 bits in the instruction.) Fill fields with numeric values whenever possible, such as for register numbers and immediate values. For unknown opcodes and func field values show some kind of name.

Show encoding of: `lw r1, 2(r3)`.



Show encoding of: `xor r4, r5, r6`.



Show encoding of: `addi r7, r8, 9`.



(c) Arm A32 is a 32-bit ISA, Arm A64 (Aarch64) is a 64-bit ISA.

What does the n in n -bit ISA refer to?

Name an application or kind of device for which a 32-bit ISA has an advantage, and describe the advantage.

Name an application or kind of device for which a 64-bit ISA is a requirement or a big advantage, and describe the requirement/advantage.

(d) In the statement below the description of how ISAs and implementations are developed is different than how they are typically developed in accepted practice.

By finalizing an ISA after its implementation is complete it is assured that the ISA exactly describes the implementation and that the implementation makes the best use of the technology at hand.

How is this statement of ISA and implementation development different than accepted practice? What is the disadvantage of the approach described in the statement (ignoring the “technology at hand” part)?

The phrase “makes the best use of the technology at hand” is correct. Explain why accepted practice of ISA and implementation development may not make the best use of technology. *Hint: think about the number of bits in a register.*

(e) Answer the following about CISC ISAs.

What feature of CISC ISAs allow them to have large, say 32-bit, immediate values?

Why can't a RISC ISA like MIPS practically have 32-bit immediates?