

Week 2

Binary numbers — any positive integer x can be represented in binary with 2^n bits, where $n = \lceil \log_2 x \rceil$

0	0
1	1
2	10
3	11
4	100
5	101
:	

N bits $\rightarrow 2^N$ possibilities

In general, for any base b :

$$(x_n x_{n-1} \dots x_0)_b = \sum_{i=0}^n x_i \cdot b^i$$

Binary \rightarrow Decimal

$$\begin{aligned} 101 &= 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 \\ &= 4 + 1 \\ &= 5 \end{aligned}$$

Binary addition

Break it down into:

1. Half adder (2 bits)
2. Full adder (3 bits)
3. Adder (arbitrary number of bits)

Decimal \rightarrow Binary

$$\begin{aligned} 87_{\text{decimal}} &= 64 + 16 + 4 + 2 + 1 \\ &= 0 \ 1 \ 0 \ 1 \ 0 \ 1 \ 1 \end{aligned}$$

2's complement

$$-x \rightarrow 2^n - x$$

Addition with 2's complement is "free"!

Representing Negative Numbers

Options:

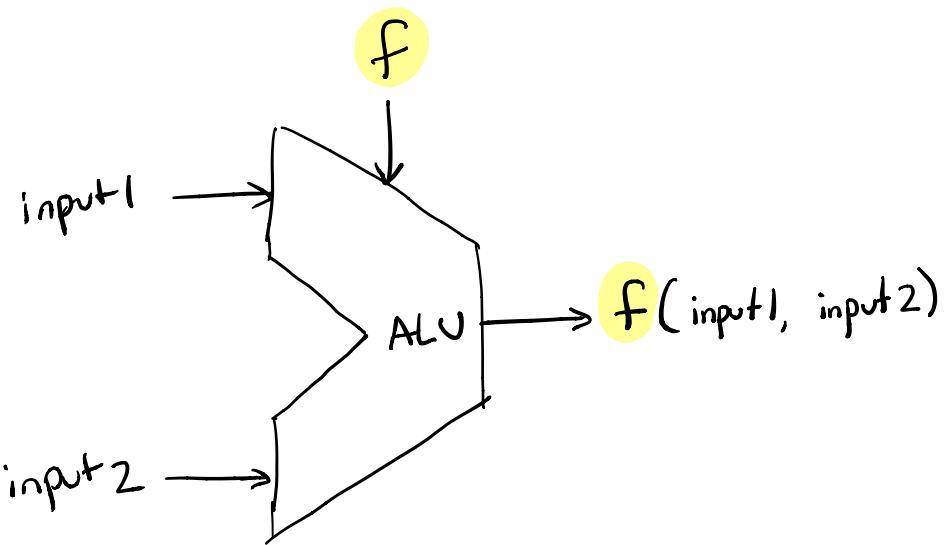
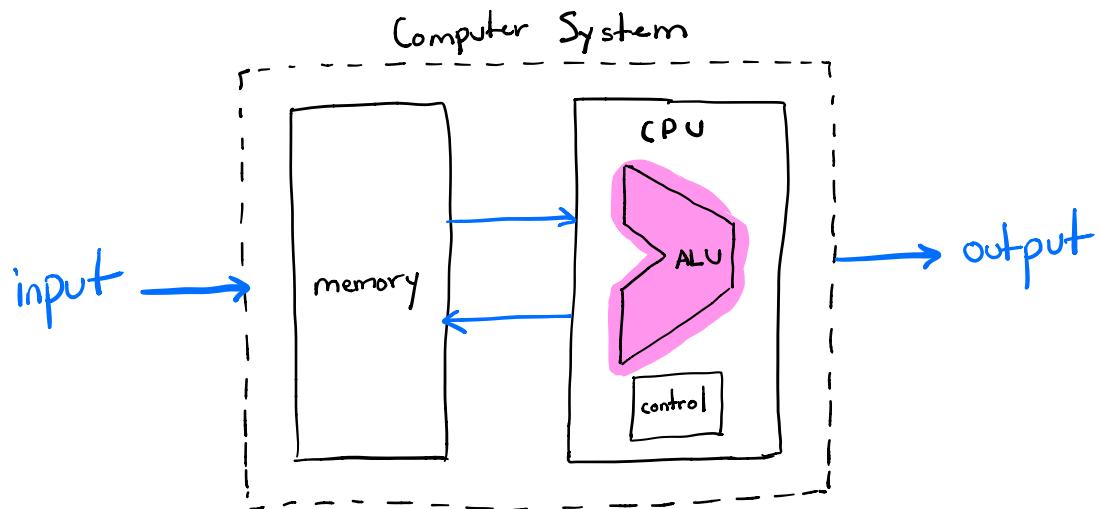
- signed magnitude representation (SMR)
- 1's complement
- 2's complement

what most computers use today!

Negation

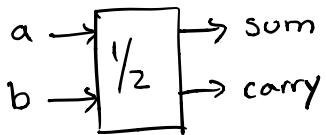
invert all the bits of x and add 1!

Von Neumann Architecture



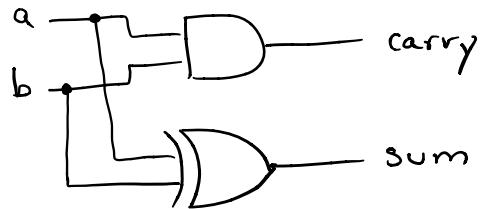
f is one out of a family of pre-defined arithmetic and logical functions.

Half adder

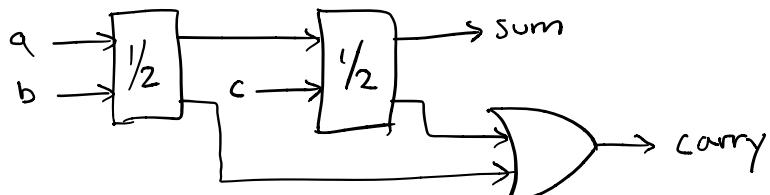
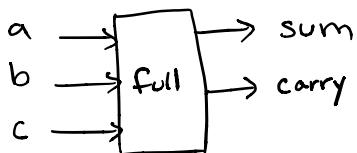


a	b	sum	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

↓ xor! ↓ and!

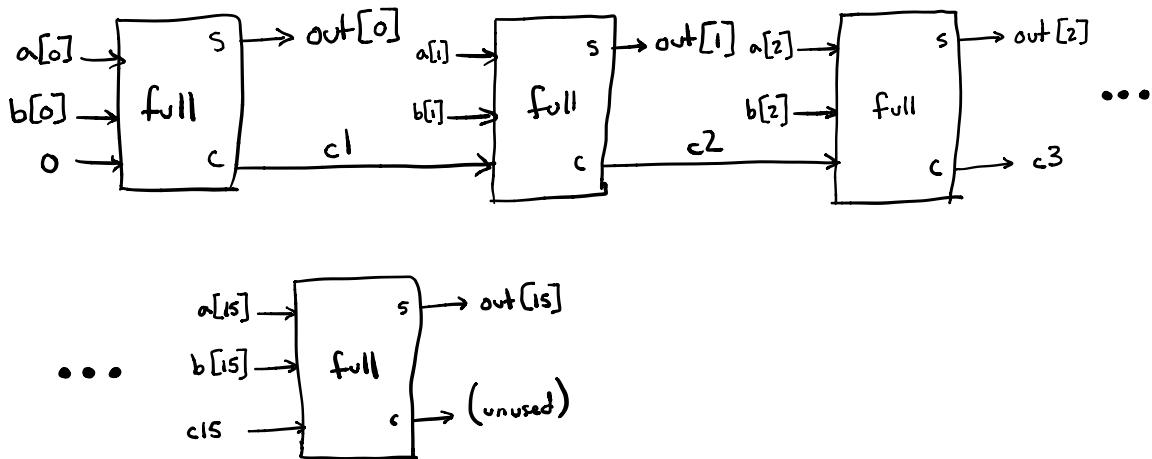


Full adder

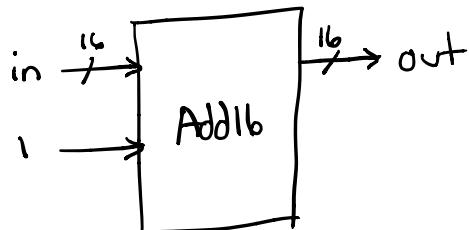


a	b	c	sum	carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Add16

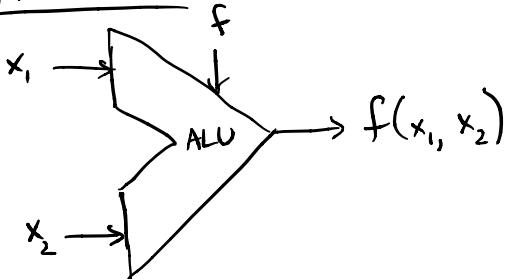


Inc16



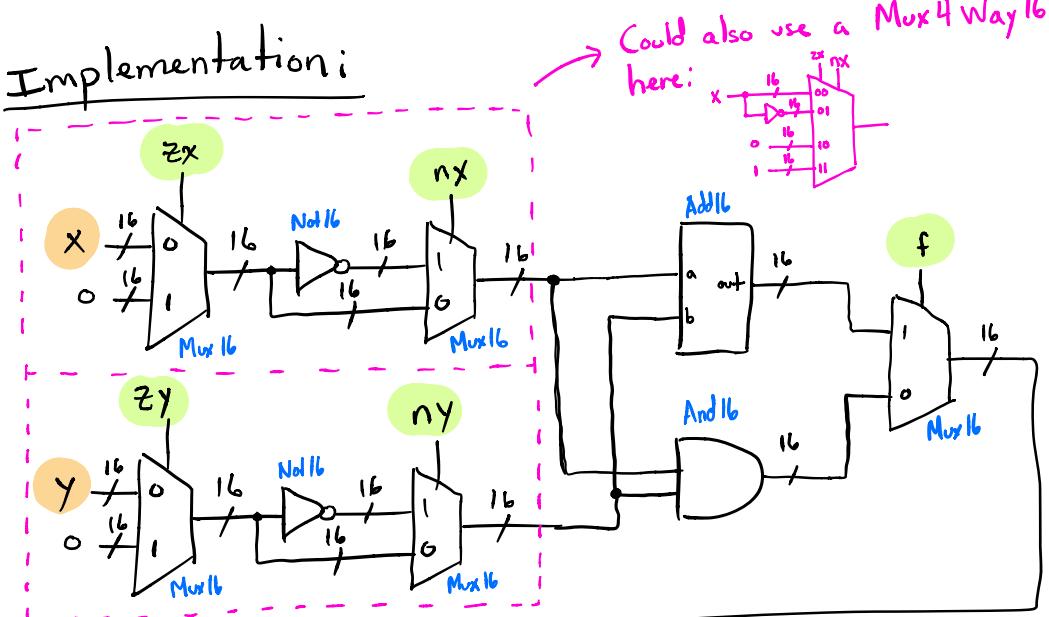
ALU (Arithmetic logic unit)

Abstraction:

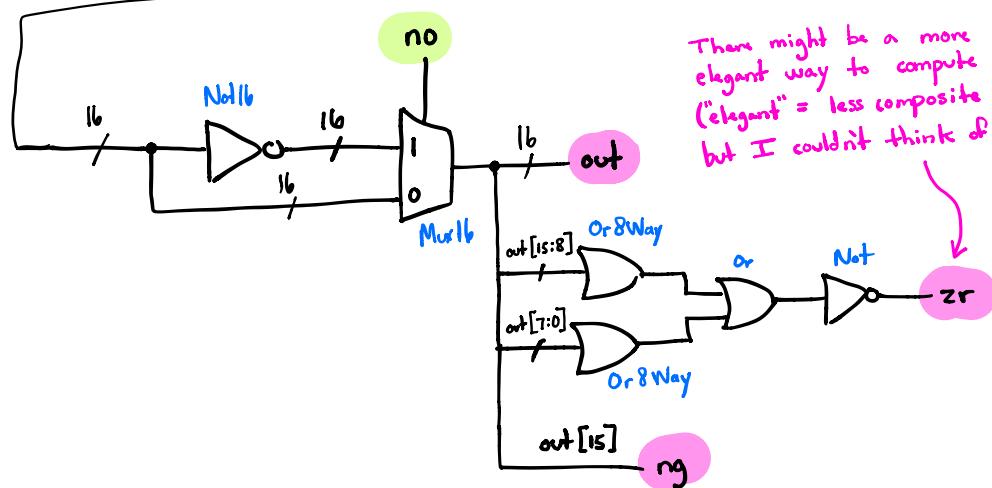
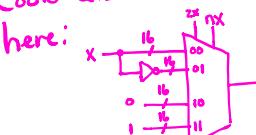


Computes one of several functions on the two inputs, and outputs the result.

Implementation:



Could also use a Mux 4 Way 16 here:



How does $y - x$ work?

$$\begin{array}{lll} z_x = 0 & z_y = 0 & f = x + y \\ n_x = 0 & n_y = 1 & n_o = 1 \end{array} \rightarrow \text{out} = \neg(x + \neg y)$$

Assume n bits.

$$1) \neg x = 2^n - 1 - x$$

$$\begin{matrix} \swarrow \\ x + \neg y \end{matrix}$$

$$\begin{aligned} x + \neg y &= x + 2^n - 1 - y \\ \neg(x + \neg y) &= 2^n - 1 - (x + 2^n - 1 - y) \\ &= \cancel{2^n} - x - \cancel{2^n} + \cancel{1} + y \\ &= y - x \end{aligned}$$

Example:

$$\begin{array}{ll} x = 1110 \quad (-2) & \rightarrow 7 \\ y = 0101 \quad (5) & \end{array}$$

↓

$$x = 1110$$

$$y = 1010$$

$$\begin{array}{l} x+y = 1000 \\ \neg(x+y) = 0111 \rightarrow 7 \end{array}$$

Perspective

Are the chips we built standard? Yes, except for the ALU, which is greatly simplified compared to typical ALUs. This is in line with a major theme of the course, which is **simplicity** and **elegance**.

e.g. multiplication and division are deferred to software. This trades **performance** and **efficiency** for **simplicity**

Some efficiency/optimization discussion

- Can make the 16-bit adder "faster" (in terms of worst-case propagation delay) by using a carry-lookahead technique → compute the carry bits independently
 - Since propagation delay in real implementations is not modeled in this course, optimizations/efficiency considerations like this are a "don't-care"