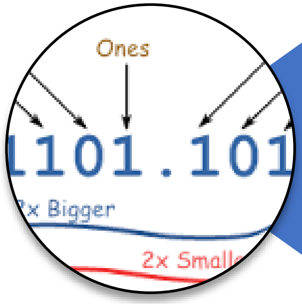
Lecture 5

Compilation and Data

Diving deeper into computer

Overview

Compilation Process



Number Bases

Data Representation on Computer

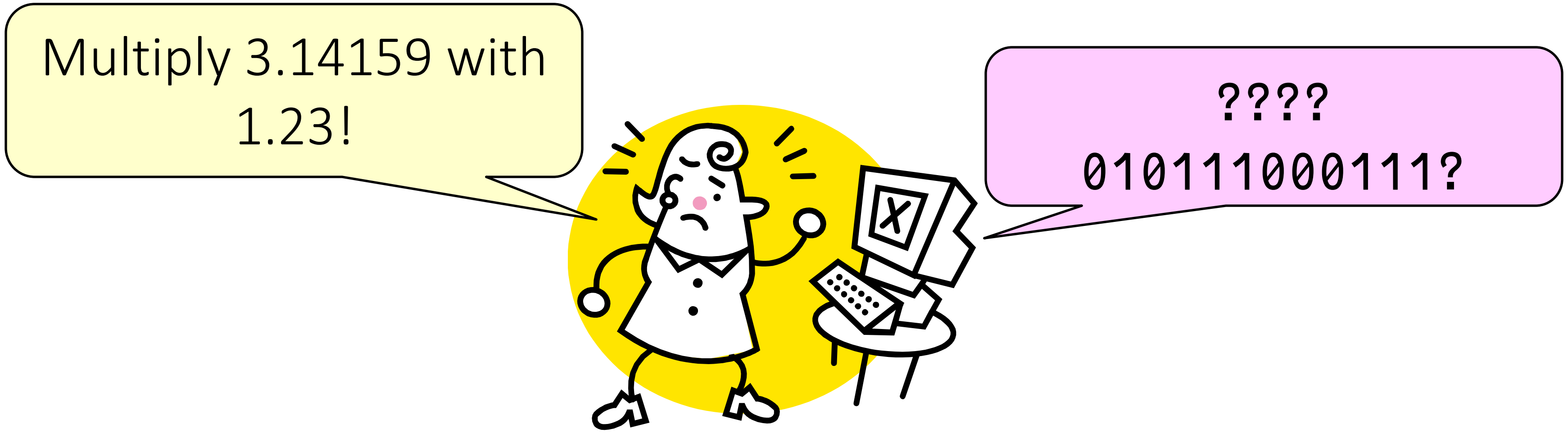
Programming Languages

At the lowest level, hardware components work on electrical signal with two states:

- On or Off (encoded as 1 or 0)
- known as binary values

Hence, instruction and data must be expressed in binary

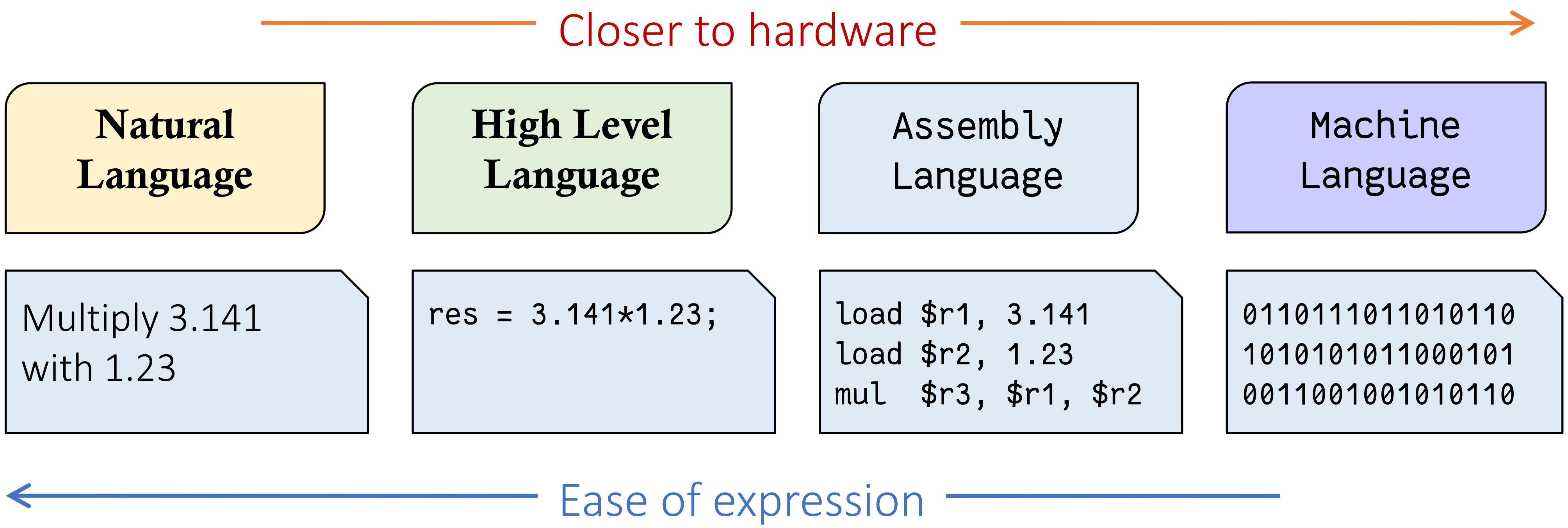
- known as Machine Language (Object Code)



Programming Languages

It is hard to express our ideas by machine language directly

– Several flavors of programming language help to bridge the gap!



High Level Language to Machine Code

Translator

- Accepts a program written in a source language and translates it to a program in a target language

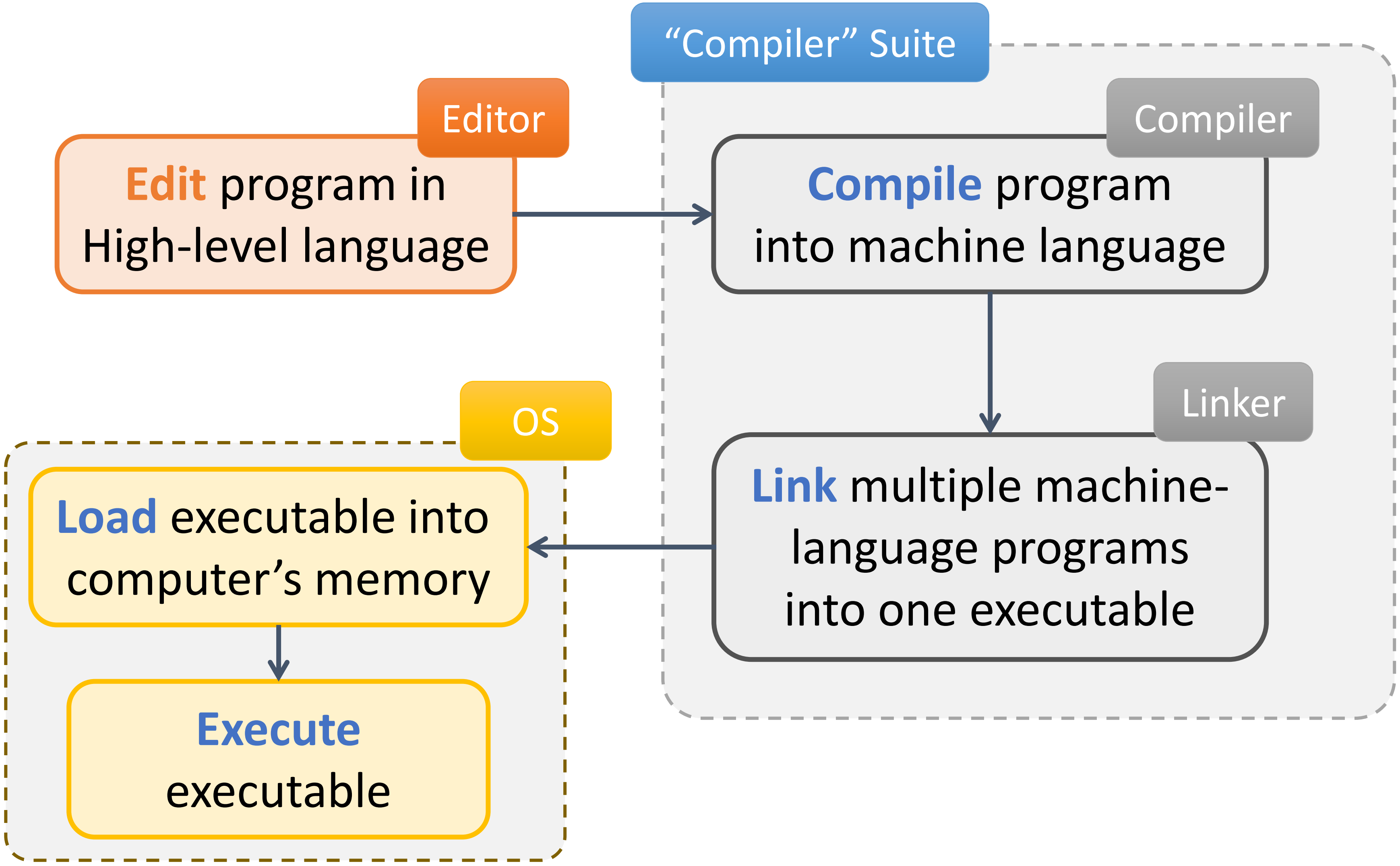
Compiler

- Standard name for a translator whose source language is a high-level language

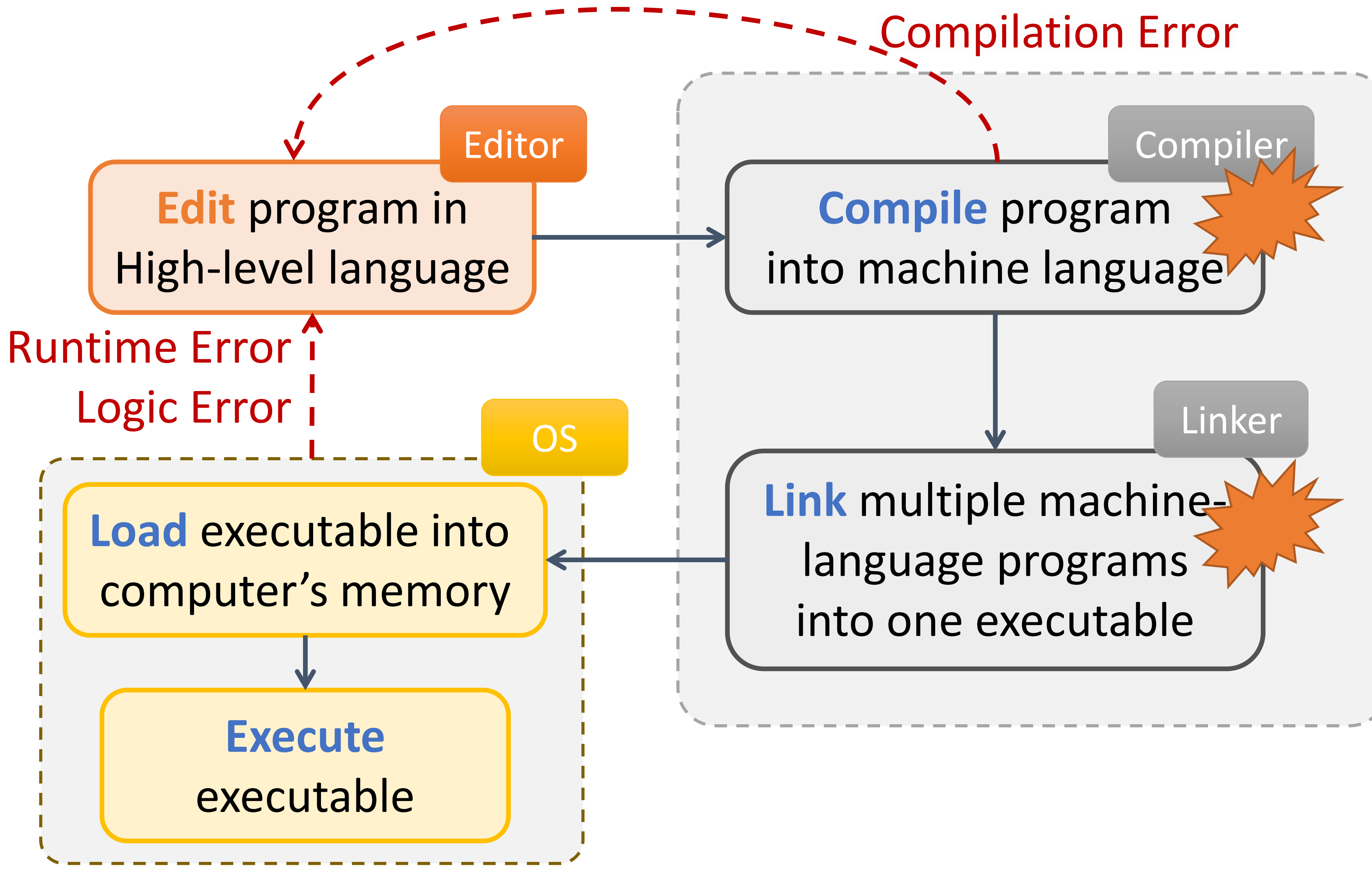
Interpreter

- Evaluates and executes a source program

Recap: The *Life* of a program



Run Cycle for Compiled Language



How does the Compiler checks syntax?

Facts:

– There are infinite number of valid C/C++ programs!

– Compiler itself is just a program

? Question: How can compiler understand all valid programs?

Example: Simplified C expression

– Only supports variables, + and –

– Examples:

```
a + c - e
total - discount + tax
a + c -
total + payment * tax
-a + c
```

invalid: dangling – at the end

invalid: * not supported ☺

invalid: negative not supported

Grammar: A language of language

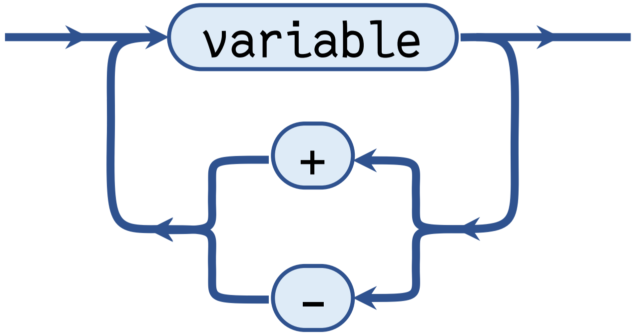
– Grammar defines the structure of a language

In Computer Science, context-free grammar is used

– A set of rules, each rule defines a name and its expansion

– Example

`expression:`



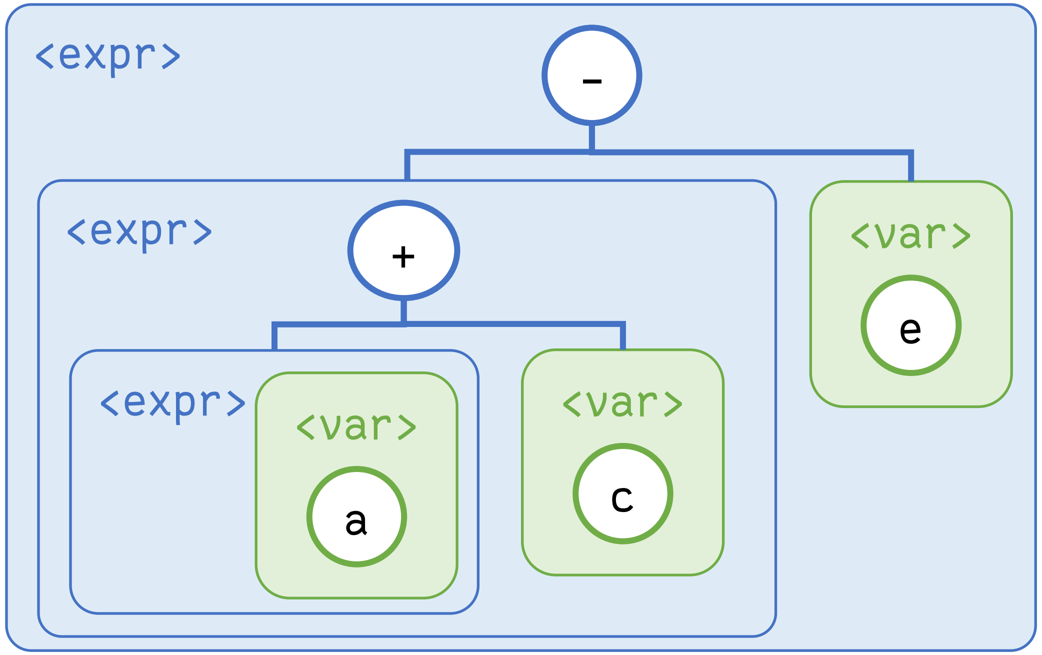
Backus-Naur Form (BNF)

A common grammar notation in Computer Science

```
<expression> ::= <variable>
               | <expression> "+" <variable>
               | <expression> "-" <variable>
```

Example:

a + c – e



Syntax Error: Compiler complains!

- When a C program violates one of the grammar rules
  - Syntax error

Error message	Meaning
`expecting ";" before ...`	Missing ";" Tips: Check the lines above
`"XYZ" undeclared (first use in this function)`	The variable XYZ is not declared
`expected declaration or statement at the end of input`	Common cause: mismatched }
`missing terminating " character`	The matching " is missing in a string
`XYZ: No such file or directory`	Common cause: Misspelt the library name in “`#include <...>`”

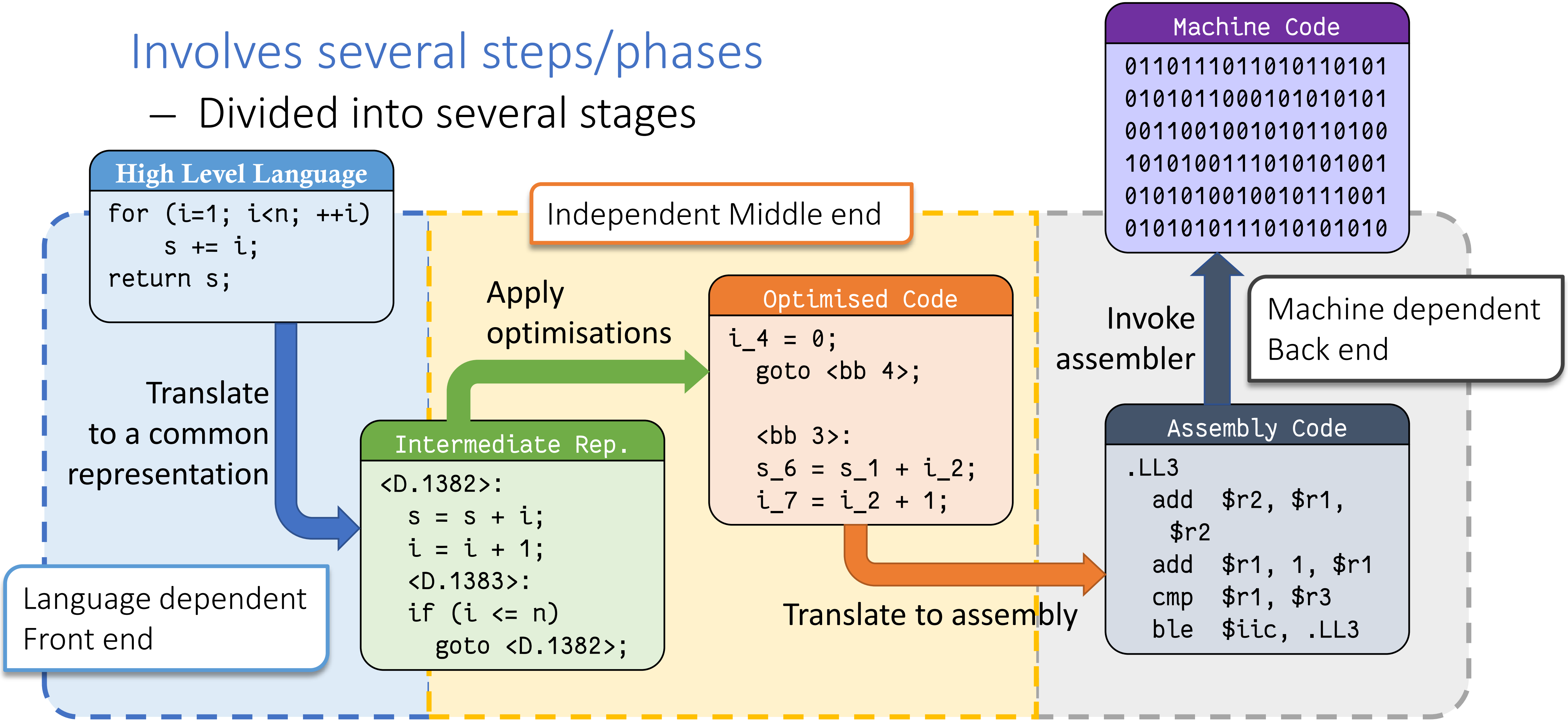
The Compilation Process

Compiler translate the high-level program into machine code

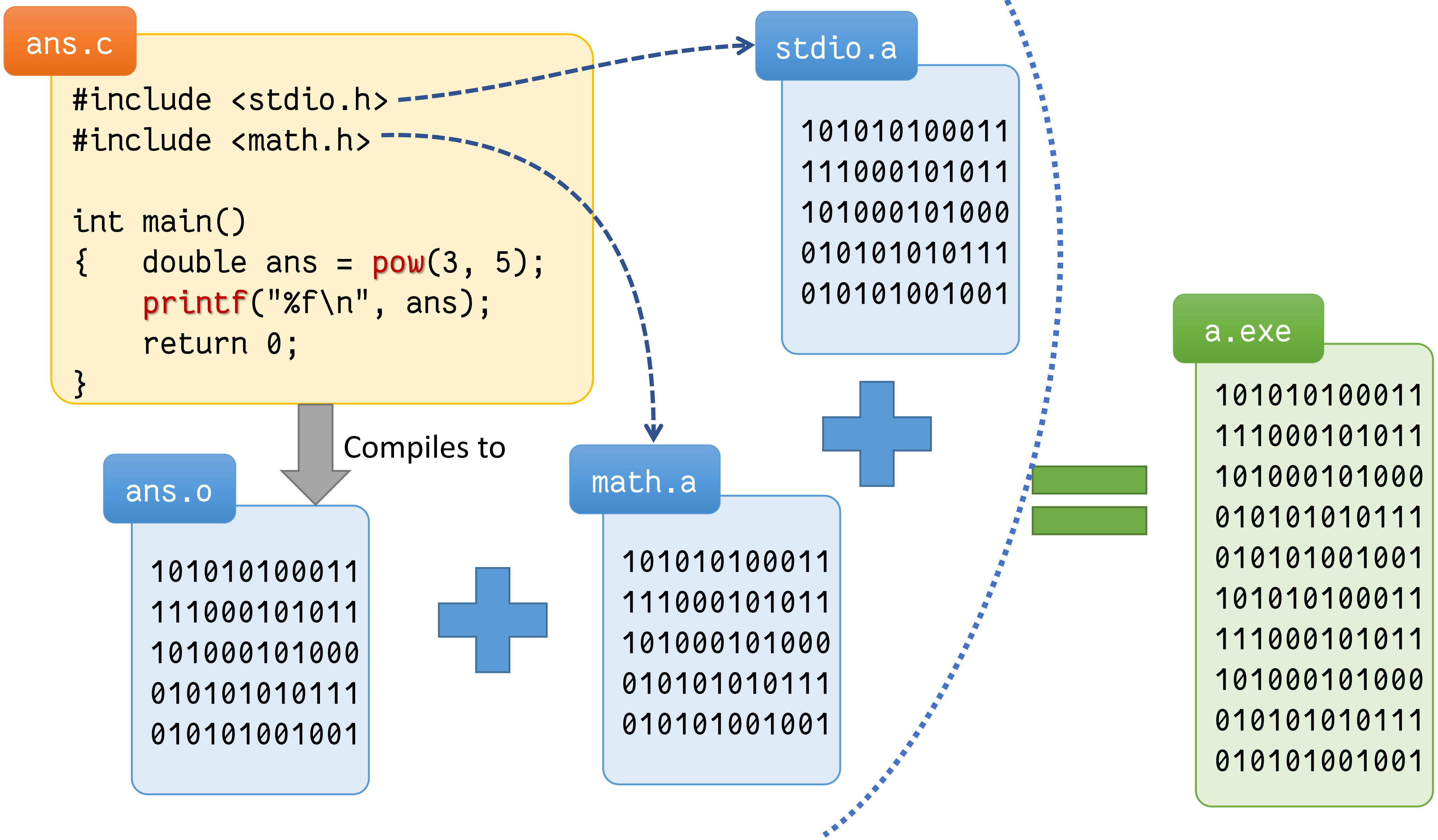
- once all syntax error have been resolved
- sometimes compiler will give warnings, depending on the configuration

The Compilation Process

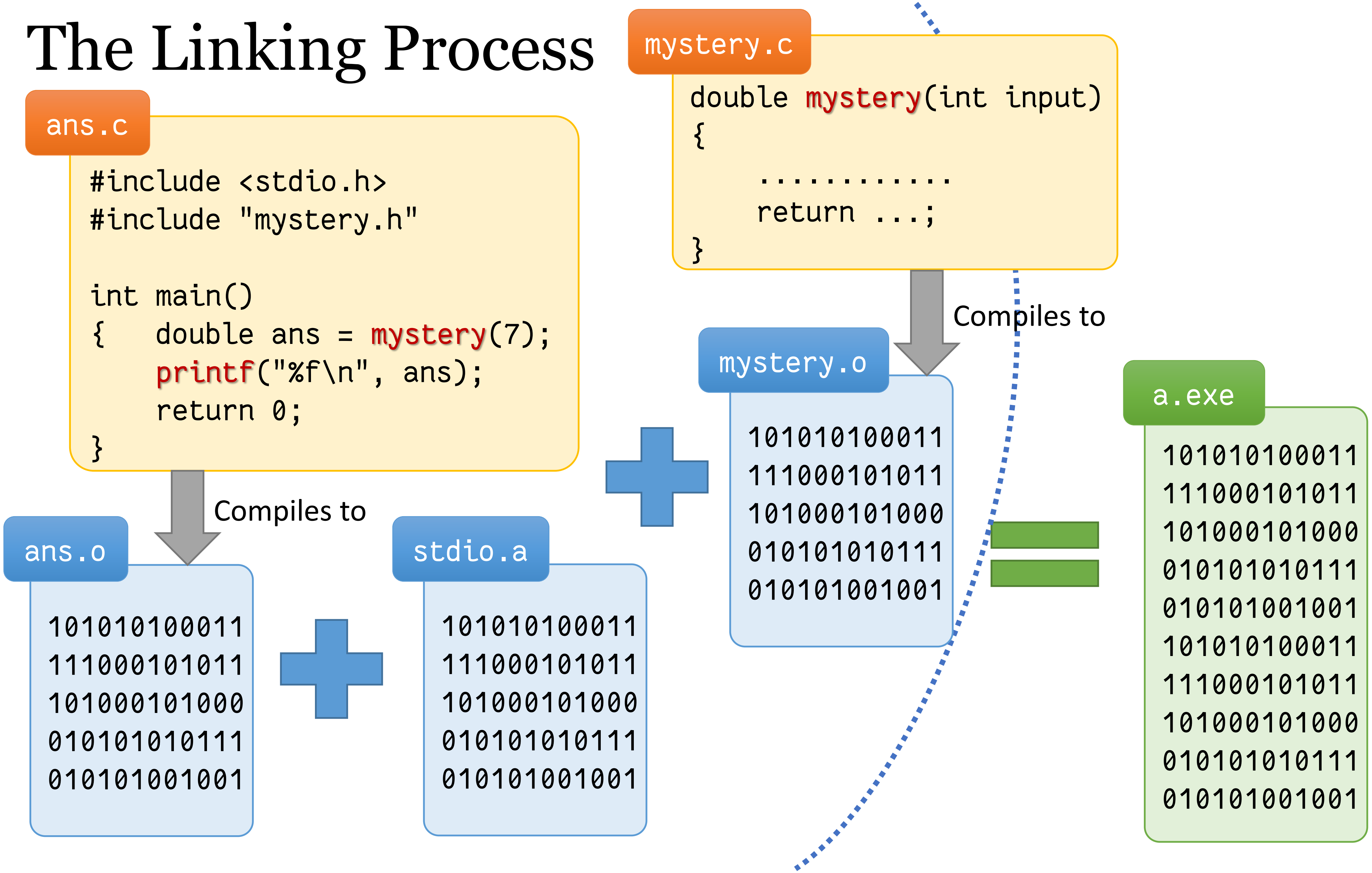
- Involves several steps/phases
  - Divided into several stages



The Linking Process



The Linking Process



The Linking Process

The linking process resolves dependency

- between multiple object codes

Example:

- The main program uses `printf()` and `pow()`, where are these functions?

Also, we can split a big program into multiple C source codes

- Separate `.h` header files that only contain function prototypes (declarations)
- Linker again helps to resolves the usage between these separate source programs

Linking Error: Not found!

- When the linker could not resolve the dependency → Linking error
  - Reported by the compiler suite and may be mistaken as "compilation" error

Error message	Meaning
`In function XYZ:` `undefined reference to 'ABC'`	Function XYZ make a call to ABC() function, but that function source code cannot be found! **Common cause: misspelling of function name**

`expected 'XYZ' but argument is of type ABC:`
 	
`<some function header shown here>`
	The parameter passed to a function is of the incompatible data type or incorrect number of parameters.
	Common cause: incorrect parameters in function call.

Yay, It compiles! But……

If an executable is produced, then there is no compilation error

- Note that compiler warning is not an error

When your executable starts running, there are two other kind of errors to worry about:

1. Runtime Error
2. Logic Error

Runtime Error: Program Crashes

There are certain operations that will cause your program to crash (terminate prematurely)

Common example:

- Division by zero
- Memory error (unlikely to happen now, may encounter later in the course)

Logic Error: Wrong Result

Program does not crash, but the result or the behavior is wrong

- Logic Error

Note that infinite loop is a result of logic error too:

- The execution behavior (never terminates) is wrong!

Interpreted Language

There are also interpreted programming languages

- Instead of a separate compilation process, program statements are translated and executed in real time
- Analogy: Translating a book (compiler), Translating a speech in real time (interpreter)

Example:

- Python, Ruby, JavaScript, SQL, etc
- Some languages support both compilation and interpretation modes

How was GCC compiled?

#trivia

A compiler is also a computer program

- It had to be itself compiled into machine code
- A compiler is needed to compile the compiler

Ans: GCC is compiled using GCC

- How does that work?!
- Analogy: An English language is explained with an English dictionary

How then is a new language created?

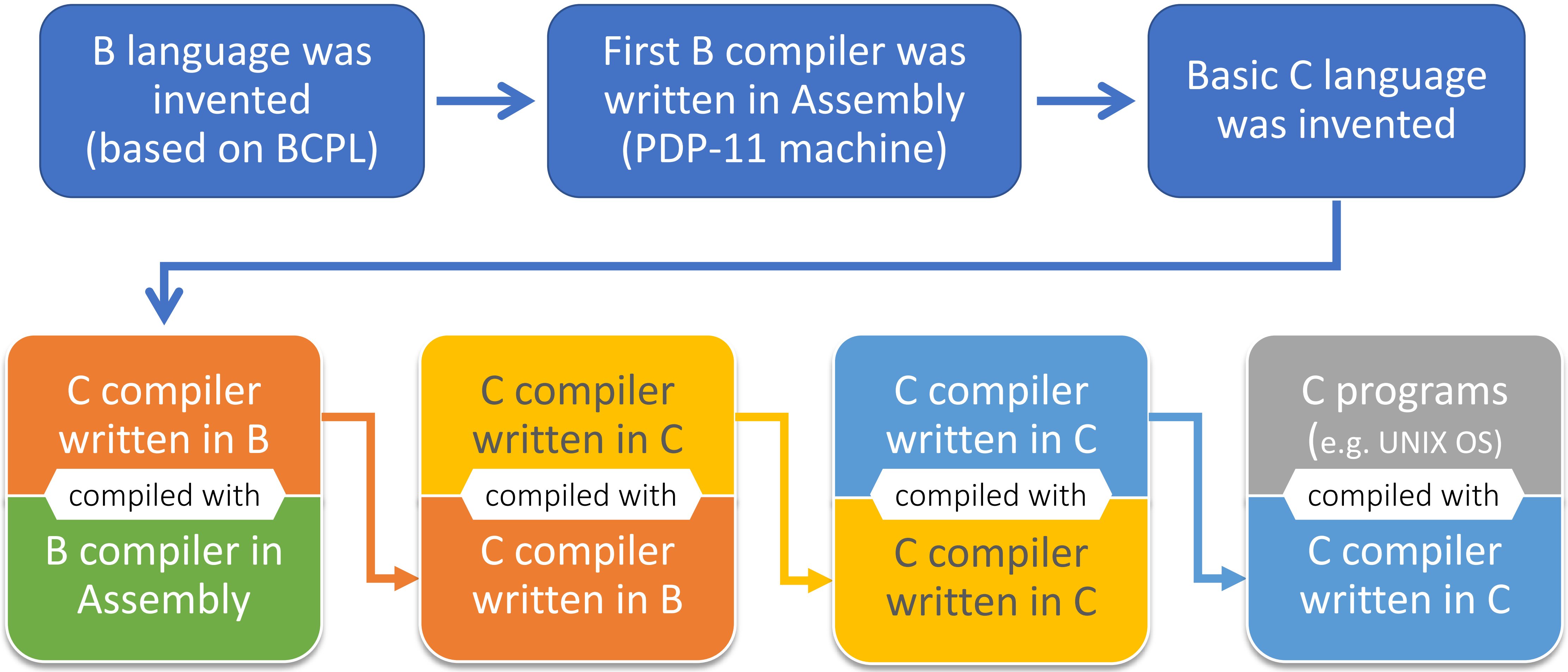
Bootstrapping: Creating a New Language

#trivia

- Suppose we want to invent a new language
  - E.g. Klingon, Elvish
- Invent a grammar and few simple phrases of Klingon
- Explain it in another language
  - Use English to explain grammar rules and simple phrases
- Now you have a simplified form of Klingon
  - Use simple phrases to explain more difficult phrases
  - E.g. a Klingon dictionary written in Klingon

Bootstrapping a native C compiler

#trivia



Number Bases

So, 1 + 1 = 10?

Numeral Systems

Ways to express numbers

- Notation for representing numbers

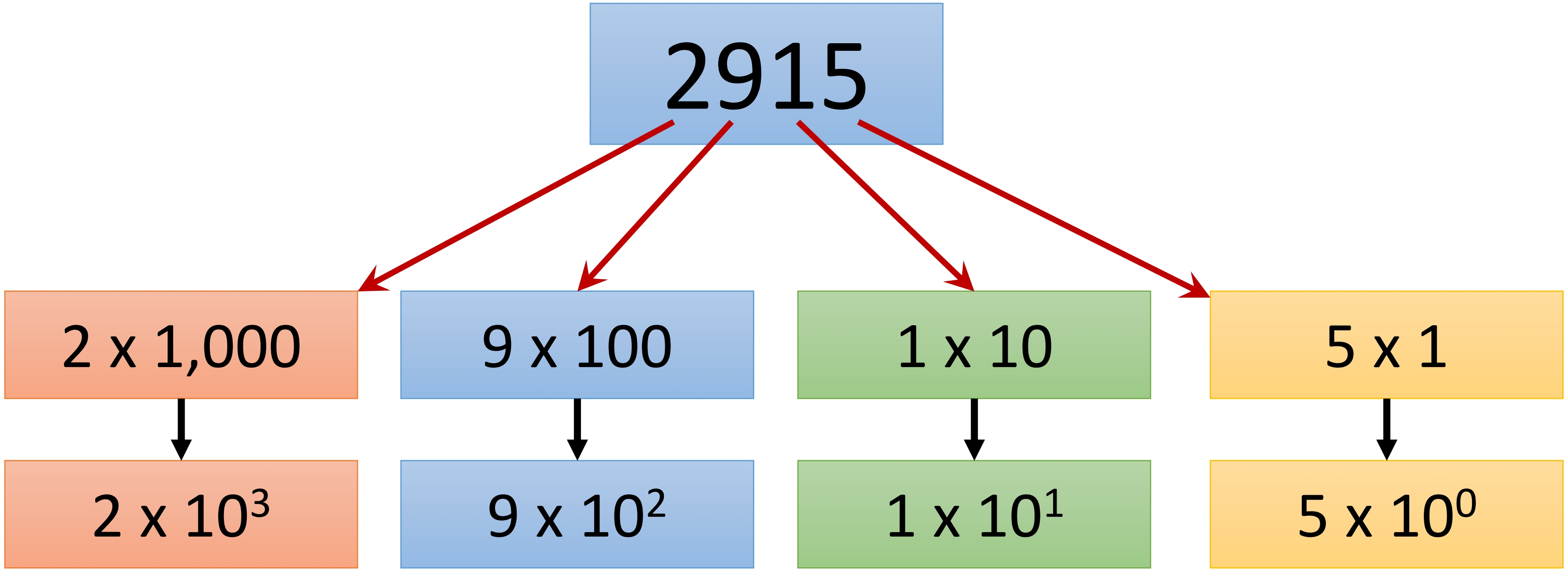
Hindu-Arabic	Roman
1	I
5	V
10	X
50	L
100	C
500	D
1,000	M

Positional Weighted System

Position of digit in a number carries different weight

Example (Base 10):

– Digits in base 10 = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}



Formal Definition

$$(a_n a_{n-1} \dots a_0 \,.\, f_1 f_2 \dots f_m)_{10} =$$

$$(a_n \times 10^n) + (a_{n-1} \times 10^{n-1}) + \cdots + (a_0 \times 10^0) +$$

$$(f_1 \times 10^{-1}) + (f_2 \times 10^{-2}) + \cdots + (f_m \times 10^{-m})$$

– The base (radix) in the above formula is 10

What if we change it to other bases?

– Give rise to different number bases (number systems)

Number System Examples

Base	Name	"Digits"	Example Number	Value in base 10
10	Decimal (Denary)	0 1 2 3 4 5 6 7 8 9	1011_{10}	$1 \times 10^3 + 1 \times 10^1 + 1 \times 10^0$ $= 1011_{10}$
2	Binary	0 1	1011_2	$1 \times 2^3 + 1 \times 2^1 + 1 \times 2^0$ $= 11_{10}$
3	Ternary	0 1 2	1011_3	$1 \times 3^3 + 1 \times 3^1 + 1 \times 3^0$ $= 31_{10}$
8	Octal	0 1 2 3 4 5 6 7	1011_8	$1 \times 8^3 + 1 \times 8^1 + 1 \times 8^0$ $= 521_{10}$
16	Hexadecimal	0 1 2 3 4 5 6 7 8 9 A B C D E F	1011_{16}	$1 \times 16^3 + 1 \times 16^1 + 1 \times 16^0$ $= 4113_{10}$

Decimal → Binary: Repeated Division-By-2

Successively divide by 2 until the quotient is 0

The remainders form the answer

- The first remainder as the least significant bit (LSB)
- The last as the most significant bit (MSB).

2	43	
2	21 r 1	← rightmost (LSB)
2	10 r 1	
2	5 r 0	
2	2 r 1	
2	1 r 0	
	0 r 1	← leftmost (MSB)

$$43_{10} = 101011_2$$

Observation: Base-2 and Base-8

Binary	Octal
0	0
1	1
10	2
11	3
100	4
101	5
110	6
111	7
1 000	10
1 001	11
1 010	12

Binary	Octal
1 011	13
1 100	14
1 101	15
1 110	16
1 111	17
10 101 010	252
11 101 100	354
1 000 001	101
111 110	76
1 000 101	105
101 010	52

Observation: Base-2 and Base-16

Binary	Hexadecimal
0	0
1	1
10	2
11	3
100	4
101	5
110	6
111	7
1000	8
1001	9
1010	A

Binary	Hexadecimal
1011	B
1100	C
1101	D
1110	E
1111	F
1 0000	10
1 0001	11
1 0010	12
1 0011	13
1 0100	14
1 0101	15

Binary	Hexadecimal
1 1110	1E
1 1111	1F
10 0000	20
10 0001	21
10 0010	22
1010 1010	AA
1110 1100	EC
100 0001	81
11 1110	3E
100 0101	45
10 1010	2A

Data Representation in Computers

Computer Data Storage

Information is stored as binary values in a computer:

- **Everything**: Picture, video, application, game, etc

Common storage units:

- bit = a single `0` or `1`
- Byte = 8 bits
  - Usually the smallest accessible unit
- Word = 4 bytes (32 bits) or 8 bytes (64 bits)
  - Platform dependent
  - Double word = 2 x word, Halfword = word / 2

Common C Data Types Mapping

The number of bits determine:

– [int, long] The range of integers
– [float, double] The accuracy of floating point numbers

Data Type	32-bit Processor	64-bit Processor
`int`	4 bytes (32 bits)	8 bytes (64 bits)
`long`	8 bytes (64 bits)	8 bytes (64 bits)
`float`	4 bytes (32 bits)	8 bytes (64 bits)
`double`	8 bytes (64 bits)	16 bytes (128 bits)
`char`	1 byte (8 bit)	

gcc compiles for 32-bit processor by default

Data Representation

For most data types, we need to figure out how to represent the values using the limited bits

- Known as **data representation**

Examples:

- Integer (both –ve and +ve): 2s Complement
- Floating point number: IEEE 754 representation
- Character: ASCII or Unicode

Representing Integers

Non-negative integers

- `unsigned` : Add to type declaration to create non-negative integers

What about negative integers?

- Reserve MSB (leftmost) to indicate +ve or −ve
- E.g. $\texttt{10000001}_2$ represents -1, $\texttt{00000001}_2$ represents 1

But, zero is doubly represented

- `00000000` or `10000000`
- Not efficient in arithmetic operations

Integer: 2s-Complement (Overview)

2s-Complement

- Unique representation of all numbers
- Efficient: subtraction can be achieved by using only addition

MSB (leftmost kth bit) represents -2^{k-1}

- E.g. $1010\ 0101_2 = -2^7 + 2^5 + 2^2 + 2^0$

$$= -128 + 32 + 4 + 1$$
$$= -91$$

Range of k-bit integer

- -2^{k-1} to $2^{k-1} - 1$
- E.g. 16-bit integer range is $-32{,}768$ to $32{,}767$

Representing Fractions

- Every number base has accuracy limitations
  - The more "digits", the more accurate the value
  - E.g. $^1/_5 = 0.2$, but $^1/_3 \cong 0.333333$
- Fixed point representation
  - Fixed digits for integer and fraction part
  - E.g. in base 10, use 4 digits for each $\boxed{0|0|1|5}.\boxed{1|2|5|0}$
- Range is limited
  - Largest value is 9999.9999
  - Closest to zero is 0000.0001

Floating Point Representation

Allow the point to "float"

– Largest value: $\boxed{9}\boxed{9}\boxed{9}\boxed{9}\boxed{9}\boxed{9}\boxed{9}\boxed{9}.$

– Closest to zero: $.\boxed{0}\boxed{0}\boxed{0}\boxed{0}\boxed{0}\boxed{0}\boxed{0}\boxed{1}$

An alternate form

– Express in scientific form: $m \times 10^n$

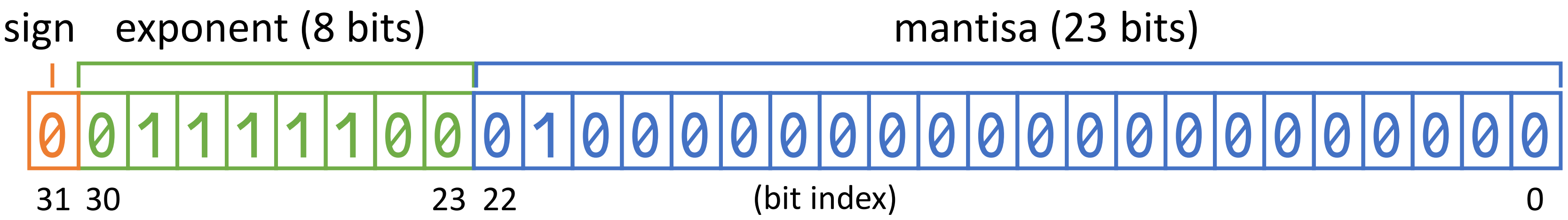
$$\underbrace{\boxed{1}.\boxed{2}\boxed{3}\boxed{4}}_{\text{Mantisa}} \times \underbrace{10^{\boxed{1}\boxed{0}\boxed{0}\boxed{0}}}_{\text{Exponent}}$$

– Largest value: 9.999×10^{9999}

– Closest to zero: 0.001×10^{-999}

Floating Point: IEEE 754

Single-precision floating point format



- sign bit: 0 = +ve, 1 = -ve
- exponent: actual exponent +127 (i.e. bias-127)
- fraction: normalized to 1.X and take X only

Floating Point: Effect of limited bits

Even in Denary system:

- With a precision of 5 digits, we cannot represent the following numbers fully:
  - PI = 3.14159<u>265359….</u>
  - 1 / 3 = **0.33333**<u>3333333….</u>

Hence, there are numbers that cannot be represented accurately if we have limited storage

- The only difference on computer system is that we need to look at the binary representation of those numbers

Character: ASCII Code

Character includes:

- Printable: ‘A’…’Z’, ‘a’…’z’, ‘0’…’9’, ‘ ’, ‘@’, ‘#’ ….
- Unprintable: NULL, bell, tab, return, ….

Originally defined as a 7-bit sequence

- 0 to 127: 128 characters are represented
- American Standard Code for Information Interchange

Subsequently extended to 8-bit

- the extended range 128 to 255 can have platform dependent encoding

ASCII Code Chart

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Examples:

- 20_{16} = Space ‘ ’
- 41_{16} = ‘A’; 61_{16} = ‘a’
- 30_{16} = ‘0’; 39_{16} = ‘9’

Picture taken from wikipedia

Character: Unicode

Computing industry standard

- maintained by Unicode Consortium
- latest version: 12.1.0 (May 2019)
- 137,994 characters across 129 scripts + symbol sets

137,994 means at least 3 bytes of storage

- $2^{16} = 65{,}536$; $2^{24} = 16{,}777{,}216$

For backward compatibility (among other reasons), a number of encoding schemes are proposed:

- UTF-8, UTF-16, UTF-32, etc….

Character: UTF-8

Proposed to be compatible with ASCII

- The first 127 values is exactly the same as ASCII
- Variable length: 1 byte to 4 bytes

Bits of code point	First code point	Last code point	Bytes in sequence	Byte 1	Byte 2	Byte 3	Byte 4
7	U+0000	U+007F	1	0xxxxxxx			
11	U+0080	U+07FF	2	110xxxxx	10xxxxxx		
16	U+0800	U+FFFF	3	1110xxxx	10xxxxxx	10xxxxxx	
21	U+10000	U+1FFFFF	4	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

Picture taken from wikipedia

Emoji: The universal language?

Emoji are supported as part of Unicode

– See http://unicode.org/emoji/charts/full-emoji-list.html

№	Code	Browser	Appl	Googd	Twtr.	One	FB	FBM	Sams.	Wind.
1	U+1F600									
2	U+1F601									
3	U+1F602									
4	U+1F923							—		
5	U+1F603									

Other Encoding Schemes

Anything processed by computer requires to represent the information in binary:

- Knowing this should shed lights to many “mysterious” numbers frequently seen in the real world

Examples:

- Harddisk capacity: 1GB = how many bytes?
- Network speed: 100MBps = how many bytes per second?

Let’s look at a few common encoding

RGB code: More colors than rainbow

Additive color model

- By mixing the three primary colours Red, Green and Blue
- Can produce any colour discernible by humans

Used in computer to represent colour

- Quantized into discrete values
- Each value represent the saturation of RGB
- A triplet (R, G, B) → a particular colour
- E.g. 8-bit for each (R,G,B) → 24-bit colour → 16.7 million colours

Common extension: RGBA (RGB + Alpha)

IP Address: Your address on the web

Do you know the meaning of these numbers?

– 64.233.160.0   104.16.2.108

They are IP (Internet Protocol) addresses

– Essentially a unique logical address of a computer in a network
– A quadruplet of 1-byte number, i.e. 4 x 8 = 32-bit

The IP addresses are "grouped" by prefix

– The left-most part of the address

MAC Address: Hardware address

Media Access Control address

- Unique address for all network interfaces
- Not supposed to change (in theory)

Common notation:

- 6 groups of 2 hexadigits, separated by ":" or "-"
- e.g. "AB:CD:EF:12:34:56"

The left most 3 groups uniquely identifies the manufacturers, vendor or organization globally

END