EGC442
Class Notes
2/28/2023

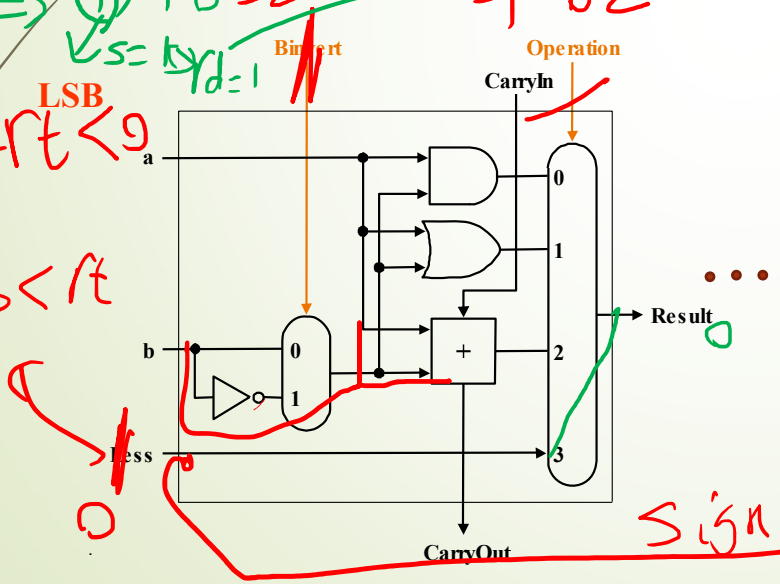
Baback Izadi
Division of Engineering Programs
bai@engr.newpaltz.edu

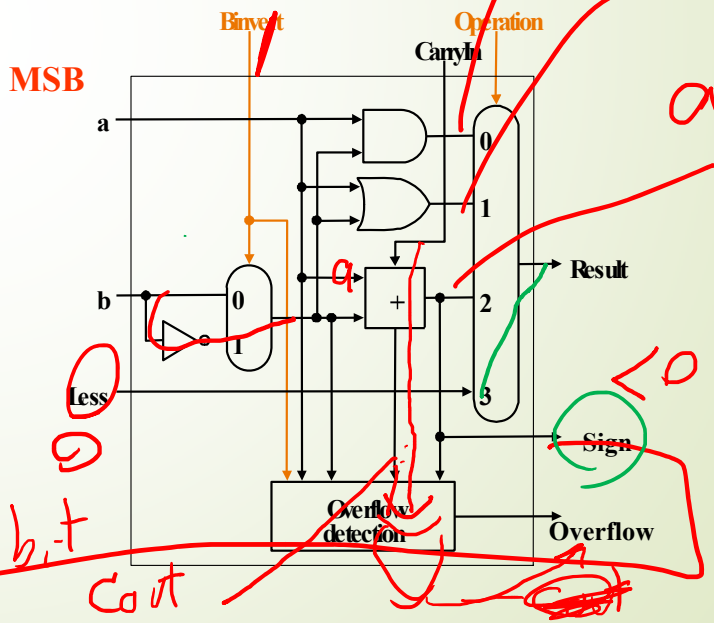
Test 1:

- Chapter 2
  - Performance problems
- Chapter 3
  - MIPS instruction set
  - C to MIPS
  - MIPS to C
  - MIPS to machine code
- Chapter 4
  - Hardware and algorithm for Multiplication
  - Floating Point
  - ALU design

Supporting *slt*

- Need to support the set-on-less-than instruction (*slt*)
  - remember: *slt* is an arithmetic instruction
  - produces a 1 if rs < rt and 0 otherwise
  - use subtraction: (a-b) < 0 implies a < b and use sign bit

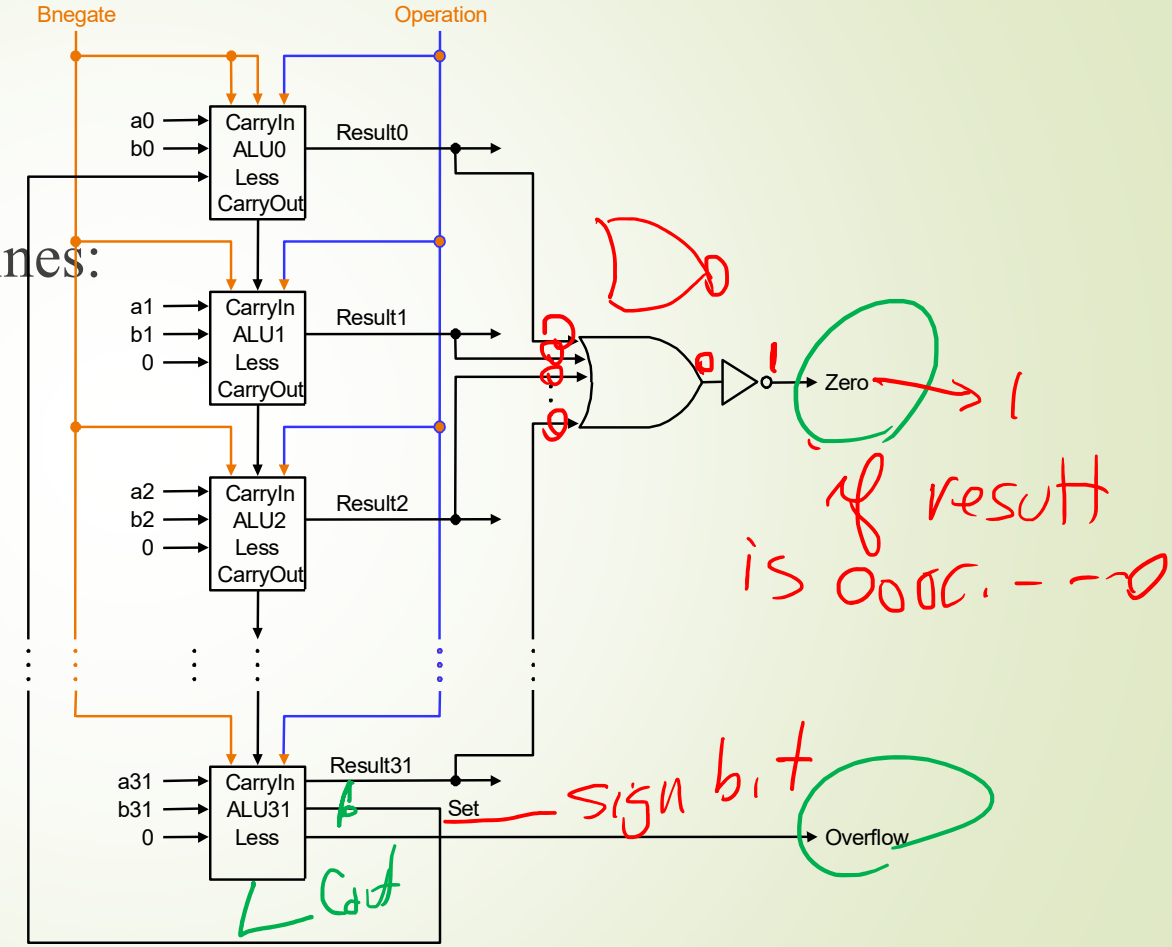




Test for equality

- Notice control lines:

000 = and
001 = or
010 = add
110 = subtract
111 = slt



5. Using Verilog design a 32 bit ALU with the following specification. Your code should include indicated flags.

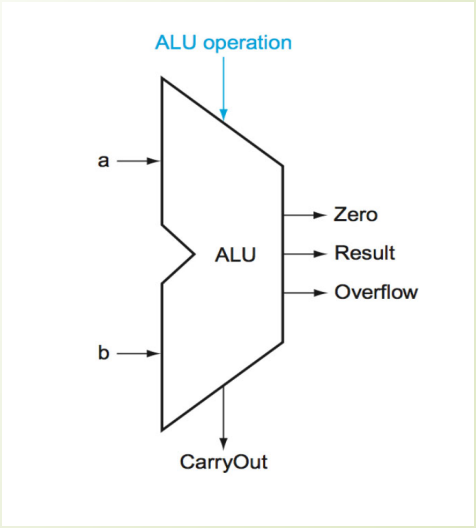
```
module alu(s, A, B, F
CY, V, Sign, Z);

input [2:0] s;

input [3:0] A, B;

wire [3:0] F, Temp;

outpuy CY, V, Sign, Z;
reg [3:0] F, Temp;

reg CY, V, Sign, Z;
always @(s or A or B)
case (s)
0: begin

F = A & B;

CY=0;

Sign = 0;

V=0;

Z= (F==0) ? 1:0;

end
1: begin

F = B | A; CY=0;

Sign = 0;
```

```
V=0;

Z= (F==0) ? 1:0;

end
2: begin

F = A ^ B;
CY=0;

Sign = 0;

V=0;

Z= (F==0) ? 1:0;

end

3: begin

F = ~(A & B);
CY=0;

Sign = 0;

V=0;

Z= (F==0) ? 1:0;

end

4: begin

 F = ~(A | B;
CY=0;
```

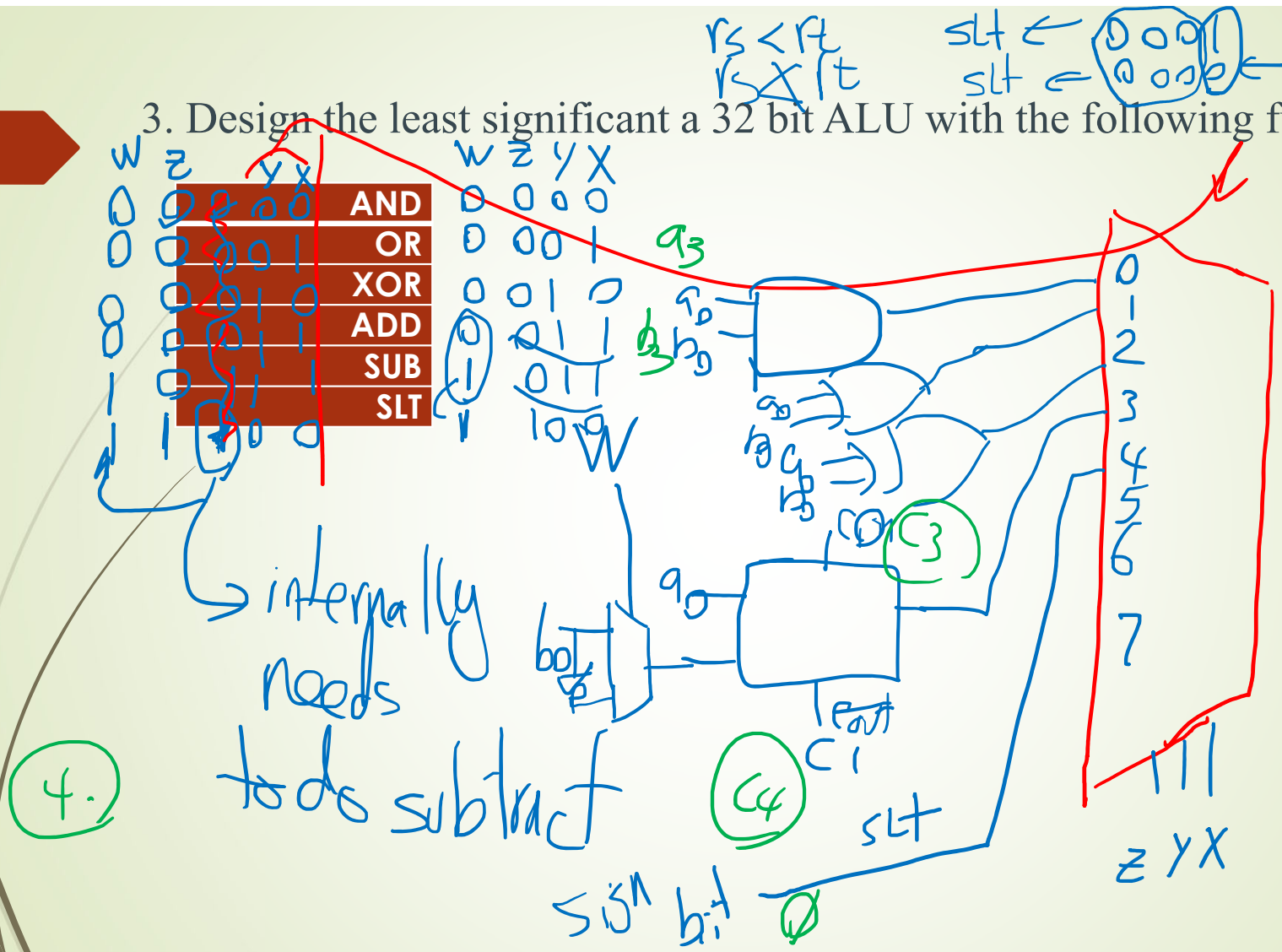
```
Sign = 0;

V=0;

Z= (F==0) ? 1:0;

end

5: begin

{CY,F} = A + B;

Sign = F[0];

{CY_T, Temp}=
A[2:0] + B[2:0];

V=CY^CY_T;

Z= (F==0) ? 1:0;

end
6: begin

{CY,F} = A - B;

Sign = F[0];

{CY_T, Temp}=
A[2:0] - B[2:0];

V=CY^CY_T;

Z= (F==0) ? 1:0;

end;
```

```
7: begin

F = (A < B)? 1:0;

CY=0;

Sign = 0;

V=0;

Z= 0;

end

endcase
endmodule
```

AND
OR
XOR
ADD
SUB
SLT



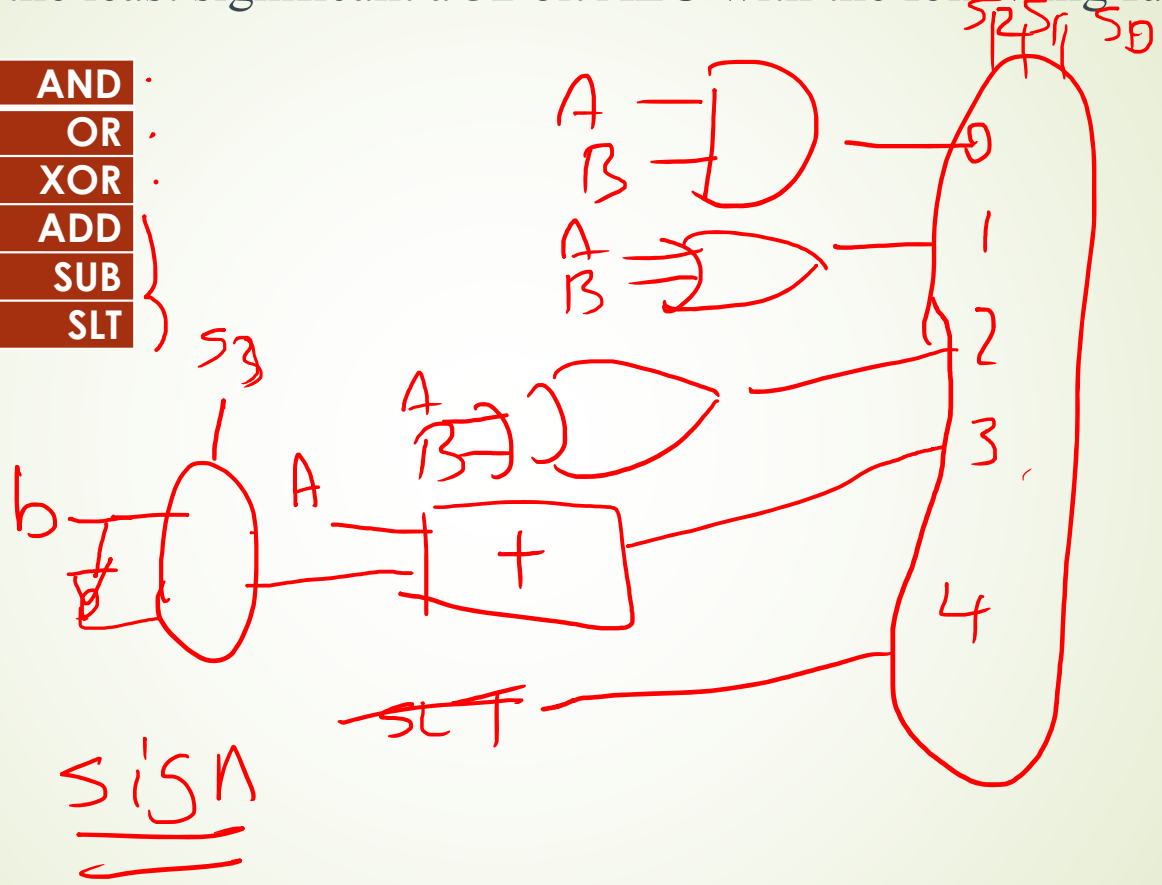
3. Design the least significant a 32 bit ALU with the following functionality.



4. Design the least significant a 32 bit ALU with the following functionality.

AND	0000
OR	0001
XOR	0010
ADD	0011
SUB	0110
SLT	0100

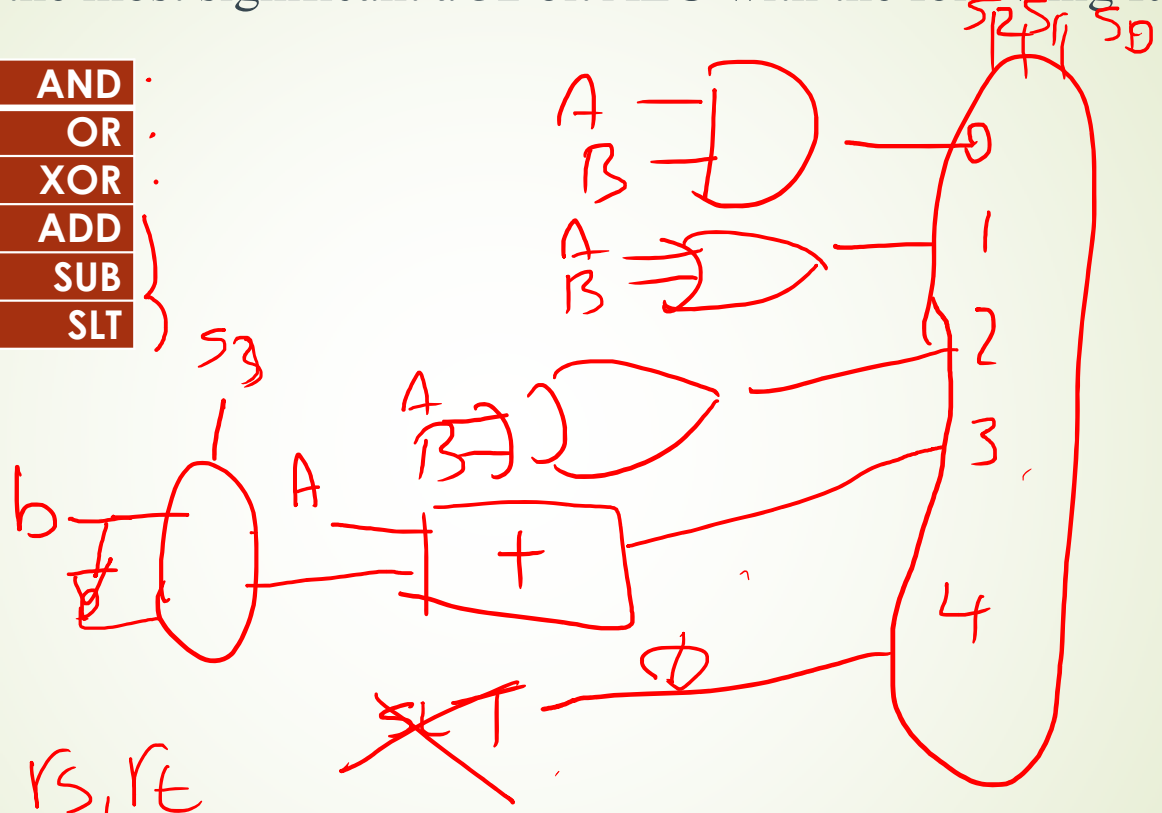
$S_3 S_2 S_1 S_0$



4. Design the most significant a 32 bit ALU with the following functionality.

AND	0000
OR	0001
XOR	0010
ADD	0011
SUB	1011
SLT	1100

$S_3 S_2 S_1 S_0$



slt rd, rs, rt

→ 000 . . . 01
000 . . . 00

$1.010 \times 2^{-3} + 0.011 \times 2^{-3} = ?$

1.010 + 0.011

1.101 $\times 2^{-3}$

c. $1.000 \times 2^{3} + 0.011 \times 2^{5} = ?$

1.1 $\times 2^{3}$

1.1

0.100 $\times 2^{3}$

1.0100 $\times 2^{4}$

Multiply -14ten and -0.25ten, or $-1.110 \times 2^{3} \times -1.000 \times 2^{-2}$. Assume 4 bits of precision.

- 1110 -.01

1. 3+127 +(−2)+127 −127 = 128

2. 1.110 x 1 = 1.110

3. 1.110

4. 1.110 x2

Floating-Point Multiplication

Multiplication example: $(1.110 \times 10^{10}) \times (9.200 \times 10^{-5})$

- Step 1: Add exponents

  $(10 + 127) + (-5 + 127) - 127 = (5 + 127)$

- Step 2: Multiply significand $1.110 \times 9.200 = 10.212000$
- Step 3: Normalize $10.212000 \times 10^{5} = 1.021 \times 10^{6}$
- Step 4: Sign of product $+1.021 \times 10^{6}$

2.7 [5] <COD §2.2, 2.3> Translate the following C code to MIPS. Assume that the variables, f,g. h, i, and j are assigned to registers $s0, $s1, $s2, $s3, $s4, and $s4, respectively. Assume that the base address of A and B are in registers $s6 and $s7, respectively. Assume that the elements of the arrays A and B are 8-byte words:

B[8] = A[i] + A[j];

2.7

```
sll     $t0, $s3, 2     # $t0 <-- 4*i
add     $t0, $t0, $s6   # $t0 <-- Addr(A[i])
lw      $t0, 0($t0)     # $t0 <-- A[i]
sll     $t1, $s4, 2     # $t1 <-- 4*j
add     $t1, $t1, $s6   # $t1 <-- Addr(A[j])
lw      $t1, 0($t1)     # $t1 <-- A[j]
add     $t0,$t0,$t1     # t1 <-- A[i] + A[j]
sw      $t0,32($s7)     # B[8] <-- A[i] + A[j]
```