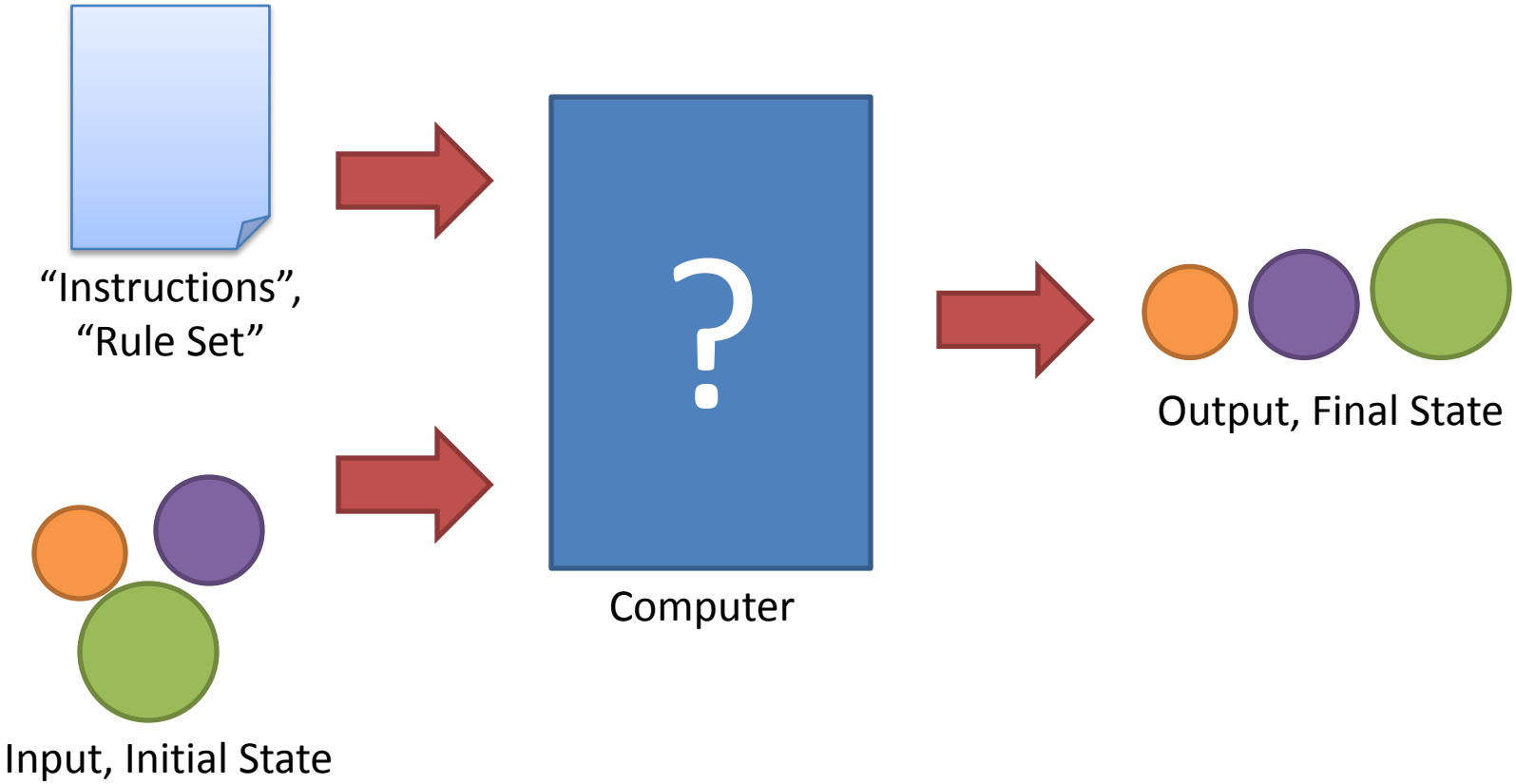
Computer Science 22: Object Oriented Programming

Lecture #1: Of Programs and Computer Programming

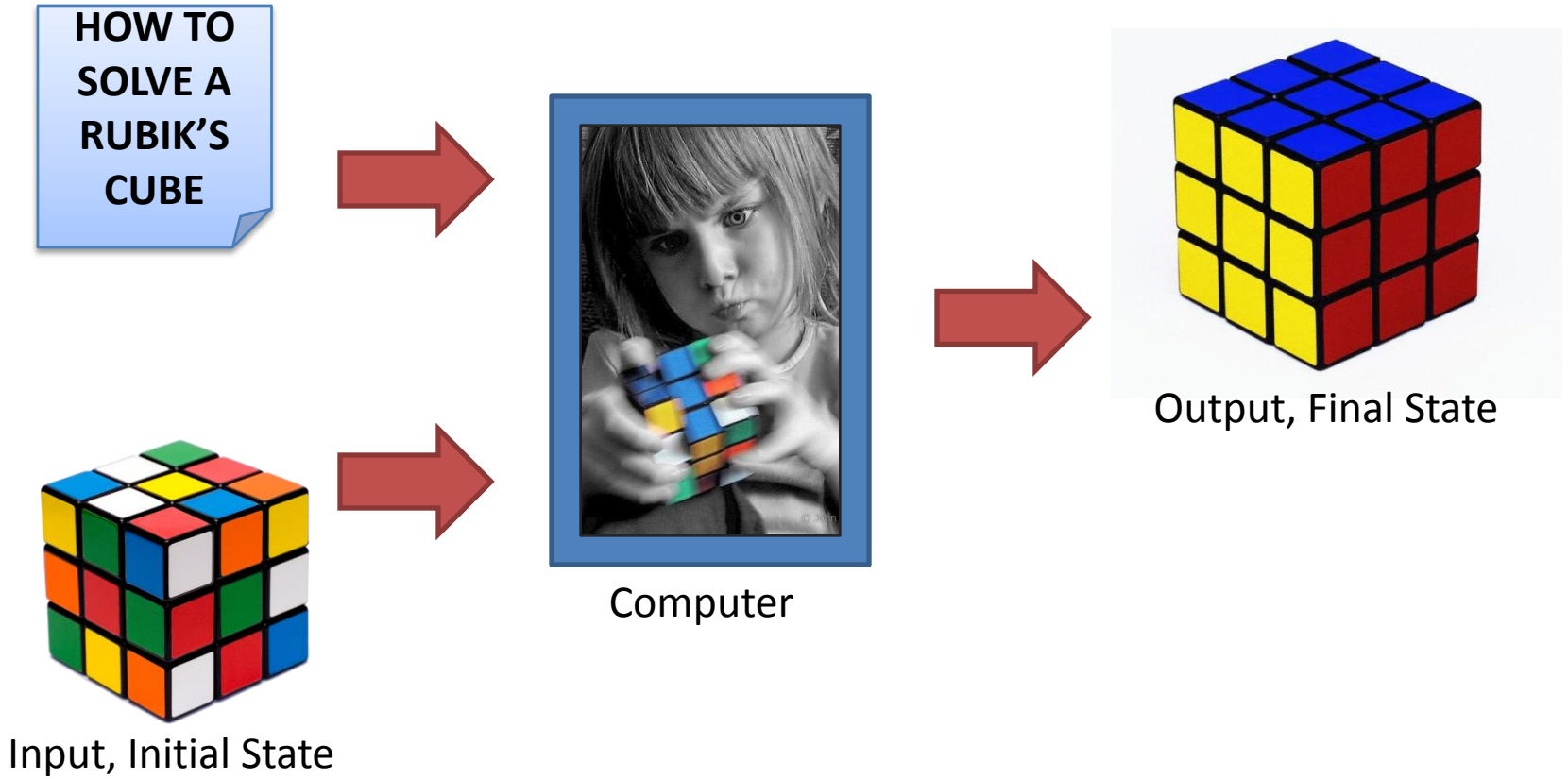
About this Lecture

- Definition of
  - Computer
  - Program/Computer Program
- Walk down memory lane
  - How computing has developed
  - How we arrived at the different **perspectives** or **paradigms** of programming
- The different **paradigms** of programming
- Object Oriented Programming

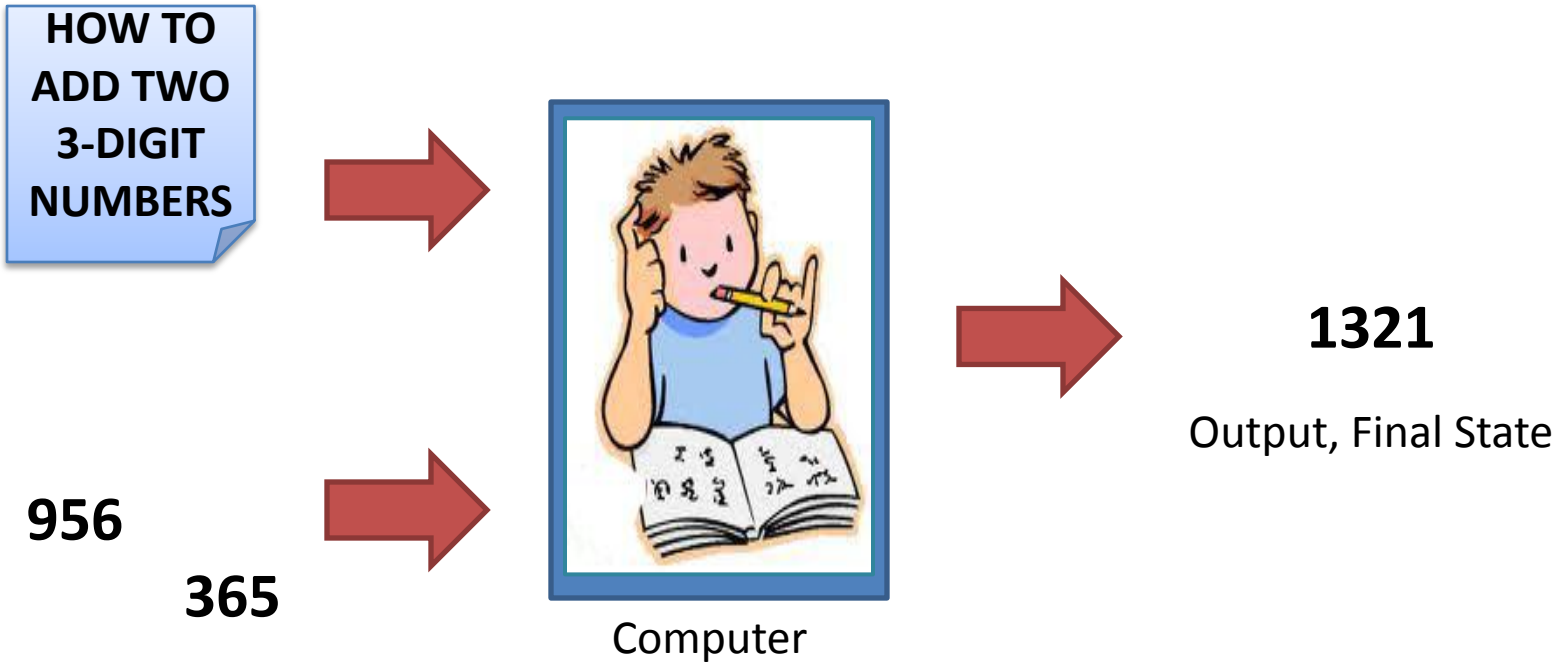
What is a Computer?



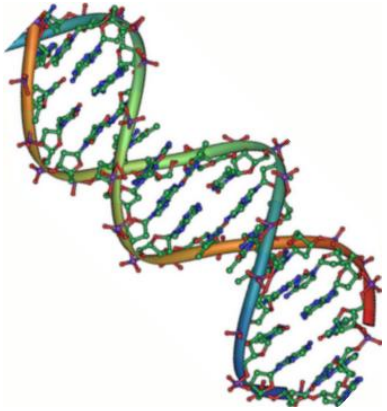
What is a Computer?

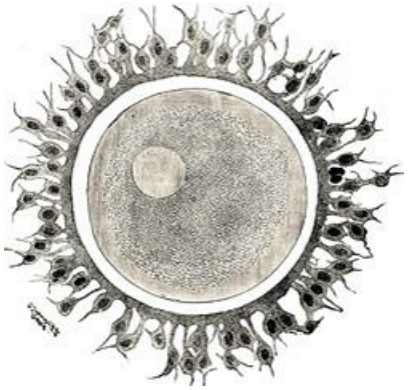


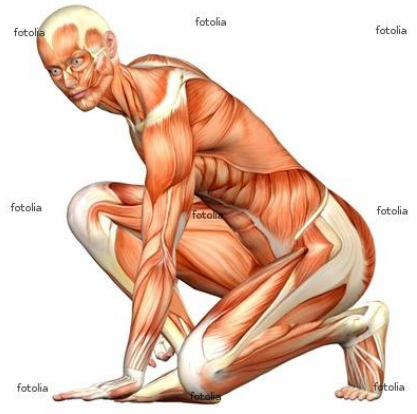
What is a Computer?



What is a Computer?



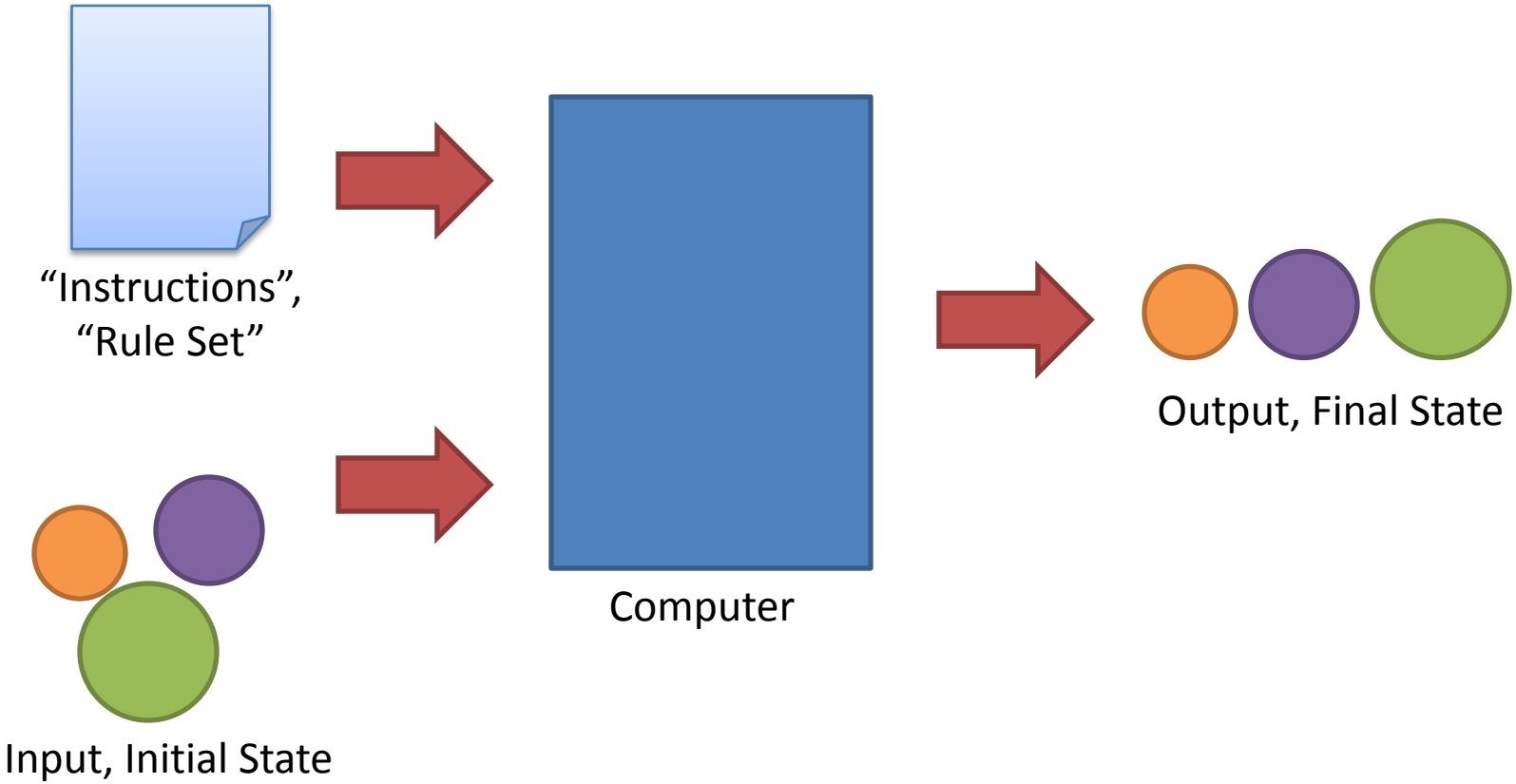


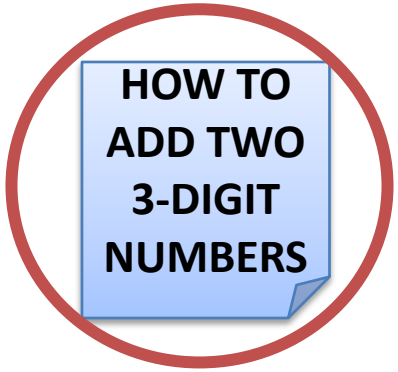


“Nutrients”
“Stimuli”

What is a Computer?

- A device capable of **computation**
  - *Computation* – Finding a solution to a problem from given inputs by means of an **algorithm.**
- Contemporary (Popular) Meaning:
  - Machine (Electronic Device) that can manipulate inputs/data according to a set of instructions

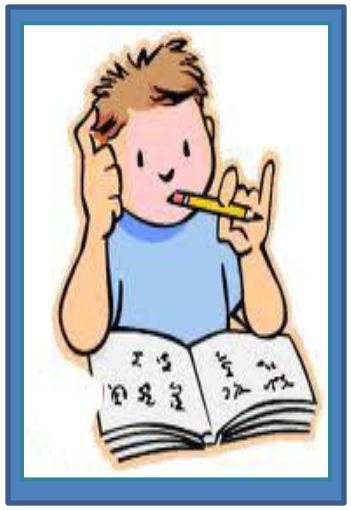

"Instructions",
"Rule Set"
Input, Initial State
Computer
Output, Final State



956

365

Input, Initial State

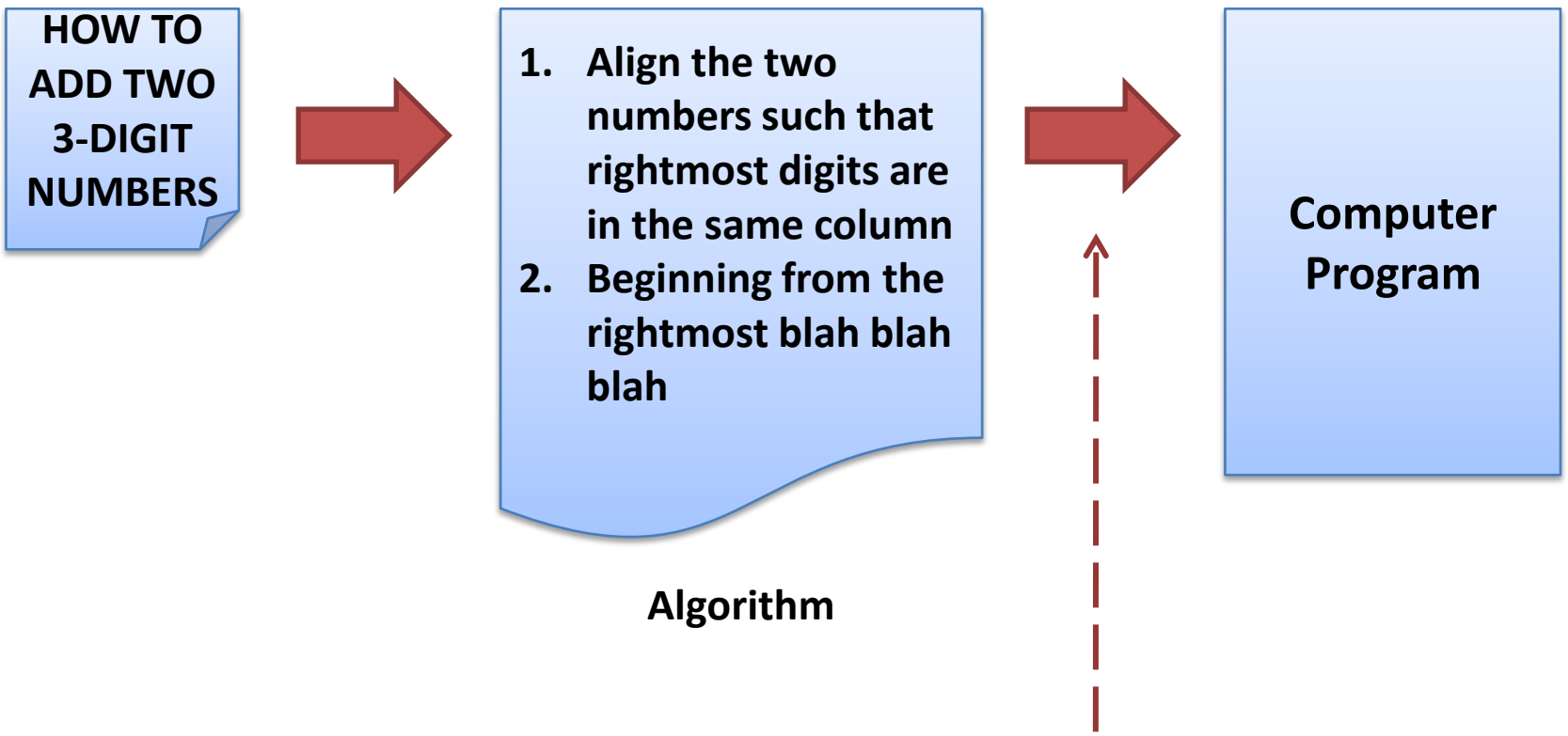


Computer

1321

Output, Final State

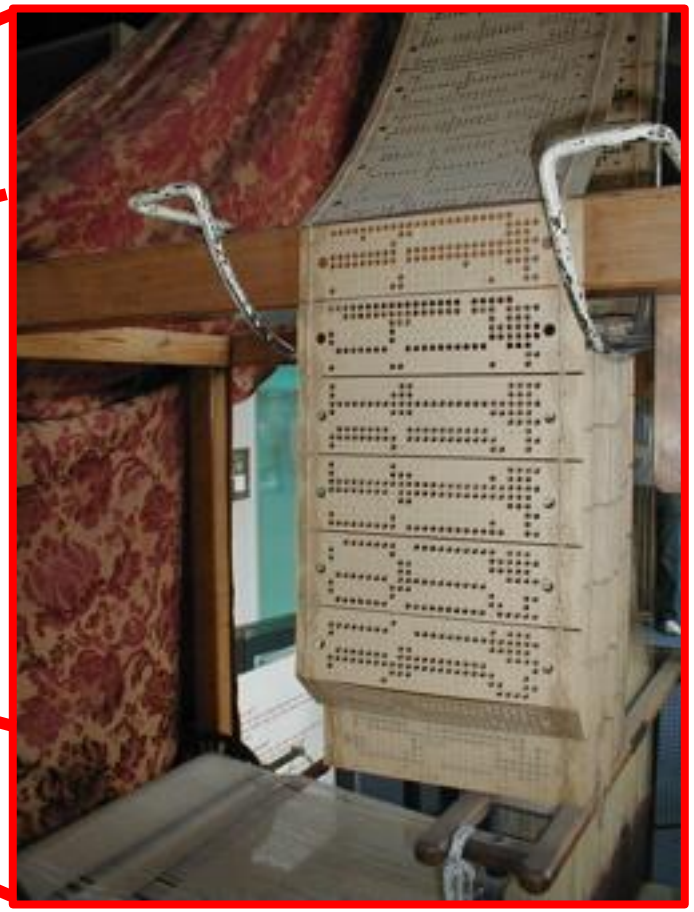
Computers and Computer Programs



What is a Computer Program?

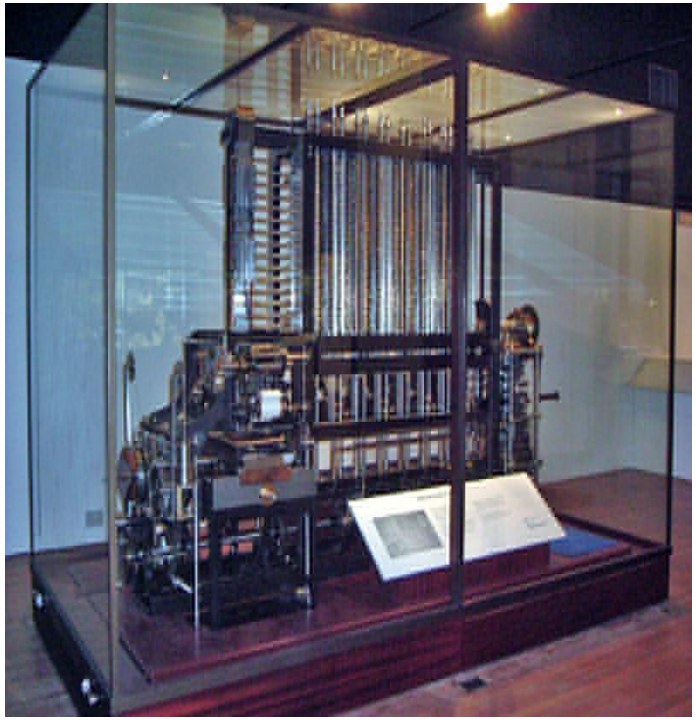
- Program
  - A list of instructions that can be interpreted by a computer
  - Embodies an algorithm
  - Algorithm in computer-encoded version

Computer and Computer Programs



The Jacquard Loom

Computers and Computer Programs



The Difference Engine

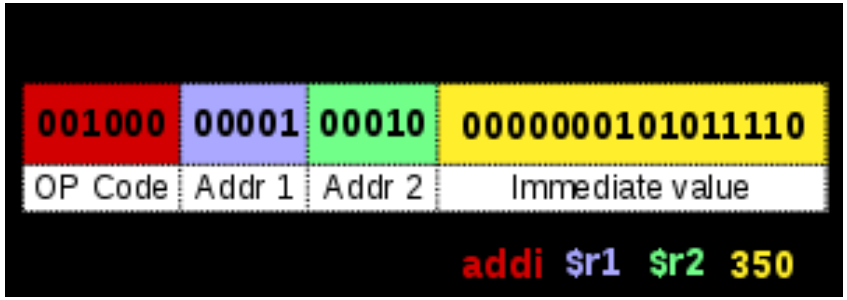
?

Computers and Computer Programming

- Question:
  - How do we write computer programs?
  - How did we write computer programs over the years and how has advancements in computer technology changed the way we program?

Computers and Computer Programming

- Electronic computers have “Instruction Sets”
  - Basic operations that the computer can do
  - i.e., Turing machine: MOVE head, READ, and WRITE
- The most primitive (baseline) language that can we use to write programs is MACHINE LANGUAGE



Computer and Computer Programming

- Machine Language is messy and error-prone for humans
- So we developed higher level languages:
  - Assembly language
  - Higher “natural” programming languages

```
MAIN PROC NEAR
        CALL B10CLR
A20:    CALL C10SET
        CALL D10DISP
        CMP CTR, 0FFH
        JE A30
        INC CTR
        ADD COL, 02
...
```

Higher Level Computer Programming Languages

- Instructions are human readable
- Programming platforms come with compilers and/or translators

```
******ERRORS INDUCED BY ARITHMETIC
OPERATONS ON SAMLL NUMBERS
REAL
SUM6,SUM7,SUM8,DIF6,DIF7,DIF8,SUMINF
     OPEN(6,FILE='PRN')
     SUM6=.9*(1.-0.1**6)/0.9
     SUM7=.9*(1.-0.1**7)/0.9
     SUM8=.9*(1.-0.1**8)/0.9
******COMPUTER SUM OF INFINITE TERMS
            SUMINF=0.9/(1.0-0.1)
******COMPUTE DIFFERENCES BETWEEN
FINITE & INFINITE SUMS
      DIF6 = SUMINF - SUM6
      DIF7 = SUMINF - SUM7
      DIF8 = SUMINF – SUM8
```

Developments in Programming Languages

- Dictated by
  - Technology advancements in computer (removing limitations)
  - Complexity of problems being solved
  - Developments in compiler design
- Improvements in many key areas made it possible for language developers to incorporate **perspectives** into programs and come up with different programming paradigms.

The Programming Paradigms

- **Structured/Procedural Programming**
  - Top down design, algorithmic perspective (sequential, imperative), procedure/function/subroutine
- **Logic Programming**
  - Declarative, uses mathematical logic
- **Functional Programming**
  - Treats computation as the evaluation of mathematical functions
- **Object Oriented Programming**
  - Programs are treated as objects that may be composed of smaller objects. Objects being with state and behavior.

Paradigms and Their Effects

- Basically, the programming paradigm dictates:
  - How you should approach a problem
  - How you should treat entities in the problem
  - How you write your code
  - How you arrange your code

```
#include<…>
int i, j, k;
char* s;

void f1( ) {
   //…
}

int f2( ) {
   //…
   return x;
}

int main( ) {
   f1( ); f2( );
}
```

Procedural Programming emphasizes top-down design, divide and conquer approach using procedures as modular components of your program

Object Oriented Programming

- Questions for you to find out:
  - How and why did OOP originate?
  - Who originated OOP?
  - History of OOP Languages
- *We will compare notes next meeting.*