CS 211: Intro to Computer Architecture

2.1: Memory, C Objects, and Data Representation

Minesh Patel

Spring 2025 – Tuesday 28 January

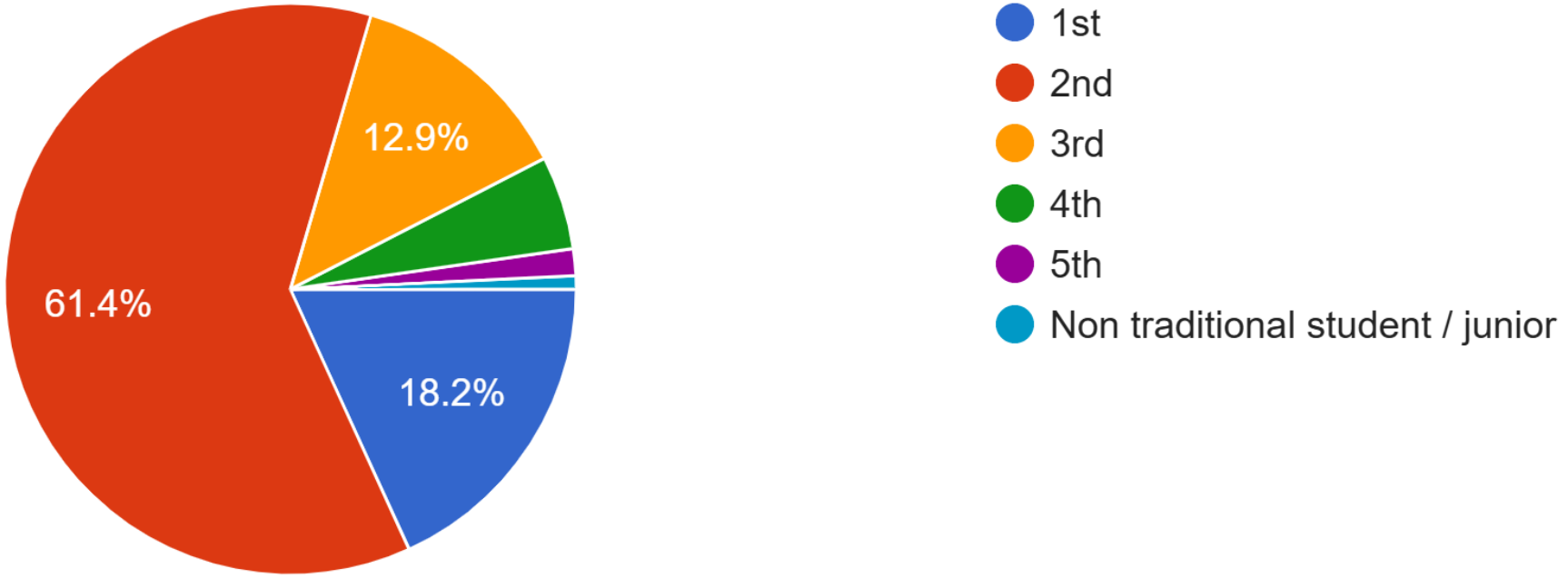
Survey #1

https://forms.gle/5hXtCrqTJt2r7ZSy8

132 / 202 students so far

What year are you in?

132 responses



Office Hours and Recitations Start Today!

Minesh Patel

Recitations: -

Office Hours:
Wednesday
11:00 - 12:00
CoRE 244

Nate Blum

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

Office Hours:
Friday
3:00 - 4:00 pm
CoRE 305

Neha Jeyaram

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

Office Hours:
Thursday
5:10 - 6:10 pm
CoRE 305

Ramesh Balaji

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

Office Hours:
Monday
8:50 - 9:50 am
CoRE 244

Jerlin Yuen

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

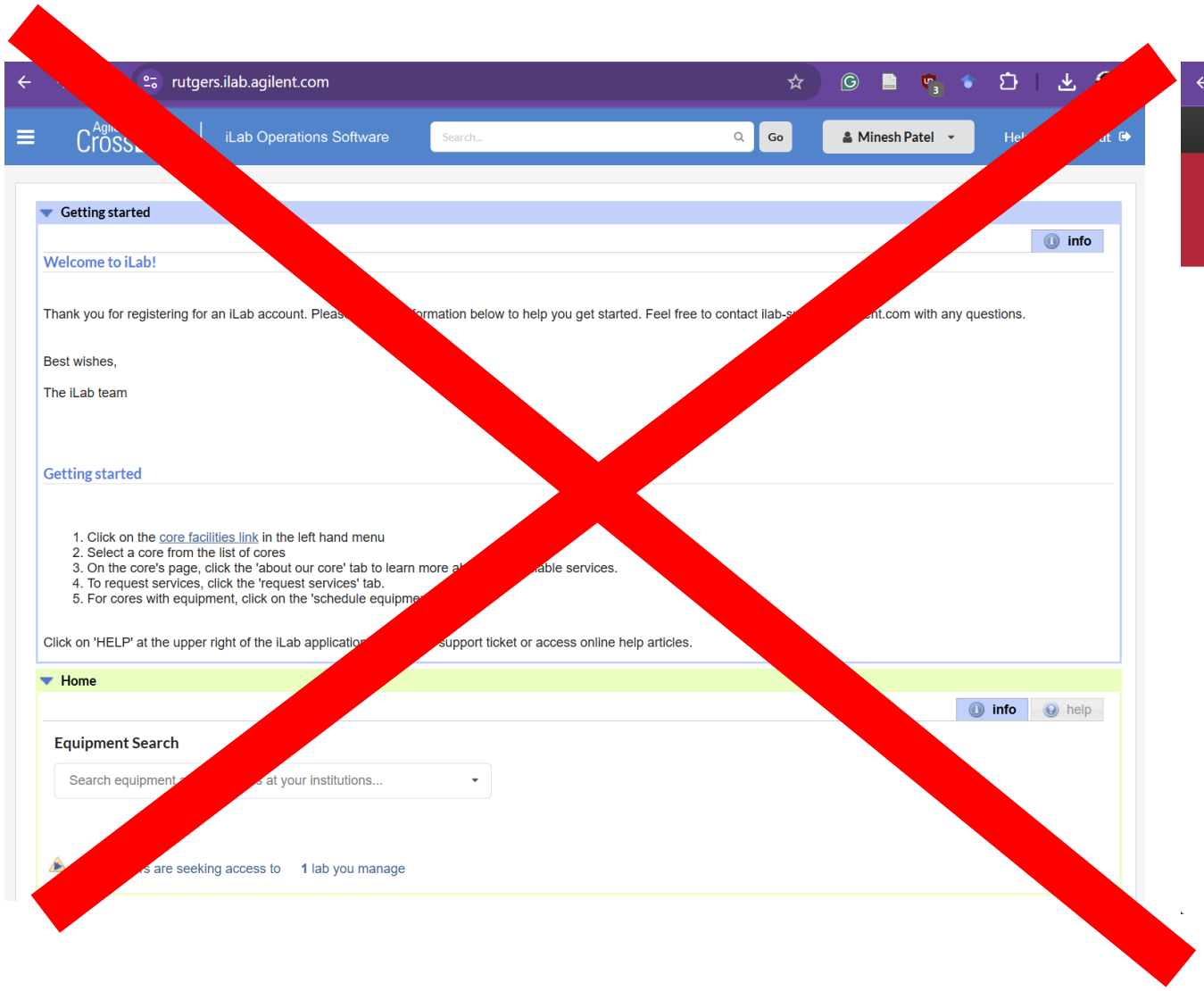
Office Hours:
Wednesday
4:30 - 5:30 pm
TIL-111L
(Livi Learning Center)

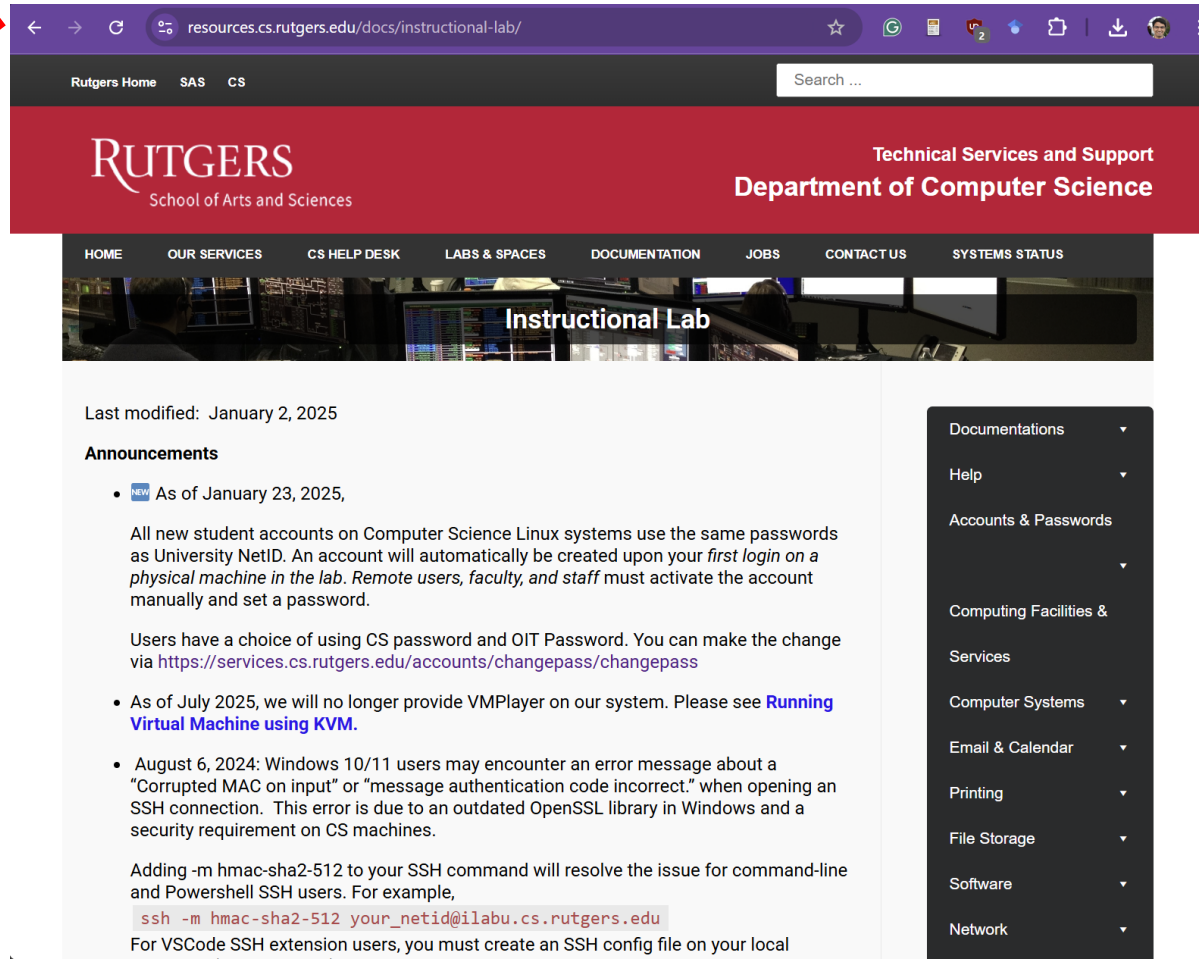
Programming Assignment 1

- 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





Layers of Abstraction for CS 211

Real-world Demands

Problem

Algorithm

Program

Runtime

System software

HW/SW Interface (ISA, ABI)

Microarchitecture

Logic gates

Circuits

Technology

Physics

Real-world Constraints

High-Level Language Program

```
int i;
for(i = 0; i < 10; i++)
    a[i] = b[i] * c[i];
```

Assembly Language Program

```
loop:  ld   a4, 0(a5)
       ld   a3, 8(a5)
       addi a5, a5,  -8
       add  a4, a4,  a3
       sd   a4, 8(a5)
       bne  a0, a5,  loop
```

Machine Language Program

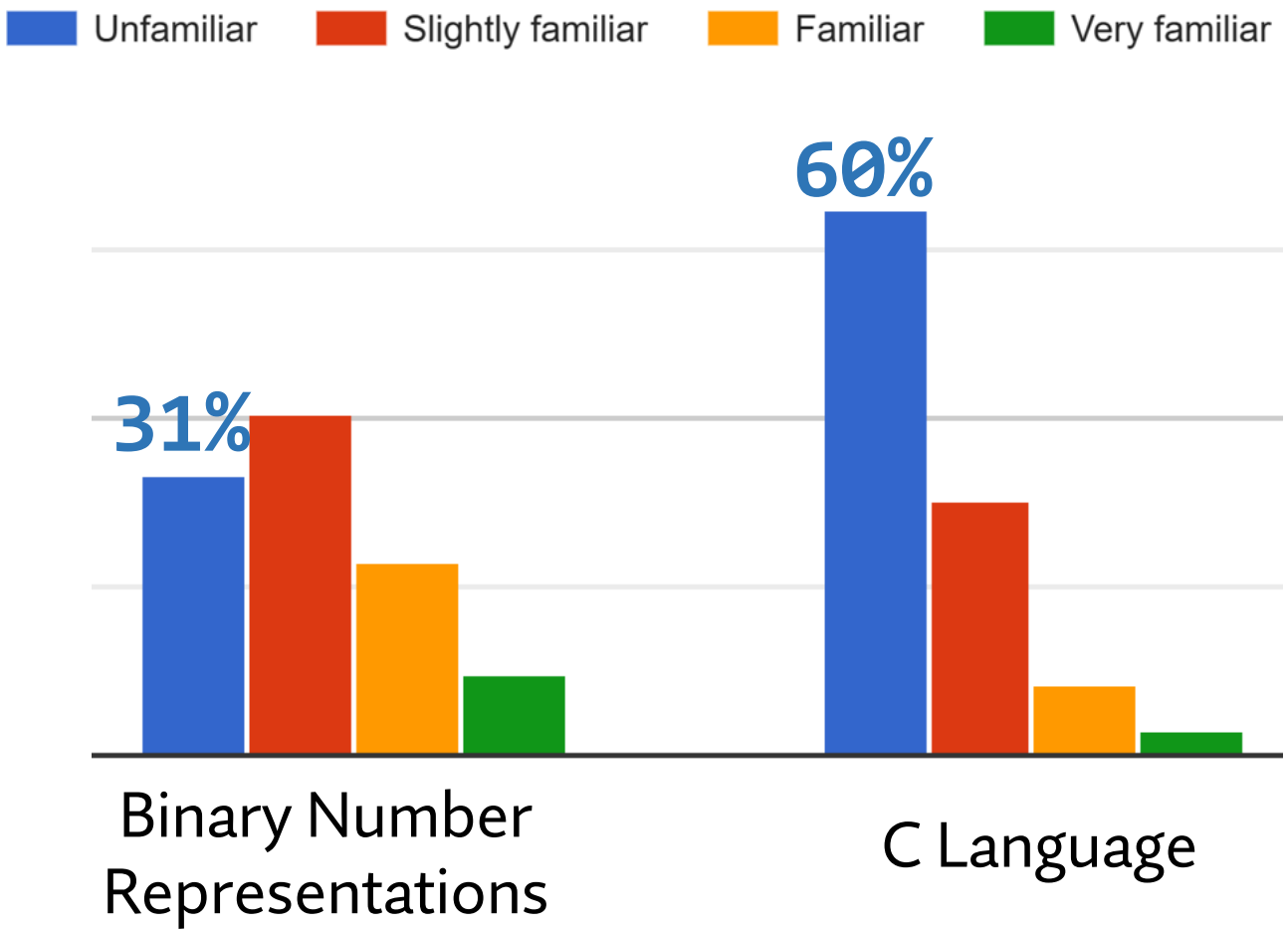
```
0000_0000_0000_0111_1011_0111_0000_0011
0000_0000_1000_0111_1011_0110_1000_0011
1111_1111_1000_0111_1000_0111_1001_0011
0000_0000_1101_0111_0000_0111_0011_0011
0000_0000_1110_0111_1011_0100_0010_0011
1111_1110_1111_0101_0001_0110_1110_0011
```

Instruction Set Architecture (ISA)

Microarchitecture

Logic Circuits

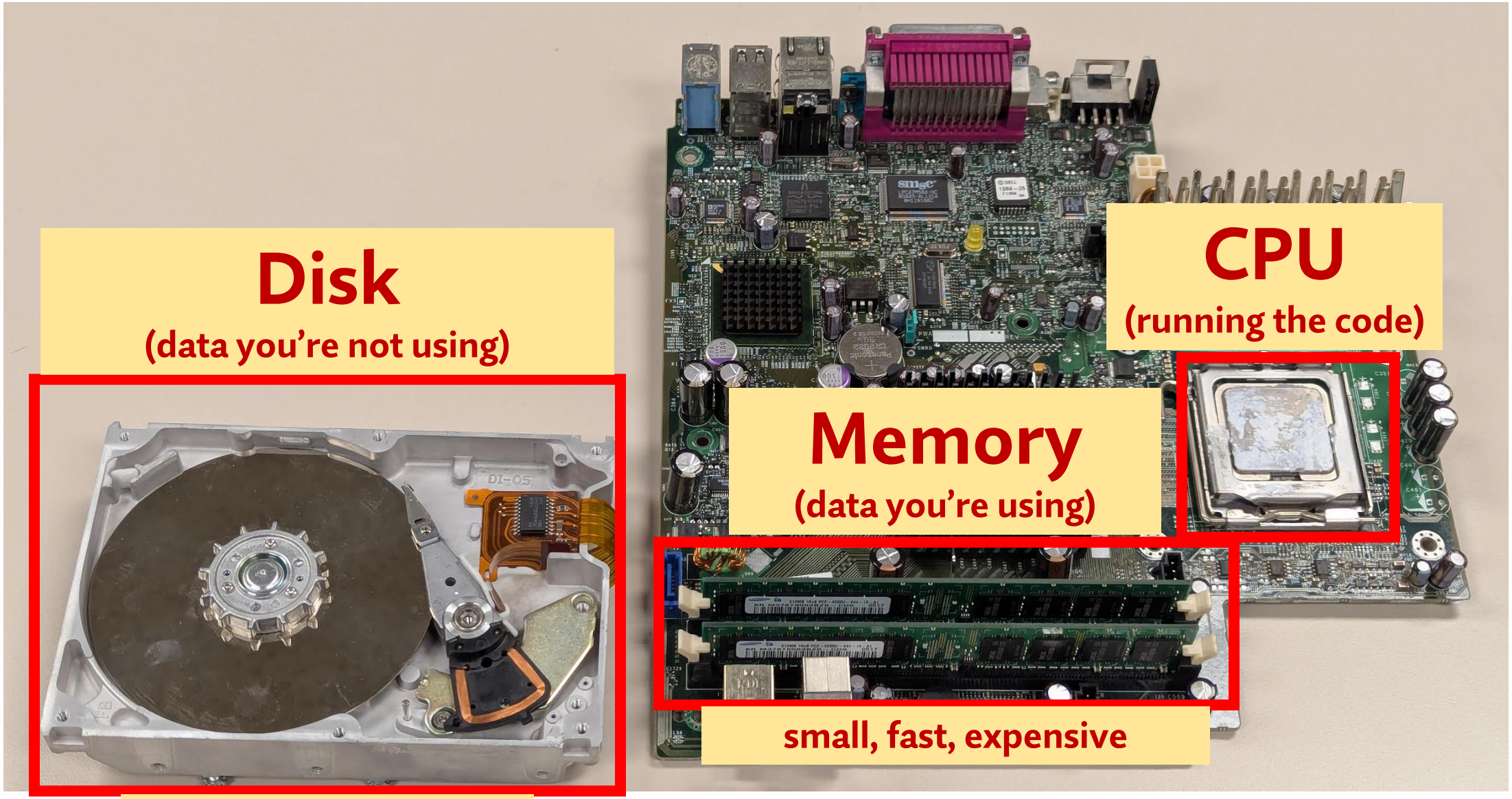
Next Several Weeks



Agenda

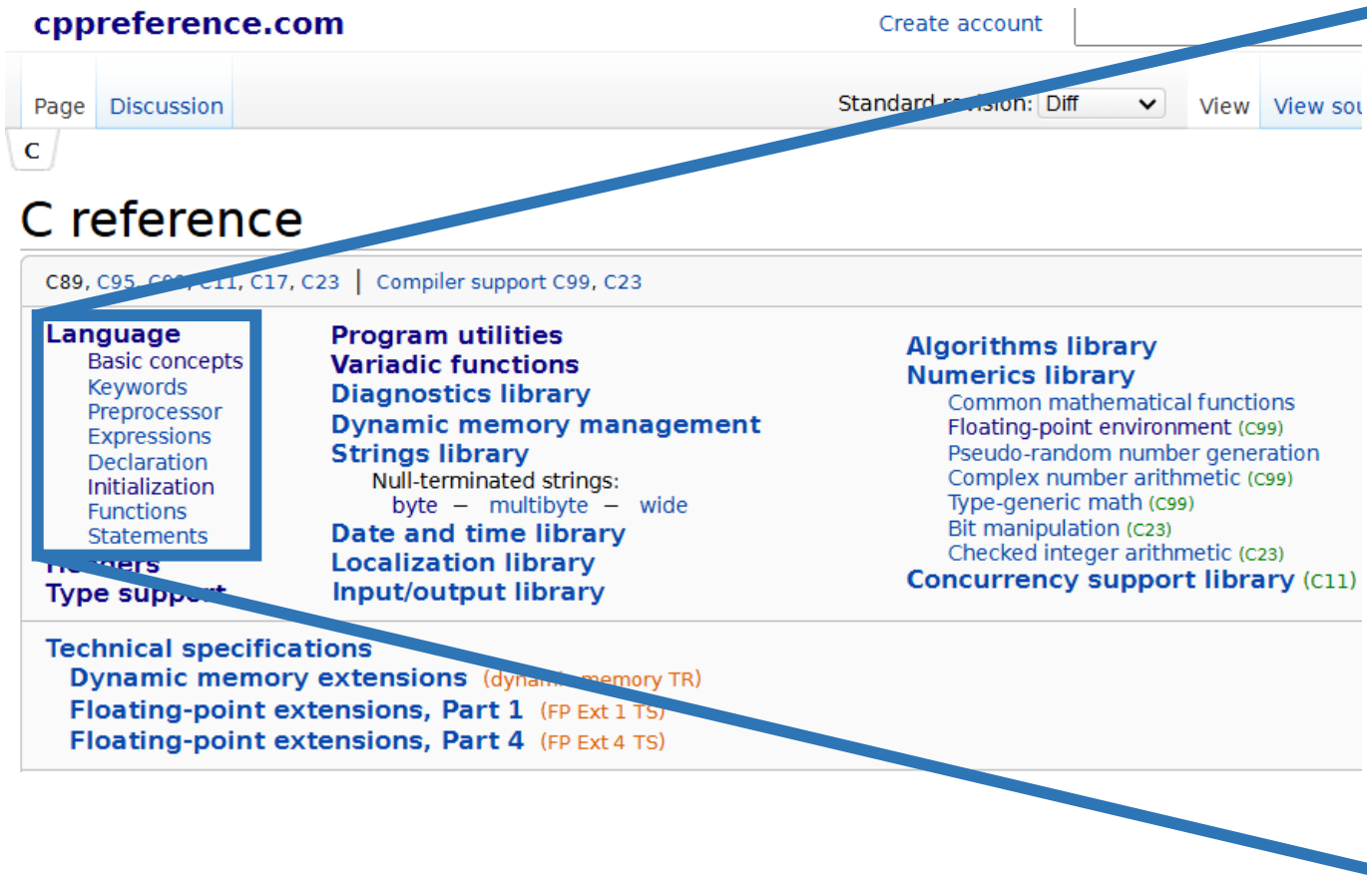
- **A Mental Model of Computer Memory**
- C “Objects”: Not Your Java Objects
- Storing Data in Bits and Bytes

Computer Memory

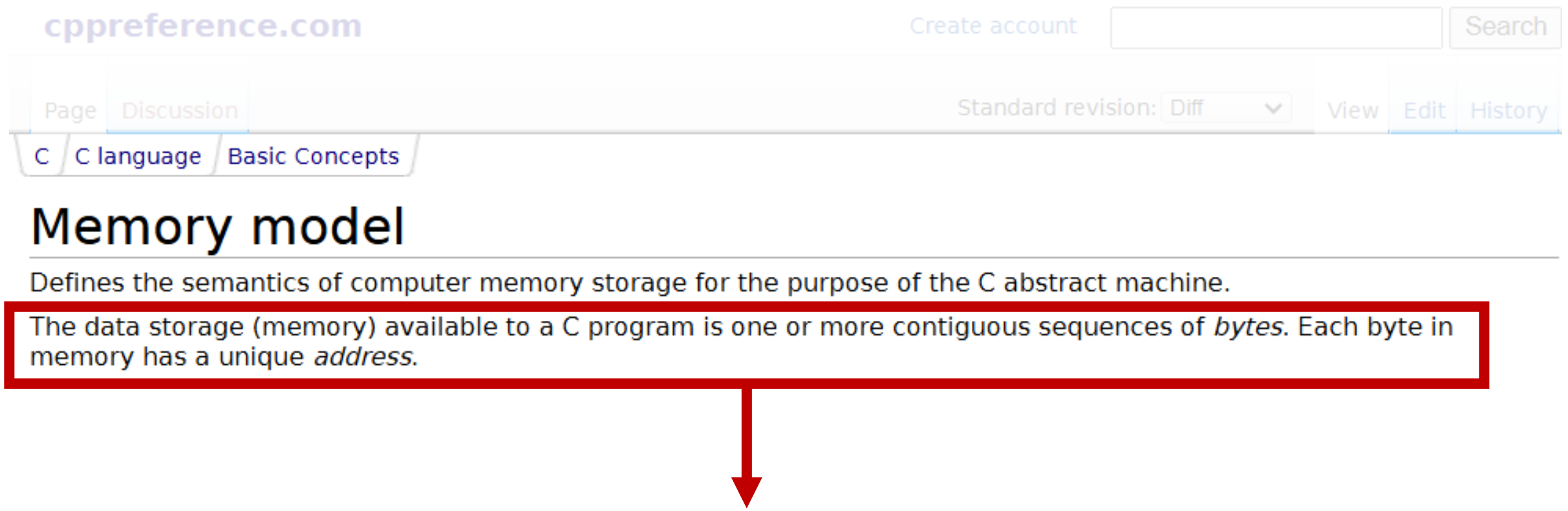


The C (and C++) Language Reference

cppreference.com



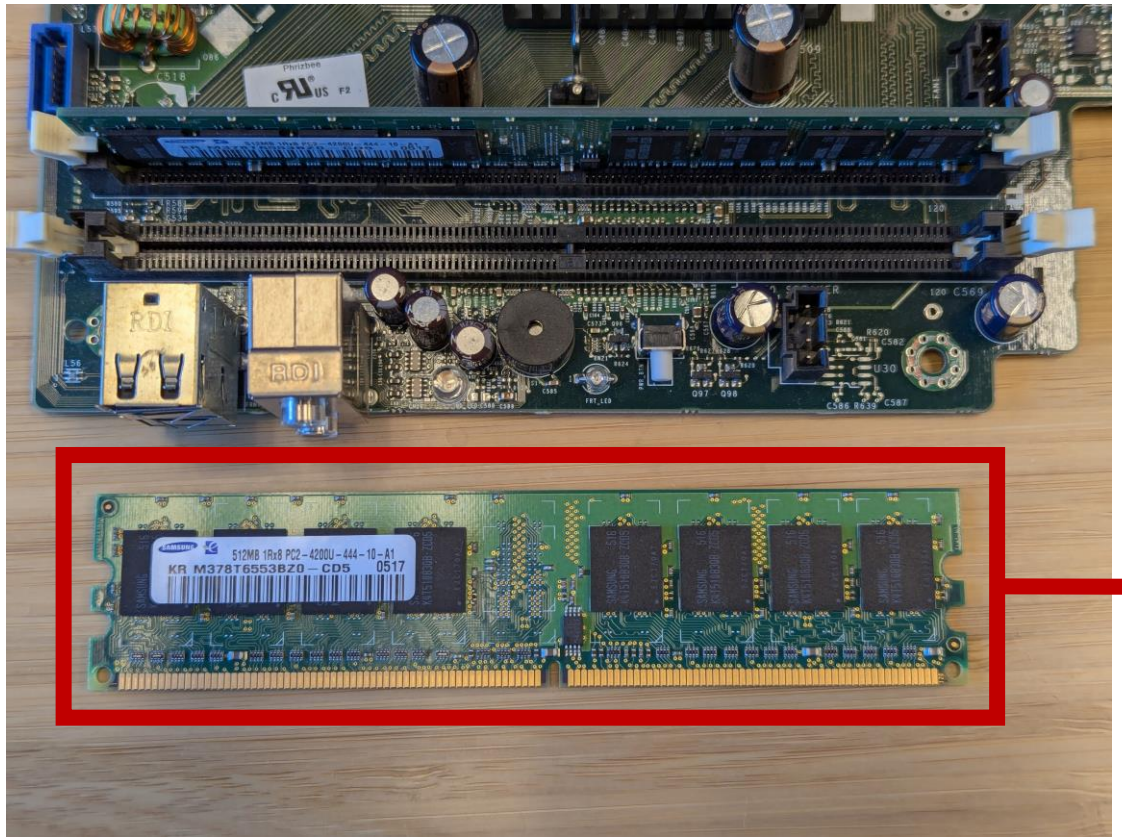
The C Memory Model

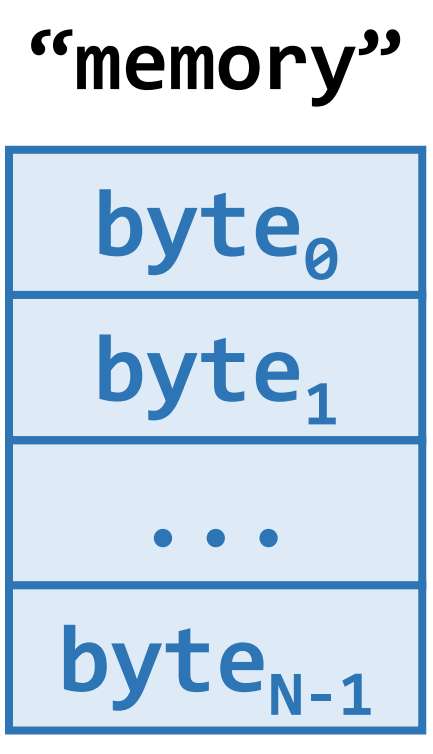


1. **Memory** is a contiguous sequence of **bytes**.
2. **Each byte** in memory has a **unique address**.

The C Memory Model

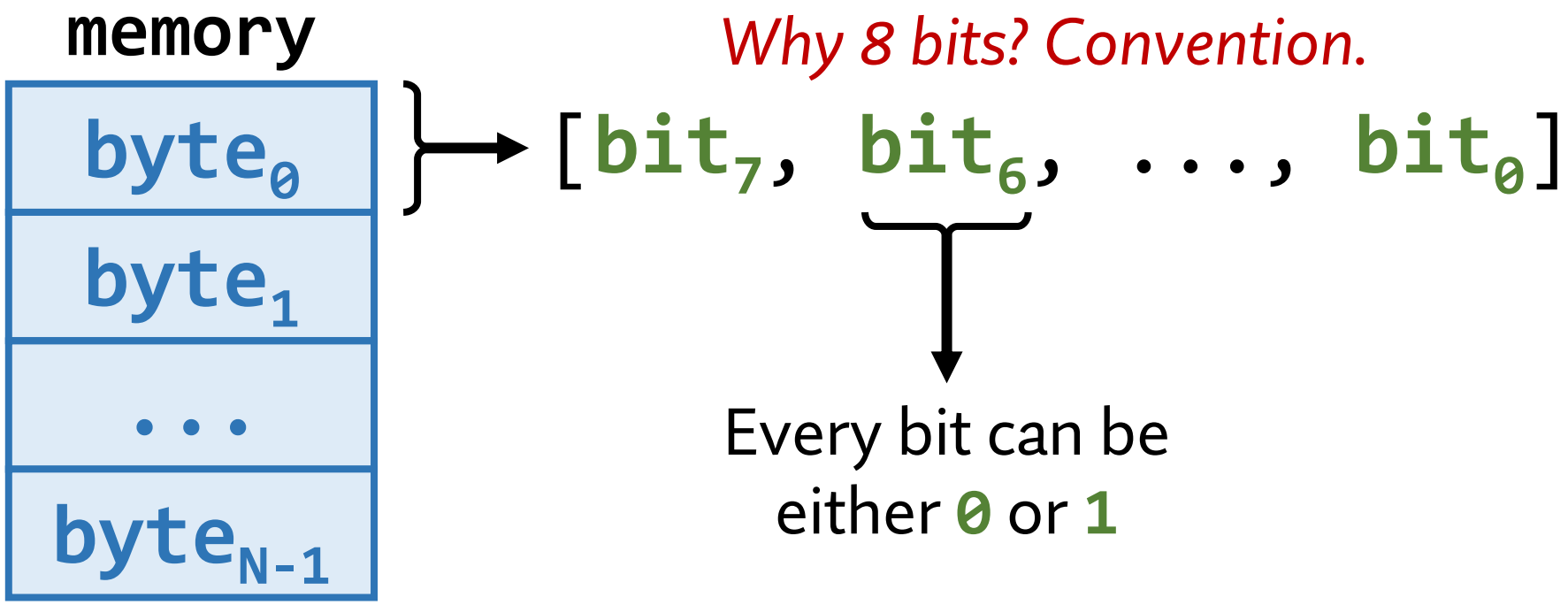
1. **Memory** is a contiguous sequence of **bytes**.





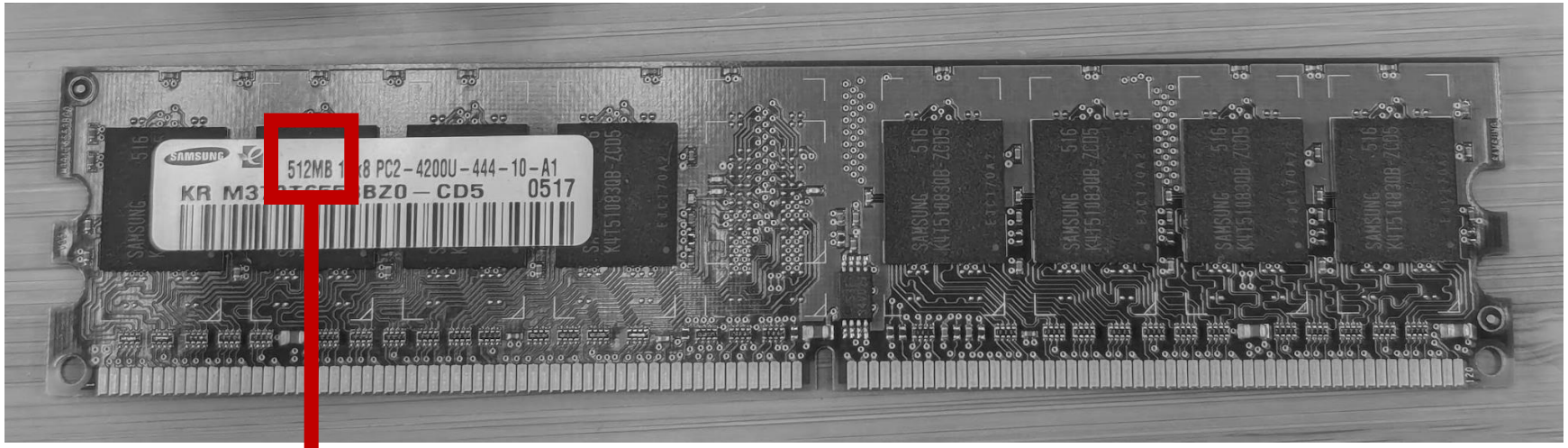
The C Memory Model

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



The C Memory Model

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

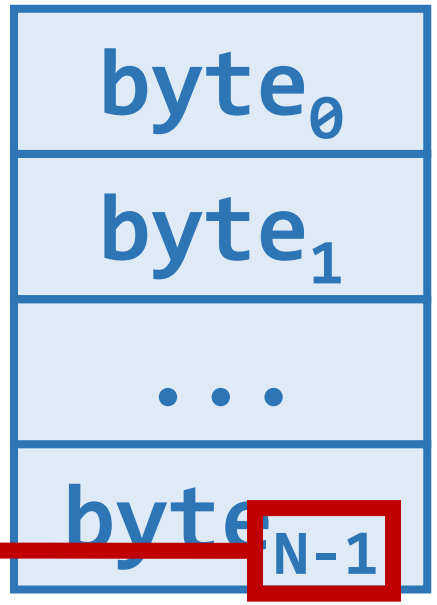


$M = 10^6$ $Mi = 2^{20} = 1048576$

$G = 10^9$ $Gi = 2^{30}$

512 MiB (2^{29} B)

"memory"



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

Why Bits?

- **BIT** = "**BI**nary digi**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

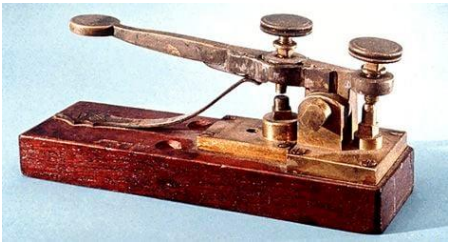
State
clipped/unclipped

Color
red/black

Line width
narrow/wide

Pixel color
Black/white

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



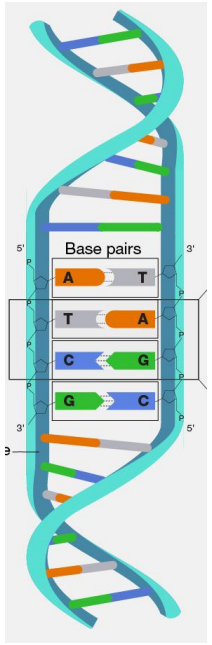
Sound Length
Long/short tap

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

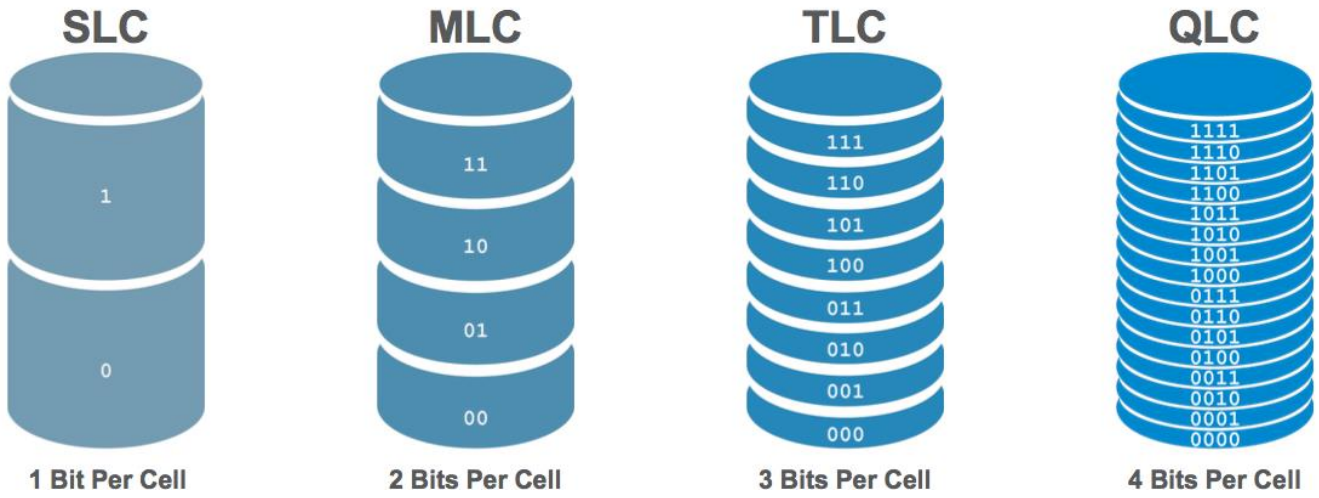


Base Choice
a/t/c/g

Knob setting
off/hi/med/lo

SSD (Flash Memory) Cell

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



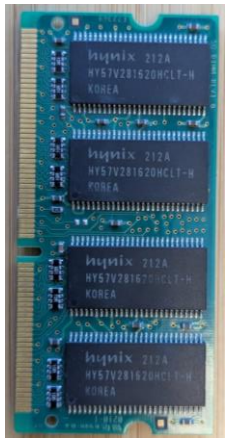
Voltage Level

Why Bits?

- **BIT** = "**BI**nary digi**T**"
  - 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

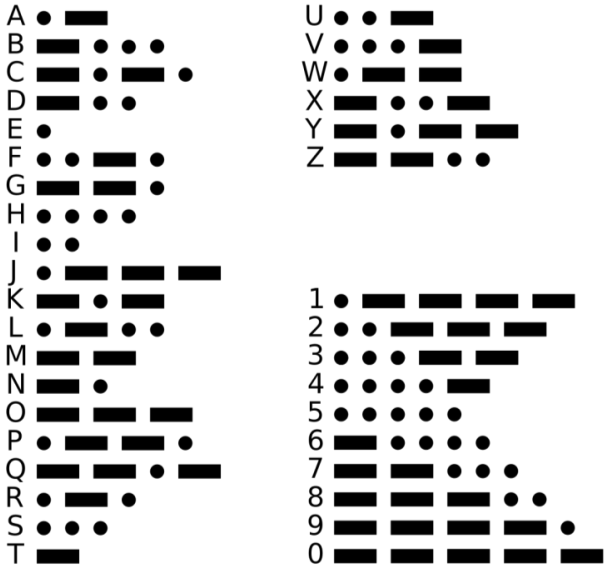
SSD
Transistor Voltage

Representing Data as Bits

- How can I store data using bits?

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

```
          String: I love CS211!
    Morse code: .. ..... .-.. --- ...- . ..... -.-. ... ..--- .---- .---- -.-.--
Bit equivalent: 00 0000000 0100 111 0001 0 0000000 1010 000 00111 01111 01111 101011
Also equivalent: LL LLLLLLL LHLL HHH LLLH L LLLLLLL HLHL LLL LLHHH LHHHH LHHHH HLHLHH
```



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

- 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

https://www.cs.cmu.edu/~pattis/15-1XX/common/handouts/ascii.html

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

base 10

$128 \leq 10^N$

$N = 3$

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

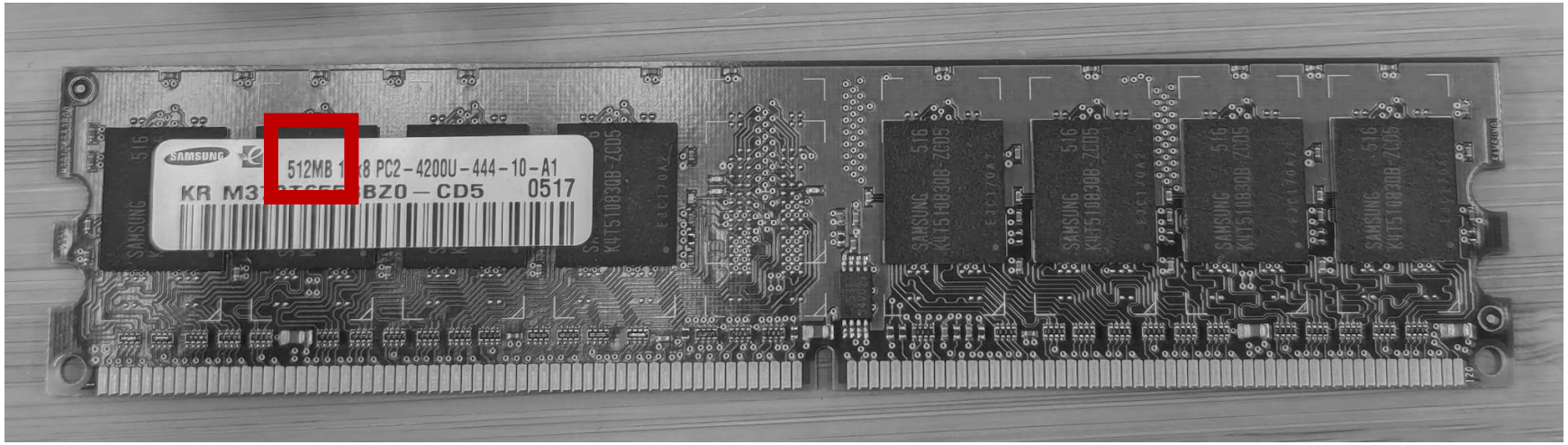
https://www.cs.cmu.edu/~pattis/15-1XX/common/handouts/ascii.html

Char → Representation

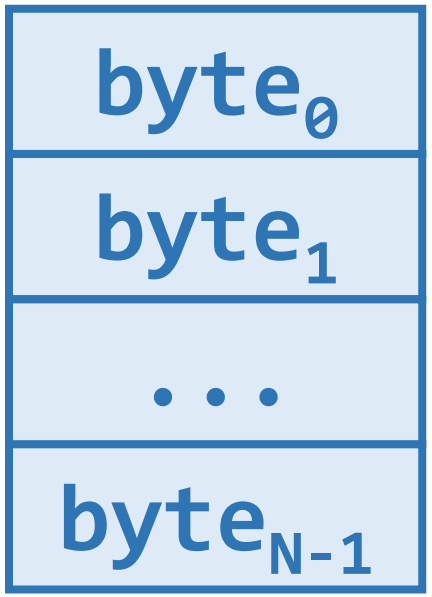
Char	Decimal	Binary
'I'	73	1001001
' '	32	0100000
'l'	108	1101100
'o'	111	1101111
'v'	118	1110110
'e'	101	1100101
' '	32	0100000
'C'	67	1000011
'S'	83	1010011
'2'	50	0110010
'1'	49	0110001
'1'	49	0110001
'!'	33	0100001

Back to The C Memory Model

1. Memory is a contiguous sequence of bytes.



"memory"

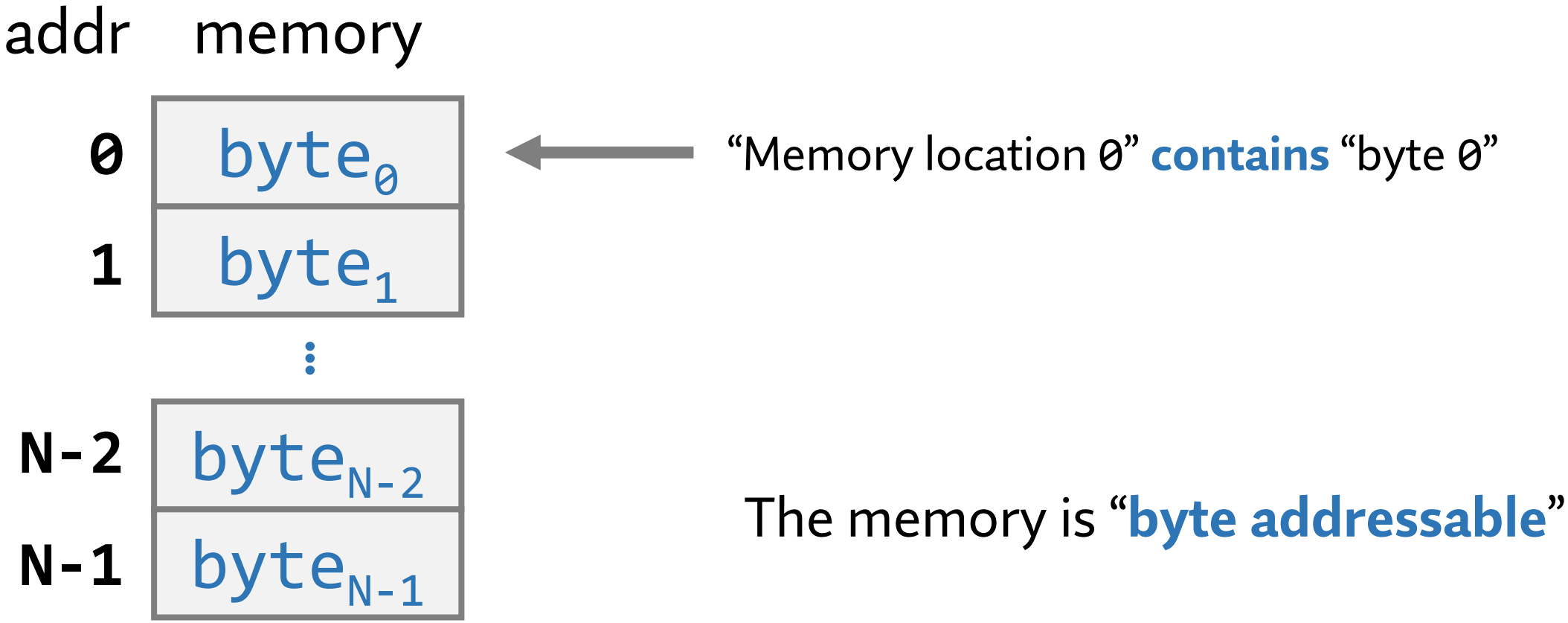


- `512 MiB` = 2^{29} bytes
- Every byte represents 2^{8} unique values

- ...what should we do with all these values?

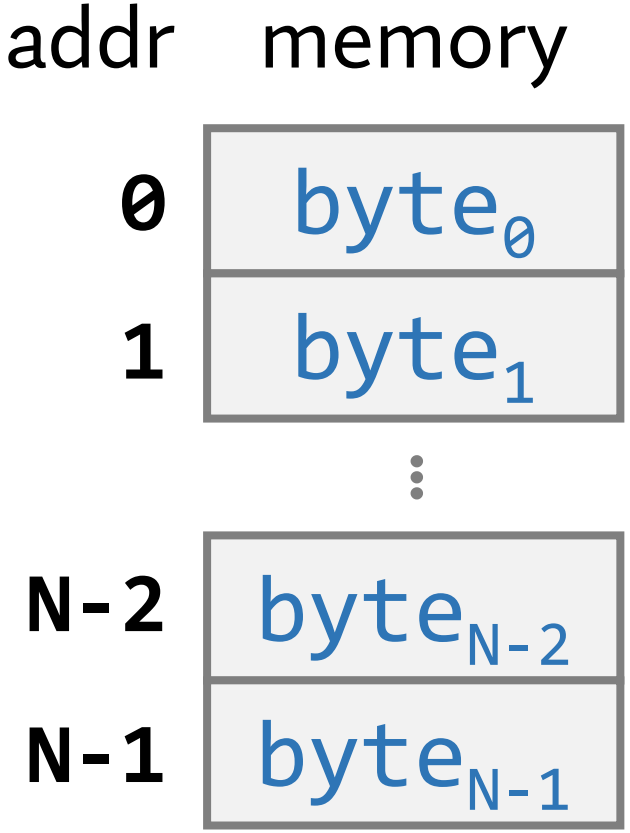
The C Memory Model

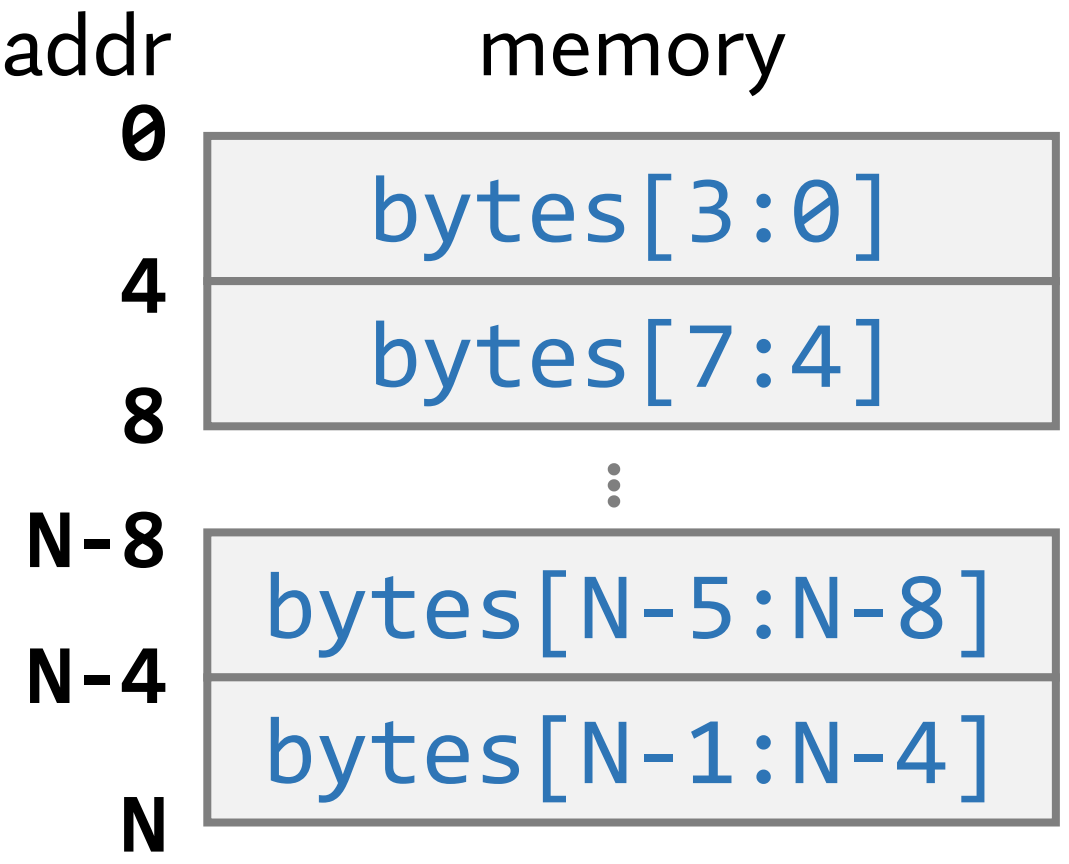
1. **Memory** is a contiguous sequence of **bytes**.
2. **Each byte** in memory has a **unique address**.



The C Memory Model

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

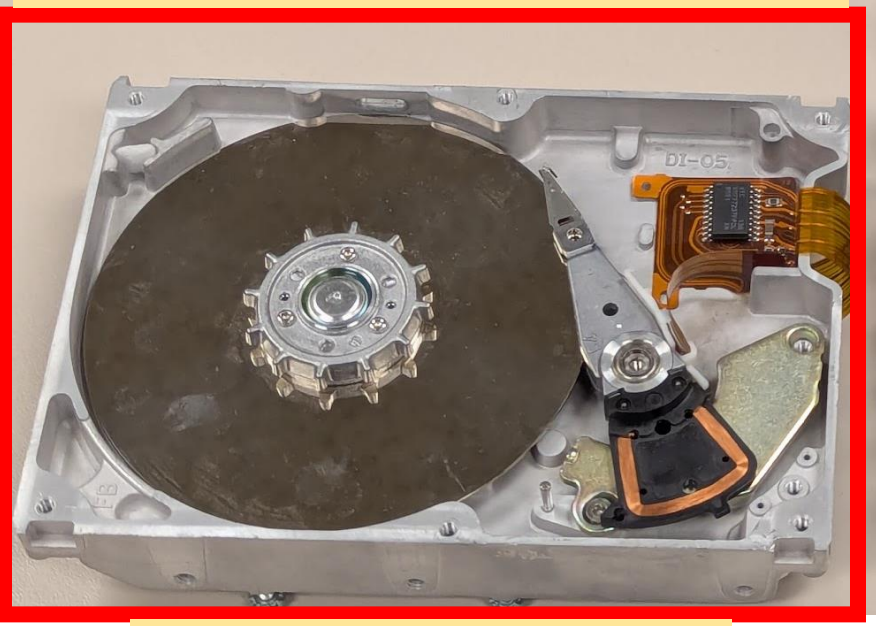




Computer Memory

Disk

(data you're not using)



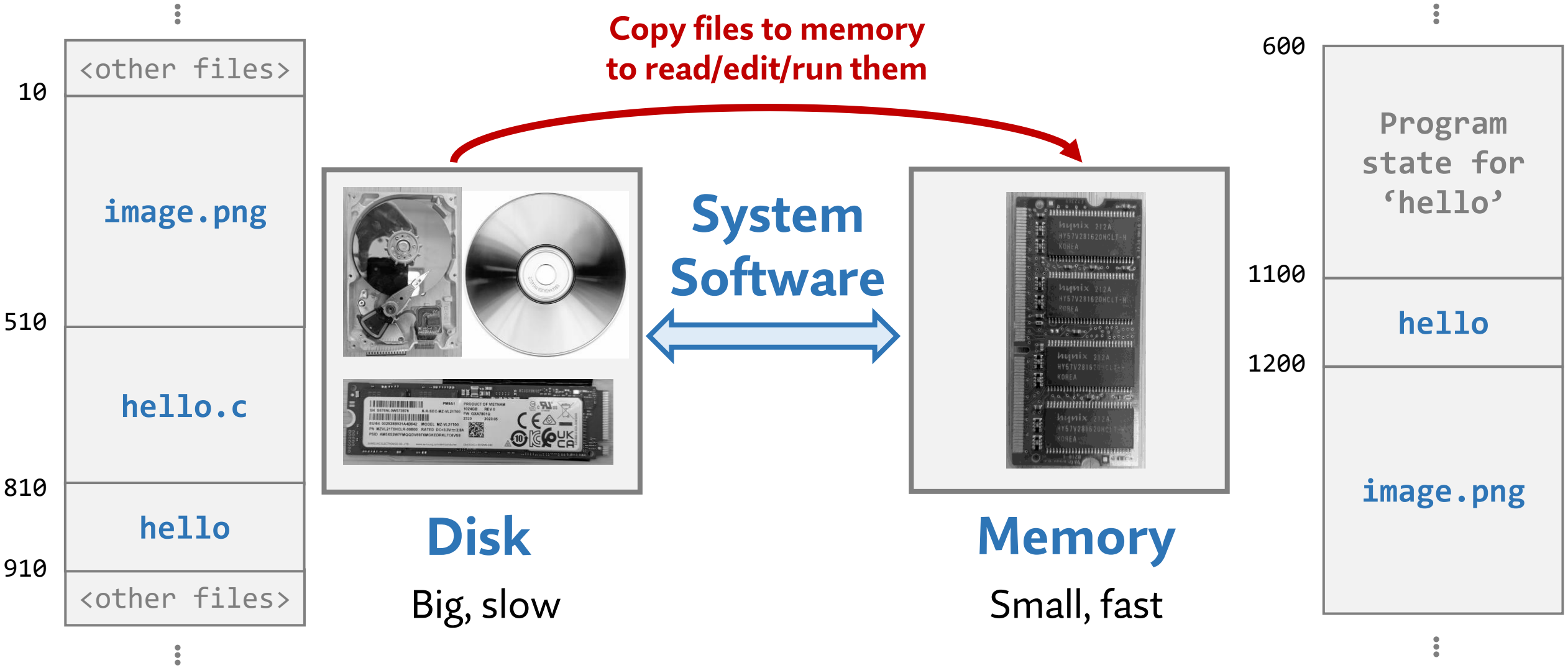
big, slow, cheap

Memory

(data you're using)

small, fast, expensive

Memory Locations: System-Level View

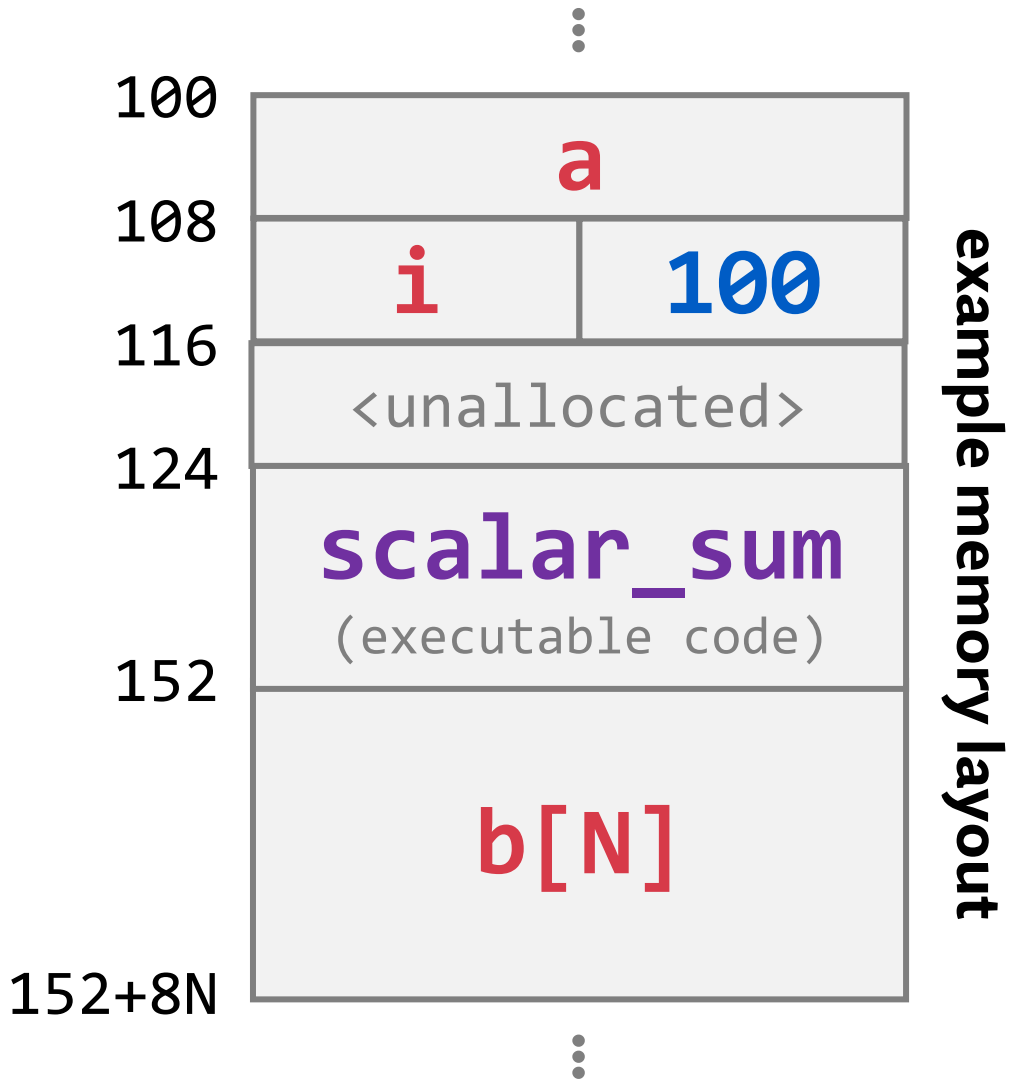


Memory Locations: Inside a C Program's Memory

```
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

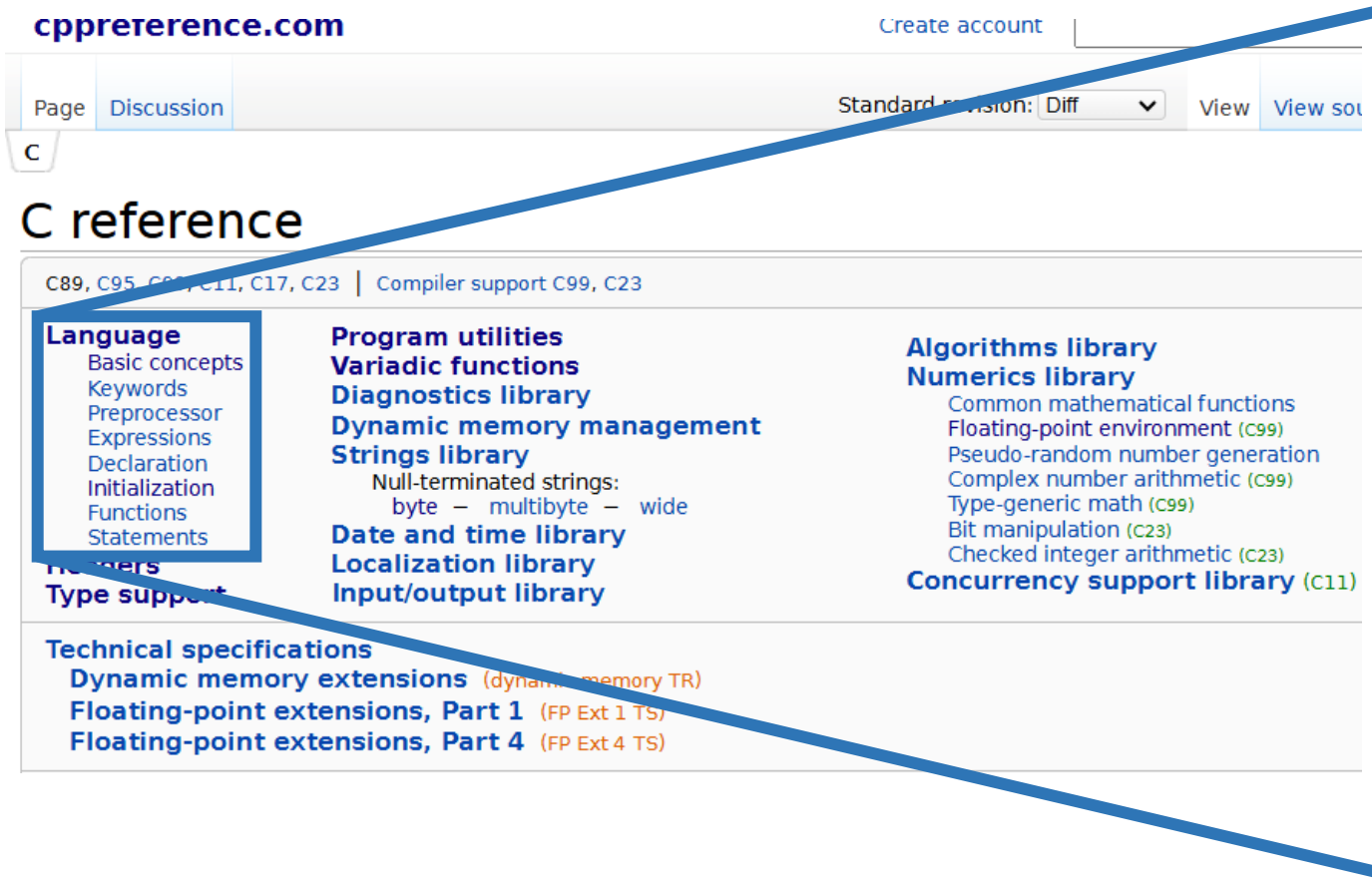
- &a = 100
- &i = 108 / 112
- &t = 112 / 108
- &b = 152
- &scalar_sum = 124

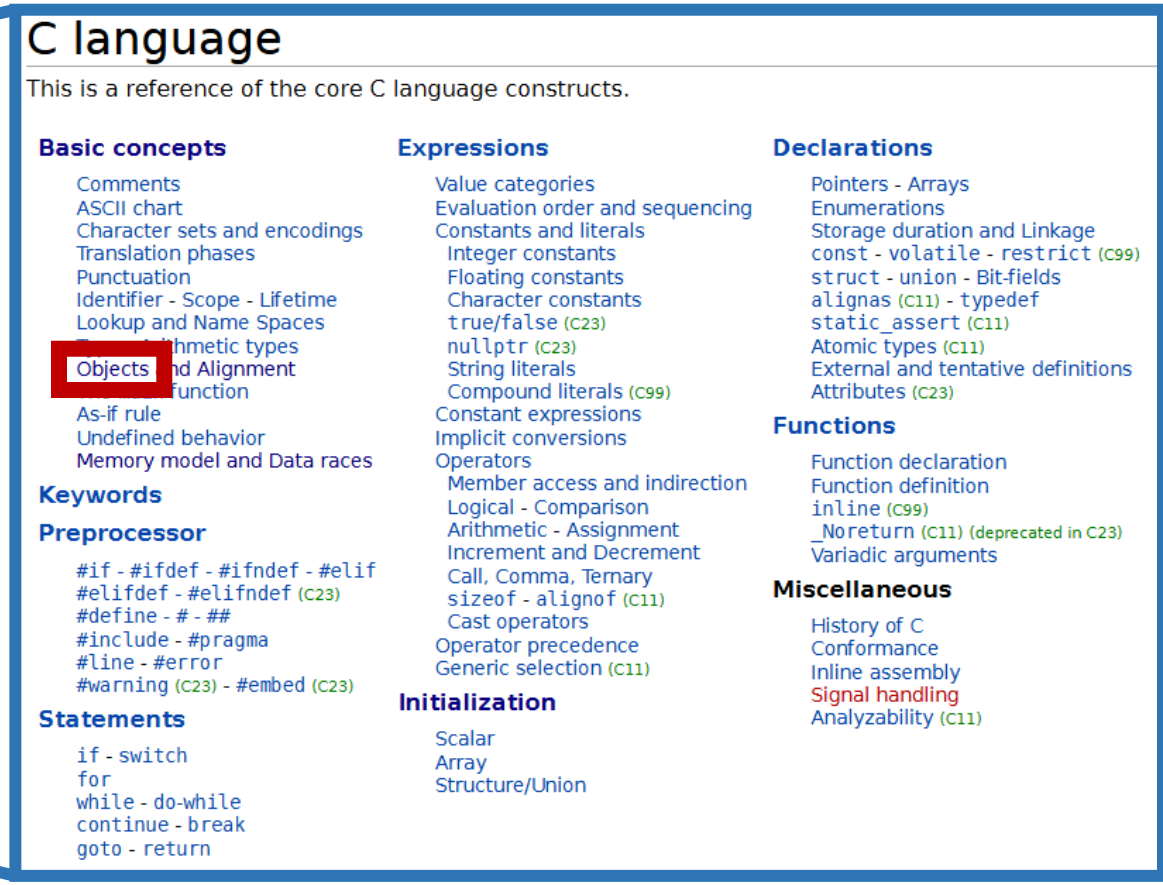


Agenda

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

The C (and C++) Language Reference





Objects in C

cppreference.com Create account Search

Page Discussion Standard revision: Diff View Edit History

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*

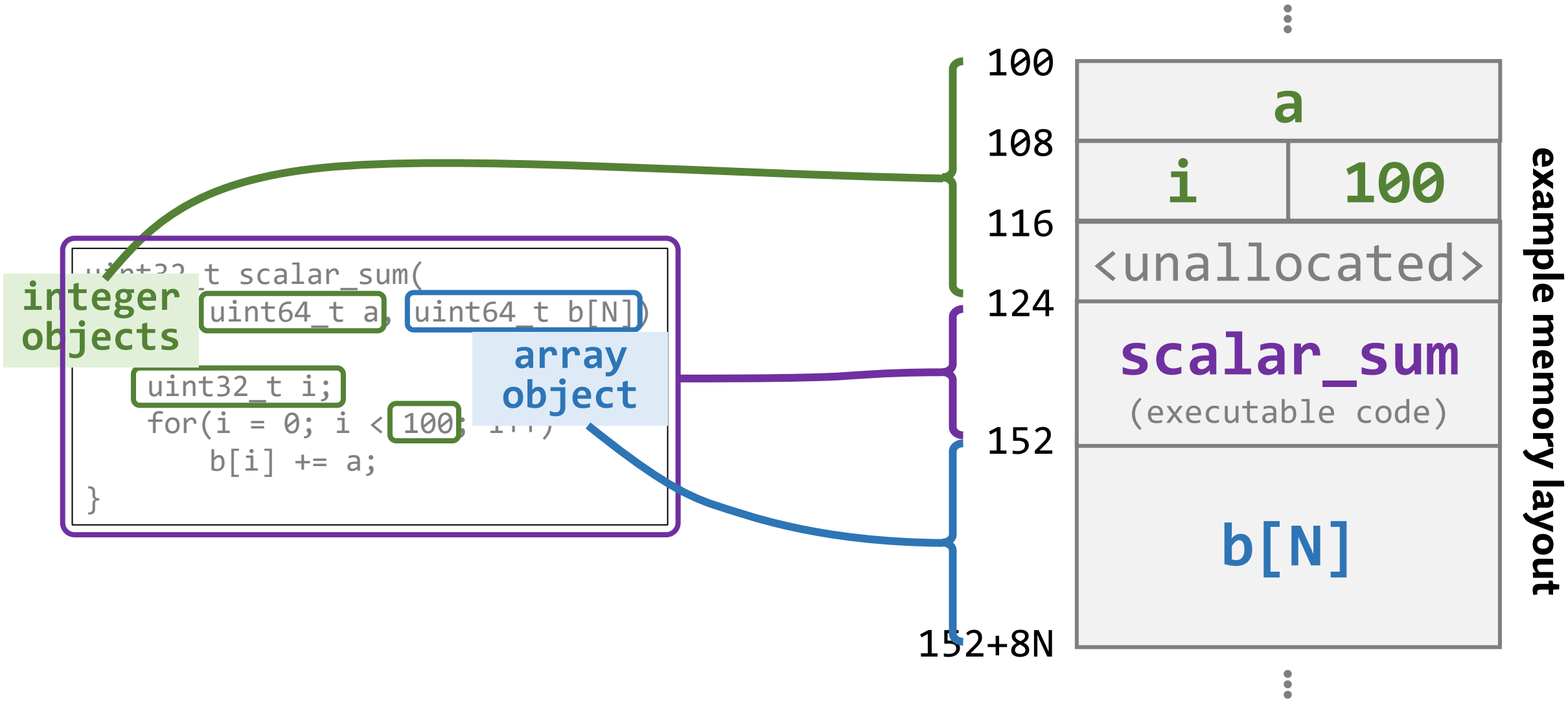
Things we would like to store

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

C objects are **completely unrelated** to:

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

C Objects: Code Perspective



C Objects

Every object has

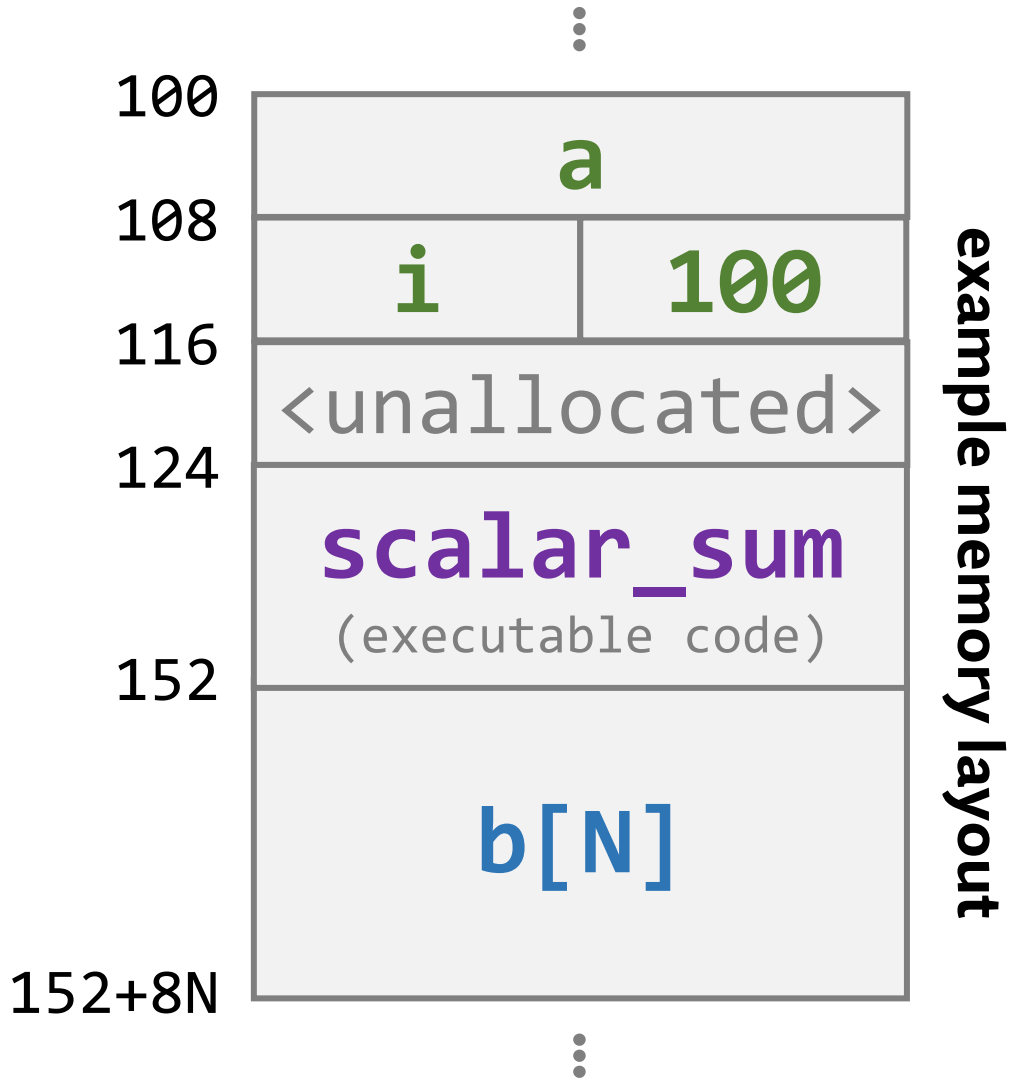
- size (can be determined with `sizeof`)
- alignment requirement (can be determined by `_Alignof`) (since C11)
- storage duration (automatic, static, allocated, thread-local)
- lifetime (equal to storage duration or temporary)
- effective type (see below)
- value (which may be indeterminate)
- optionally,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

"&" = "address of" operator

- `&a` = 100

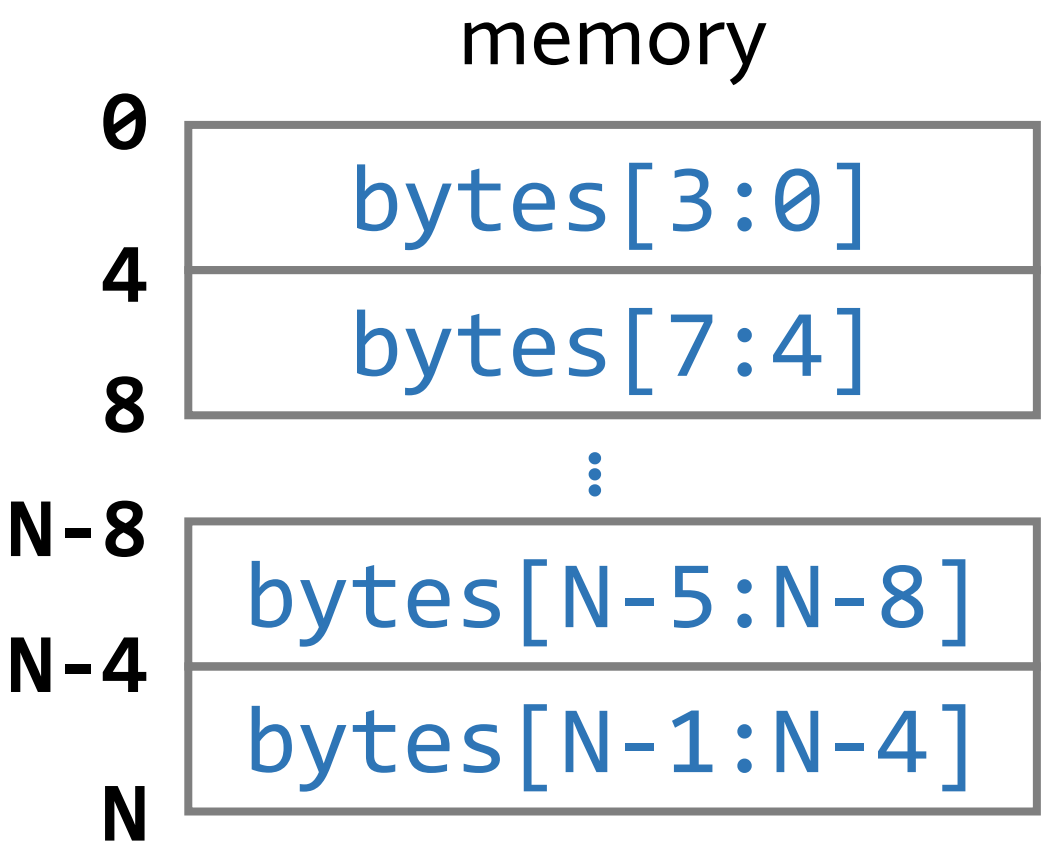
"sizeof()" operator

- `sizeof(a)` = 8 (bytes)
- `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**.

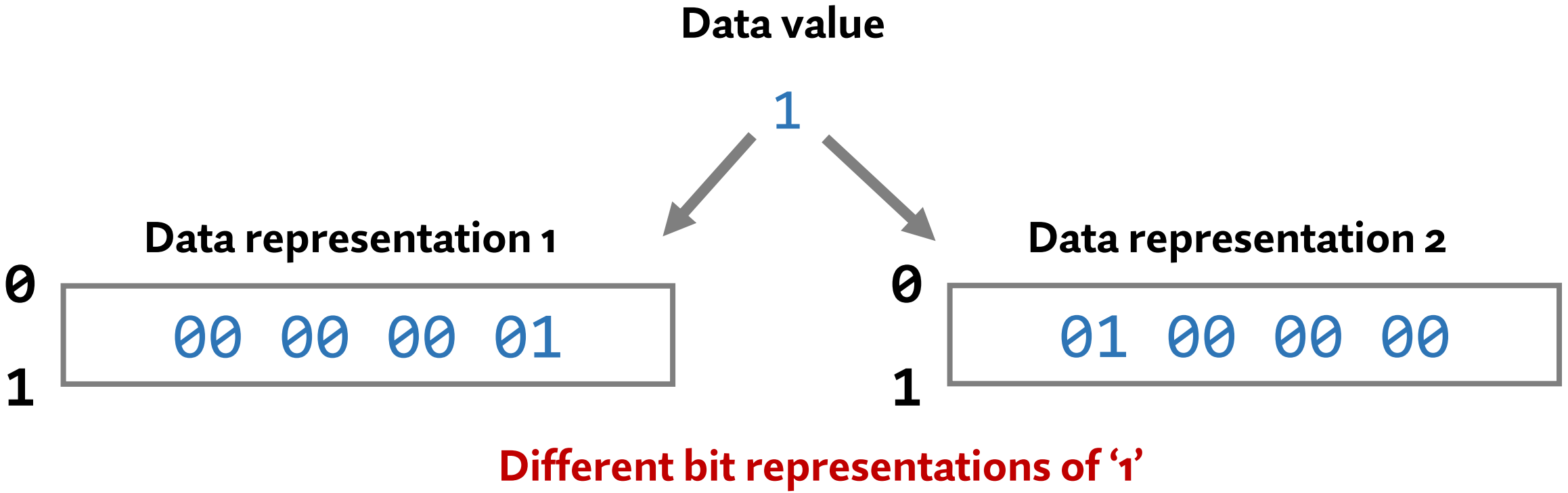


Agenda

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

Representing Values in Objects

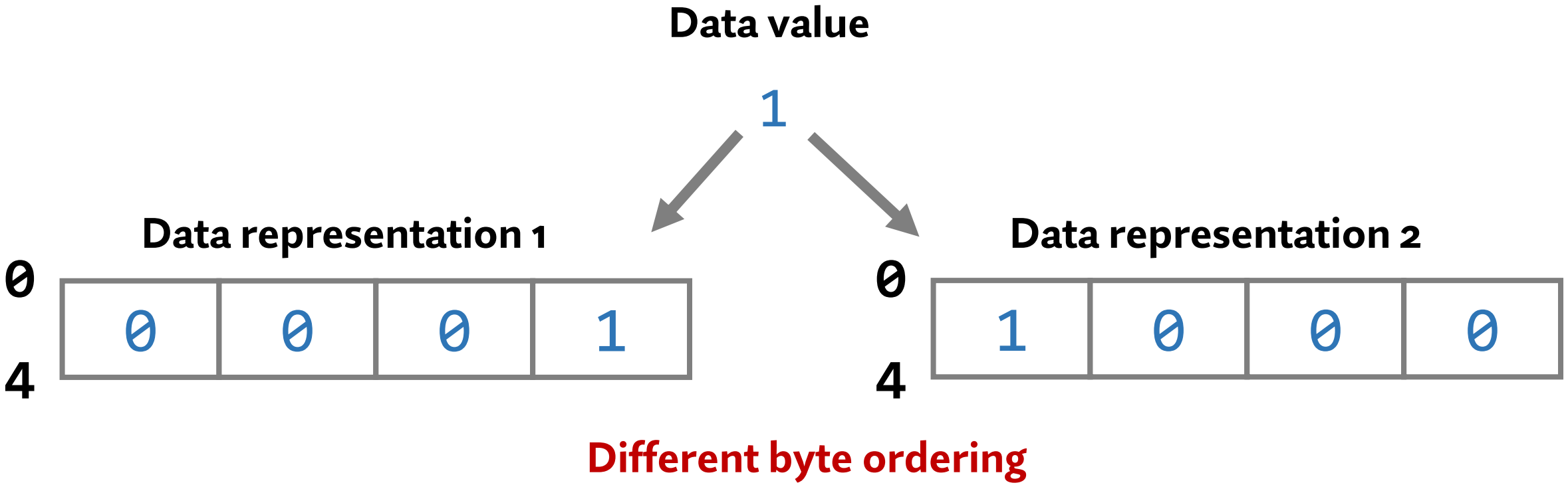
- **Data values:** things we would like to keep in an object
- **Data representation:** how an object represents the value



- 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

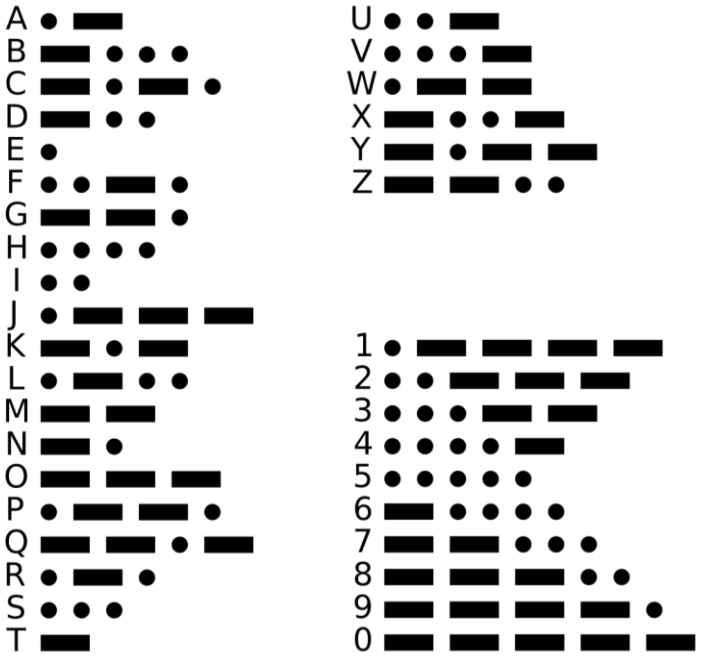
Value Representations

- **Today:** Number Systems, Value Representations, Characters
- **Today + 1:** Integers (e.g., 1)
- **Today + 2:** Floating point (e.g., `6.022 x 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”



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 = <a bit complicated>

Positional Codes (for Numbers)

- Weights are **implied by digit position**

Value	Representation	Code
$(1699)_{10}$	$(1 * \mathbf{10}^3) + (6 * \mathbf{10}^2) + (9 * \mathbf{10}^1) + (9 * \mathbf{10}^0)$	$\sum_i d^i * 10^i$

Base 10 digits {0,1,…,9}

Most significant digit

Least significant digit

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

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

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+/}

$(63)_2$ = —

$(63)_4$ = —

$(63)_8$ = $6 \cdot 8^1 + 3 \cdot 8^0 = (51)_{10}$

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

$(63)_{16}$ = $6 \cdot 16^1 + 3 \cdot 16^0 = (99)_0$

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

$(f8)_2$ = —

$(f8)_4$ = —

$(f8)_8$ = —

$(f8)_{10}$ = —

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

$(f8)_{64}$ = ...

$(58)_{10}$

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+/}

$(63)_2$ = <invalid>

$(63)_4$ = <invalid>

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

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

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

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

$(f8)_2$ = <invalid>

$(f8)_4$ = <invalid>

$(f8)_8$ = <invalid>

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

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

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

Special Notation for Powers of 2

Base	Nickname	Digit Name	Notation Example
2	binary	bit	`0b1010010101`
4	quaternary	-	$(1230123)_4$
8	octal	octet	`0o134061304`
10	decimal	digit	$(907315904)_{10}$
16	hexadecimal (hex)	nibble	`0xfe3b91ad31`

Base Conversions

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

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

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

$$16 \overline{)429} \quad 26 \text{ R } 13$$

$$16 \overline{)26} \quad 1 \text{ R } 10$$

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

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

$$= (1\,a\,d)_{16} = 0x1ad$$

Base Conversions Between Powers of 2

$$(000000010010001101000101011100111)_2$$

$$(00\ 00\ 00\ 01\ 00\ 10\ 00\ 11\ 01\ 00\ 01\ 01\ 01\ 10\ 01\ 11)_2$$
$$(0\ 0\ 0\ 1\ 0\ 2\ 0\ 3\ 1\ 0\ 1\ 1\ 1\ 2\ 1\ 3)_4$$

$$(00\ 000\ 001\ 001\ 000\ 110\ 100\ 010\ 101\ 100\ 111)_2$$
$$(0\ 0\ 1\ 1\ 0\ 6\ 4\ 2\ 5\ 4\ 7)_8$$

$$(0000\ 0001\ 0010\ 0011\ 0100\ 0101\ 0110\ 0111)_2$$
$$(0\ \ 1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6\ \ 7)_{16}$$

CS 211: Intro to Computer Architecture

2.1: Memory, C Objects, and Data Representation

Minesh Patel

Spring 2025 – Tuesday 28 January