FIT 1047

Introduction to computer systems, networks and security

MONASH University

From logic to algebra

- Boolean algebra rules
- Introducing time
- Flip Flops

So far, you have mainly seen Boolean logic

The term Boolean algebra implies that we might be able to do arithmetic on symbols.

Algebra

any of various systems or branches of mathematics or logic concerned with the properties and relationships of abstract entities (as complex numbers, matrices, sets, vectors, groups, rings, or fields) manipulated in symbolic form under operations often analogous to those of arithmetic

(Merriam Webster)

Laws of Boolean Algebra

- Identity Law
- Null Law (or Dominance Law)
- Idempotent Law
- Complement Law
- Commutative Law
- Associative Law
- Distributive Law
- Absorption Law
- DeMorgans Law
- Double Complement Law

Identity Law

AND Form	OR Form
1A = A	0+A = A

Null Law (Dominance Law)

AND Form	OR Form
0A = 0	1+A = 1

Idempotent Law

AND Form	OR Form
AA = A	A+A = A

Complement Law

AND Form	OR Form
$A\overline{A} = 0$	$A+\overline{A} = 1$

Commutative Law

AND Form	OR Form
AB = BA	A+B = B+A

Associative Law

AND Form	OR Form
(AB)C = A(BC)	A+(B+C) = (A+B)+C

Distributive Law

AND Form	OR Form
A+(BC) = (A+B)(A+C)	A(B+C) = AB+AC

Absorption Law

AND Form	OR Form
A(A+B) = A	A+AB = A

DeMorgans Law

AND Form	OR Form
$\overline{AB} = \overline{A}+\overline{B}$	$\overline{(A+B)} = \overline{AB}$

Double Complement Law

$$\overline{\overline{A}} = A$$

Optimization of Boolean functions

When realizing a function as circuit, one would like to minimize gates. Boolean functions can be minimized using the different laws.

However, determining the correct order of applying the laws is sometimes difficult.

In addition, one would like to minimize the use of different types of gates. Therefore, normalized forms can be useful.

For generic approach for minimizing (smaller) Boolean functions are Karnaugh maps or K-maps.

One of the most common forms of simplification in Boolean algebra is the following:

$A\overline{B} + AB = A(\overline{B} + B) = A$

The same with three variables:

$A\overline{BC} + AB\overline{C} = A\overline{C}$

Both functions are independent from the value of B.

K-maps provide an easy graphical way to find minimal terms that are then combined with OR to get the complete function.

Truth table for A AND B

A	B	AB
0	0	0
0	1	0
1	0	0
1	1	1

K-map for A AND B

		B	
		0	1
A	0	0	0
	1	0	1

Let's look at a K-map with 3 variables:

$\overline{B}A\overline{C} + AB\overline{C} + ABC + B\overline{A}C + \overline{A}C\overline{B}$

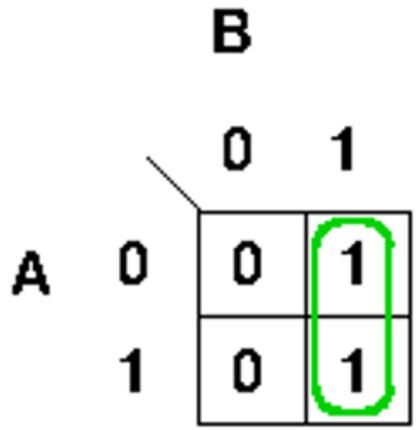
		BC			
		00	01	11	10
A	0	0	0	1	1
	1	1	0	1	1

Simplified version: $B + A\overline{C}$

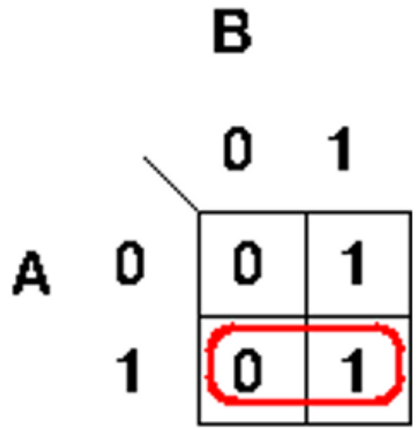
7 rules for working with K-maps

A \ B	0	1
0	0	1
1	0	1

Rule 1: No group can contain a zero.

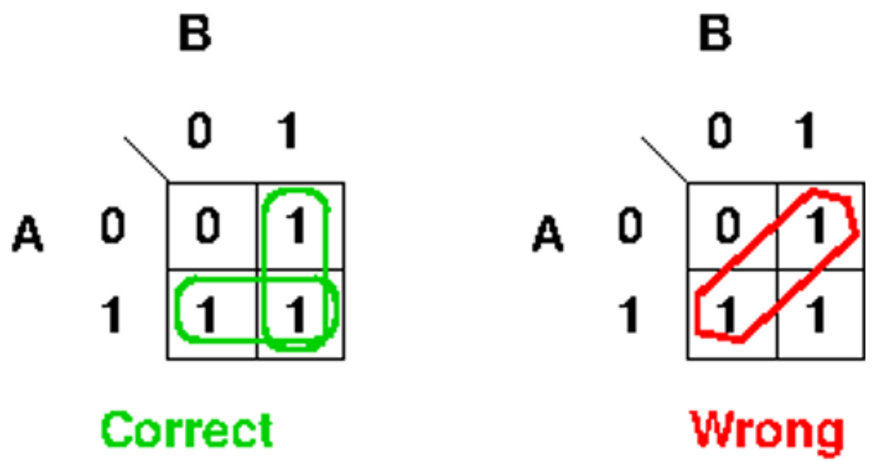


Correct

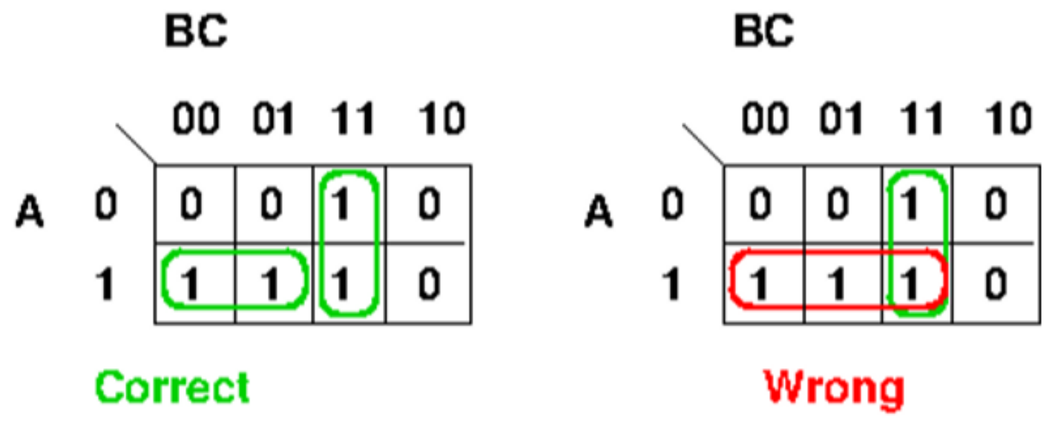


Wrong

Rule 2: Groups may be horizontal/vertical/square,
but never diagonal.


B
0
1
A
0
0
1
1
1
1
Correct
B
0
1
A
0
0
1
1
1
1
Wrong

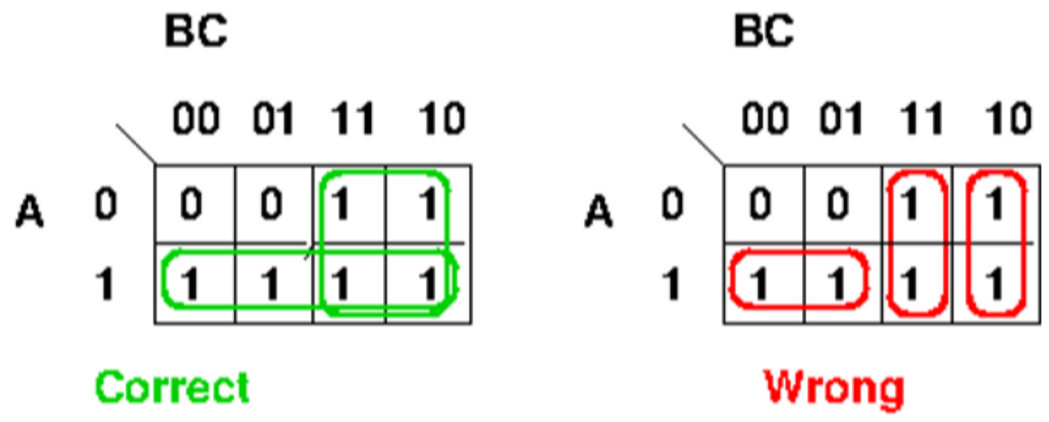
Rule 3: Groups must contain 1,2,4,8,16,32,... (powers of 2).


BC
00 01 11 10
A
0 0 0 1 0
1 1 1 1 0
Correct
BC
00 01 11 10
A
0 0 0 1 0
1 1 1 1 0
Wrong

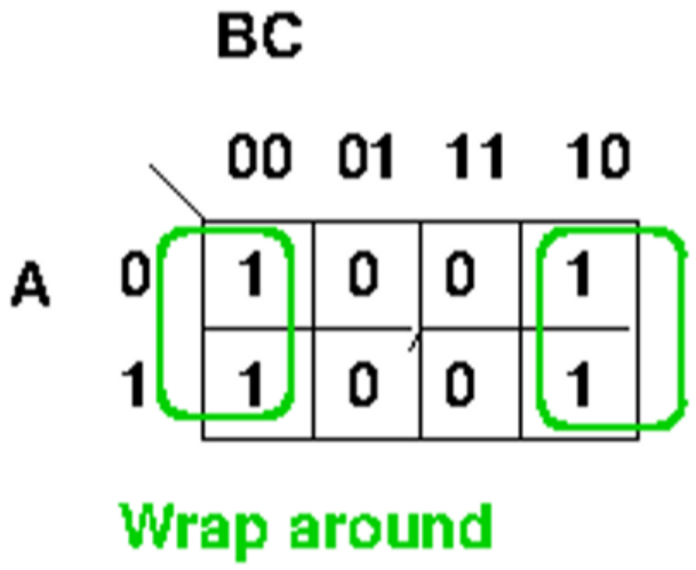
Rule 4: Each group must be as large as possible.

Rule 5: Groups can overlap.

Rule 6: Each 1 must be part of at least one group.


BC
00 01 11 10
A
0 0 0 1 1
1 1 1 1 1
Correct
BC
00 01 11 10
A
0 0 0 1 1
1 1 1 1 1
Wrong

Rule 7: Groups may wrap around the map.


BC
00 01 11 10
A
0 1 0 0 1
1 1 0 0 1
Wrap around

Rule 1: No group can contain a zero.

Rule 2: Groups may be horizontal/vertical/square, but never diagonal.

Rule 3: Groups must contain 1,2,4,8,16,32,... (powers of 2).

Rule 4: Each group must be as large as possible.

Rule 5: Groups can overlap.

Rule 6: Each 1 must be part of at least one group.

Rule 7: Groups may wrap around the map.

K-maps are useful for smaller Boolean functions.
Automated algorithms for optimization are used for bigger functions.

Universal gates

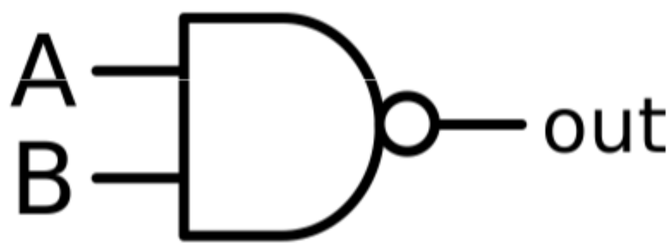
The NAND and NOR gate have special properties:

1. NAND can be realised very efficiently.
2. All other gates can be build only using NAND gates.

NAND is $\overline{AB}$

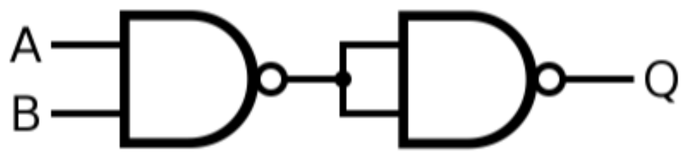
Truth table and symbol for NAND

A	B	$\overline{AB}$
0	0	1
0	1	1
1	0	1
1	1	0

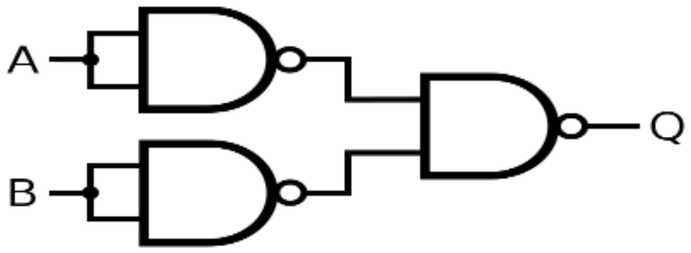


Building NOT with NAND gates

Building AND with NAND gates

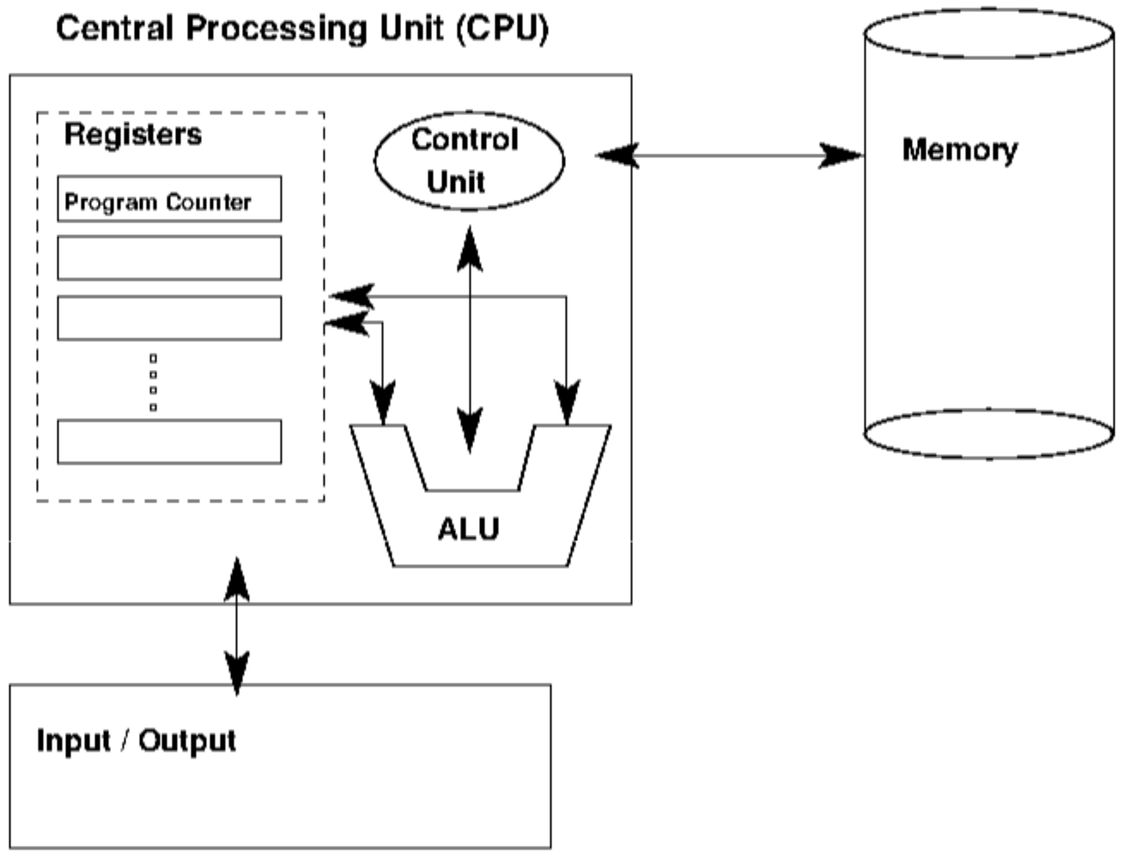


Building OR with NAND gates



A model for a computer

The von Neumann model (published by Hungarian mathematician John von Neumann):

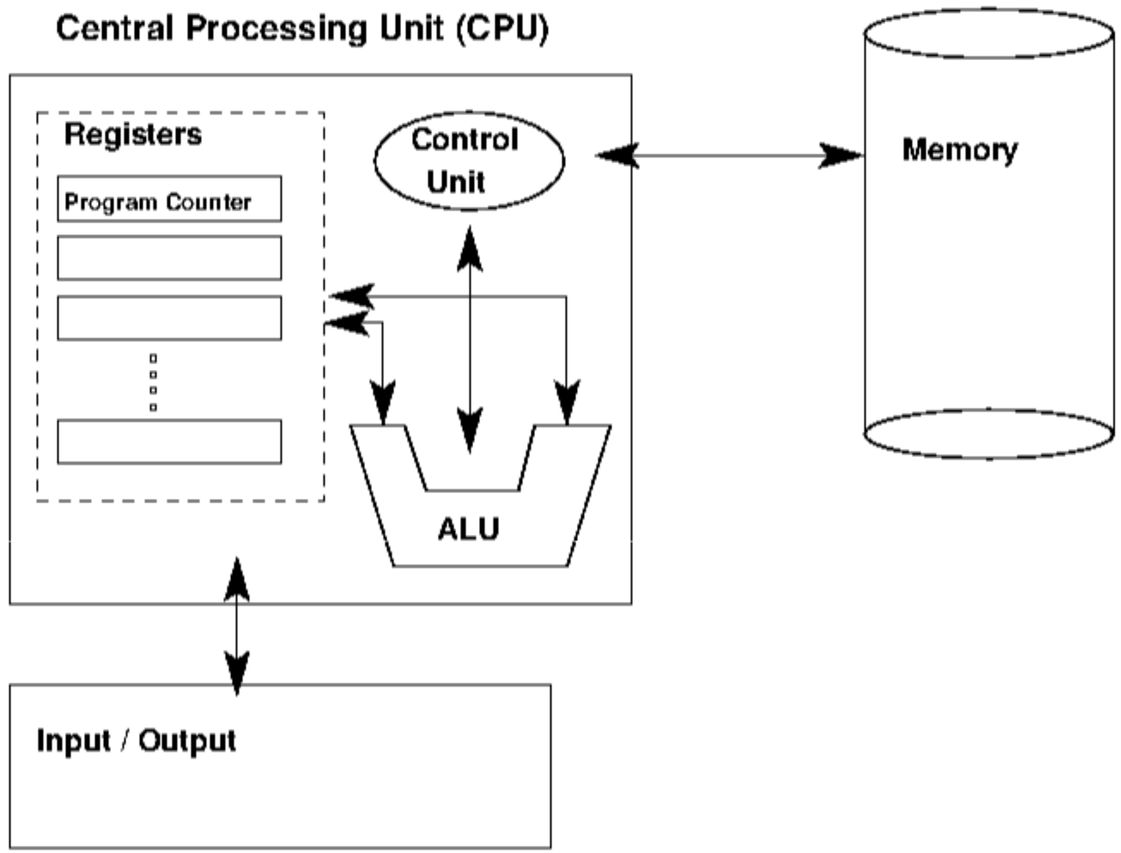


Memory

Stores data and program code

A model for a computer

The von Neumann model (published by Hungarian mathematician John von Neumann):

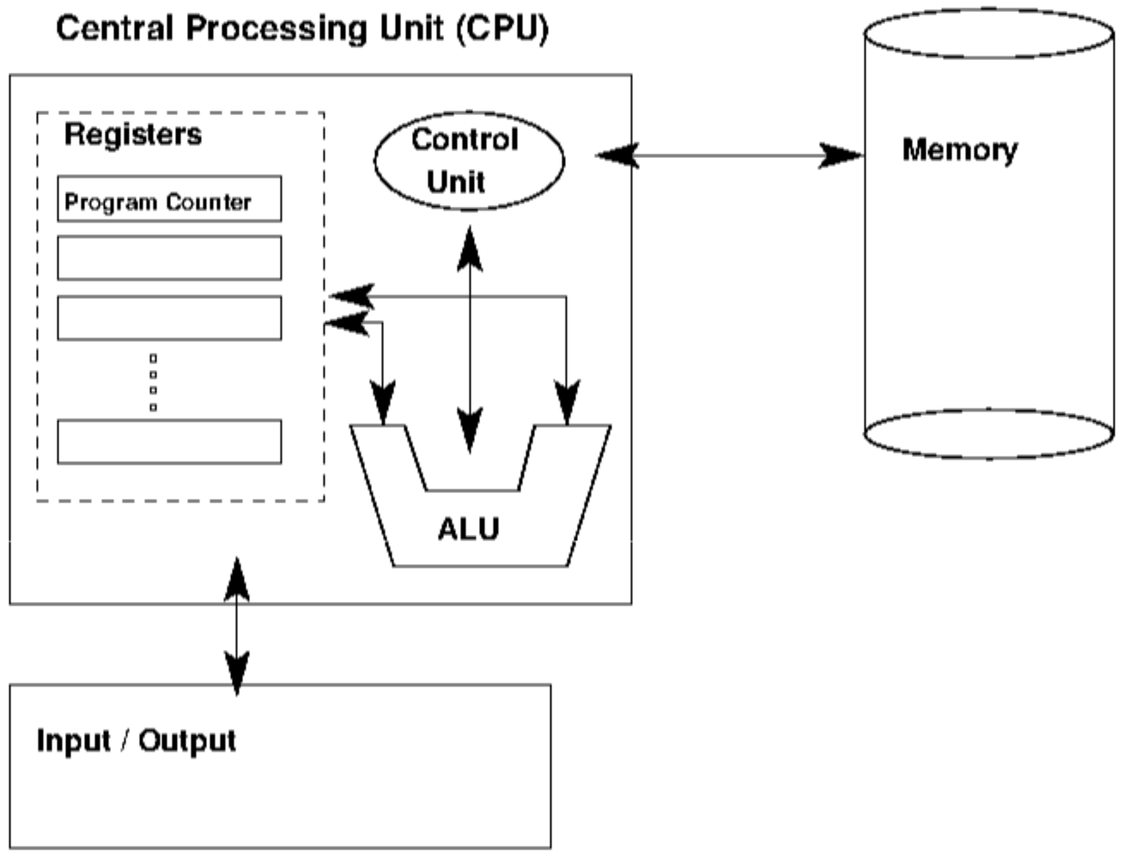


Control Unit

- Fetches program instructions from the memory
- Program counter defines where next instruction is located
- Writes results into the memory

A model for a computer

The von Neumann model (published by Hungarian mathematician John von Neumann):

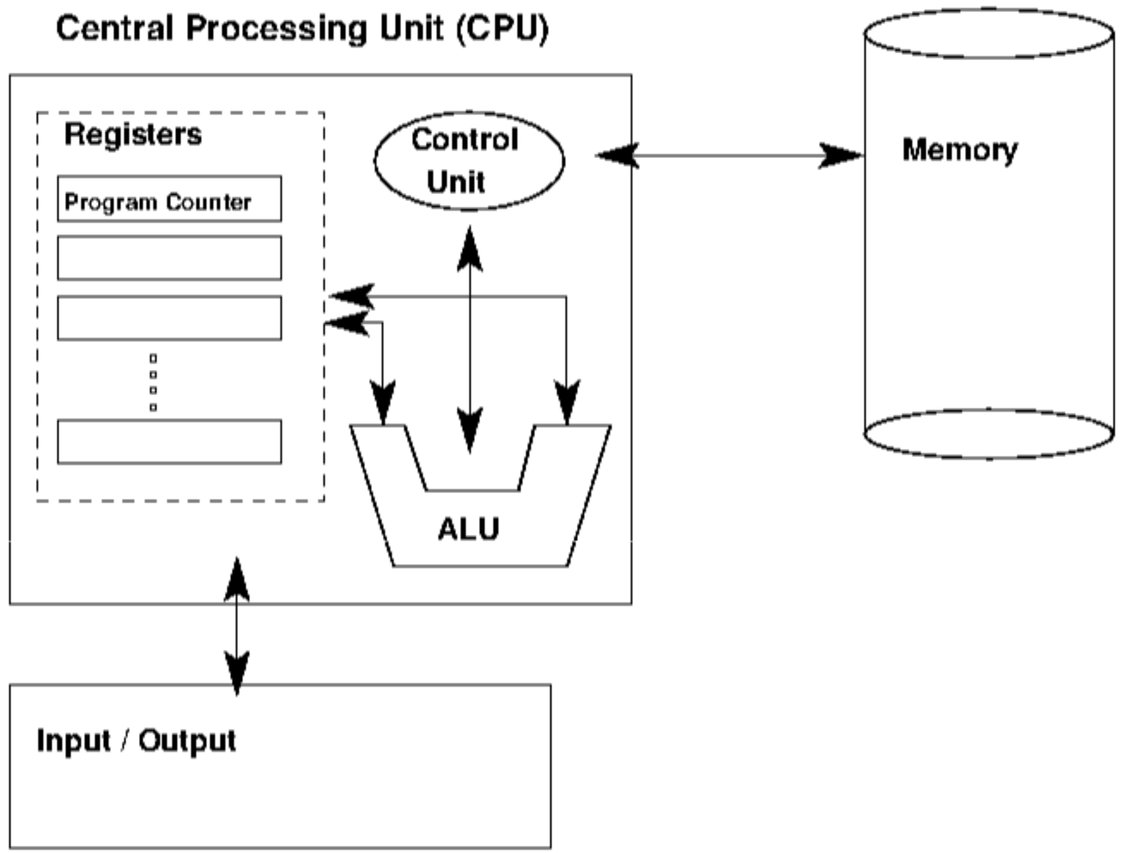


Arithmetic Logic Unit (ALU)

Actually executes instructions using data in registers and Input writes results into registers, into memory and/or Output

A model for a computer

The von Neumann model (published by Hungarian mathematician John von Neumann):



Input/Output System

Provides the interface to the world.

- User interaction (screen, keyboard, punch-cards, mouse, printer, sound, etc.)
- Communication with other machines/computers
- Sensors
- Actuators (e.g. in robotics)

Summarize topics of first 2 weeks

- A little bit of history
- Vacuum tubes and transistors
- bit, byte, word (8bit/16bit/32bit/64bit)
- Numbering systems (base 2, base 10, base 16)
- Conversion between numbering systems

- Signed integer representations (sign/magnitude, 1's complement, 2's complement)
- Properties of 2's complement, adding 2's complement numbers
- Floating point representation
- Properties of floating point, precision, rounding
- Characters: ASCII and Unicode

- Error detection: Parity bits, Checksum CRC
- Boolean Logic AND, OR, NOT, XOR, NAND, NOR
- Logic gates
- Simple logic circuits
- Boolean Algebra, Laws
- Karnaugh Maps
- von Neumann Model