COMP-202

Introduction to Computing 1

Section 1

Ernesto Posse

ENGTR 0060

MWF 12:30 - 13:30

Course website:

`http://www.cs.mcgill.ca/~cs202`

This course is about...

- Computer programming: solving problems involving information by means of instructing a computer

- Algorithms: An algorithm is a well-defined procedure to solve a problem

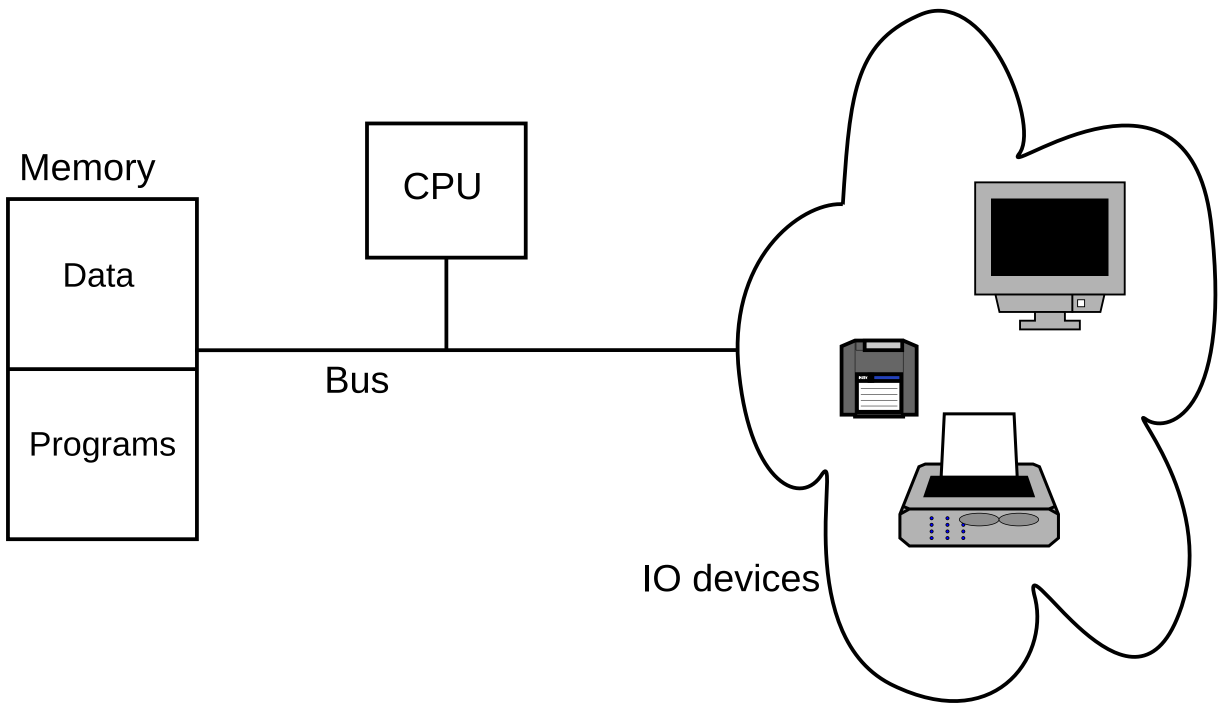
- Programming Language: A formal language used to express algorithms

- Programs: The realization of some algorithm in a programming language

Computers and Information

- What is a computer?
- How computers work?
- How is information stored/represented in a computer?

Computers and Information



Memory, data and programs

- Memory:
  - Memory is a very long (but finite) list of *cells* or *memory locations*
  - Each cell is assigned a unique *address* (a natural number)
  - Each cell contains some piece of information (of fixed size)
  - Some cells contain just data
  - Other cells contain instructions for the processor

- Programs
  - A program is a sequence of instructions
  - A program can be stored in memory
  - Programs manipulate the data which is stored in other memory locations
  - Programs are data which is *executable* by the processor (Von Neumann Architecture)

Data representation

- A bit is the fundamental unit of information: 0 or 1
- To represent more complex things, we can form sequences of bits: 000101, 1101001, 00, 111111111111, 1010101010, ...
- Bit sequences represent binary numbers: numbers in base 2:
  - 0 is 0
  - 1 is 1
  - 2 is 10
  - 3 is 11
  - 4 is 100
  - 5 is 101
  - ...
- Binary numbers are ordinary numbers which are written with only two digits (0 and 1) instead of ten (0 to 9).

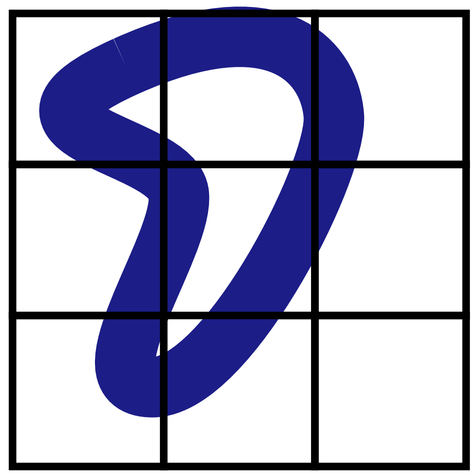
Data representation

- Bit sequences can represent other things: e.g. letters
  - 'a' is 01100001
  - 'b' is 01100010
  - 'c' is 01100011
  - ...
  - 'e' is 01100101
  - ...
- And therefore text: "hello" is 01101000 01100101 01101100 01101100 01101111

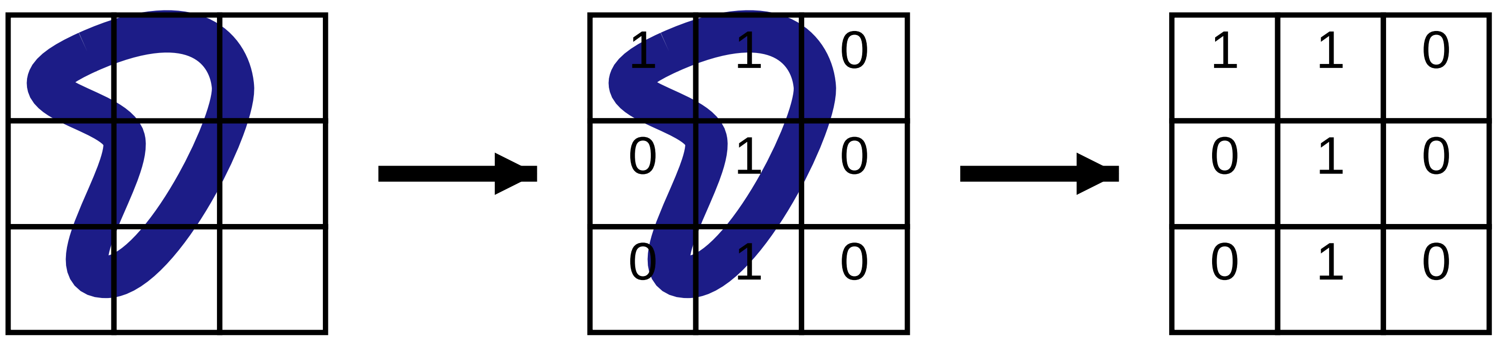
Data representation

- They can also represent images

Data representation



Data representation

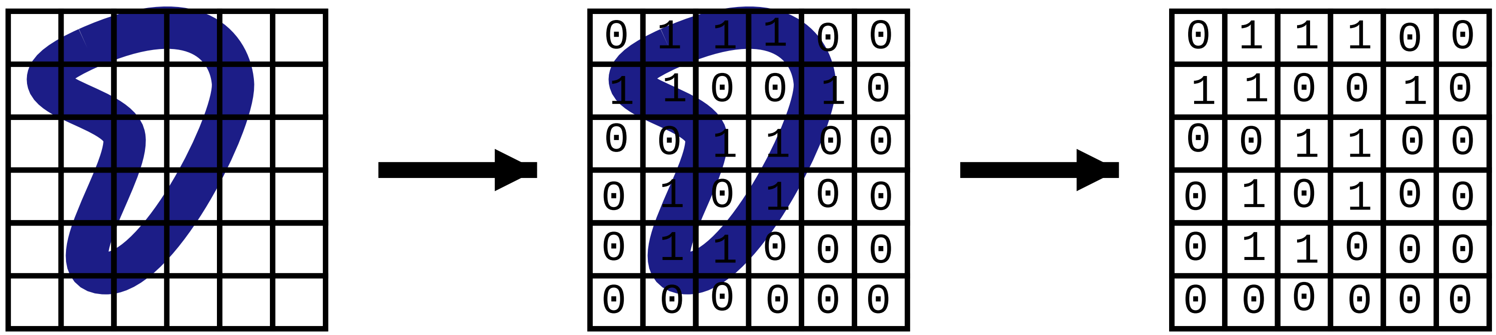


110010010

or

100111000

Data representation



011100110010001100010100011000000000

or

010000110110101010101100010000000000

Data representation

- Bit sequences can represent other things: e.g. letters
  - 'a' is 01100001 which is 97 in decimal
  - 'b' is 01100010 which is 98
  - 'c' is 01100011 which is 99
  - ...
  - 'e' is 01100101 which is 101
  - ...
- And therefore text: "hello" is 01101000 01100101 01101100 01101100 01101111
- or ... 104 101 108 108 111

Data in memory

- Each memory cell can contain a fixed number of bits: 32 bits, or 64 bits

- Some terminology:
  - A sequence of bits with the size of a memory cell is called a *word*
  - A sequence of 8 bits is called a *byte*
  - A sequence of 1024 bytes is called a *kilobyte* of KB ($1024 = 2^{10}$)
  - A sequence of 1024 kilobytes is a *megabyte* (MB)
  - A sequence of 1024 megabytes is a *gigabyte* (GB)
  - A sequence of 1024 gigabytes is a *terabyte* (TB)

Data in memory

- How much information can be represented by *n* bits?
  - 1 bit: 2 possible values
  - 2 bits: 4 possible values
  - 3 bits: 8 possible values
  - 4 bits: 16 possible values
  - ...
  - n bits: 2^n possible values

- To represent the English alphabet we need ? bits

- If we have q possible values, how many bits do we need?: $\lceil log_2 q \rceil$

- The ASCII code uses 8 bits: letters, decimal digits, symbols, etc.

- Unicode uses 16 bits: accents, different alphabets, more symbols, etc.

Binary to decimal conversion

- Problem: given a sequence of *n* bits, what is the decimal (base 10) representation of the sequence?
- Examples:
  - 0000000000 is 0
  - 1 is 1
  - 0010 is 2
  - 11 is 3
  - 100 is 4
  - ...
- Notation: Let the sequence be $b = b_{n-1}b_{n-2}\cdots b_2b_1b_0$ (indexed from right to left, starting from 0)

Binary to decimal conversion

- Solution:

$$dec(b) = \sum_{i=0}^{n-1} b_i \cdot 2^i$$

- Examples:

$$\begin{aligned} dec(1101) &= 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 \\ &= 1 \cdot 8 + 1 \cdot 4 + 0 \cdot 2 + 1 \cdot 1 \\ &= 8 + 4 + 1 \\ &= 13 \end{aligned}$$

$$\begin{aligned} dec(101101) &= 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 \\ &= 1 \cdot 32 + 0 \cdot 16 + 1 \cdot 8 + 1 \cdot 4 + 0 \cdot 2 + 1 \cdot 1 \\ &= 32 + 8 + 4 + 1 \\ &= 45 \end{aligned}$$

Decimal to binary conversion

- Problem: given a natural number (positive integer or 0) m, what is its binary (base 2) representation?

- Analysis:
  - Given m, find the sequence of bits $b = b_{n-1}b_{n-2} \cdots b_2 b_1 b_0$ such that $m = dec(b)$
  - Inputs: a natural number m
  - Ouput: a sequence of bits b such that $m = dec(b)$

Decimal to binary conversion

- Algorithm:

1. Divide m by 2. This yields a quotient q_0 and a remainder r_0 which is 0 or 1. (why?)

2. Divide q_0 by 2. This yields a quotient q_1 and a remainder r_1

3. Divide q_1 by 2. This yields a quotient q_2 and a remainder r_2

4. ...

5. ... until you reach 0

6. Then let $b = r_l r_{l-1} \cdots r_2 r_1 r_0$

Decimal to binary conversion

- Example: Consider $m = 114$

1. Divide 114 by 2. The result is 57 and the remainder is 0
2. Divide 57 by 2. The result is 28 and the remainder is 1
3. Divide 28 by 2. The result is 14 and the remainder is 0
4. Divide 14 by 2. The result is 7 and the remainder is 0
5. Divide 7 by 2. The result is 3 and the remainder is 1
6. Divide 3 by 2. The result is 1 and the remainder is 1
7. Divide 1 by 2. The result is 0 and the remainder is 1
8. The result is 1110010

- To check this:

$$\begin{aligned} dec(1110010) &= 1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^1 \\ &= 64 + 32 + 16 + 2 \\ &= 114 \end{aligned}$$

Decimal to binary conversion

1. Let b be "" (the empty sequence)
2. Let q be m
3. While q is not 0 repeat the following:
   (a) Let new_q be q divided by 2, and
   (b) Let r be the remainder of q divided by 2
   (c) Append r in the front of the sequence b
   (d) Set q to be new_q
   (e) Repeat (from line 3)

Decimal to binary conversion

- Trace of execution
- Example: Consider the case $m = 44$

iteration	q	new_q	r	b
0	44			""
1	22	22	0	"0"
2	11	11	0	"00"
3	5	5	1	"100"
4	2	2	1	"1100"
5	1	1	0	"01100"
6	0	0	1	"101100"

Decimal to binary conversion

- Trace of execution
- Example: Consider the case $m = 26$

iteration	q	new_q	r	b
0	26			“”
1	13	13	0	“0”
2	6	6	1	“10”
3	3	3	0	“010”
4	1	1	1	“1010”
5	0	0	1	“11010”

Elements of algorithms

- Variables to store values (such as numbers, sequences, etc.)

- Instructions organized and executed in sequence: order of execution matters

- Instructions for:
  - computing values (e.g. divide by)
  - assigning values to variables
  - repeating a set of instructions
  - etc.

Elements of algorithms

- Solving a problem: (General methodology)

1. Stating the problem
2. Understanding the problem -> Analysis
3. Designing a possible solution -> Algorithm
4. Implementing the algorithm using a programming language

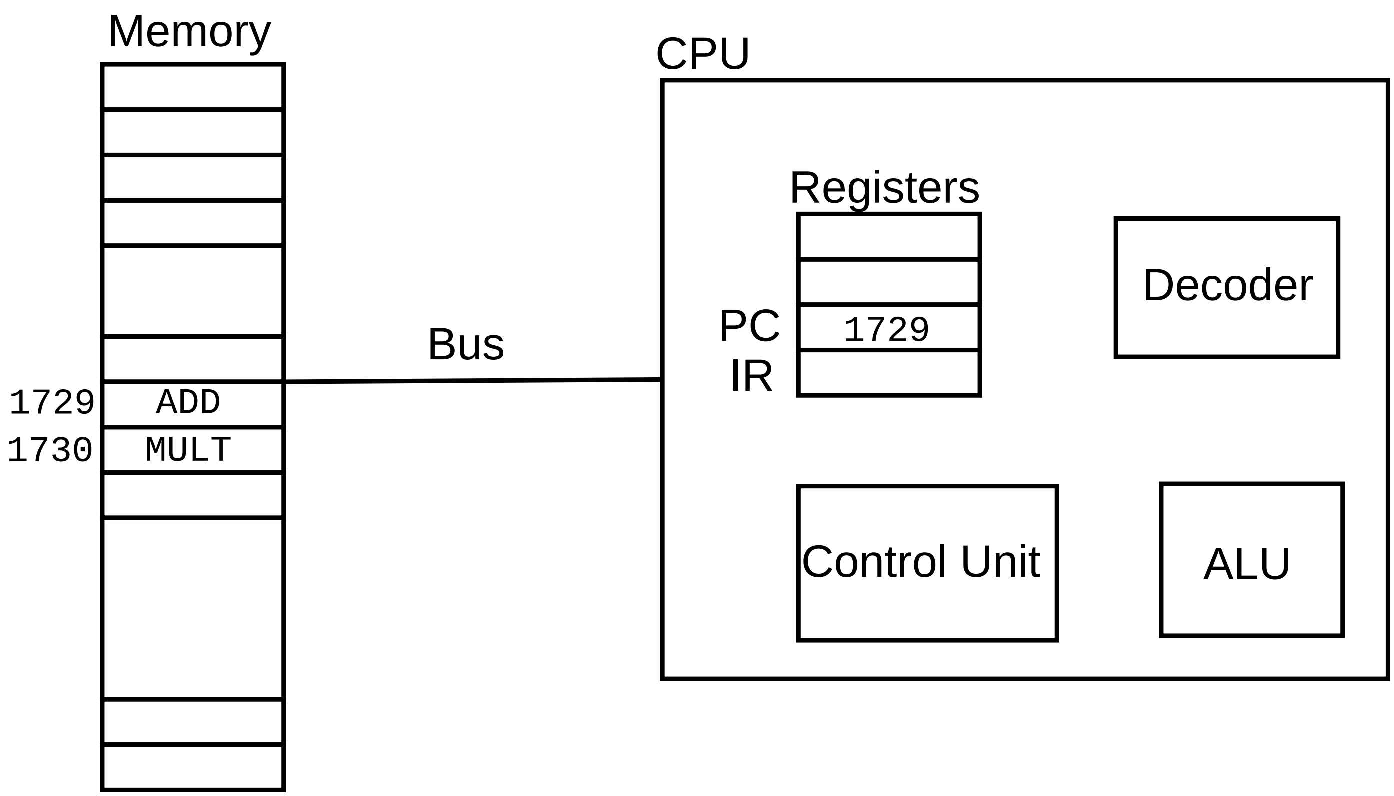
Computer Architecture

- Components:
  - CPU
  - Memory
  - IO devices
  - The Bus
- CPU:
  - Registers (PC, IR, ...)
  - ALU (Arithmetic-Logic Unit)
  - Control Unit
  - Decoder

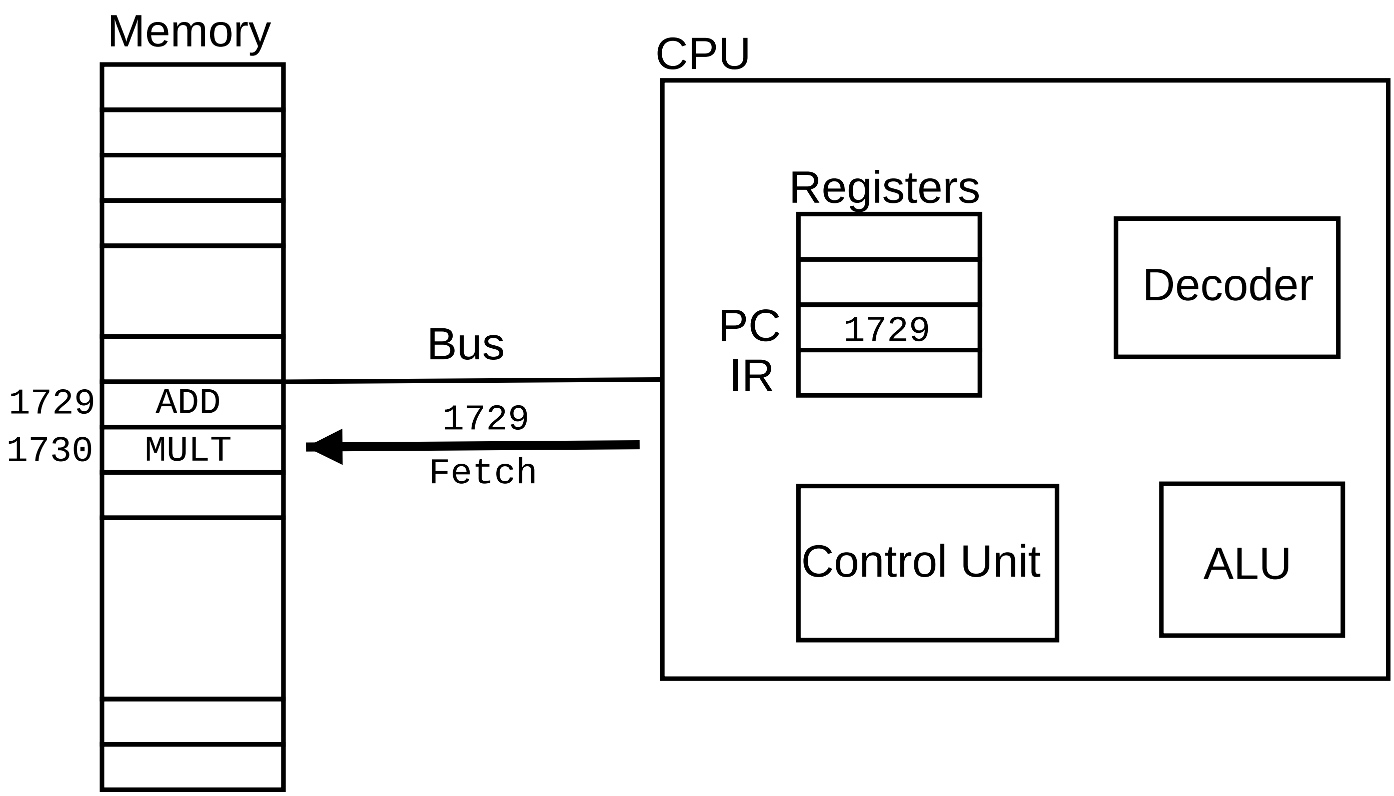
Computer Architecture

- A program is a sequence of instructions stored in memory
- Execution cycle: (Fetch-Decode-Execute)
  1. Fetch: The PC (program counter) register contains the address of the next instruction to be executed
     (a) The Control Unit sends this address to memory
     (b) Memory sends back the instruction stored in that address
     (c) The instruction is stored in the IR (instruction register)
  2. Decode: The instruction in the IR is passed to the Decoder which sends it to the appropriate circuit for execution
  3. Execute: The instruction is performed.
     (a) If the instruction is arithmetic or logic, it is executed by the ALU
  4. The PC register is updated to the next instruction
  5. Repeat

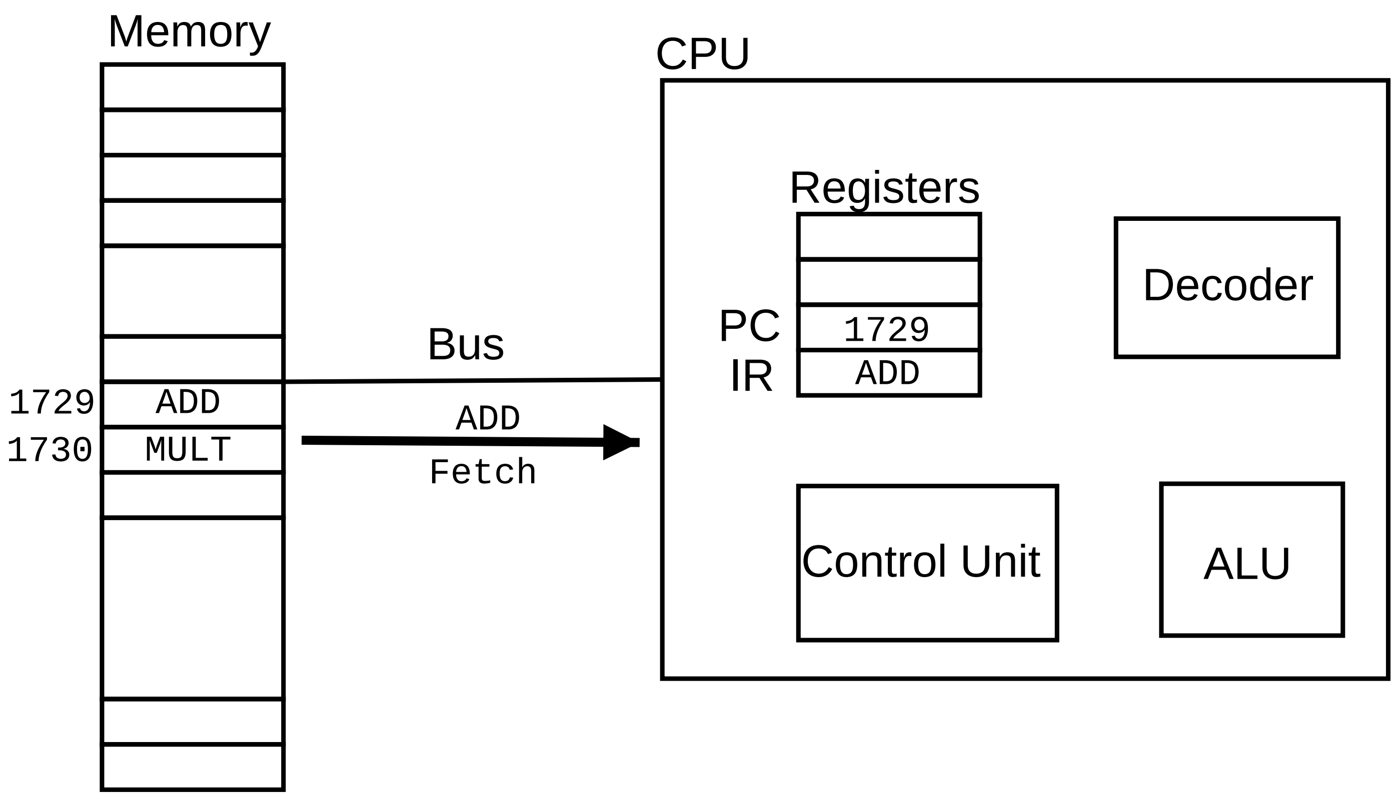
Computer Architecture



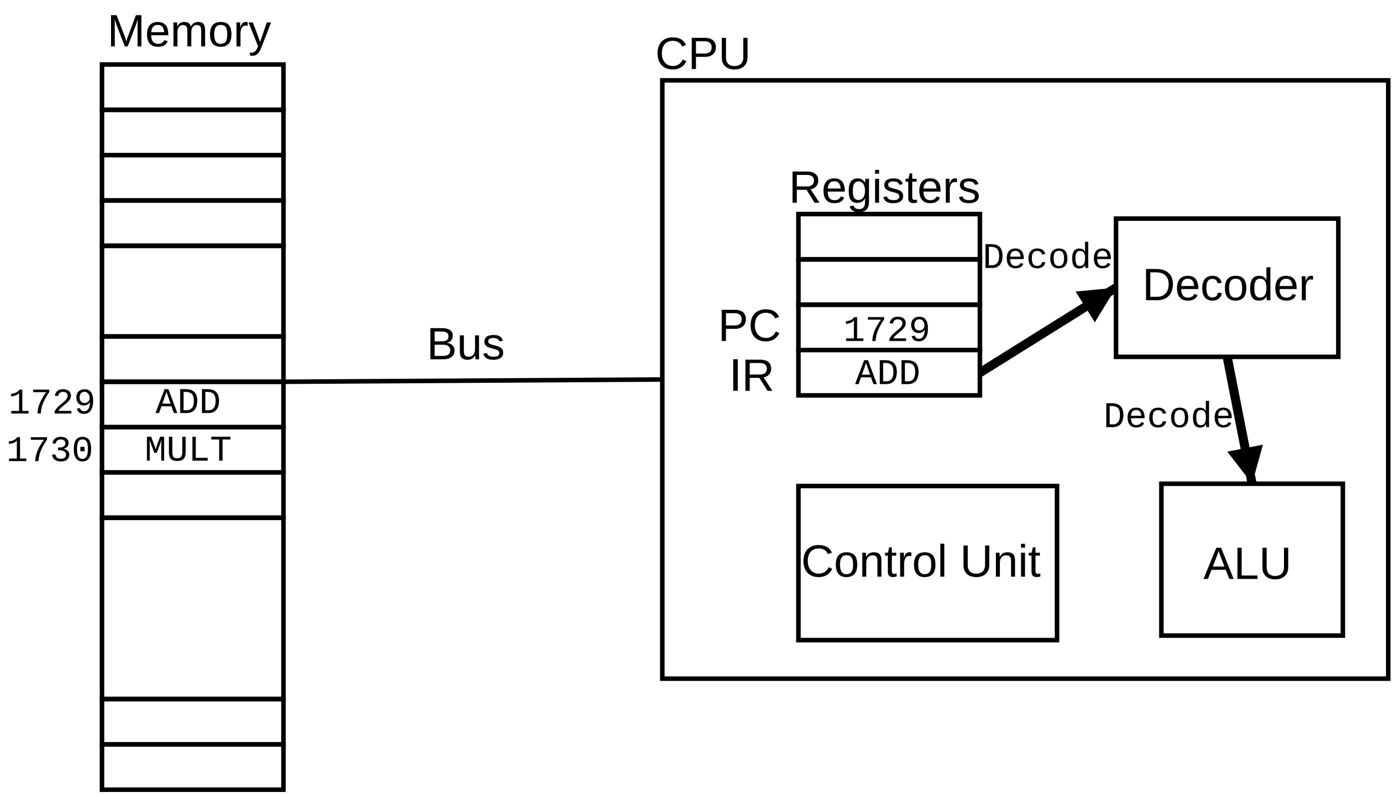
Computer Architecture



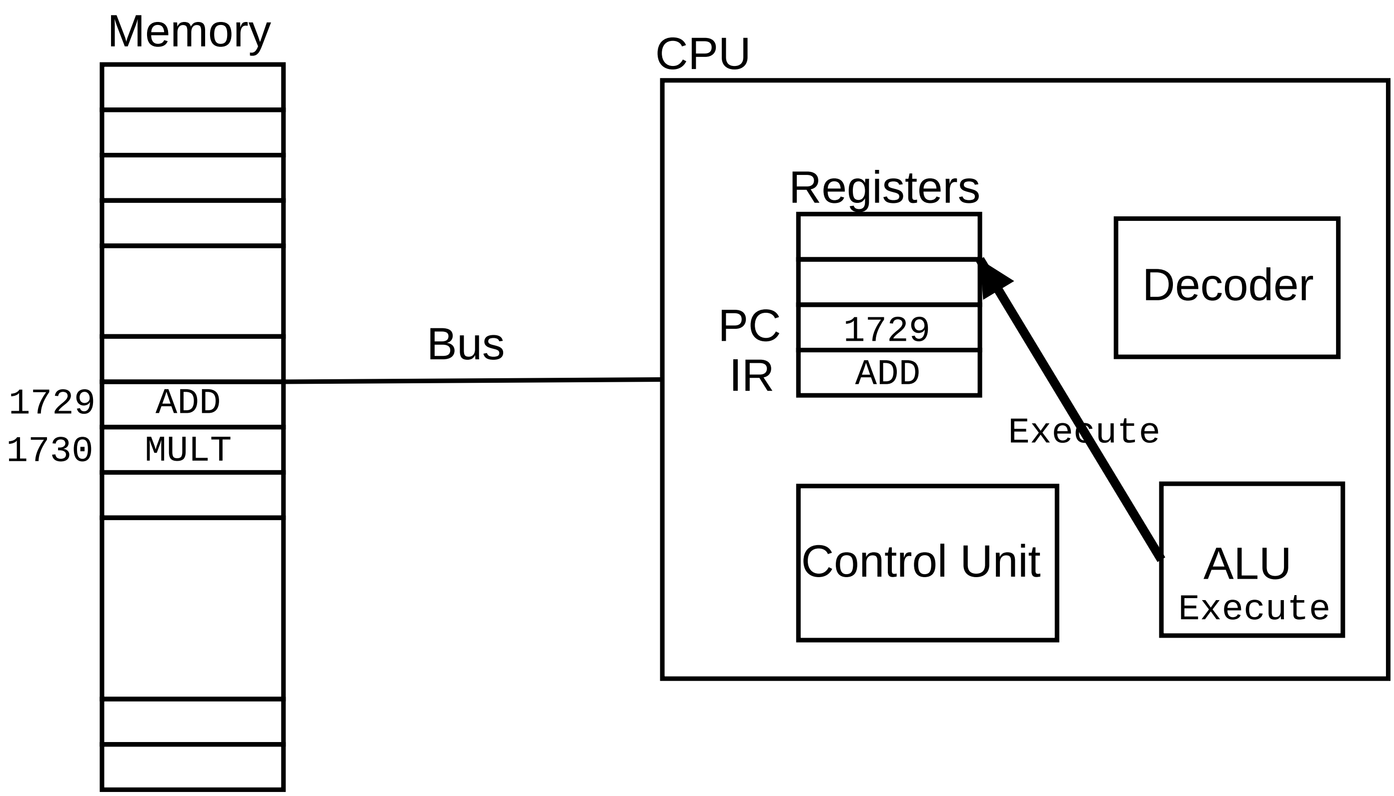
Computer Architecture



Computer Architecture



Computer Architecture



Computer Architecture