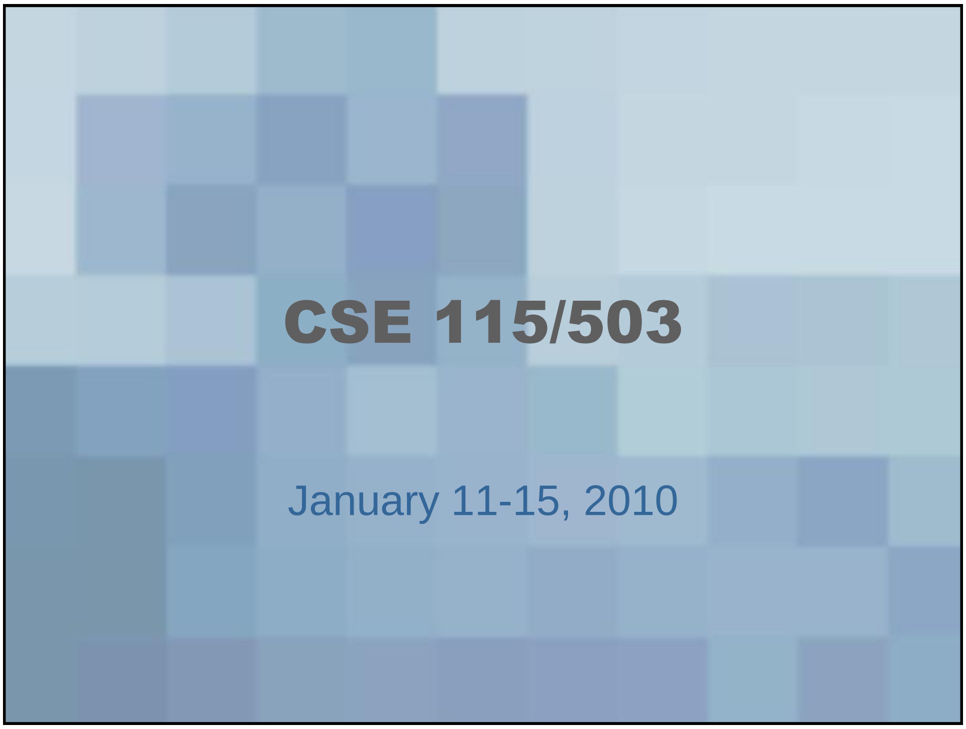
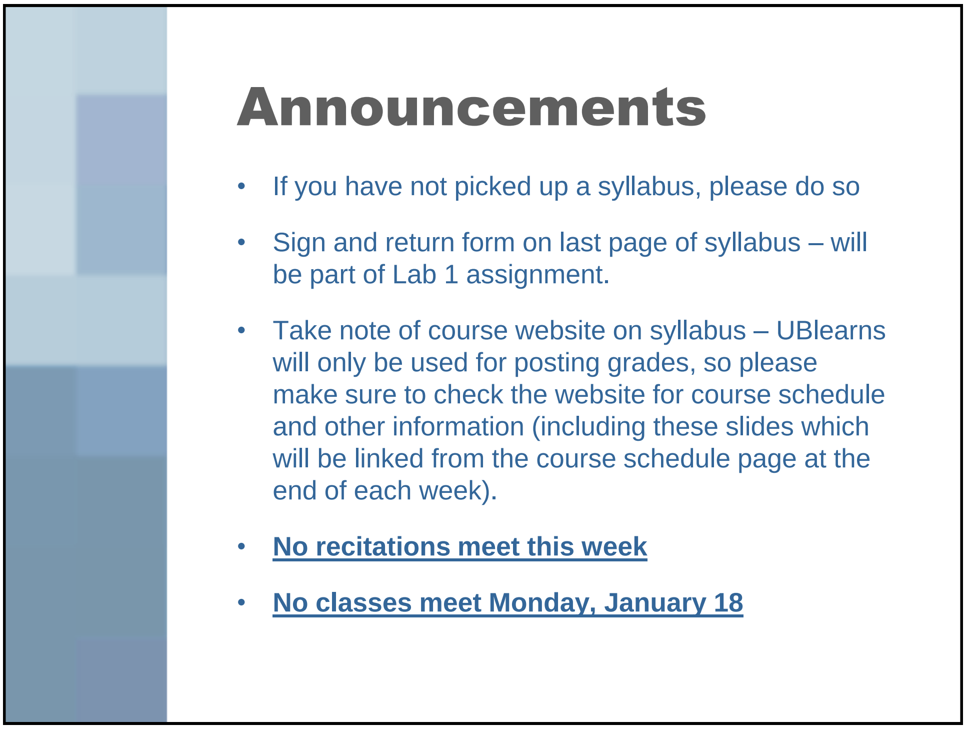


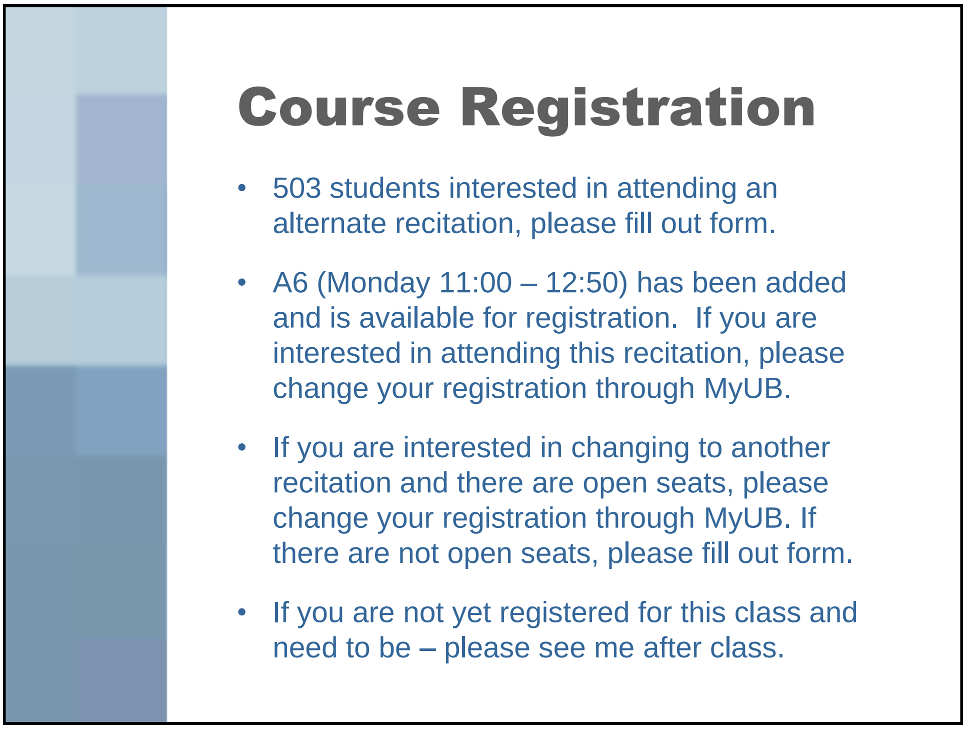
CSE 115/503

January 11-15, 2010



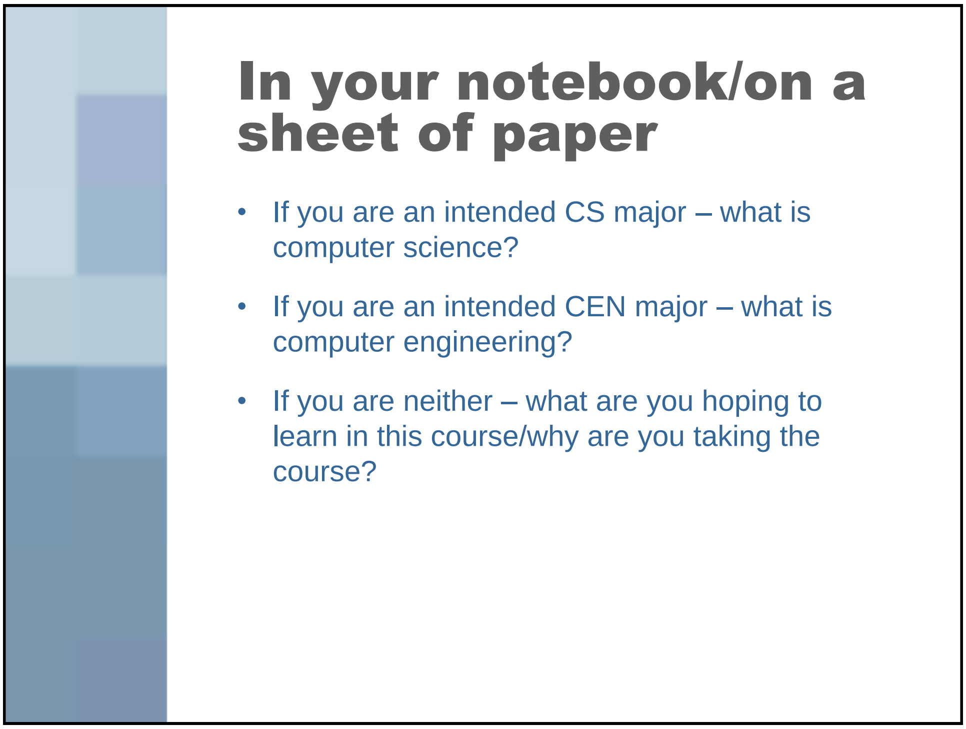
Announcements

- If you have not picked up a syllabus, please do so
- Sign and return form on last page of syllabus – will be part of Lab 1 assignment.
- Take note of course website on syllabus – UBlearns will only be used for posting grades, so please make sure to check the website for course schedule and other information (including these slides which will be linked from the course schedule page at the end of each week).
- **No recitations meet this week**
- **No classes meet Monday, January 18**



Course Registration

- 503 students interested in attending an alternate recitation, please fill out form.
- A6 (Monday 11:00 – 12:50) has been added and is available for registration. If you are interested in attending this recitation, please change your registration through MyUB.
- If you are interested in changing to another recitation and there are open seats, please change your registration through MyUB. If there are not open seats, please fill out form.
- If you are not yet registered for this class and need to be – please see me after class.



In your notebook/on a sheet of paper

- If you are an intended CS major – what is computer science?
- If you are an intended CEN major – what is computer engineering?
- If you are neither – what are you hoping to learn in this course/why are you taking the course?

Answers

- What is computer science?

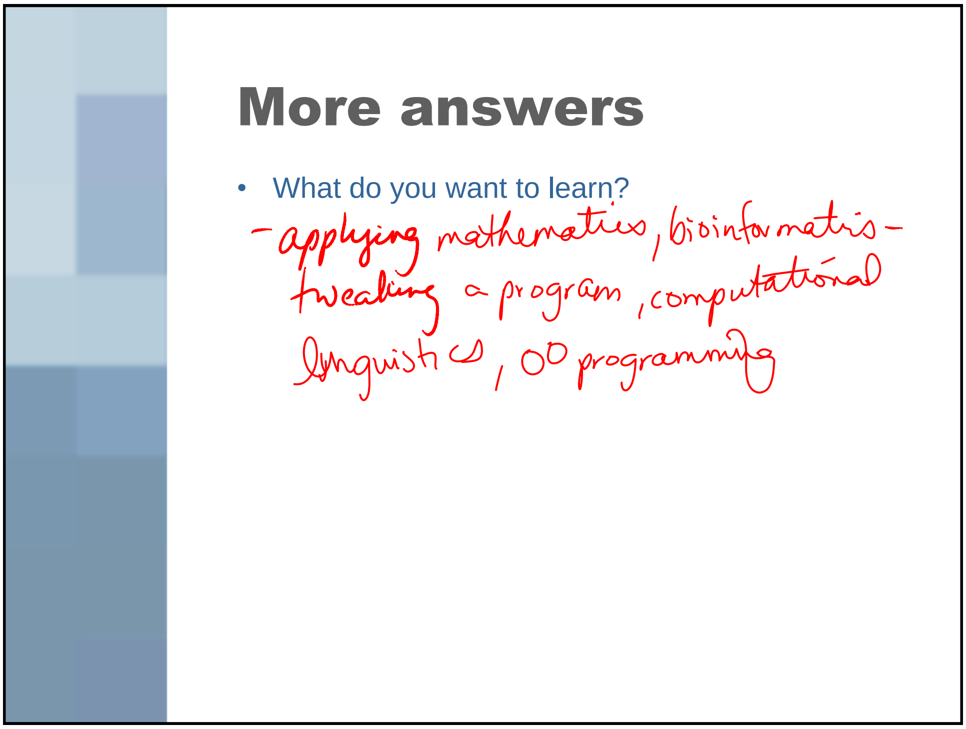
Software, how the computer runs, computation, algorithms, theory, problem solving with a computer, data integration & exchange; computer related technology

- What is computer engineering?

implementation, applicable, less technical than EE, what's under the hood, physical components of computer

So – what are they?

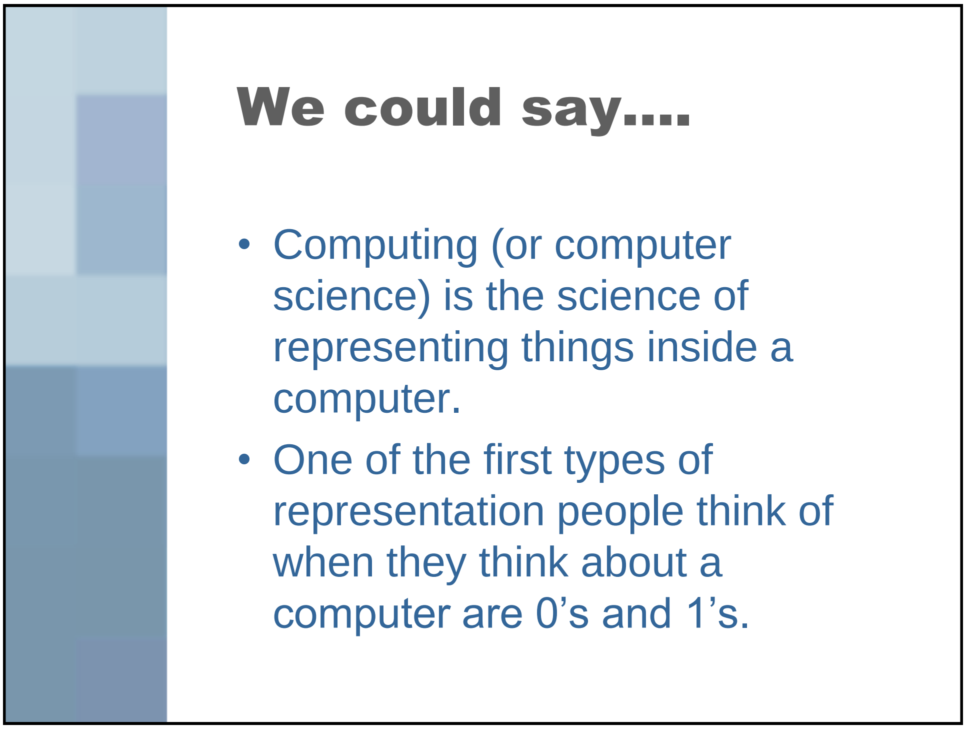
- Well, they are all these things. There isn't a universally agreed-upon definition for these two terms, but the areas tend to overlap in many cases. In computing, your eventual job description is sometimes of your own creation.



More answers

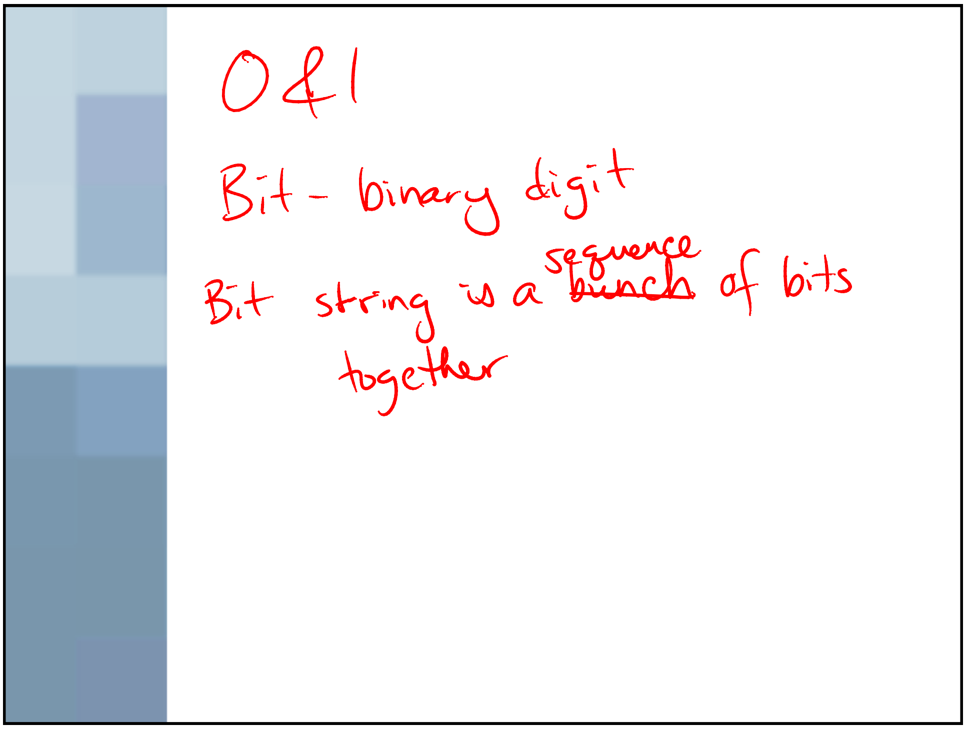
- What do you want to learn?

- applying mathematics, bioinformatics - tweaking a program, computational linguistics, OO programming



We could say….

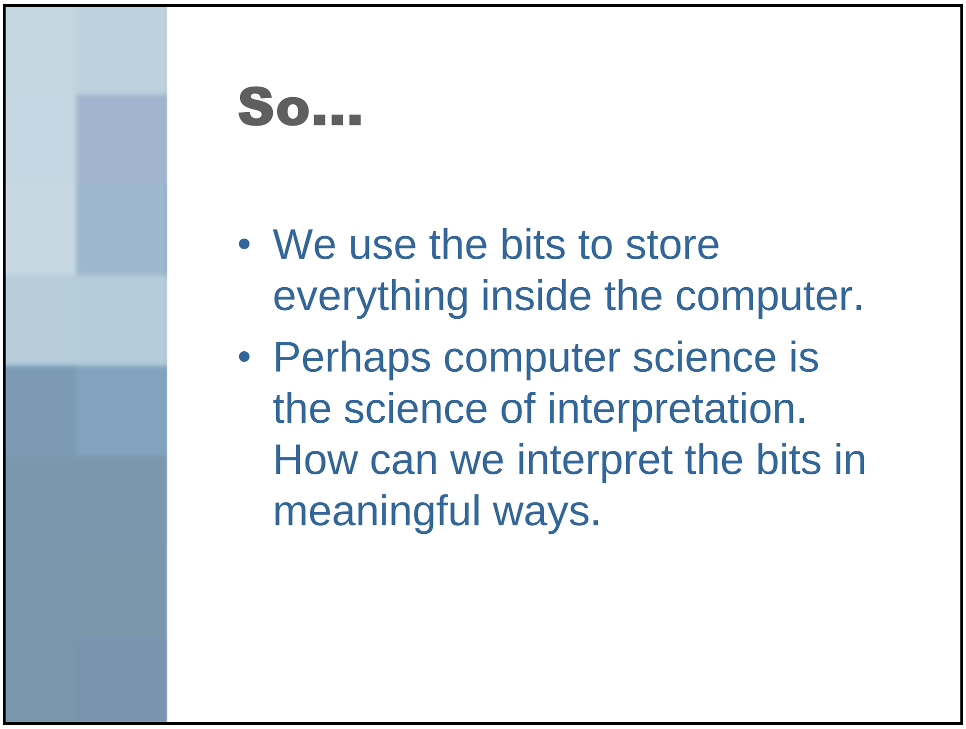
- Computing (or computer science) is the science of representing things inside a computer.
- One of the first types of representation people think of when they think about a computer are 0's and 1's.



0 & 1

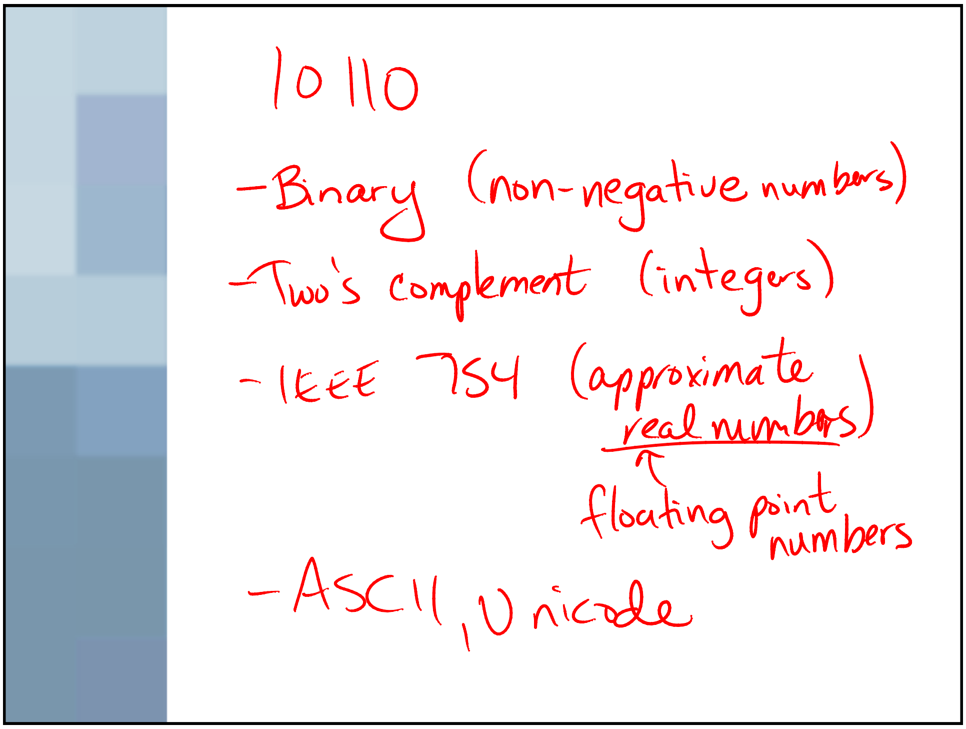
Bit - binary digit

Bit string is a ~~bunch~~ sequence of bits together



So…

- We use the bits to store everything inside the computer.
- Perhaps computer science is the science of interpretation. How can we interpret the bits in meaningful ways.

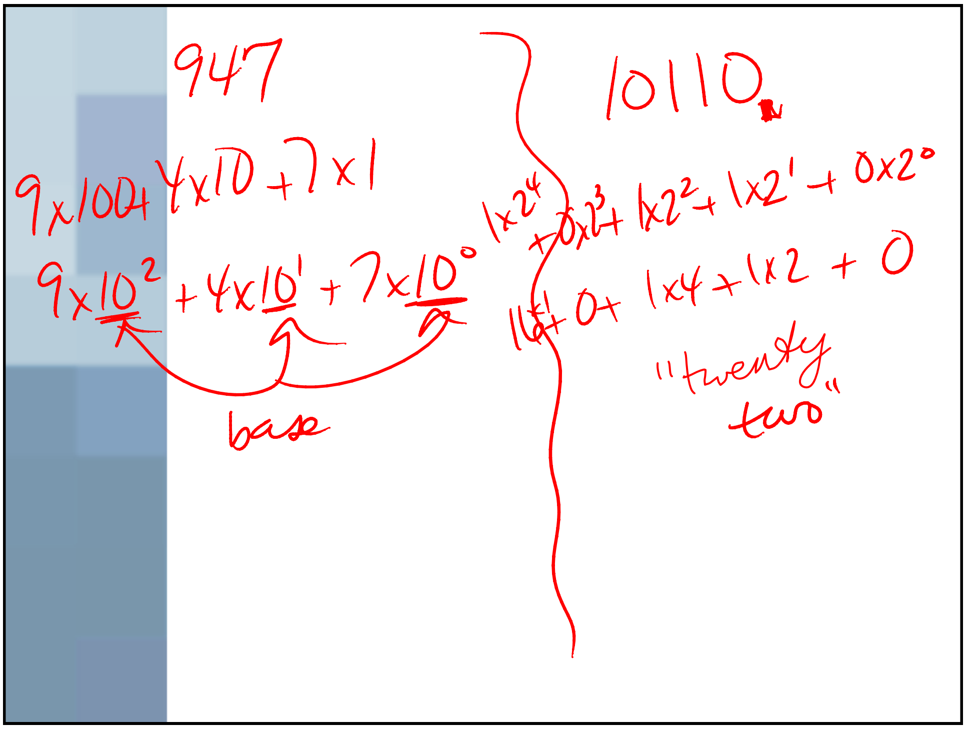


10110

- Binary (non-negative numbers)
- Two's complement (integers)
- IEEE 754 (approximate real numbers)

↑ floating point numbers

- ASCII, Unicode



947

$9 \times 100 + 4 \times 10 + 7 \times 1$

$9 \times \underline{10}^2 + 4 \times \underline{10}^1 + 7 \times \underline{10}^0$

base

10110_2

$1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$

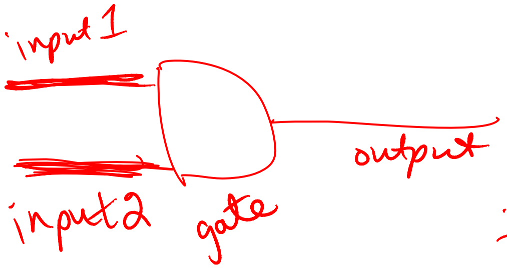
$16 + 0 + 1 \times 4 + 1 \times 2 + 0$

"twenty two"

Looking into the machine

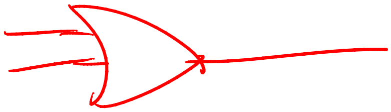
- We can say:
  - Computers are simple machines
  - Computers really only manipulate voltages

Gates are used to control voltage flow inside the machine



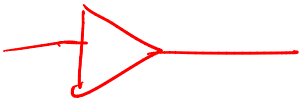
inputs	output
00	0
01	0
10	0
11	1

"or" gate



inputs	output
00	0
01	1
10	1
11	1

"not"



input	output
0	1
1	0

Fun fact

- It turns out that addition (and other arithmetic) in the binary number system is fairly simple

$0+0=0$

$0+1=1$

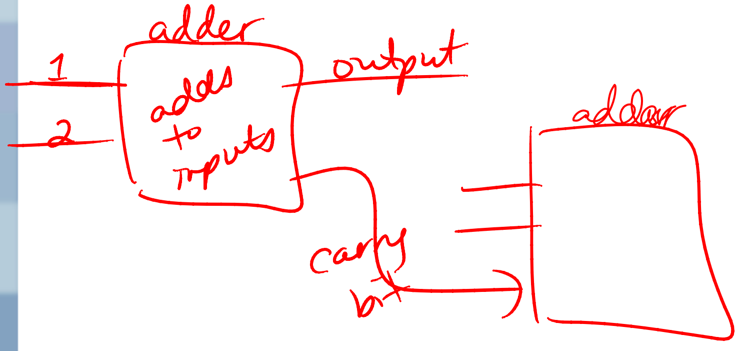
$1+0=1$

$1+1=10$

$1+1+1=11$

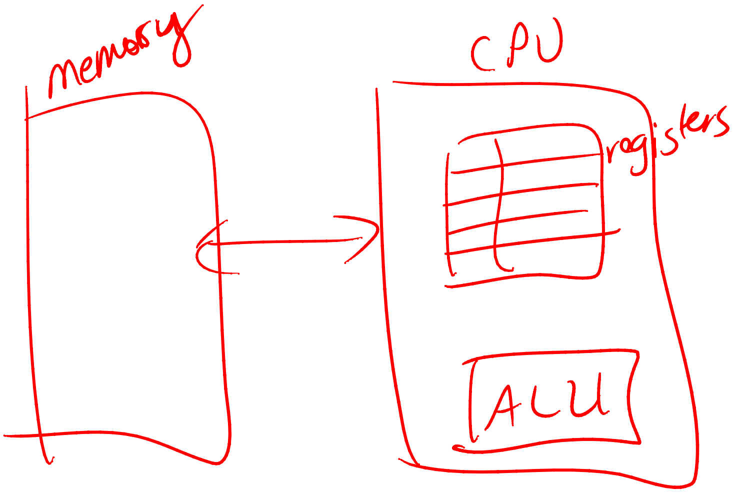
Therefore,

- We can easily teach the computer the “rules” of binary arithmetic



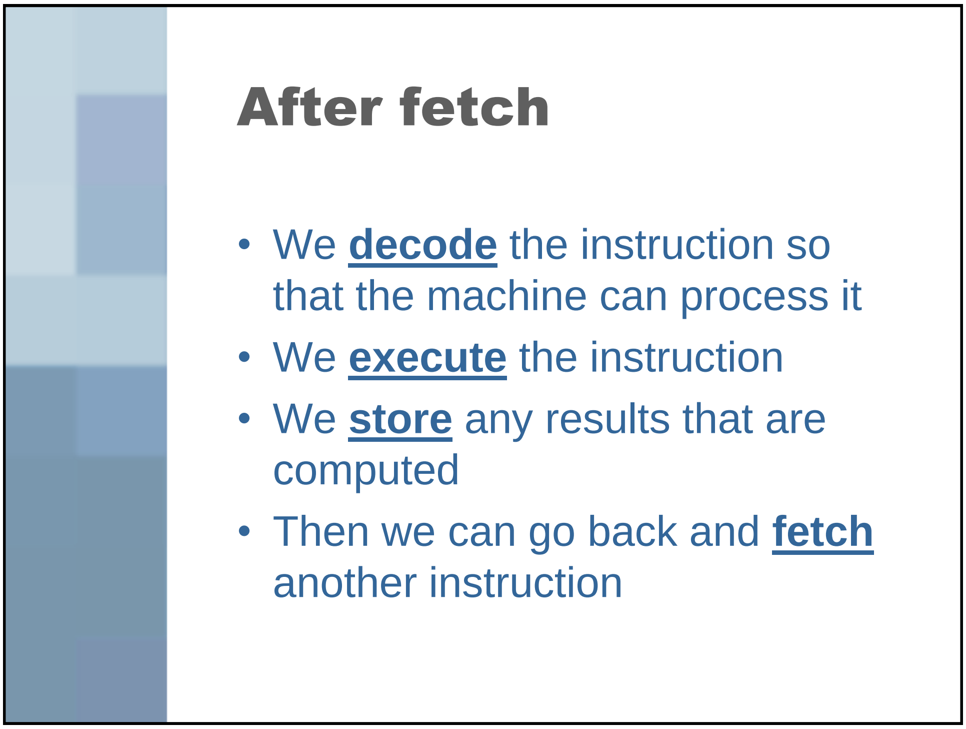
Then

- We can put the rules (circuits, gates, physical parts) into the ALU [Arithmetic Logic Unit] which is one of the two main parts of the CPU [Central Processing Unit].
  - The other part of the CPU is registers, which are places to store data



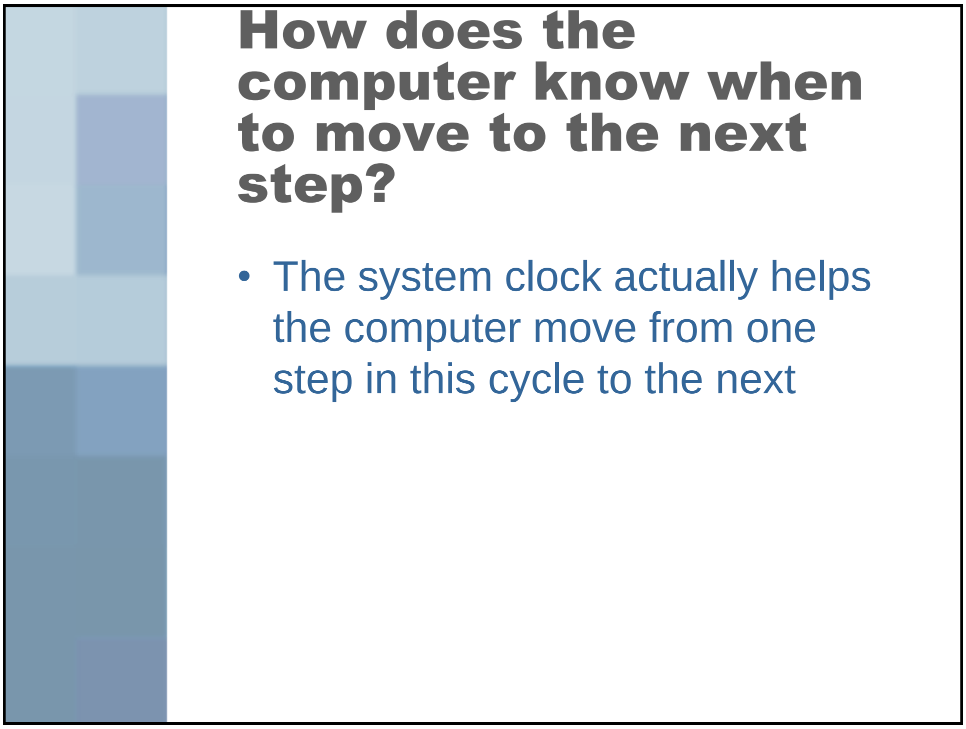
Getting the computer to execute instructions

- We store instructions inside the memory.
- When we are ready to do something, we **fetch** an instruction from memory and load it into the registers.



After fetch

- We **decode** the instruction so that the machine can process it
- We **execute** the instruction
- We **store** any results that are computed
- Then we can go back and **fetch** another instruction



How does the computer know when to move to the next step?

- The system clock actually helps the computer move from one step in this cycle to the next