# Computer Architecture

Lecture 1

## Course Information

- Course Name: Computer Architecture
- Course Code: EE2013
- Semester: Spring 2026
- Prerequisites: Computer Organization and Assembly Language (EE2003)

# Textbook and References

**Text Books:**

- David A. Patterson, John L. Hennessy, Computer Organization and Design: The hardware/software interface, RISC-V Edition.
- David A. Patterson, John L. Hennessy, Computer Architecture: A Quantitative Approach 6th edition.

**Reference books:**

- John Paul Shen and Mikko H. Lipasti Modern Processor Design: Fundamentals of Superscalar Processors.
- M. Morris Mano, Computer System Architecture 3rd Edition 1993, Prentice Hall.
- William Stallings, Computer Organization and Architecture: Designing for Performance, Prentice Hall, 11th edition.

## Course Objectives

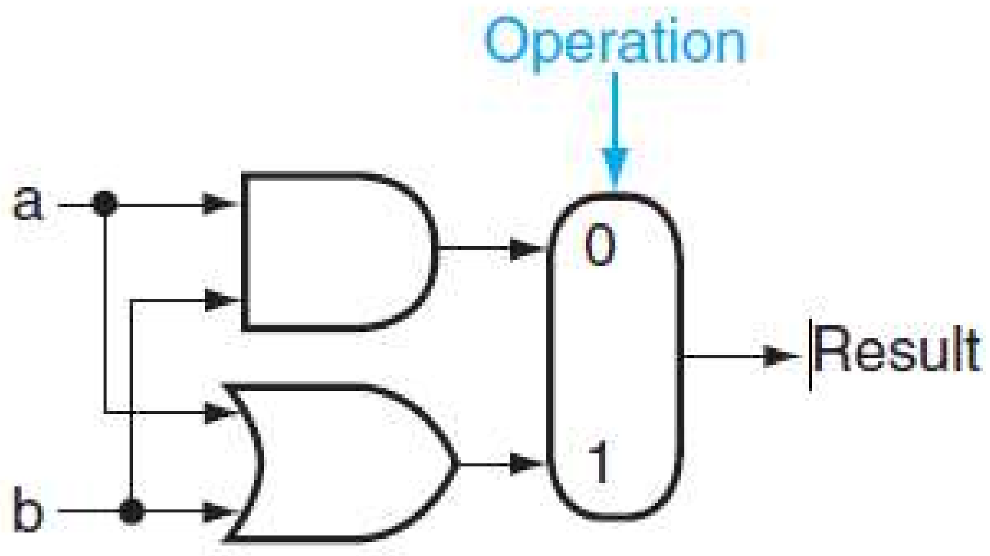
- The main objective of this course is to provide a profound understanding of the architectural design and internal working of a microprocessor, enabling computer science students to appreciate concepts like optimization and hardware level performance issues. This course also introduces advanced concepts including pipelining and superscalar architecture. Additionally, the fundamentals of GPUs are discussed.

# Grading

|  |  |
|---|---|
| • Quizzes | 15% |
| • Assignments | 10% |
| • 2 Midterm Exam | 30% |
| • Final Exam | 45% |

## Constructing a basic Arithmetic Logic Unit (Textbook 1, Appendix A.5)

## 1-bit logical unit for AND and OR.



## Review: 2's Complement Binary Representation

| | 2'sc binary | decimal |
|---|---|---|
| $-2^3$ = | 1000 | -8 |
| $-(2^3 - 1)$ = | 1001 | -7 |
| | 1010 | -6 |
| | 1011 | -5 |
| | 1100 | -4 |
| | 1101 | -3 |
| | 1110 | -2 |
| | 1111 | -1 |
| | 0000 | 0 |
| | 0001 | 1 |
| | 0010 | 2 |
| | 0011 | 3 |
| | 0100 | 4 |
| | 0101 | 5 |
| | 0110 | 6 |
| $2^3 - 1$ = | 0111 | 7 |

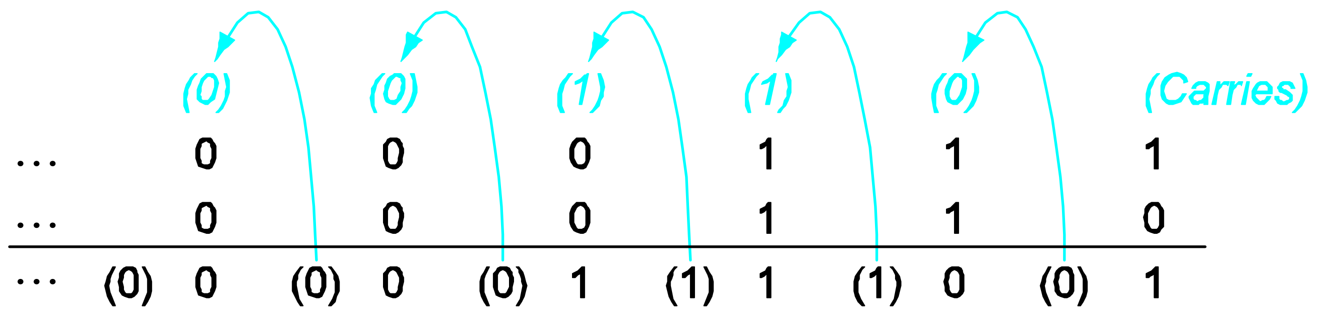
1011

and add a 1

1010

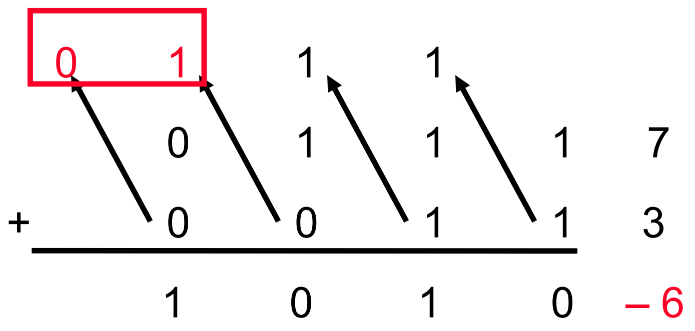
complement all the bits

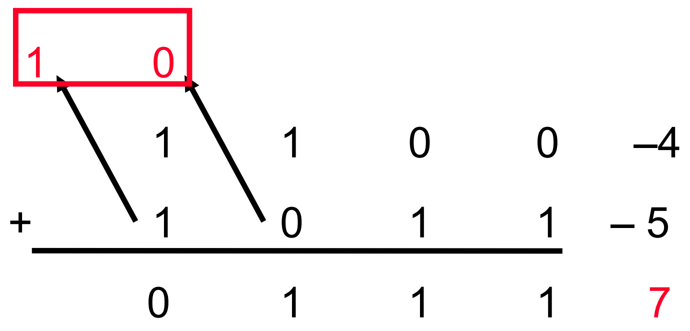
## Binary Addition



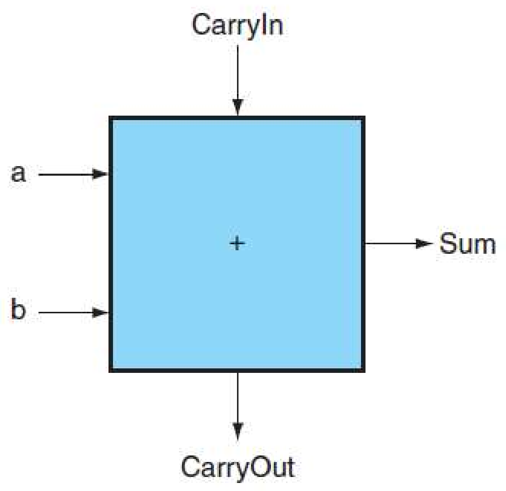
## Overflow Detection

- Overflow: the result is too large to represent in 32 bits
- Overflow occurs when
  - adding two positives yields a negative
  - or, adding two negatives gives a positive
  - or, subtract a negative from a positive gives a negative
  - or, subtract a positive from a negative gives a positive
- On your own: Prove you can detect overflow by:
  - Carry *into* MSB xor Carry *out of* MSB, ex for 4 bit signed numbers





## A 1-bit Adder



| A | B | carry_in | carry_out | S |
|---|---|---|---|---|
| 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 |

$$\text{CarryOut} = (b \cdot \text{CarryIn}) + (a \cdot \text{CarryIn}) + (a \cdot b)$$

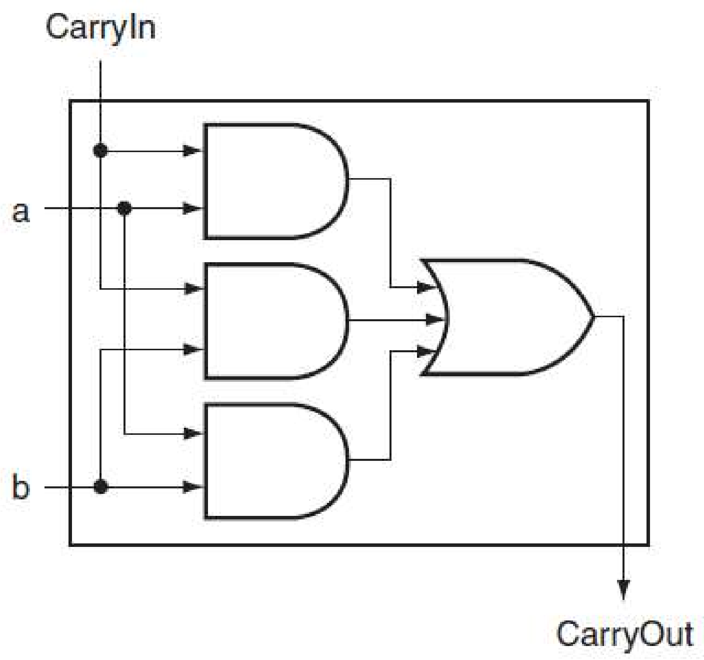
$$\text{Sum} = (a \cdot \overline{b} \cdot \overline{\text{CarryIn}}) + (\overline{a} \cdot b \cdot \overline{\text{CarryIn}}) + (\overline{a} \cdot \overline{b} \cdot \text{CarryIn}) + (a \cdot b \cdot \text{CarryIn})$$

- How can we use it to build a 64-bit adder?
- How can we modify it easily to build an adder/subtractor?

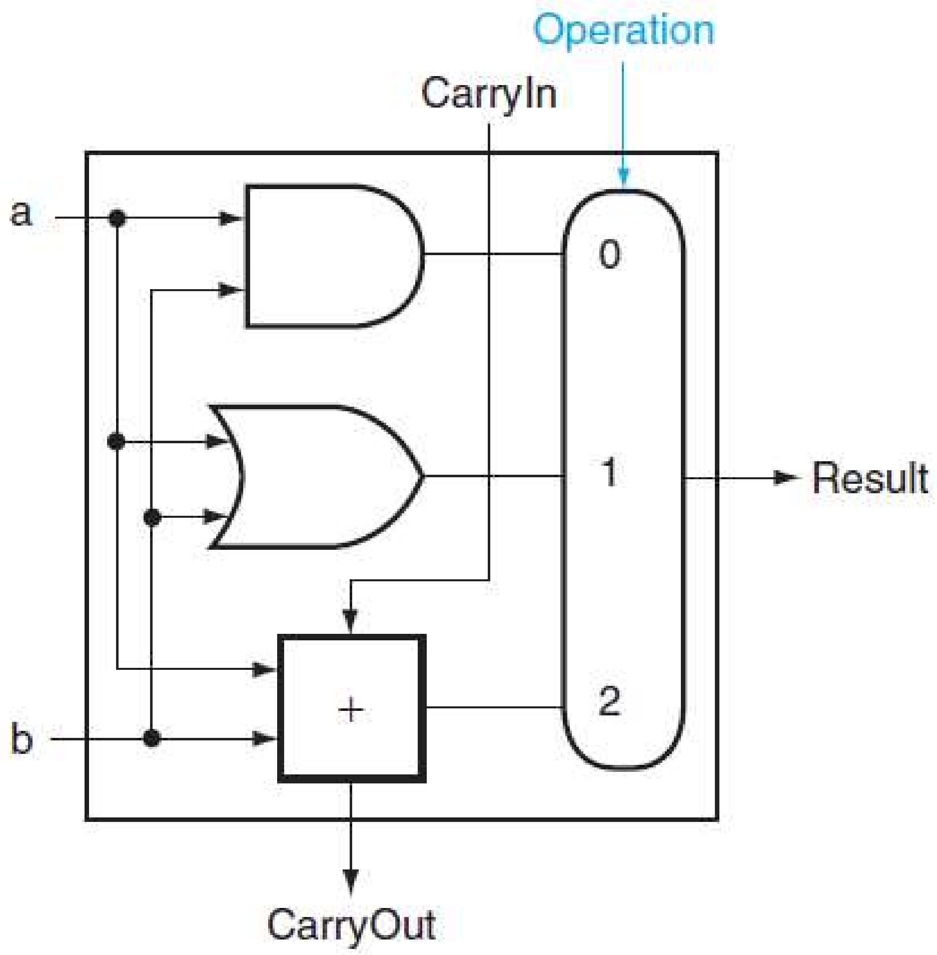
## Adder hardware for CarryOut signal

| Inputs | | |
|---|---|---|
| a | b | CarryIn |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |
| 1 | 1 | 1 |

- CarryOut = (b . CarryIn) + (a . CarryIn) + (a . b)

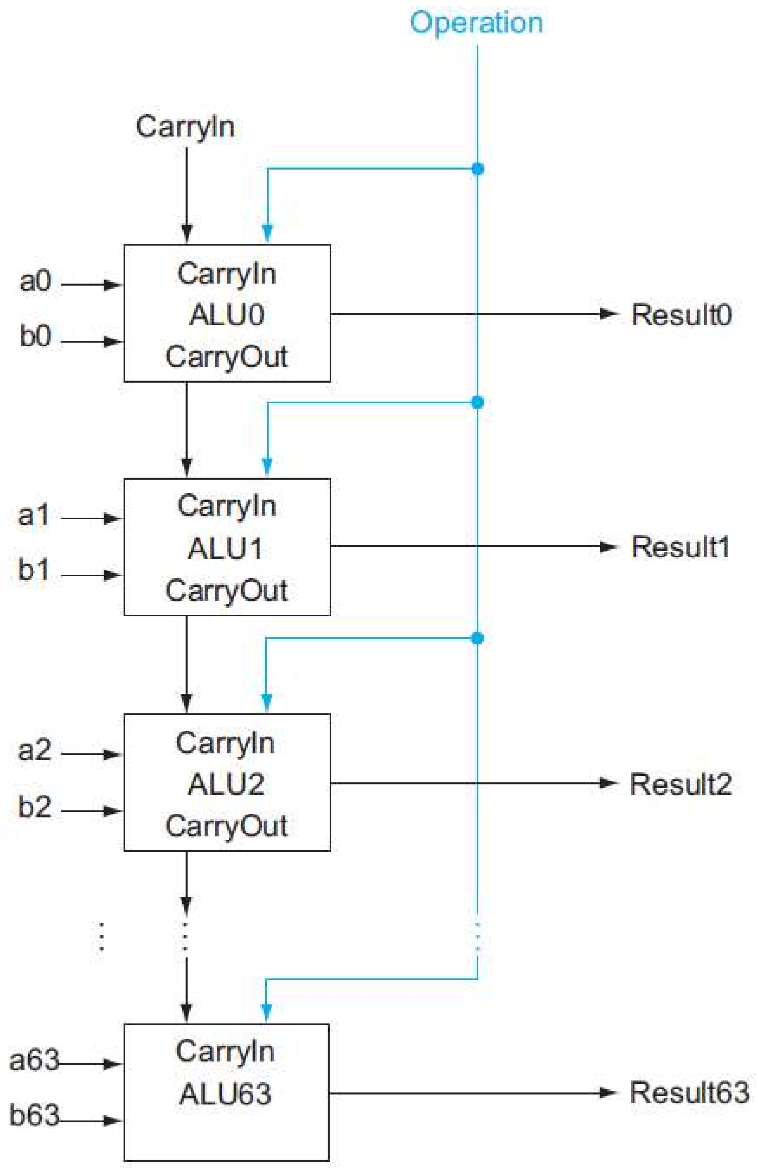


## A 1-bit ALU that performs AND, OR, and addition



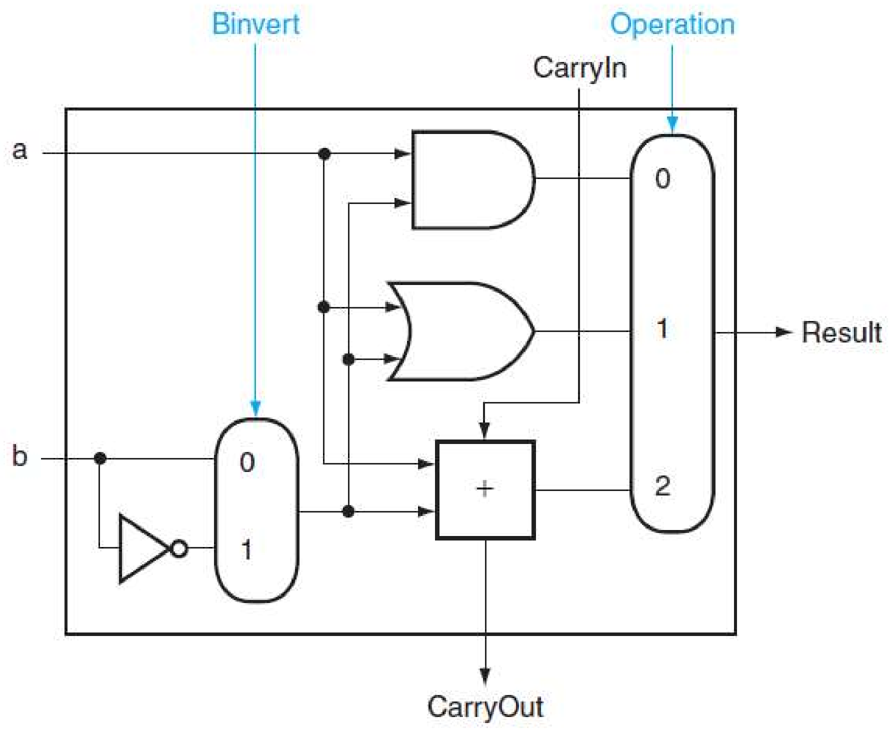
| Function | Operation |
|---|---|
| and | 00 |
| or | 01 |
| add | 10 |

## A 64-bit ALU

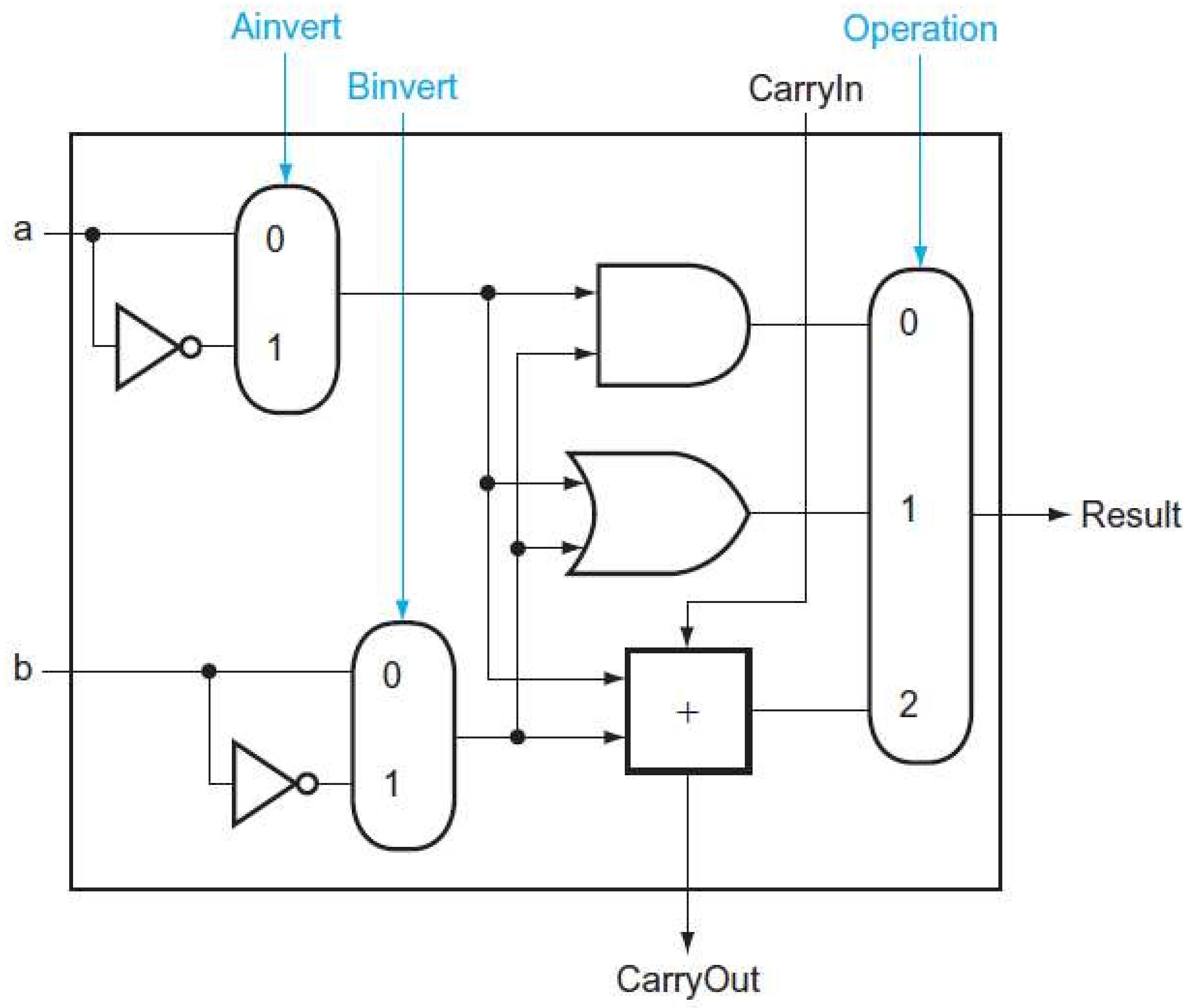

Operation
CarryIn
a0
b0
CarryIn
ALU0
CarryOut
Result0
a1
b1
CarryIn
ALU1
CarryOut
Result1
a2
b2
CarryIn
ALU2
CarryOut
Result2
a63
b63
CarryIn
ALU63
Result63

## A 1-bit ALU that performs AND, OR, and addition on a and b or a and b-invert (for subtraction)

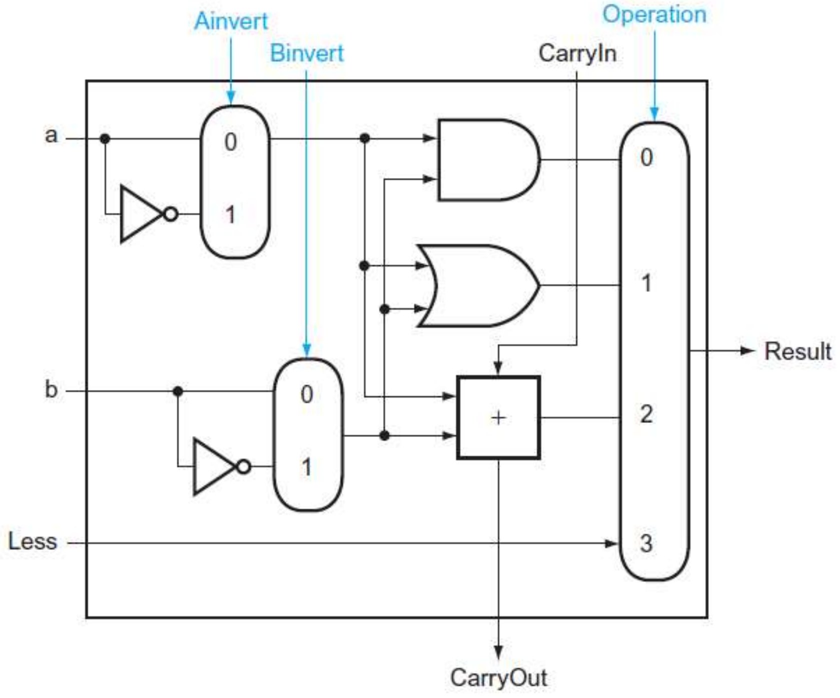
| Function | Binvert | CarryIn | Operation |
|---|---|---|---|
| and | 0 | 0 | 00 |
| or | 0 | 0 | 01 |
| add | 0 | 0 | 10 |
| sub | 1 | 1 | 10 |



## A 1-bit ALU that performs AND, OR, and addition on a and b or a-invert and b-invert.

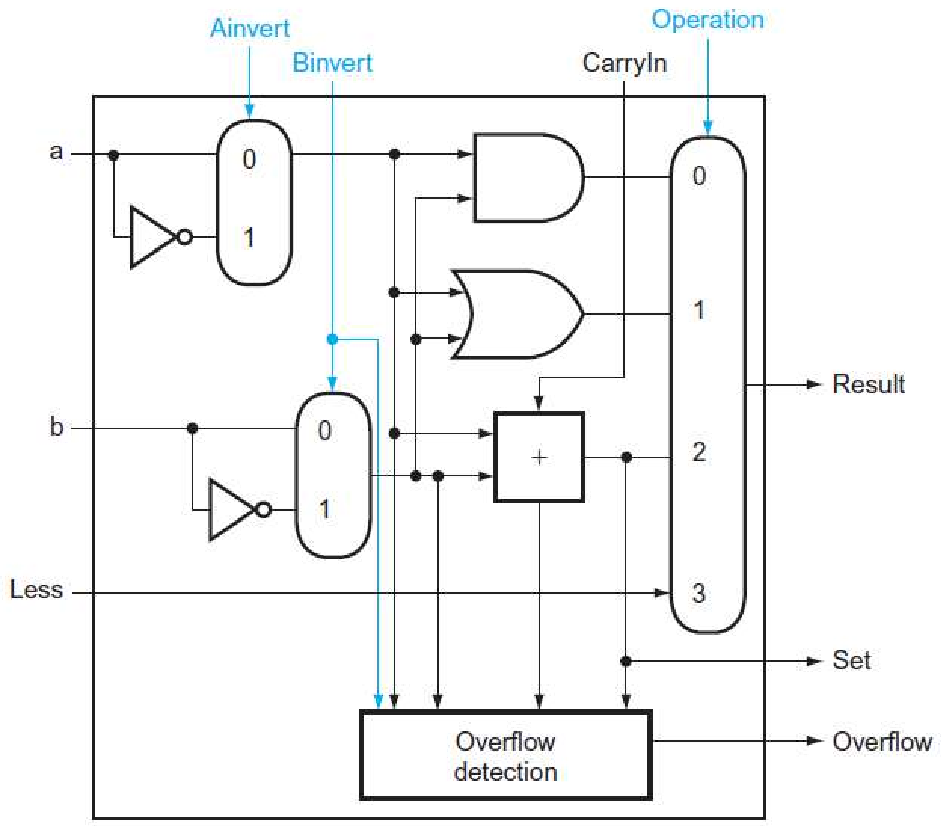


# Adding slt operation



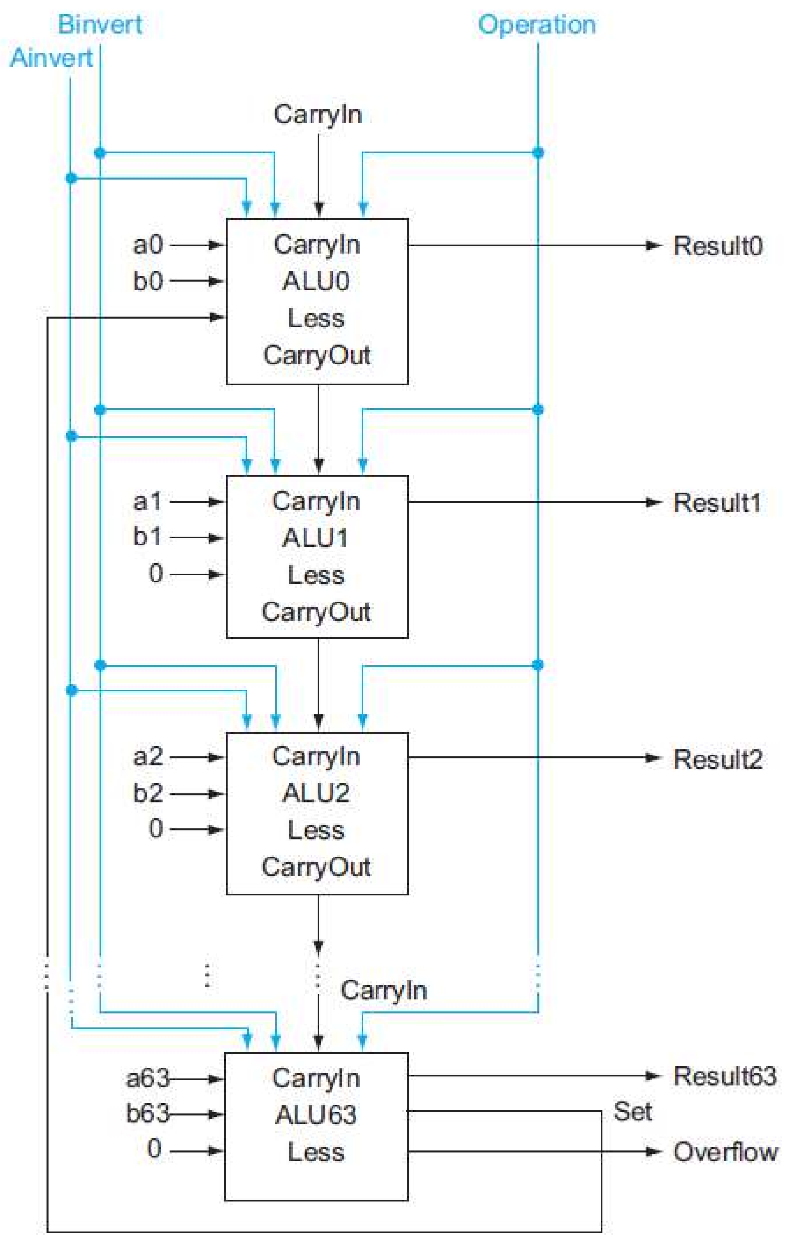
| Function | Binvert | Operation |
|---|---|---|
| and | 0 | 00 |
| or | 0 | 01 |
| add | 0 | 10 |
| sub | 1 | 10 |
| slt | 1 | 11 |

## ALU

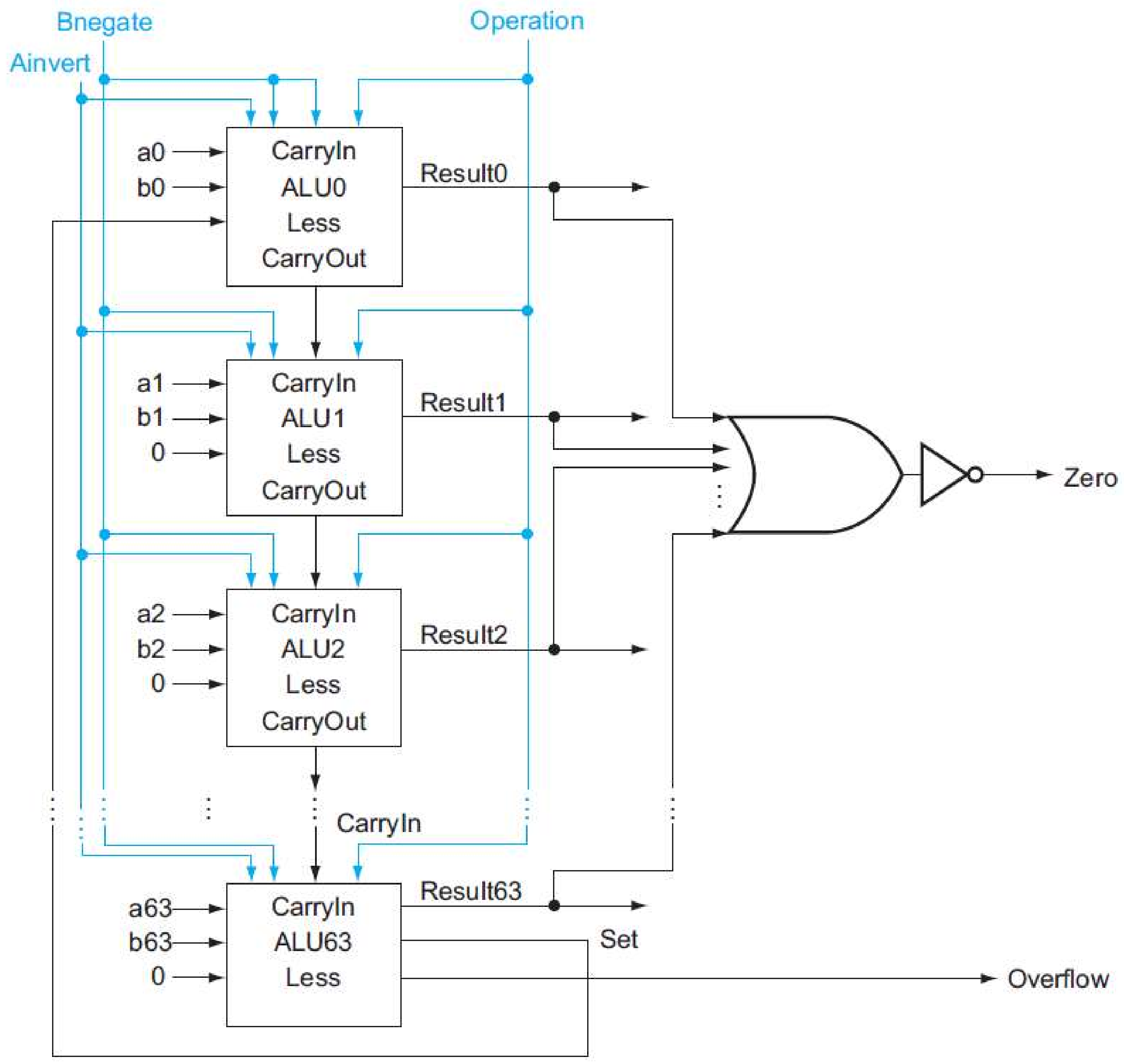


| Function | Bnegate | Operation |
|---|---|---|
| and | 0 | 00 |
| or | 0 | 01 |
| add | 0 | 10 |
| sub | 1 | 10 |
| slt | 1 | 11 |

# ALU


Binvert
Ainvert
Operation
CarryIn
a0
b0
CarryIn
ALU0
Less
CarryOut
Result0
a1
b1
0
CarryIn
ALU1
Less
CarryOut
Result1
a2
b2
0
CarryIn
ALU2
Less
CarryOut
Result2
CarryIn
a63
b63
0
CarryIn
ALU63
Less
Result63
Set
Overflow

# ALU with a zero detector



Bnegate
Operation
Ainvert
a0
b0
CarryIn
ALU0
Less
CarryOut
Result0
a1
b1
0
CarryIn
ALU1
Less
CarryOut
Result1
Zero
a2
b2
0
CarryIn
ALU2
Less
CarryOut
Result2
CarryIn
a63
b63
0
CarryIn
ALU63
Less
Result63
Set
Overflow