Chapter 3

Arithmetic for Computers
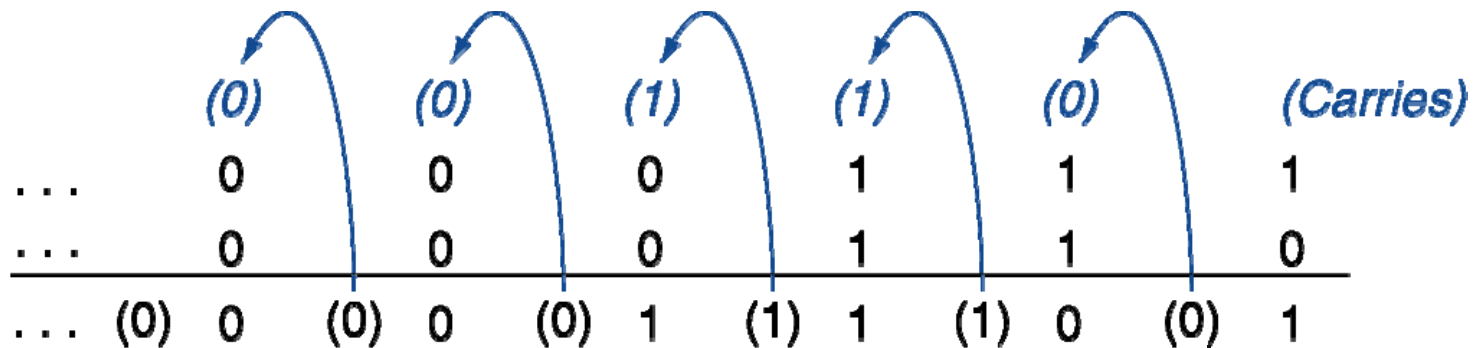
Arithmetic for Computers

- ## Operations on integers

  - ### Addition and subtraction

  - ### Multiplication and division

  - ### Dealing with overflow

- ## Floating-point real numbers

  - ### Representation and operations

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
```

- ## Overflow if result out of range

  - ### 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

Integer Subtraction

- Add negation of second operand

- Example: 7 – 6 = 7 + (–6)

  ```
  +7:     0000 0000 … 0000 0111
  –6:     1111 1111 … 1111 1010
  +1:     0000 0000 … 0000 0001
  ```

- Overflow if result out of range

  - Subtracting two +ve or two –ve operands, no overflow

  - Subtracting +ve from –ve operand
    - Overflow if result sign is 0

  - Subtracting –ve from +ve operand
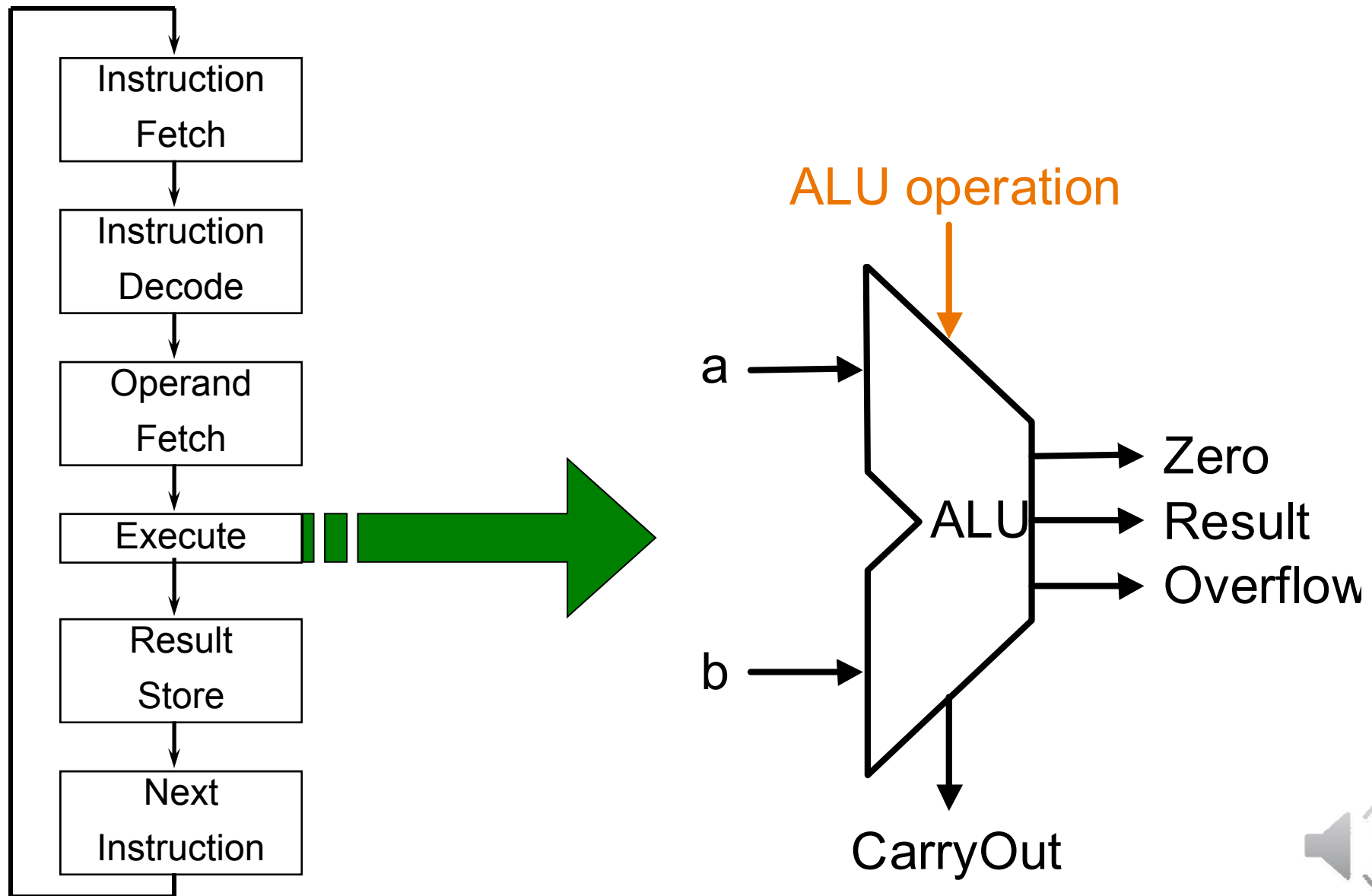   - Overflow if result sign is 1

Dealing with Overflow

- ## Some languages (e.g., C) ignore overflow
  - Use MIPS `addu`, `addui` , `subu` instructions
- ## Other languages (e.g., Ada, Fortran) require raising an exception
  - Use MIPS `add`, `addi` , `sub` instructions
  - On overflow, invoke exception handler
    - Save PC in exception program counter (EPC) register
    - Jump to predefined handler address
    - `mfc0` (move from coprocessor reg) instruction can retrieve EPC value, to return after corrective action
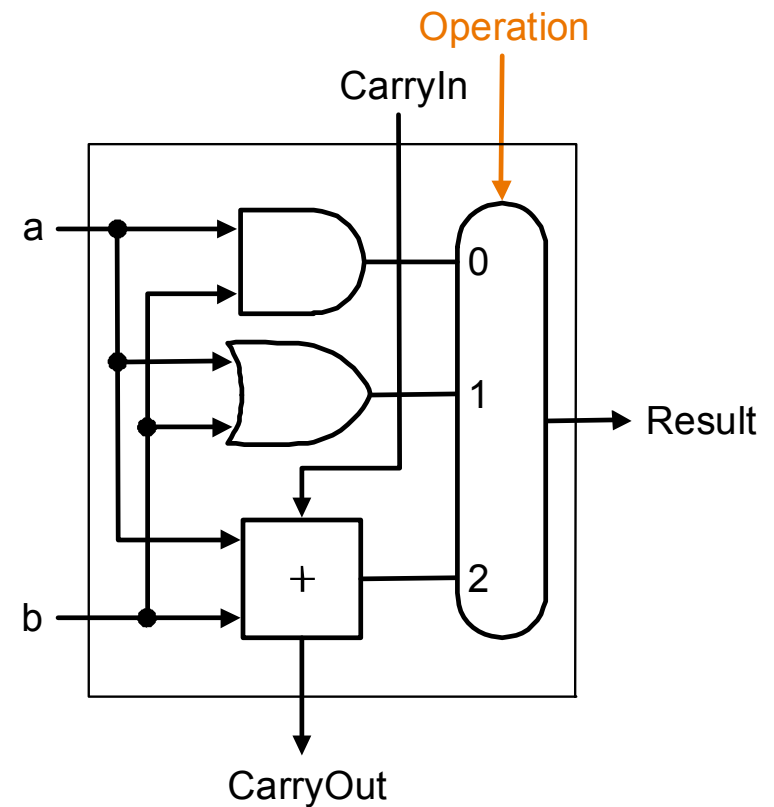
Arithmetic Logic Unit Design

Instruction Fetch

Instruction Decode

Operand Fetch

Execute

Result Store

Next Instruction

ALU operation

a

b

ALU

Zero

Result

Overflow

CarryOut

One Bit ALU

- Performs AND, OR, and ADD
  - on 1-bit operands
  - components:
    - AND gate
    - OR gate
    - 1-bit adder
    - Multiplexor

Operation

CarryIn

a

0

b

1

Result

+

2

CarryOut

Reinman 4-7

One Bit Full Adder

- ## Also known as a (3,2) adder
- ## Half Adder
  - ## no CarryIn
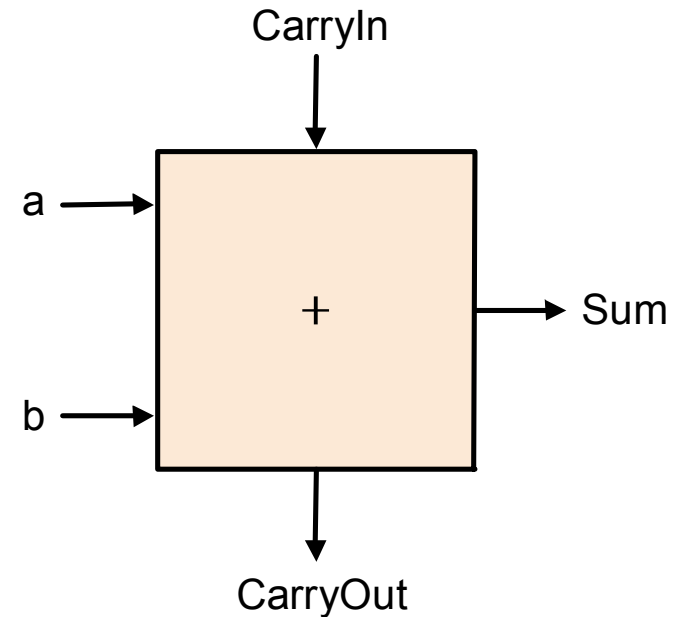
Inputs			Outputs		Comments
a	b	CarryIn	CarryOut	Sum	
0	0	0	0	0	0+0+0=00
0	0	1	0	1	0+0+1=01
0	1	0	0	1	0+1+0=01
0	1	1	1	0	0+1+1=10
1	0	0	0	1	1+0+0=01
1	0	1	1	0	1+0+1=10
1	1	0	1	0	1+0+1=10
1	1	1	1	1	1+1+1=11

CarryIn

a →

+ → Sum

b →

CarryOut

Reinman 4-8

CarryOut Logic Equation

- CarryOut = (!a & b & CarryIn) | (a & !b & CarryIn) | (a & b & !CarryIn) | (a & b & CarryIn)

- CarryOut = (b & CarryIn) | (a & CarryIn) | (a & b)

Inputs			Outputs		Comments
a	b	CarryIn	CarryOut	Sum	
0	0	0	0	0	0+0+0=00
0	0	1	0	1	0+0+1=01
0	1	0	0	1	0+1+0=01
0	1	1	1	0	0+1+1=10
1	0	0	0	1	1+0+0=01
1	0	1	1	0	1+0+1=10
1	1	0	1	0	1+0+1=10
1	1	1	1	1	1+1+1=11

Reinman 4-9

Sum Logic Equation

- Sum = (!a & !b & CarryIn) | (!a & b & !CarryIn)
  | (a & !b & !CarryIn) | (a & b & CarryIn)

Inputs			Outputs		Comments
a	b	CarryIn	CarryOut	Sum	
0	0	0	0	0	0+0+0=00
0	0	1	0	1	0+0+1=01
0	1	0	0	1	0+1+0=01
0	1	1	1	0	0+1+1=10
1	0	0	0	1	1+0+0=01
1	0	1	1	0	1+0+1=10
1	1	0	1	0	1+0+1=10
1	1	1	1	1	1+1+1=11

Reinman 4-10