Principles of Programming Languages

http://fm.zju.edu.cn

2014-09-24

Chocolate Cake Receipt

- Materials:
  - 1/2 cup butter
  - 4 ounce bittersweet chocolate
  - 2 eggs
  - 2 egg yolks
  - 1/4 cup white sugar
  - 2 teaspoons all purpose flour
- Step:
  - Preheat oven to 450°F.
  - Heat butter and chocolate until chocolate is almost melted.
  - Beat eggs, yolks and sugar until light colored and thick.
  - Mix chocolate and butter, and slowly pour into egg mixture, stirring constantly. Stir in flour until just combined.
  - Pour batter into molds and bake for 6 to 7 minutes. Invert molds on plates, let sit 15 seconds, and unmold. Serve with whipped cream.

A Receipt Is Like a Program

Receipt:

- tells you how to make a chocolate cake
- Has inputs (butter, eggs, chocolate, flour, sugar) & output (chocolate cake)
- Define a procedure
- Instruct how processors (oven, mixer) process inputs to generate output
- Can be expressed in different languages

Program:

- You tell a computer how to do a computation
- Has inputs and outputs
- Define a procedure (algorithm)
- Instruct how processors process inputs to generate outputs
- Can be expressed in different languages

Questions

- Given two languages, how do they differ in expressing the same receipt/algorithm?
- Which language is better?

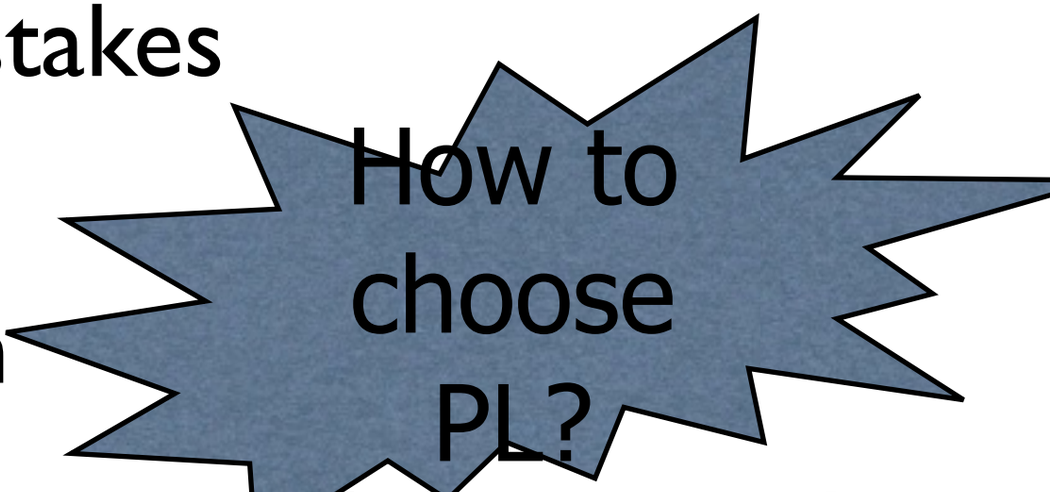
  → How to evaluate "goodness" of languages?
- Why are there so many different languages?
- What is "programming language" anyway?
- Why does a programming language have so many different features?
- How are these features implemented?
- …

A Programming Language Is

- An artificial language designed to express computations or algorithms that can be performed by a computer
  -- Wikipedia
  - A language is a means of expressing your thoughts to others
  - In the case of PL, it is a means of expressing your thoughts (algorithms) to a computer
  - Natural languages such as Chinese and English are not used because they cannot be easily translated into machine language executable by the computer

Why PPL Important?

- A language is a framework for problem-solving
  - It may facilitate or hinder your thoughts and, thus, the abilities to solve problems
  - It may help you make fewer mistakes
  - Example: tense and gender, e.g. "He was doing great!" in English
  - Example: a C language that supports only static and global variables → no malloc()
    - How to implement hash table? linked list?



(Ref.: John Mitchell, http://www.stanford.edu/class/cs242)

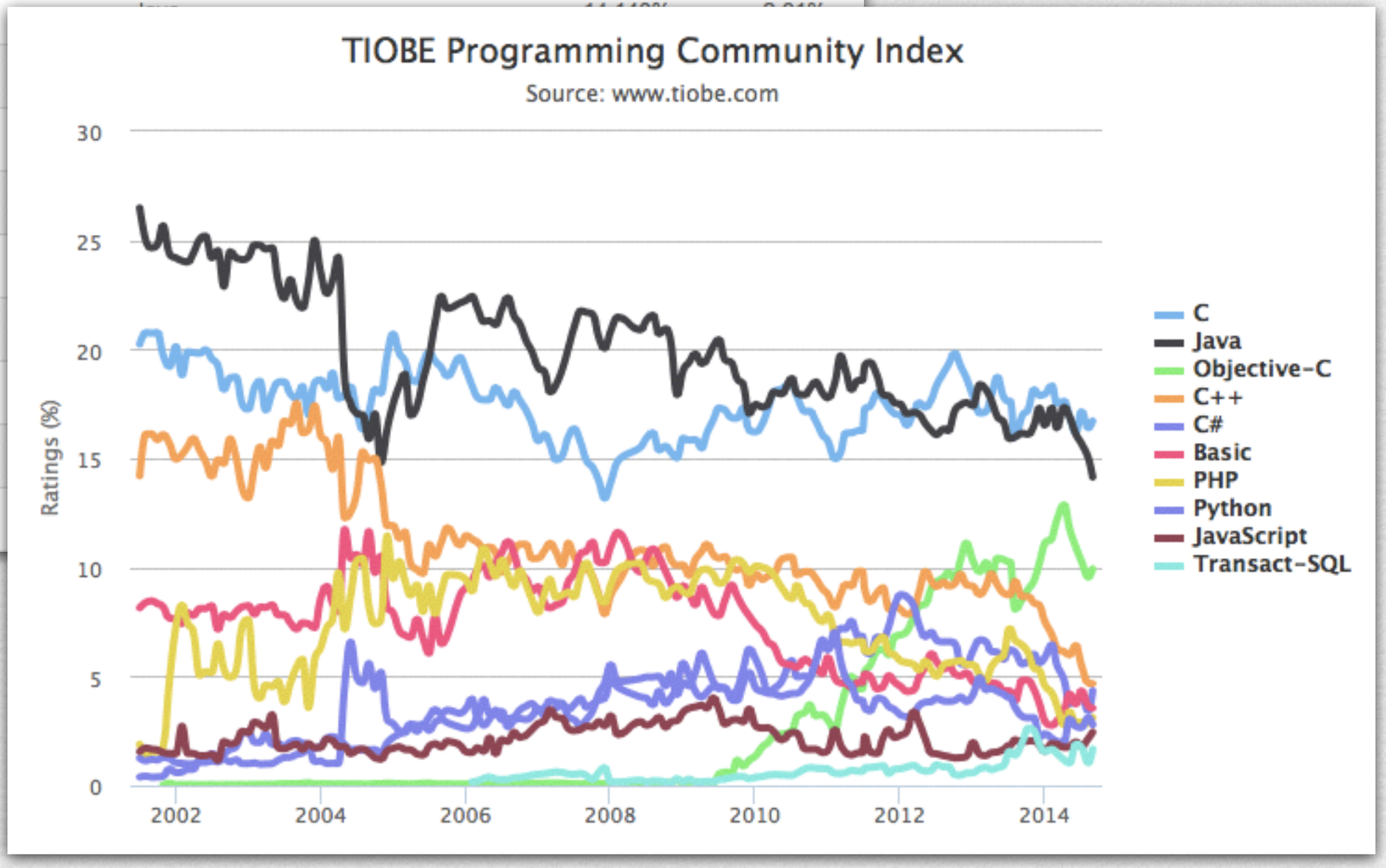
Top Languages

Sep 2014	Sep 2013	Change	Programming Language	Ratings	Change
1	1		C	16.721%	-0.25%
2	2		Java	14.140%	-2.01%
3	4	▲	Objective-C	9.935%	+1.37%
4	3	▼	C++	4.674%	-3.99%
5	6	▲	C#	4.352%	-1.21%
6	7	▲	Basic	3.547%	-1.29%
7	5	▼	PHP	3.121%	-3.31%
8	8		Python	2.782%	-0.39%
9	9		JavaScript	2.448%	+0.43%
10	10		Transact-SQL	1.675%	-0.32%

图片来源：http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

Languages

Sep 2014	Sep 2013	Change	Programming Language	Ratings	Change
1	1		C	16.721%	-0.25%
2	2				
3	4	▲			
4	3	▼			
5	6	▲			
6	7	▲			
7	5	▼			
8	8				
9	9				
10	10				



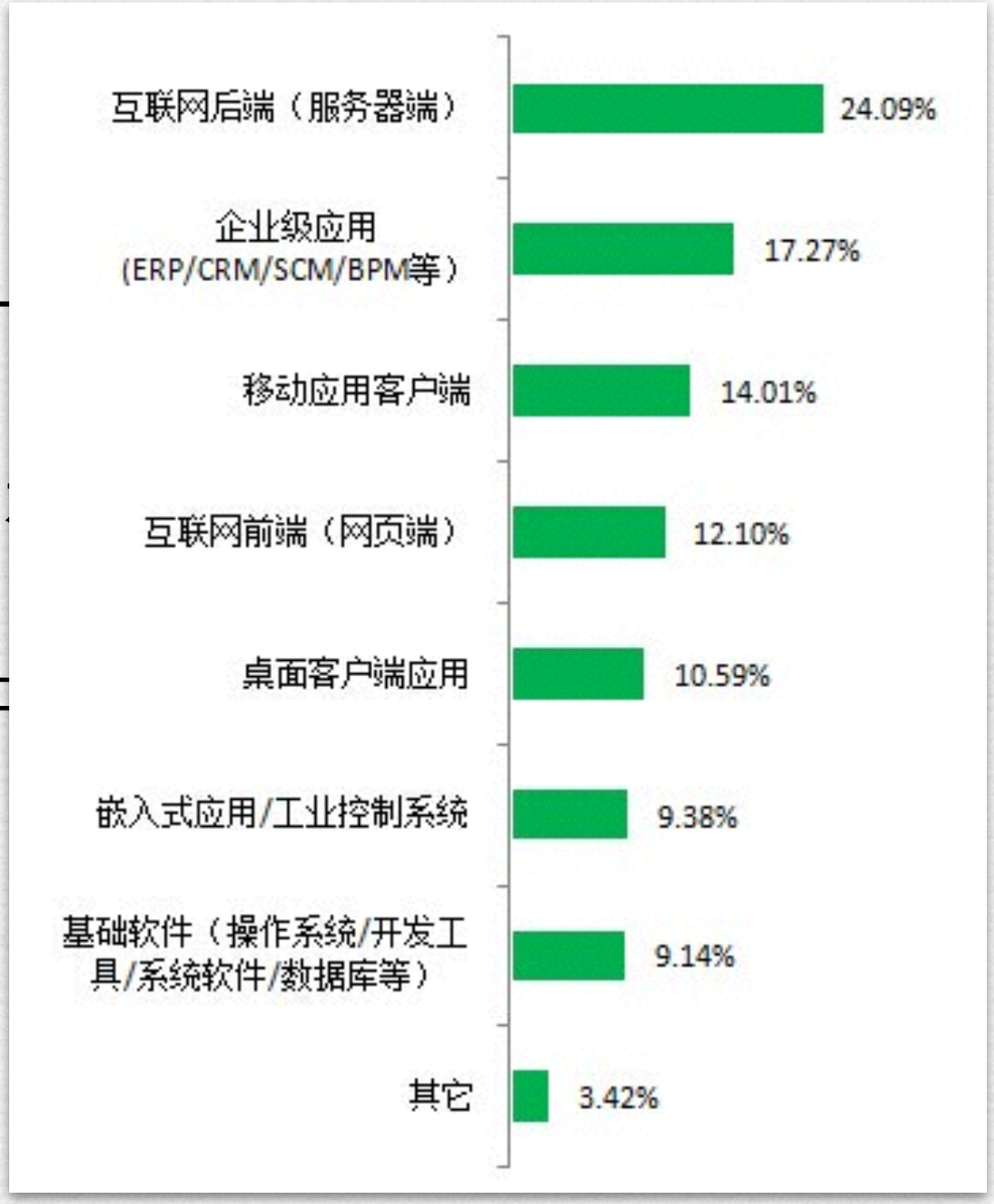
图片来源：http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

2014中国软件开发者调查

http://www.csdn.net/article/2014-06-06/2820116

软件在哪里?

- 桌面上：C#、C-
- 服务器中：Java
  - 也包括浏览器中
- 设备里：C



http://www.csdn.net/article/2014-06-06/2820116

Important to Know PPL by Trend

- Increasing use of type-safe languages: Java, C#, …
- Scripting languages for web applications with increasing client-side functionality
- More on expressing algorithms than syntax
- Runtime environment and virtualization with continuous compilation, analysis, and checking
- More program analysis abilities: automated error detection and recovery

(Ref.: John Mitchell, http://www.stanford.edu/class/cs242)

Important to Know PPL by Tradeoffs

Factors influencing programming language

- Expressiveness:
  - Application domains
  - Programming methods: multiprogramming, interactive systems,…
- Implementation: efficiency
  - Computer architecture, OS, toolchain, library
  - Every convenience has its cost; must recognize cost of presenting an abstract view of machine
    - Understand trade-offs in programming language design

(Ref.: M. Sirjani, http://ut.ac.ir/classpages/ProgrammingLanguages)

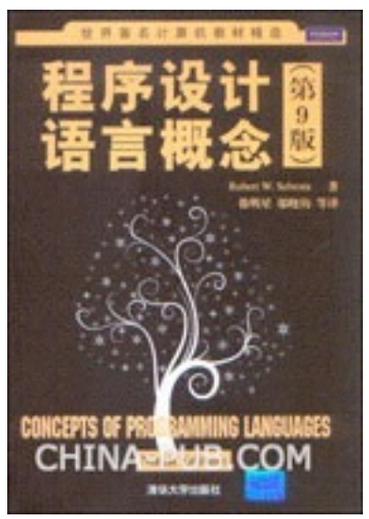
PPL as a Course

- What is not
  - Do not teach you a programming language
  - Do not teach you how to program
- What is
  - Introduce fundamental concepts of programming languages
  - Discuss design issues of various language constructs
  - Examine design/implementation choices for these constructs
  - Compare design alternatives
- Need to be familiar in at least one PL

Why Study PPL?

- To improve your ability to develop effective algorithms and to use your language
  - O-O features, recursion
  - Call by value, call by reference
- To allow a better choice of PL
- Increased ability to learn new languages
- To make it easier to design a new language
- To understand significance of implementation
  - E.g. the efficiency of a recursive function

Text Books



- 《程序设计语言概念》, "Concepts of Programming Languages", by Robert W. Sebesta, translated by 徐明星 and 邬晓钧, 清华大学出版社, 2011, ISBN 978-7-302-22956-8（原书第九版）

Assessment

- Class Performance: 30%
- Assignments: 40%, each for four weeks
- Final Exam: 30%

MOOC

- 华盛顿大学：编程语言
- https://www.coursera.org/course/proglang
- 需要注册学习，次周课内讨论

Sites

- http://fm.zju.edu.cn

Connection

- wengkai@zju.edu.cn
- 新浪微博： http://weibo.com/ba5ag
- 人人： http://www.renren.com/359926703
- 网易云课堂： http://study.163.com/u/wengkai
- cc98： http://cc98.org/fatmouse

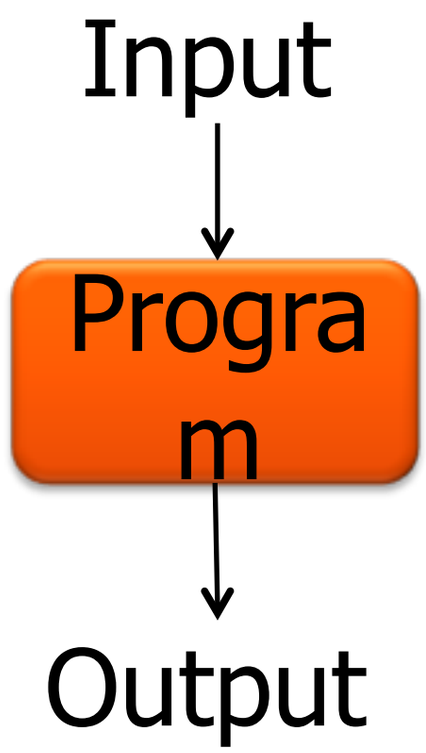
SIGPLAN

- SIGPLAN is a Special Interest Group of ACM that focuses on Programming Languages. In particular, SIGPLAN explores the design, implementation, theory, and efficient use of programming languages and associated tools. Its members are programming language users, developers, implementers, theoreticians, researchers and educators.

Programming Language

- A programming language is an artificial language designed to express computations or algorithms that can be performed by a computer
  -- Wikipedia

- A program is computer coding of an algorithm that
  - Takes input
  - Performs some calculations on the input
  - Generates output



Models of Programming Languages

Programming is like …

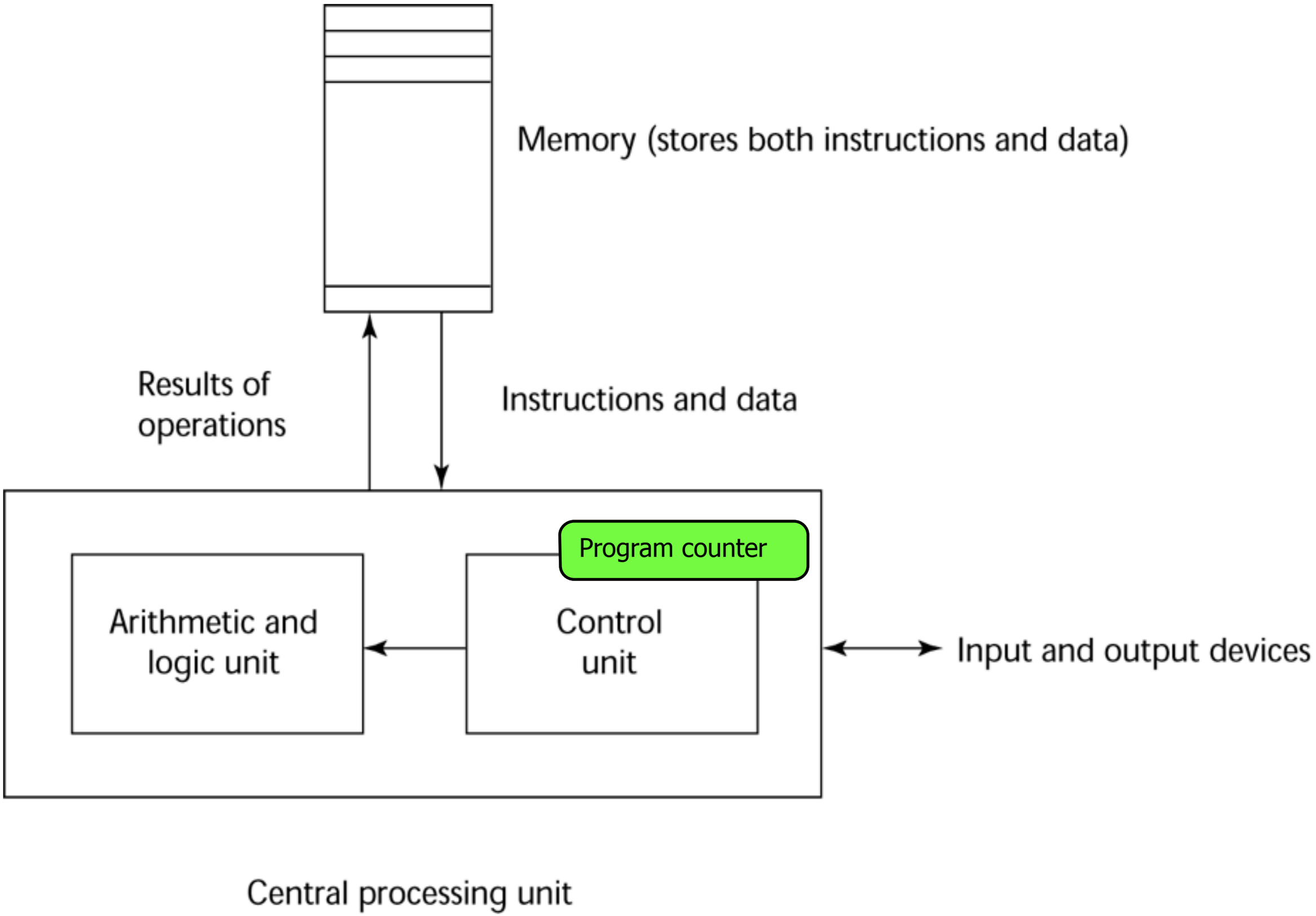
1st View: Imperative & Procedural

- Computers take commands and do operations
- Thus, programming is like …
  issuing procedural commands to the computer
  - Example: a factorial function in C

```
int fact(int n) {
   int sofar = 1;
   while (n>0) sofar *= n--;
   return sofar;
}
```

- Since almost all computers today use the von Neumann architecture → PL mimic the arch.

von Neumann Architecture



von Neumann Architecture

- Key features:
  - Data and programs stored in memory
  - Instructions and data are piped from memory to CPU
  - Fetch-execute-cycle for each machine instruction

```
initialize the program counter (PC)

repeat forever

   fetch the instruction pointed by PC

   increment the counter

   decode the instruction

   execute the instruction

end repeat
```

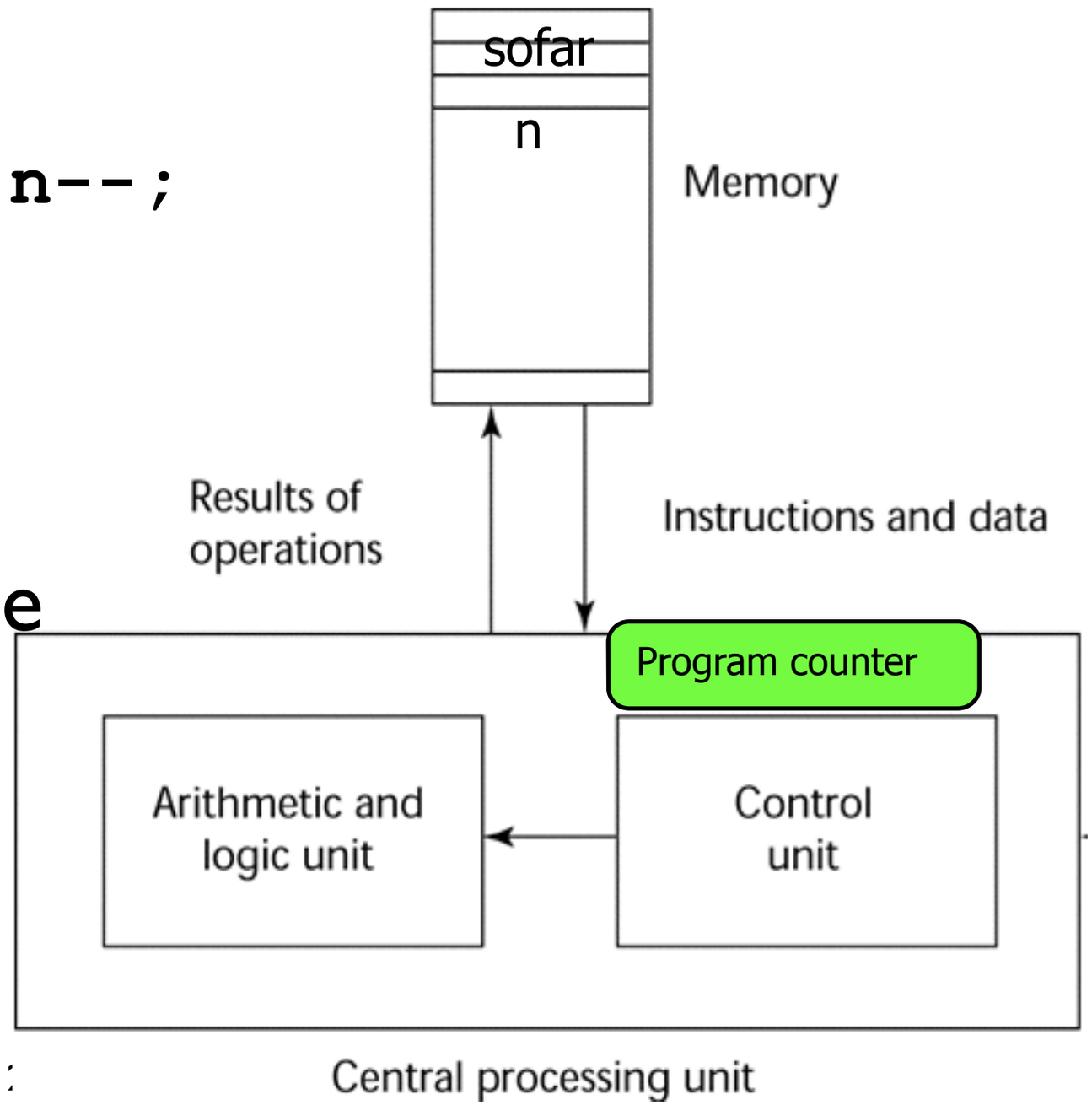
Imperative Language and Arch.

- Example: a factorial function in C

```
int fact(int n) {
   int sofar = 1;
   while (n>0) sofar *= n--;
   return sofar;
}
```

- Indicates that data **`n`**, **`sofar`**, and program code are stored in memory
- Program code instructs CPU to do operations

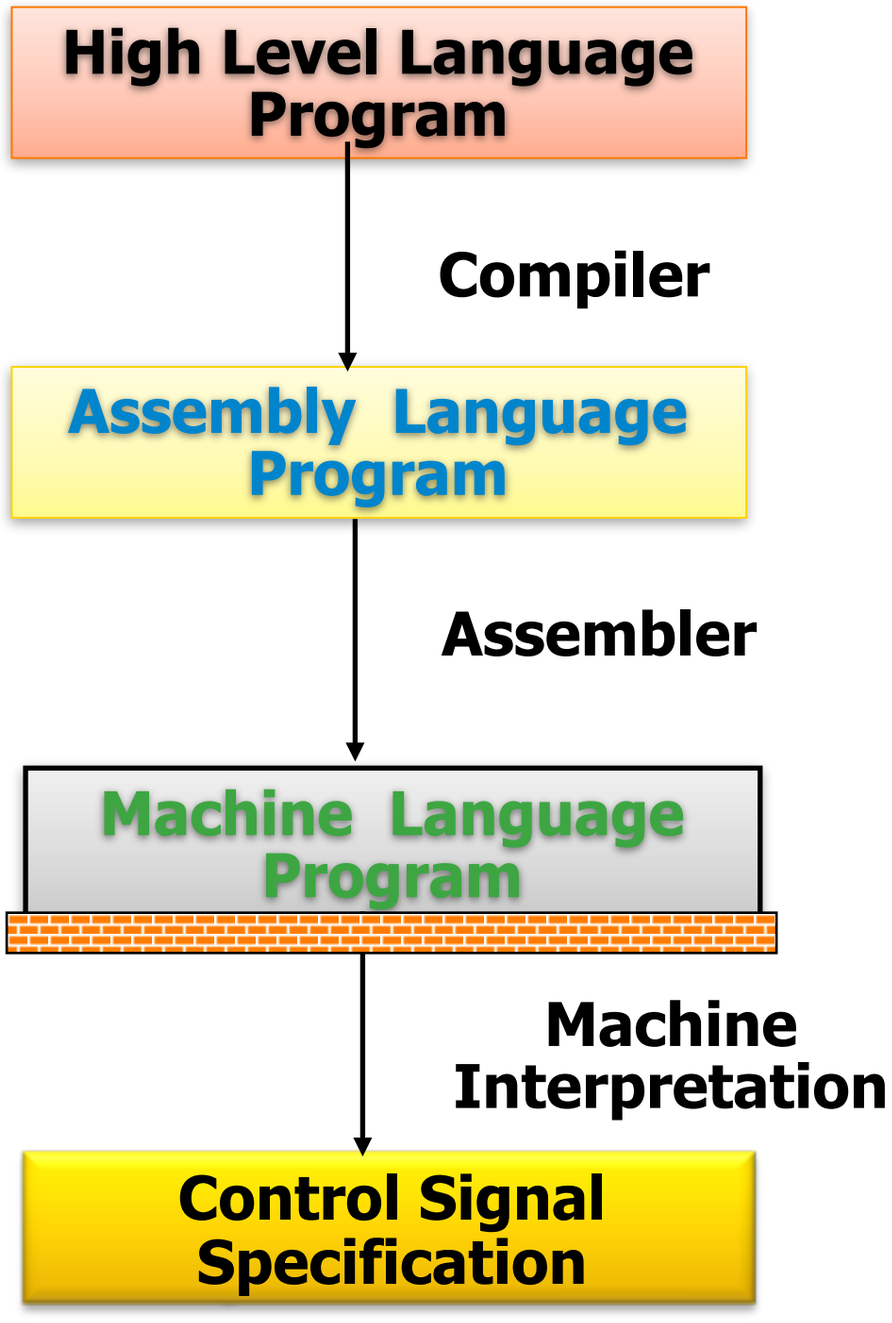
```
int fact(int n) {
  int sofar = 1;
  while (n>0) sofar *= n--;
  return sofar; }
```



Imperative Languages and Arch.

- Imperative languages, e.g., C, C++, Java, which dominate programming, mimic von Neumann architecture
  - Variables ←→ memory cells
  - Assignment statements ←→ data piping between memory and CPU
  - Operations and expressions ←→ CPU executions
  - Explicit control of execution flows ←→ prog. counter
- Allow efficient mapping between language and hardware for good execution performance, but limited by von Neumann bottleneck

Layers of Abstraction/ Translation




```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

```
lw      $15, 0($2)
lw      $16, 4($2)
sw$16, 0($2)
sw$15, 4($2)
```

```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```

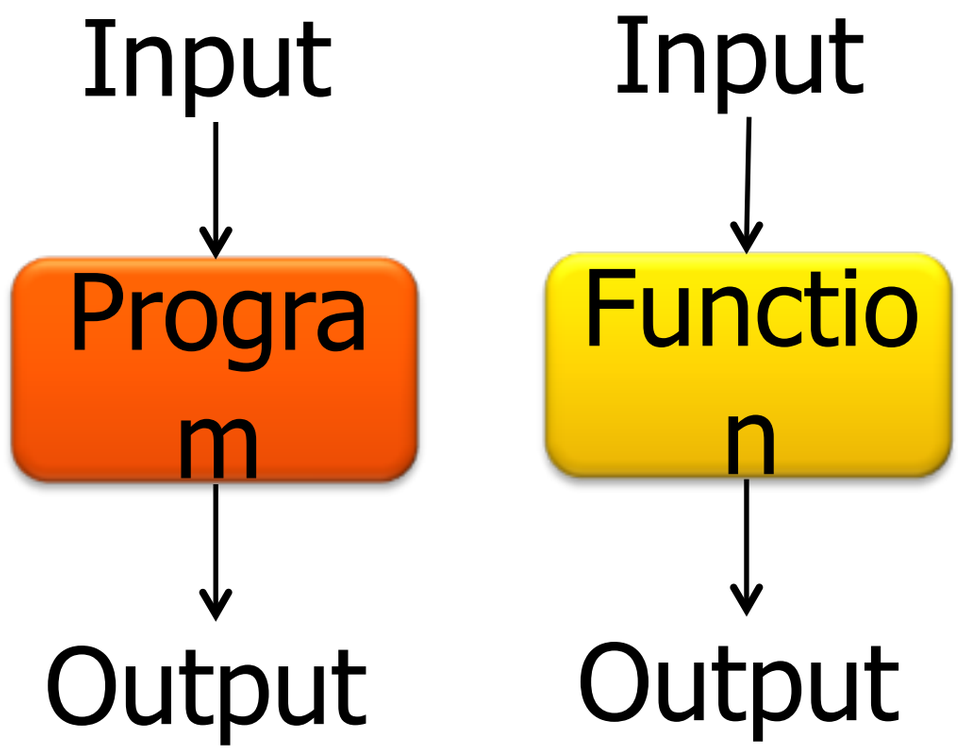
All imperative!

```
ALUOP[0:3] <= InstReg[9:11] & MASK
```

2nd View: Functional

- Programming is like …
  solving mathematical functions, e.g.,
  z = f(y, g(h(x)))
  - A program, and its subprograms, are just implementations of mathematical functions
  - Example: a factorial function in ML

```
fun fact x =
   if x <= 0
   then 1
   else x * fact(x-1);
```



Another Functional Language: LISP

- Example: a factorial function in Lisp

```
(defun fact (x)
   (if (<= x 0) 1 (* x (fact (- x 1)))))
```

  - Computations by applying functions to parameters

- No concept of variables (storage) or assignment
  - Single-valued variables: no assignment, not storage
- Control via recursion and conditional expressions
  - Branches → conditional expressions
  - Iterations → recursion

3rd View: Logic

- Programming is like …
  logic induction
  - Program expressed as rules in formal logic
  - Execution by rule resolution
  - Example: relationship among people

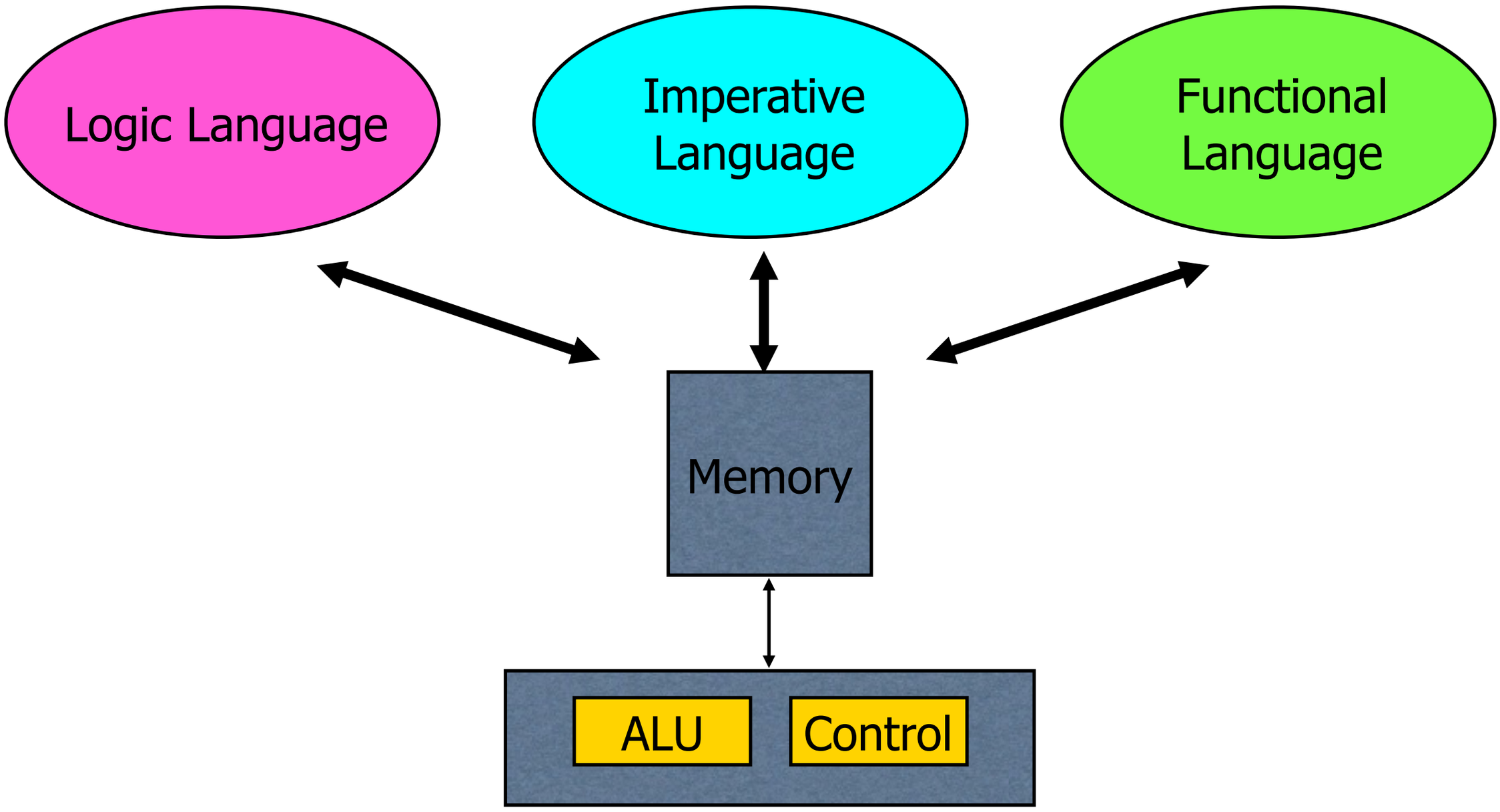
```
fact:   mother(joanne,jake).
        father(vern,joanne).

rule:   grandparent(X,Z)  :-
parent(X,Y),
                    parent(Y,Z).
goal:   grandparent(vern,jake).
```

Logic Programming

- Non-procedural
  - Only supply relevant facts (predicate calculus) and inference rules (resolutions)
  - System then infer the truth of given queries/goals
- Highly inefficient, small application areas (database, AI)

```
fact(X,1) :-  X =:= 1.
fact(X,Fact) :-
   X > 1, NewX is X - 1,
   fact(NewX,NF),
   Fact is X * NF.
```

Summary: Language Categories



von Neumann Architecture

Summary: Language Categories

- Imperative
  - Variables, assignment statements, and iteration
  - Include languages that support object-oriented programming, scripting languages, visual languages
  - Ex.: C, Java, Perl, JavaScript, Visual BASIC .NET
- Functional
  - Computing by applying functions to given parameters
  - Ex.: LISP, Scheme, ML
- Logic
  - Rule-based (rules are specified in no particular order)
  - Ex.: Prolog