

CS 211: Intro to Computer Architecture

2.1: Memory, C Objects, and Data Representation

Minesh Patel

Spring 2025 – Tuesday 28 January

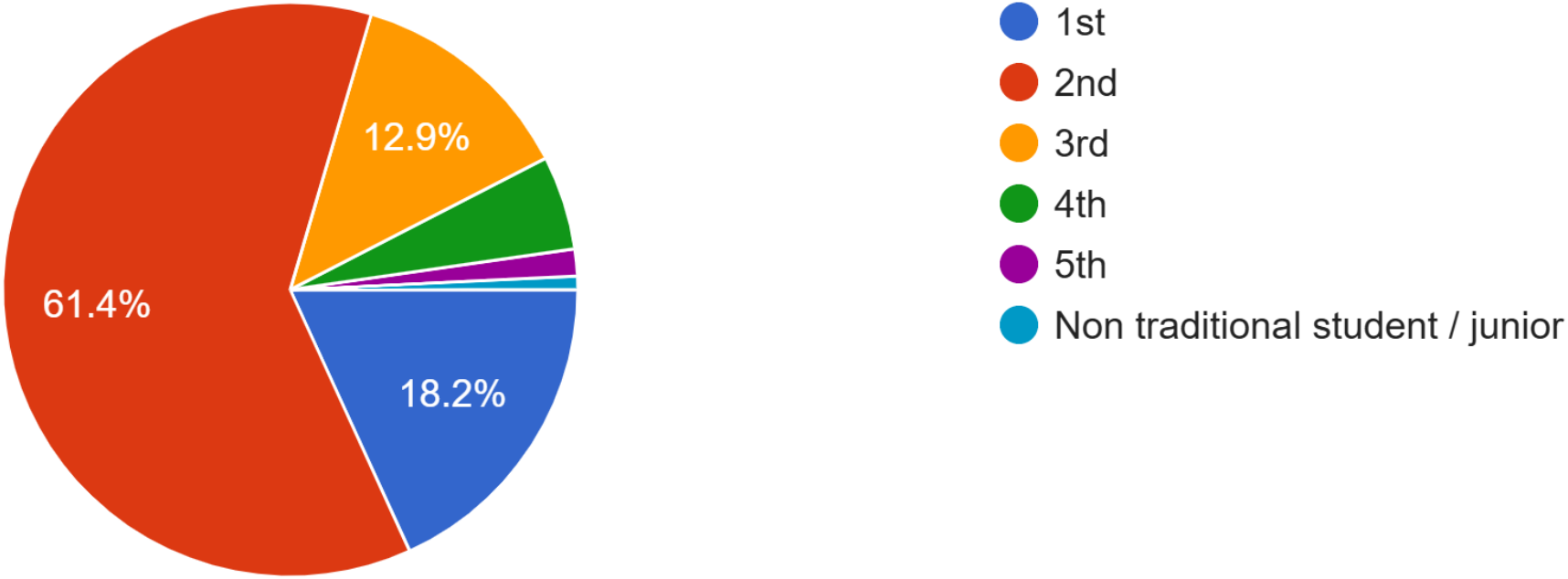
Survey #1

<https://forms.gle/5hXtCrqTJt2r7ZSy8>

132 / 202 students so far

What year are you in?

132 responses



Office Hours and Recitations Start Today!



Minesh Patel



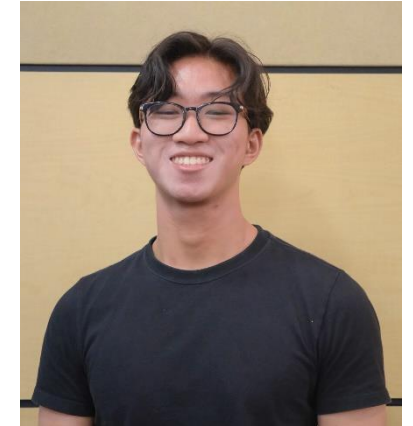
Nate Blum



Neha Jeyaram



Ramesh Balaji



Jerlin Yuen

Recitations

-

Section 5
Thursday
7:45 - 8:40 PM
ARC-105

Section 6
Tuesday
7:45 - 8:40 PM
ARC-105

Section 7
Thursday
5:55 - 6:50 PM
SEC-202

Section 8
Tuesday
5:55 - 6:50 PM
SEC-203

Office Hours

Wednesday
11:00 - 12:00
CoRE 244

Friday
3:00 - 4:00 pm
CoRE 305

Thursday
5:10 - 6:10 pm
CoRE 305

Monday
8:50 - 9:50 am
CoRE 244

Wednesday
4:30 - 5:30 pm
TIL-111L
(Livi Learning Center) **3**

Programming Assignment 1

- We have 54 submissions already (26.7% of the class)
- It's not a race, but don't procrastinate on tooling problems
 - Office hours can help with unexpected issues

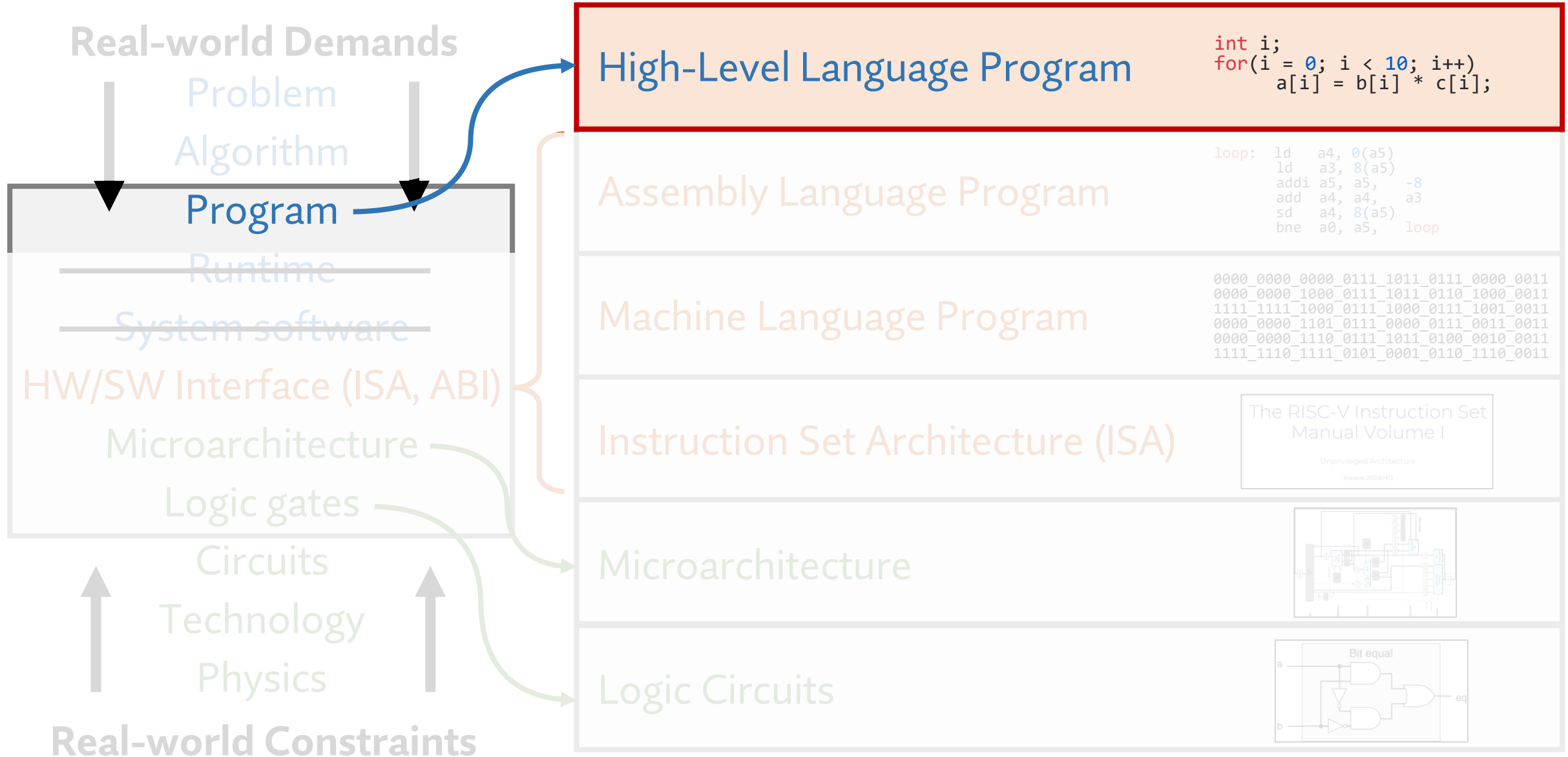
Use the Correct iLab



The screenshot shows the Rutgers iLab portal interface. The browser address bar displays 'rutgers.ilab.agilent.com'. The page header includes 'Agilent CrossLab iLab Operations Software' and a search bar. The main content area features a 'Getting started' section with a 'Welcome to iLab!' message and a list of steps for getting started. A large red 'X' is overlaid across the entire screenshot.

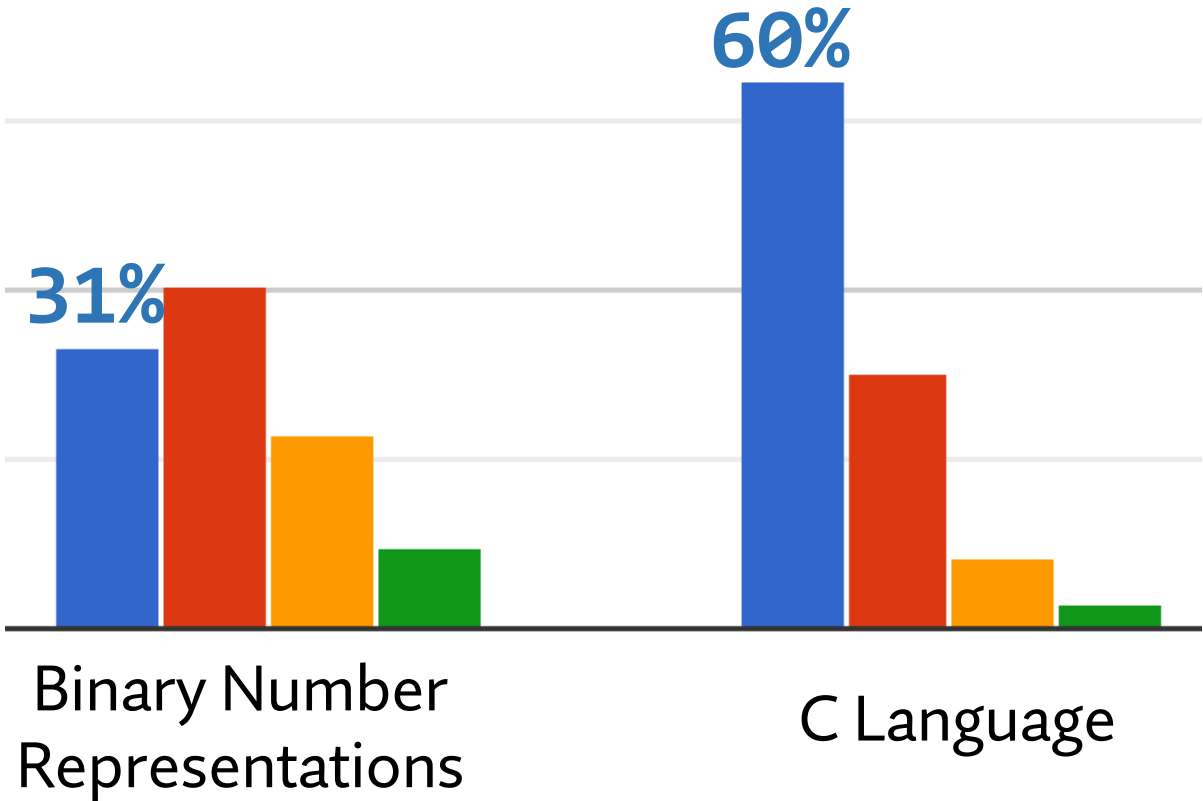
The screenshot shows the Rutgers Department of Computer Science Instructional Lab website. The browser address bar displays 'resources.cs.rutgers.edu/docs/instructional-lab/'. The page header includes the Rutgers logo and 'Technical Services and Support Department of Computer Science'. The main content area features a navigation menu with links like 'HOME', 'OUR SERVICES', 'CS HELP DESK', 'LABS & SPACES', 'DOCUMENTATION', 'JOBS', 'CONTACT US', and 'SYSTEMS STATUS'. Below the navigation menu is a banner for 'Instructional Lab' and a section for 'Announcements' with several news items. A sidebar on the right contains a list of links: 'Documentations', 'Help', 'Accounts & Passwords', 'Computing Facilities & Services', 'Computer Systems', 'Email & Calendar', 'Printing', 'File Storage', 'Software', and 'Network'.

Layers of Abstraction for CS 211



Next Several Weeks

■ Unfamiliar ■ Slightly familiar ■ Familiar ■ Very familiar



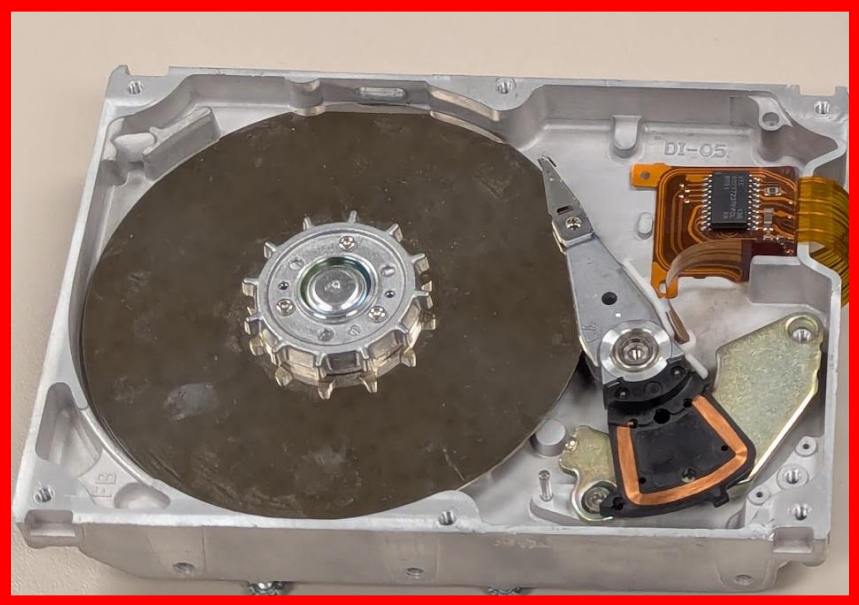
Agenda

- **A Mental Model of Computer Memory**
- C “Objects”: Not Your Java Objects
- Storing Data in Bits and Bytes

Computer Memory

Disk

(data you're not using)



big, slow, cheap

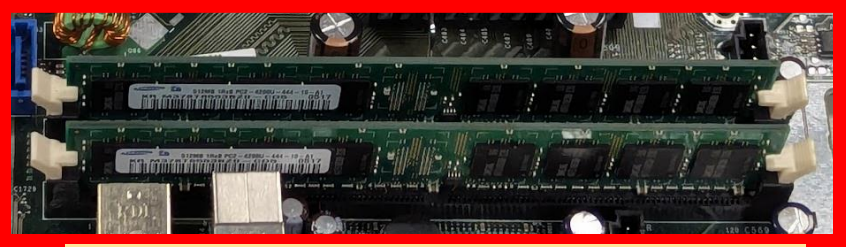
CPU

(running the code)



Memory

(data you're using)



small, fast, expensive

The C (and C++) Language Reference

cppreference.com

cppreference.com

Create account

Page Discussion

Standard revision: Diff

View View source

C

C reference

C89, C95, C99, C11, C17, C23 | Compiler support C99, C23

Language

- Basic concepts
- Keywords
- Preprocessor
- Expressions
- Declaration
- Initialization
- Functions
- Statements

Numbers

Type support

Program utilities

Variadic functions

Diagnostics library

Dynamic memory management

Strings library

Null-terminated strings:
byte - multibyte - wide

Date and time library

Localization library

Input/output library

Algorithms library

Numerics library

- Common mathematical functions
- Floating-point environment (C99)
- Pseudo-random number generation
- Complex number arithmetic (C99)
- Type-generic math (C99)
- Bit manipulation (C23)
- Checked integer arithmetic (C23)

Concurrency support library (C11)

Technical specifications

Dynamic memory extensions (dynamic memory TR)

Floating-point extensions, Part 1 (FP Ext 1 TS)

Floating-point extensions, Part 4 (FP Ext 4 TS)

C language

This is a reference of the core C language constructs.

Basic concepts

- Comments
- ASCII chart
- Character sets and encodings
- Translation phases
- Punctuation
- Identifier - Scope - Lifetime
- Lookup and Name Spaces
- Type - Arithmetic types
- Objects and Alignment
- The main function
- As-if rule

Memory model or Data races

Keywords

Preprocessor

- #if - #ifdef - #ifndef - #elif
- #elifdef - #elifndef (C23)
- #define - # - ##
- #include - #pragma
- #line - #error
- #warning (C23) - #embed (C23)

Statements

- if - switch
- for
- while - do-while
- continue - break
- goto - return

Expressions

- Value categories
- Evaluation order and sequencing
- Constants and literals
- Integer constants
- Floating constants
- Character constants
- true/false (C23)
- nullptr (C23)
- String literals
- Compound literals (C99)
- Constant expressions
- Implicit conversions
- Operators
- Member access and indirection
- Logical - Comparison
- Arithmetic - Assignment
- Increment and Decrement
- Call, Comma, Ternary
- sizeof - alignof (C11)
- Cast operators
- Operator precedence
- Generic selection (C11)

Initialization

- Scalar
- Array
- Structure/Union

Declarations

- Pointers - Arrays
- Enumerations
- Storage duration and Linkage
- const - volatile - restrict (C99)
- struct - union - Bit-fields
- alignas (C11) - typedef
- static_assert (C11)
- Atomic types (C11)
- External and tentative definitions
- Attributes (C23)

Functions

- Function declaration
- Function definition
- inline (C99)
- _Noreturn (C11) (deprecated in C23)
- Variadic arguments

Miscellaneous

- History of C
- Conformance
- Inline assembly
- Signal handling
- Analyzability (C11)

The C Memory Model

cppreference.com Create account Search

Page Discussion Standard revision: Diff View Edit History

C / C language / Basic Concepts

Memory model

Defines the semantics of computer memory storage for the purpose of the C abstract machine.

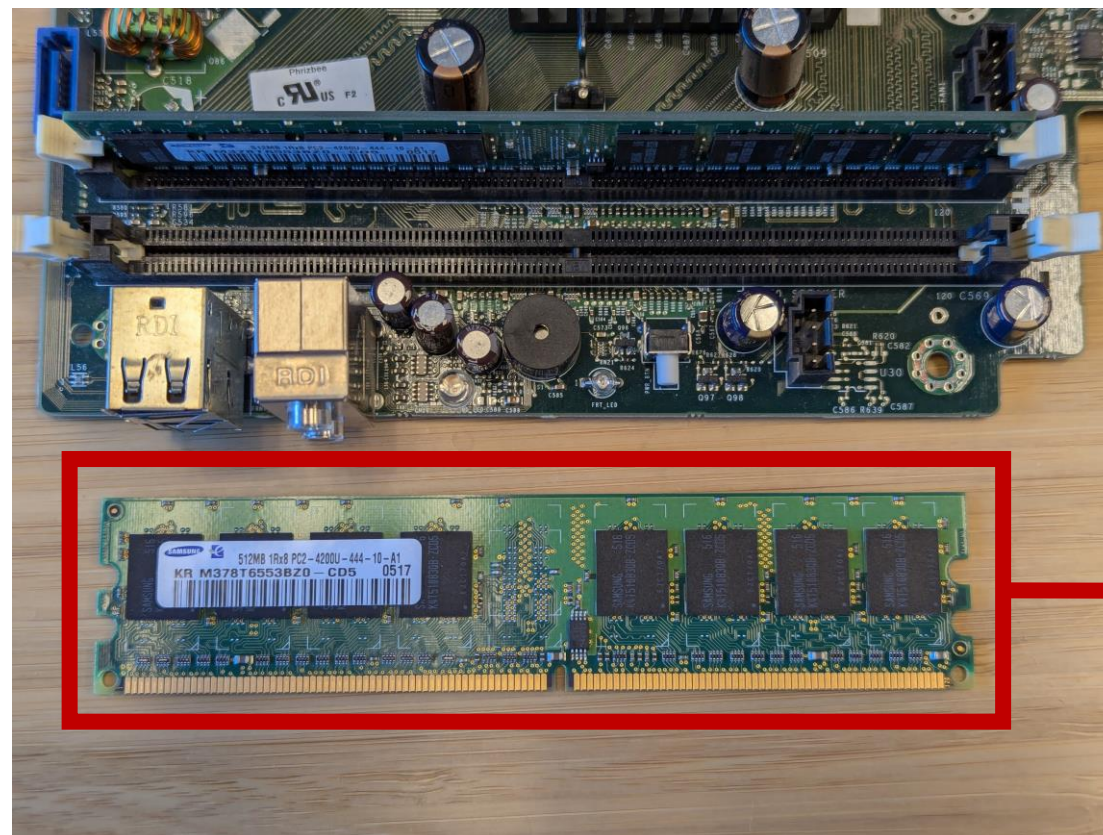
The data storage (memory) available to a C program is one or more contiguous sequences of *bytes*. Each byte in memory has a unique *address*.



- 1. Memory** is a contiguous sequence of **bytes**.
- 2. Each byte** in memory has a **unique address**.

The C Memory Model

1. **Memory** is a contiguous sequence of **bytes**.



“memory”

byte₀

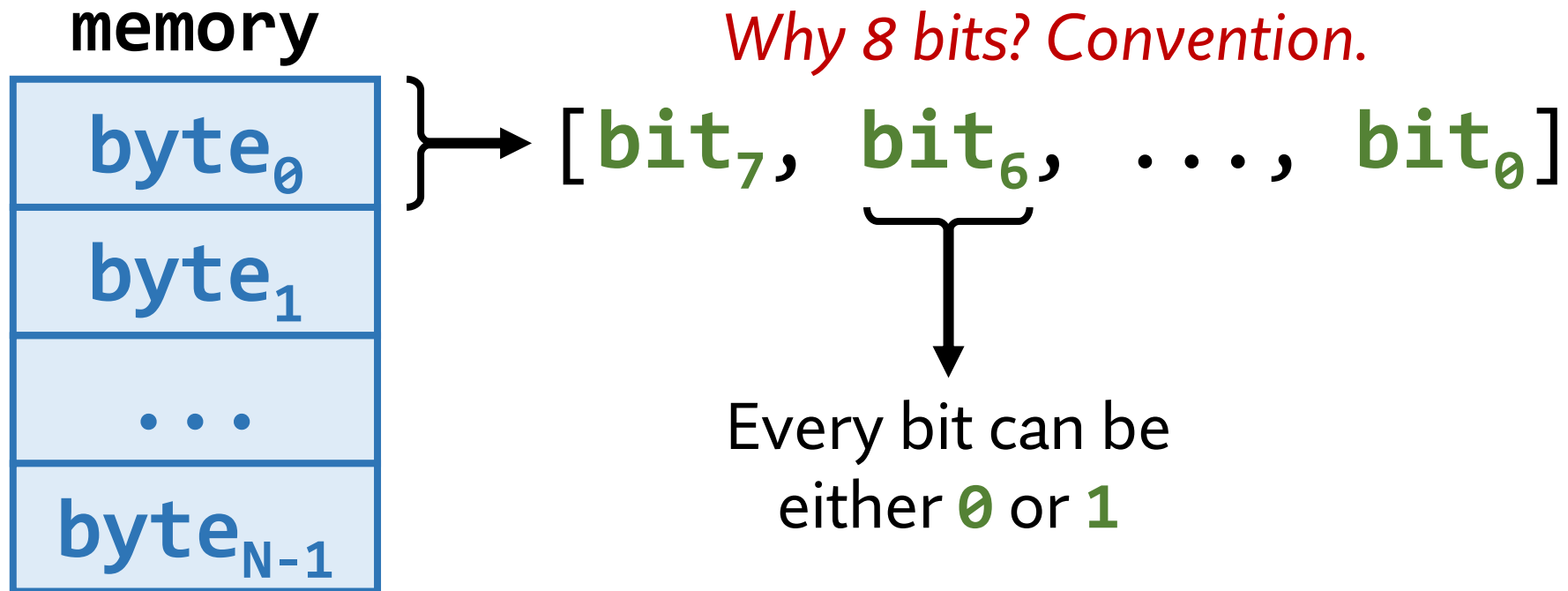
byte₁

...

byte_{N-1}

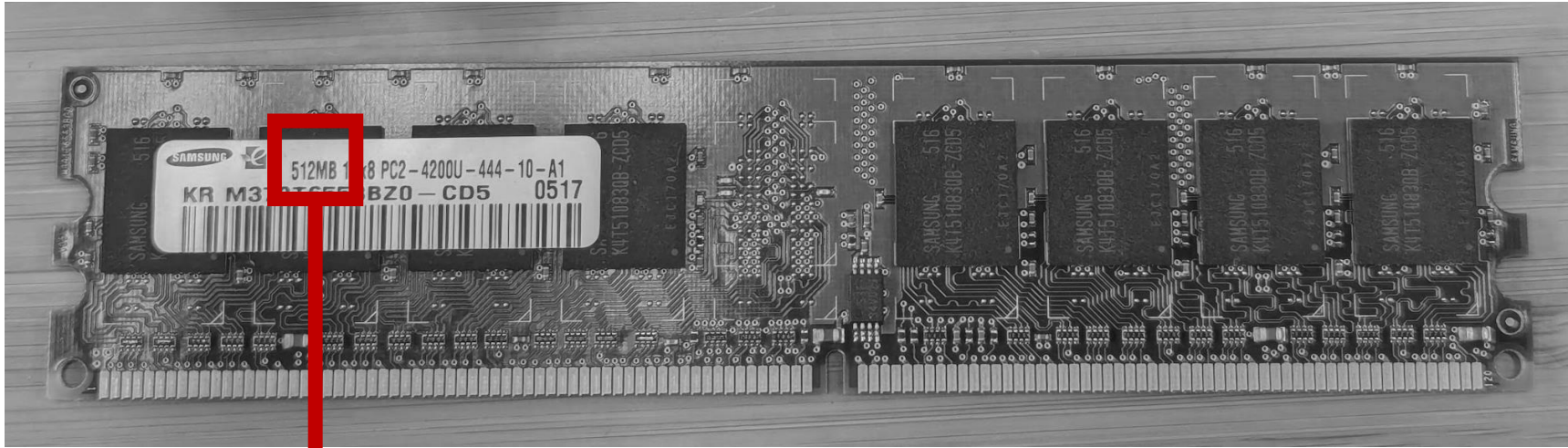
The C Memory Model

1. **Memory** is a contiguous sequence of **bytes**.



The C Memory Model

1. **Memory** is a contiguous sequence of **bytes**.

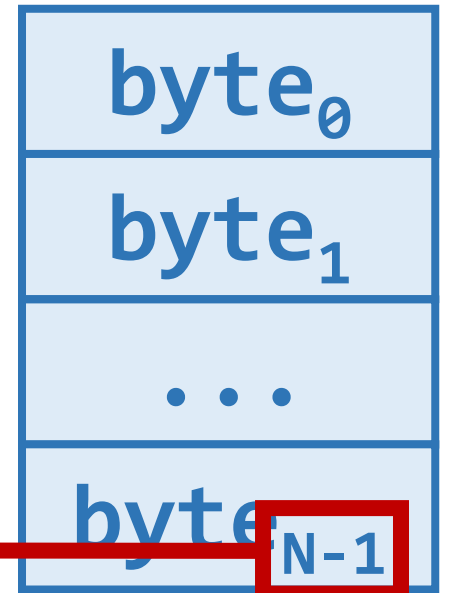


$$M = 10^6 \quad M_i = 2^{20} = 1048576$$
$$G = 10^9 \quad G_i = 2^{30}$$

512 MiB (2²⁹ B)

536,870,912 bytes = 4,294,967,296 bits

“memory”



Why Bits?

- **BIT** = “**B**inary **di**gi**T**”
 - Binary: two states (1/0, on/off, high/low, etc.)
 - Digit: denoting a number
- **Bistable systems** are simple, cheap, reliable, etc.



Position
up/down



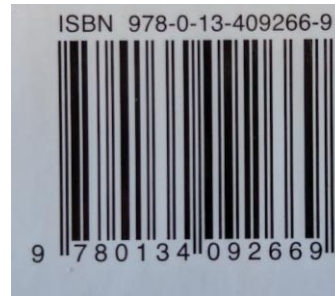
Sound
on/off



State
clipped/unclipped



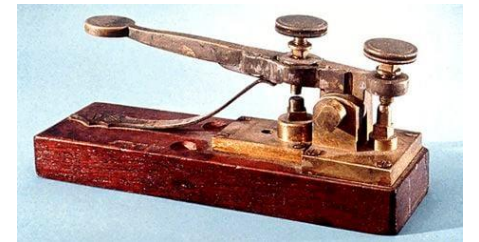
Color
red/black



Line width
narrow/wide



Pixel color
Black/white



Sound Length
Long/short tap

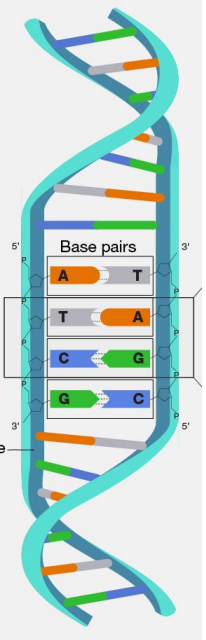
<https://www.smithsonianmag.com/arts-culture/how-the-telegraph-went-from- semaphore-to-communication-game-changer-1403433/>

Aside: {Ternary, Quaternary, ...} Systems

- They exist. They are often more complex, expensive, unreliable, etc.

4-State Systems

<https://www.genome.gov/genetics-glossary/Deoxyribonucleic-Acid-DNA>



Base Choice
a/t/c/g

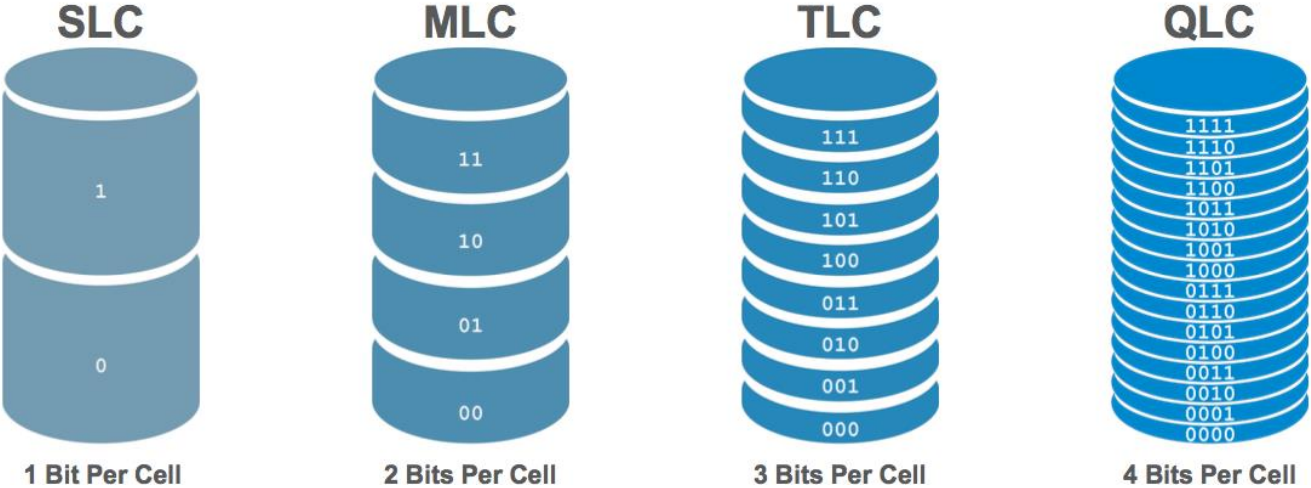


Knob setting
off/hi/med/lo

SSD (Flash Memory) Cell



<https://www.dell.com/wp-uploads/2021/07/Blog-image.png>



Voltage Level

Why Bits?

- **BIT** = “**BI**nary **digiT**”
 - Binary: two states (1/0, on/off, high/low, etc.)
 - Digit: denoting a number
- **Bistable systems** are simple, cheap, reliable, etc.



Hard Drive
Ferromagnetic



DRAM
Capacitor Voltage

https://upload.wikimedia.org/wikipedia/commons/e/e8/CD_autolev_crop_new.jpg



Compact Disc
Physical Depth



SSD
Transistor Voltage

Representing Data as Bits

- How can I store data using bits?

String: I love CS211!

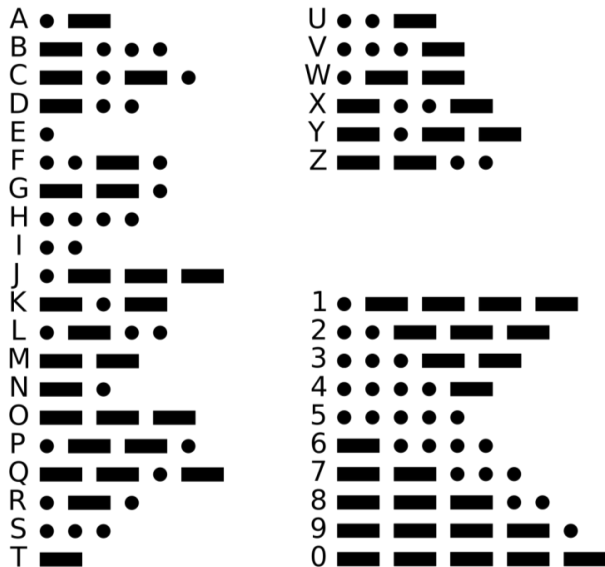
Any two-state notation will work
{1, 0} is just convention

Morse code: -.-... --- ...- -.-.- - - - - . - - - - -

Bit equivalent: 00 000000 0100 111 0001 0 000000 1010 000 00111 01111 01111 101011



Also equivalent: LL LLLLLLL LHLL HHH LLLH L LLLLLLLL HLHL LLL LLHHH LHHHH LHHHH HLHLHH



- We can store **any value** using bits if we agree on its **data representation**
 - Text values (strings)
 - Number values (integers, fractions, etc.)
 - Logical values (e.g., true, false)
 - Binary values
 - Images, video
 - Programs (executable code)
 - Programs that run other programs (OS) 😊

How Many Digits Do We Need?

- Given **N** digits, we can represent **base^N** different things

Decimal base = 10	decimal digits	unique values
	1	10: {0, 1, ..., 8, 9}
	2	100: {00, 01, 02, ..., 98, 99}
	3	1000: {000, 001, ..., 998, 999}
	N	10^N
Binary base = 2	binary digits (bits)	unique values
	1	2: {1, 0}
	2	4: {00, 01, 10, 11}
	3	8: {000, 001, 010, 011}
	N	2^N

Example Data Representation: English Text

ASCII Table

Dec	Char	Dec	Char	Dec	Char	Dec	Char
0	NUL (null)	32	SPACE	64	@	96	`
1	SOH (start of heading)	33	!	65	A	97	a
2	STX (start of text)	34	"	66	B	98	b
3	ETX (end of text)	35	#	67	C	99	c
4	EOT (end of transmission)	36	\$	68	D	100	d
5	ENQ (enquiry)	37	%	69	E	101	e
6	ACK (acknowledge)	38	&	70	F	102	f
7	BEL (bell)	39	'	71	G	103	g
8	BS (backspace)	40	(72	H	104	h
9	TAB (horizontal tab)	41)	73	I	105	i
10	LF (NL line feed, new line)	42	*	74	J	106	j
11	VT (vertical tab)	43	+	75	K	107	k
12	FF (NP form feed, new page)	44	,	76	L	108	l
13	CR (carriage return)	45	-	77	M	109	m
14	SO (shift out)	46	.	78	N	110	n
15	SI (shift in)	47	/	79	O	111	o
16	DLE (data link escape)	48	0	80	P	112	p
17	DC1 (device control 1)	49	1	81	Q	113	q
18	DC2 (device control 2)	50	2	82	R	114	r
19	DC3 (device control 3)	51	3	83	S	115	s
20	DC4 (device control 4)	52	4	84	T	116	t
21	NAK (negative acknowledge)	53	5	85	U	117	u
22	SYN (synchronous idle)	54	6	86	V	118	v
23	ETB (end of trans. block)	55	7	87	W	119	w
24	CAN (cancel)	56	8	88	X	120	x
25	EM (end of medium)	57	9	89	Y	121	y
26	SUB (substitute)	58	:	90	Z	122	z
27	ESC (escape)	59	;	91	[123	{
28	FS (file separator)	60	<	92	\	124	
29	GS (group separator)	61	=	93]	125	}
30	RS (record separator)	62	>	94	^	126	~
31	US (unit separator)	63	?	95	_	127	DEL

- How many digits do we need to represent every value?

base 10

~~128~~ $128 \leq 10^N$
 $N = 3$

base 2

$128 \leq 2^N$
 $N? = 7$

Example Data Representation: English Text

ASCII Table

Dec	Char	Dec	Char	Dec	Char	Dec	Char
0	NUL (null)	32	SPACE	64	@	96	`
1	SOH (start of heading)	33	!	65	A	97	a
2	STX (start of text)	34	"	66	B	98	b
3	ETX (end of text)	35	#	67	C	99	c
4	EOT (end of transmission)	36	\$	68	D	100	d
5	ENQ (enquiry)	37	%	69	E	101	e
6	ACK (acknowledge)	38	&	70	F	102	f
7	BEL (bell)	39	'	71	G	103	g
8	BS (backspace)	40	(72	H	104	h
9	TAB (horizontal tab)	41)	73	I	105	i
10	LF (NL line feed, new line)	42	*	74	J	106	j
11	VT (vertical tab)	43	+	75	K	107	k
12	FF (NP form feed, new page)	44	,	76	L	108	l
13	CR (carriage return)	45	-	77	M	109	m
14	SO (shift out)	46	.	78	N	110	n
15	SI (shift in)	47	/	79	O	111	o
16	DLE (data link escape)	48	0	80	P	112	p
17	DC1 (device control 1)	49	1	81	Q	113	q
18	DC2 (device control 2)	50	2	82	R	114	r
19	DC3 (device control 3)	51	3	83	S	115	s
20	DC4 (device control 4)	52	4	84	T	116	t
21	NAK (negative acknowledge)	53	5	85	U	117	u
22	SYN (synchronous idle)	54	6	86	V	118	v
23	ETB (end of trans. block)	55	7	87	W	119	w
24	CAN (cancel)	56	8	88	X	120	x
25	EM (end of medium)	57	9	89	Y	121	y
26	SUB (substitute)	58	:	90	Z	122	z
27	ESC (escape)	59	;	91	[123	{
28	FS (file separator)	60	<	92	\	124	
29	GS (group separator)	61	=	93]	125	}
30	RS (record separator)	62	>	94	^	126	~
31	US (unit separator)	63	?	95	_	127	DEL

Char

Representation

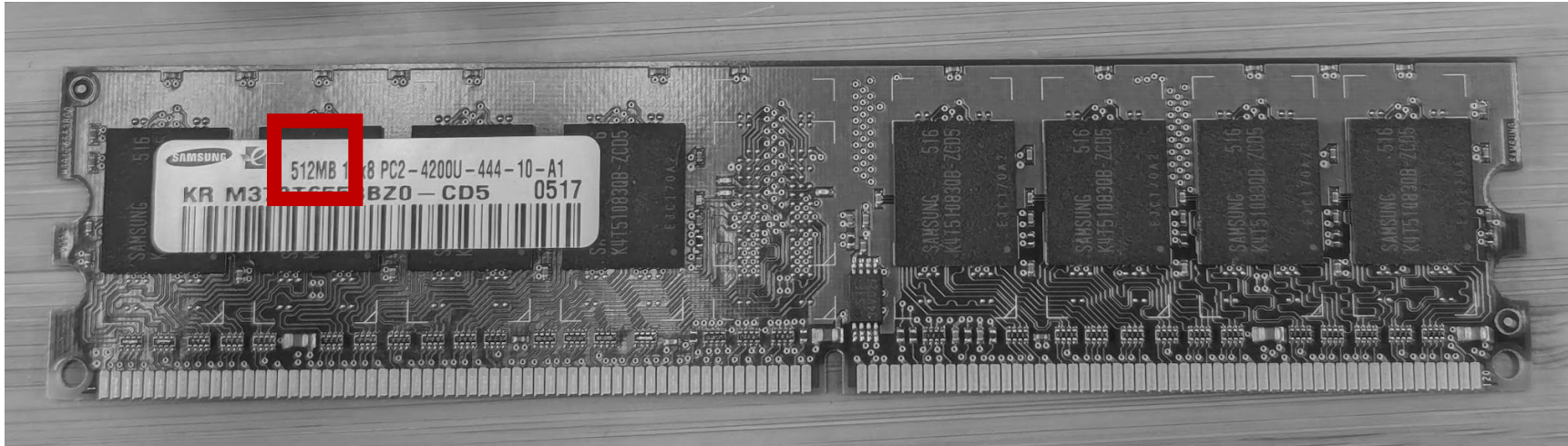
Decimal

Binary

'I'	73	1001001
' '	32	0100000
'l'	108	1101100
'o'	111	1101111
'v'	118	1110110
'e'	101	1100101
' '	32	0100000
'C'	67	1000011
'S'	83	1010011
'2'	50	0110010
'1'	49	0110001
'1'	49	0110001
'!'	33	0100001

Back to The C Memory Model

1. **Memory** is a contiguous sequence of **bytes**.



- 512 MiB = 2^{29} bytes
- Every byte represents 2^8 unique values
- ...what should we do with all these values?

“memory”

byte₀

byte₁

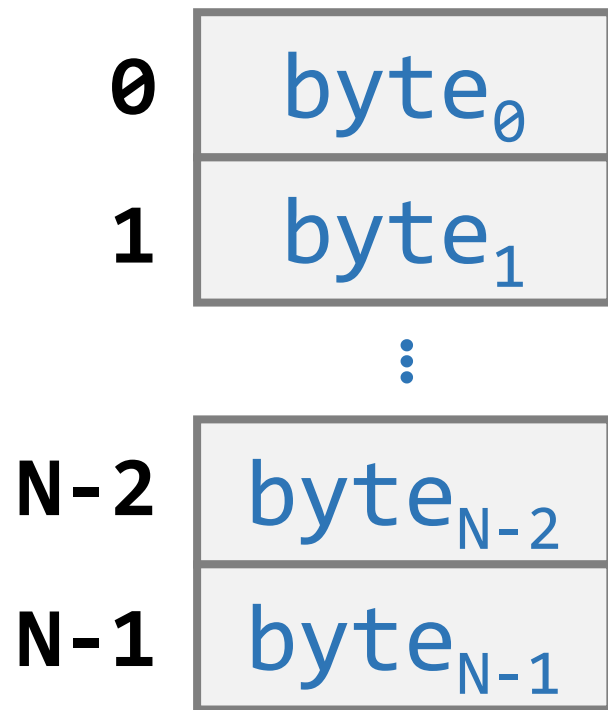
...

byte_{N-1}

The C Memory Model

1. **Memory** is a contiguous sequence of **bytes**.
2. **Each byte** in memory has a **unique address**.

addr memory



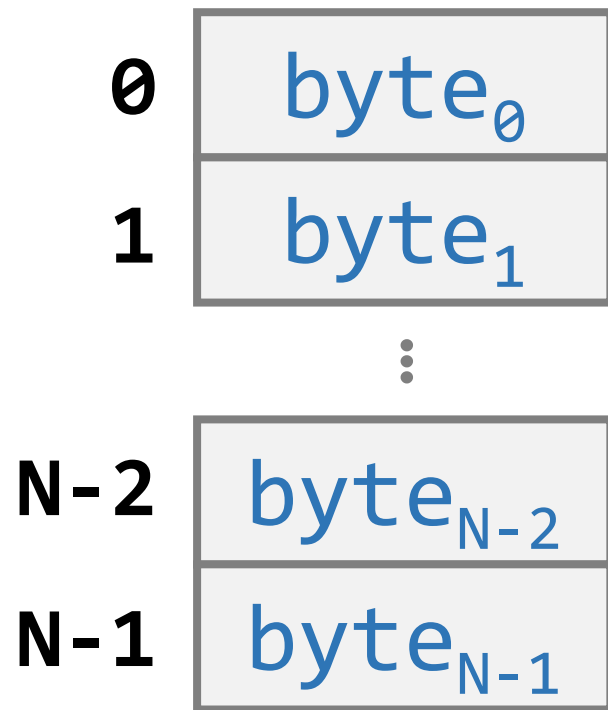
“Memory location 0” **contains** “byte 0”

The memory is “**byte addressable**”

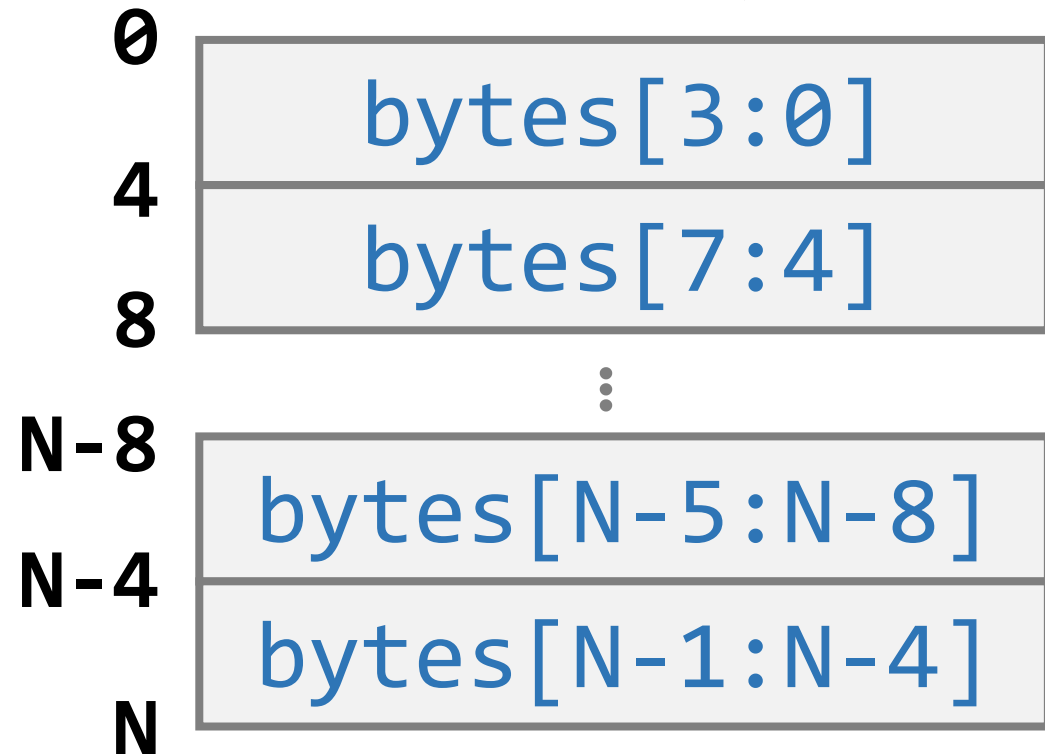
The C Memory Model

- 1. Memory** is a contiguous sequence of **bytes**.
- 2. Each byte** in memory has a **unique address**.

addr memory



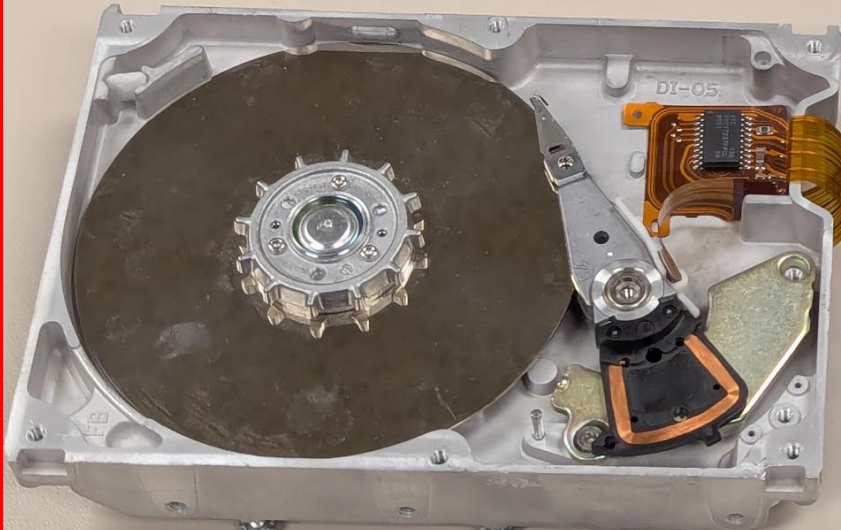
addr memory



Computer Memory

Disk

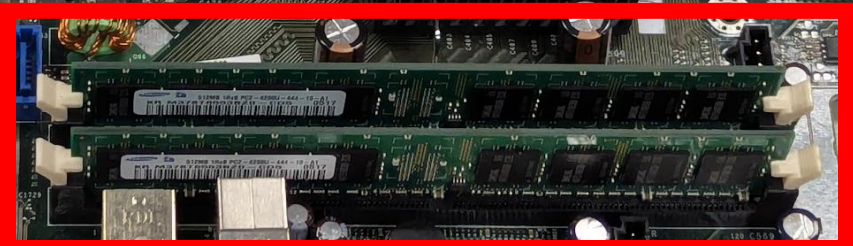
(data you're not using)



big, slow, cheap

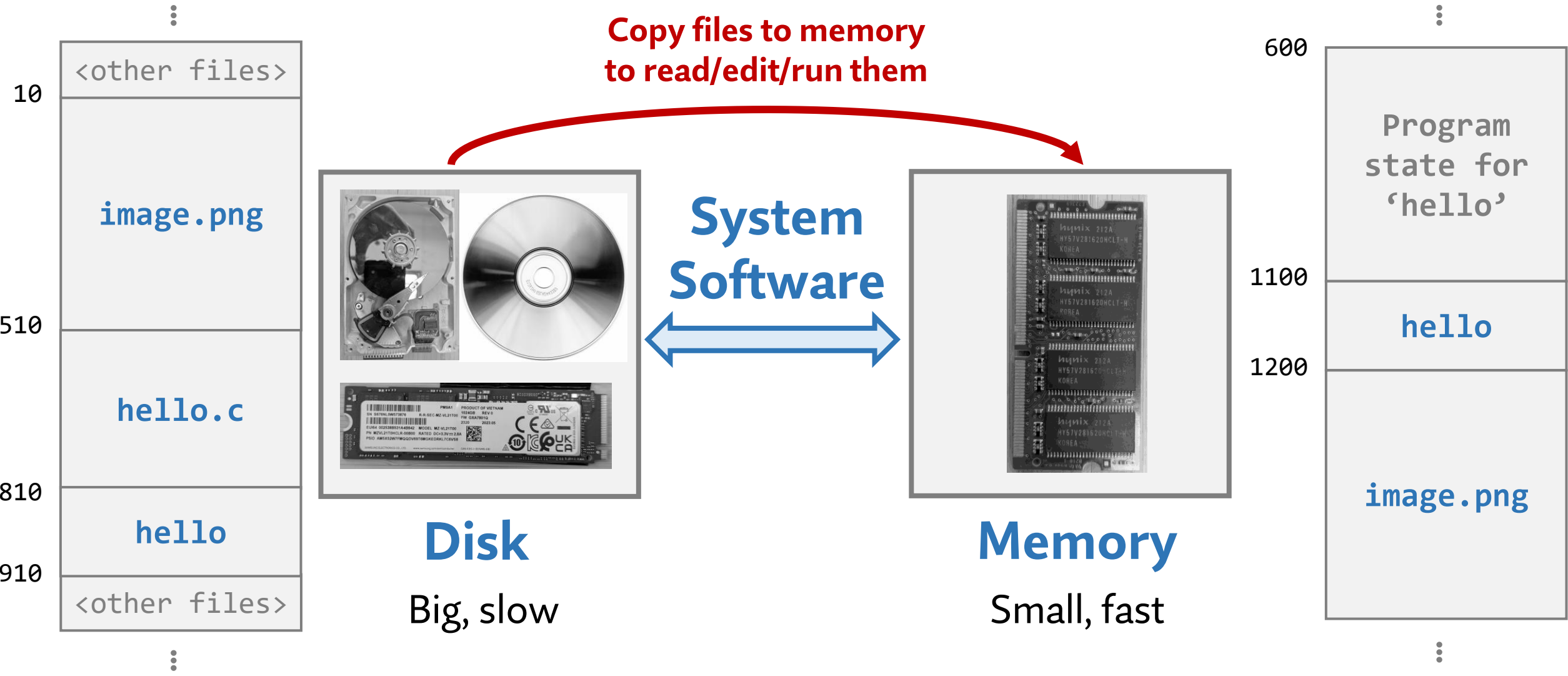
Memory

(data you're using)



small, fast, expensive

Memory Locations: System-Level View

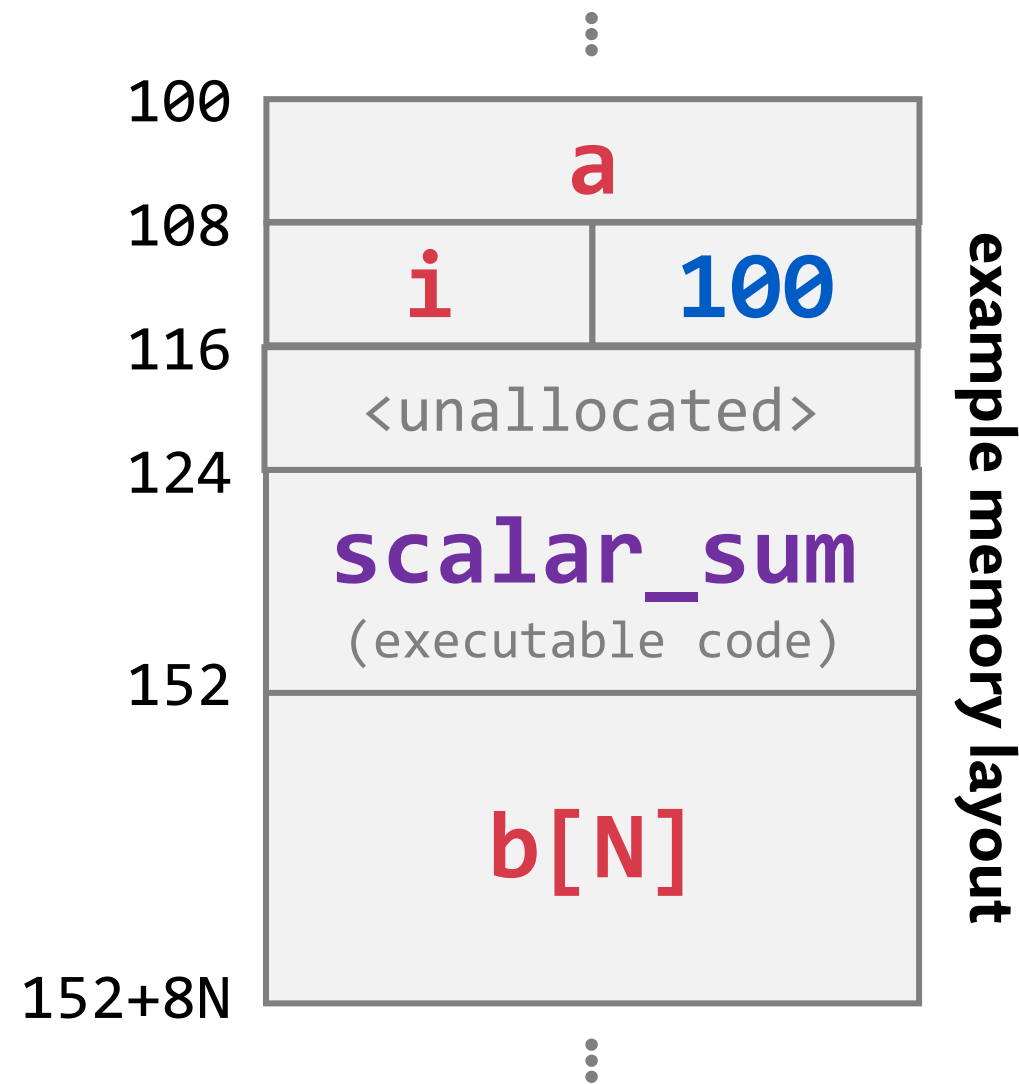


Memory Locations: Inside a C Program's Memory

```
uint32_t scalar_sum(uint64_t a, uint64_t b[N])
{
    uint32_t i;
    for(i = 0; i < 100; i++)
        b[i] += a;
}
```

“&” = “address of” operator

- $\&a = 100$
- $\&i = 108 / 112$
- $\&t = 112 / 108$
- $\&b = 152$
- $\&scalar_sum = 124$



Agenda

- A Mental Model of Computer Memory
- **C “Objects”: Not Your Java Objects**
- Storing Data in Bits and Bytes

The C (and C++) Language Reference

cppreference.com

Create account

Page Discussion

Standard revision: Diff

View View source

C

C reference

C89, C95, C99, C11, C17, C23 | Compiler support C99, C23

Language

- Basic concepts
- Keywords
- Preprocessor
- Expressions
- Declaration
- Initialization
- Functions
- Statements

Numbers

Type support

Technical specifications

- Dynamic memory extensions (dynamic memory TR)
- Floating-point extensions, Part 1 (FP Ext 1 TS)
- Floating-point extensions, Part 4 (FP Ext 4 TS)

Program utilities

Variadic functions

Diagnostics library

Dynamic memory management

Strings library

Null-terminated strings:
byte - multibyte - wide

Date and time library

Localization library

Input/output library

Algorithms library

Numerics library

- Common mathematical functions
- Floating-point environment (C99)
- Pseudo-random number generation
- Complex number arithmetic (C99)
- Type-generic math (C99)
- Bit manipulation (C23)
- Checked integer arithmetic (C23)

Concurrency support library (C11)

C language

This is a reference of the core C language constructs.

Basic concepts

- Comments
- ASCII chart
- Character sets and encodings
- Translation phases
- Punctuation
- Identifier - Scope - Lifetime
- Lookup and Name Spaces
- Arithmetic types
- Alignment
- Function
- As-if rule
- Undefined behavior
- Memory model and Data races

Keywords

Preprocessor

- #if - #ifdef - #ifndef - #elif
- #elifdef - #elifndef (C23)
- #define - # - ##
- #include - #pragma
- #line - #error
- #warning (C23) - #embed (C23)

Statements

- if - switch
- for
- while - do-while
- continue - break
- goto - return

Expressions

- Value categories
- Evaluation order and sequencing
- Constants and literals
- Integer constants
- Floating constants
- Character constants
- true/false (C23)
- nullptr (C23)
- String literals
- Compound literals (C99)
- Constant expressions
- Implicit conversions
- Operators
- Member access and indirection
- Logical - Comparison
- Arithmetic - Assignment
- Increment and Decrement
- Call, Comma, Ternary
- sizeof - alignof (C11)
- Cast operators
- Operator precedence
- Generic selection (C11)

Initialization

- Scalar
- Array
- Structure/Union

Declarations

- Pointers - Arrays
- Enumerations
- Storage duration and Linkage
- const - volatile - restrict (C99)
- struct - union - Bit-fields
- alignas (C11) - typedef
- static_assert (C11)
- Atomic types (C11)
- External and tentative definitions
- Attributes (C23)

Functions

- Function declaration
- Function definition
- inline (C99)
- _Noreturn (C11) (deprecated in C23)
- Variadic arguments

Miscellaneous

- History of C
- Conformance
- Inline assembly
- Signal handling
- Analyzability (C11)

Objects in C

cppreference.com Create account

Page Discussion Standard revision: Diff

C / C language / Basic Concepts

Objects and alignment

C programs create, destroy, access, and manipulate objects.

An object in C is a region of data storage in the execution environment, the contents of which can represent *values* (a value is the meaning of the contents of an object, when interpreted as having a specific type).

Every object has

- size (can be determined with `sizeof`)
- alignment requirement (can be determined by `_Alignof` (since C11))
- storage duration (automatic, static, allocated, thread-local)
- lifetime (equal to storage duration or temporary)
- effective type (see below)
- value (which may be indeterminate)
- optionally, an identifier that denotes this object.

Objects are created by declarations, allocation functions, string literals, compound literals, and by non-lvalue expressions that return structures or unions with array members.

Objects in C

regions of memory

An object in C is a region of *data storage* in the execution environment the contents of which can represent *values*

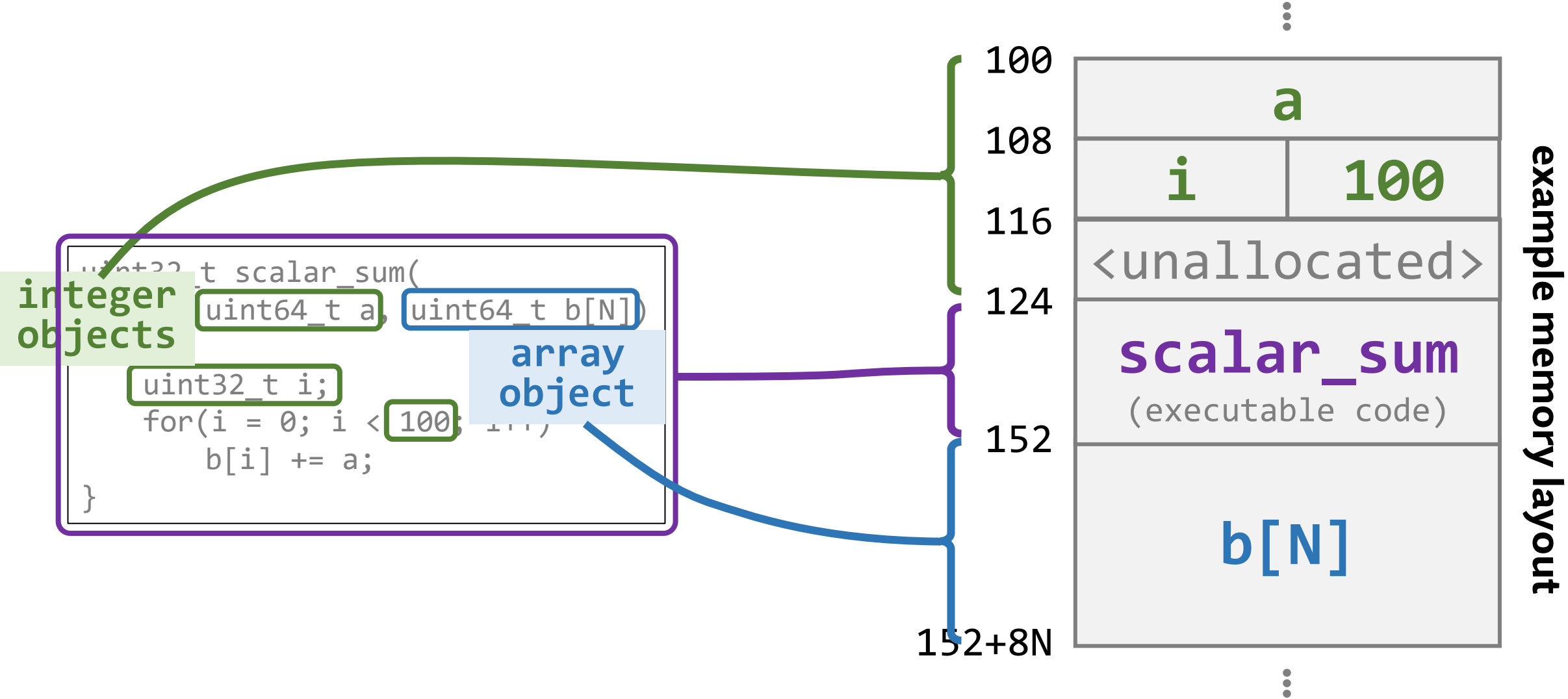
C objects are **completely unrelated** to:

- Object-Oriented Programming (OOP)
- Python/Java/C++ classes

Things we would like to store

- Numbers
- Strings (characters)
- Binary data (images, video)
- Executable code
- ...

C Objects: Code Perspective



example memory layout

C Objects

Every object has

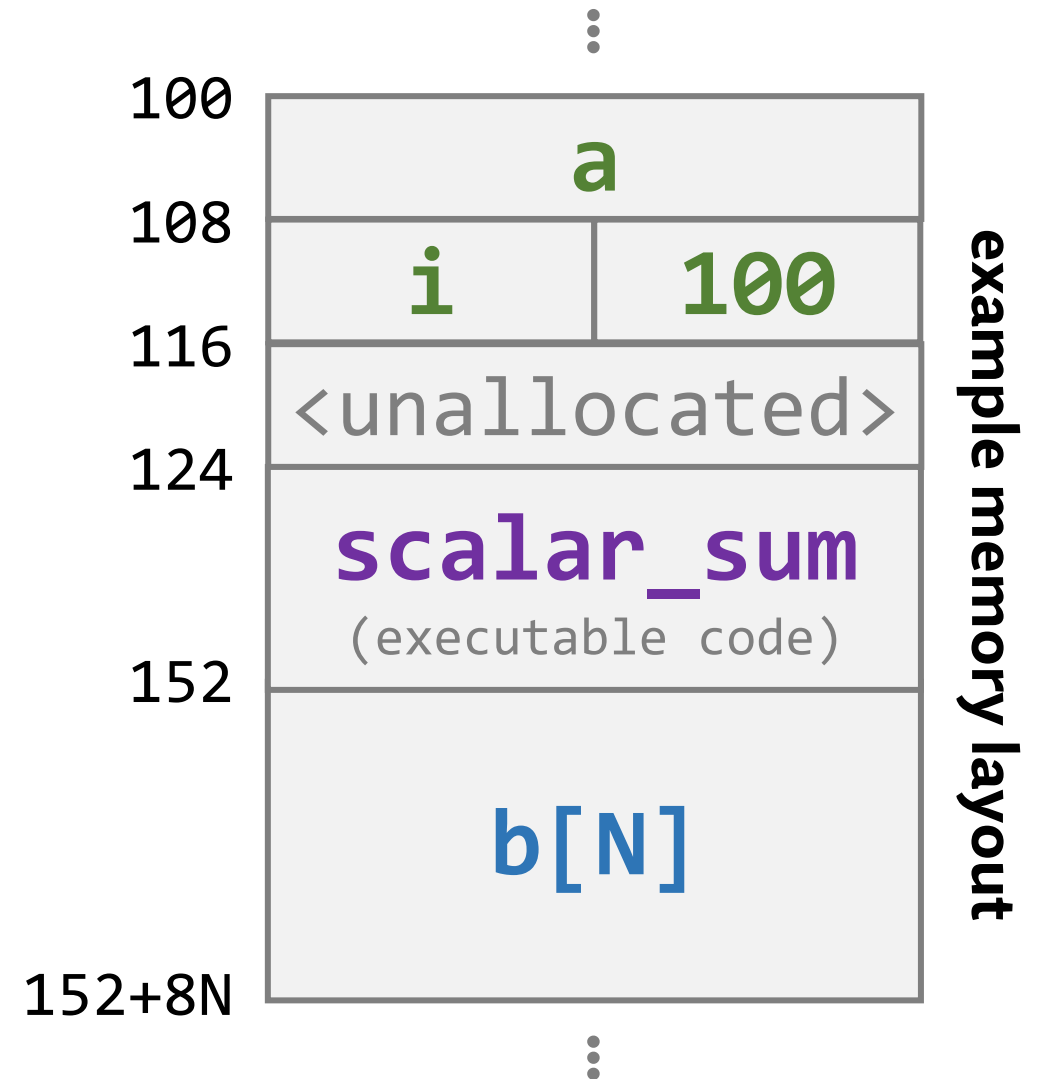
- size (can be determined with `sizeof`)
- alignment requirement (can be determined by `_Alignof` (since C11))
- storage duration (automatic, static, allocated, thread-local)
- lifetime (equal to storage duration or temporary)
- effective type (see below)
- value (which may be indeterminate)
- optionally, an `identifier` that denotes this object.

“&” = “address of” operator

- $\&a = 100$

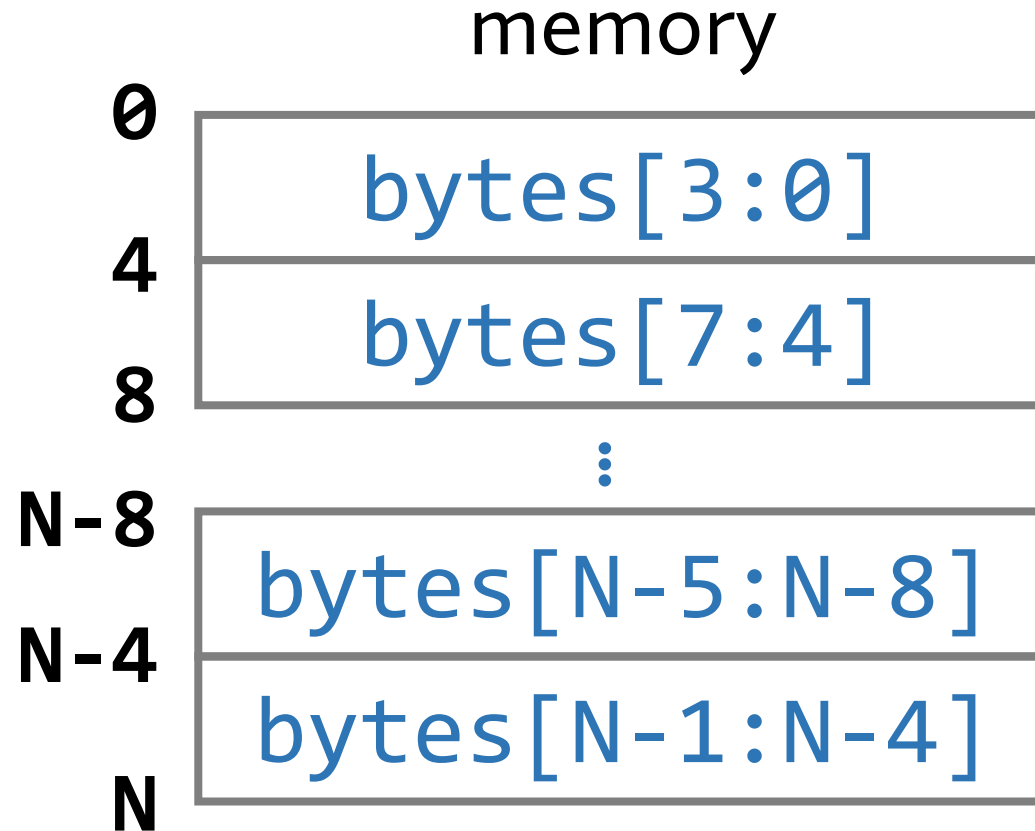
“`sizeof()`” operator

- $\text{sizeof}(a) = 8$ (bytes)
- $\text{sizeof}(b) = 8N$



Recap: The C Memory Model

1. **Memory** is a contiguous sequences of **bytes**.
2. **Each byte** in memory has a **unique address**.

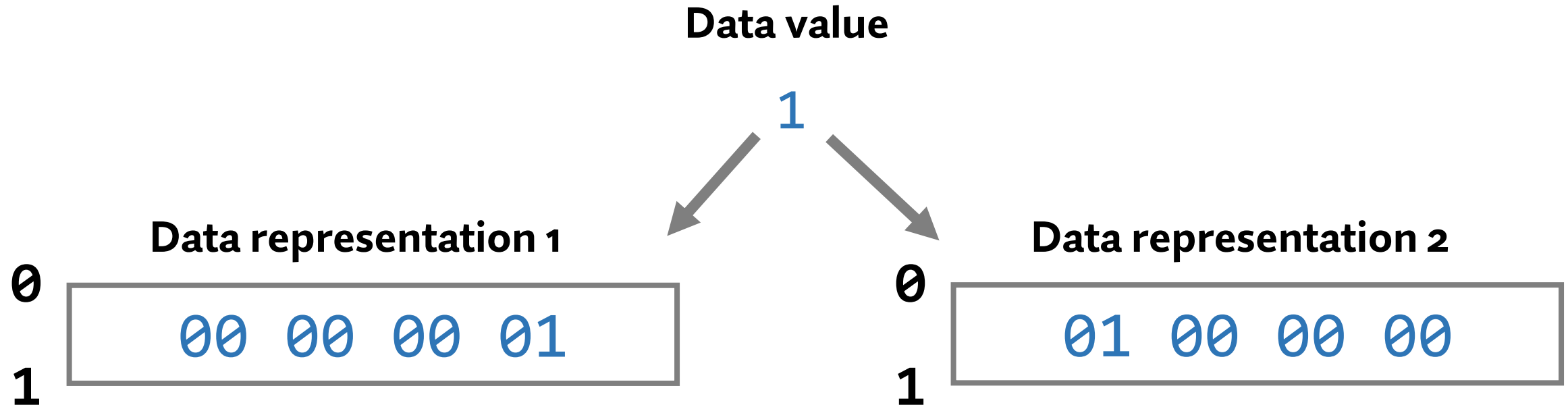


Agenda

- A Mental Model of Computer Memory
- C “Objects”: Not Your Java Objects
- **Storing Data in Bits and Bytes**

Representing Values in Objects

- **Data values:** things we would like to keep in an object
- **Data representation:** how an object represents the value

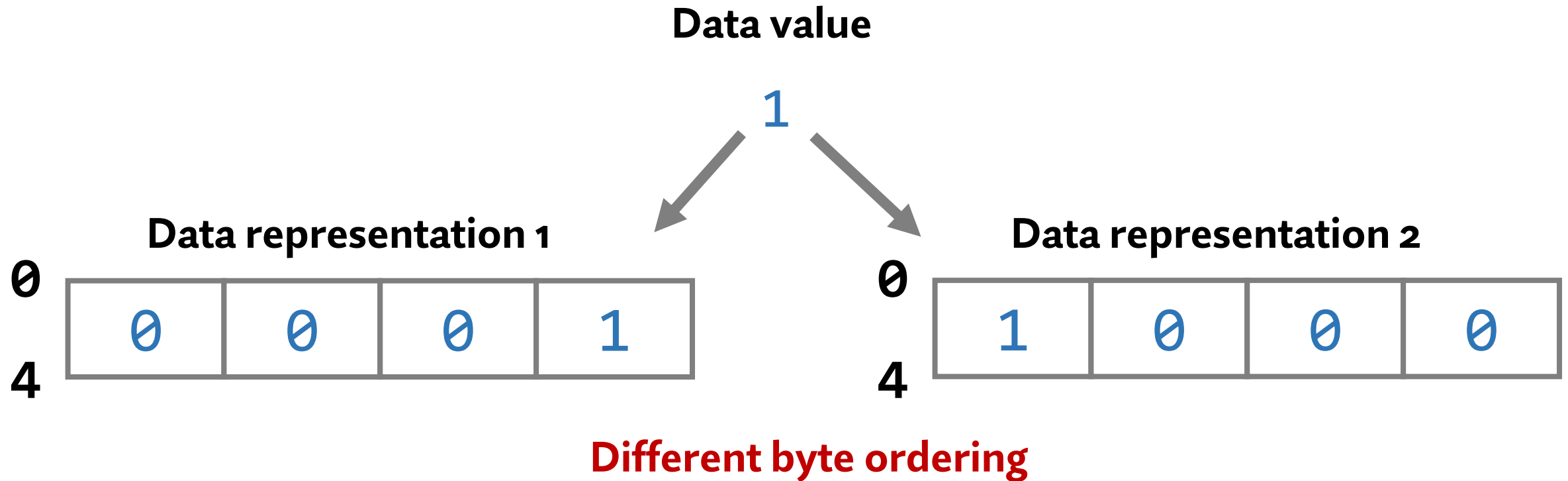


Different bit representations of '1'

- Only works if **we all agree** on the representation

Representing Values in Objects

- **Data values:** things we would like to keep in an object
- **Data representation:** how an object represents the value



- Only works if **we all agree** on the representation

Value Representations

- **Today:** Number Systems, Value Representations, Characters
- **Today + 1:** Integers (e.g., 1)
- **Today + 2:** Floating point (e.g., 6.022×10^{23})
- **Today + 3:** Derived C Objects (arrays, structs, etc.)

Codes (or “Interpretations”)

- Any mapping from **value** to **representation**
- Simplest form: a lookup table (i.e., an *enumeration* or *codebook*)

Morse Code “Codebook”

A	• —	U	• • —
B	— • • •	V	• • • —
C	— • — •	W	• — —
D	— • •	X	— • • —
E	•	Y	— • — —
F	• • — •	Z	— — • •
G	— — •		
H	• • • •		
I	• •		
J	• — — —		
K	— • —	1	• — — — —
L	• — • •	2	• • — — —
M	— —	3	• • • — —
N	— •	4	• • • • —
O	— — —	5	• • • • •
P	• — — •	6	— • • • •
Q	— — • —	7	— — • • •
R	• — •	8	— — — • •
S	• • •	9	— — — — •
T	—	0	— — — — —

https://en.wikipedia.org/wiki/Morse_code

Weighted Codes

- Instead of a lookup table, we use a mathematical formula

Value	Representation	Code
11,525 sec.	3 h, 12 m, 5 s	$v = \{h, m, s\} * \{3600, 60, 1\}$
318 inches	8 yd, 2 ft, 6 in	$v = \{y, f, i\} * \{36, 12, 1\}$
8,992 days	24 yr, 7 mo, 3 wk, 1 dy	$v = \langle \text{a bit complicated} \rangle$

Positional Codes (for Numbers)

- Weights are **implied by digit position**

Value	Representation	Code
$(1699)_{10}$	$(1 * 10^3) + (6 * 10^2) + (9 * 10^1) + (9 * 10^0)$	$\sum_i d^i * 10^i$
Base 10 digits {0,1,...,9}	Most significant digit	Least significant digit

- We can generalize positional codes to **other bases**

Value	Representation	Code
$(1001)_2$	$(1 * 2^3) + (0 * 2^2) + (0 * 2^1) + (1 * 2^0)$	$\sum_i d^i * 2^i$
Base 2 digits {0,1}		

Positional Codes in Different Bases

- Base 2 digits: {0 1}
- Base 4 digits: {0 1 2 3}
- Base 8 digits: {0 1 2 3 4 5 6 7}
- Base 10 digits: {0 1 2 3 4 5 6 7 8 9}
- Base 16 digits: {0 1 2 3 4 5 6 7 8 9 a b c d e f}
- Base 64 digits: {ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/-}

$$(63)_2 = \text{---}$$

$$(63)_4 = \text{---}$$

$$(63)_8 = 6 \cdot 8^1 + 3 \cdot 8^0 = (57)_{10}$$

$$(63)_{10} = 6 \cdot 10^1 + 3 \cdot 10^0 = (63)_{10}$$

$$(63)_{16} = 6 \cdot 16^1 + 3 \cdot 16^0 = (99)_{10}$$

$$(63)_{64} = 58 \cdot 64^1 + 55 = \dots$$

$$(f8)_2 = \text{---}$$

$$(f8)_4 = \text{---}$$

$$(f8)_8 = \text{---}$$

$$(f8)_{10} = \text{---}$$

$$(f8)_{16} = 15 \cdot 16^1 + 8 \cdot 16^0 = 248$$

$$(f8)_{64} = \dots$$

$(58)_{10}$

Positional Codes in Different Bases

- Base 2 digits: {0 1}
- Base 4 digits: {0 1 2 3}
- Base 8 digits: {0 1 2 3 4 5 6 7}
- Base 10 digits: {0 1 2 3 4 5 6 7 8 9}
- Base 16 digits: {0 1 2 3 4 5 6 7 8 9 a b c d e f}
- Base 64 digits: {ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/}

$$(63)_2 = \text{<invalid>}$$

$$(f8)_2 = \text{<invalid>}$$

$$(63)_4 = \text{<invalid>}$$

$$(f8)_4 = \text{<invalid>}$$

$$(63)_8 = (6 * 8^1) + (3 * 8^0) = 51$$

$$(f8)_8 = \text{<invalid>}$$

$$(63)_{10} = (6 * 10^1) + (3 * 10^0) = 63$$

$$(f8)_{10} = \text{<invalid>}$$

$$(63)_{16} = (6 * 16^1) + (3 * 16^0) = 99$$

$$(f8)_{16} = (f * 16^1) + (8 * 16^0) = 248$$

$$(63)_{64} = (6 * 64^1) + (3 * 64^0) = 387$$

$$(f8)_{64} = (f * 64^1) + (3 * 64^0) = 2039$$

Special Notation for Powers of 2

Base	Nickname	Digit Name	Notation Example
2	binary	bit	0b 1010010101
4	quaternary	-	(1230123) ₄
8	octal	octet	0o 134061304
10	decimal	digit	(907315904) ₁₀
16	hexadecimal (hex)	nibble	0x fe3b91ad31

Base Conversions

- $(429)_{10}$ to base 16:

$$(429)_{10} = (d_1 * 16^1) + (d_0 * 16^0)$$

- **Hint:** it's just division!

$$\begin{array}{r} 26 \\ 16 \overline{)429} \\ \underline{32} \\ 109 \\ \underline{96} \\ 13 \end{array}$$
$$\begin{array}{r} 1 \\ 16 \overline{)26} \\ \underline{16} \\ 10 \end{array}$$

$$(429)_{10} = (26)_{10} \cdot 16^1 + (13)_{10} \cdot 16^0$$

$$= (17)_{10} \cdot 16^2 + (10)_{10} \cdot 16^1 + (13)_{10} \cdot 16^0$$

$$= (1ad)_{16} = \underline{0x1ad}$$

Base Conversions Between Powers of 2

$(00000001001000110100010101100111)_2$

$(00\ 00\ 00\ 01\ 00\ 10\ 00\ 11\ 01\ 00\ 01\ 01\ 01\ 10\ 01\ 11)_2$

$(0\ 0\ 0\ 1\ 0\ 2\ 0\ 3\ 1\ 0\ 1\ 1\ 1\ 2\ 1\ 3)_4$

$(00\ 000\ 001\ 001\ 000\ 110\ 100\ 010\ 101\ 100\ 111)_2$

$(0\ 0\ 1\ 1\ 0\ 6\ 4\ 2\ 5\ 4\ 7)_8$

$(0000\ 0001\ 0010\ 0011\ 0100\ 0101\ 0110\ 0111)_2$

$(0\ 1\ 2\ 3\ 4\ 5\ 6\ 7)_{16}$

CS 211: Intro to Computer Architecture

2.1: Memory, C Objects, and Data Representation

Minesh Patel

Spring 2025 – Tuesday 28 January