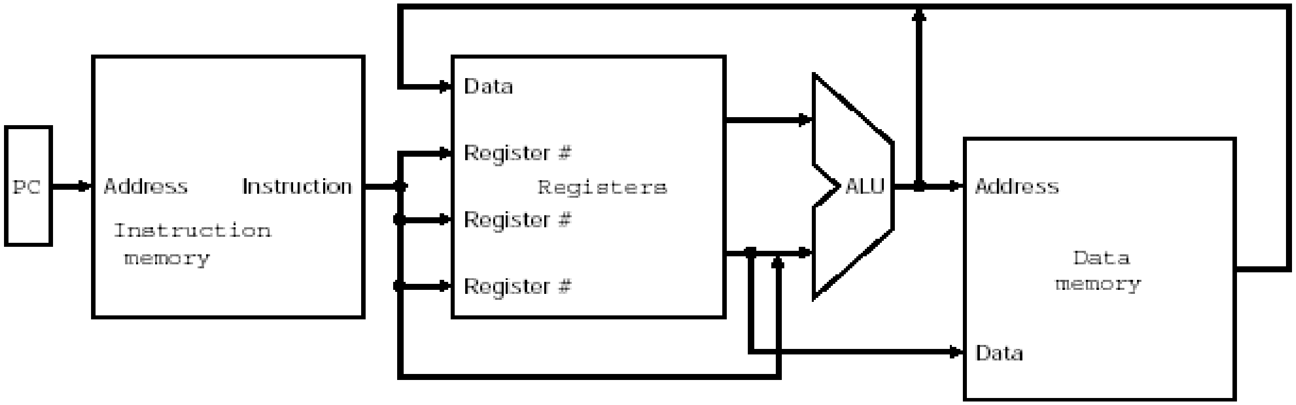
CPU



CPU = datapath + control

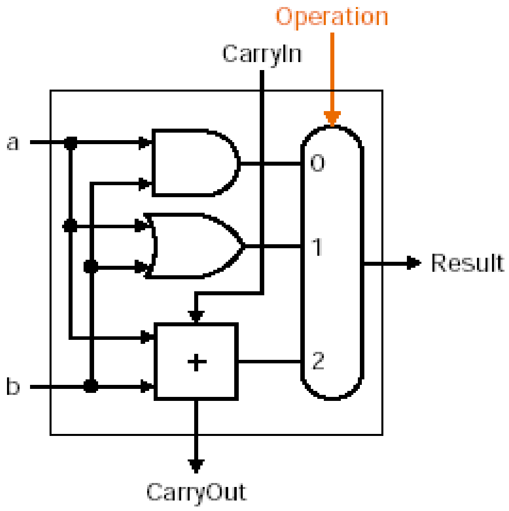
ALU: arithmetic/logic unit

performs operations to execute arithmetic and logical instructions

ALU: 1-bit

We now have the ingredients for a simple 1-bit arithmetic-logic unit (ALU)

Operations:	ADD	$a + b + c_{in}$
	AND	a AND b
	OR	a OR b
Inputs:	data:	a, b, c_{in}
	control:	con_1, con_2
Outputs:		$result, c_{out}$
Components:	AND gate	
	OR gate	
	Full adder	
	4-1 MUX	



1-bit ALU (Fig. 4.14)
(Operation is 2-bit control con_1, con_2)

Can construct k-bit ALU by combining k 1-bit ALUs
What other operations could we have?

ALU: 1-bit

How about subtraction?

We can use the adder to add the negated form of the operand

`a - b = a + (-b)`

Add an inverter to the circuit to negate b

This gives 1's complement

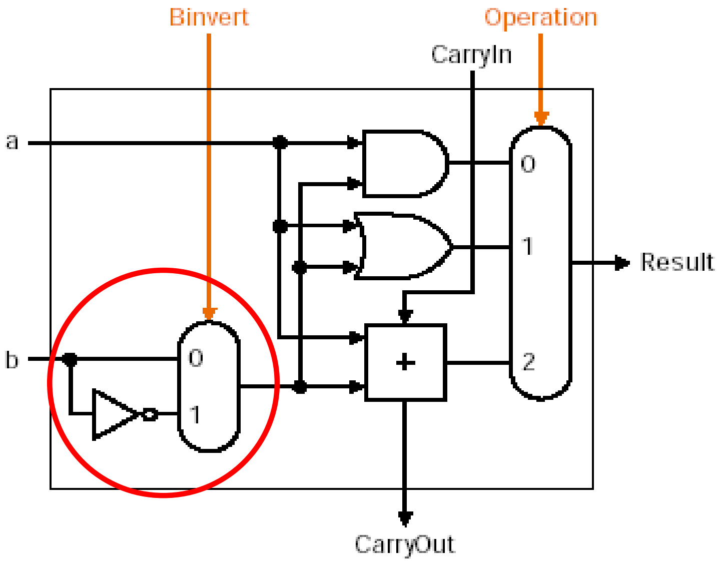
How do we get 2C value?

Use c_{in} = 1 for least significant bit

`a + ~b + 1 = a + (~b + 1) = a + (-b) = a - b`

Another MUX with control input Binvert

can select b or -b



1-bit ALU with subtraction
(Fig. 4.16)

ALU: 1-bit

This ALU can perform most of the data operations in the MIPS instruction set

Another operation, useful for branching: set on less than (`slt`)

Set the lsb to 1 if rs < rt, and 0 otherwise

If (a-b) is negative, then a < b:

```
(a - b) < 0
(a - b) + b < 0 + b
a < b
```

Result is same as sign bit from subtraction: Connect sign bit from adder to lsb of output

Unfortunately, we can only do 1 ALU operation at a time (add or `slt`)

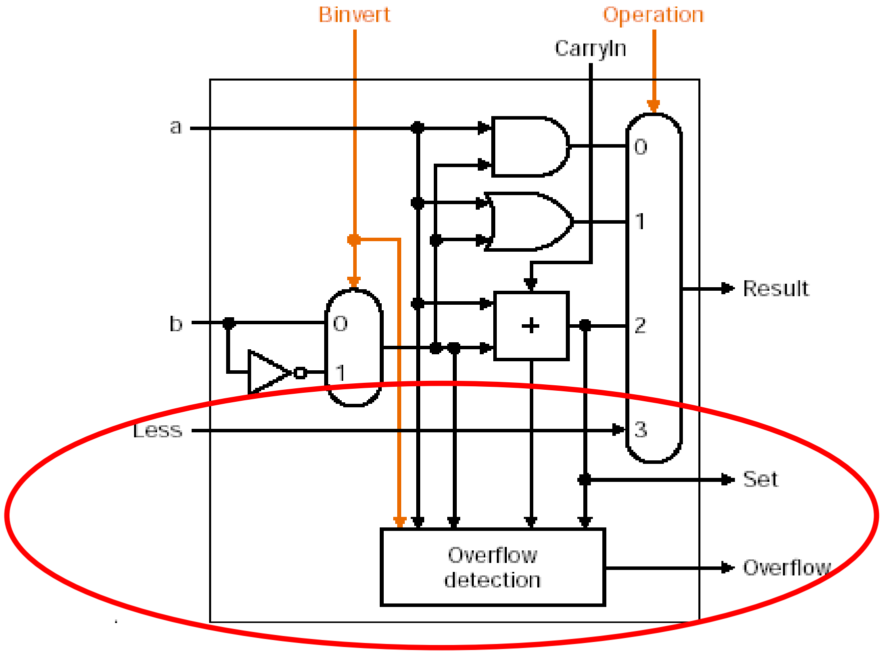
Need a new 1-bit ALU for the msb

- with an extra output from adder
- Extra output: `Set`

Additional MUX data input: `Less`

- 0 for all except lsb
- `Set` value for lsb

Also add overflow detection

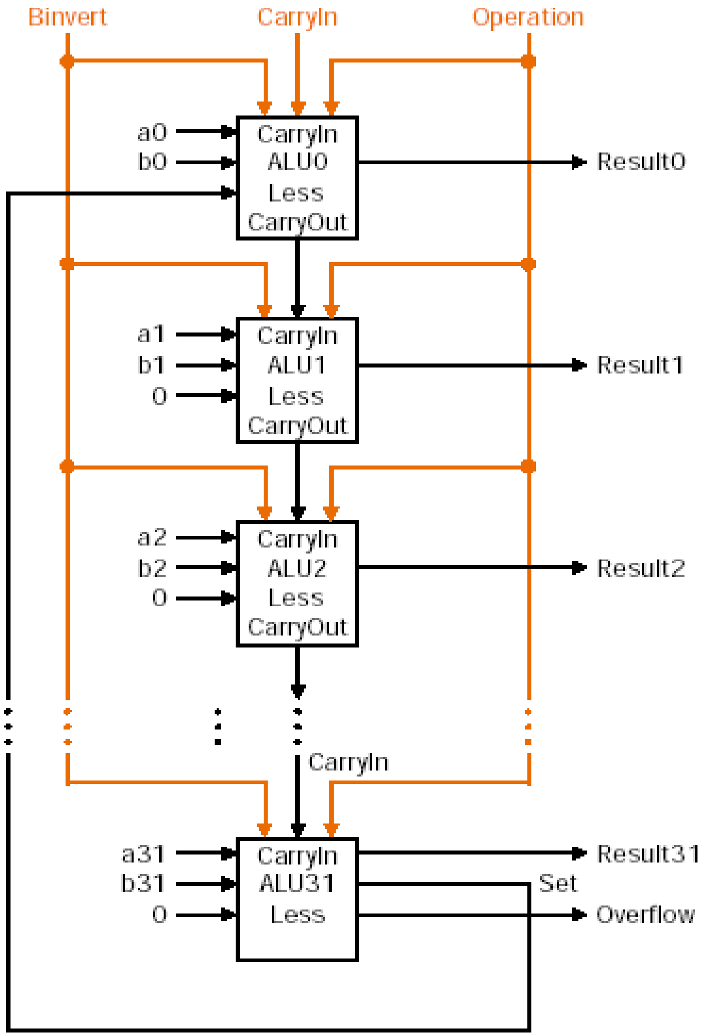


1-bit ALU with set on less than
(Fig. 4.17b)

ALU: k-bit

To operate on k-bit values,
we can connect k 1-bit ALU's

32-bit ALU is constructed using 32
1-bit ALU's
Input bits are connected in pairs
Control bits (Binvert, Operation)
are connected to each ALU
c_{out} from each ALU is connected to
c_{in} of next most significant bit ALU
(ripple carry)
c_{in} for lsb is 1 for subtract operation
Set from ALU31 (msb) is connected
to Less input of ALU0 (lsb)
(0 input for all other ALUs)
Overflow from ALU31 is additional output



32-bit ALU (Fig. 4.18)

ALU: k-bit

What about conditional branch?

- Branch if 2 values are either equal or not equal
- Easiest way to test if a == b: subtract, test result:
  - OR all result bits together and complement

One more refinement: combine Binvert and CarryIn control values into Bnegate:

- subtract: both are 1
- add or logical ops: both are 0

Bnegate (1 bit) and Operation (2 bits) are 3-bit control for MUX:

control	function
000	AND
001	OR
010	ADD
110	SUB
111	SLT

ALU: k-bit

Additions:

Bnegate control input
to subtract

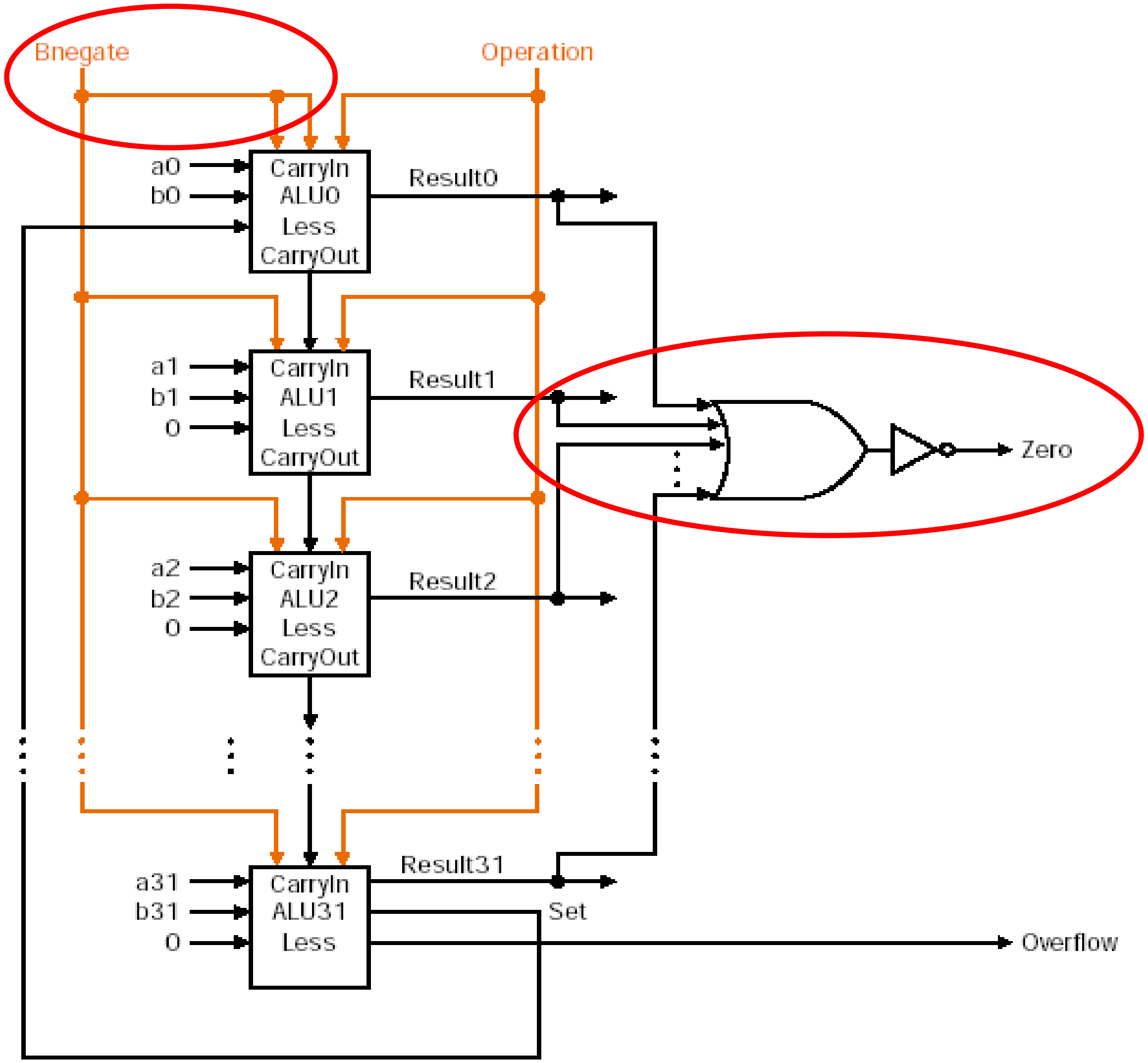
Zero output:
inverted OR of
all outputs

Input: a, b

Control:
Bnegate
Operation

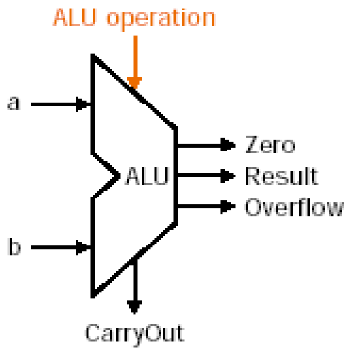
Output:
Result
Zero
Overflow

32-bit ALU with zero detection
(Fig. 4.19)



ALU: k-bit

Universal symbol to represent ALU
Can also be used for adder alone, so labeled accordingly



(Fig. 4.21)