

# CS 110

## Computer Architecture

### Lecture 2: *Introduction to C, Part I*

Instructor:  
Sören Schwertfeger

<http://shtech.org/courses/ca/>

School of Information Science and Technology SIST

ShanghaiTech University

Slides based on UC Berkley's CS61C

# Agenda

- Compile vs. Interpret
- Administrivia
- Quick Start Introduction to C
- News/Technology Break
- Pointers
- And in Conclusion, ...

# Agenda

- **Compile vs. Interpret**
- C vs. Java vs. Python
- Administrivia
- Quick Start Introduction to C
- News/Technology Break
- Pointers
- And in Conclusion, ...

# Two's-Complement Examples

- Assume for simplicity 4 bit width, -8 to +7 represented

$$\begin{array}{r} 3 \quad 0011 \\ +2 \quad 0010 \\ \hline 5 \quad 0101 \end{array}$$

$$\begin{array}{r} 3 \quad 0011 \\ + (-2) \quad 1110 \\ \hline 1 \quad 1 \quad 0001 \end{array}$$

$$\begin{array}{r} -3 \quad 1101 \\ + (-2) \quad 1110 \\ \hline -5 \quad 1 \quad 1011 \end{array}$$

**Overflow when magnitude of result too big to fit into result representation**

$$\begin{array}{r} 7 \quad 0111 \\ +1 \quad 0001 \\ \hline -8 \quad 1000 \end{array}$$

$$\begin{array}{r} -8 \quad 1000 \\ + (-1) \quad 1111 \\ \hline +7 \quad 1 \quad 0111 \end{array}$$

**Overflow!**

**Overflow!**

Carry into MSB =  
Carry Out MSB

Carry into MSB  $\neq$   
Carry Out MSB

**Carry in = carry from less significant bits**  
**Carry out = carry to more significant bits**



Suppose we had a 5-bit word. What integers can be represented in two's complement?

-32 to +31

0 to +31

-16 to +15

-15 to +16

Suppose we had a 5 bit word. What integers can be represented in two's complement?

-32 to +31

0 to +31

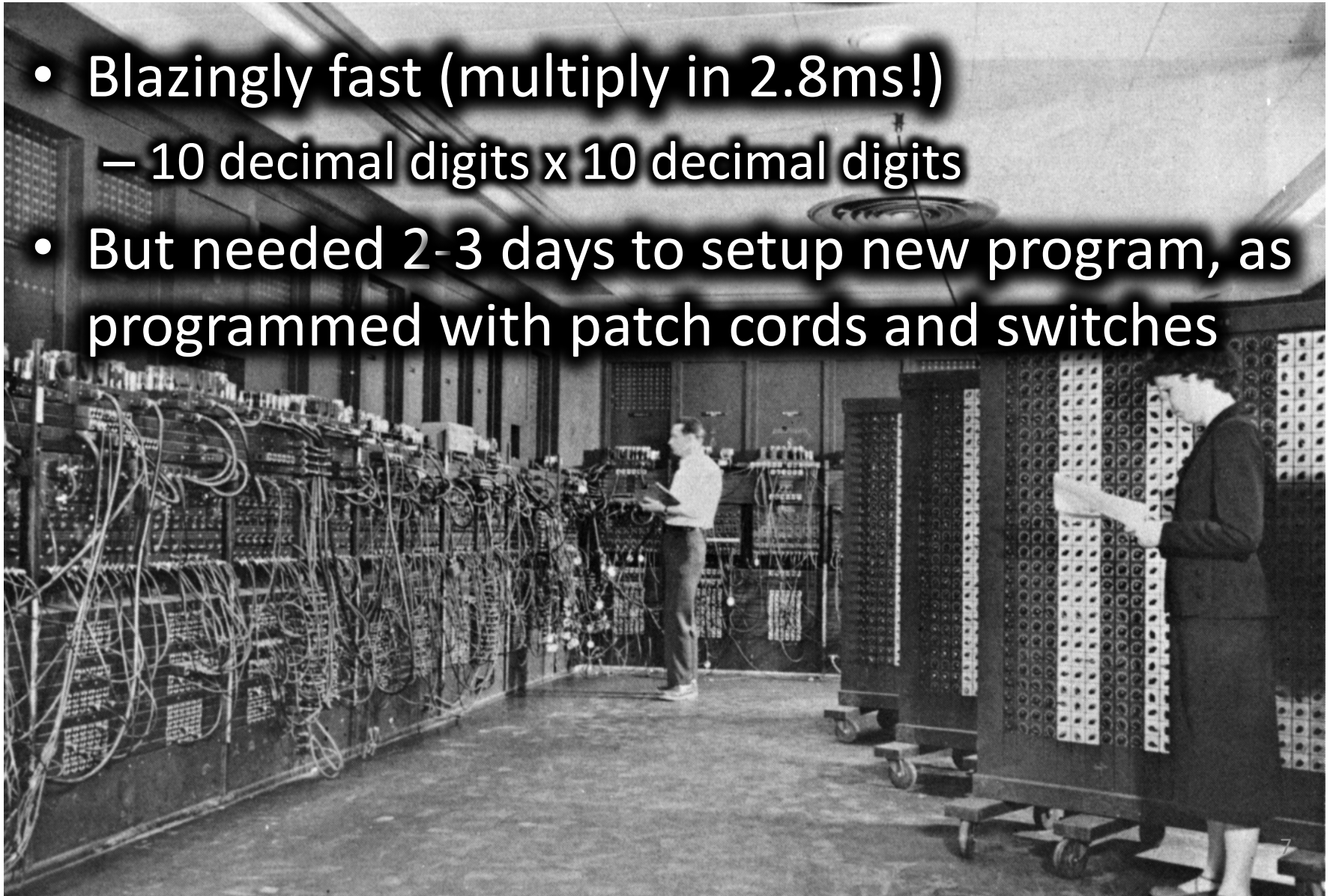
-16 to +15

-15 to +16

# ENIAC (U.Penn., 1946)

## First Electronic General-Purpose Computer

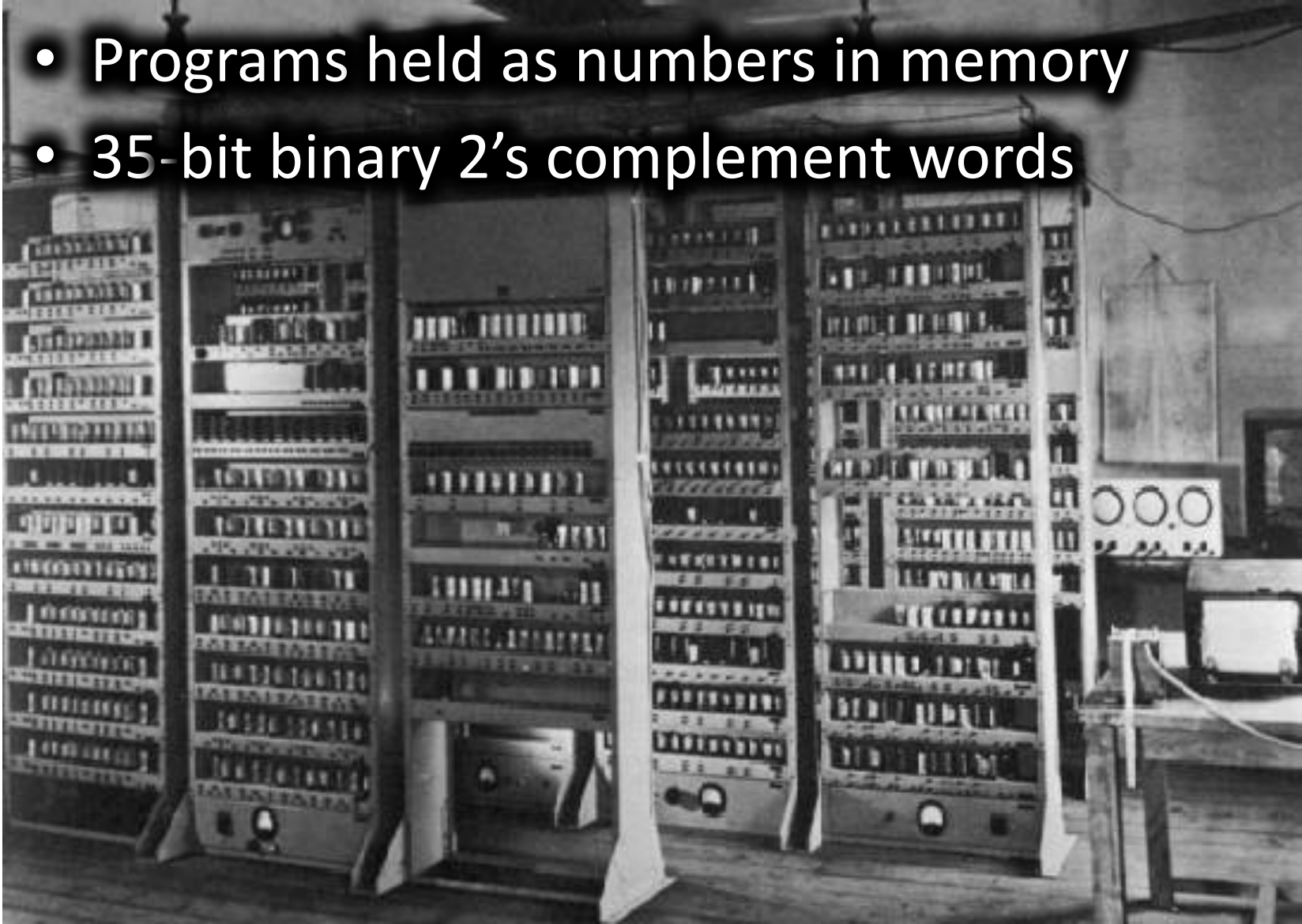
- Blazingly fast (multiply in 2.8ms!)
  - 10 decimal digits x 10 decimal digits
- But needed 2-3 days to setup new program, as programmed with patch cords and switches



# EDSAC (Cambridge, 1949)

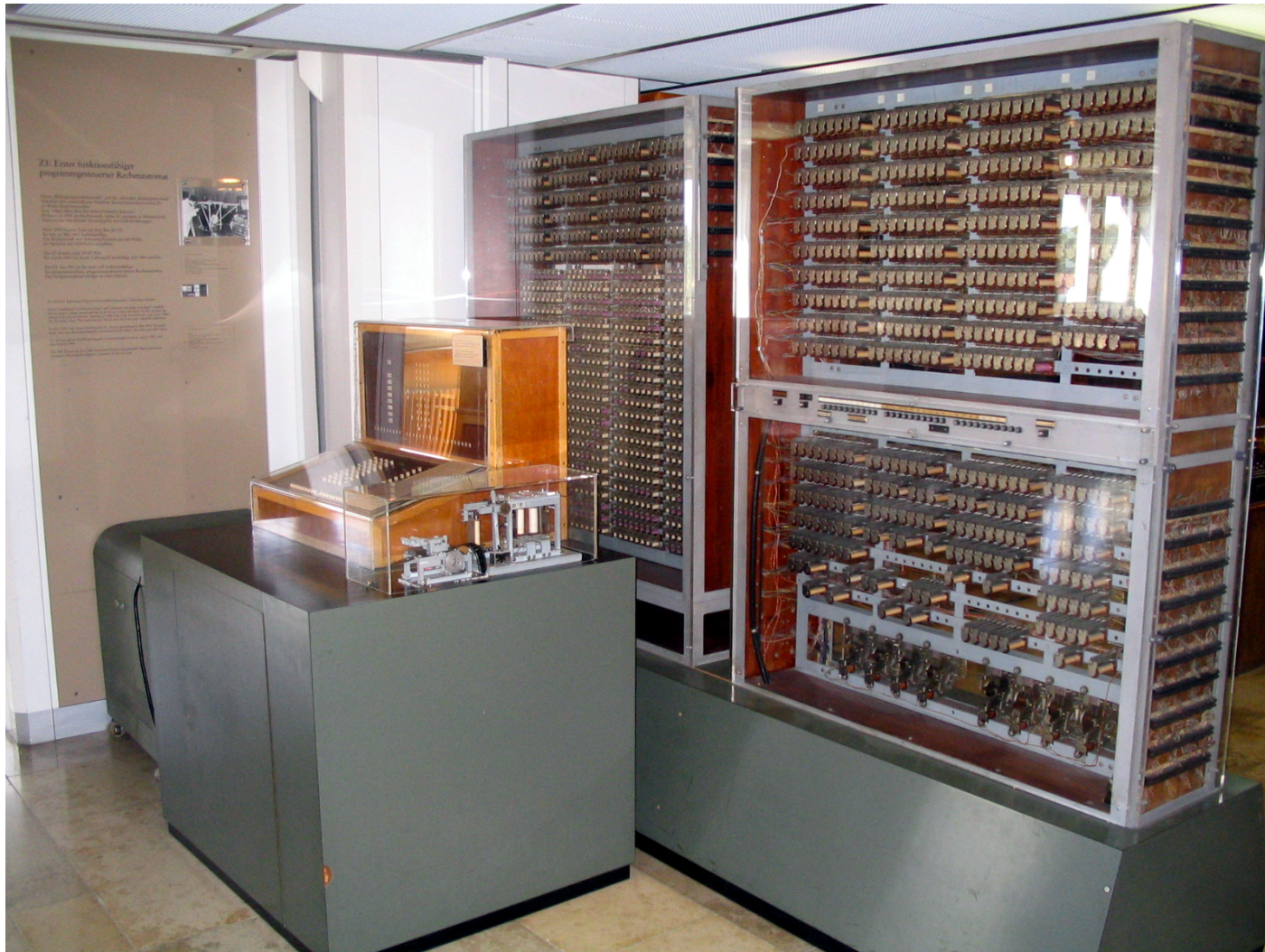
## First General Stored-Program Computer

- Programs held as numbers in memory
- 35-bit binary 2's complement words

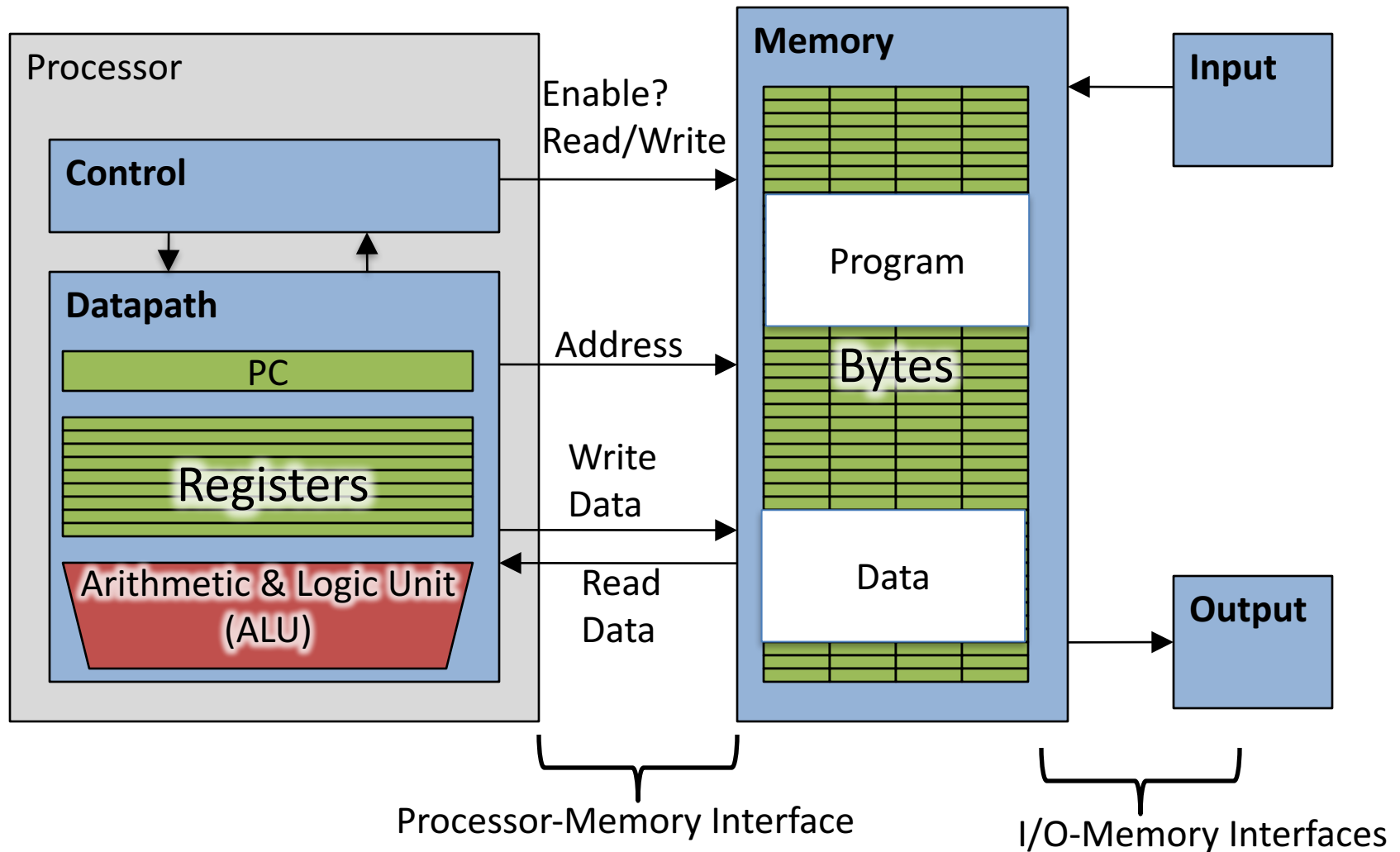




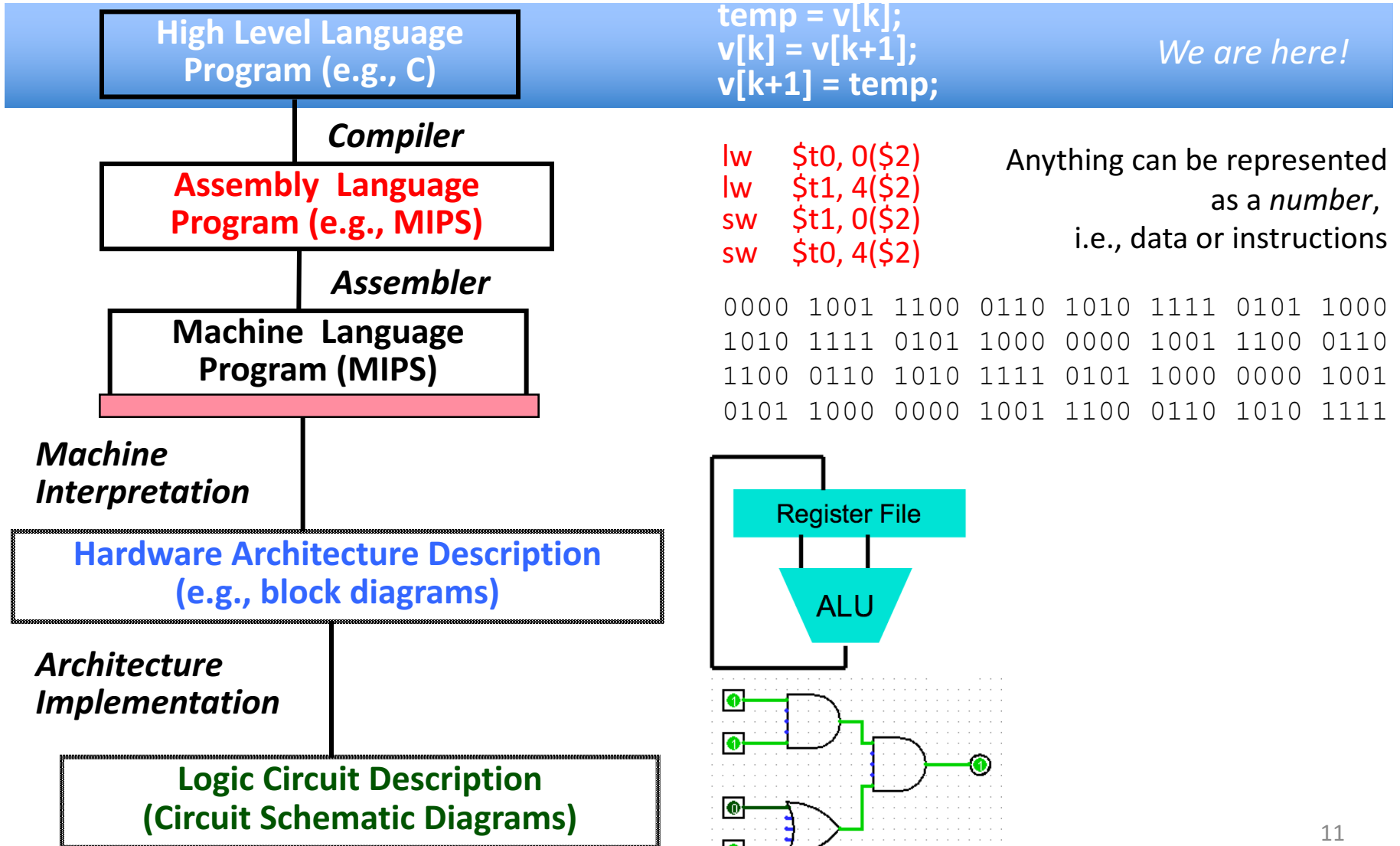
But actually: first working programmable, fully automatic digital computer: Zuse Z3 (Germany 1941)



# Components of a Computer

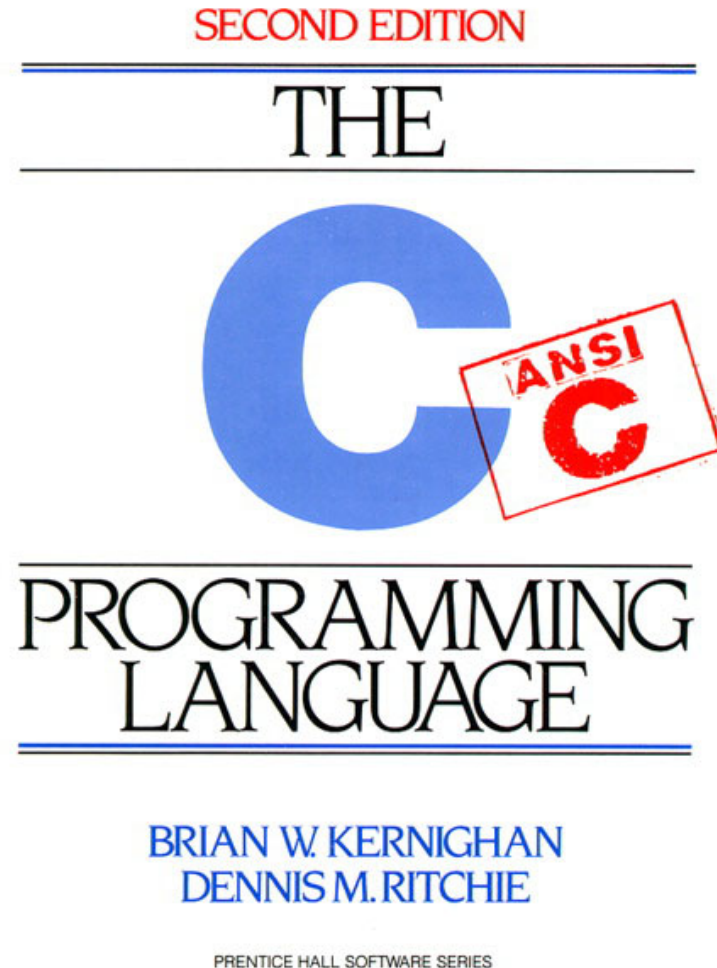


# Great Idea: Levels of Representation/Interpretation



# Introduction to C

## “The Universal Assembly Language”





# Intro to C

- *C is not a “very high-level” language, nor a “big” one, and is not specialized to any particular area of application. But its absence of restrictions and its generality make it more convenient and effective for many tasks than supposedly more powerful languages.*
  - Kernighan and Ritchie
- Enabled first operating system not written in assembly language: *UNIX* - A portable OS!

# Intro to C

- *Why C?: we can write programs that allow us to exploit underlying features of the architecture – memory management, special instructions, parallelism*
- C and derivatives (C++/Obj-C/C#) still one of the most popular application programming languages after >40 years!

# TIOBE Index of Language Popularity

Feb 2017	Feb 2016	Change	Programming Language	Ratings	Change
1	1		Java	16.676%	-4.47%
2	2		C	8.445%	-7.15%
3	3		C++	5.429%	-1.48%
4	4		C#	4.902%	+0.50%
5	5		Python	4.043%	-0.14%
6	6		PHP	3.072%	+0.30%
7	9	▲	JavaScript	2.872%	+0.67%
8	7	▼	Visual Basic .NET	2.824%	+0.37%
9	10	▲	Delphi/Object Pascal	2.479%	+0.32%
10	8	▼	Perl	2.171%	-0.08%

The ratings are based on the number of skilled engineers world-wide, courses and third party vendors.

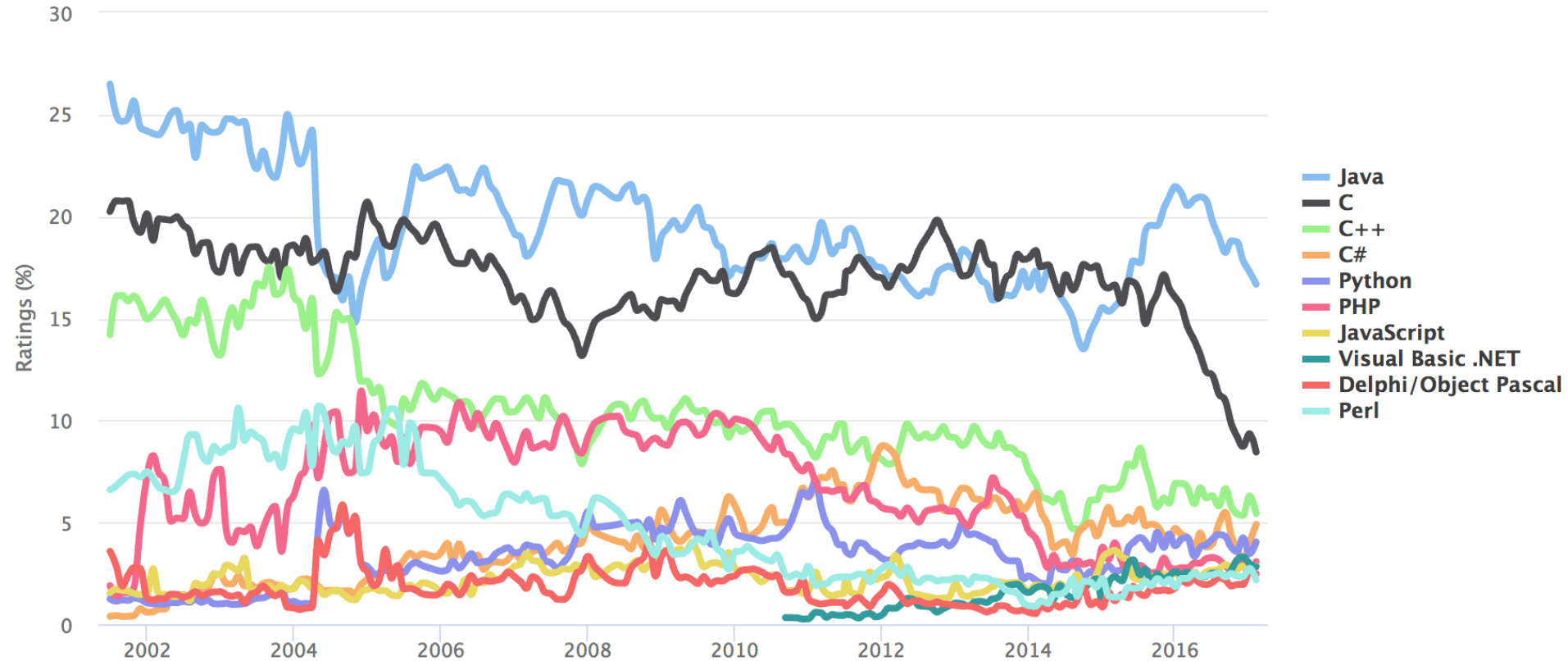
<http://www.tiobe.com>

38	Haskell	0.489%
39	Apex	0.405%
40	Rust	0.358%
41	Erlang	0.354%
42	Ladder Logic	0.342%

# TIOBE Programming Community Index

## TIOBE Programming Community Index

Source: [www.tiobe.com](http://www.tiobe.com)



# Disclaimer

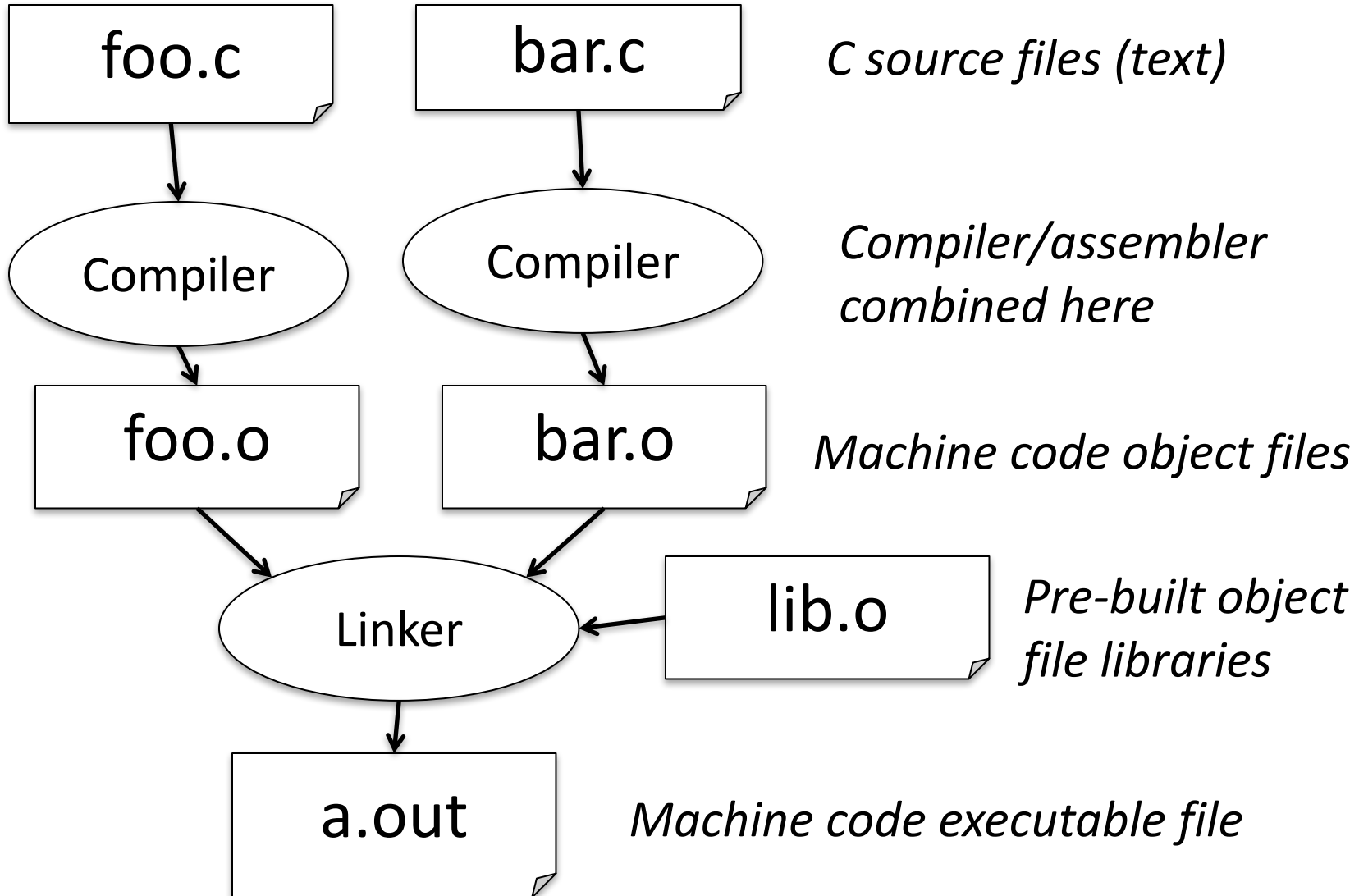
- You will not learn how to fully code in C in these lectures! You'll still need your C reference for this course
  - K&R is a must-have
    - Check online for more sources
- Key C concepts: Pointers, Arrays, Implications for Memory management
- We will use ANSI C89 – original "old school" C

# Compilation: Overview

- *C compilers* map C programs into architecture-specific machine code (string of 1s and 0s)
  - Unlike *Java*, which converts to architecture-independent *bytecode*
  - Unlike *Python* environments, which *interpret* the code
  - These differ mainly in exactly when your program is converted to low-level machine instructions (“levels of interpretation”)
  - For C, generally a two part process of compiling .c files to .o files, then linking the .o files into executables;
  - Assembling is also done (but is hidden, i.e., done automatically, by default); we’ll talk about that later

# C Compilation Simplified Overview

(more later in course)





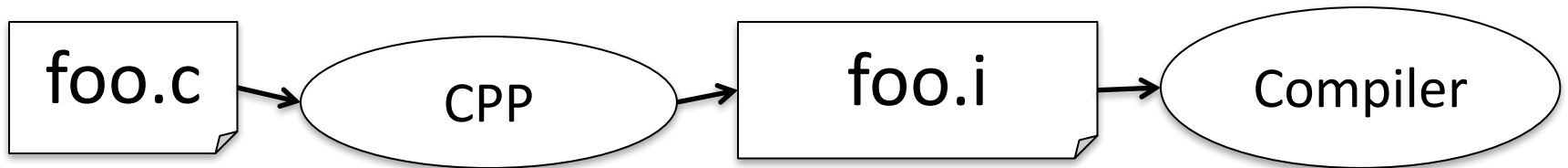
# Compilation: Advantages

- Excellent run-time performance: generally much faster than Scheme or Java for comparable code (because it optimizes for a given architecture)
- Reasonable compilation time: enhancements in compilation procedure (Makefiles) allow only modified files to be recompiled

# Compilation: Disadvantages

- Compiled files, including the executable, are architecture-specific, depending on processor type (e.g., MIPS vs. RISC-V) and the operating system (e.g., Windows vs. Linux)
- Executable must be rebuilt on each new system
  - I.e., “porting your code” to a new architecture
- “Change → Compile → Run [repeat]” iteration cycle can be slow during development
  - but Make tool only rebuilds changed pieces, and can do compiles in parallel (linker is sequential though -> Amdahl’s Law)

# C Pre-Processor (CPP)



- C source files first pass through macro processor, CPP, before compiler sees code
- CPP replaces comments with a single space
- CPP commands begin with “#”
- `#include “file.h” /* Inserts file.h into output */`
- `#include <stdio.h> /* Looks for file in standard location */`
- `#define M_PI (3.14159) /* Define constant */`
- `#if/#endif /* Conditional inclusion of text */`
- Use `-save-temps` option to `gcc` to see result of preprocessing
- Full documentation at: <http://gcc.gnu.org/onlinedocs/cpp/>

# Typed Variables in C

```
int    variable1    = 2;  
float  variable2    = 1.618;  
char   variable3    = 'A';
```

- Must declare the type of data a variable will hold
  - Types can't change

<b>Type</b>	<b>Description</b>	<b>Examples</b>
int	integer numbers, including negatives	0, 78, -1400
unsigned int	integer numbers (no negatives)	0, 46, 900
long	larger signed integer	-6,000,000,000
char	single text character or symbol	'a', 'D', '?'
float	floating point decimal numbers	0.0, 1.618, -1.4
double	greater precision/big FP number	10E100

# Integers: Python vs. Java vs. C

Language	sizeof(int)
Python	$\geq 32$ bits (plain ints), infinite (long ints)
Java	32 bits
C	Depends on computer; 16 or 32 or 64

- C: `int` should be integer type that target processor works with most efficiently
- Only guarantee: `sizeof(long long)  $\geq$  sizeof(long)  $\geq$  sizeof(int)  $\geq$  sizeof(short)`
  - Also, `short  $\geq$  16 bits`, `long  $\geq$  32 bits`
  - All could be 64 bits

# Consts and Enums in C

- Constant is assigned a typed value once in the declaration; value can't change during entire execution of program

```
const float golden_ratio = 1.618;  
const int days_in_week = 7;
```

- You can have a constant version of any of the standard C variable types
- Enums: a group of related integer constants. Ex:

```
enum cardsuit {CLUBS, DIAMONDS, HEARTS, SPADES};  
enum color {RED, GREEN, BLUE};
```

Compare “`#define PI 3.14`” and  
“`const float pi=3.14`” – which is true?

A: Constants “PI” and “pi” have same type

B: Can assign to “PI” but not “pi”

C: Code runs at same speed using “PI” or “pi”

D: “pi” takes more memory space than “PI”

E: Both behave the same in all situations

# Agenda

- Compile vs. Interpret
- C vs. Java vs. Python
- **Administrivia**
- Quick Start Introduction to C
- News/Technology Break
- Pointers
- And in Conclusion, ...



# Administrivia

- Find a partner for the lab and send your selection to Xu Qingwen (xuqw) – **due Friday, 11am!** (Submissions after that loose 25% of Lab 1 score! Write an email if you cannot find a partner!)
- Labs start next week! Check your schedule! You cannot get checked without a partner!
- The tasks for Lab 1 will be posted on the website today. Prepare for it over the weekend.
- HW1 will be posted today. Ask questions about it on piazza. Or get help during the lab or during OH.

# Agenda

- Compile vs. Interpret
- C vs. Java vs. Python
- Administrivia
- **Quick Start Introduction to C**
- News/Technology Break
- Pointers
- And in Conclusion, ...

# Typed Functions in C

```
int number_of_people ()
{
    return 3;
}

float dollars_and_cents ()
{
    return 10.33;
}

int sum ( int x, int y)
{
    return x + y;
}
```

- You have to *declare* the type of data you plan to return from a function
- Return type can be any C variable type, and is placed to the left of the function name
- You can also specify the return type as **void**
  - Just think of this as saying that no value will be returned
- Also necessary to declare types for values passed into a function
- Variables and functions **MUST** be declared before they are used

# Structs in C

- Structs are structured groups of variables, e.g.,

```
typedef struct {  
    int length_in_seconds;  
    int year_recorded;  
} Song;
```

Dot notation: **x.y = value**

```
Song song1;
```

```
song1.length_in_seconds = 213;  
song1.year_recorded     = 1994;
```

```
Song song2;
```

```
song2.length_in_seconds = 248;  
song2.year_recorded     = 1988;
```

# A First C Program: Hello World

Original C:

```
main()  
{  
    printf("\nHello World\n");  
}
```

ANSI Standard C:

```
#include <stdio.h>  
  
int main(void)  
{  
    printf("\nHello World\n");  
    return 0;  
}
```

# C Syntax: `main`

- When C program starts
  - C executable `a.out` is loaded into memory by operating system (OS)
  - OS sets up stack, then calls into C runtime library,
  - Runtime 1<sup>st</sup> initializes memory and other libraries,
  - then calls your procedure named `main ()`
- We'll see how to retrieve command-line arguments in `main()` later...