Introduction

Digital systems (logic design, digital logic, switching circuits) are employed in:

- computers
- data communication
- control systems
- many other fields

Things you'll be learning:

- how computers work at the gate level (the lowest level to discuss without an engineering background)
- how to build a simple CPU (central processing unit)

Why learn this stuff?

- you want to call yourself a “computer scientist”
- you need to make a purchasing decision or offer “expert” advice

Computers

What is a computer?

- input (mouse, keyboard)
- output (display, printer)
- memory (disk drives, DRAM, SRAM, CD)
- network

Impossible to understand by looking at each transistor.

We need ...

Levels of abstraction

Software:

- Application
- Operating system
- Firmware

Instruction set architecture:

- Data type and structures: encodings and machine representation
- Instruction set
- Instruction formats
- Addressing modes and accessing data and instructions

Hardware:

- Instruction set processing
- I/O System
- Digital design
- Circuit design
- Layout

Example: from instructions to gates

Programming level

C:

- A = B + C

Assembler:

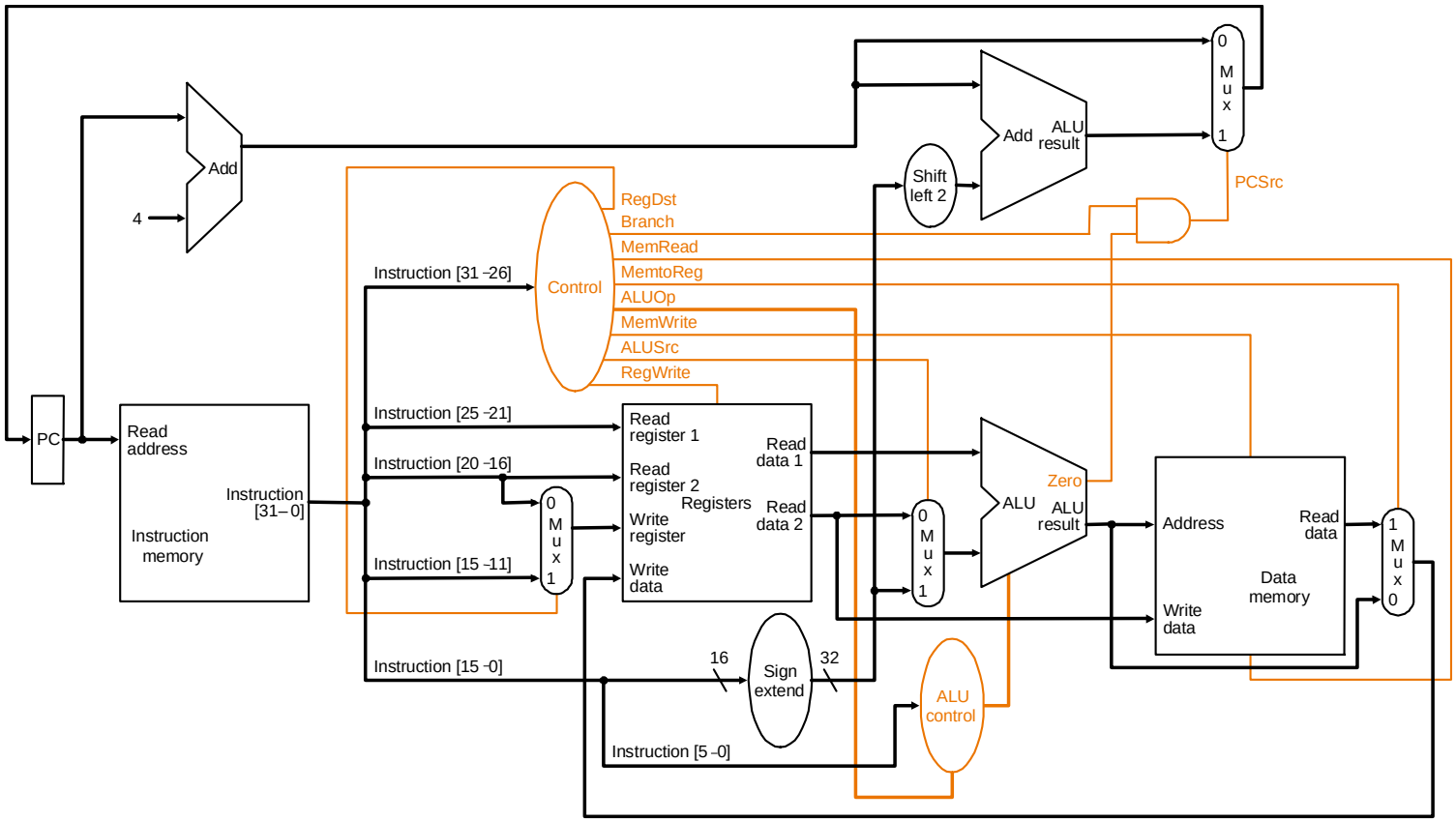
- add $t0, $s1, $s2

Machine instruction:

- decimal: 0 17 18 8 0 32
- binary: 000000 10001 10010 01000 00000 100000

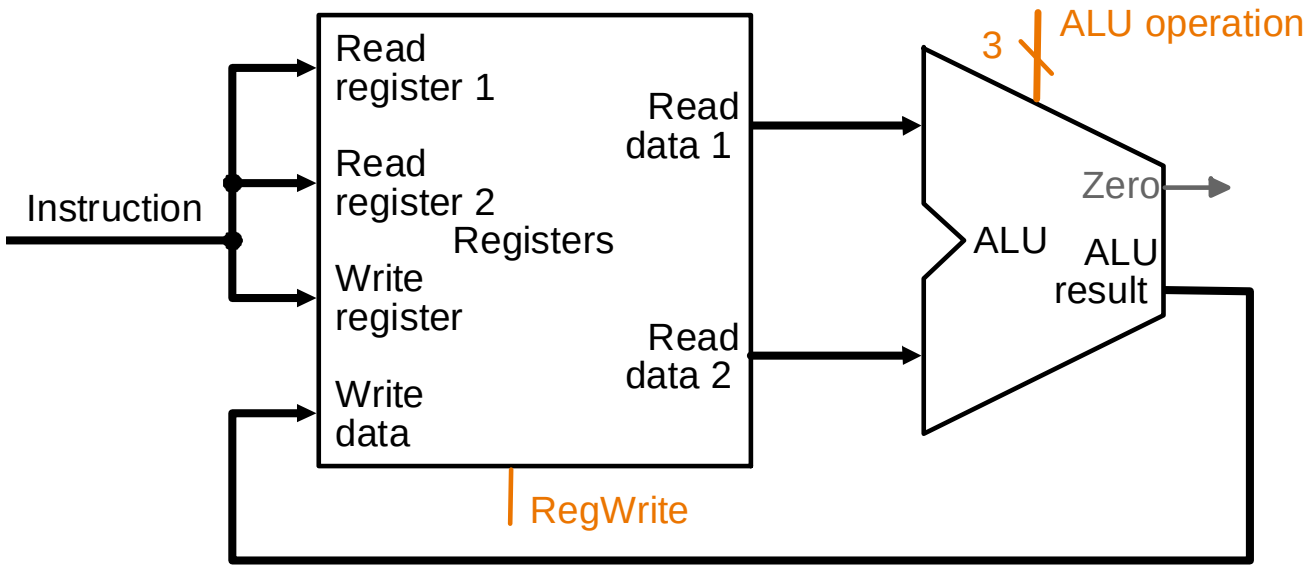
Example: from instructions to gates

Datapath and control



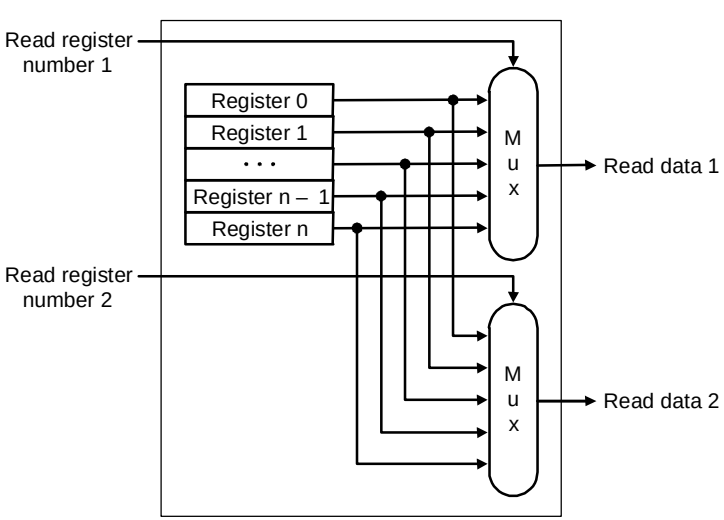
Example: from instructions to gates

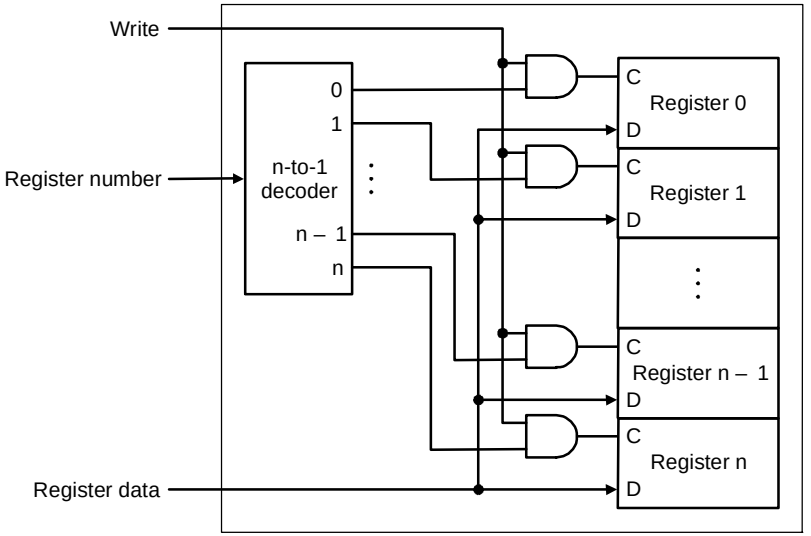
Register file and ALU



Example: from instructions to gates

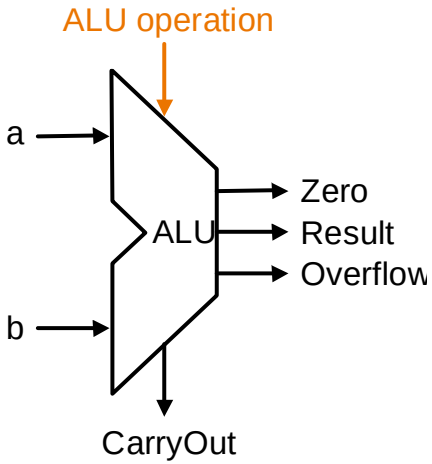
Inside register file

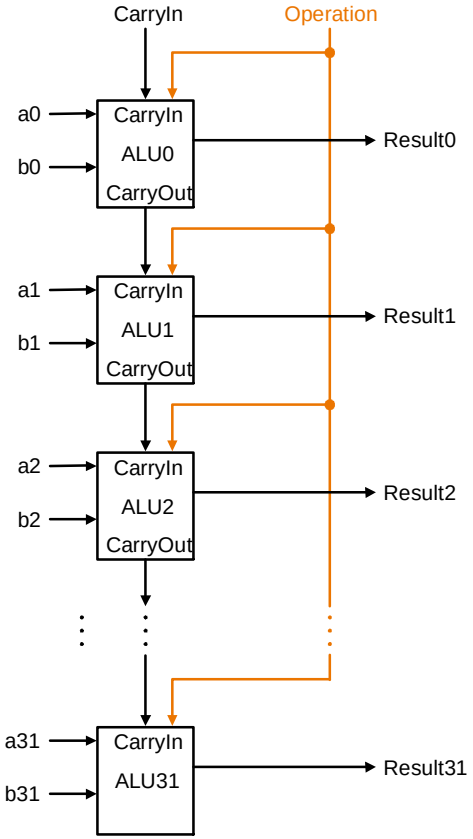


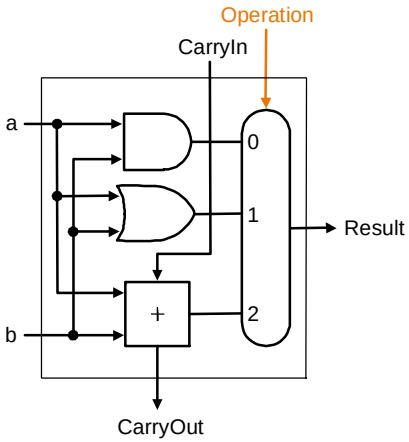


Example: from instructions to gates

Arithmetic Logic Unit (ALU)

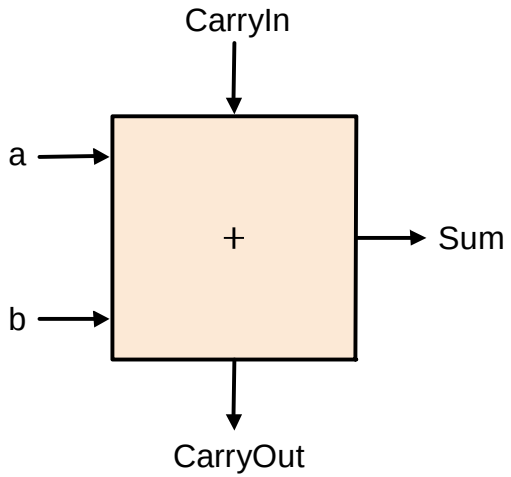




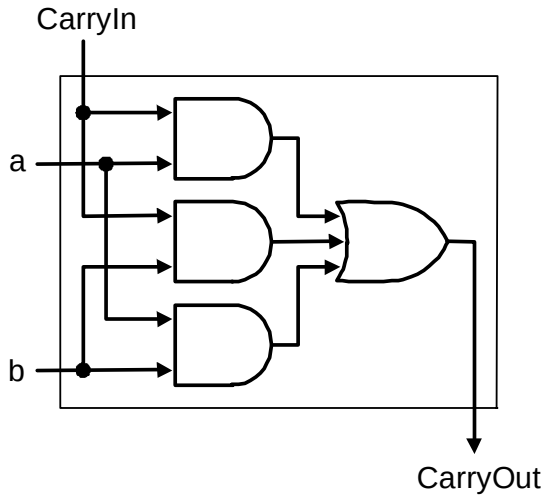


Example: from instructions to gates

Full Adder



Inputs			Outputs		Comments
a	b	CarryIn	CarryOut	Sum	
0	0	0	0	0	$0 + 0 + 0 = 00_{two}$
0	0	1	0	1	$0 + 0 + 1 = 01_{two}$
0	1	0	0	1	$0 + 1 + 0 = 01_{two}$
0	1	1	1	0	$0 + 1 + 1 = 10_{two}$
1	0	0	0	1	$1 + 0 + 0 = 01_{two}$
1	0	1	1	0	$1 + 0 + 1 = 10_{two}$
1	1	0	1	0	$1 + 1 + 0 = 10_{two}$
1	1	1	1	1	$1 + 1 + 1 = 11_{two}$



$$CarryOut = b.CarryIn + a.CarryIn + a.b + a.b.CarryIn$$

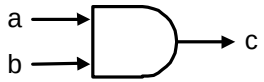
or

$$CarryOut = b.CarryIn + a.CarryIn + a.b$$

Example: from instructions to gates

Logic gates

1. AND gate (c = a . b)

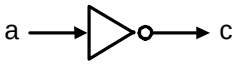


a	b	c = a . b
0	0	0
0	1	0
1	0	0
1	1	1

2. OR gate (c = a + b)

a	b	c = a + b
0	0	0
0	1	1
1	0	1
1	1	1

3. Inverter (c = a'



a	c = a'
0	1
1	0

Example: from instructions to gates

Boolean algebra

Axioms of Boolean algebra (set of elements B, two binary operators: + and .):

- Closure
- Identity element
  x+0=0+x=x
  x.1=1.x=x
- Commutative law
  x+y=y+x
  x.y=y.x
- Distributive law
  x.(y+z)=(x.y)+(x.z)
  x+(y.z)=(x+y).(x+z)
- Complement: for every x there exists x', such that:
  x+x'=1
  x.x'=0

Basic theorems

x+x=x, x.x=x
x+1=1, x.0=0
(x')'=x (involution)
x+(y+z)=(x+y)+z, x.(y.z)=(x.y).z (associativity)
(x+y)'=x'.y', (x.y)'=x'+y' (DeMorgan's law)
x+xy=x, x.(x+y)=x (absorption)