IA010: Principles of Programming Languages

Introduction and course organization

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?

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

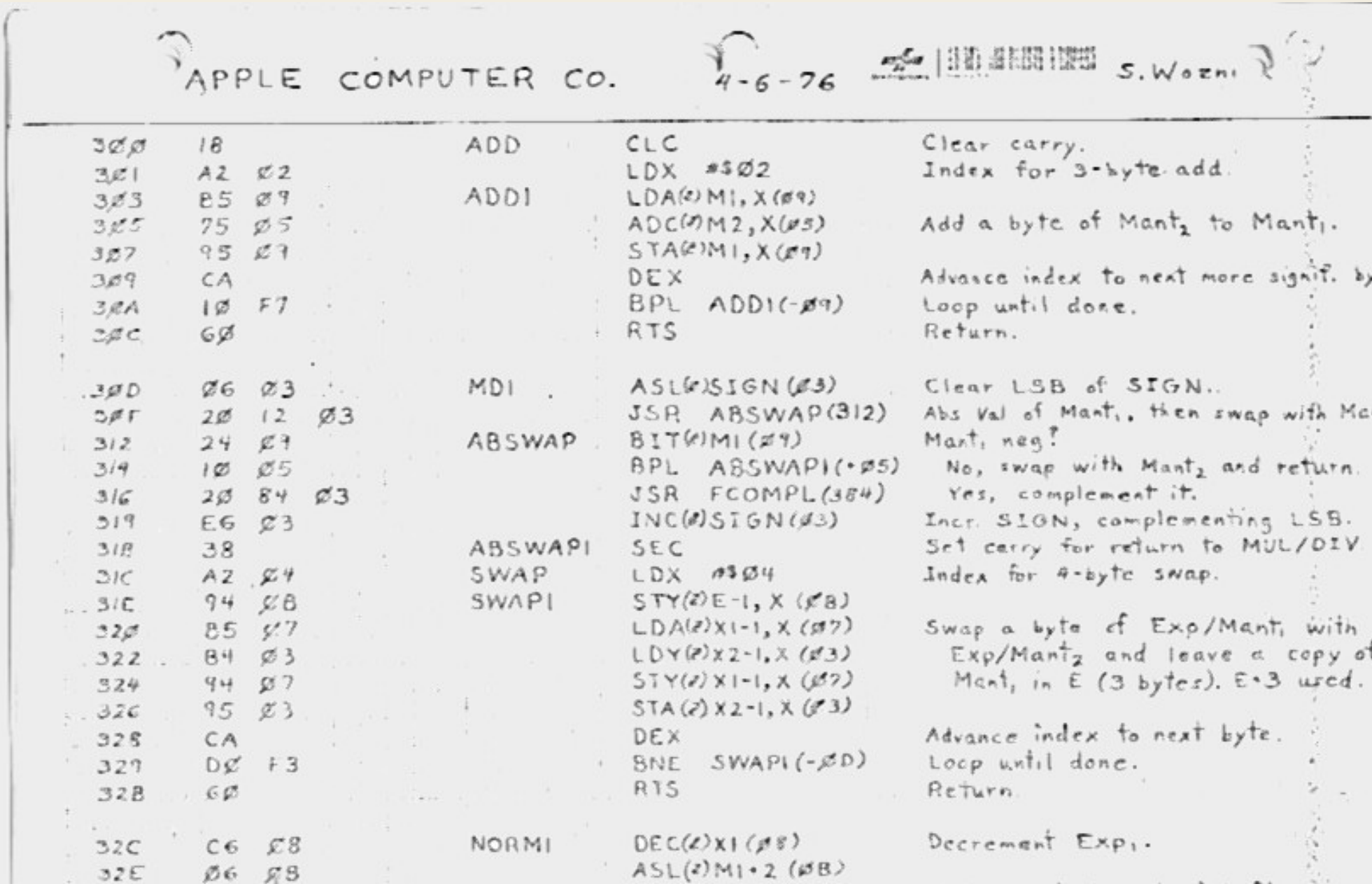
Prints “Hello World!”

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

(more info: `http://en.wikipedia.org/wiki/Brainfuck`)

Before high-level programming languages …



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

```
3ØØ  18          ADD     CLC              Clear carry.
3Ø1  A2 Ø2               LDX #$Ø2         Index for 3-byte add.
3Ø3  B5 Ø9       ADD1    LDA(Ø)M1,X(Ø9)
3Ø5  75 Ø5               ADC(Ø)M2,X(Ø5)   Add a byte of Mant2 to Mant1.
3Ø7  95 Ø9               STA(Ø)M1,X(Ø9)
3Ø9  CA                  DEX              Advance index to next more signif. by
3ØA  1Ø F7               BPL ADD1(-Ø9)    Loop until done.
3ØC  6Ø                  RTS              Return.

3ØD  Ø6 Ø3       MD1     ASL(Ø)SIGN(Ø3)   Clear LSB of SIGN.
3ØF  2Ø 12 Ø3            JSR ABSWAP(312)  Abs Val of Mant1, then swap with Ma
312  24 Ø9       ABSWAP  BIT(Ø)M1(Ø9)     Mant1 neg?
314  1Ø Ø5               BPL ABSWAP1(+Ø5)   No, swap with Mant2 and return.
316  2Ø 84 Ø3            JSR FCOMPL(384)    Yes, complement it.
319  E6 Ø3               INC(Ø)SIGN(Ø3)   Incr. SIGN, complementing LSB.
31B  38          ABSWAP1 SEC              Set carry for return to MUL/DIV
31C  A2 Ø4       SWAP    LDX #$Ø4         Index for 4-byte swap.
31E  94 Ø8       SWAP1   STY(Ø)E-1,X(Ø8)
32Ø  B5 Ø7               LDA(Ø)X1-1,X(Ø7) Swap a byte of Exp/Mant1 with
322  B4 Ø3               LDY(Ø)X2-1,X(Ø3)   Exp/Mant2 and leave a copy of
324  94 Ø7               STY(Ø)X1-1,X(Ø7)   Mant1 in E (3 bytes). E+3 used.
326  95 Ø3               STA(Ø)X2-1,X(Ø3)
328  CA                  DEX              Advance index to next byte.
329  DØ F3               BNE SWAP1(-ØD)   Loop until done.
32B  6Ø                  RTS              Return.

32C  C6 Ø8       NORM1   DEC(Ø)X1(Ø8)     Decrement Exp1.
32E  Ø6 Ø8               ASL(Ø)M1+2(Ø8)
```

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?

Profanity is the one language all programmers know best.

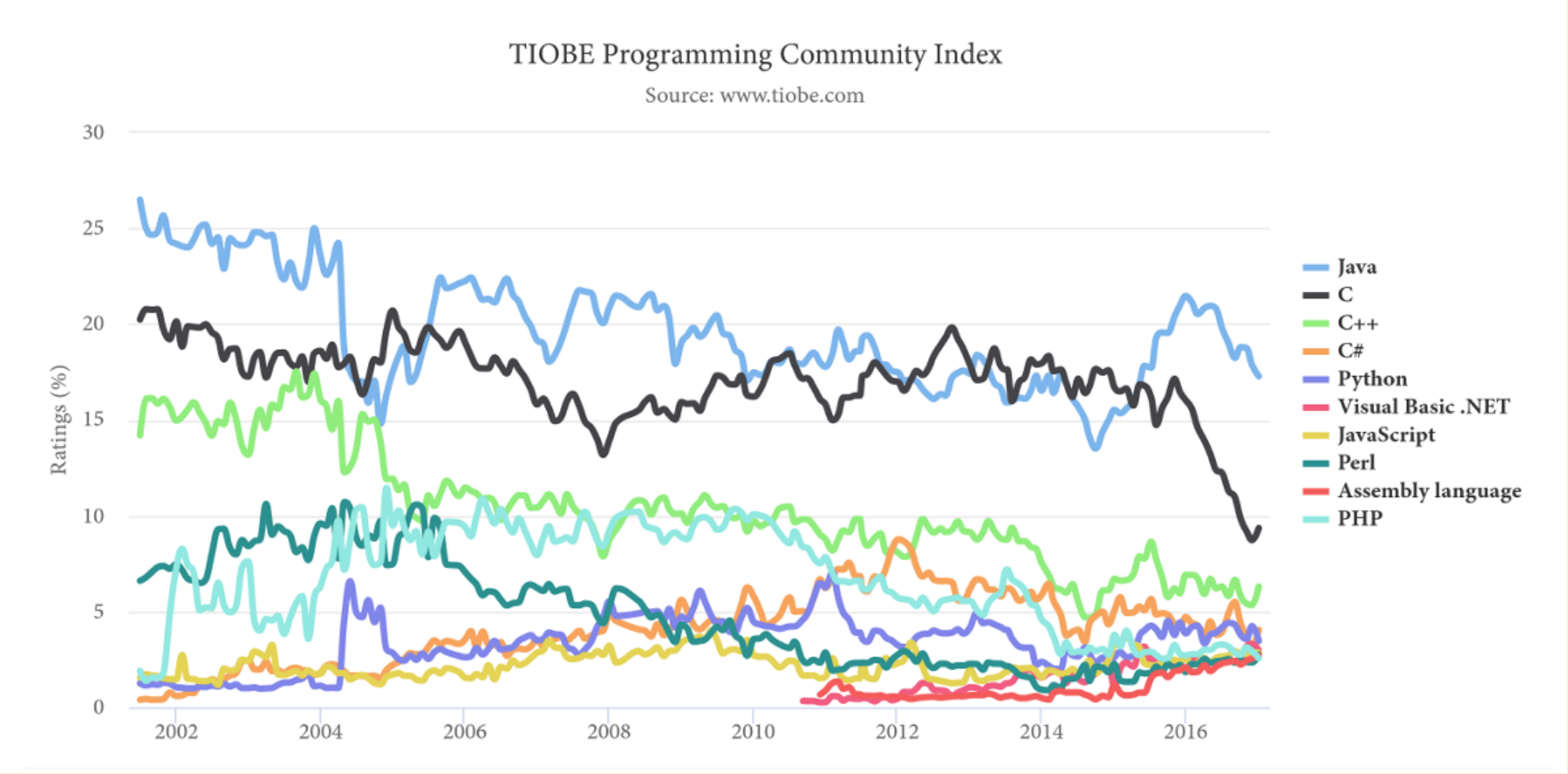
Anon.

PL 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	⌄⌄	PHP	2.564%	-0.14%
11	12	⌃	Delphi/Object Pascal	2.561%	+0.78%
12	10	⌄	Ruby	2.546%	+0.50%
13	54	⌃⌃	Go	2.325%	+2.16%
14	14		Swift	1.932%	+0.57%
15	13	⌄⌄	Visual Basic	1.912%	+0.23%

PL 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
- …

Coding style choices

Coding style choices

- *readability:* the code is easy to understand
- *reliability:* the program does what it is supposed to
- *maintainability:* it is easy to fix bugs and add new features
- *efficiency:* the program runs fast

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

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 one to use?

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 one to use?

Choose the right tools for the job!

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 one to use?

Choose the right tools for the job!

Multi-paradigm languages

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

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, IB005/IA006 Formal Languages

Semantics – the *meaning* of programs.

Tells us what behaviour we can expect from a program.

IA011 Programming Language Semantics

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, 10:00, A318**
- language: English
- video recordings will be available on IS
- some slides and lecture notes (in progress) will also be made available

Examination

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

Prerequisites

- no *formal* requirements
- knowledge of at least one of C/C++/Java
- knowledge of at least one **functional language** (Haskell, ML)
- knowledge of object-oriented programming (**OOP**)
- the more languages you know the better ☺

Study materials

Course information, additional resources

- `https://is.muni.cz/auth/el/1433/jaro2016/IA010/index.qwarp`
- Rosetta code `rosettacode.org`

Books

- 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.

Topics covered

- brief history of programming languages
- names, bindings, scope
- types, type checking, type inference
- data abstraction: abstract data types
- control flow
- subprograms (functions, methods, …)
- exceptions
- object-oriented programming
- concurrency
- functional programming