

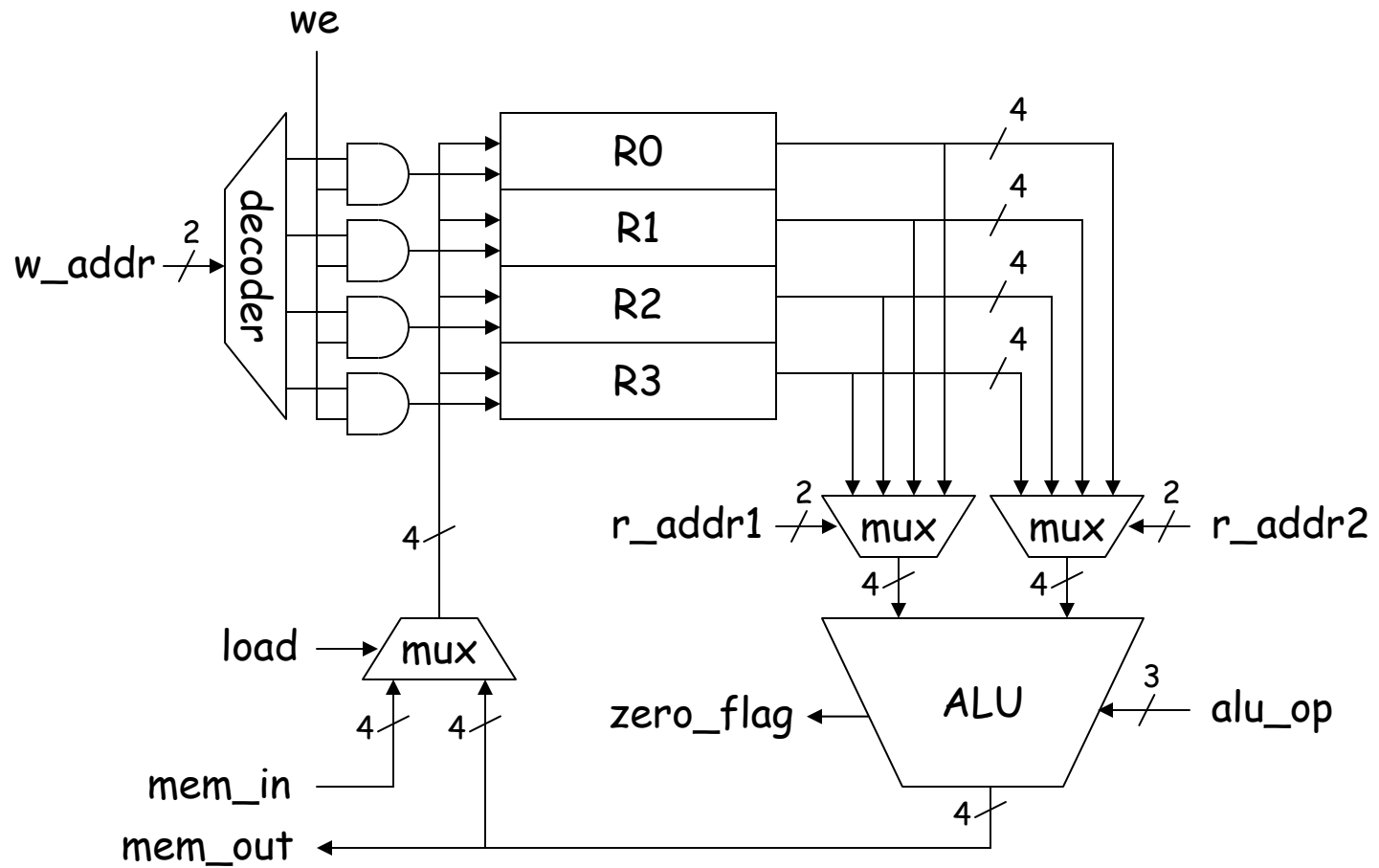
Lab 4

Due date: Friday, December 5th

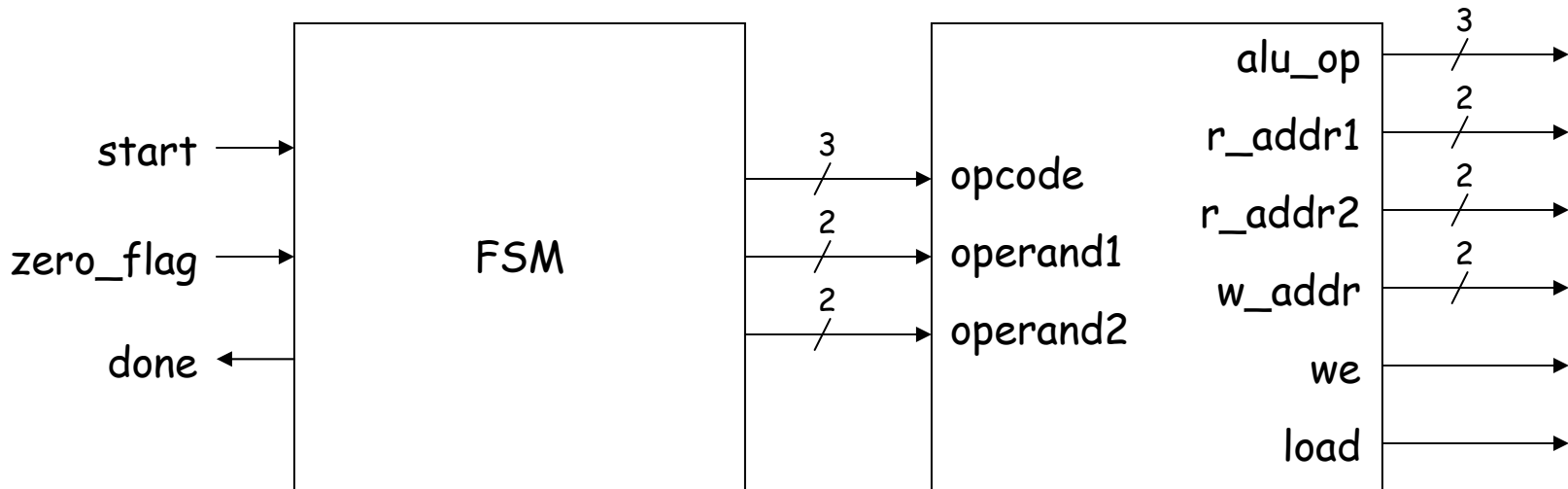
CSE 140L

Fall 2003

Datapath



FSM & Nano-decoder



- Instruction format: $\langle \text{opcode}, \text{operand1}, \text{operand2} \rangle$

- Instructions:

000 -- -- noop

001 aa -- set $R[aa] = 1$

010 aa -- increment $R[aa] = R[aa] + 1$

011 aa -- decrement $R[aa] = R[aa] - 1$

100 aa -- load $R[aa] = \text{mem_in}$

101 aa -- store $\text{mem_out} = R[aa]$

110 aa bb add $R[aa] = R[aa] + R[bb]$

111 aa bb copy $R[aa] = R[bb]$

Nano-decoder

opcode	opcode	opr1	opr2	alu_op	raddr1	raddr2	waddr	we	load
noop	000	--	--	000	--	--	--	0	-
set	001	aa	--	001	--	--	aa	1	0
incr	010	aa	--	010	aa	--	aa	1	0
decr	011	aa	--	011	aa	--	aa	1	0
load	100	aa	--	100	--	--	aa	1	1
store	101	aa	--	101	aa	--	--	0	-
add	110	aa	bb	110	aa	bb	aa	1	0
copy	111	aa	bb	111	bb	--	aa	1	0

- Hints: can assign don't cares so that `alu_op` = `opcode`, and `raddr2` = `opr2`
- `waddr` (write address), `we` (write enable), `load`, and `raddr1` logic more complicated

More on Datapath

- Assume 4-bits represent only positive numbers 0 ... 15
- Increment can be implemented as
 - $A + 0$, carry_in = 1
- Decrement can be implemented as
 - $A + \text{"1111"}$, carry_in = 0
- zero_flag = 1 if the output of the ALU is "0000"

Fibonacci Sequence

- Fibonacci sequence

- $F(1) = 1$
- $F(2) = 1$
- $F(N) = F(N-1) + F(N-2)$
- e.g. 1, 1, 2, 3, 5, 8, 13 ...
- With a 4-bit datapath, can only count up to 15, or assume $N \leq 7$, $F(7) = 13$

Pseudo code

```
int fibonaaci (int N)
{
    int N1 = 1, N2 = 1;
    int F, temp, c;

    for (c = N-2; c > 0; c--) {
        temp = N1;
        N1 = N1 + N2;
        N2 = temp;
    }
    F = N1;

    return F;
}
```

Pseudo Machine Code

- Specification
 - Wait until start = 1
 - Assume N-2 is provided at mem_in for computing F(N)
 - Use store instruction to output the final answer F(N) to mem_out and set done = 1 (done = 0 during other cycles)
- Register allocation
 - R0: count
 - R1: N1
 - R2: N2
 - R3: temp
- Pseudo machine code

```
while (not start) {
    noop
}
load R0      // set c = N-2
set R1       // set N1 = 1
set R2       // set N2 = 1
store R0     // test zero
while (not zero_flag) {
    copy R3 R1
    add R1 R2
    copy R2 R3
    decr R0
}
store R1, done = 1
go back to initial state
```

FSM (Moore Machine)

	state	start/zero_flag			opcode	opr1	opr2	done
		00	01	1-				
A: while (not start) { noop }	A	A	A	B	000	--	--	0
B: load R0	B	C	C	B	100	R0	--	0
C: set R1	C	D	D	B	001	R1	--	0
D: set R2	D	E	E	B	001	R2	--	0
E: store R0	E	F	J	B	101	R0	--	0
while (not zero_flag) {								
F: copy R3 R1	F	G	G	B	111	R3	R1	0
G: add R1 R2	G	H	H	B	110	R1	R2	0
H: copy R2 R3	H	I	I	B	111	R2	R3	0
I: decr R0	I	F	J	B	011	R0	--	0
}								
J: store R1, done = 1 go back to initial state	J	A	A	B	101	R1	--	1

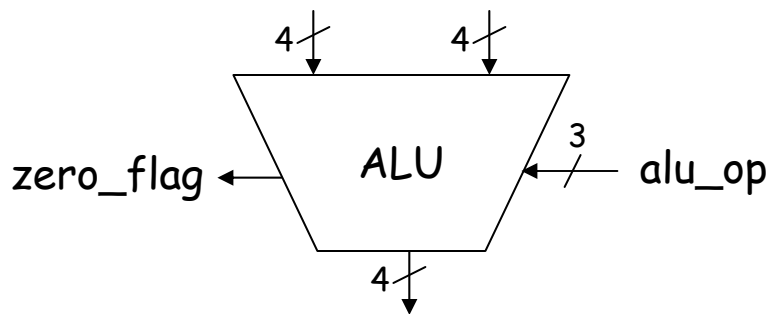
* Note: when start = 1, just start the calculation over

Grading

- Part I (12 out of 36 points)
- Part II (12 out of 36 points)
- Part III (12 out of 36 points)
- Total 36 points
- 40% of overall grade

Part I

- Part I (12 out of 36 points)
 - Test the ALU to make sure that the combinational logic works for these alu_ops (set, incr, decr, store, add, copy)



- For single operand operations, assume operand provided at operand1 (left arm of ALU)

- Sample test cases
 1. set (just set output to 1)
 2. incr 4
 3. dec 1
 4. store (just pass operand1 to output)
 5. add 2 + 4
 6. copy (just pass operand1 to output)
- 2 points for each of the 6 test cases.
- Actual values for test cases during live demo will change

Part II

- Part II (12 out of 36 points)
 - Test entire datapath and nano-decoder by running through sequences of instructions.
 - You'll be given two sequences
 - Each sequence is worth 6 points.
 - In each sequence, there will be 3 places where **mem_out** will be checked and 1 place where **zero_flag** will be checked, a total of 4 checks, each worth 1.5 points (4 x 1.5 = 6)
 - Actual test sequences will change at live demo
- Sample sequence I
 - set R1 // R1 = 1
 - set R2 // R2 = 1
 - add R1 R2 // R1 = 2
 - **store R1 // mem_out = R1**
 - add R1 R2 // R1 = 3
 - **decr R2 // R2 = 0**
// zero_flag = 1
 - **store R1 // mem_out = R1**
 - **store R2 // mem_out = R2**
- Sample sequence II
 - load R0 // R0 = 7 (mem_in)
 - load R3 // R3 = 5 (mem_in)
 - **store R3 // mem_out = R3**
 - add R3 R0 // R3 = 12
 - **decr R0 // R0 = 6**
// zero_flag = 0
 - incr R3 // R3 = 13
 - **store R1 // mem_out = R0**
 - **store R2 // mem_out = R3**

Part III

- Part III (12 out of 36 points)
 - Test to see if the fibonacci calculator works
 - $N-2$ is provided for calculating $F(N)$
 - N will be less than or equal to 7, which means $N-2$ provided at `mem_in` will be less than or equal to 5
 - Two fibonacci calculations will be tested, each worth 6 points
 - For each test, either it works or it doesn't, i.e. 0 or 6 points
- Sample case I
 - Provided with $N-2 = 5$
 - Calculate $F(N) = F(7) = 13$
- Sample case II
 - Provided with $N-2 = 1$
 - Calculate $F(N) = F(3) = 2$
- Assume
 - $N-2 \geq 0$
 - Implies $N \geq 2$