CS465: Computer Systems Architecture

Lecture 5:
Arithmetic for Computers

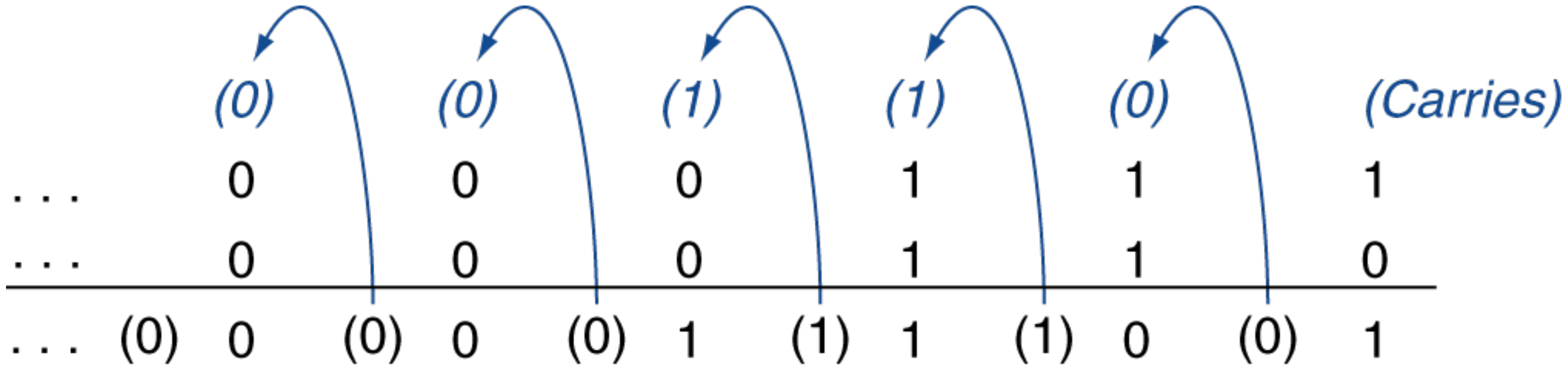
*Slides adapted from Computer Organization and Design by Patterson and Henessey

Outline

- Operations on integers
  - Addition and subtraction (check review material; Ch3.2)
  - Multiplication and division
  - Dealing with overflow
- Floating-point numbers
  - Representation (check review material; Ch 3.5)
  - Addition and multiplication

Example: Integer Addition

- Example: 7 + 6



```
            (0)    (0)    (1)    (1)    (0)    (Carries)
. . .        0      0      0      1      1      1
. . .        0      0      0      1      1      0
. . .  (0)   0 (0)  0 (0)  1 (1)  1 (1)  0 (0)  1
```

When Overflow Occurs ?

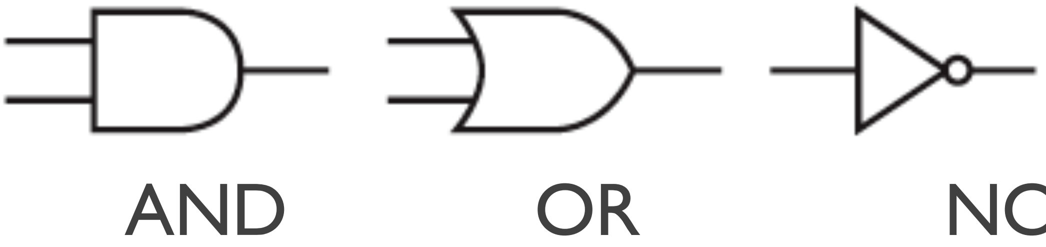
- Overflow if result out of range
- For unsigned integers:
  - Adding has a carry from MSB
- For signed integers:
  - Adding +ve and –ve operands, no overflow
  - Adding two +ve operands
    - Overflow if result sign is 1
  - Adding two –ve operands
    - Overflow if result sign is 0

Dealing with Overflow

- MIPS unsigned integer arithmetic instructions ignore overflow:
  - `addu`, `addui`, `subu`
  - Useful for C unsigned integers
- MIPS signed integer arithmetic instructions trigger an exception on overflow
  - `add`, `addi`, `sub`
  - Useful for languages that require raising an exception (e.g., Ada, Fortran)
  - On overflow, invoke exception handler
    - Save PC in exception program counter (EPC) register
    - Jump to predefined handler address
    - `mfc0` (move from coprocessor 0 reg) instruction can retrieve EPC value, to return after corrective action

Review: Boolean Algebra & Gates

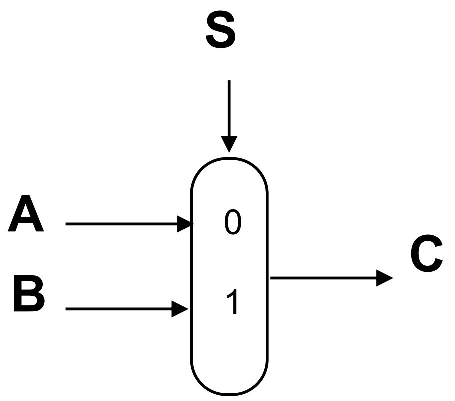
- Basic operations
  - AND, OR, NOT
- Complicated operations
  - XOR, NOR, NAND
- Logic gates



- See details in Appendix B of textbook

Review: Multiplexor

- Selects one of the inputs to be the output, based on a control input



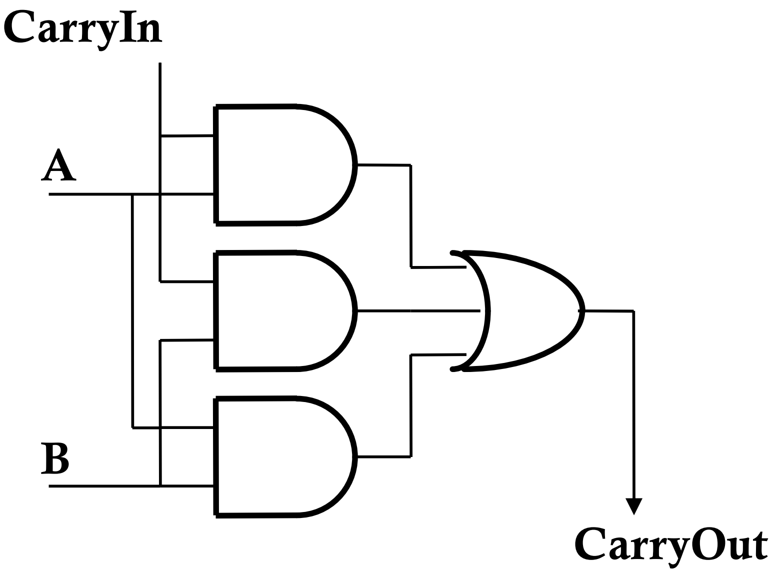
```
Note: we call this a 2-input
mux even though it has 3
inputs!
```

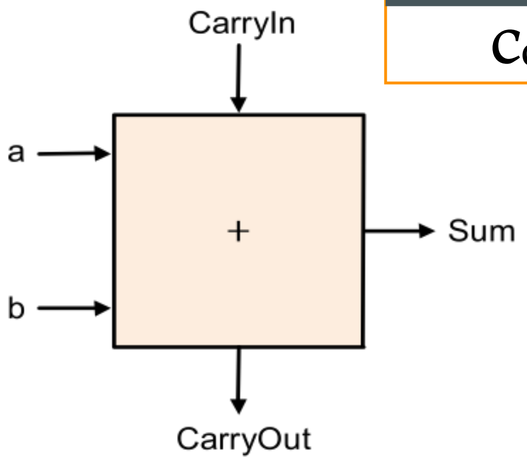
- MUX is needed for building ALU
  - Arithmetic Logic Unit: a digital circuit that performs Arithmetic (Add, Sub, . . .) and Logical (AND, OR, NOT) operations

1-bit Adder

- 1-bit addition generates two result bits
  - c_{out} = (a and b) or (a and c_{in}) or (b and c_{in})
  - sum = a xor b xor c_{in}

$$\begin{array}{rr} & a \\ + & b\ (c_{in}) \\ \hline & c_{out}\ sum \end{array}$$

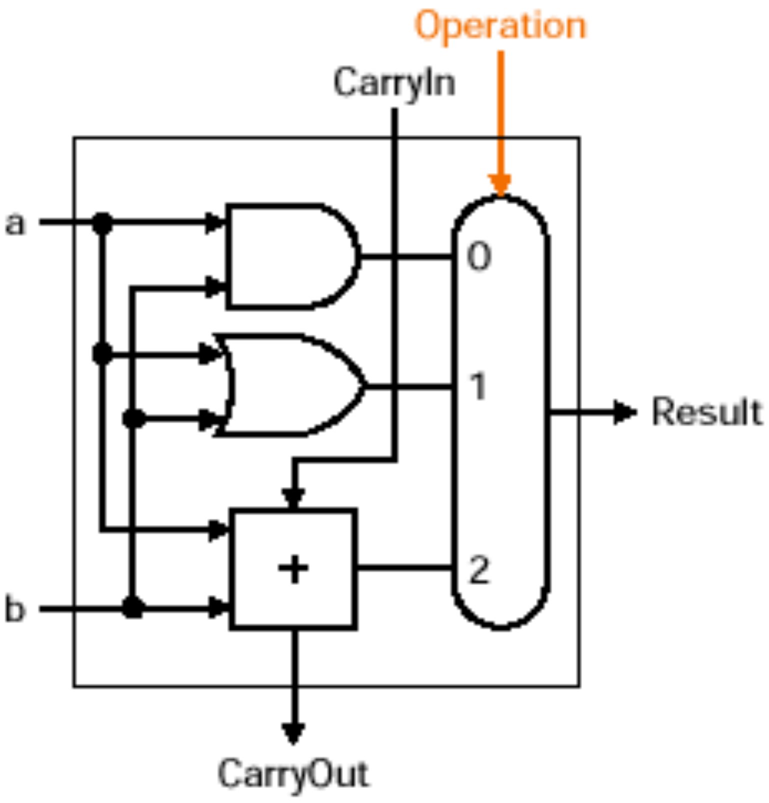




(3, 2) adder

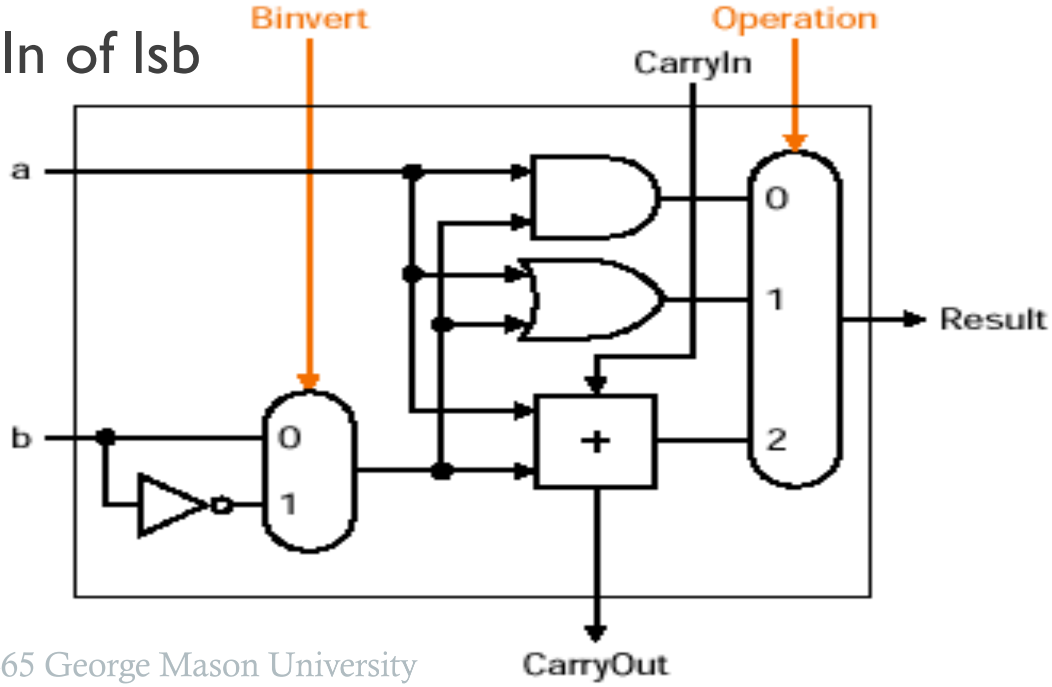
1-bit ALU

- Design trick: take pieces you know and try to put them together
  - Multiplexor
- A 1-bit ALU that performs AND, OR, and addition



What about Subtraction?

- Remember a-b = a+ (-b)
  - Two’s complement of (-b): invert each bit (by inverter) of b and add 1 at lsb
- How do we implement?
  - Bit invert: simple
  - “Add 1”: set the CarryIn of lsb

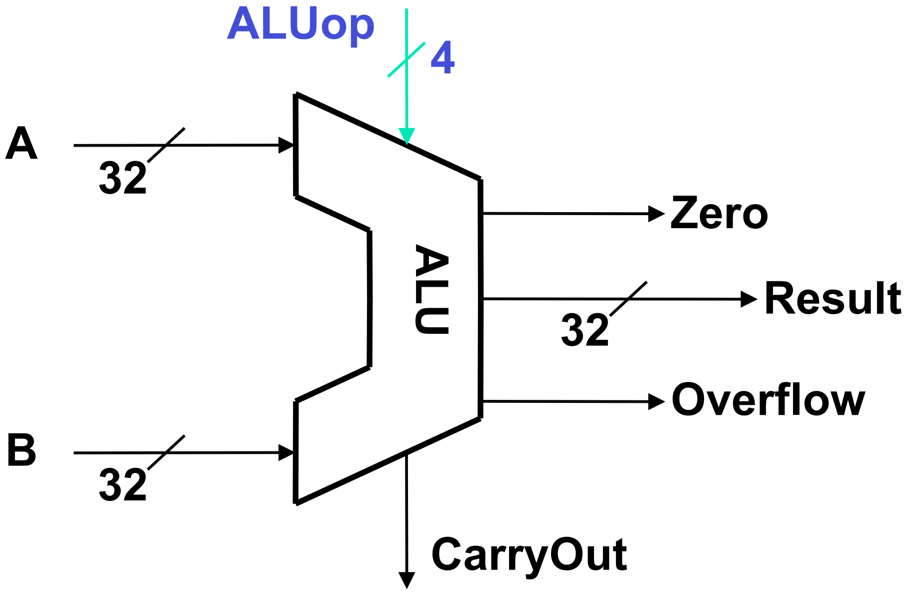


Expanding ALU

- Expanding data length
  - By paralleling the one-bit ALUs and some other modification on the logical circuits, we can create bigger ALUs
- Expanding functionality for MIPS
  - slt $t0, $s1, $s2
    - Idea: set $t0 according to the sign bit of ($s1-$s2)
  - beq $s1,$s2,label
    - Idea: or all the bits of ($s1-$s2) for zero checking

32-bit ALU for MIPS

- Operations supported: and, or, nor, add, sub, slt, beq/bne
- See Appendix B for details



Roadmap

- Operations on integers
  - Addition and subtraction
  - Multiplication
  - Division
- Floating-point numbers
  - Representation
  - Addition and multiplication