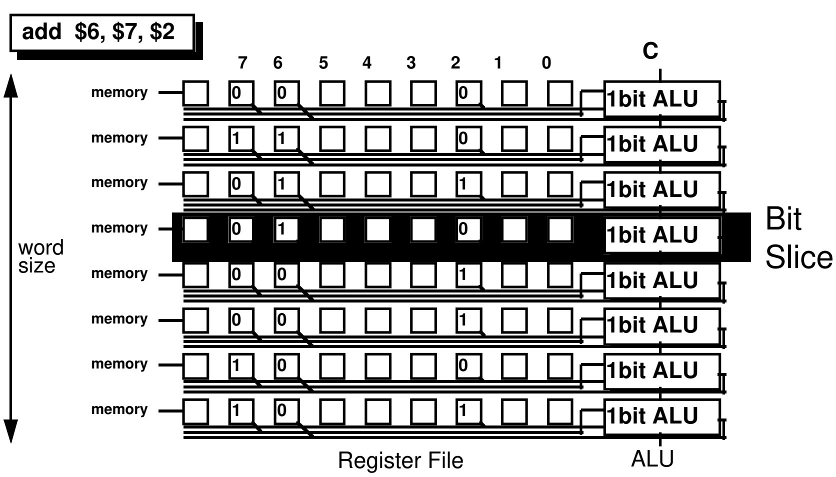
Construciton of an Arithmetic/Logic Unit

The ALU performs all of the basic operations of the machine. RISC architectures attempt to make all of the operations have similar complexity

Data Path Structure



Logic Gates*

AND(a,b): return c = a · b

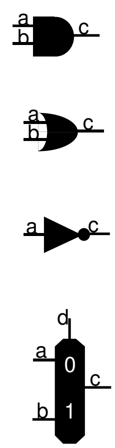
OR(a,b): return c = a + b

NOT(a): return c = ~a

MUX(d,a,b): return c =

(if d=0 then

a else b)



a	b	c
0	0	0
0	1	0
1	0	0
1	1	1

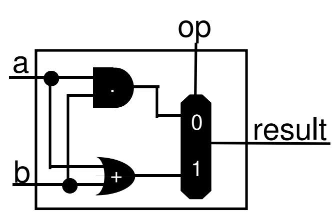
a	b	c
0	0	0
0	1	1
1	0	1
1	1	1

a	c
0	1
1	0

d	c
0	a
1	b

*Bill's nickname in college?

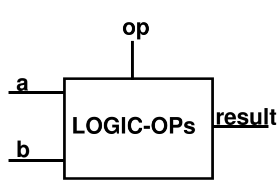
1Bit ALU: Logic Operations



Bitwise Logic Operations

AND $result,$a,$b

OR $result,$a,$b

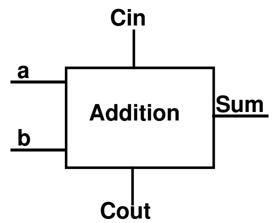


LOGIC-OPs: *MUX*(op, *AND*(a,b), *OR*(a,b))

1Bit Adder, CarryOut

3 inputs (a, b, Cin), 2 outputs (Sum, Cout)

a	b	cin	cout	sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

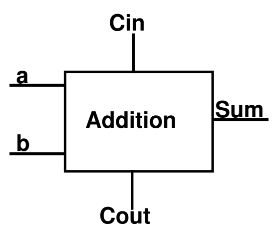


$$Cout = (b \cdot Cin) + (a \cdot Cin) + (a \cdot b) + (a \cdot b \cdot Cin)$$

Cout: *OR*(*AND*(b,Cin), *AND*(a, Cin), *AND*(a,b))

1Bit Adder, Sum

a	b	cin	cout	sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



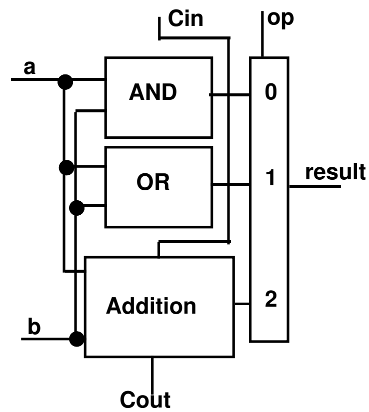
$$Sum = (a \cdot \overline{b} \cdot \overline{Cin}) + (\overline{a} \cdot b \cdot \overline{Cin}) \\ + (\overline{a} \cdot \overline{b} \cdot Cin) + (a \cdot b \cdot Cin)$$

Sum: *OR*(*AND*(a, *NOT*(b), NOT(Cin)), *AND*(*NOT*(a), b, NOT(Cin)), *AND*(*NOT*(a), *NOT*(b), Cin), *AND*(a,b,Cin))

1-Bit And/Or/Add ALU

Combine components

Set op=1 for and $result, $a,$b
Set op=2 for or $result, $a,$b
Set op=3 for add $result, $a,$b



32-Bit ALU

Compose 32 bit-slices