CS / EE 320
Computer Organization and Assembly Language
Spring 2024
Lecture 13

Shahid Masud

Topics: Topics: Introduction to CPU Design – Simple 1 Bit ALU

Topics

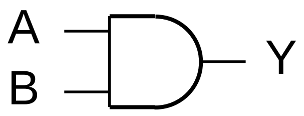
- Start Chapter 4 of P&H Textbook
- Some material in Appendix B of P&H Textbook, search Internet
- Digital Elements that Preserve States (D Flip Flops)
- CPU visualization as Combinational Element placed between two state elements, the Data path and Control path
- Develop a 1 Bit ALU and scale up to 32 Bits
- Basic Operations of ALU, AND,OR, ADD, SUB, NAND, NOR, Zero Detect, Compare, Overflow Detect
- QUIZ 3 TODAY

Simple Processor Design

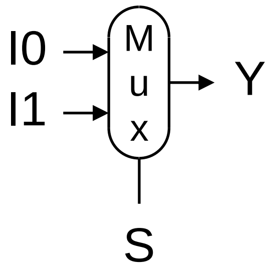
- MIPS subset for implementation
- Design overview
- Division into data path and control
- Building blocks - combinational and sequential
- Clock and timings
- Components required for MIPS subset

Combinational Elements

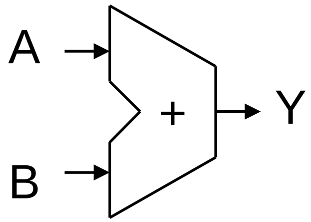
- AND-gate
  - Y = A & B



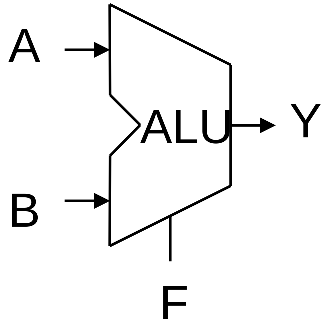
- Multiplexer
  - Y = S ? I1 : I0



- Adder
  - Y = A + B



- Arithmetic/Logic Unit
  - Y = F(A, B)



Building Blocks for 1 Bit ALU

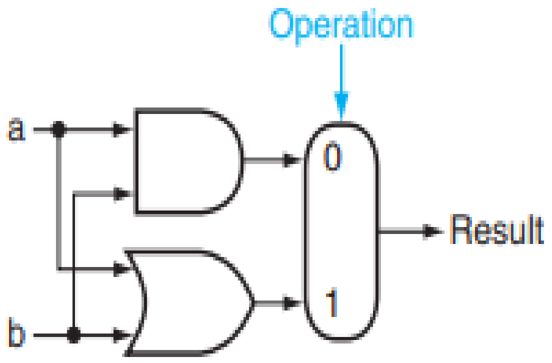


FIGURE B.5.1 The 1-bit logical unit for AND and OR.

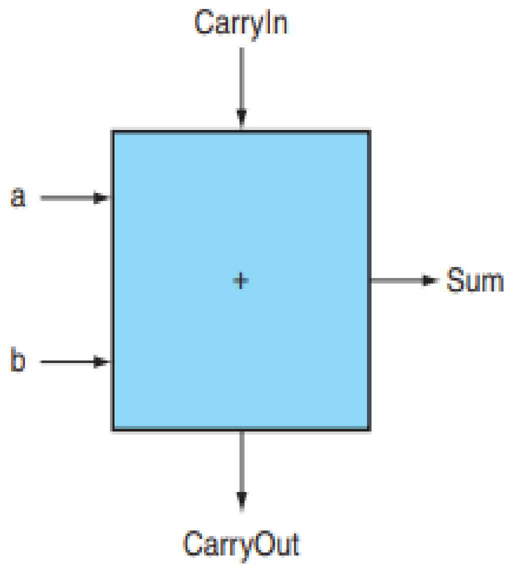


FIGURE B.5.2 A 1-bit adder. This adder is called a full adder; it is also called a (3,2) adder because it has 3 inputs and 2 outputs. An adder with only the a and b inputs is called a (2,2) adder or half-adder.

Carry Out and simple 1 Bit ALU

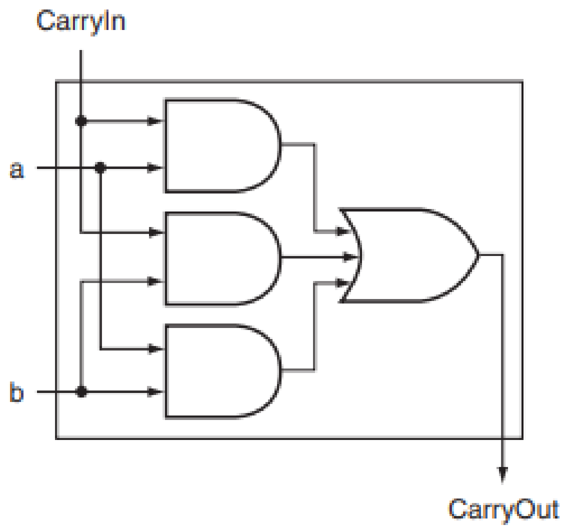


FIGURE B.5.5 Adder hardware for the CarryOut signal. The rest of the adder hardware is the logic for the Sum output given in the equation on this page.

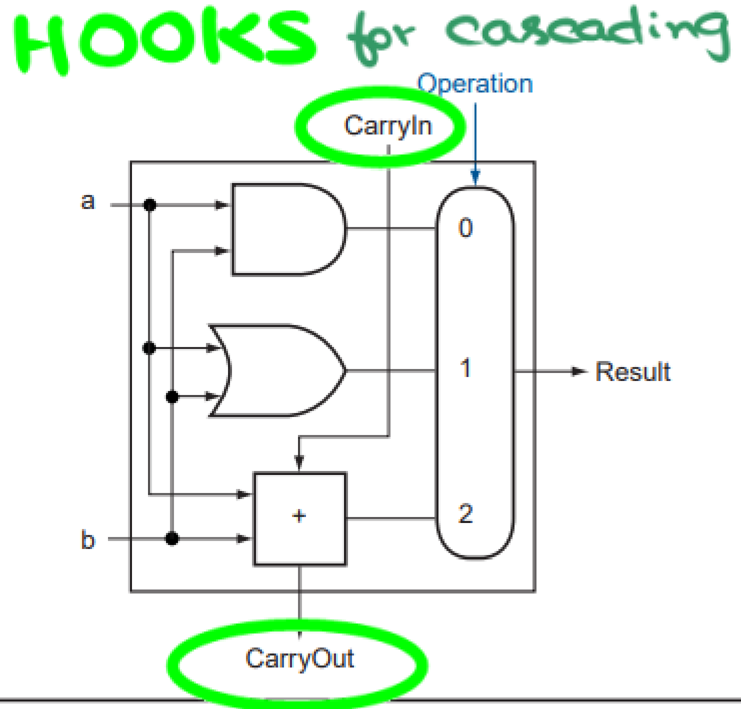


FIGURE B.5.6 A 1-bit ALU that performs AND, OR, and addition (see Figure B.5.5).

Simple 1 Bit ALU

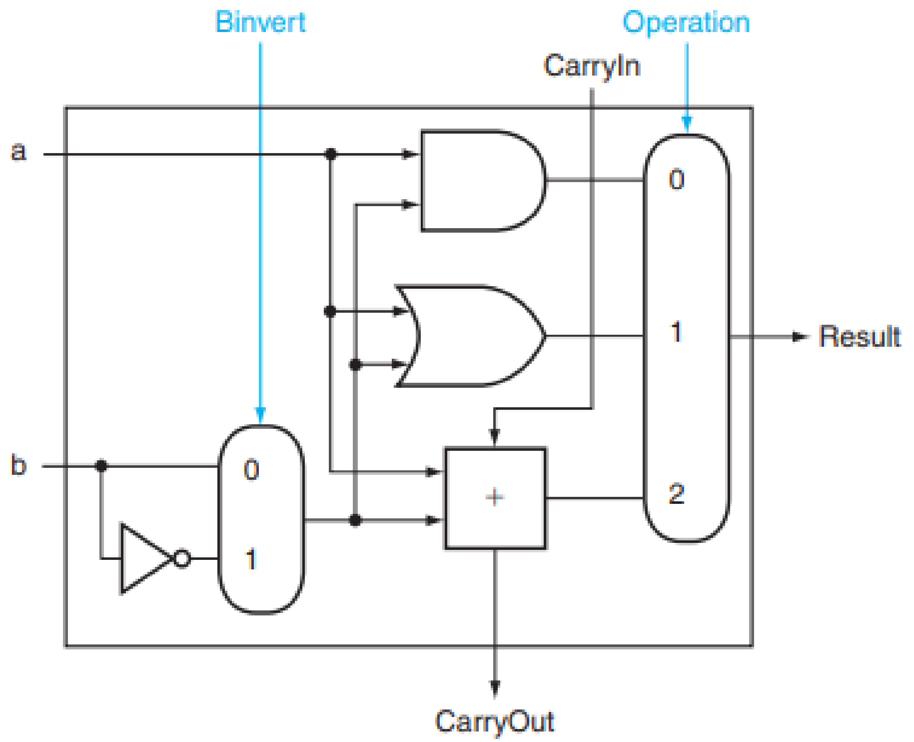


FIGURE B.5.8 A 1-bit ALU that performs AND, OR, and addition on a and b or a and $\bar{b}$. By selecting $\bar{b}$ (Binvert = 1) and setting CarryIn to 1 in the least significant bit of the ALU, we get two's complement subtraction of b from a instead of addition of b to a.

Some more functions in 1 Bit ALU

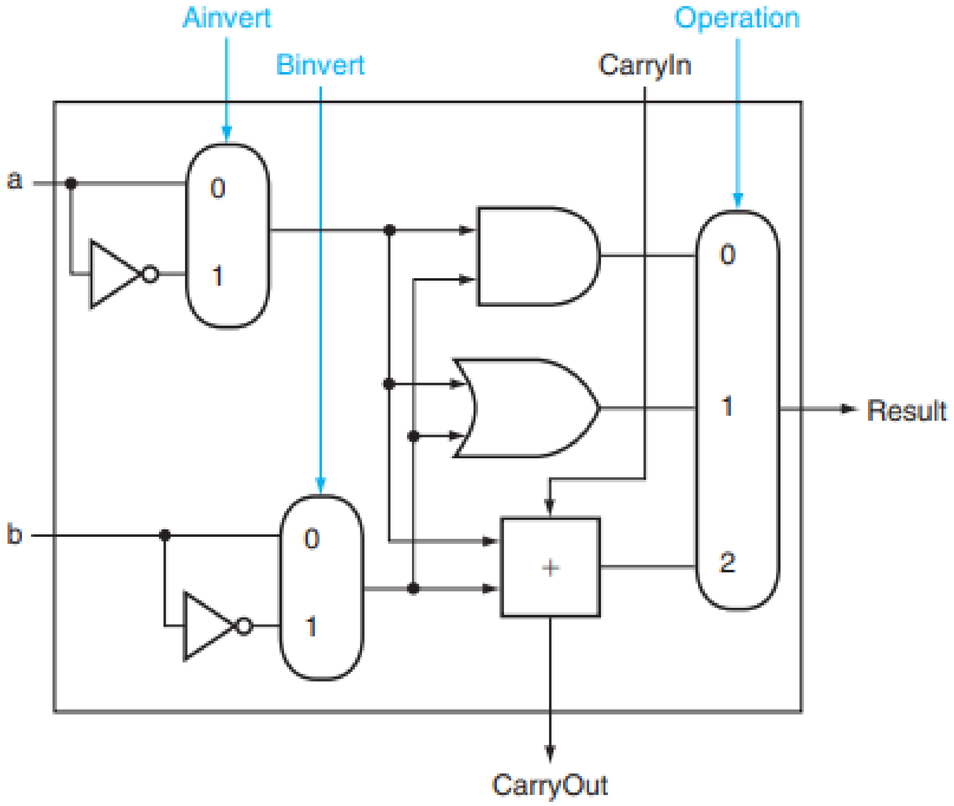
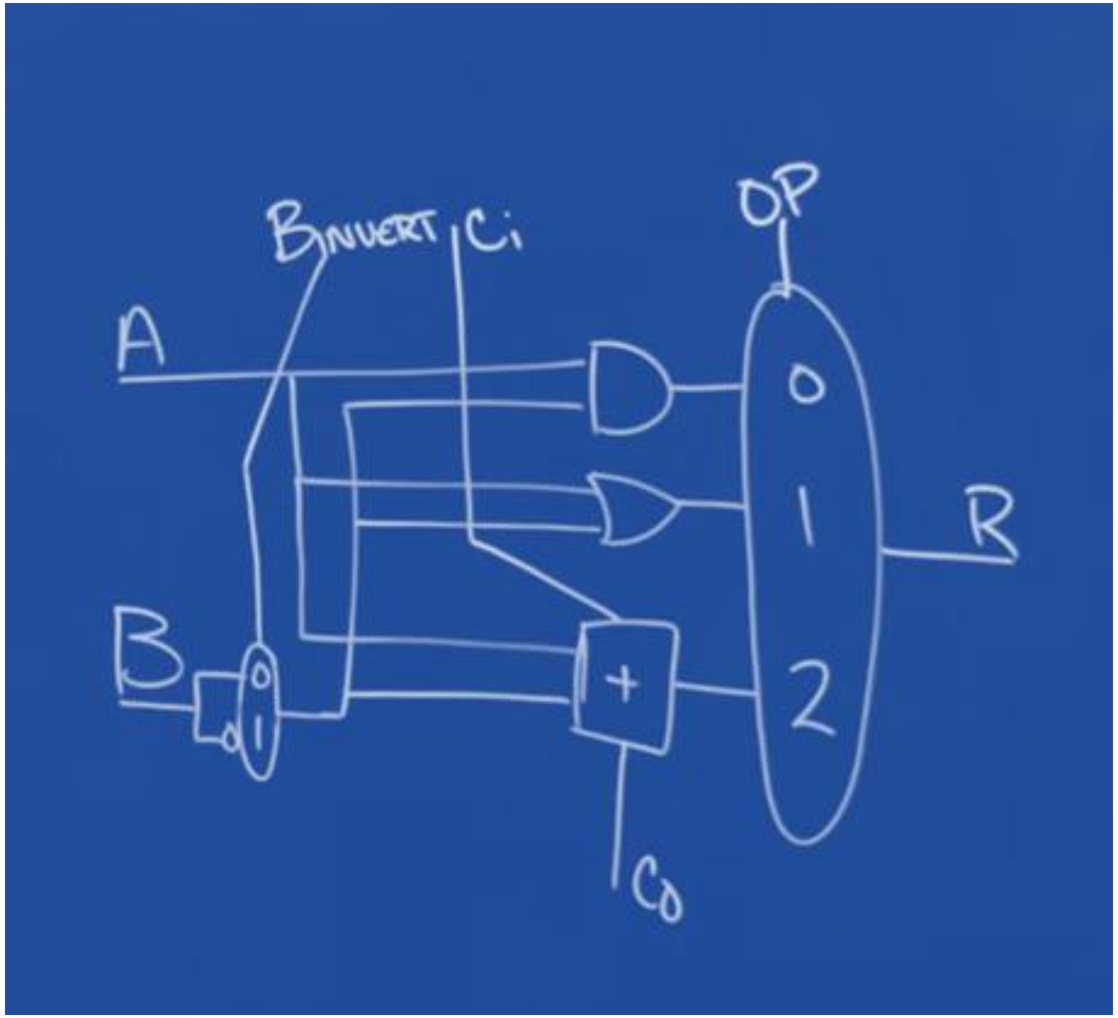


FIGURE B.5.9 A 1-bit ALU that performs AND, OR, and addition on a and b or $\bar{a}$ and $\bar{b}$. By selecting $\bar{a}$ (Ainvert = 1) and $\bar{b}$ (Binvert = 1), we get a NOR b instead of a AND b.

Developing Adder and Subtractor

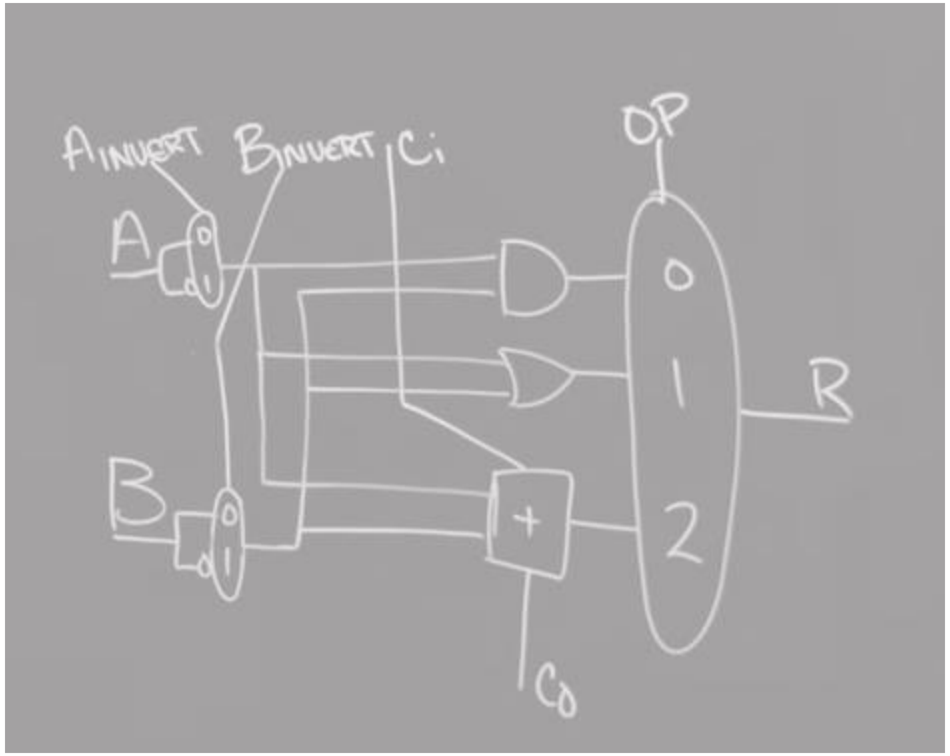
AND
OR
ADD
SUB

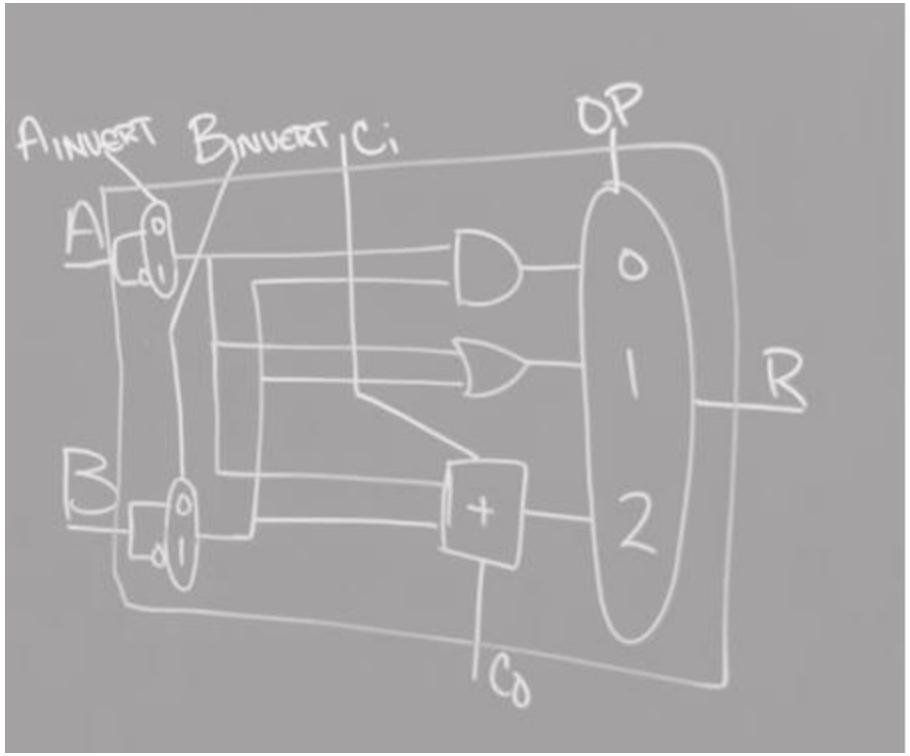


NAND and NOR added to 1 bit ALU

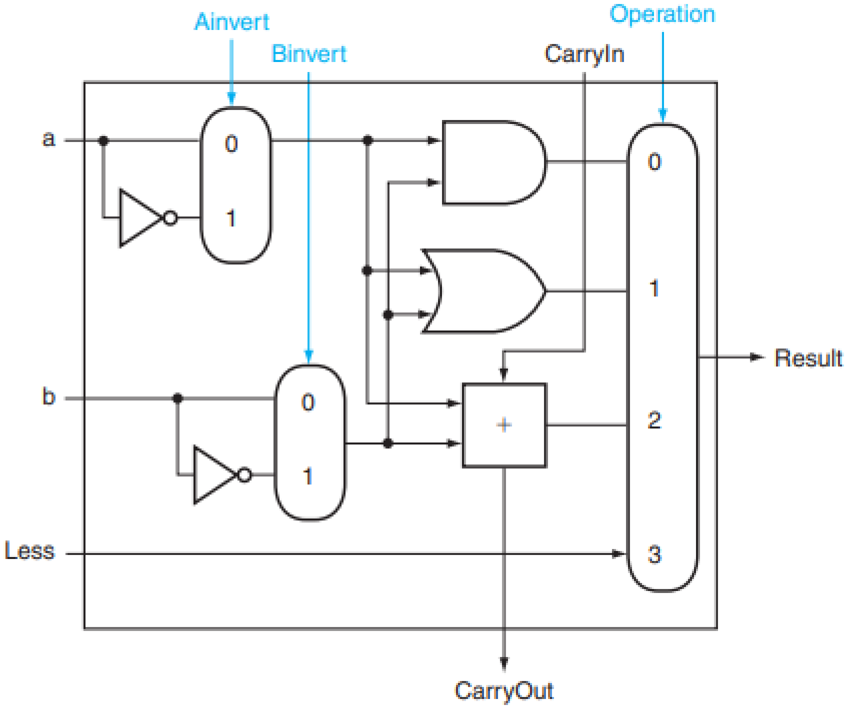
AND
OR
ADD
SUB

NAND
NOR





1 Bit ALU with Overflow



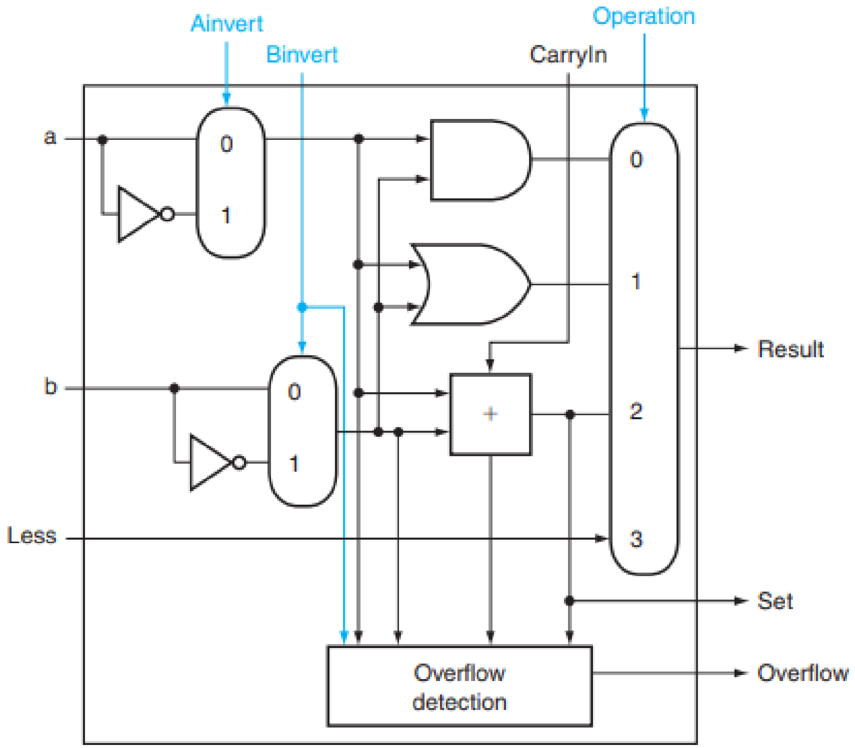


FIGURE B.5.10 (Top) A 1-bit ALU that performs AND, OR, and addition on a and b or $\bar{b}$, and (bottom) a 1-bit ALU for the most significant bit. The top drawing includes a direct input that is connected to perform the set on less than operation (see Figure B.5.11); the bottom has a direct output from the adder for the less than comparison called Set. (See Exercise B.24 at the end of this appendix to see how to calculate overflow with fewer inputs.)

Symbol of 1 Bit ALU

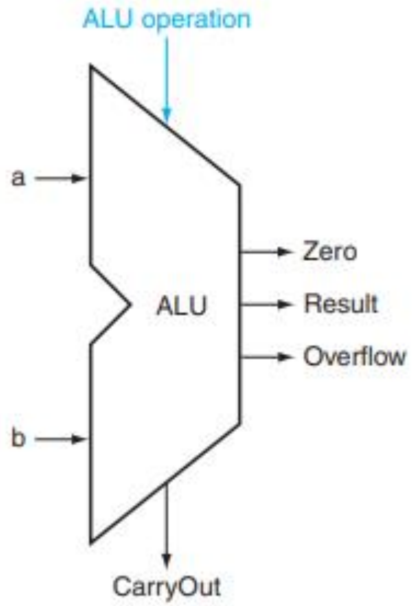
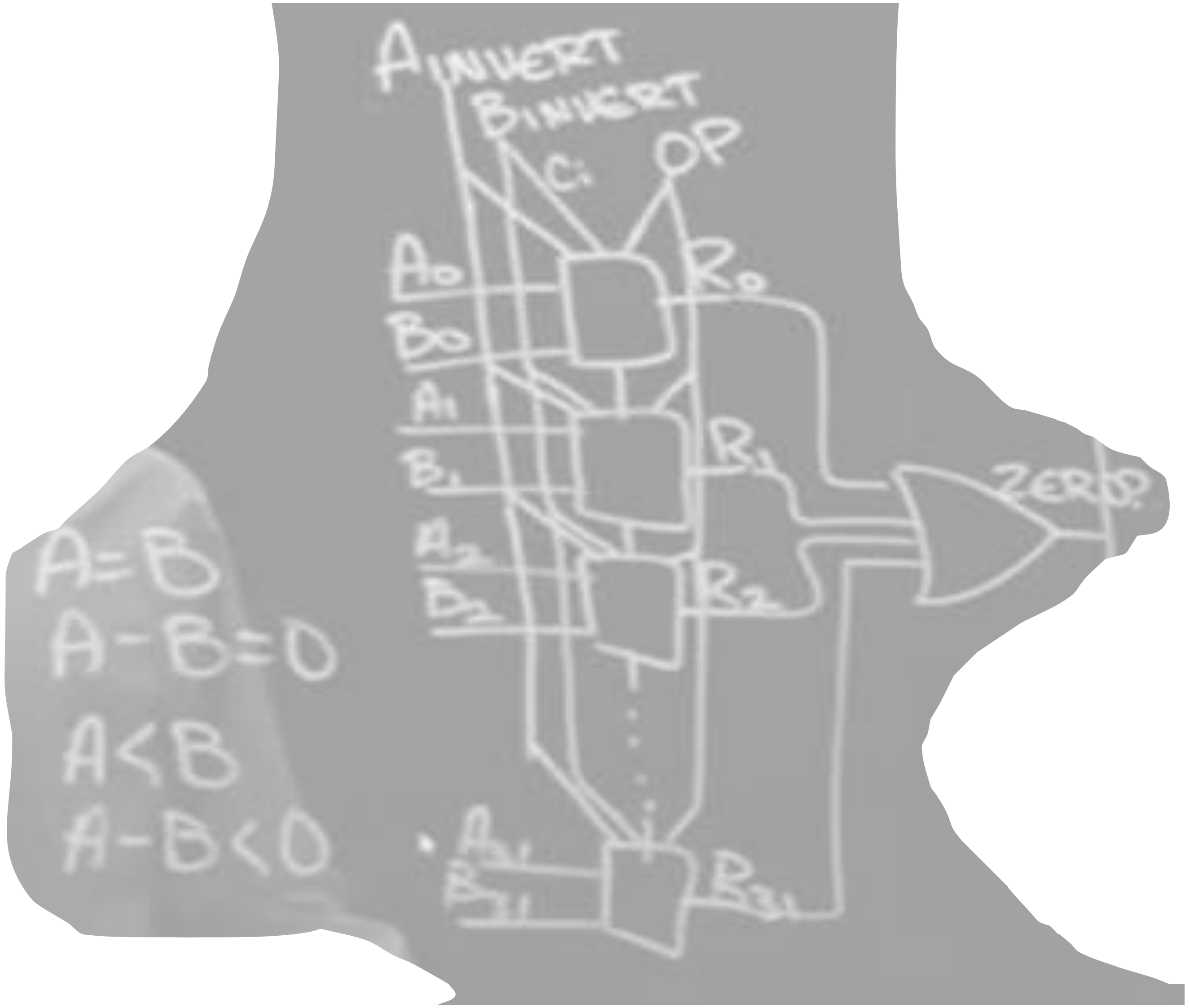


FIGURE B.5.14 The symbol commonly used to represent an ALU, as shown in Figure B.5.12. This symbol is also used to represent an adder, so it is normally labeled either with ALU or Adder.

Repeat 32 ALU to make a 32 Bit ALU

Adding Compare Instructions

Detect Zero
Use Adder and Subtractor
To Determine Sign of MSB.
This gives indication of
what number is bigger



‘Less’ than Instruction

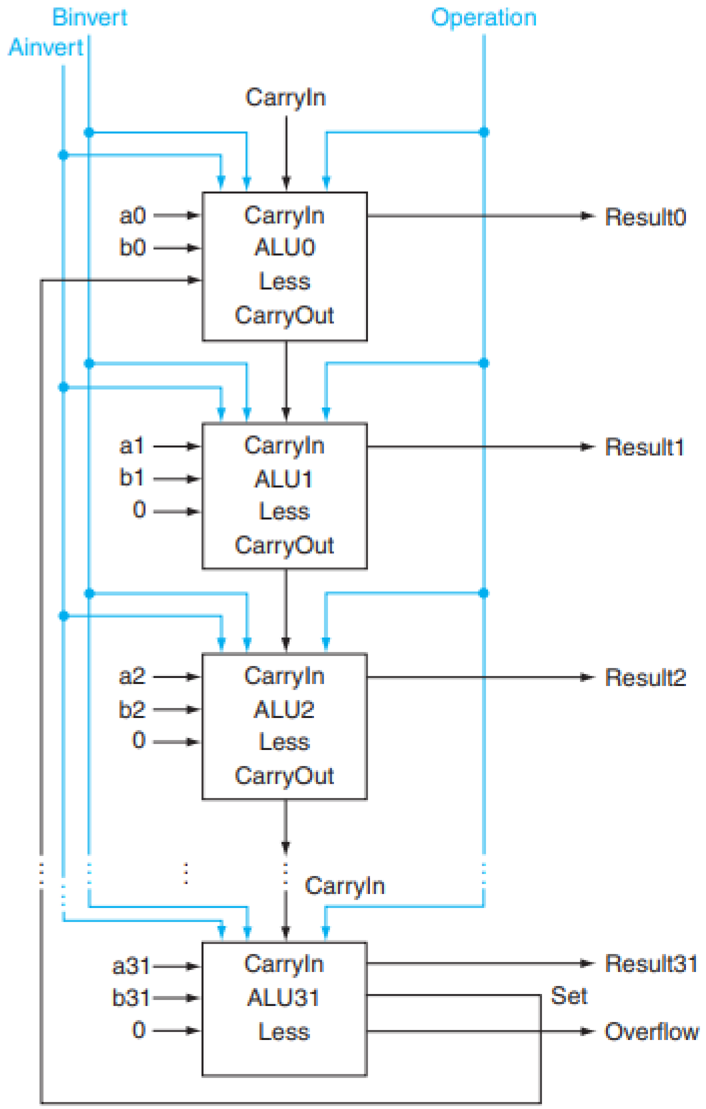


FIGURE B.5.11 A 32-bit ALU constructed from the 31 copies of the 1-bit ALU in the top of Figure B.5.10 and one 1-bit ALU in the bottom of that figure. The Less inputs are connected to 0 except for the least significant bit, which is connected to the Set output of the most significant bit. If the ALU performs a − b and we select the input 3 in the multiplexor in Figure B.5.10, then Result = 0 … 001 if a < b, and Result = 0 … 000 otherwise.

Final ALU

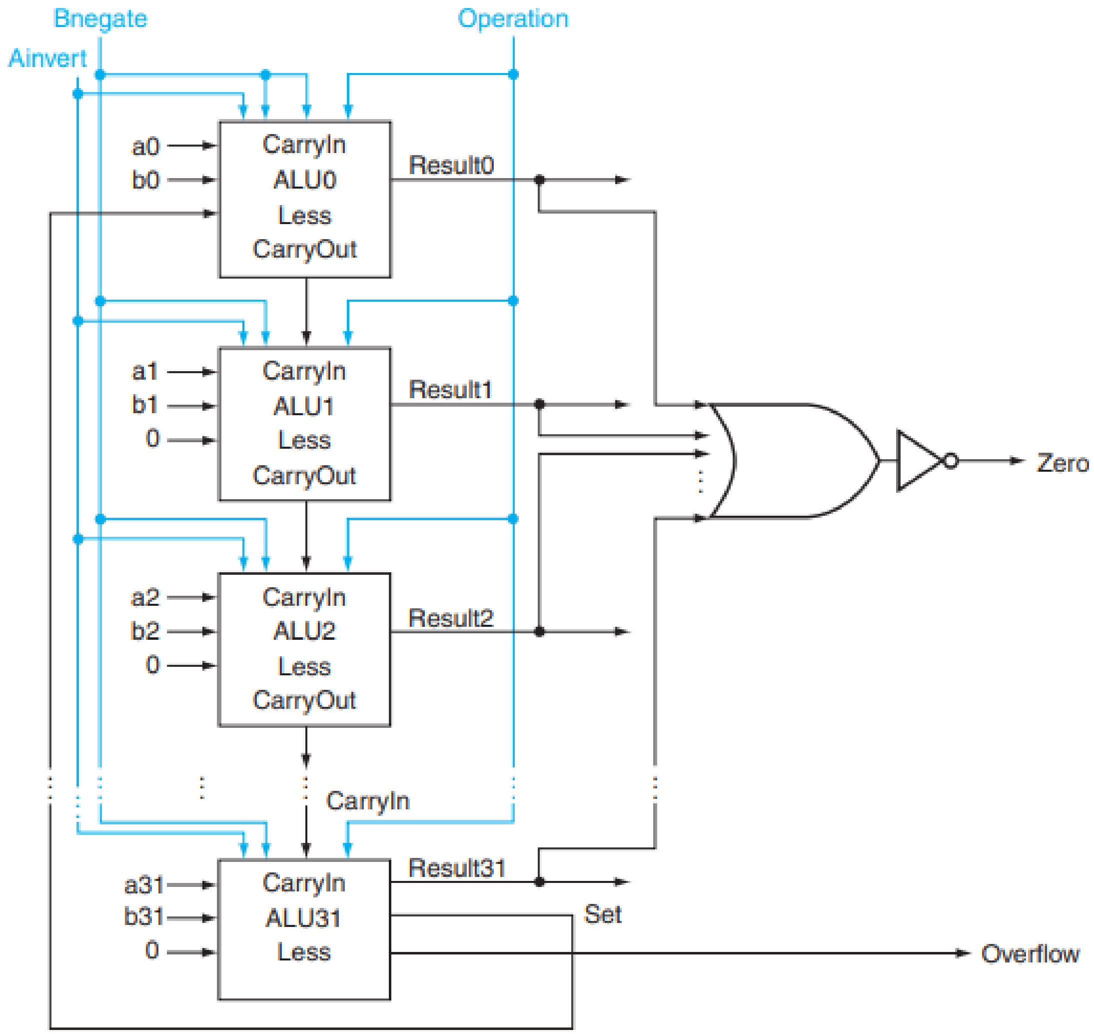


FIGURE B.5.12 The final 32-bit ALU. This adds a Zero detector to Figure B.5.11.

ALU control lines	Function
0000	AND
0001	OR
0010	add
0110	subtract
0111	set on less than
1100	NOR

FIGURE B.5.13 The values of the three ALU control lines, Bnegate, and Operation, and the corresponding ALU operations.

32 Bit Adder and Carry Chain

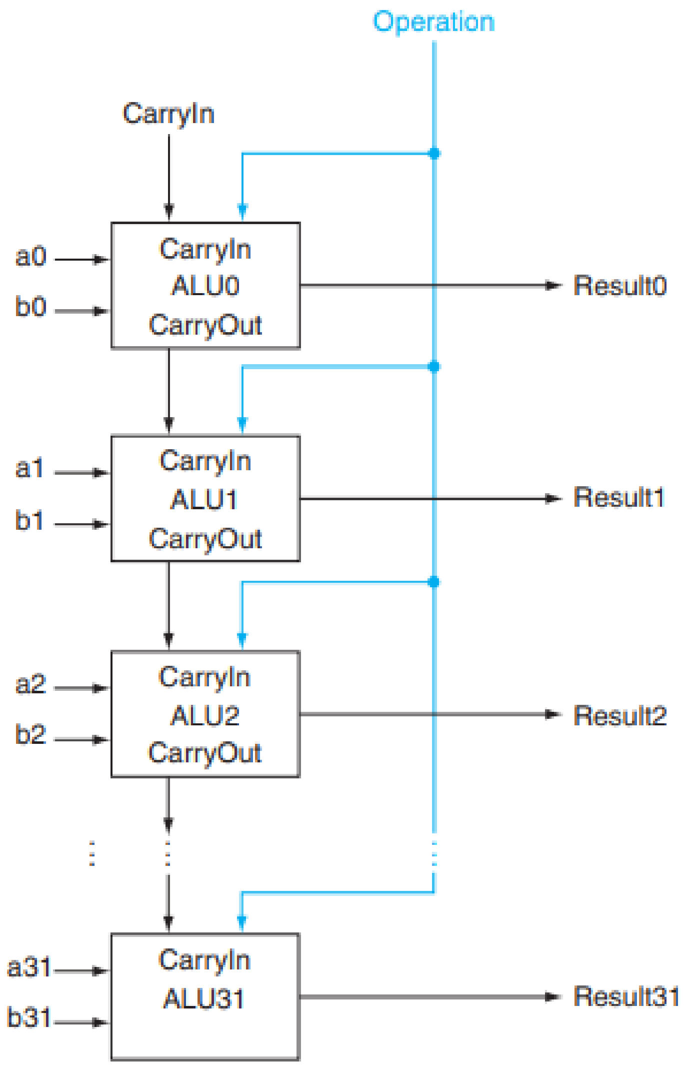


FIGURE B.5.7 A 32-bit ALU constructed from 32 1-bit ALUs. CarryOut of the less significant bit is connected to the CarryIn of the more significant bit. This organization is called ripple carry.

Nor and Nand Instructions

A MIPS ALU also needs a NOR function. Instead of adding a separate gate for NOR, we can reuse much of the hardware already in the ALU, like we did for subtract. The insight comes from the following truth about NOR:

DeMorgan's theorem

$$\boxed{\overline{(a + b)} = \bar{a} \cdot \bar{b}}$$

That is, NOT (a OR b) is equivalent to NOT a AND NOT b. This fact is called DeMorgan's theorem and is explored in the exercises in more depth.

Since we have AND and NOT b, we only need to add NOT a to the ALU. Figure B.5.9 shows that change.

$\equiv$

$R = \overline{a+b}$

$R = \bar{a} \cdot \bar{b}$

NOR

$\equiv$

$R = \overline{a \cdot b}$

NAND

Zero detector Instruction

$$\text{Zero} = \overline{(\text{Result31} + \text{Result30} + \ldots + \text{Result2} + \text{Result1} + \text{Result0})}$$

Figure B.5.12 shows the revised 32-bit ALU. We can think of the combination of the 1-bit Ainvert line, the 1-bit Binvert line, and the 2-bit Operation lines as 4-bit control lines for the ALU, telling it to perform add, subtract, AND, OR, or set on less than. Figure B.5.13 shows the ALU control lines and the corresponding ALU operation.

Less 0

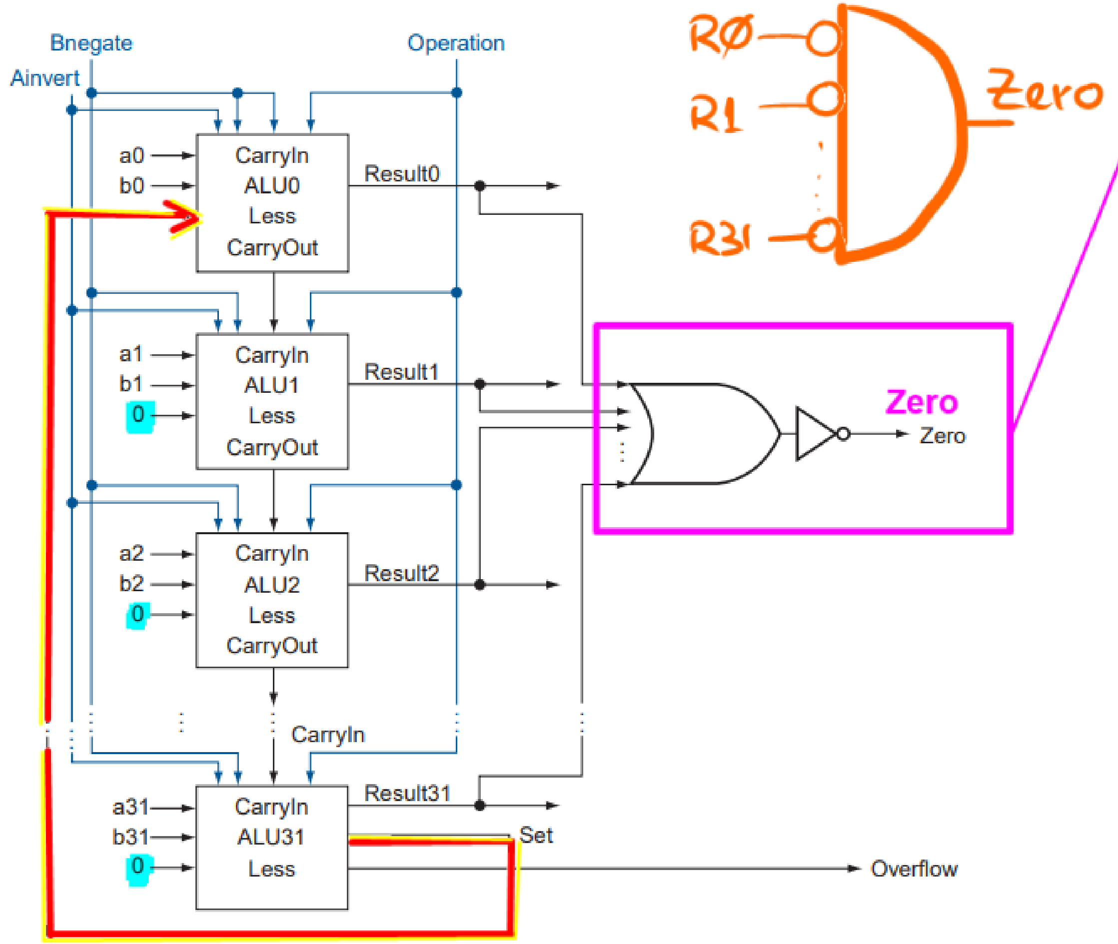
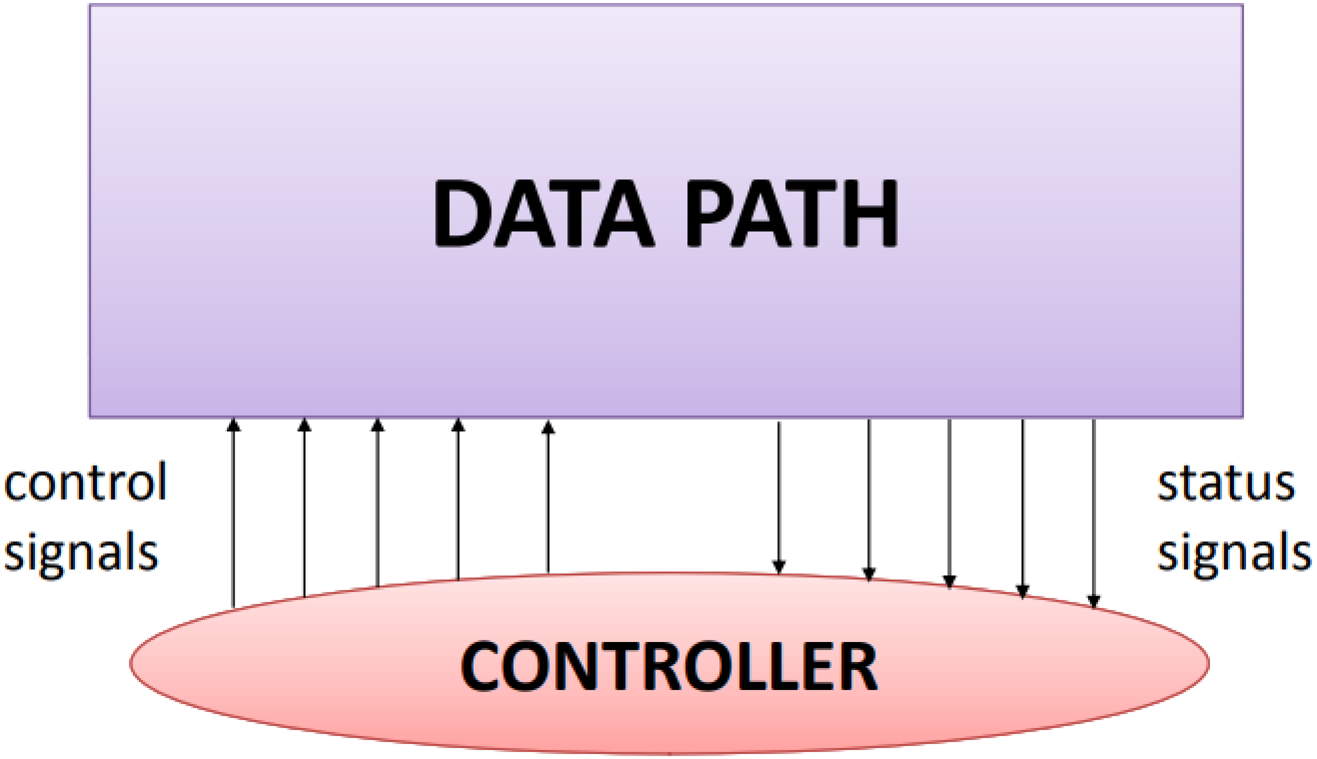


FIGURE B.5.12 The final 32-bit ALU. This adds a Zero detector to Figure B.5.11.

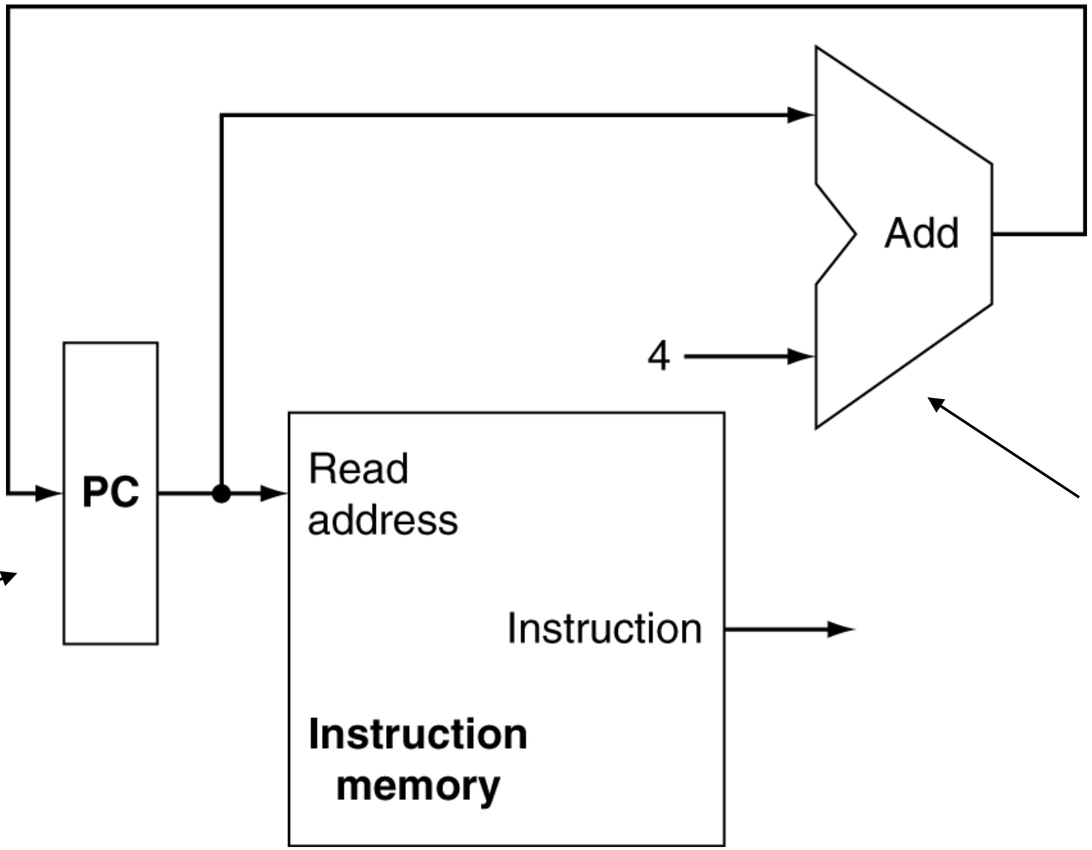
Data path and Control path in a CPU



Building a Datapath

- Datapath
  - Elements that process data and addresses in the CPU
    - Registers, ALUs, mux’s, memories, …
- We will build a MIPS datapath incrementally
  - Refining the overview design

Instruction Fetch



32-bit register

Increment by 4 for next instruction

Readings

- Chap 4 of P&H Textbook
- Appendix B of P&H Textbook