Menissa Kong
11/7/2024

Computer Architecture HW7

```
                  1   1  ← carry in
  0 0 1           0   0   1   a
- 0 1 0   =   +   1   0   1   b
                  ─────────
                  1   1   1   ← becomes signed
```

```
  0 1 0            0 1 0
- 0 0 1     =      1 1 0
                   ─────
                   0 0 1
```

Find a<b

when a < b, there is no carry out bit

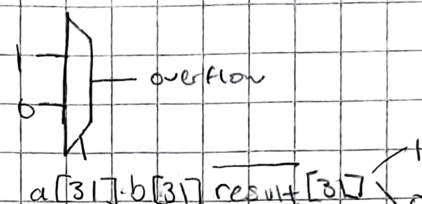
when a > b, there is a carry out bit

Therefore out = c_out (before edge cases)

ALU

Overflow: addition and subtraction

a[31] b[31] result[31]		
1+1=0	(−)+(−) = +?	overflow = 1
0+0=1	(+)+(+) = −?	overflow = 1
1+1=1	(−)+(−) = −?	overflow = 0
0+0=0	(+)+(+) = +?	overflow = 0



equal: a == b

xor: check if same

a[N-1:0] (xor) b[N-1:0] → 0... if equal

unary / bitwise

1&1&1&...&1 = 1
0&0&0&...&0 = 0
0&1&0&...&1 = 0

Zero:

result

000...000 → &(~result[N-1:0])

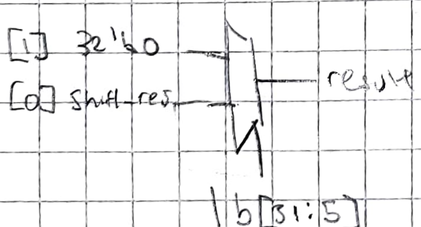
unary/bitwise operator!

1&1&1&... = 1
0&0&0... = 0
0&1&0... = 0

(results are flipped with results)

Shifters

| b[31:5] → goes down and checks if any other bits are 1



slt → − overflow

add → + overflow

else → 0