CS 101:
Computer Programming and Utilization

July-Nov 2016

Prof. Bernard L Menezes
(cs101@cse.iitb.ac.in)

Lecture 2: How Computers Work
(A very high level view)

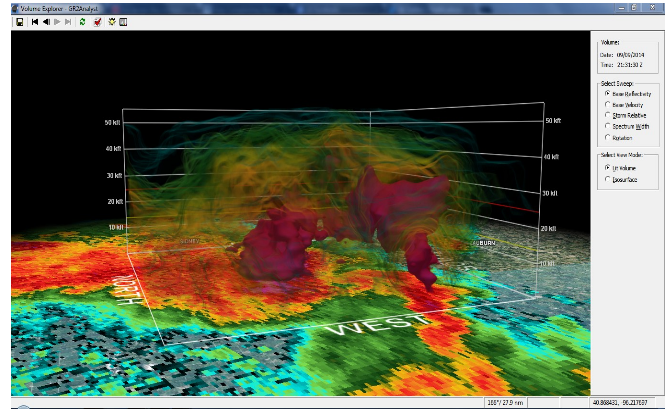
About These Slides

- Based on Chapter 2 of the book
  An Introduction to Programming Through C++
  by Abhiram Ranade (Tata McGraw Hill, 2014)

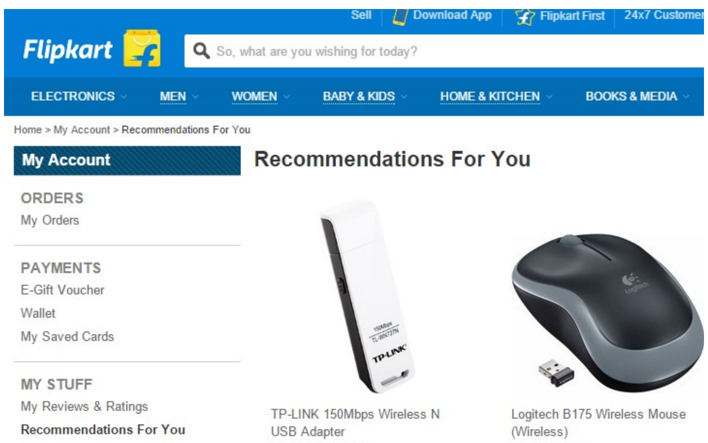
- Original slides by Abhiram Ranade
  – First update by Varsha Apte
  – Second update by Uday Khedker

A Computer Can Do Many Things

Control railway switching and signals



Weather prediction



Recommendations of what to buy

Perform super-realistic graphics and movement

How ?

How Does A Computer Work

Very simply:

A computer is a large* circuit with parts that

- read numbers from external world (Input)
- store numbers (Storage)
- perform arithmetic on numbers (Processing)
- send numbers to external world (Output)

*How large can large be?

- Physically small: ~1000 mm^2
- Logically large: ~10^9 transistors

How To Use A Computer To Solve Real-Life Problems?

1. Express the problem as a problem on numbers
2. Think of a solution in terms of the computations need to be performed, possibly repeatedly and conditionally

   = program

3. Feed the
   - the data in term of numbers, and
   - the program to the computer
4. Get the output on a screen or elsewhere (?)

Digging Deeper - Outline

- Examples of expressing real life problems as numerical problems
  - Picture processing
  - Predicting the weather
  - Understanding and processing language
- Algorithms and Programs
- High level design of a computer
  - Digital circuits
  - Parts of a computer
  - Stored program, compilation

“What is in this picture?”

Can we ask this question to a computer and get an answer?

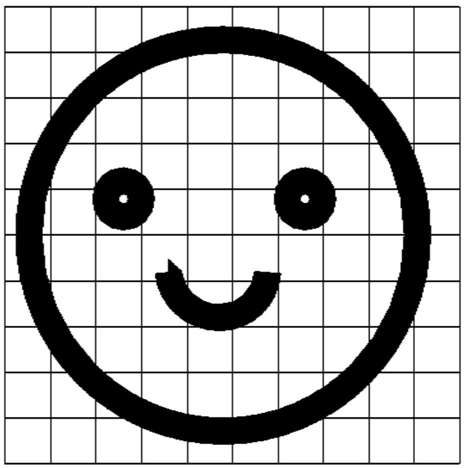
Start With Baby Steps

- Black and white picture representation and comprehension

How to represent black and white pictures?

- Suppose the picture is 10cm x 10cm
- Divide it into 0.1 mm x 0.1 mm squares
- The number of squares (or pixels) is 1000 x 1000
- If a square is mostly white, represent it by 0
- If a square is mostly black, represent it by 1
- Thus, our picture = 1 million numbers!

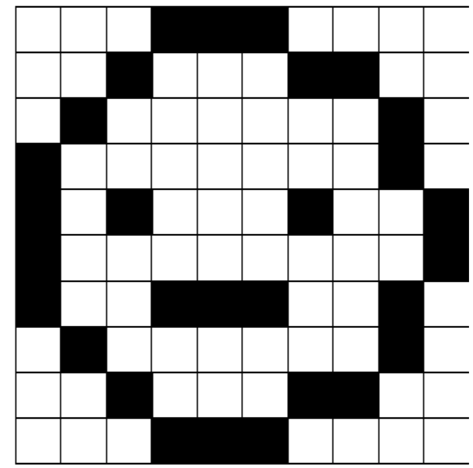
Picture Representation and Reconstruction



(a)

0	0	0	1	1	1	0	0	0	0
0	0	1	0	0	0	1	1	0	0
0	1	0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0	1	0
1	0	1	0	0	0	1	0	0	1
1	0	0	0	0	0	0	0	0	1
1	0	0	1	1	1	0	0	1	0
0	1	0	0	0	0	0	0	1	0
0	0	1	0	0	0	1	1	0	0
0	0	0	1	1	1	0	0	0	0

(b)



(c)

Image Recognition
Is this a picture of a vertical line?

- Input:

  A sequence of 1 million numbers (0 or 1) representing a 10cm x 10cm black & white picture
- What property does the sequence need to have if it is to contain a vertical line somewhere?
- All 0s, except for 1s at positions

  i, i+1000, i+2000, i+3000, i+4000,…

  for some *i*

A question about a picture converted into a question about numbers!

Doing More...

- Better representation if the picture is divided into more cells
- Pictures with different gray levels: sequence of numbers indicating degree of darkness

  e.g. 1 for light grey, 2 for medium grey, 3 for black
- Pictures with colours: 3 numbers per pixel

  each representing intensity of red, blue and green

  3 sequences for each : red component, blue component, green component

More *Recognition* problems

- Is the picture largely red in colour?
- Is there a square in the picture?
- Are two pictures similar?
- ...

Coming back to: Does the picture contain a Chameleon?

This question will need to be expressed as:

- Does the sequence of numbers representing the picture contain a subsequence satisfying certain properties?
- Which properties?
  - Enormous ingenuity needed to specify.
  - Very difficult problem
  - Main concern of a deep subject called Computer Vision

Weather Prediction

- Divide the surface of the earth into small regions (like pixels)
- For each region i, let p_i, t_i, h_i represent pressure, temperature, humidity
- Laws of physics will tell us what relationships p_i, t_i, h_i must satisfy
- We can measure some pressure, humidity, temperature values, and predict the rest
- Laws of physics also tell us how the values will change with time
- Smaller the regions, better will be the accuracy (Smaller the pixels, better will be the picture representation)

Representing A Language Using Numbers (1)

Define a numeric code for representing letters

- ASCII (American Standard Code for Information Interchange) is the commonly used code
- Letter ‘a’ = 97 in ASCII, ‘b’ = 98, …
- Uppercase letters, symbols, digits also have codes
- Code also for space character
- Words = sequences of ASCII codes of letters in the word

  ‘computer’ = 99, 111,109,112,117,116,101,114

Representing A Language Using Numbers (2)

- Sentences/paragraphs = larger sequences
- Does the word “computer” occur in a paragraph?
- Does a certain sequence of numbers occur inside another sequence of numbers?

A Summary

- Questions about pictures, weather, documents can be converted to questions about properties of number sequences
- Finding answers requires solving interesting math problems
- How will you represent Chess playing as a question on numbers?

Algorithm

- A precise sequence of mathematical operations using which a given (INPUT) sequences of numbers is changed into another (OUTPUT) sequence of numbers
- Allowed mathematical operations: arithmetic operations
- Possible to have conditions selection

add 1 to x if y is greater than 0

- Repetition is also allowed iteration

Repeat the following operations n times

Algorithms You Already Know

- The procedures you have learnt in primary school for arithmetic on numbers with many digits are really algorithms.
- Primary school algorithms contain all ingredients described earlier
  1. First add the least significant digit of the first number to the least significant digit of the second number
  2. If the sum is greater than 9 then carry the most significant digits
  3. Repeat as many times as there are digits
- Algorithms for determining whether a number is prime, finding the greatest common divisor of two numbers

More About Algorithms

- Early algorithms were invented for computing using paper and pencil

  eg. *Babylonian algorithm for square root*
- Many such algorithms are useful even today with computers
- More general notion of algorithms:
  - Precise description of steps needed to perform any clearly defined task, not necessarily computational

  An Algorithm to cook rice
    - s = number of servings
    - Put s/2 cups of rice and s cups of water into a pot
    - Put on stove and bring the pot contents to boil
    - …

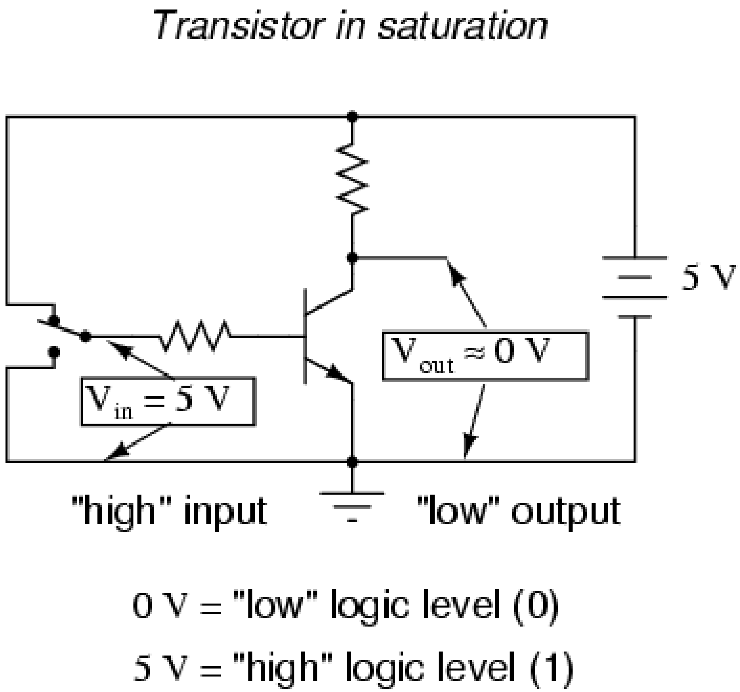
Programs

- Algorithms written using a precise notation
- Many different notations/languages possible
- C++ is one such language
- Other languages: C, Java, Python, Basic, Lisp, …

The Hardware
(A very high level glimpse)

- How do we store numbers in hardware?
- How is an instruction expressed in hardware?
- How is it executed?
- How do we output the numbers?

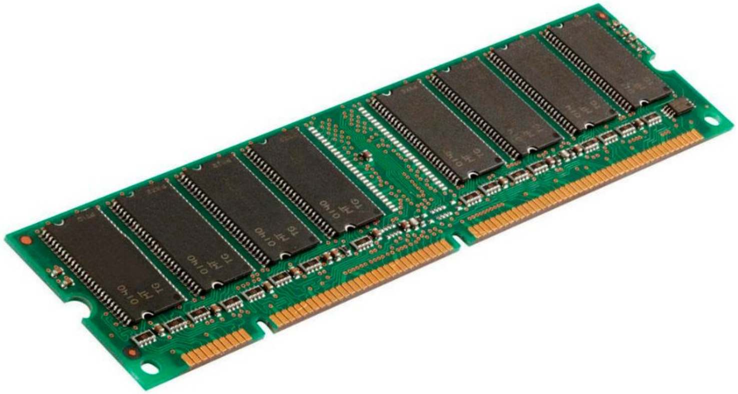
Digital Circuits - Operations



An inverter circuit

- Building blocks of computers
- Circuits have wires that carry current, and are at a certain electric potential.
- Digital circuits: interpret electrical potential/voltage as numbers.
- Simplest convention
  - Voltage above 5 volt = number 1
  - Voltage between 0 and 0.2 volt = number 0
  - Circuit designed so that voltage will never be between 0.2 and 5 volt, hence no ambiguity.

Digital Circuits - Storage

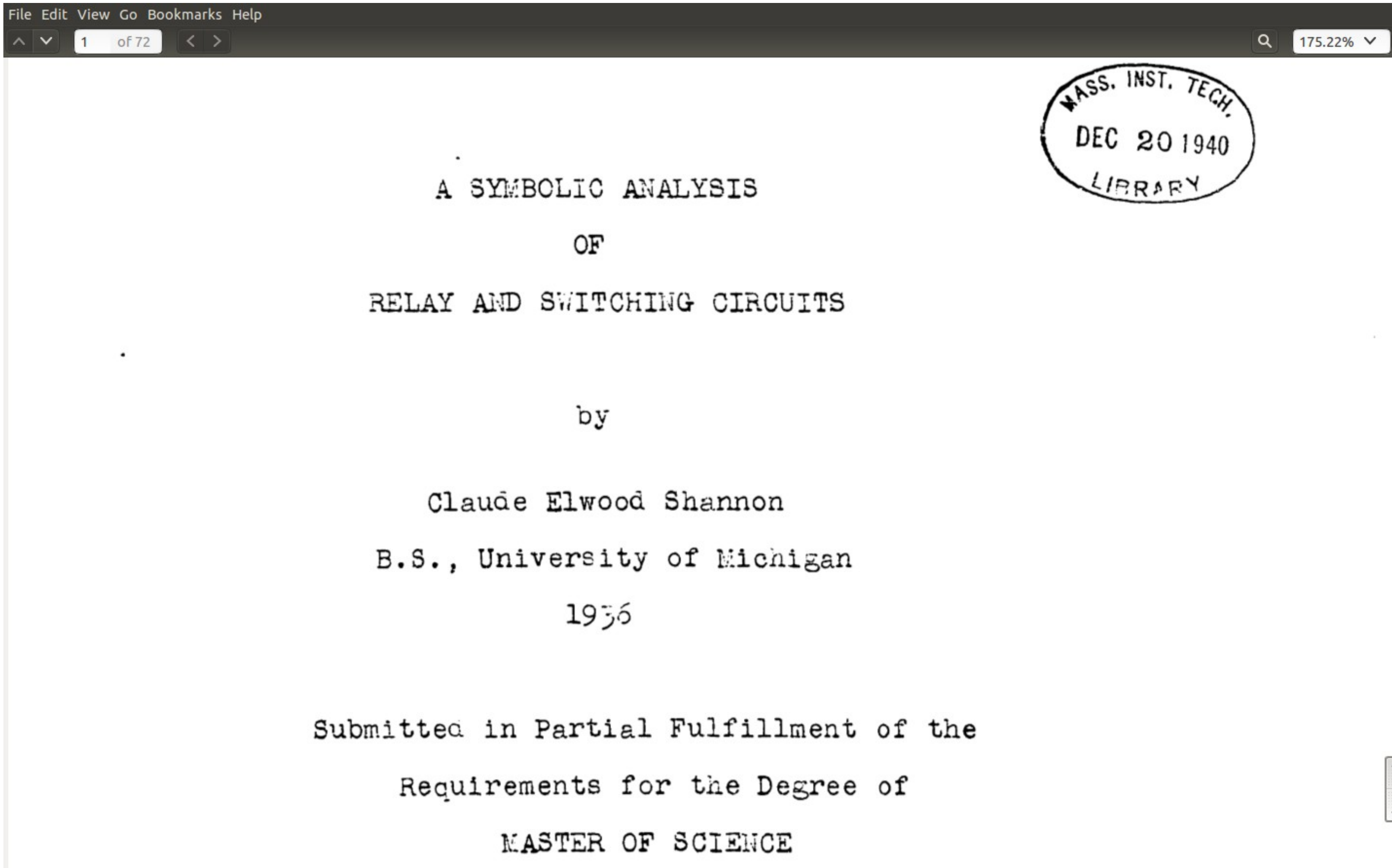


- Capacitors (like batteries) can store electrical charges
- Charge stored on a capacitor may also denote numbers
  - Capacitor has low charge = value 0 stored on it.
  - Capacitor has high charge = value 1 stored on it.
  - Once charge is stored on a capacitor, it persists. Memory
- Building blocks of DRAMs (Dynamic Random Access Memory)

Representing Numbers

- How to represent numbers using this capability?
- Key idea : Binary number system
  - Represent all numbers using only 1's and 0's
  - Also called "Bits": "Binary digits"
- Details on conversion in next lecture
  - For now assume that all decimal numbers can be converted into binary numbers...i.e. into a sequence of 1' s and 0's

The Most Celebrated Masters Thesis Ever (1)



A SYMBOLIC ANALYSIS

OF

RELAY AND SWITCHING CIRCUITS

by

Claude Elwood Shannon

B.S., University of Michigan

1936

Submitted in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

The Most Celebrated Masters Thesis Ever (2)

File Edit View Go Bookmarks Help

5 of 72 175.22%

The method of solution of these problems which will be developed here may be described briefly as follows: Any circuit is represented by a set of equations, the terms of the equations representing the various relays and switches of the circuit. A calculus is developed for manipulating these equations by simple mathematical processes, most of which are similar to ordinary algebraic algorisms. This calculus is shown to be exactly analogous to the Calculus of Propositions used in the symbolic study of logic. For the synthesis problem the desired characteristics are first written as a system of equations, and the equations are then manipulated into the form representing the simplest circuit. The circuit may

The Most Celebrated Masters Thesis Ever (3)

Fundamental Definitions and Postulates. We shall limit our treatment to circuits containing only relay contacts and switches, and therefore at any given time the circuit between any two terminals must be either open (infinite impedance) or closed (zero impedance). Let us associate a symbol X_{ab} or more simply X, with the terminals a and b. This variable, a function of time, will be called the hinderance of the two terminal circuit a-b. The symbol 0 (zero) will be used to represent the hinderance of a closed circuit, and the symbol 1 (unity) to represent the hinderance of an open circuit. Thus when the circuit a-b is open $X_{ab} = 1$ and when closed $X_{ab} = 0$. Two hinderances X_{ab} and X_{cd} will be said to be equal

Representing Numbers

Key idea:

Store each bit of the number on a separate capacitor

Example: 25 Decimal = 11001 binary

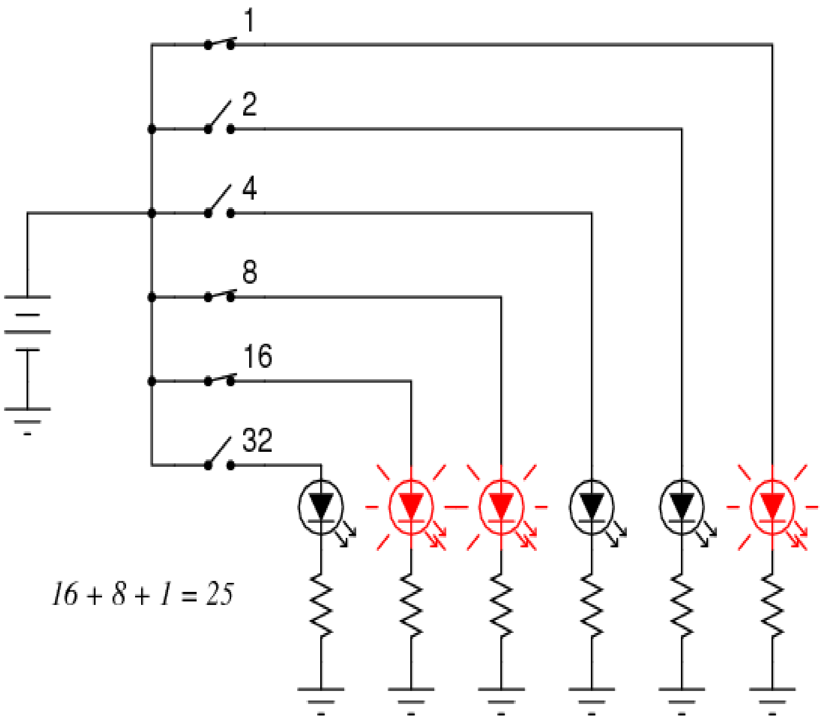
- Use 5 capacitors
- Store high charge on 1st, 2nd, 5th, and low charge on 3rd, 4th

To transmit 25 from one part of the computer to another

- Use 5 wires and raise the wires to appropriate voltages at one end.
- Sense the voltages at the other end

Bit = wire/capacitor, depending upon context

A digital circuit representing the number 25:



Bits, bytes, half-words, words

Bit = 1 capacitor/wire

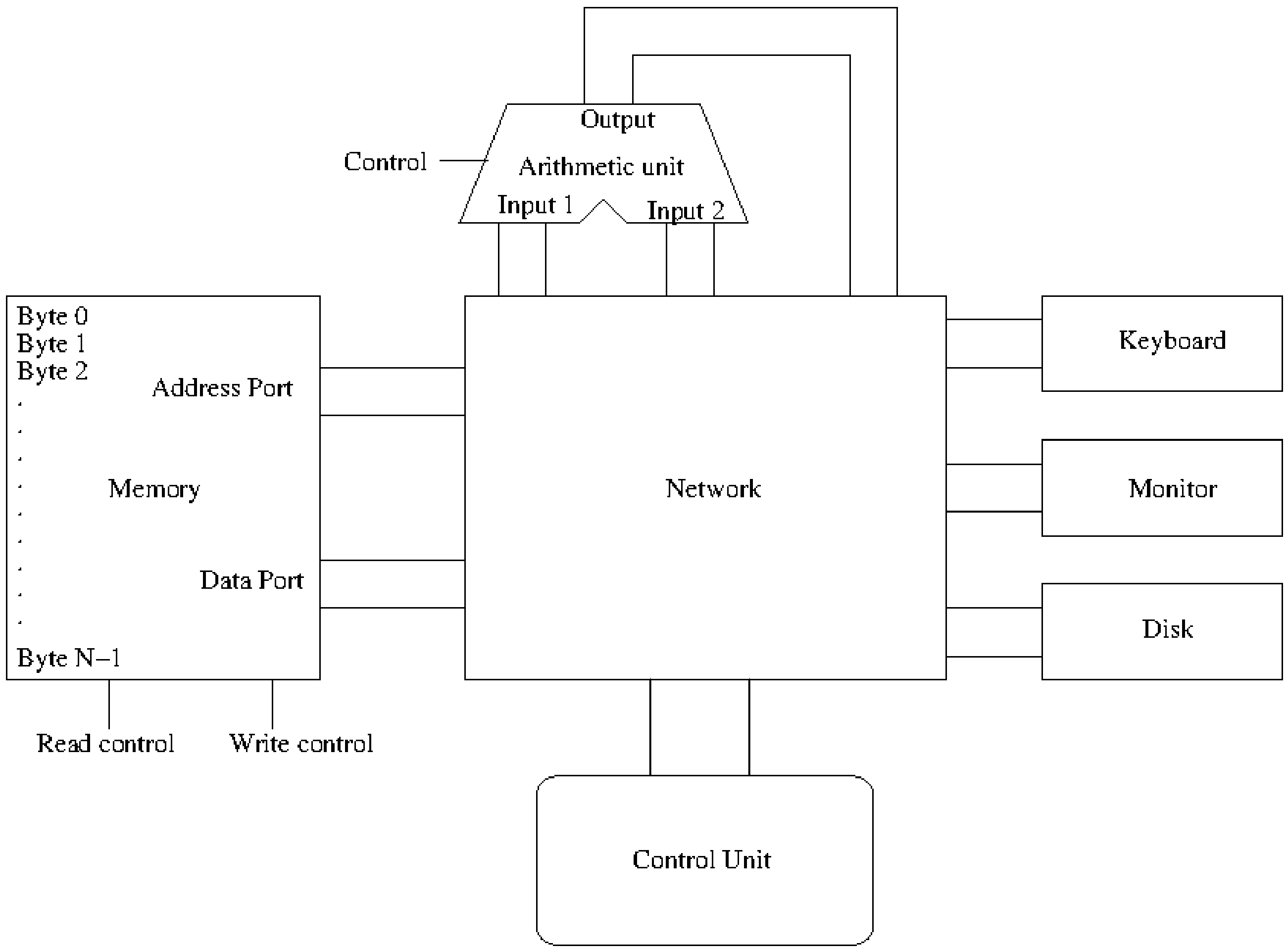
byte = 8 capacitors/wires

half-word = 16 capacitors/wires

word = 32 capacitors/wires

double word = 64 capacitors/wires

Organization of a computer



Memory

Organized into bytes (groups of 8 capacitors)

Memories of present day computers contain few Gigabytes, Giga=2^{30}

Each byte in the memory is assigned a distinct number, or an address.

In a memory with N bytes, the addresses go from 0 to N-1

0	**0**	**0**	**0**	**1**	**1**	**0**	**1**	**0**
1								
2								
3								
4								
5	1	0	1	1	1	1	0	1
6								
7								
8								
9								

Memory Ports

Memory has 3 ports: address port, data port, control port.

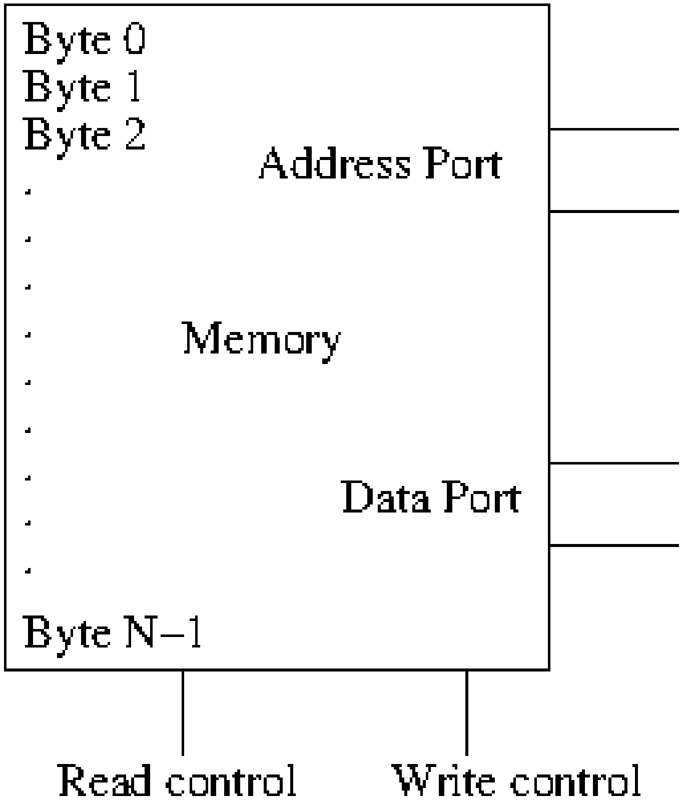
Address port consists of log N wires. (N = number of bytes in memory)

You can place numbers 0..N-1 on address port.

Control Port may be just 1 wire.

- Wire = 0: Memory to perform read operation.
- Wire = 1: Memory to perform write operation.

Data port will have w wires, where w is a multiple of 8. Say w=8m.



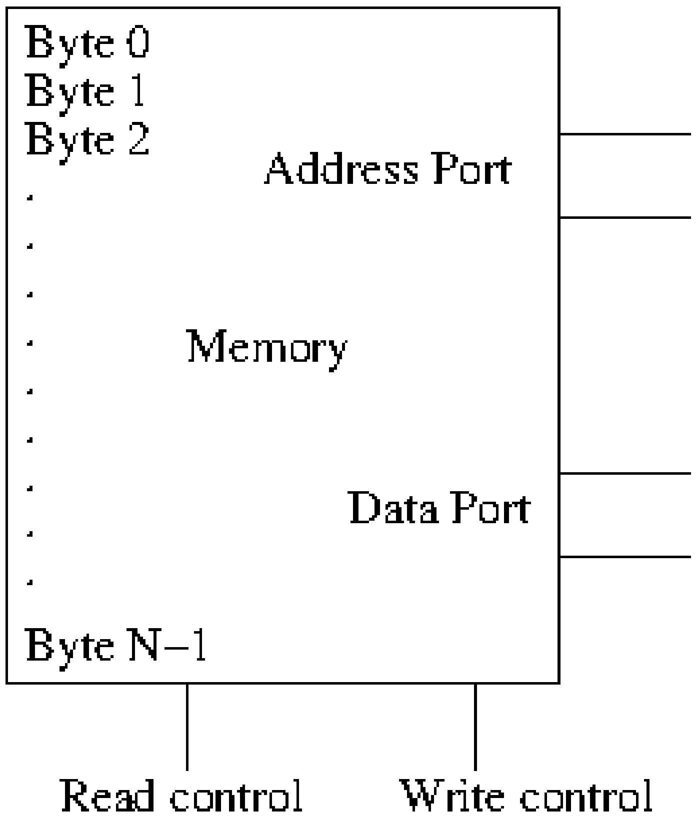
Write Operation

Control Port must be set to 1.

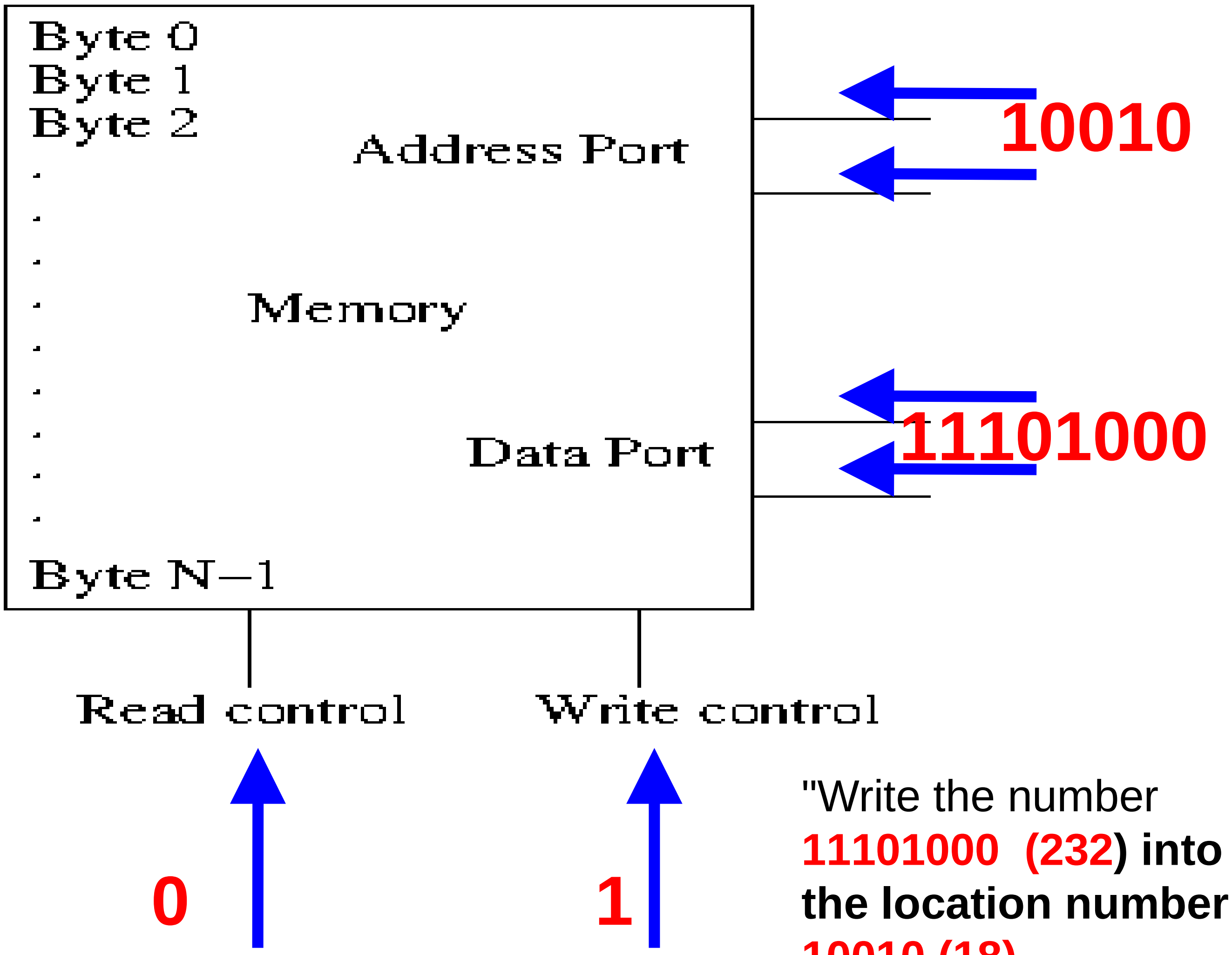
If A is placed on the address port, and D on data port, then D will be stored in the m bytes starting at byte A.

(Remember that the data port had 8m wires, and so m bytes are available on the data port)

Yes, it is possible to build circuits that can do this!



Write Operation



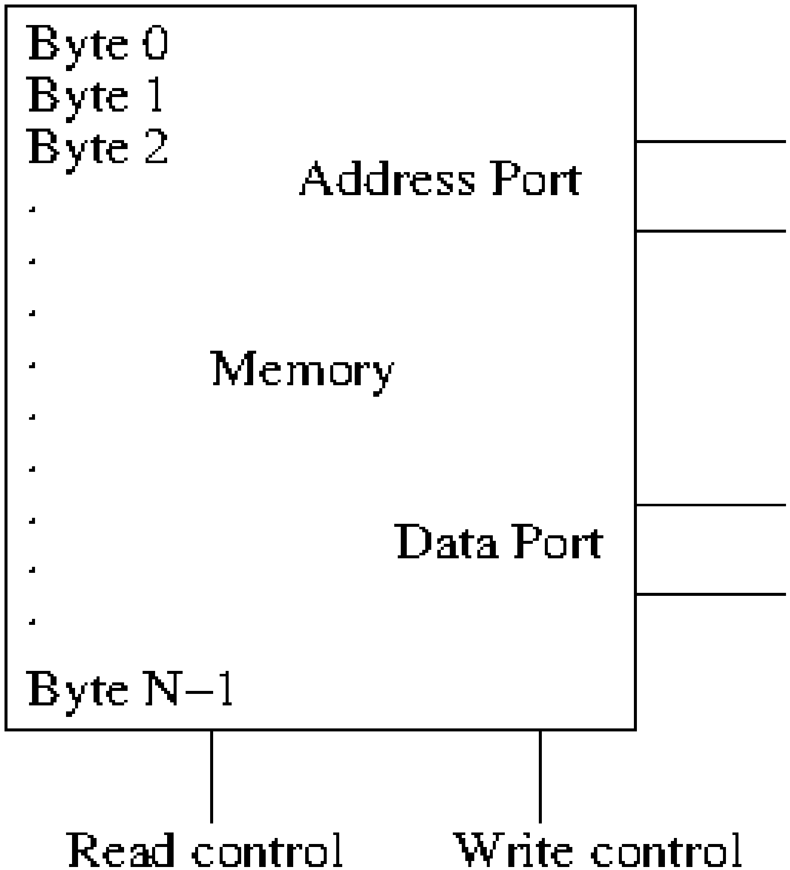
"Write the number **11101000 (232) into the location number 10010 (18)**

Read Operation

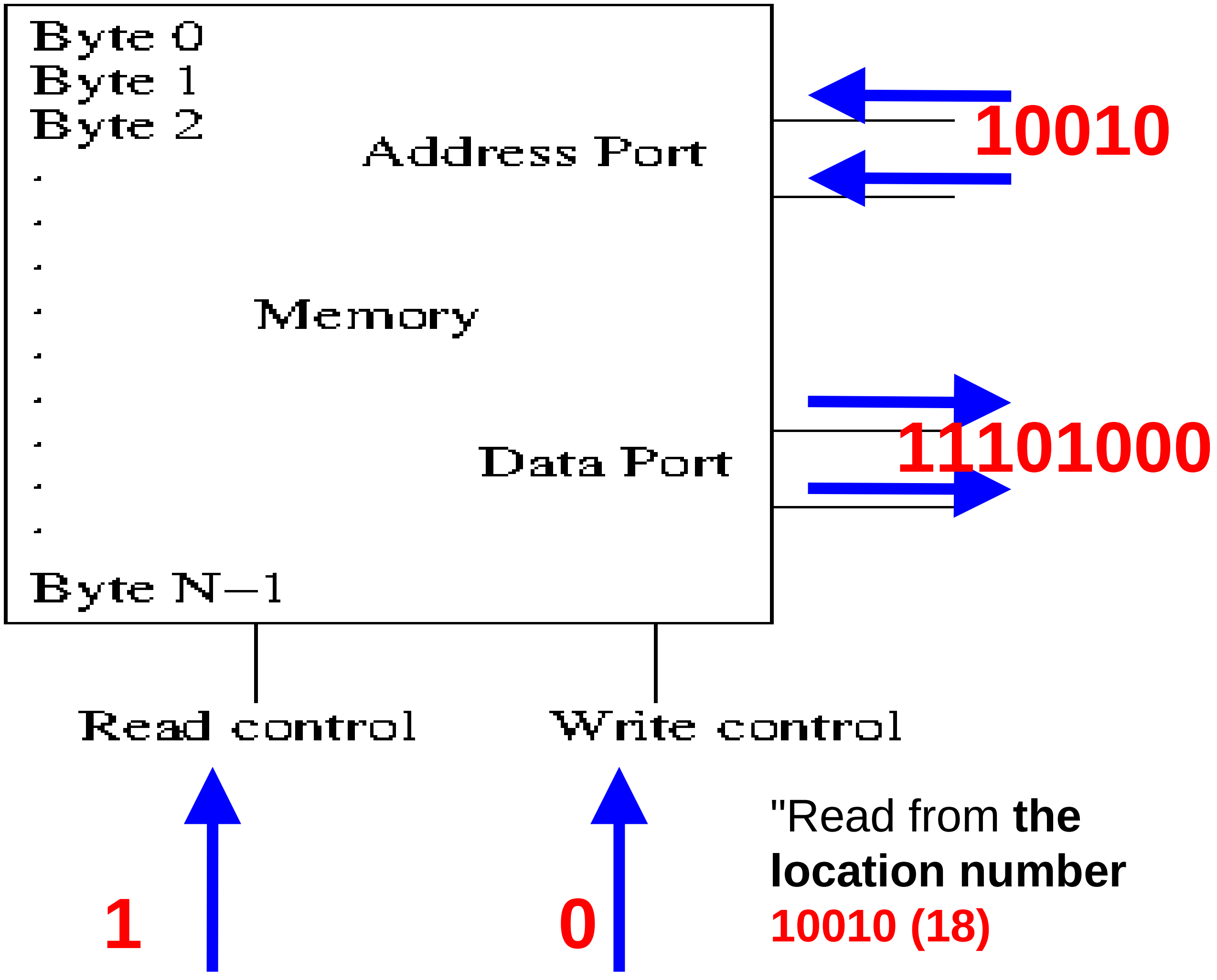
Control Port must be set to 0.

If A is placed on the address port, the m bytes starting at byte A will appear on the data port

(Data port has 8m wires, and so m bytes will fit on the data port)

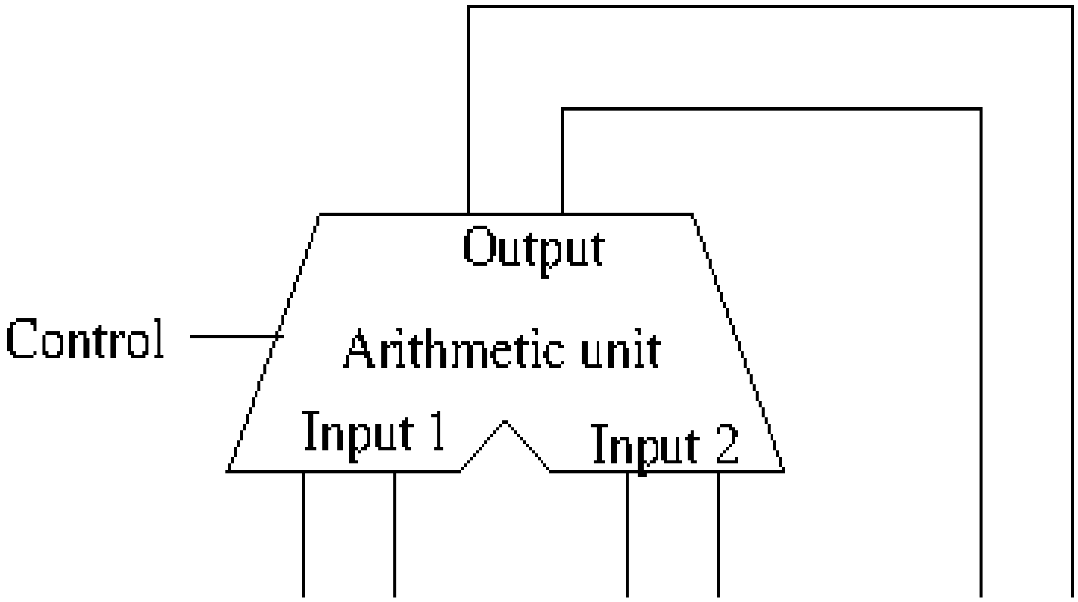


Read Operation



"Read from **the location number** **10010 (18)**

Arithmetic Unit



Ports: Input1, Input2, Output,Control

Typically Input1, Input2, Output will consist of w wires, w = size of memory data port

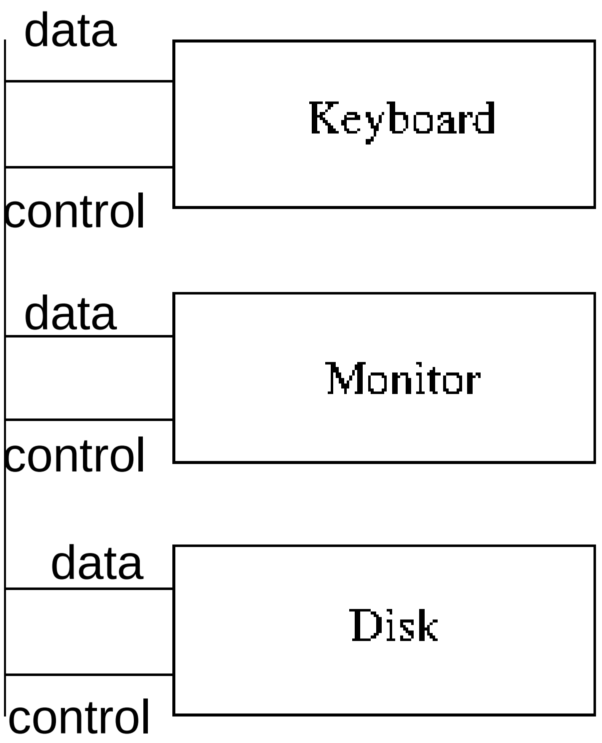
Control = several wires. Number appearing on the control wires will say what operation should be performed.

1 cycle after values are placed on Control, the Output will take the commanded value: sum, product, …

Peripherals: keyboard, screen, disk…

Also have control port and data port like organization.

Depending upon value placed on control port, the peripheral decides what to do with the value on the data port, or itself places values on the data port.



Control Unit

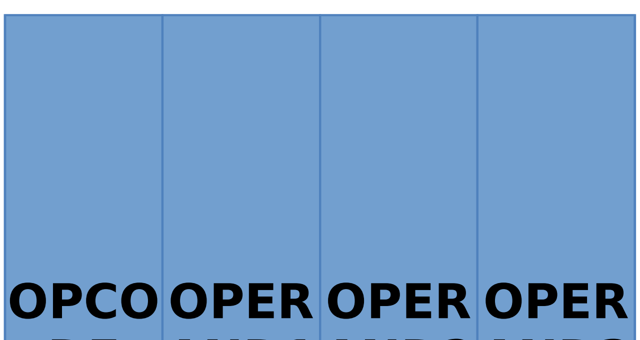
Tells other parts of the computer what to do.

Sends numbers on control wires of each unit

The control unit decides what to tell other units by reading a “machine language program” from the memory of the computer.

Machine language progam

- Program = Sequence of instructions
- Instruction = sequence of numbers
  - First number is OPERATION CODE (OPCODE). This is the code that tells the Control Unit what OPERATION should do.
  - Subsquent numbers are OPERANDS. These are "arguments" to the operation.



Example

57	**100**	**200**	**300**

7 might mean:

umbers following 57 as addresses.

at the first two addresses and send metic unit.

ithmetic unit to perform multiplication by sending appropriate number on its control wires.

– Store the result from the arithmetic unit into the word at the third address

Machine language progam

Program = Sequence of instructions

Instruction = sequence of numbers, first of which is an “operation code” which denotes what operation to perform

Example: operation code 57 might mean:

- Interpret the 3 numbers following 57 as addresses.
- Read the words at the first two addresses and send them to the Arithmetic unit.
- Command the arithmetic unit to perform multiplication by sending appropriate number on its control wires.
- Store the result from the arithmetic unit into the word at the third address

The sequence 57, 100, 200, 300 is an instruction that would cause the product of the numbers stored in addresses 100, 200 to be stored in the address 300.

The operation codes are defined by the computer designer.

- She will assign a different code for each operation that she would like the computer to perform.
- Example: 58 might mean the same thing as above, except that the numbers would be added.

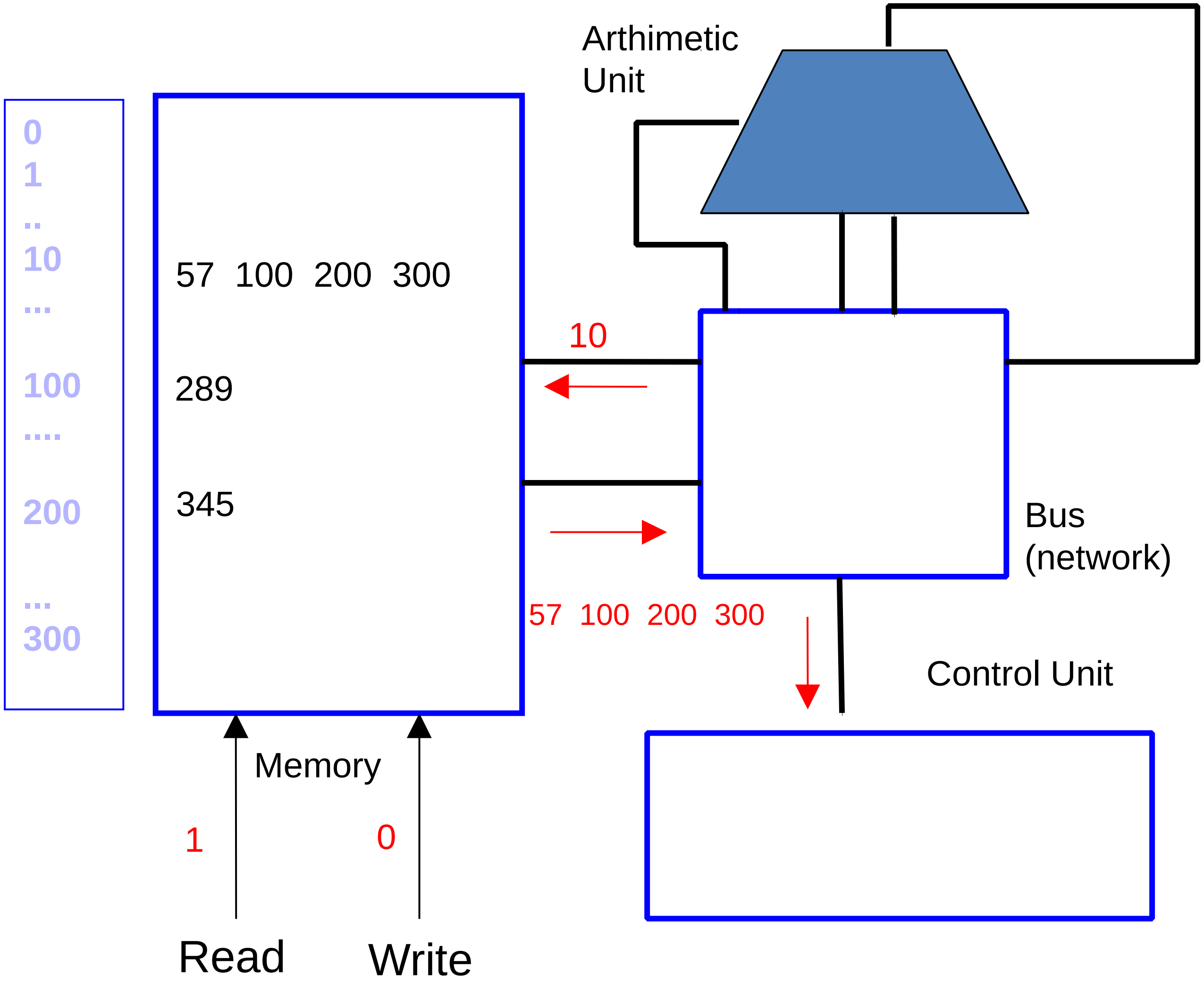
Control unit operation

Control unit must be told where the machine language program is in memory.

The control unit then fetches the instructions constituting the program, interprets the codes, and performs the required operation.

After one instruction is fetched and executed, it fetches the next instruction and repeats the process.

Putting it together

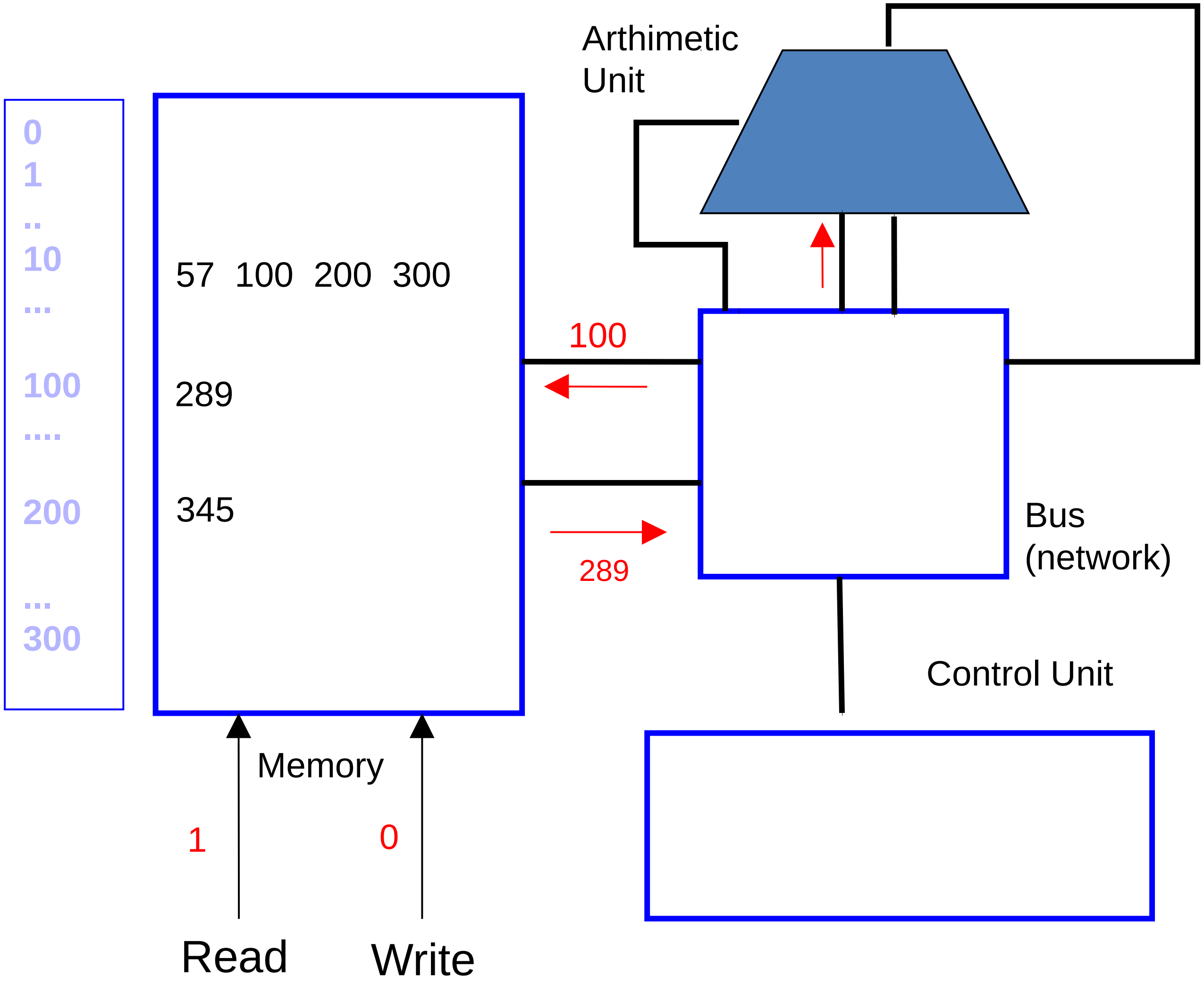


1. Control unit is told the address of the instruction to fetch (e.g. instruction is at location 10)

2. A read operation is performed on memory location 10

3. instruction 57 100 200 300 is now "loaded" into the control unit

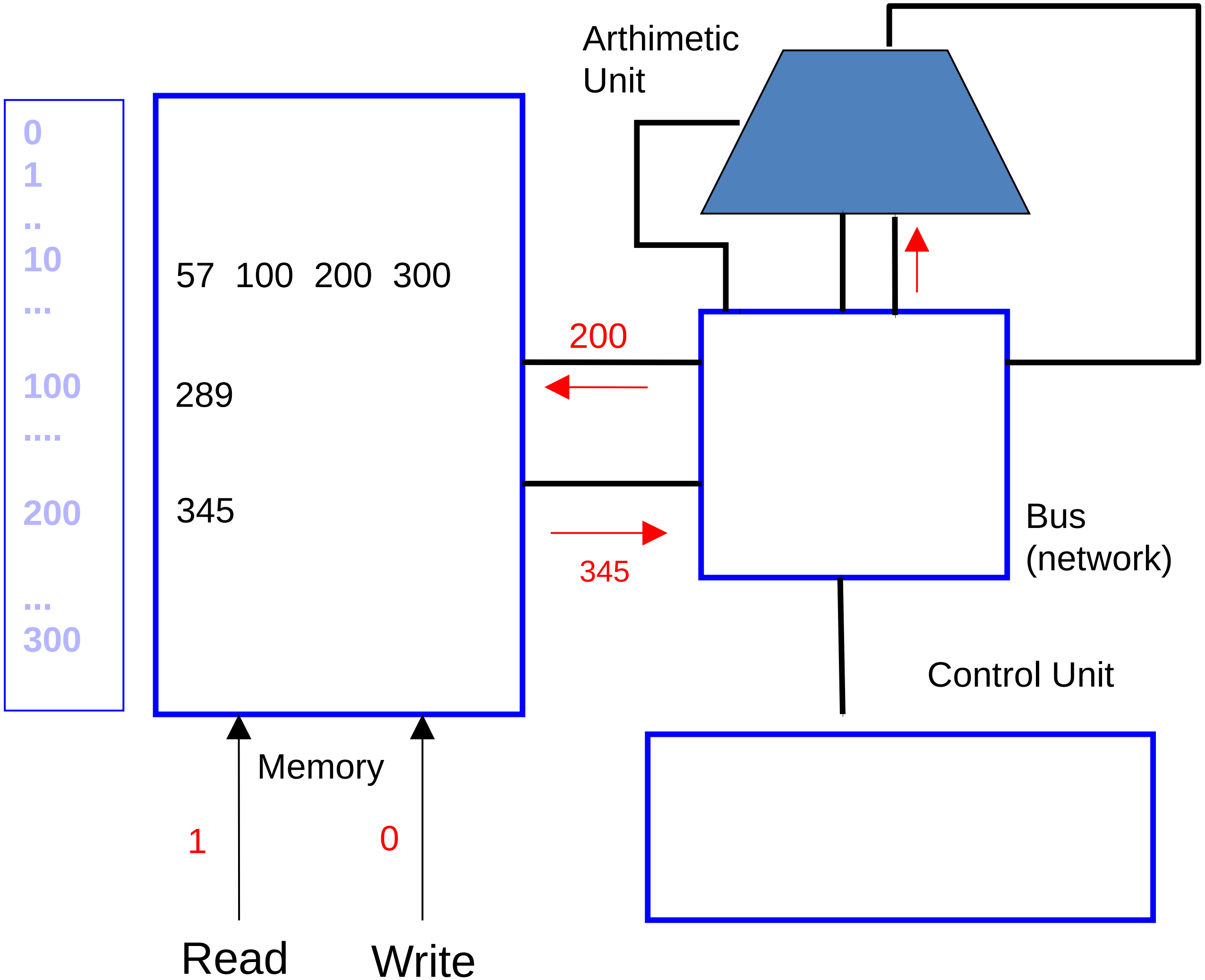
Putting it together



1. Now control unit performs a read operation of address 100

2. The number 289 is sent to input 1 of arithmetic unit

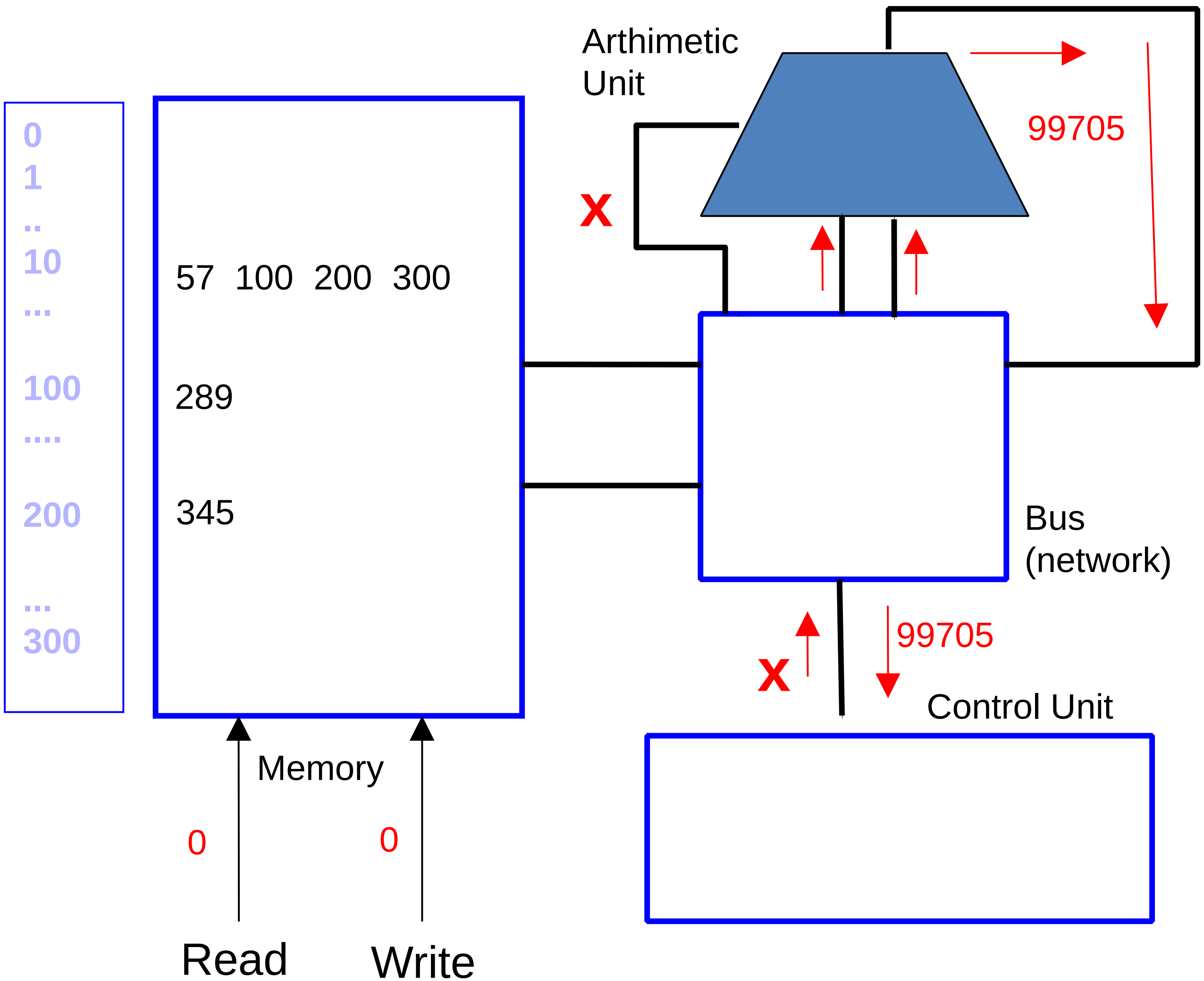
Putting it together



1. Now control unit performs a read operation of address 200

2. The number 345 is sent to input 2 of arithmetic unit

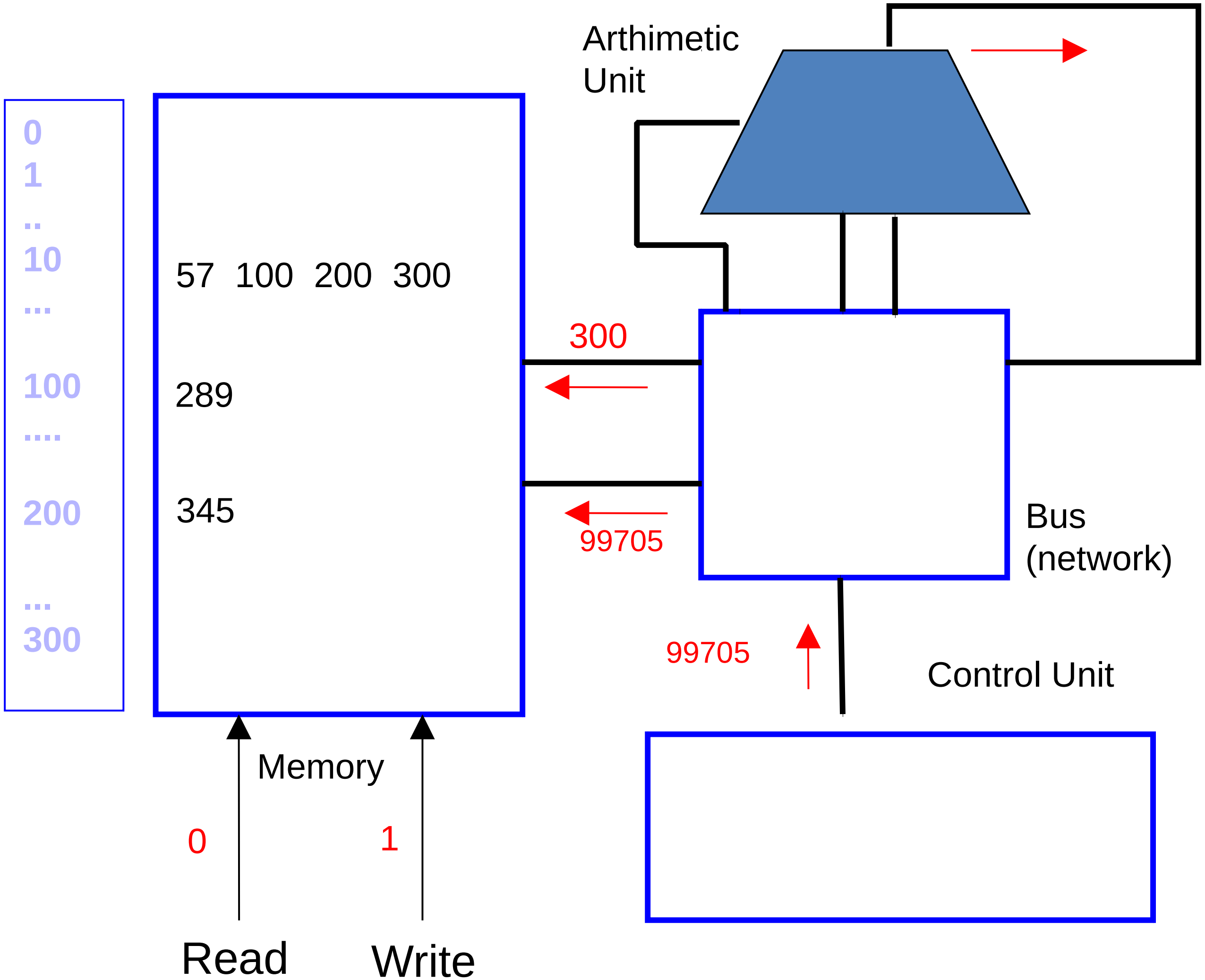
Putting it together



1. Now control unit instructs the arithmetic unit to perform a multiply operation (the "control" wires are set to the operation code of "multiply")

2. Control unit stores the product (289 x 345 = 99705) temporarily inside its own unit

Putting it together



1. Now control unit performs a write operation on address 300

2. The number 99705 is sent on the data port of memory and 300 is sent on the address port of the memory

Instruction execution is COMPLETE

Machine language program example

Example:

57, 100, 100, 100

57, 100, 100, 100

This contains two instructions.

Both instructions cause the word at address 100 to be multiplied by itself and the result stored back in address 100.

After executing the first instruction, address 100 would contain the square of the number that was present before.

The second operation would repeat the squaring operation.

Thus this is a machine language program to compute the fourth power of a number.

More complex instructions

Example 1: operation code 59 might mean: “Shut down the computer”

Example 2: operation code 60 might mean:

Interpret the next number in the program as an address

Instead of next executing the instruction following the current one, execute the instruction starting at this address.

Example 3: operation code 61 might mean:

Interpret the next number in the program as an address.

If the last result produced by the arithmetic unit was 0, then execute the instruction starting at this address.

If the last result produced by the arithmetic unit was not 0, then execute the instruction following the current one.

Analogous to the repeat statement of chapter 1.

Using such instructions, we will be able to perform an operation 100s of times without making our machine language very long.

Machine language programs and C++ programs

On early computers, you would have to write machine language programs.

- Find out what operation you want.
- Look up the manual and find its code.
- Enter the code into the memory of the computer.
- Repeat.
- Process is laborious, error-prone.

Modern computers also need machine language programs.

- You write C++ program.
- A prewritten program, "compiler", translates your C++ program to a machine language program.
- Another program, "loader", will load it into the memory and start its execution.

Concluding Remarks

Key idea 1: use numerical codes to represent non numerical entities

letters and other symbols: ASCII code

operations to perform on the computer: Operation codes

Key idea 2: Current/charge/voltage values in the computer circuits represent bits (0 or 1).

Concluding Remarks

Memory is organized into bytes. Each byte has an address.

What the computer does is determined by a machine language program that must be stored in the memory.

Computer users do not need to themselves write a machine language program, but can write a C++ program which is then compiled by a compiler.