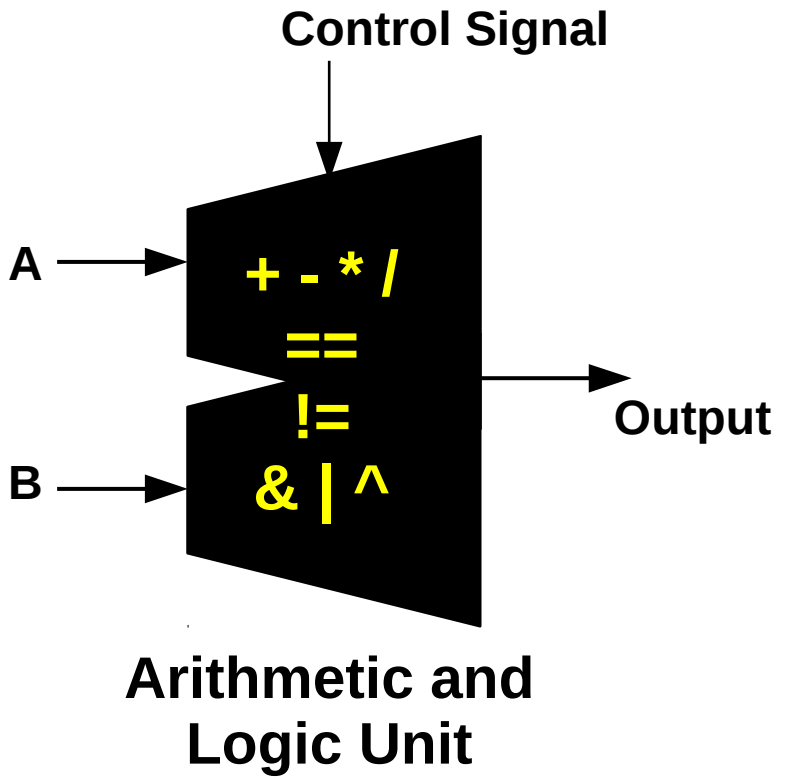
M3 – ALU Design

Module Outline

- Integer Arithmetic
  - Adder, Subtractor, Multiplier, Divider
- Arithmetic and Logical Unit Design
  - ALU Design in SystemC

Arithmetic Logic Unit

- Arithmetic operations
- Logic operations
- Comparison (Equal)



ALU

- Start with the simplest operations.

ALU

- Start with the simplest operations.
  - AND, OR
  - Control signal

ALU

- Start with the simplest operations.
  - AND, OR
  - Control signal
- Incrementally add functionality and control

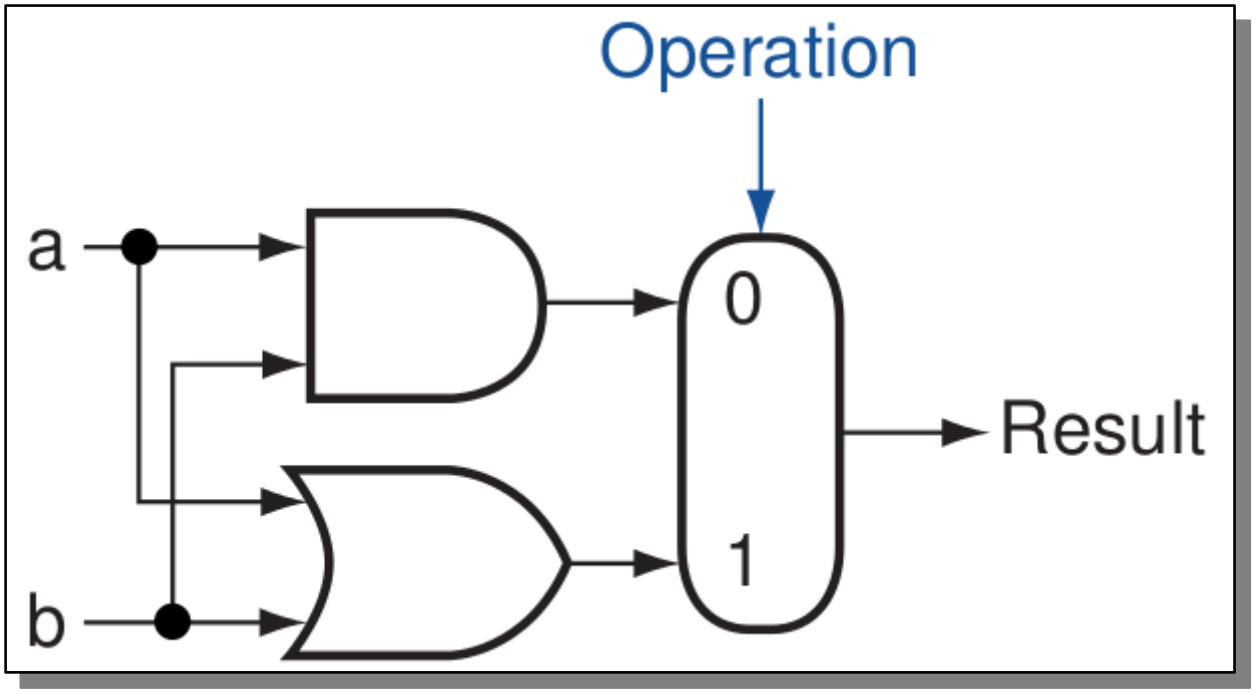
ALU

- Start with the simplest operations.
  - AND, OR
  - Control signal
- Incrementally add functionality and control
- Start with a 1-bit ALU

ALU

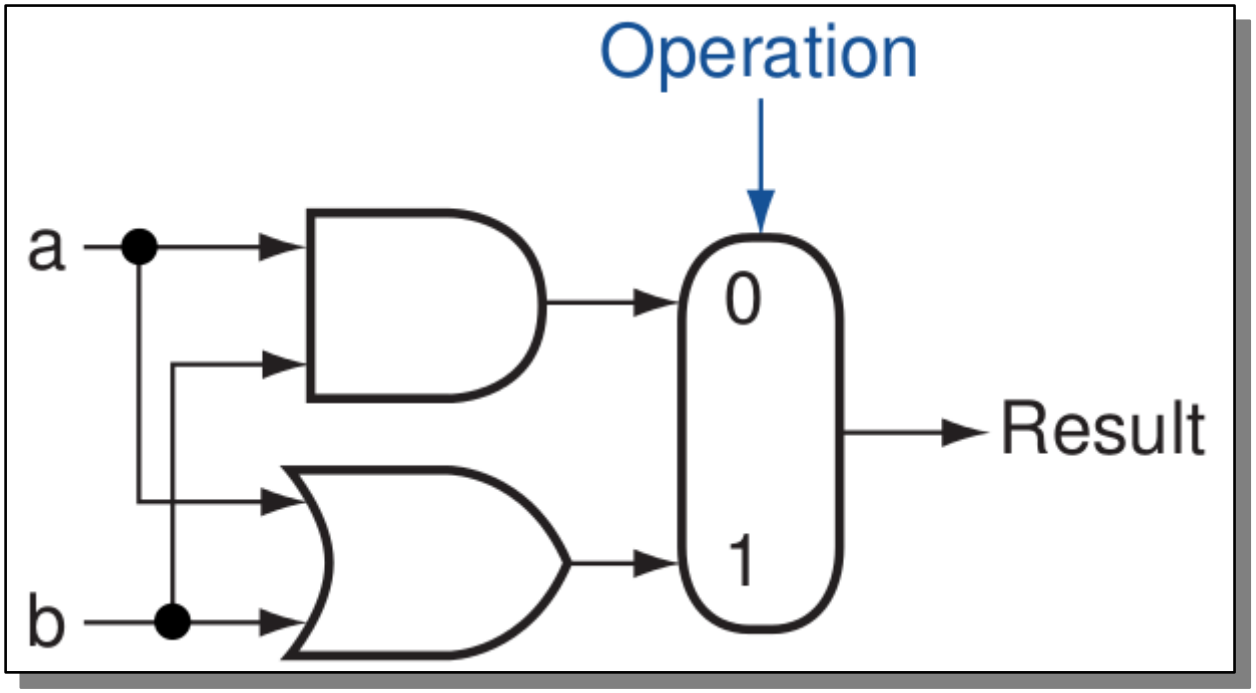
- Start with the simplest operations.
  - AND, OR
  - Control signal
- Incrementally add functionality and control
- Start with a 1-bit ALU
- Modify design for a n-bit ALU

Basic ALU Design



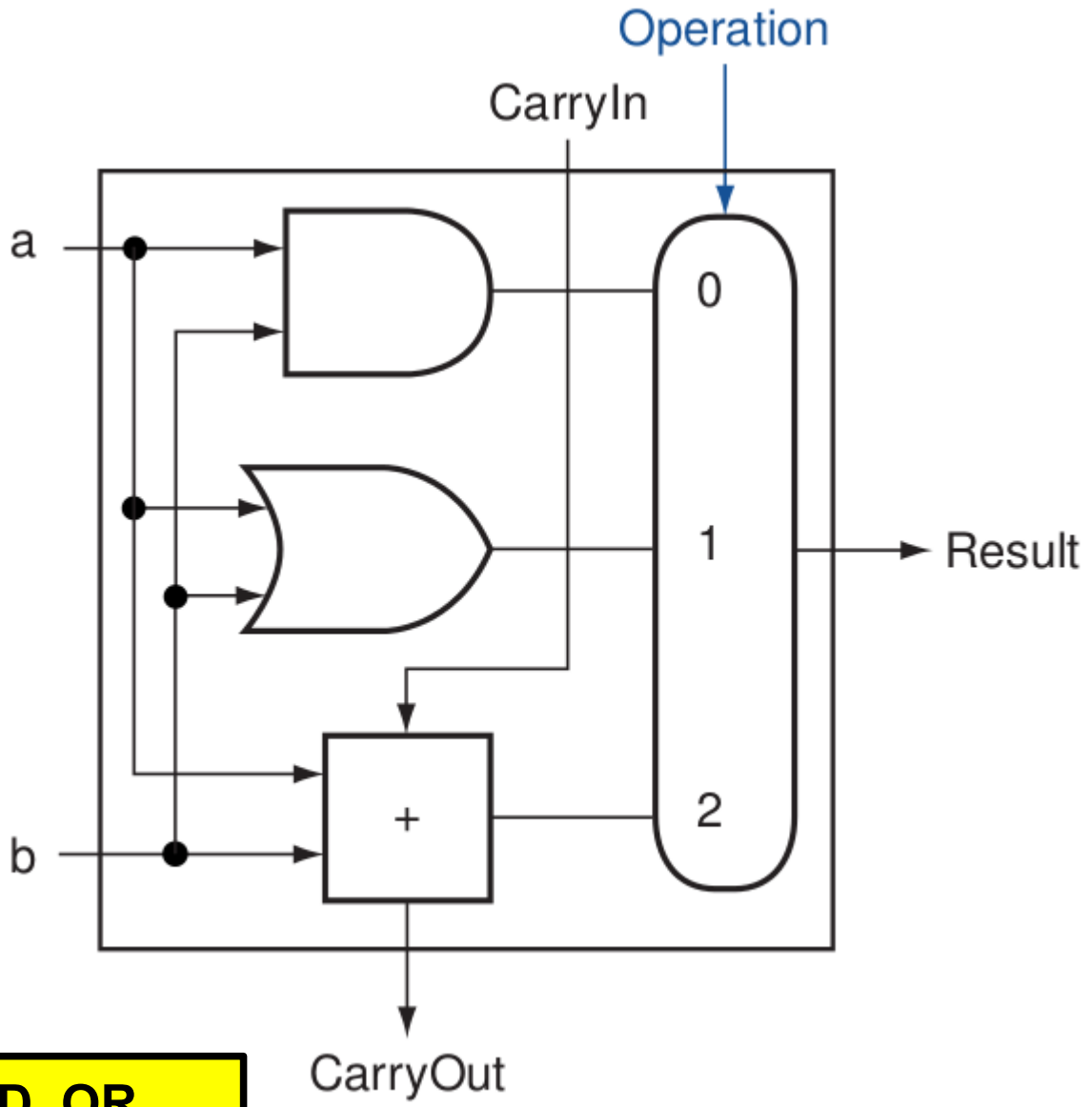
AND, OR

Basic ALU Design



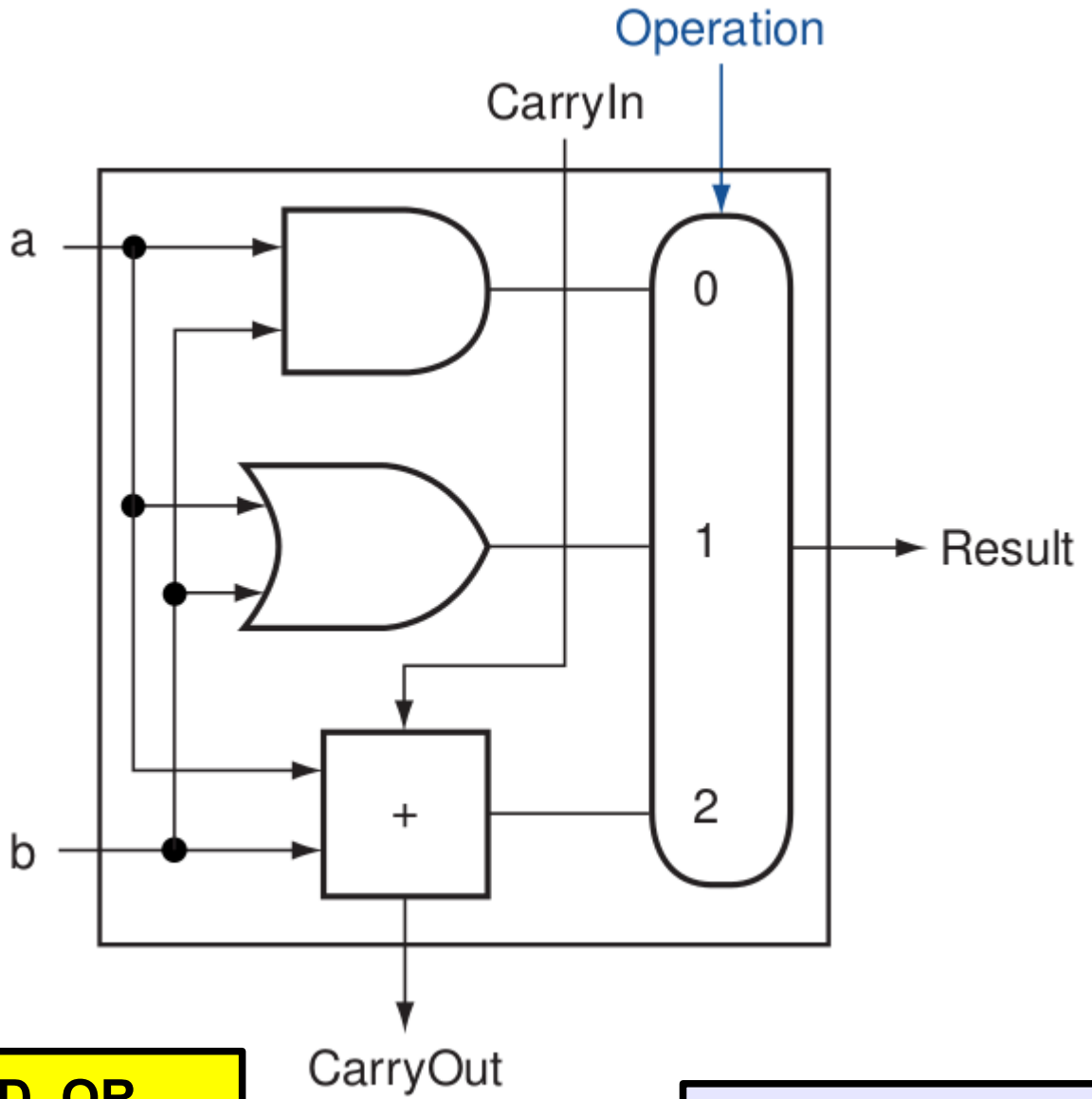
Include a Full Adder

Basic ALU Design



ADDER, AND, OR

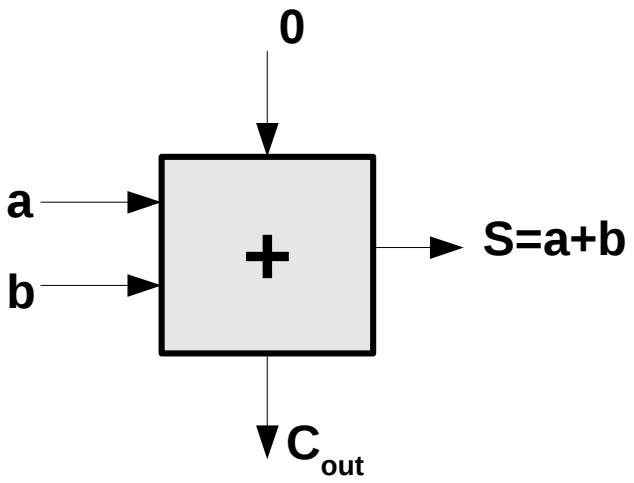
Basic ALU Design



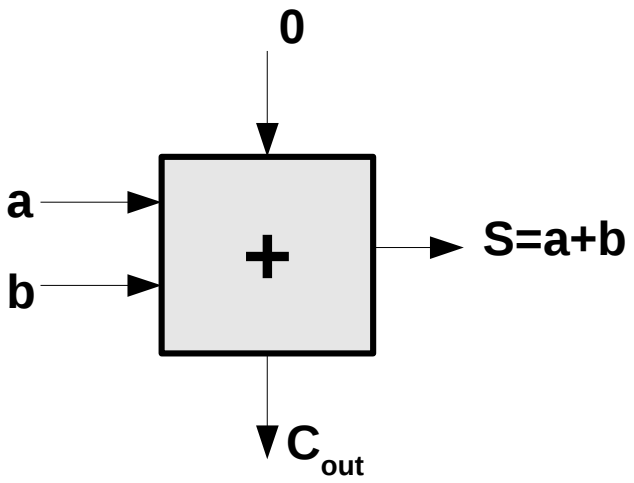
ADDER, AND, OR

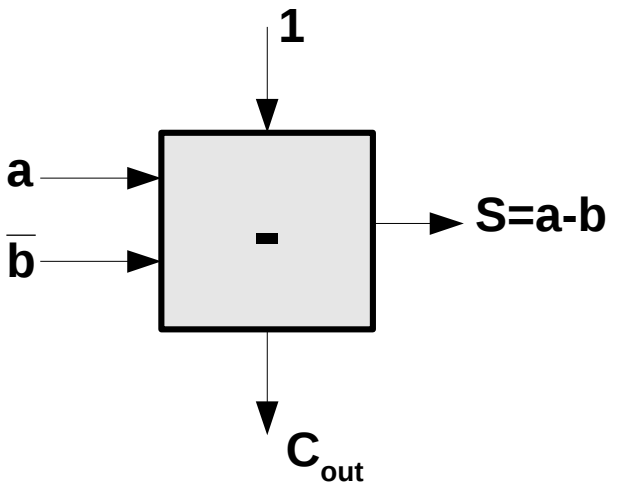
Subtraction?

Adder/Subtractor

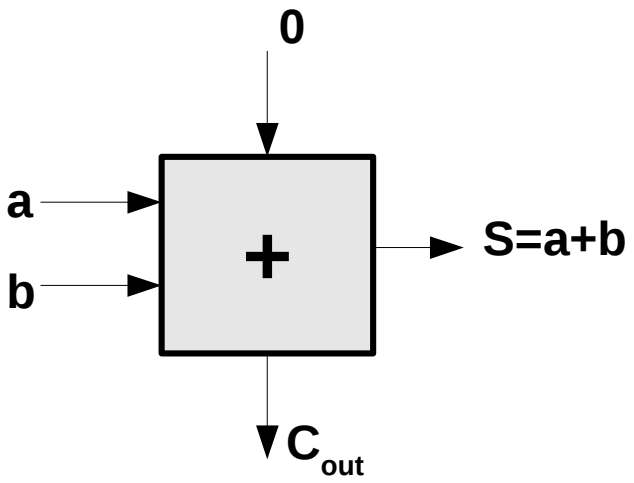


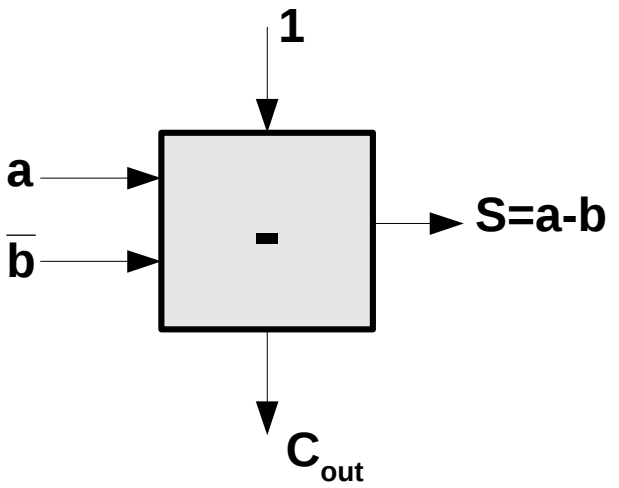
Adder/Subtractor





Adder/Subtractor

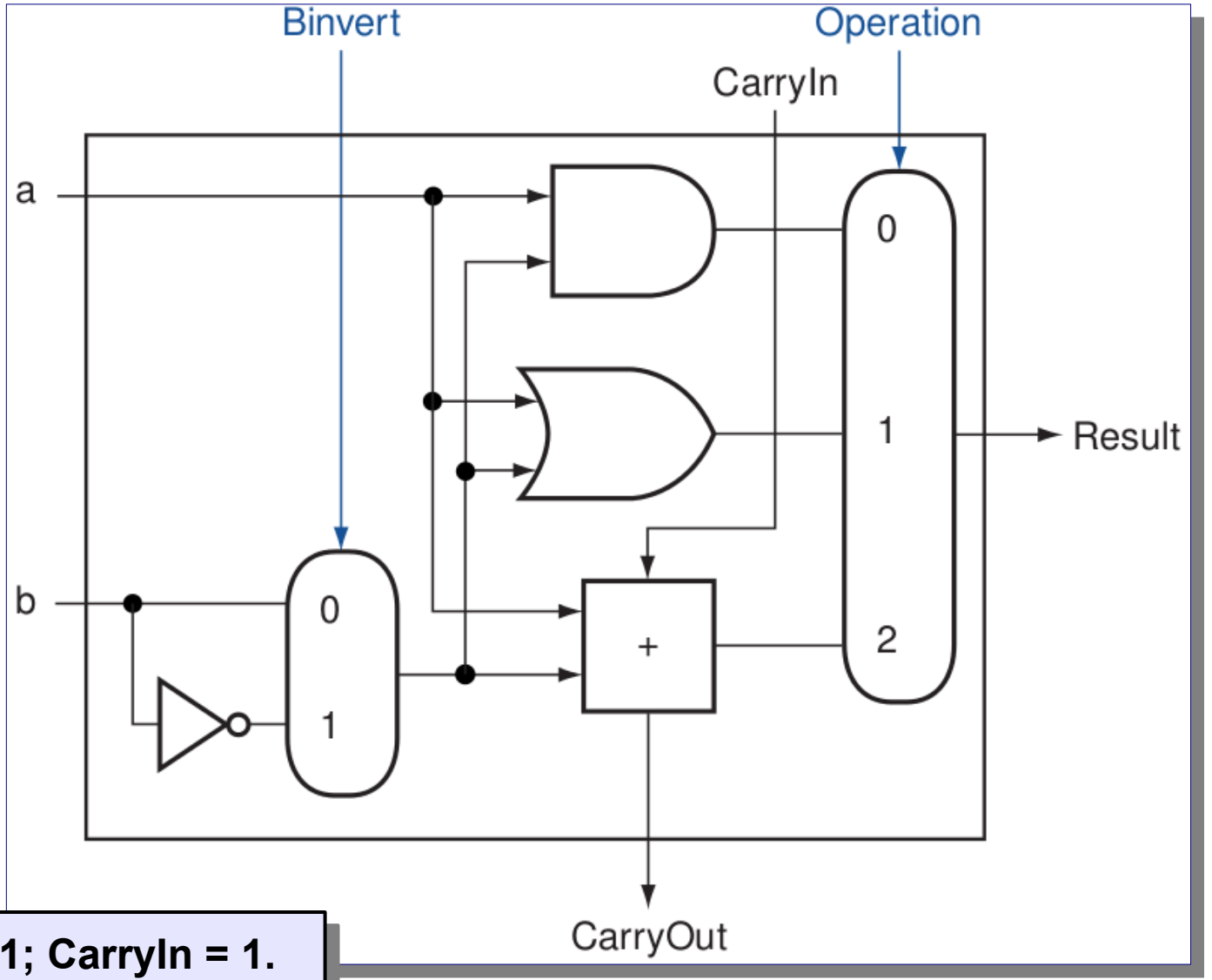




$$-b = \overline{b} + 1$$

A±B, AND, OR

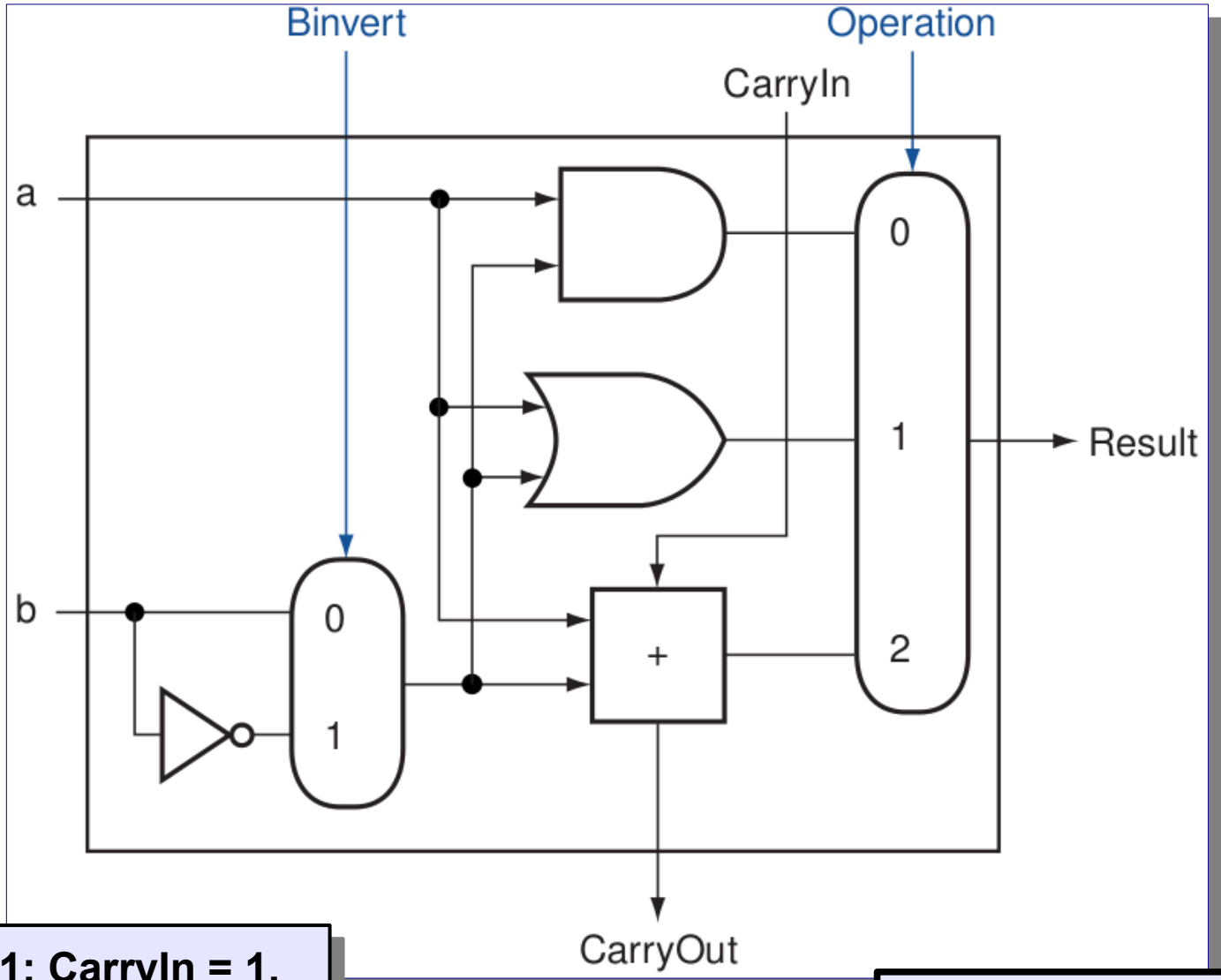
ALU Design



A–B: Binvert = 1; CarryIn = 1.
A+B: Binvert = 0; CarryIn = 0.

A±B, AND, OR

ALU Design



A–B: Binvert = 1; CarryIn = 1.
A+B: Binvert = 0; CarryIn = 0.

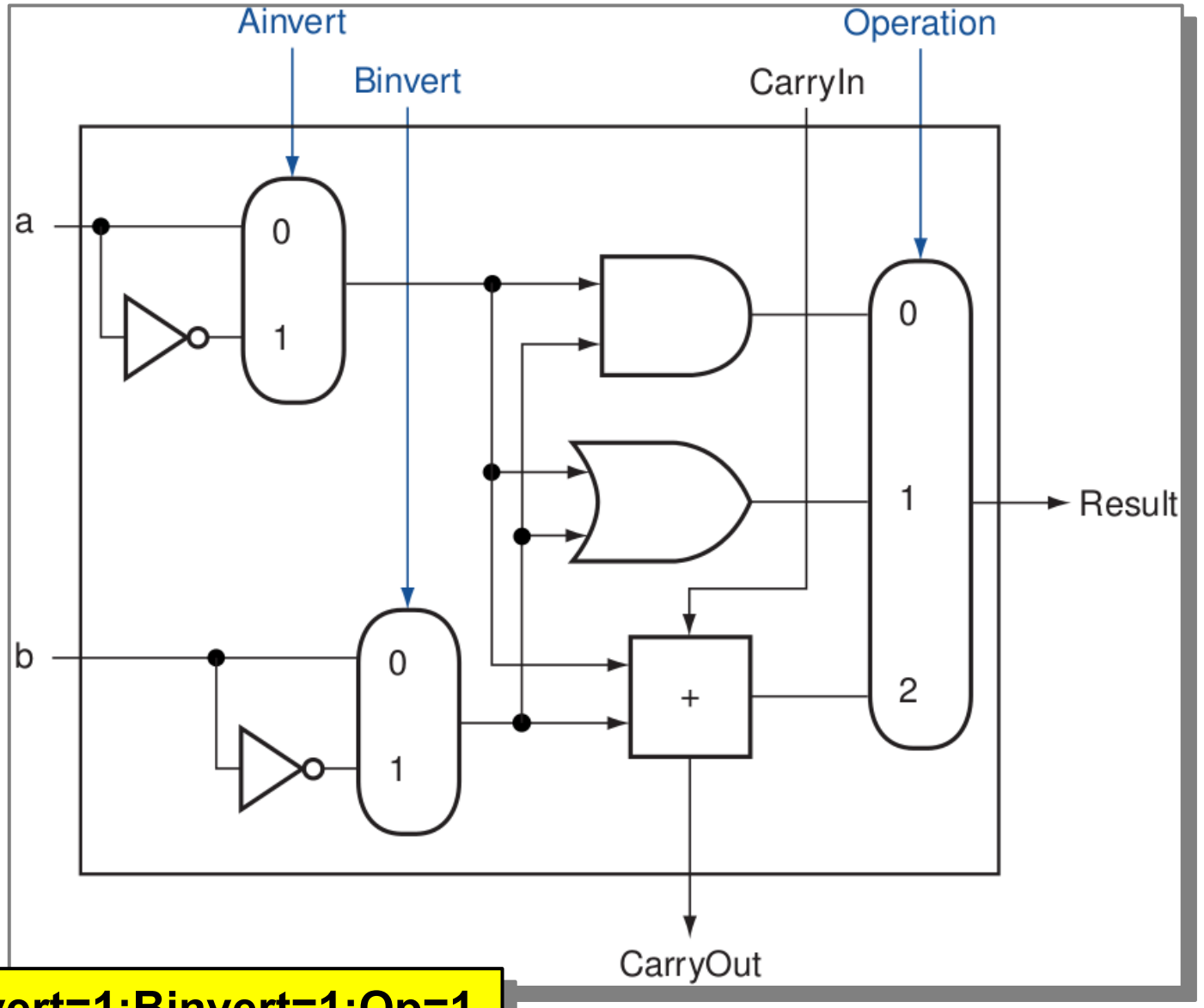
NAND, NOR?

NAND, NOR, NOT

A NAND B = $\overline{(A \text{ AND } B)}$
= $\overline{A}$ OR $\overline{B}$

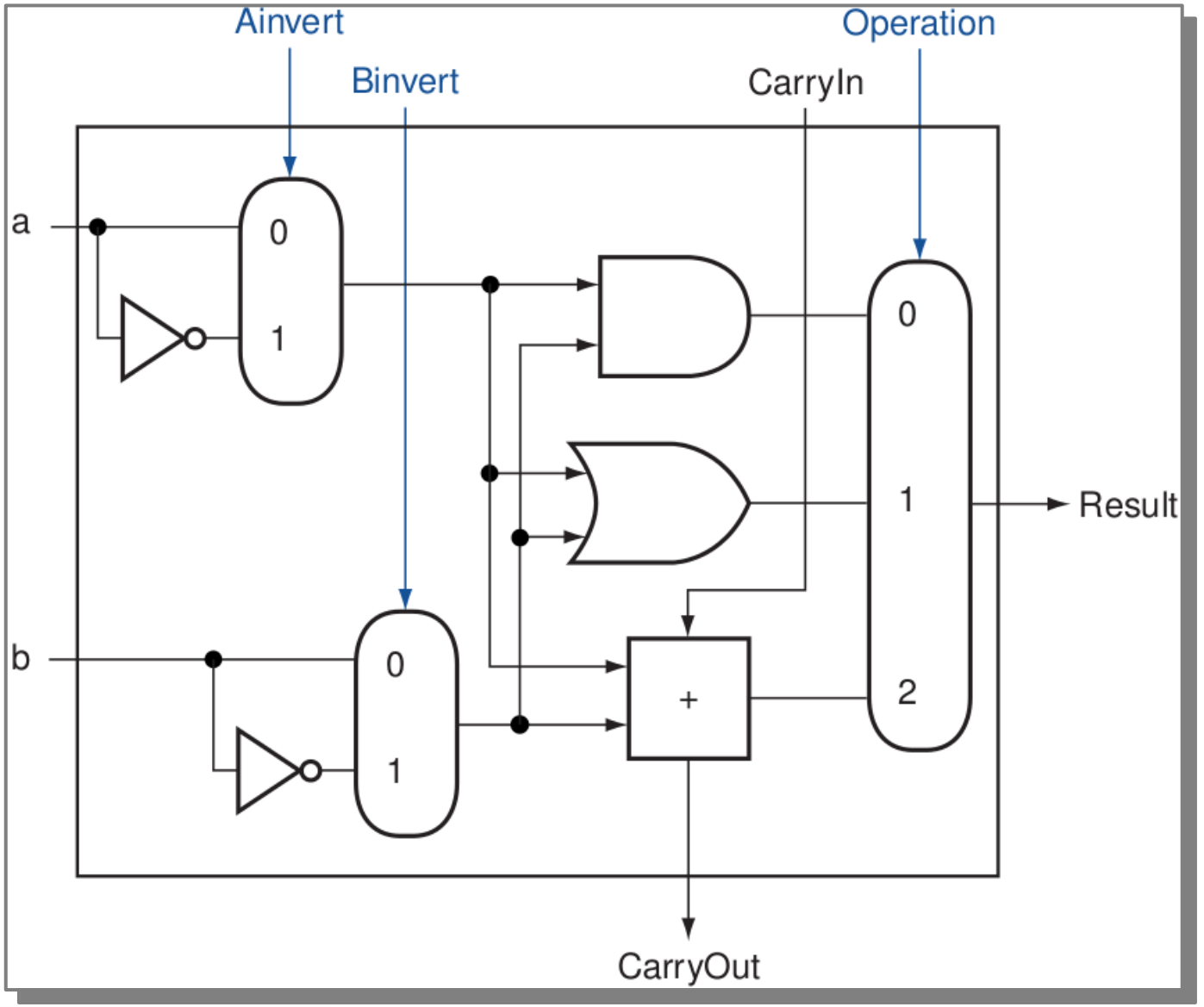
A NOR B = $\overline{(A \text{ OR } B)}$
= $\overline{A}$ AND $\overline{B}$

ALU Design



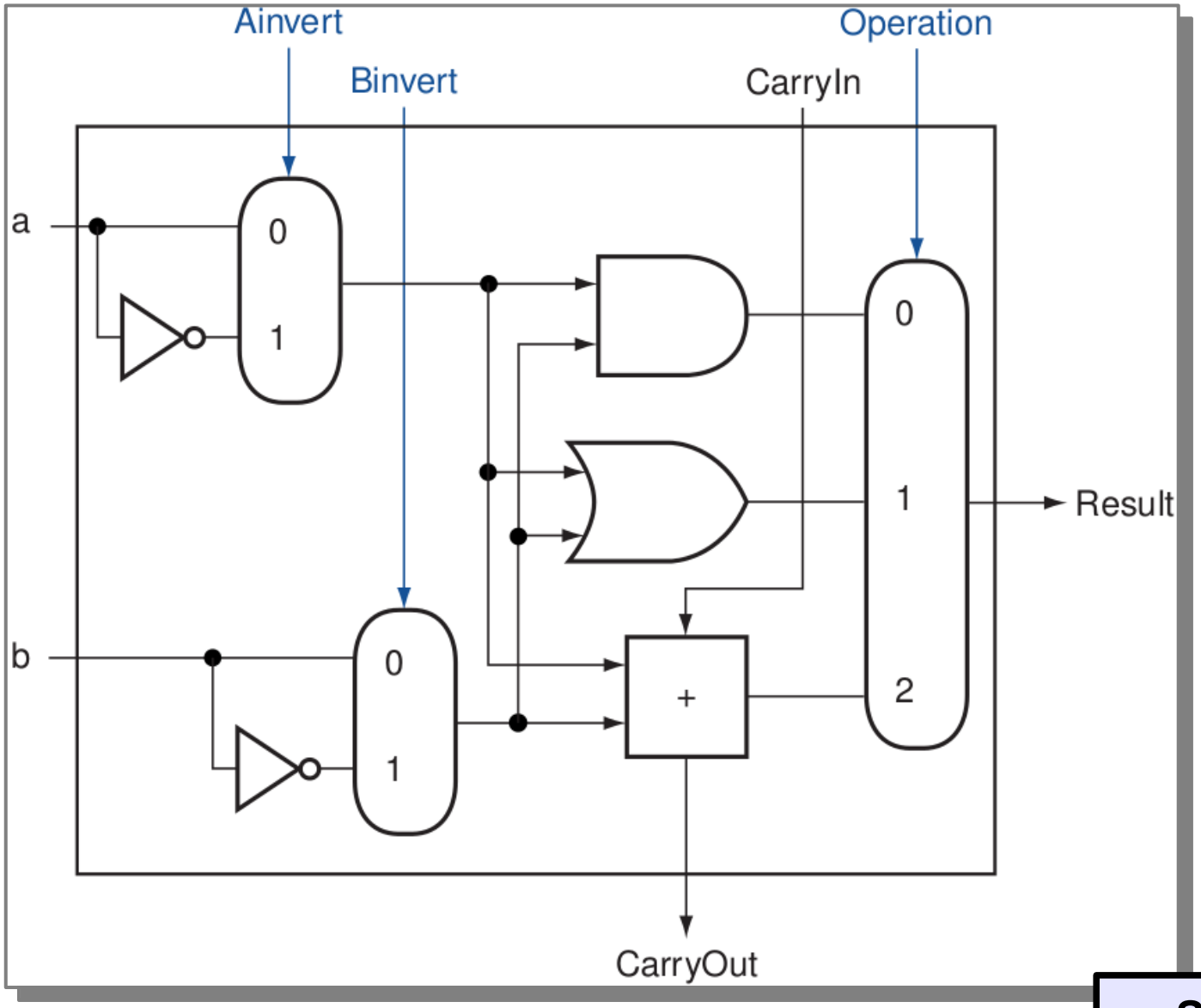
a NAND b: Ainvert=1;Binvert=1;Op=1

ALU Design



A±B, AND, OR, NOR, NAND

ALU Design

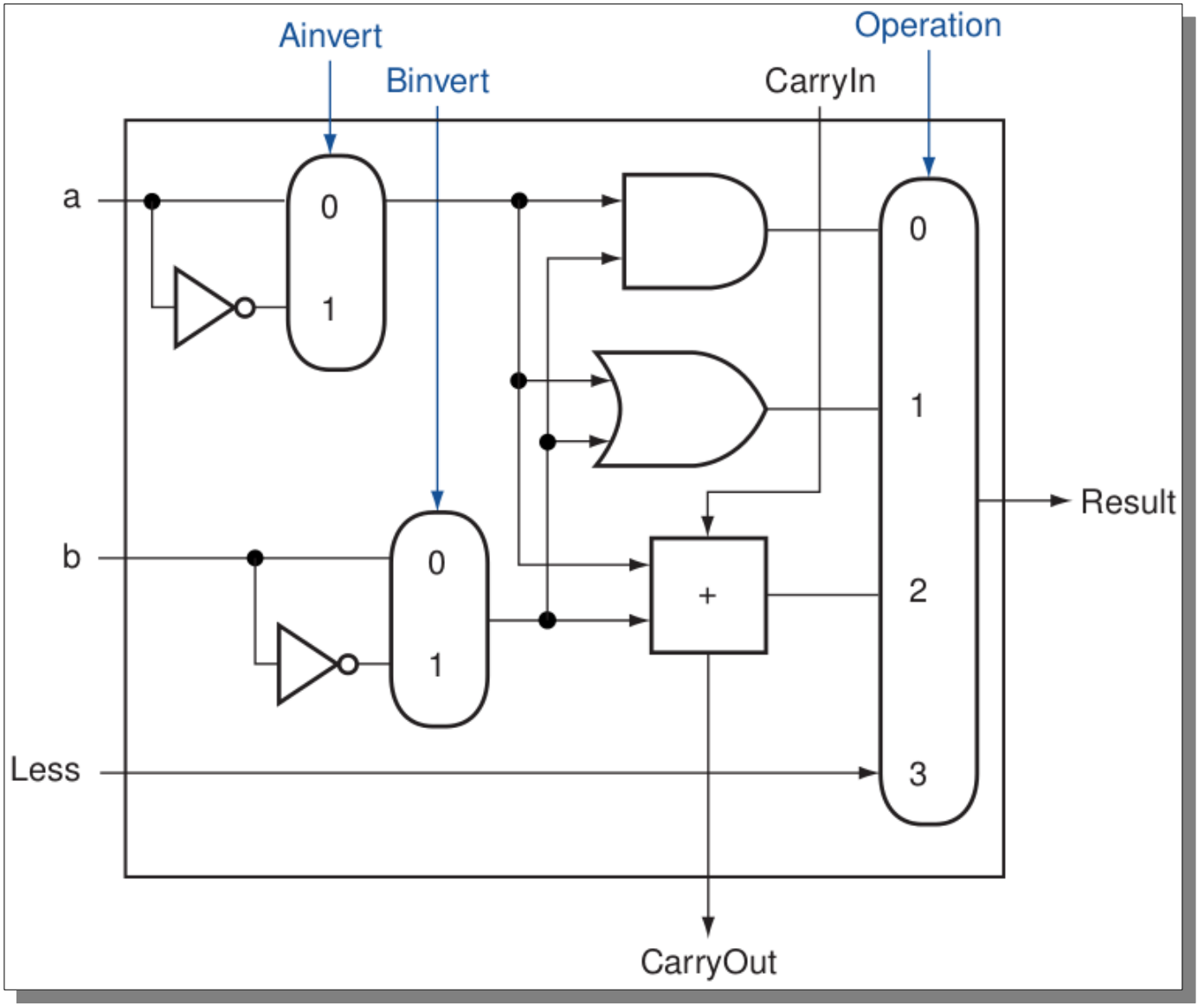


SLT?

Set Less Than

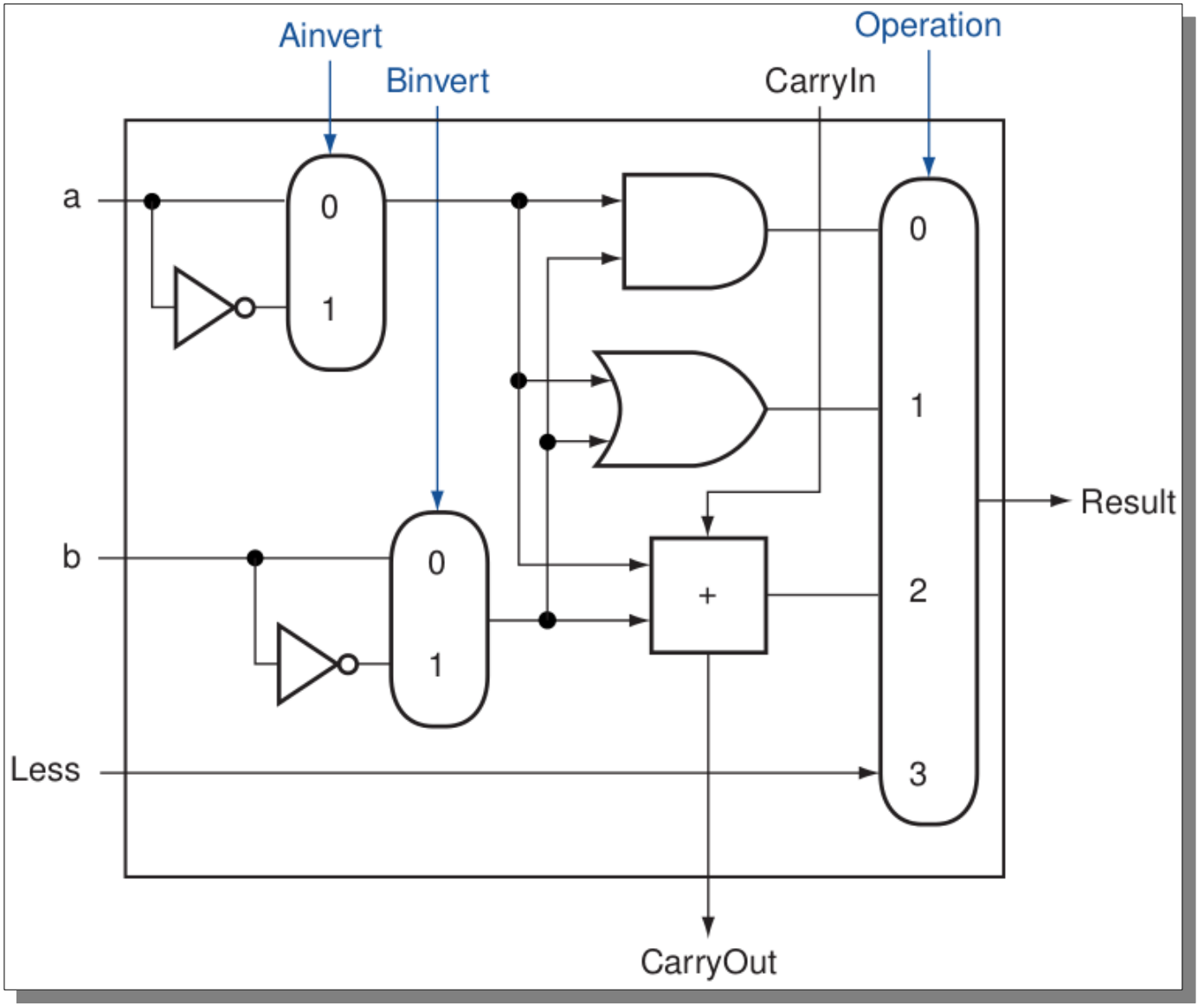
- Set a status bit if (a < b)
- If (a<b); (a-b) is -ve
- Calculate (a-b). Observe sign bit of the result

ALU Design



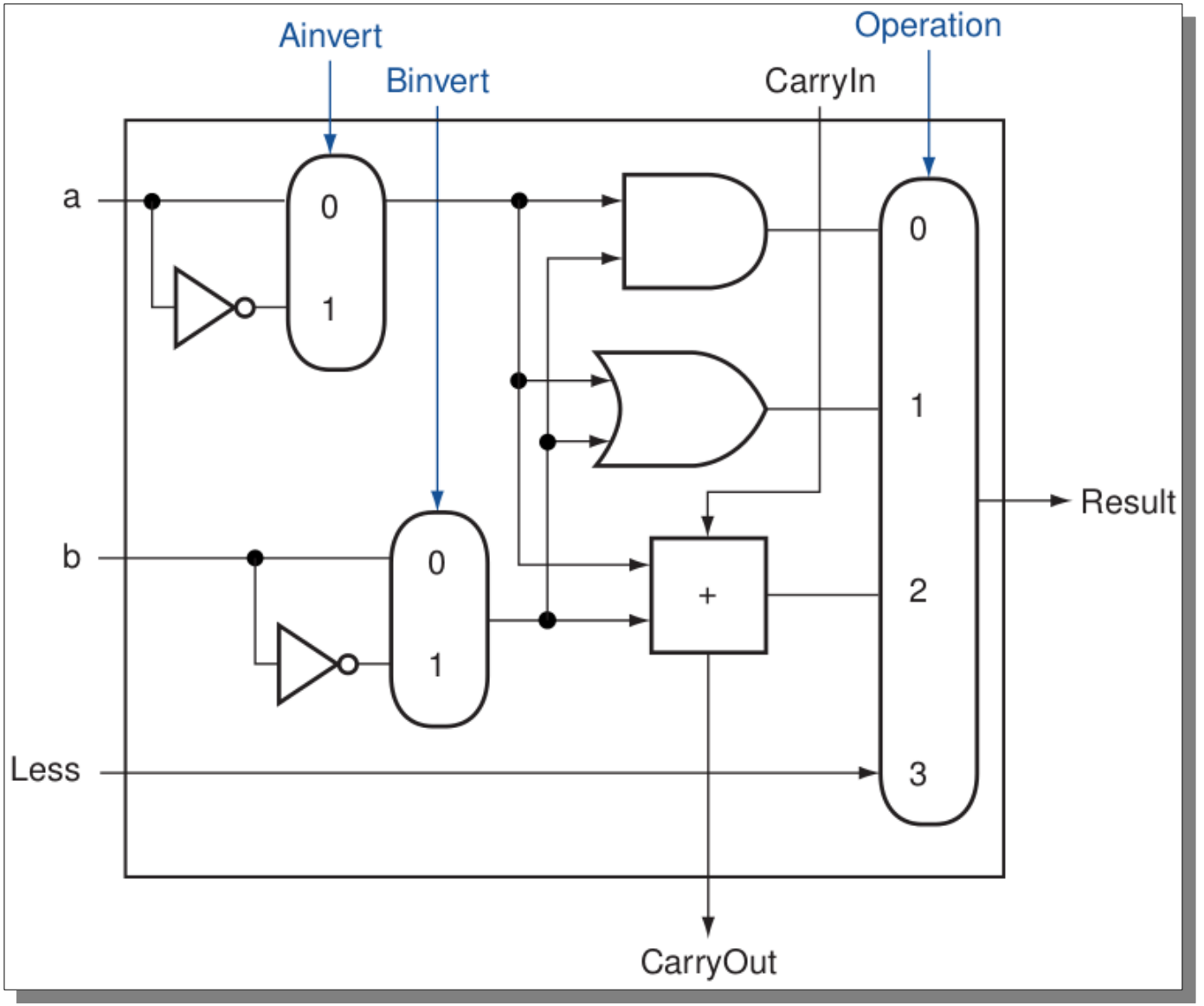
slt(a,b): Binvert=1; CarryIn=1; Operation=3;

ALU Design



A±B, AND, OR, NOR, NAND, SLT

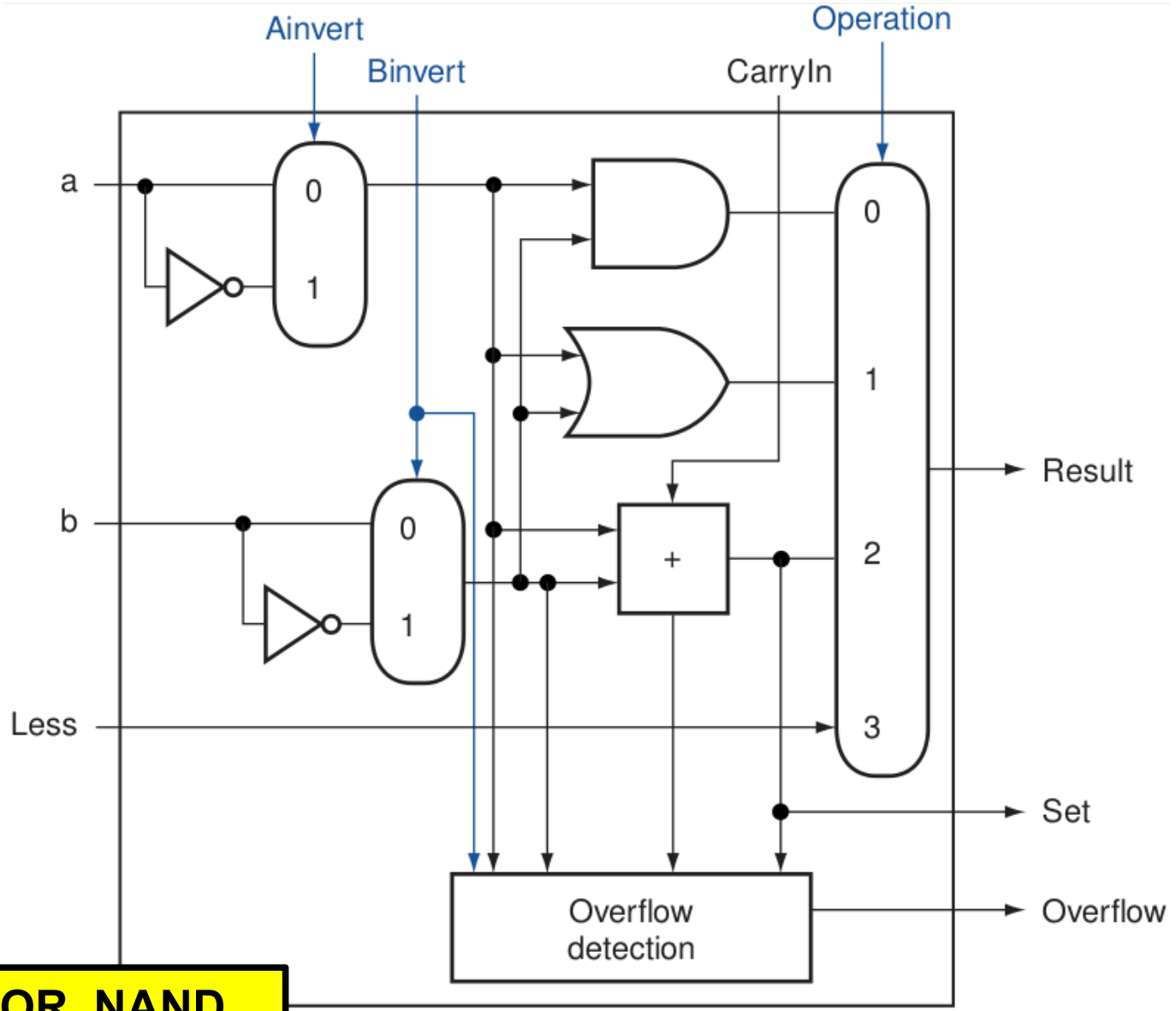
ALU Design



A±B, AND, OR, NOR, NAND, SLT

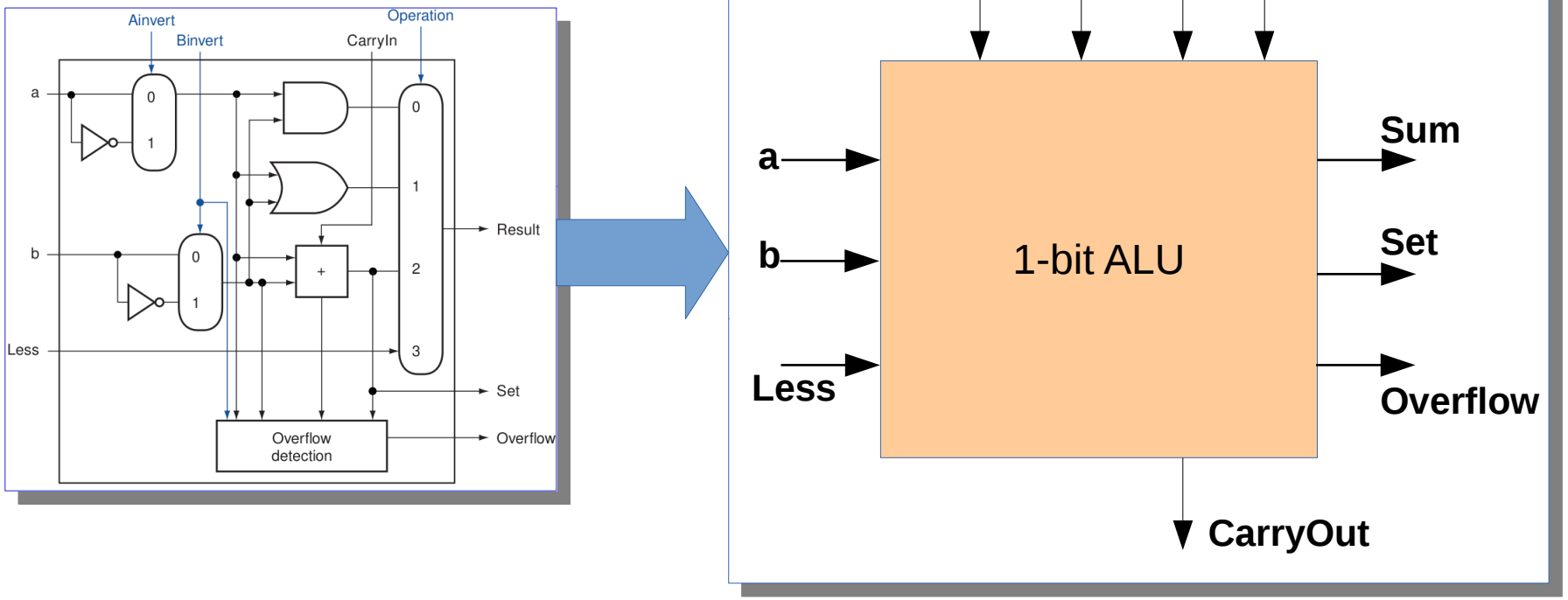
Overflow?

ALU Design

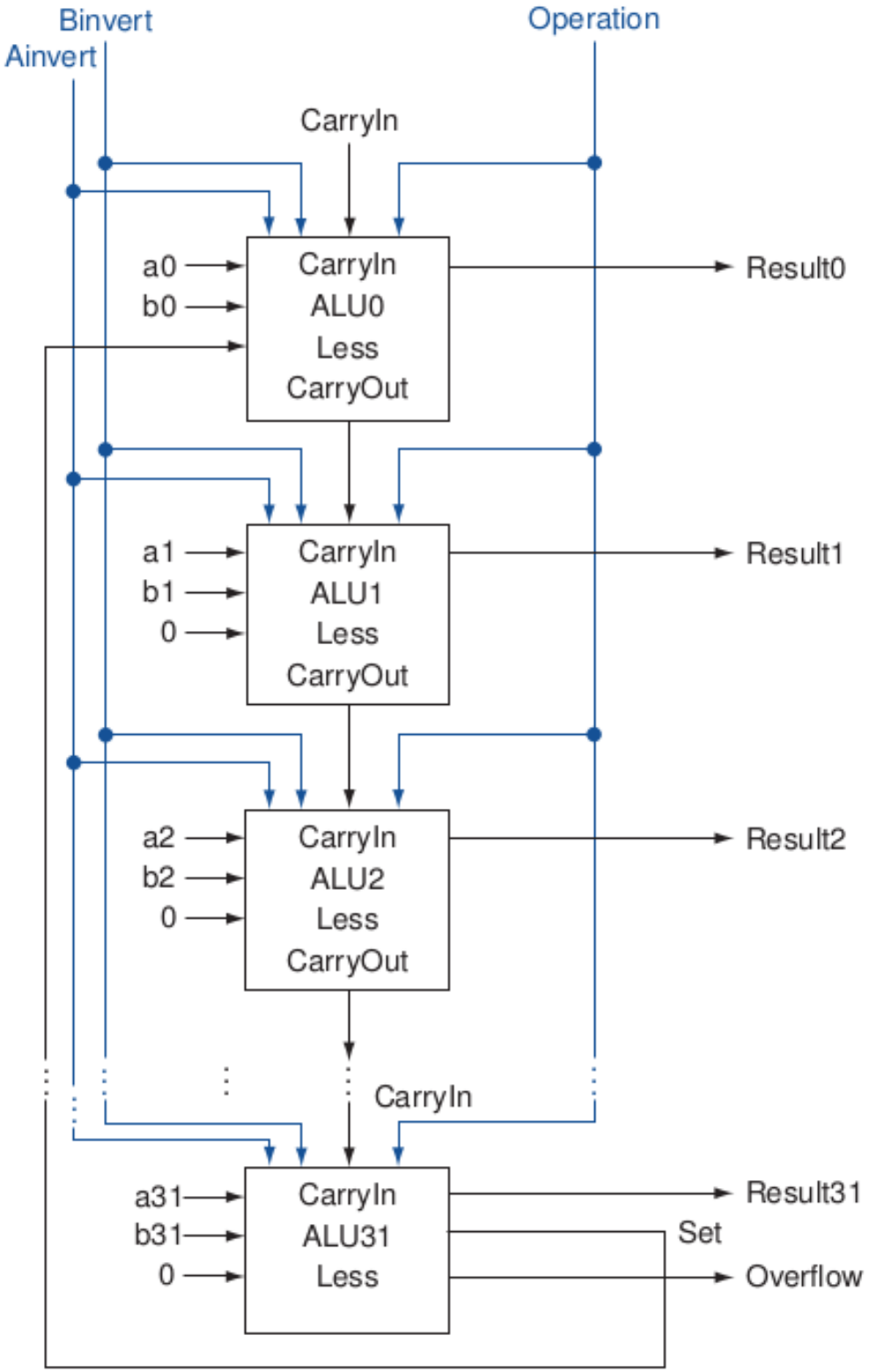


A±B, AND, OR, NOR, NAND, SLT, Overflow detection

ALU Design

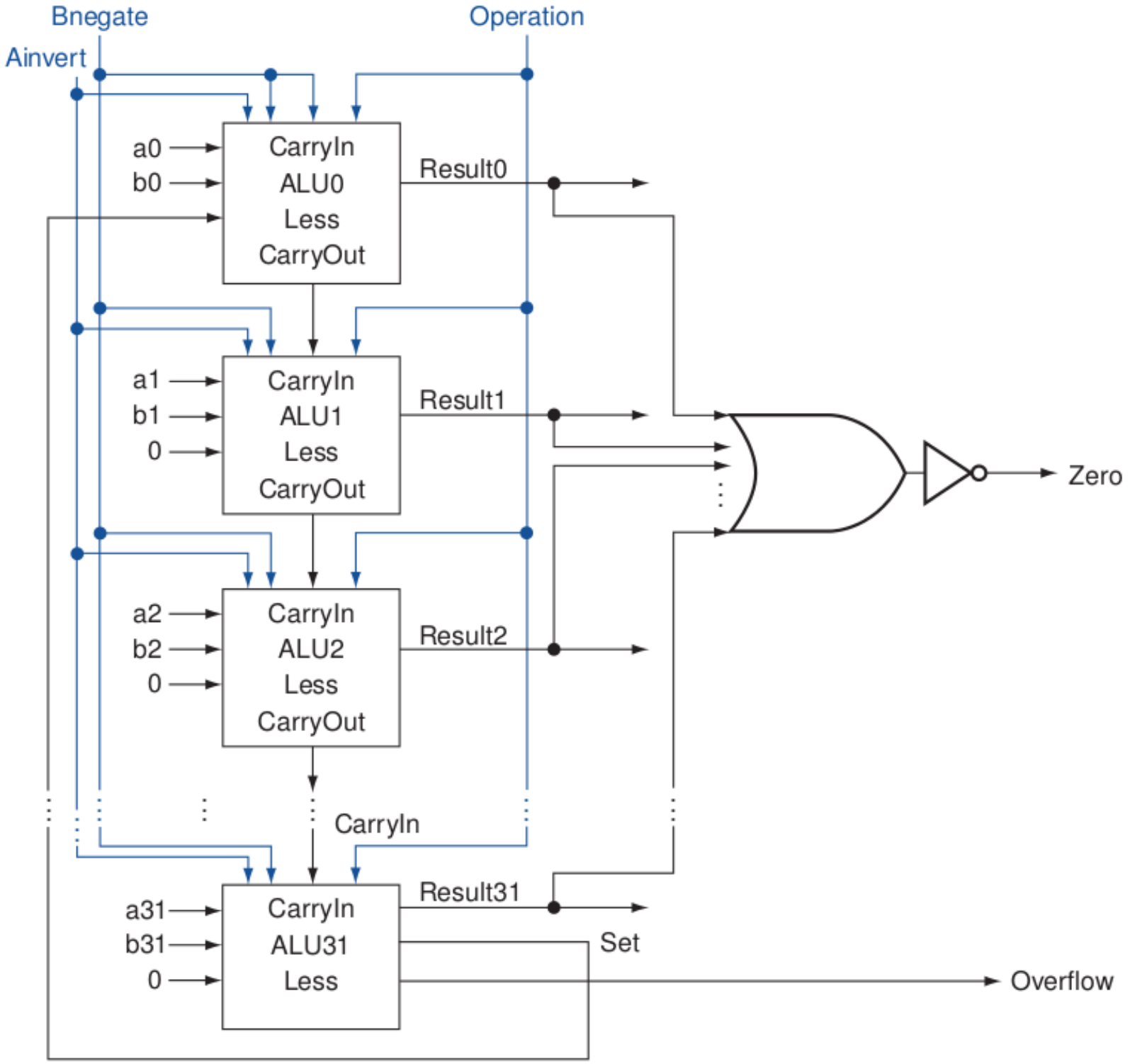

Ainvert
Binvert
CarryIn
Operation
a
b
Less
0
1
2
3
+
Result
Set
Overflow
detection
Overflow
Binvert
Operation
Ainvert
CarryIn
a
b
Less
1-bit ALU
Sum
Set
Overflow
CarryOut

32-bit ALU

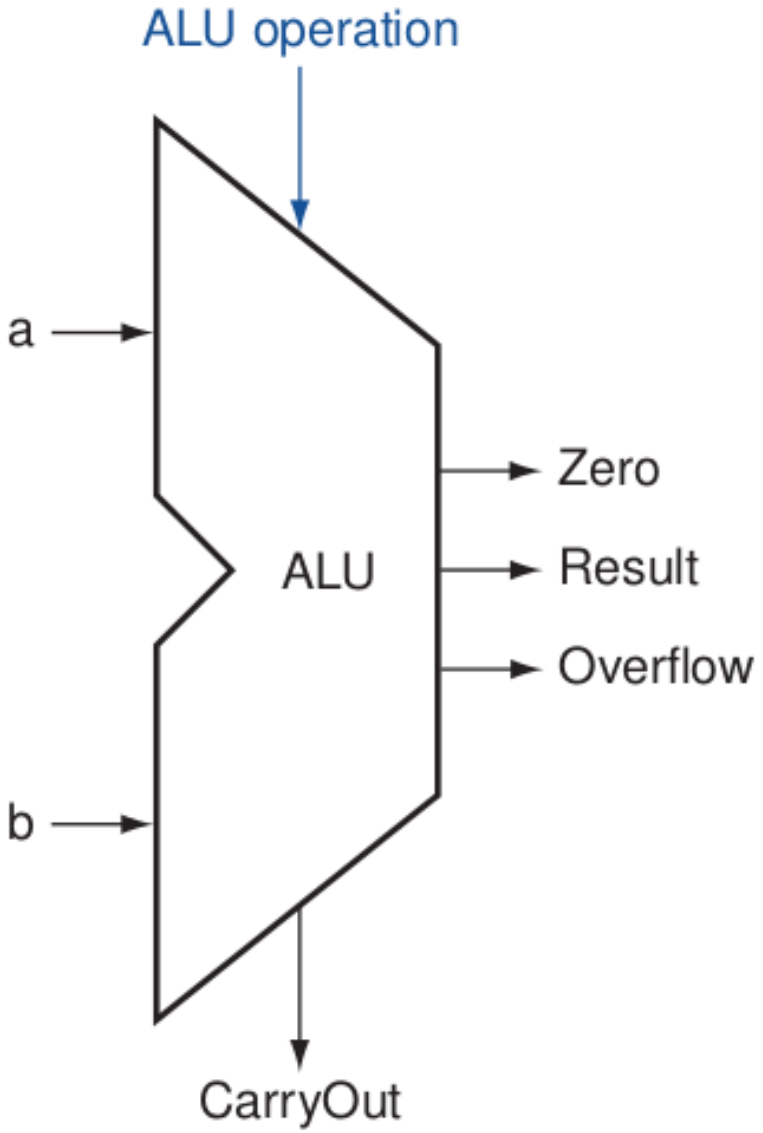


Zero detection?

32-bit ALU

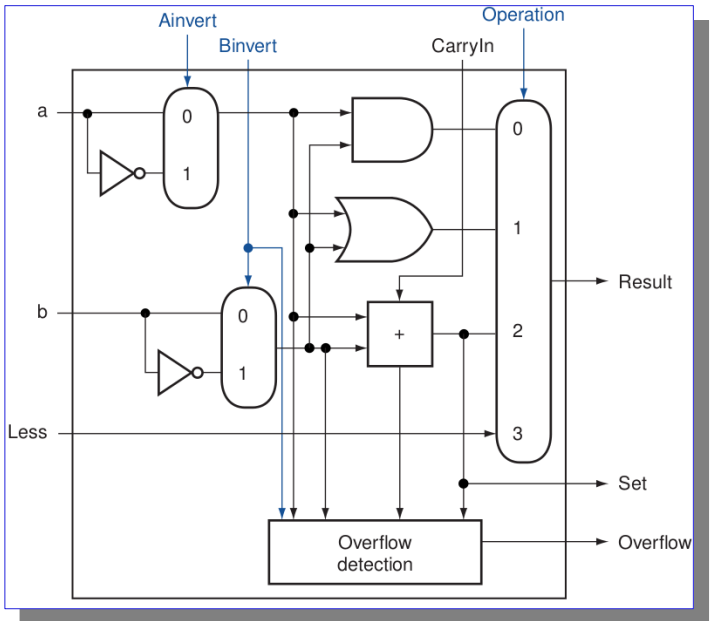

Bnegate
Ainvert
Operation
a0
b0
CarryIn
ALU0
Less
CarryOut
Result0
a1
b1
0
CarryIn
ALU1
Less
CarryOut
Result1
a2
b2
0
CarryIn
ALU2
Less
CarryOut
Result2
CarryIn
a31
b31
0
CarryIn
ALU31
Less
Result31
Set
Overflow
Zero

ALU



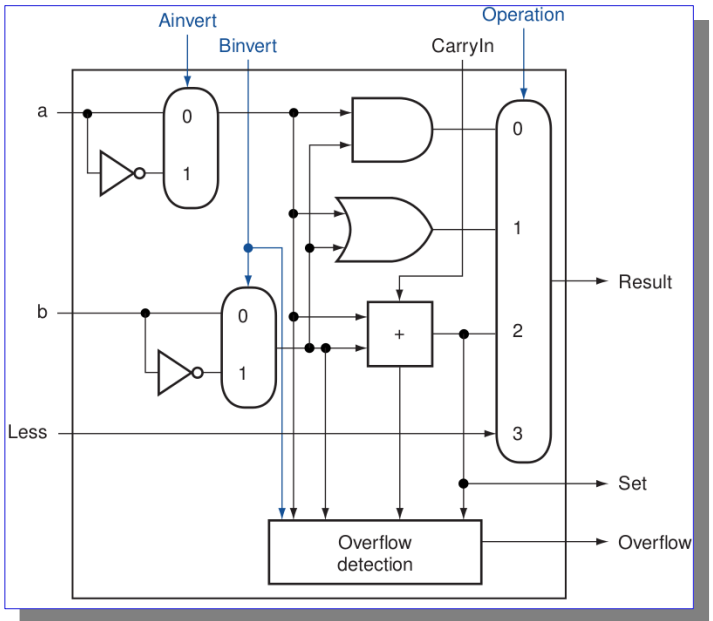
ALU – Control Signals

	Ainvert	Binvert	CarryIn	Operation
AND				
OR				
ADD				
SUB				
NAND				
NOR				
SLT				



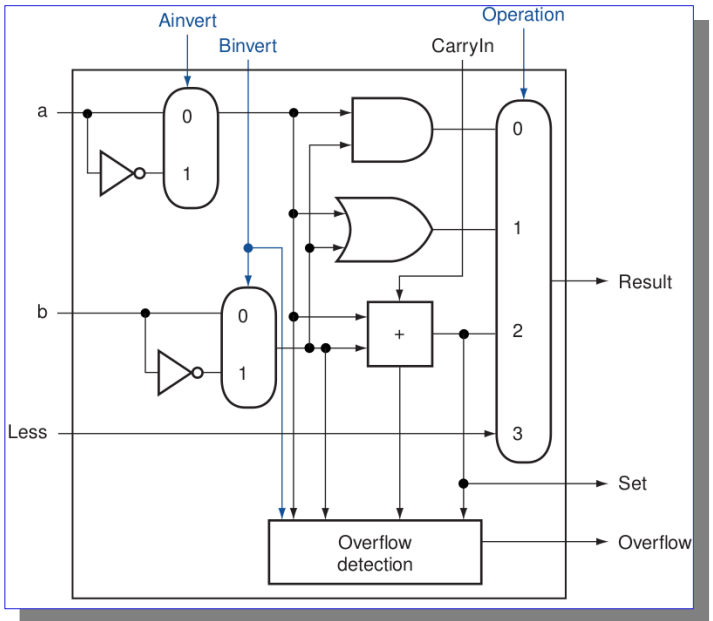
ALU – Control Signals

	Ainvert	Binvert	CarryIn	Operation
AND	0	0	0	00
OR				
ADD				
SUB				
NAND				
NOR				
SLT				



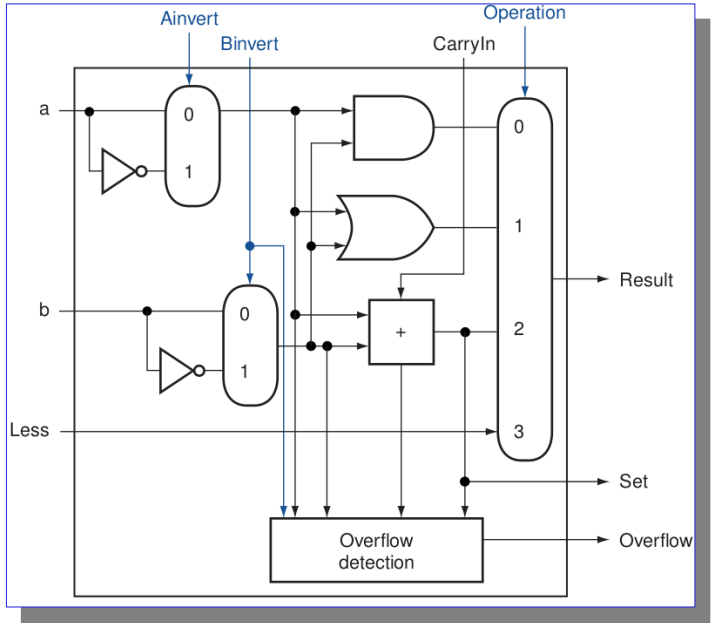
ALU – Control Signals

	Ainvert	Binvert	CarryIn	Operation
AND	0	0	0	00
OR	0	0	0	01
ADD				
SUB				
NAND				
NOR				
SLT				



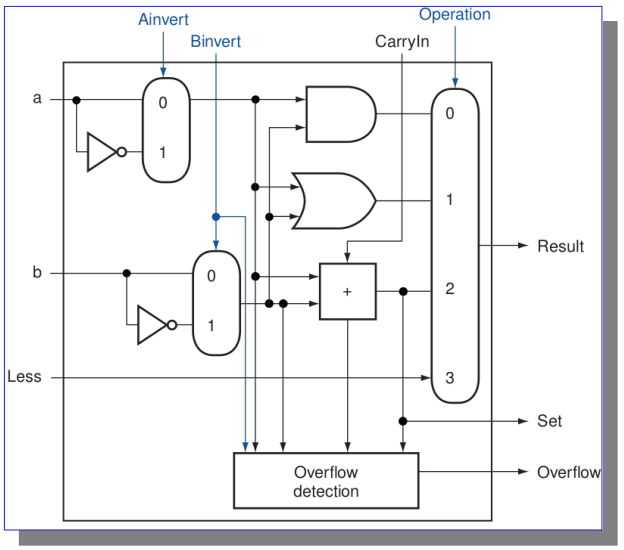
ALU – Control Signals

	Ainvert	Binvert	CarryIn	Operation
AND	0	0	0	00
OR	0	0	0	01
ADD	0	0	0	10
SUB	0	1	1	10
NAND				
NOR				
SLT				



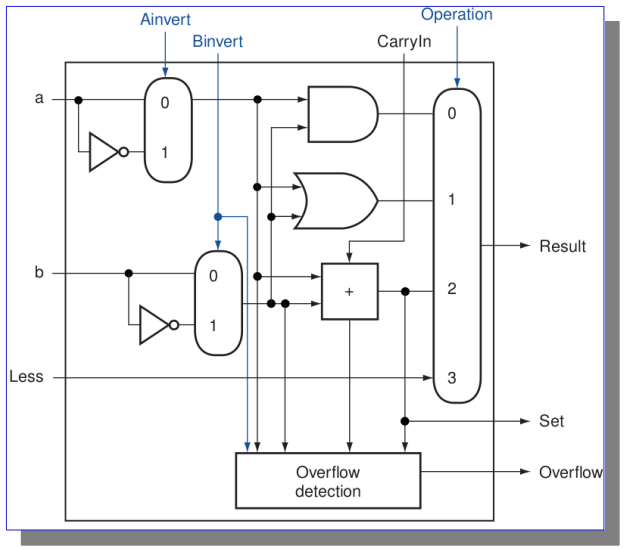
ALU – Control Signals

	Ainvert	Binvert	CarryIn	Operation
AND	0	0	0	00
OR	0	0	0	01
ADD	0	0	0	10
SUB	0	1	1	10
NAND	1	1	0	01
NOR	1	1	0	00
SLT				



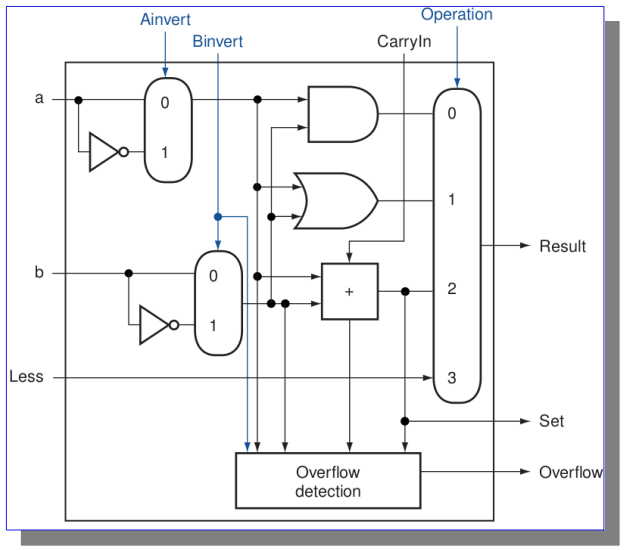
ALU – Control Signals

	Ainvert	Binvert	CarryIn	Operation
AND	0	0	0	00
OR	0	0	0	01
ADD	0	0	0	10
SUB	0	1	1	10
NAND	1	1	0	01
NOR	1	1	0	00
SLT	0	1	1	11



ALU – Control Signals

	Ainvert	Binvert	CarryIn	Operation
AND	0	0	0	00
OR	0	0	0	01
ADD	0	0	0	10
SUB	0	1	1	10
NAND	1	1	0	01
NOR	1	1	0	00
SLT	0	1	1	11

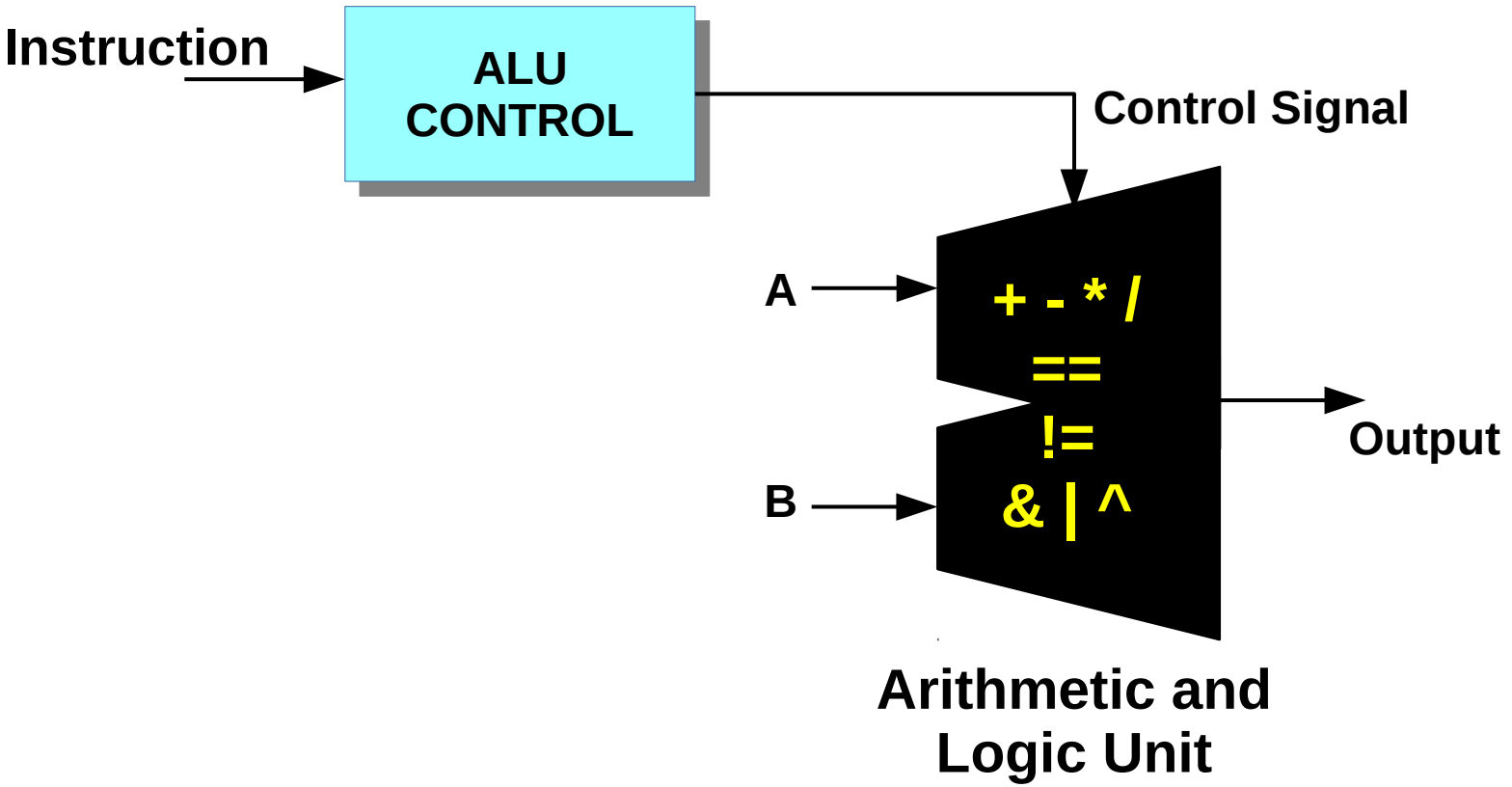


Ignore the effect of CarryIn during NAND and NOR execution. **Binvert == CarryIn.**

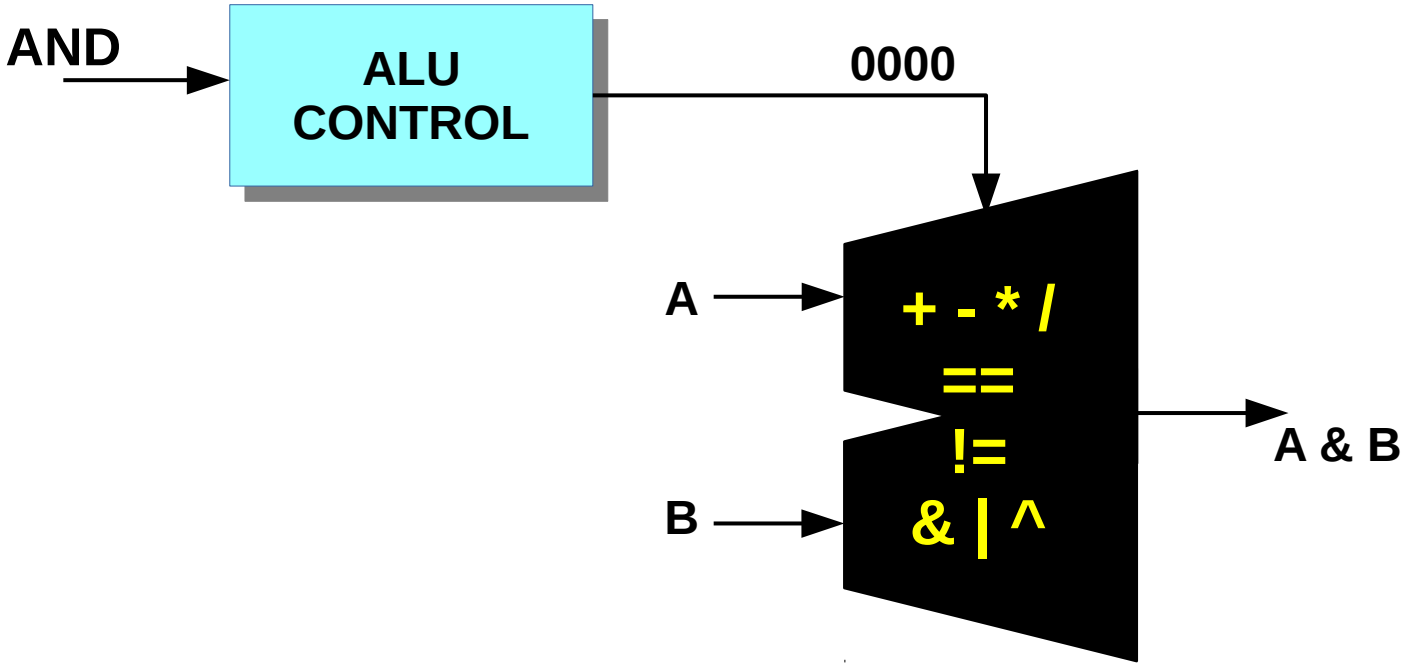
ALU – Control Signals

	$\overline{\overline{A}}$BOp
AND	**0000**
OR	**0001**
ADD	**0010**
SUB	**0110**
NAND	**1101**
NOR	**1100**
SLT	**0111**

ALU Control


Instruction
ALU
CONTROL
Control Signal
A
B
+ - * /
==
!=
& | ^
Output
Arithmetic and
Logic Unit

ALU Control – AND Instruction



Instructions using the ALU

- Arithmetic and Logic Instructions
  - R-type and I-type
- Memory transfer instructions
  - Effective address calculation
  - I-type
- Branch instructions (BEQ)
  - For Zero detection
  - I-type

ALU – Control Signals

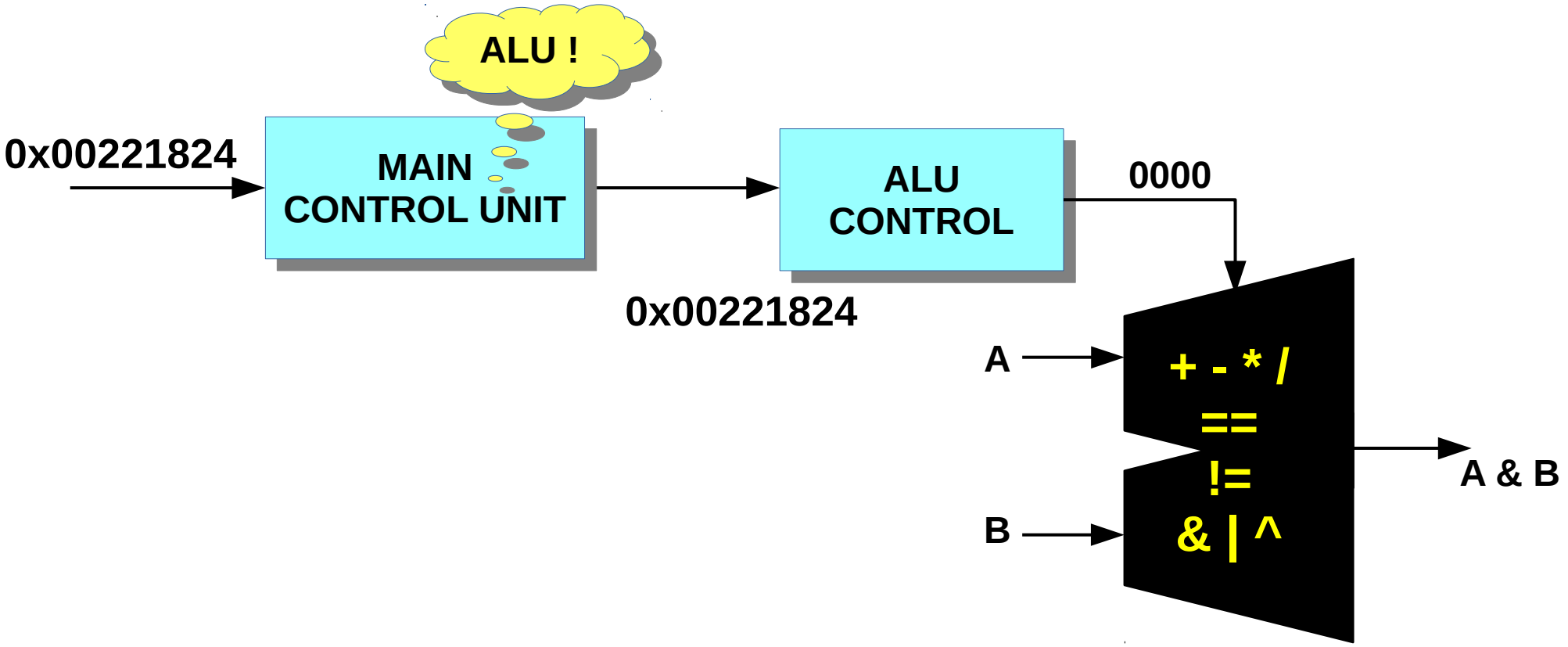
	$\overline{\text{AB}}$Op
AND	**0000**
OR	**0001**
ADD	**0010**
SUB	**0110**
NAND	**1101**
NOR	**1100**
SLT	**0111**

Which control signals for LW, SW, BEQ?

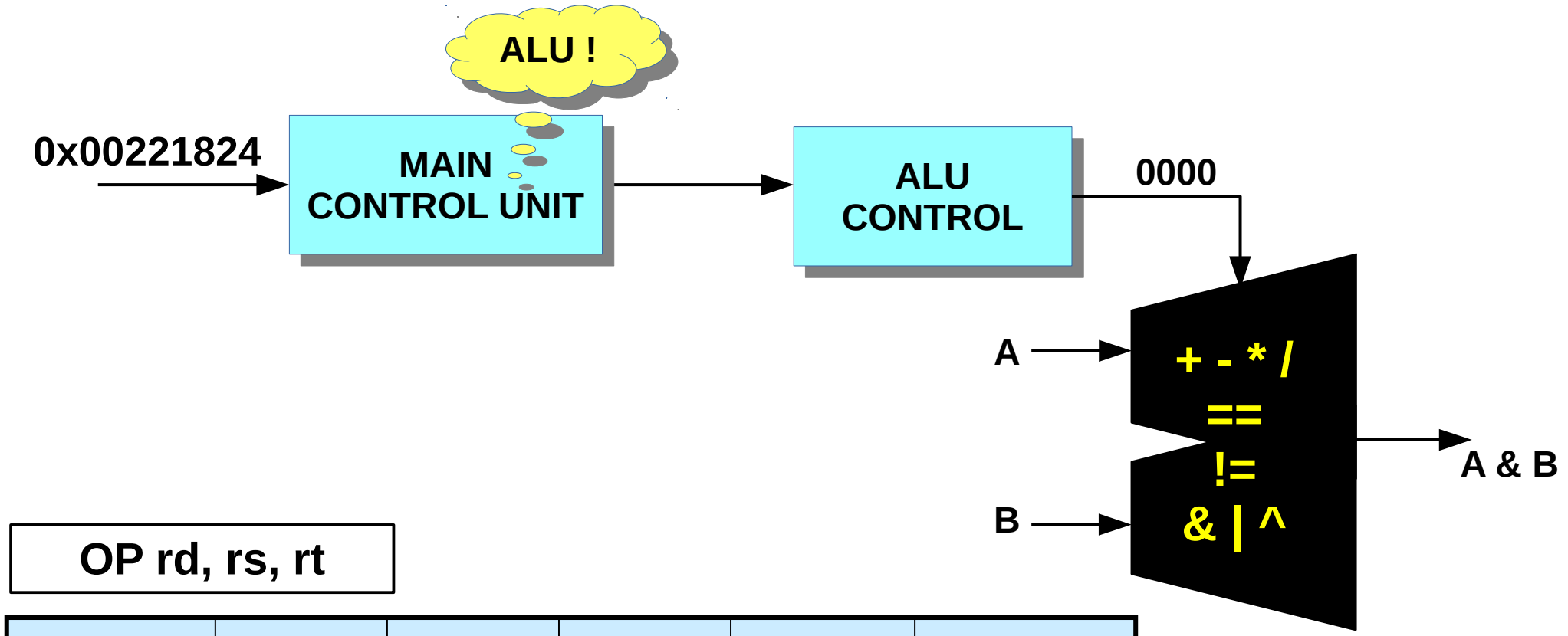
ALU – Control Signals

Input to ALU CU	Output Control Signals $\overline{AB}Op$
AND	**0000**
OR	**0001**
ADD, LW, SW	**0010**
SUB, BEQ	**0110**
NAND	**1101**
NOR	**1100**
SLT	**0111**

ALU Control – AND Instruction



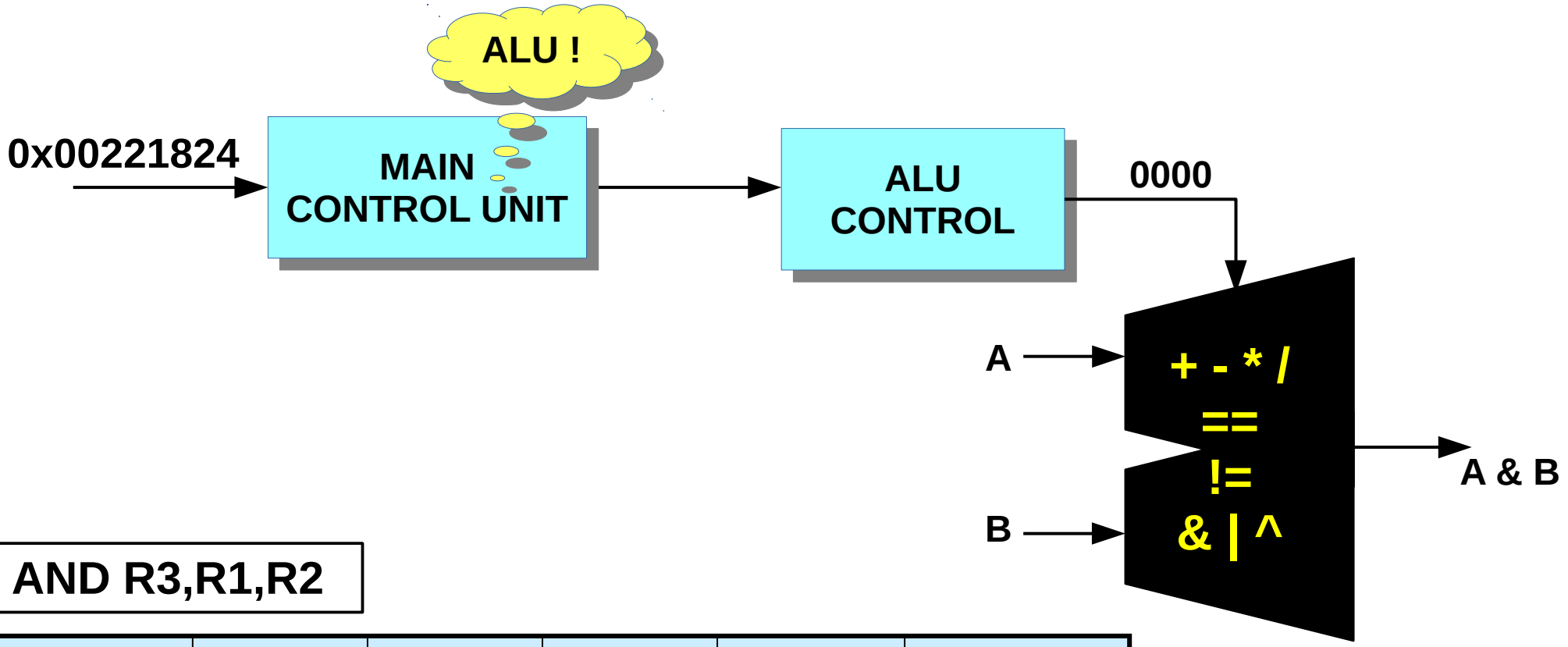
ALU Control – AND Instruction



OP rd, rs, rt

op	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

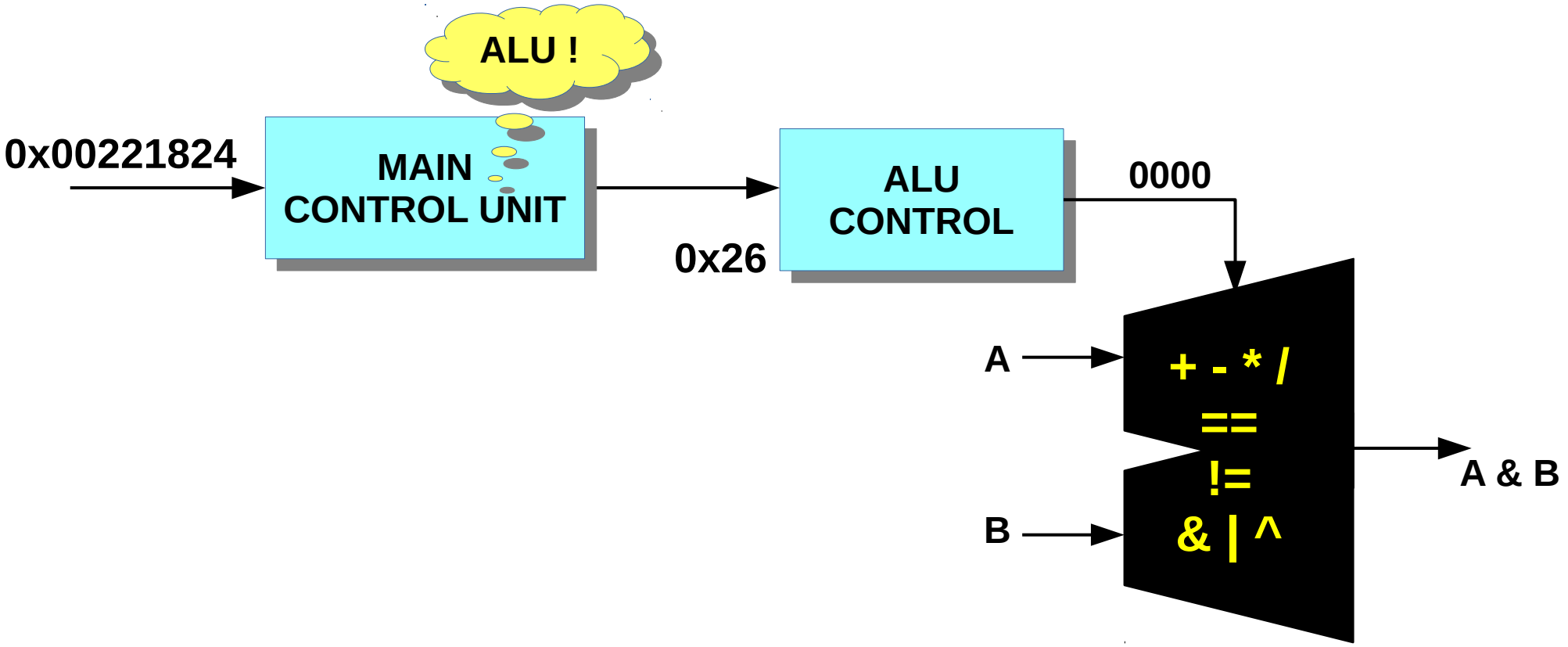
ALU Control – AND Instruction



AND R3,R1,R2

0x0	1	2	3	0	0x26
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

ALU Control – AND (R-type)

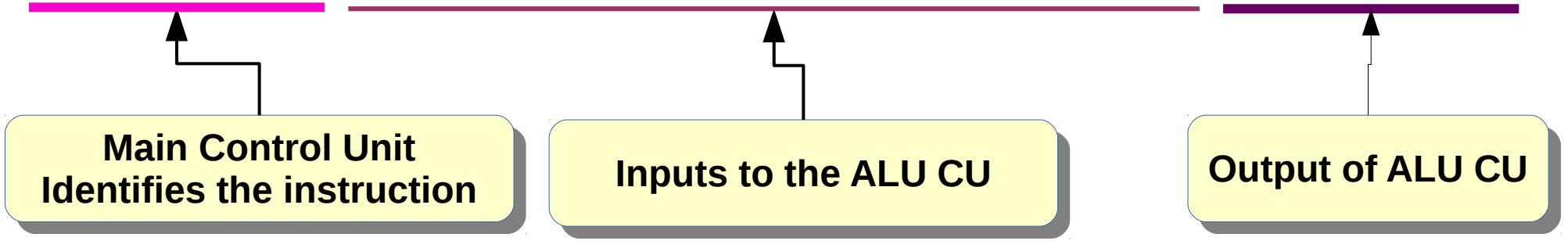


Truth Table of the ALU CU

Instruction	ALU action	Type	Funct field	Control Signal
LW				
SW				
BEQ				
ALU instruction				
ALU Immediate				

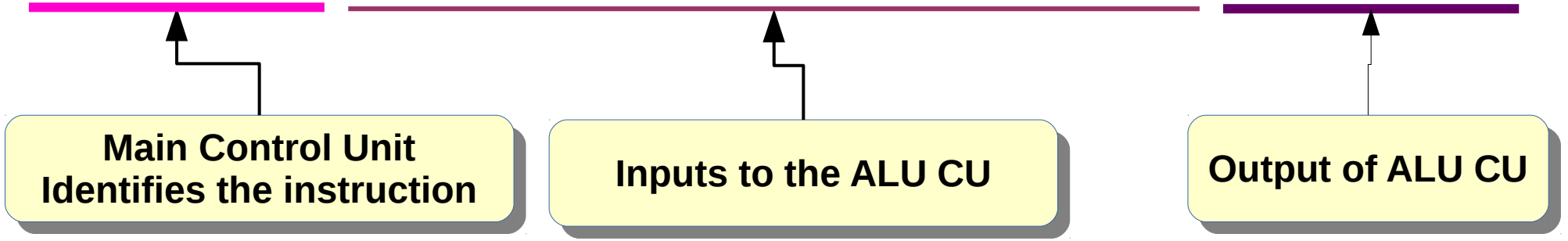
Truth Table of the ALU CU

Instruction	ALU action	Type	Funct field	Control Signal
LW				
SW				
BEQ				
ALU instruction				
ALU Immediate				



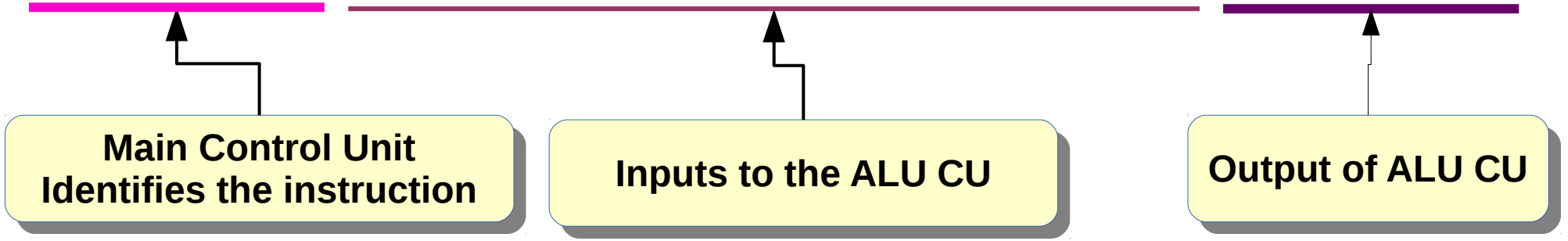
Truth Table of the ALU CU

Instruction	ALU action	Type	Funct field	Control Signal
LW	add	I-type	Invalid	0010
SW				
BEQ				
ALU instruction				
ALU Immediate				



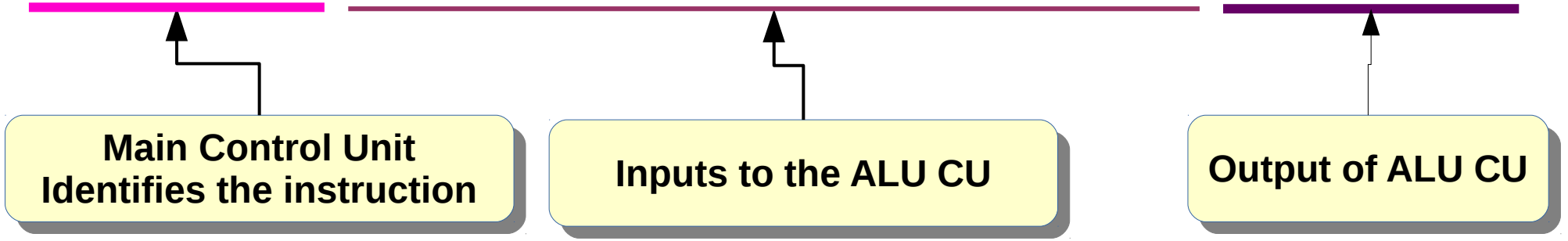
Truth Table of the ALU CU

Instruction	ALU action	Type	Funct field	Control Signal
LW	add	I-type	Invalid	0010
SW	add	I-type	Invalid	0010
BEQ				
ALU instruction				
ALU Immediate				



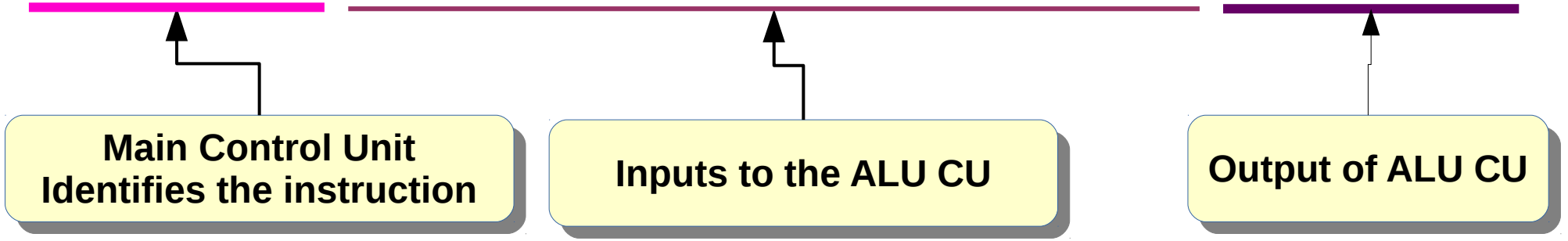
Truth Table of the ALU CU

Instruction	ALU action	Type	Funct field	Control Signal
LW	add	I-type	Invalid	0010
SW	add	I-type	Invalid	0010
BEQ	sub	I-type	Invalid	0110
ALU instruction				
ALU Immediate				



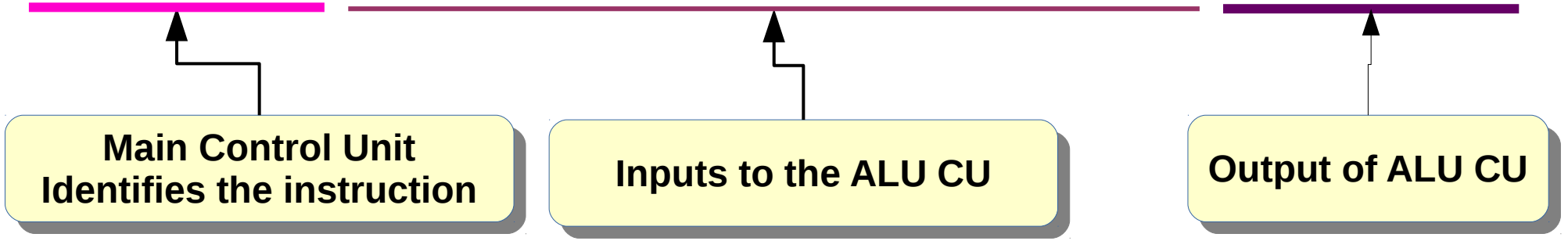
Truth Table of the ALU CU

Instruction	ALU action	Type	Funct field	Control Signal
LW	add	I-type	Invalid	0010
SW	add	I-type	Invalid	0010
BEQ	sub	I-type	Invalid	0110
ALU instruction	As in instruction	R-type	Unique per instruction	As per instruction
ALU Immediate				



Truth Table of the ALU CU

Instruction	ALU action	Type	Funct field	Control Signal
LW	add	I-type	Invalid	0010
SW	add	I-type	Invalid	0010
BEQ	sub	I-type	Invalid	0110
ALU instruction	As in instruction	R-type	Unique per instruction	As per instruction
ALU Immediate	As in instruction	I-type	Invalid	As in instruction



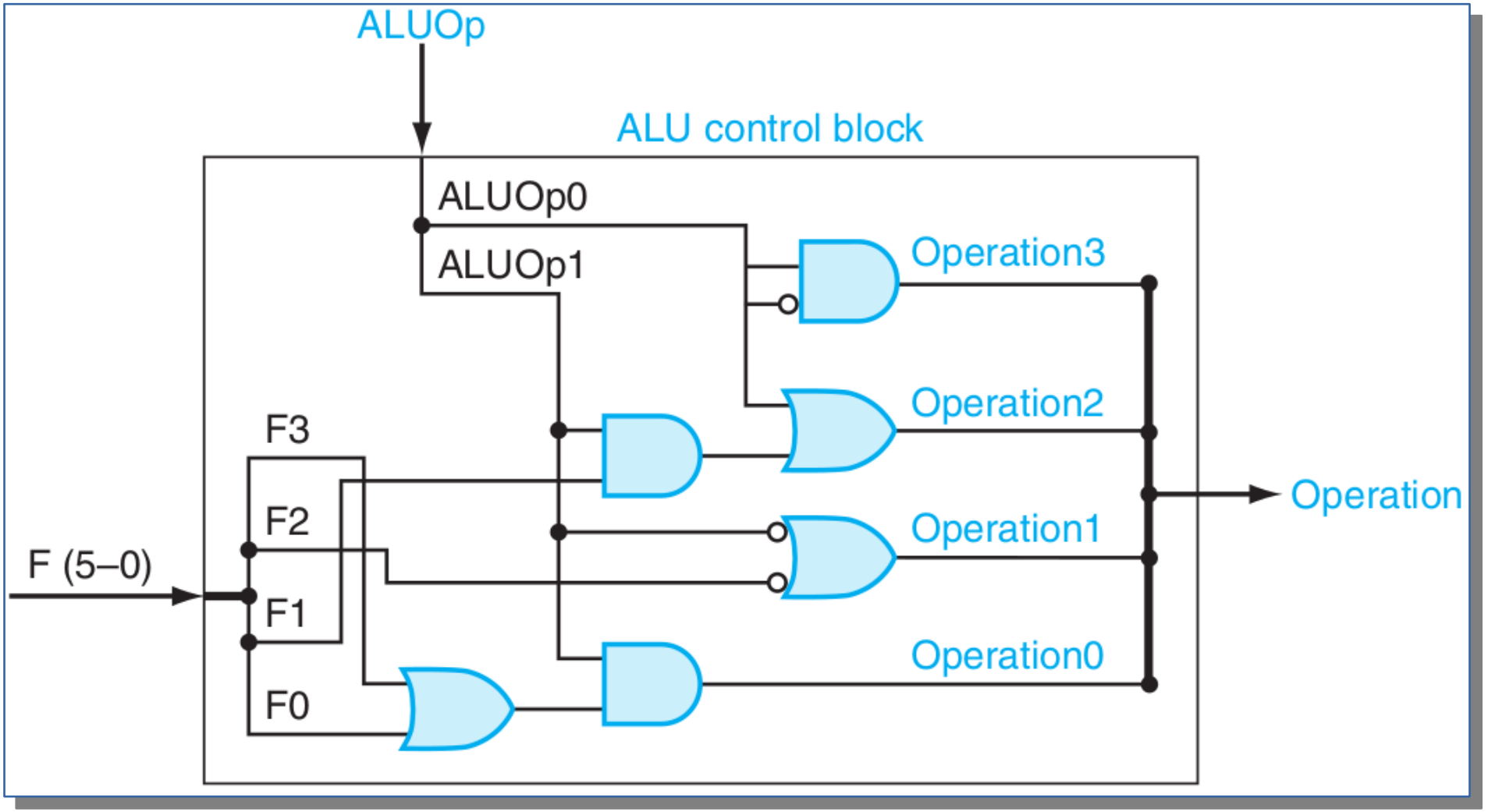
Truth Table of the ALU CU

Instruction opcode	ALUOp	Instruction operation	Funct field	Desired ALU action	ALU control input
LW	00	load word	XXXXXX	add	0010
SW	00	store word	XXXXXX	add	0010
Branch equal	01	branch equal	XXXXXX	subtract	0110
R-type	10	add	100000	add	0010
R-type	10	subtract	100010	subtract	0110
R-type	10	AND	100100	AND	0000
R-type	10	OR	100101	OR	0001
R-type	10	set on less than	101010	set on less than	0111

Truth Table of the ALU CU

ALUOp		Funct field						Operation
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0	
0	0	X	X	X	X	X	X	0010
X	1	X	X	X	X	X	X	0110
1	X	X	X	0	0	0	0	0010
1	X	X	X	0	0	1	0	0110
1	X	X	X	0	1	0	0	0000
1	X	X	X	0	1	0	1	0001
1	X	X	X	1	0	1	0	0111

ALU Control Unit

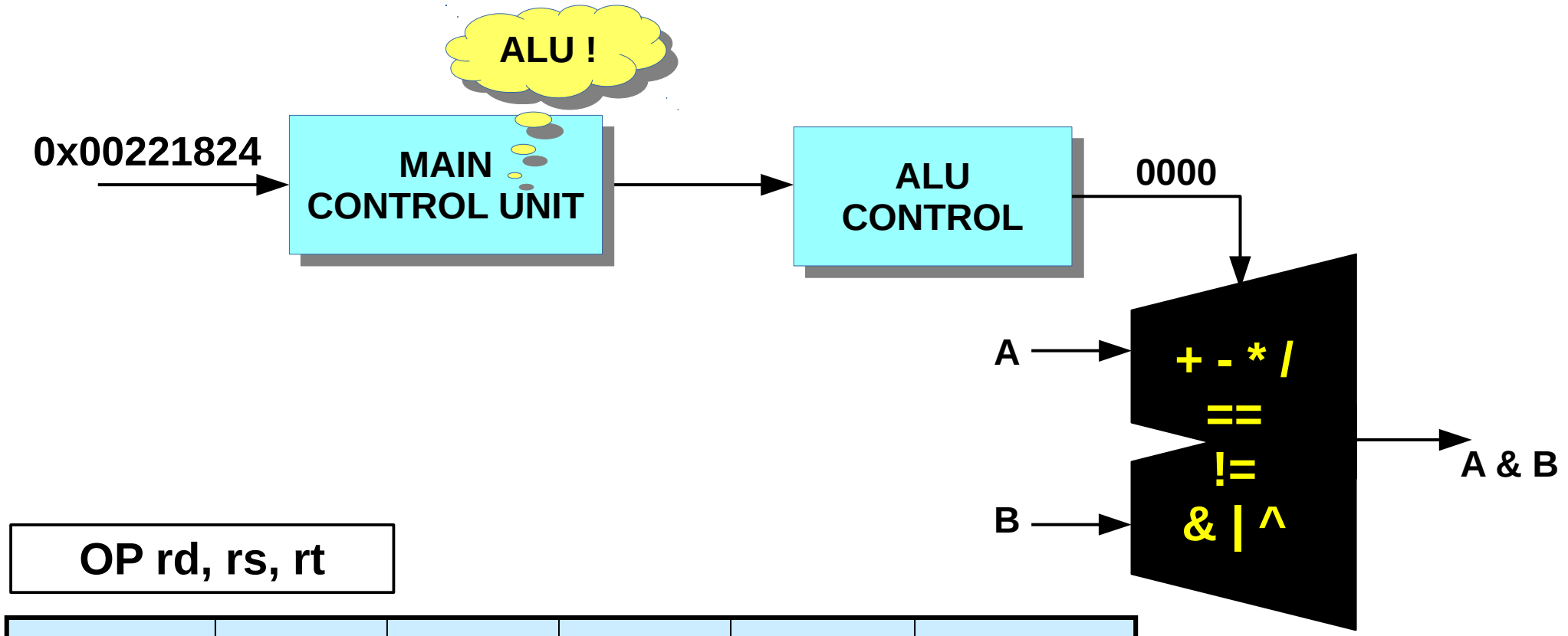


Module Outline

- Integer Arithmetic
  - Adder, Subtractor, Multiplier, Divider
- Arithmetic and Logical Unit Design
  - ALU Design in SystemC

Backup

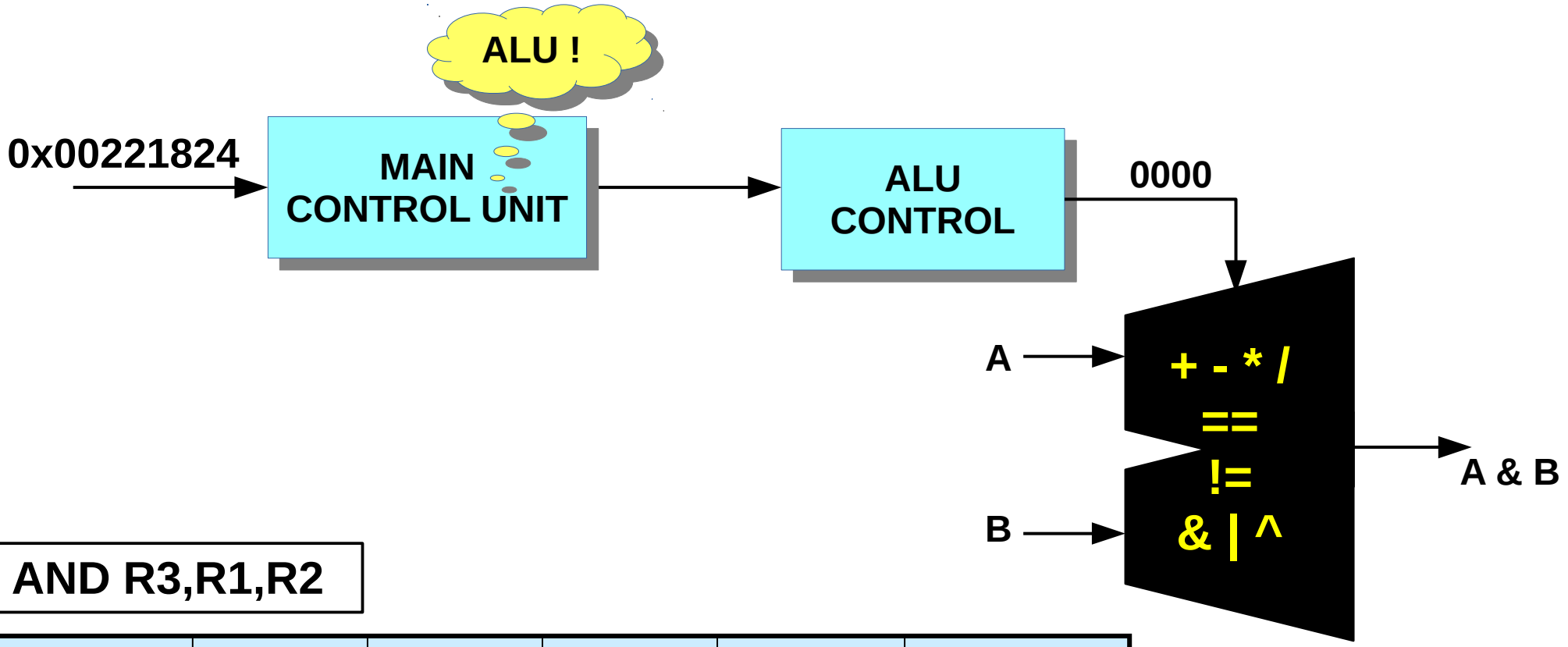
ALU Control – AND Instruction



OP rd, rs, rt

op	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

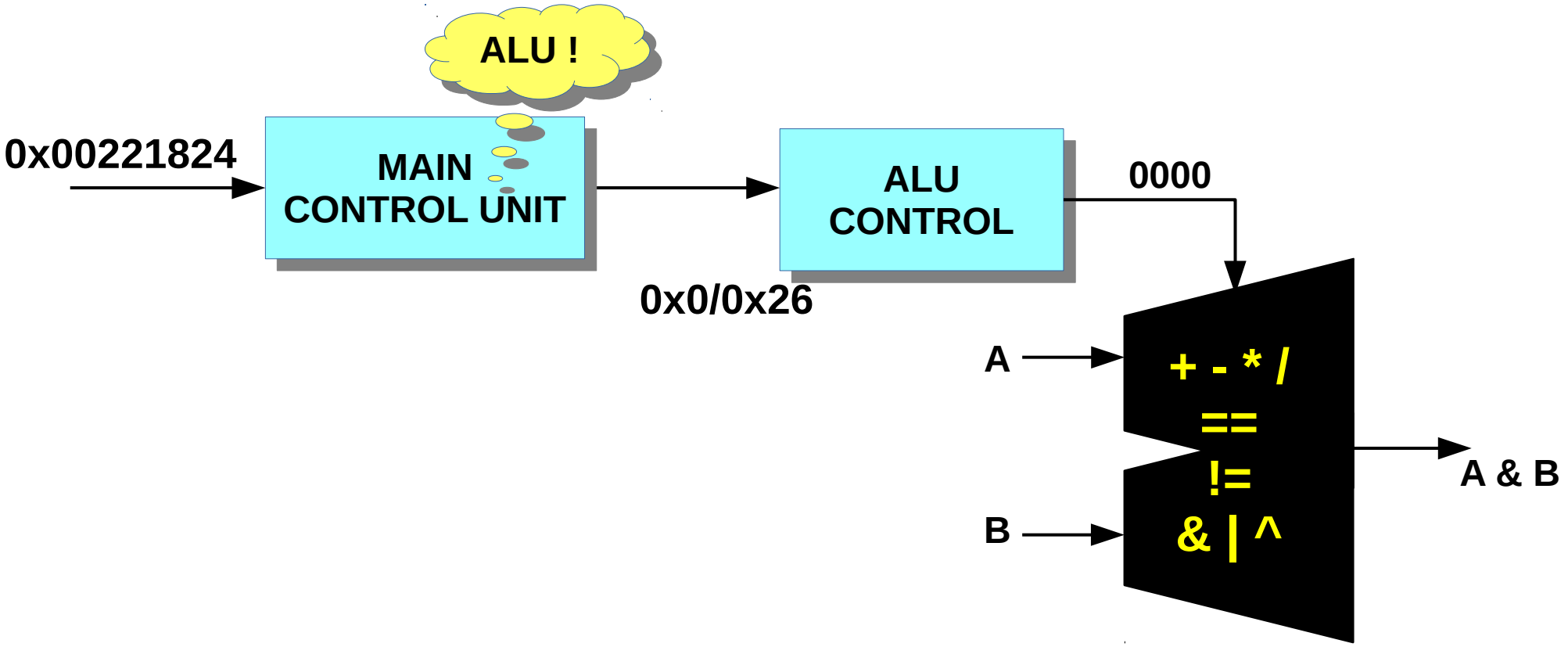
ALU Control – AND Instruction



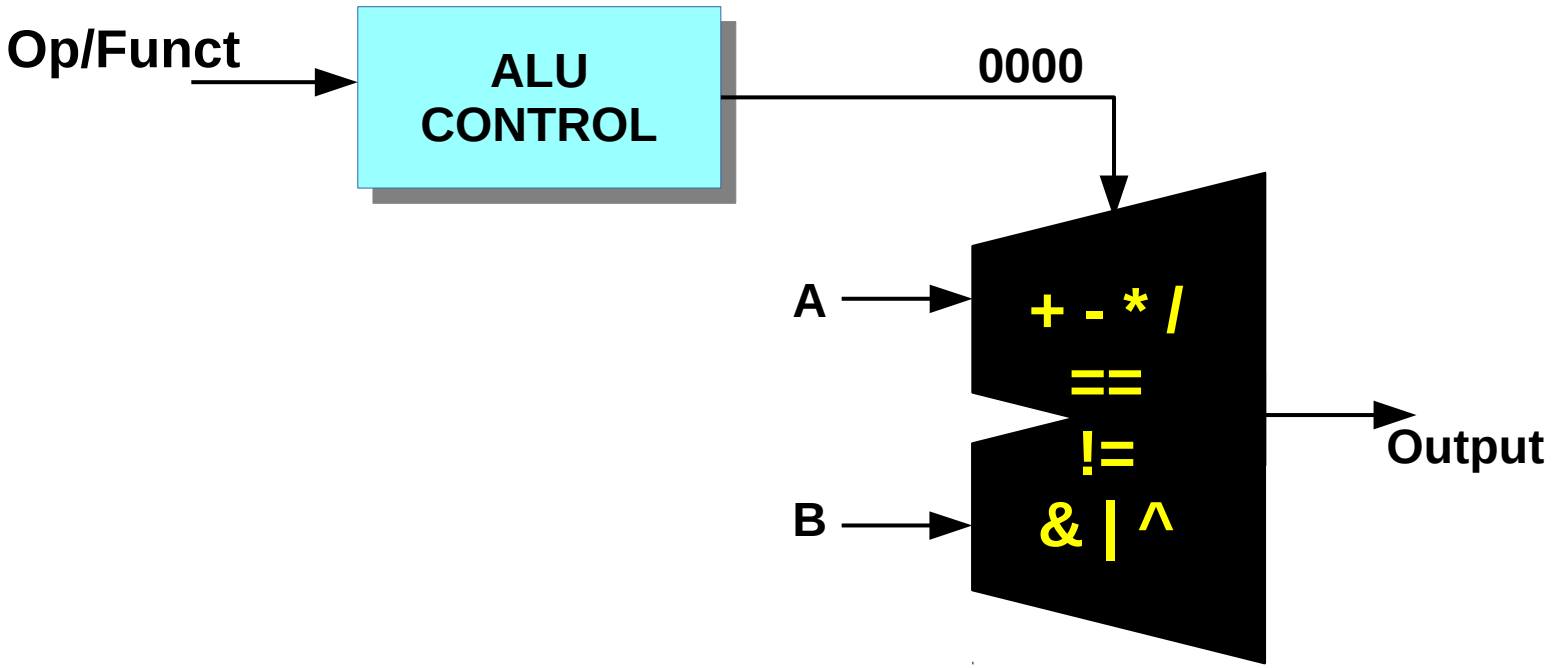
AND R3,R1,R2

0x0	1	2	3	0	0x26
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

ALU Control – AND Instruction



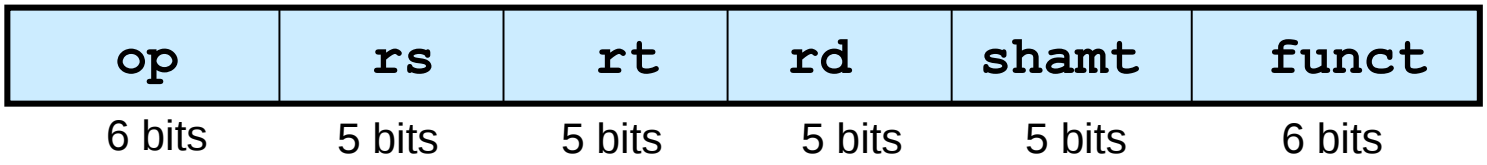
ALU Control



ALU Instructions

- ALU instructions (R type), Memory Transfer (effective address calculation), Branches (BEQ)

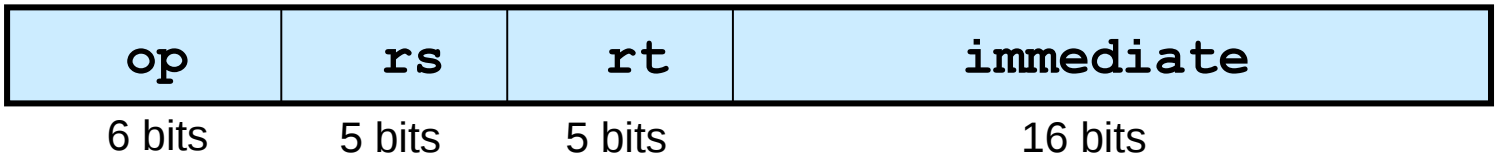
- R-type



OP rd, rs, rt

op: Opcode (class of instruction). Eg. ALU
funct: Which subunit of the ALU to activate?

- I-type



OP rt, rs, IMM

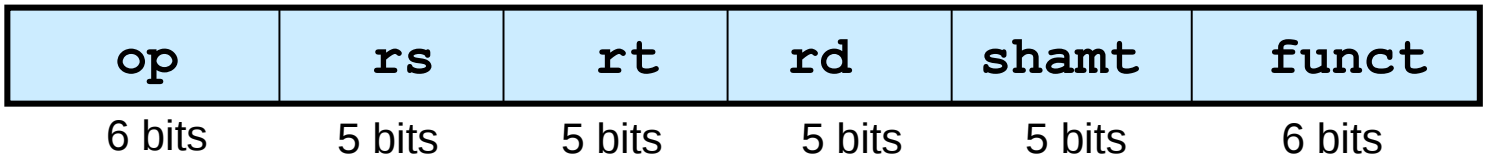
ALU Instructions

- ALU instructions (R type), Memory Transfer (effective address calculation), Branches (BEQ)
- Identified by Opcode fields and Funct fields

ALU Instructions

- ALU instructions (R type), Memory Transfer (effective address calculation), Branches (BEQ)

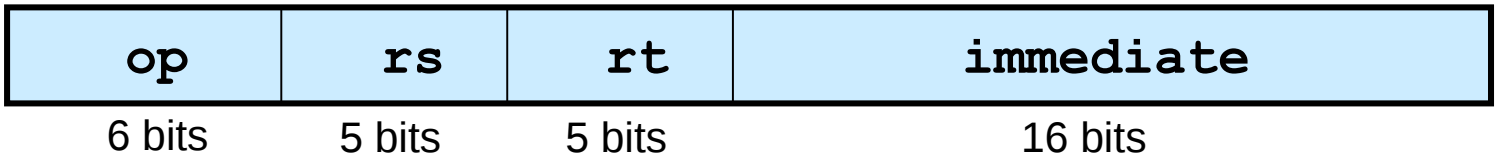
- R-type



OP rd, rs, rt

op: Opcode (class of instruction). Eg. ALU
funct: Which subunit of the ALU to activate?

- I-type



OP rt, rs, IMM

ALU Control – LW (I-type)