IA010: Principles of Programming Languages

Introduction

Achim Blumensath
blumens@fi.muni.cz

Faculty of Informatics, Masaryk University, Brno

Warm-up: A Quiz

What does this program do?

```
++++++++++[>+++++++>++++++++++>+++>+<<<<-]>++.>+.+++++++
..+++.>++.<<+++++++++++++++.>.+++.------.--------.>+.>.
```

Warm-up: A Quiz

What does this program do?

```
++++++++++[>+++++++>++++++++++>+++>+<<<<-]>++.>+.+++++++
..+++.>++.<<+++++++++++++++.>.+++.------.--------.>+.>.
```

Prints “Hello World!”

Warm-up: A Quiz

What does this program do?

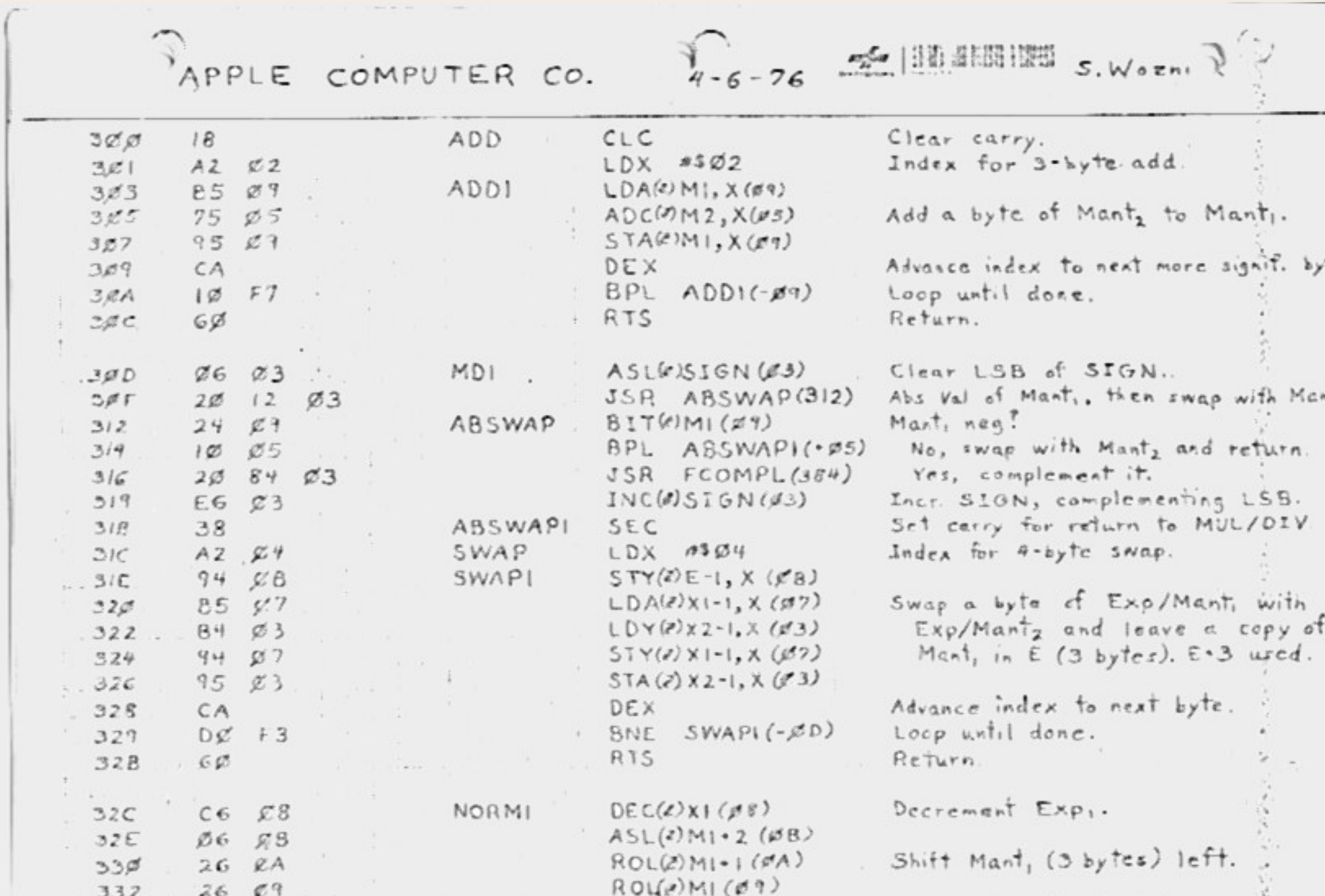
```
++++++++++[>+++++++>++++++++++>+++>+<<<<-]>++.>+.+++++++
..+++.>++.<<+++++++++++++++.>.+++.------.--------.>+.>.
```

Prints “Hello World!”

Brainfuck (1993)

- Turing-complete programming language
- tape containing numbers (inc/dec), a data pointer (l/r), input/output, conditional jump
- compiler of size 100 bytes known to exist

Before high-level programming languages …



APPLE COMPUTER CO. 4-6-76 S. Wozni

```
300  18          ADD     CLC             Clear carry.
301  A2 02               LDX #$02        Index for 3-byte add.
303  B5 09       ADD1    LDA(2)M1,X(09)
305  75 05               ADC(2)M2,X(05)  Add a byte of Mant2 to Mant1.
307  95 09               STA(2)M1,X(09)
309  CA                  DEX             Advance index to next more signif. by
30A  10 F7               BPL ADD1(-09)   Loop until done.
30C  60                  RTS             Return.

30D  06 03       MD1     ASL(6)SIGN(03)  Clear LSB of SIGN..
30F  20 12 03            JSR ABSWAP(312) Abs Val of Mant1, then swap with Ma
312  24 09       ABSWAP  BIT(6)M1(09)    Mant1 neg?
314  10 05               BPL ABSWAP1(+05)  No, swap with Mant2 and return.
316  20 84 03            JSR FCOMPL(384)   Yes, complement it.
319  E6 03               INC(6)SIGN(03)  Incr. SIGN, complementing LSB.
31B  38          ABSWAP1 SEC             Set carry for return to MUL/DIV
31C  A2 04       SWAP    LDX #$04        Index for 4-byte swap.
31E  94 08       SWAP1   STY(2)E-1,X(08)
320  B5 07               LDA(2)X1-1,X(07)  Swap a byte of Exp/Mant1 with
322  B4 03               LDY(2)X2-1,X(03)    Exp/Mant2 and leave a copy of
324  94 07               STY(2)X1-1,X(07)    Mant1 in E (3 bytes). E=3 used.
326  95 03               STA(2)X2-1,X(03)
328  CA                  DEX             Advance index to next byte.
329  D0 F3               BNE SWAP1(-0D)  Loop until done.
32B  60                  RTS             Return.

32C  C6 08       NORM1   DEC(2)X1(08)    Decrement Exp1.
32E  06 08               ASL(2)M1+2(0B)
330  26 0A               ROL(2)M1+1(0A)  Shift Mant1 (3 bytes) left.
332  26 09               ROL(2)M1(09)
```

Now …

C	Python	Haskell	Scala
C++	PHP	OCaml	Rust
Java	JavaScript	F#	Go
C#	VisualBasic	Scheme	Swift
Ada	Perl	…	

Now …

C	Python	Haskell	Scala
C++	PHP	OCaml	Rust
Java	JavaScript	F#	Go
C#	VisualBasic	Scheme	Swift
Ada	Perl	…	

A zoo of programming languages

Now …

C	Python	Haskell	Scala
C++	PHP	OCaml	Rust
Java	JavaScript	F#	Go
C#	VisualBasic	Scheme	Swift
Ada	Perl	…	

A zoo of programming languages

Can we somehow categorise them?

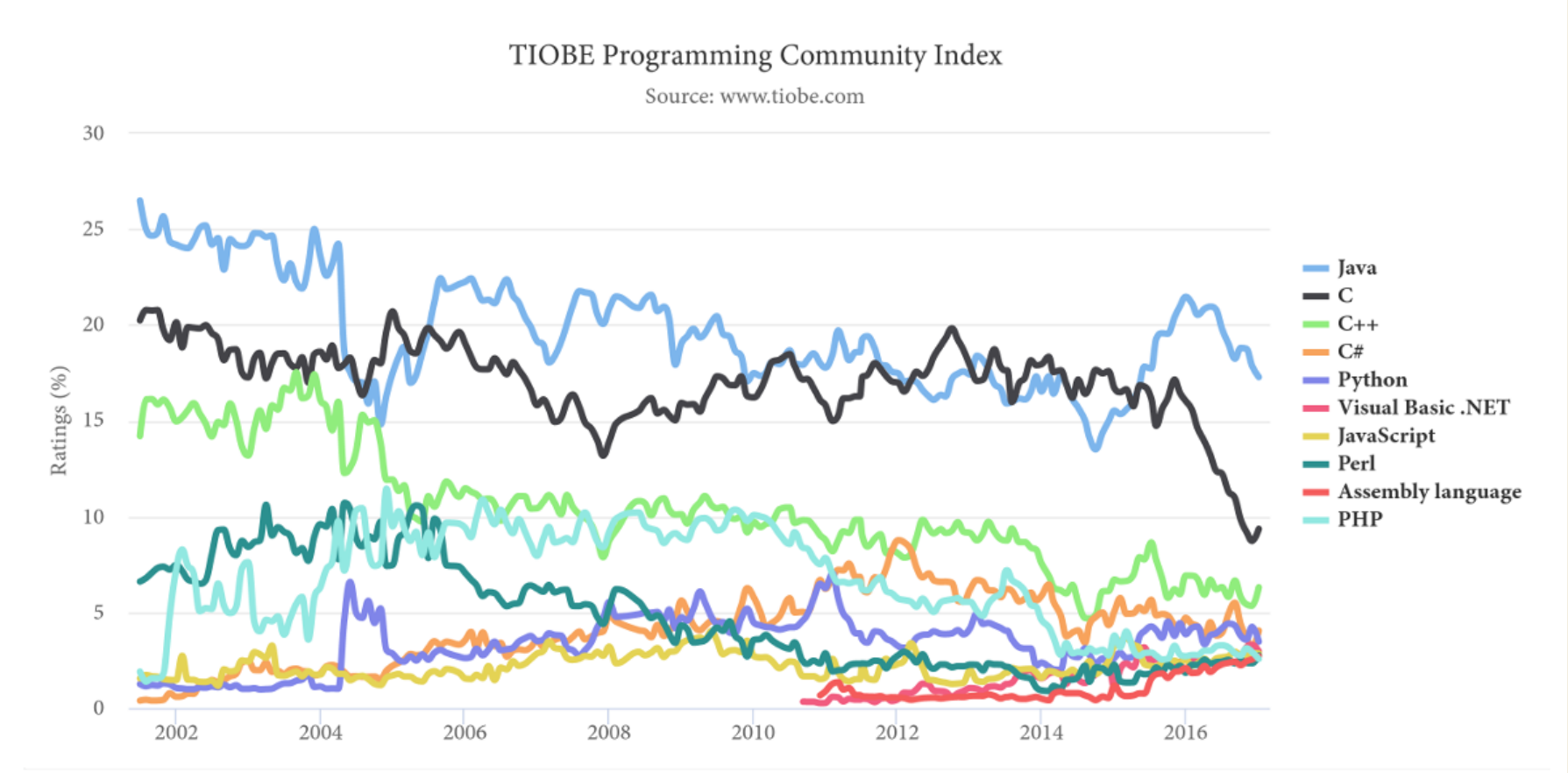
How do we choose one?

Language popularity

TIOBE index, January 2017, www.tiobe.com

Jan 2017	Jan 2016	Change	Programming Language	Ratings	Change
1	1		Java	17.278%	-4.19%
2	2		C	9.349%	-6.69%
3	3		C++	6.301%	-0.61%
4	4		C#	4.039%	-0.67%
5	5		Python	3.465%	-0.39%
6	7	^	Visual Basic .NET	2.960%	+0.38%
7	8	^	JavaScript	2.850%	+0.29%
8	11	^	Perl	2.750%	+0.91%
9	9		Assembly language	2.701%	+0.61%
10	6	v	PHP	2.564%	-0.14%
11	12	^	Delphi/Object Pascal	2.561%	+0.78%
12	10	v	Ruby	2.546%	+0.50%
13	54	^	Go	2.325%	+2.16%
14	14		Swift	1.932%	+0.57%
15	13	v	Visual Basic	1.912%	+0.23%

Language popularity



Desirable language features

Desirable language features

- simplicity
- orthogonality
- clear (and defined) semantics
- ease of use
- easy to learn
- clean and readable syntax
- expressive power
- support for many paradigms and coding styles
- strong safety guarantees
- produces fast code
- compilation speed
- reduced memory usage
- good library and tool chain support
- standardisation and documentation
- interoperability with other languages
- hardware and system independence
- support for hardware and system programming
- usability by non-programmers
- …

Kinds of software

Kinds of software

- business applications
- office software, graphics software
- server software
- video games
- number crunching
- phone apps
- control software for embedded devices
- scripts, utilities

Programming paradigms

Programming paradigms

- **procedural:** program is structured as a collection of procedures/functions
- **imperative:** list of commands
- **functional:** expressions that compute a value
- **declarative:** describe what you want to compute, not how
- **object-oriented:** objects communicating via messages
- **data-oriented:** layout of your data in memory
- **reactive:** network of components that react to events

Which language/paradigm/coding style is the best?

Which language/paradigm/coding style is the best?

Choose the right tools for the job!

Which language/paradigm/coding style is the best?

Choose the right tools for the job!

$\Rightarrow$ the more tools available, the better

Which language/paradigm/coding style is the best?

Choose the right tools for the job!

$\Rightarrow$ the more tools available, the better

$\Rightarrow$ need to be familiar with many styles and paradigms

Which language/paradigm/coding style is the best?

Choose the right tools for the job!

$\Rightarrow$ the more tools available, the better

$\Rightarrow$ need to be familiar with many styles and paradigms

Multi-paradigm languages

The more paradigms your language support, the more tools you have in your toolbox.

State of the art

- functional programming, dependent types: Idris
- linear types, borrow checker: Rust
- imperative programming, error handling: Zig
- imperative programming, design by contract: Dafny, Whiley
- module system: SML, Ocaml
- declarative programming: Mercury
- object-oriented programming: Scala
- concurrency: Go, Pony

(list somewhat biased and certainly incomplete)

Why study programming languages and paradigms?

The study of **language features** and **programming styles** helps you to

- choose a language **most appropriate** for a given task
- think about problems in **new ways**
- learn new ways to **express** your ideas and **structure** your code ($\Rightarrow$ more tools in your toolbox)
- read **other peoples code**
- **learn** new languages faster (you only need to learn a new syntax)
- understand the design/implementation decisions and limitations of a given language, so you can **use it better:**
  - You can choose between **alternative ways** of expressing things.
  - You understand more **obscure features**.
  - You can **simulate features** not available in this particular language.

Aspects of programming languages

Syntax: the **structure** of programs.

Describes how the various constructs (statements, expressions, …) can be combined into well-formed programs.

Semantics: the **meaning** of programs.

Tells us what behaviour we can expect from a program.

Pragmatics: the **use** of programming languages.

In which way is the language intended to be used in practice?
What are the various language constructions good for?

Aspects of programming languages

Syntax: the **structure** of programs.

Describes how the various constructs (statements, expressions, …) can be combined into well-formed programs.

PA008 Compiler Construction, PA037 Compiler Project,

IB005/IA006 Formal Languages

Semantics: the **meaning** of programs.

Tells us what behaviour we can expect from a program.

IA011 Programming Language Semantics, IA014 Advanced Functional Programming

Pragmatics: the **use** of programming languages.

In which way is the language intended to be used in practice?
What are the various language constructions good for?

this course

Course organisation

Lectures

- **Thursday, 16:00, A217**
- language: English
- slides, lecture notes, and source code can be found in IS
- video recordings will also be made available there

Examination

- final written exam, in English
- **k** and **z** completion possible

Prerequisites

- no **formal** requirements
- knowledge of at least one programming language
- some basic knowledge of Haskell helpful
- the more languages you know the better

Study materials

Books (only somewhat relevant)

- P. V. Roy, S. Haridi, **Concepts, Techniques, and Models of Computer Programming,** 1st ed., MIT Press, 2004.
- R. W. Sebesta, **Concepts of Programming Languages,** 10th ed., Addison-Wesley, 2012.
- **Programming language pragmatics,** (Ed. M. L. Scott) 3rd ed. Oxford, Elsevier Science, 2009.

Additional resources

- Crafting Interpreters, `www.craftinginterpreters.com`

Topics covered

- a brief history of programming languages
- expressions and functions
- types, type checking, type inference
- state and side-effects
- modules
- control-flow
- declarative programming
- object-oriented programming
- concurrency