

A Free Online Textbook Introducing Computer Architecture Topics

Tia Newhall

Swarthmore College
Swarthmore, PA USA



Suzanne J. Matthews

U.S. Military Academy
West Point, NY USA



Kevin C. Webb

Swarthmore College
Swarthmore, PA USA

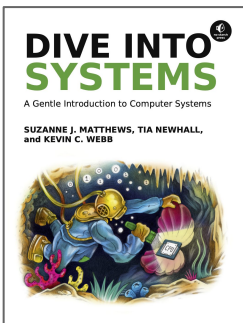


diveintosystems.org

The opinions expressed in this presentation are solely of the authors and do not necessarily reflect those of the U.S. Military Academy, the DoD or the U.S. Army.

Dive into Systems:

Free, online textbook introducing systems, architecture & parallel computing, available online at diveintosystems.org
Anyone with internet access can use our book!



No Starch Press, September 2022

Also new low-cost print version

- Published by No Starch Press
- For readers who want a print version

Will always also remain free online!

Dive Into Systems

5.2. The von Neumann Architecture

The von Neumann architecture serves as the foundation for most modern computers. In this section, we briefly characterize the architecture's major components.

The von Neumann architecture (depicted in Figure 1) consists of five main components:

1. The **processing unit** executes program instructions.
2. The **control unit** drives program instruction execution on the processing unit. Together, the processing and control units make up the CPU.
3. The **memory unit** stores program data and instructions.
4. The **input unit(s)** load program data and instructions on the computer and initiate program execution.
5. The **output unit(s)** store or receive program results.

Buses connect the units, and are used by the units to send control and data information to one another. A bus is a communication channel that transfers binary values between communication endpoints (the senders and receivers of the values). For example, a data bus that connects the memory unit and the CPU could be implemented as 32 parallel wires that together transfer a 4-byte value, 1-bit transferred on each wire. Typically, architectures have separate buses for sending data, memory addresses, and control between units. The units use the control bus to send control signals that request or notify other units of actions, the address bus to send the memory address of a read or write request to the memory unit, and the data bus to transfer data between units.

The CPU

1. Processing Unit
ALU registers

2. Control Unit
PC IR

3. Memory Unit

4. Input Units

5. Output Units

address bus

control bus

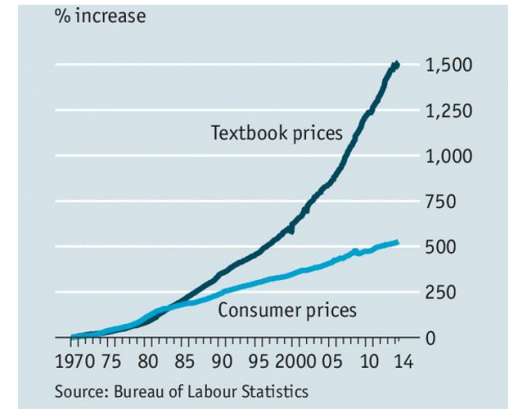
Why a free online textbook?

Selfish: We couldn't find "right fit" textbook for our courses

Intro to broad range of systems, architecture, parallel topics
at the intro sequence level (assume only CS1 background)

Altruistic: Create Useful Resource to Share Widely

- Free (**cost not a barrier to access**)
- Online (**easy to access**) and update
- Useful resource for lots of different uses
 - "Mix and match" content easily
 - Primary text: intro. systems, computer organization, C programming, ...
 - Supplemental text: Arch, OS, Compilers, P&D, DB, ...



Source: The Economist

Content Overview

Three Main Themes:

1. How a computer runs a program
2. How systems costs affect program performance (Memory Hierarchy, other)
3. How to leverage the power of parallel computing

Main Architecture Content

- Chapter 5 on Computer Architecture
- Chapter 11 on Memory Hierarchy and Caching
- Binary Representation & Arithmetic
- Some Parallel Architecture Coverage: Chapter 5, 11, 14, 15
- HW-OS interface: TLB, VM, interrupts, user/kernel level

1. C Intro
2. C Depth
3. C Debugging
12. Code Optimization

6. - 10. Assembly
IA32, x86_64, ARM 64

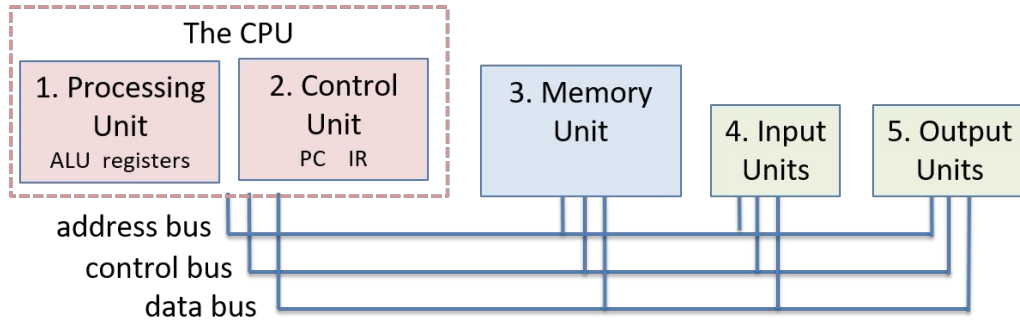
4. Binary Representation
11. Memory Hierarchy
5. Architecture

13. Operating Systems

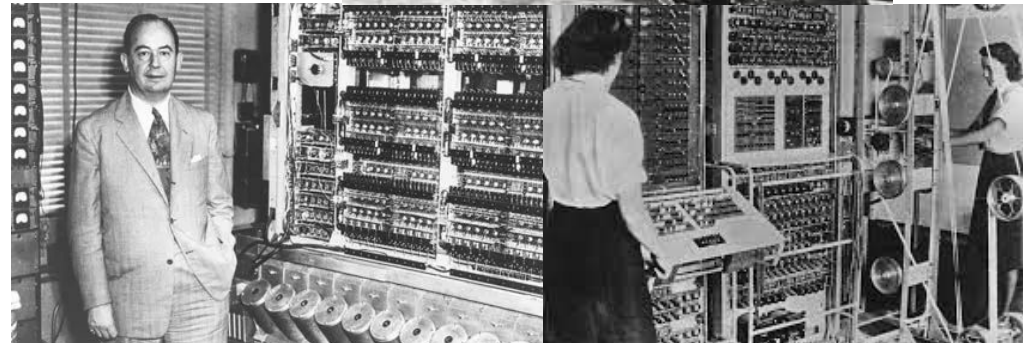
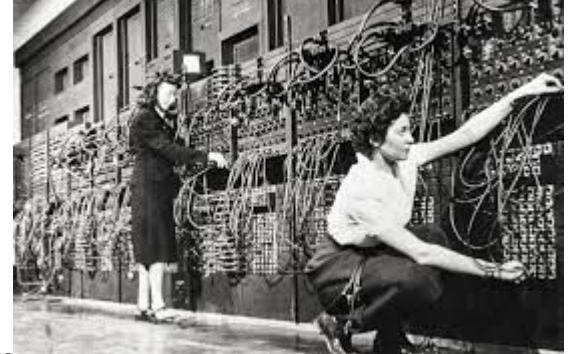
14. Shared Memory Parallel
15. Other Parallel

Coming soon:
Using Unix
Appendix

Von Neumann Architecture and Computer Architecture History



How it executes instructions:
Fetch-Decode-Execute-StoreResult
PC and IR
Instruction: opcode & operands



CPU Architecture: How Computer Runs a Program

Build simple CPU from bottom up, 1-bit circuits from logic gates

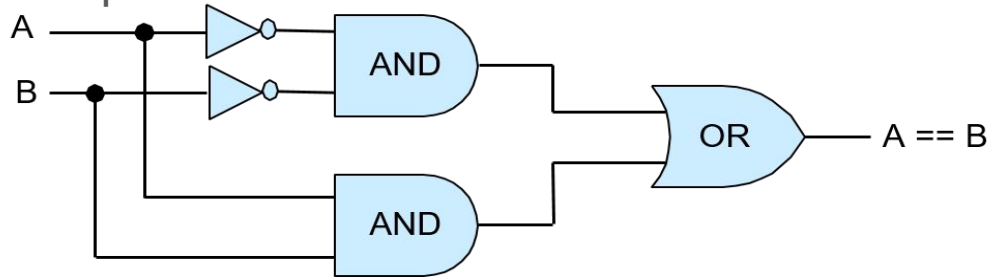
1. Create truth table for operation

A	B	A == B
0	0	1
0	1	0
1	0	0
1	1	1

2. Expressions for rows w/output 1 using AND, OR, NOT, combine rows with OR:

$$(\text{NOT}(A) \text{ AND NOT } (B)) \text{ OR } (A \text{ AND } B)$$

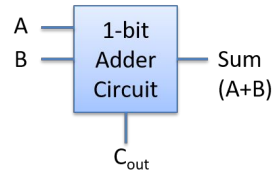
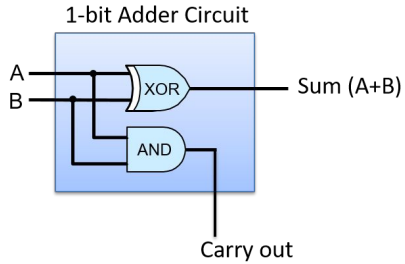
3. Translate expression into sequence of logic gates from inputs to output



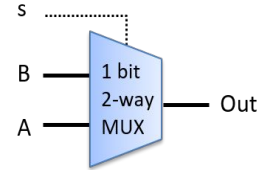
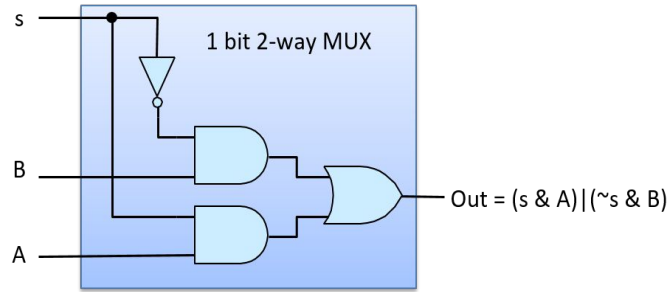
3 Types of Circuits:

mirror Von Neumann Architecture

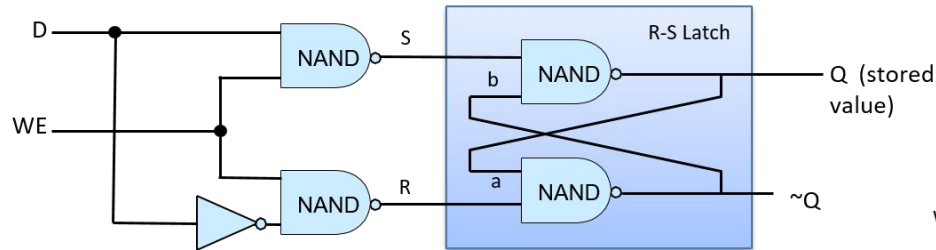
Arithmetic/Logic



Control

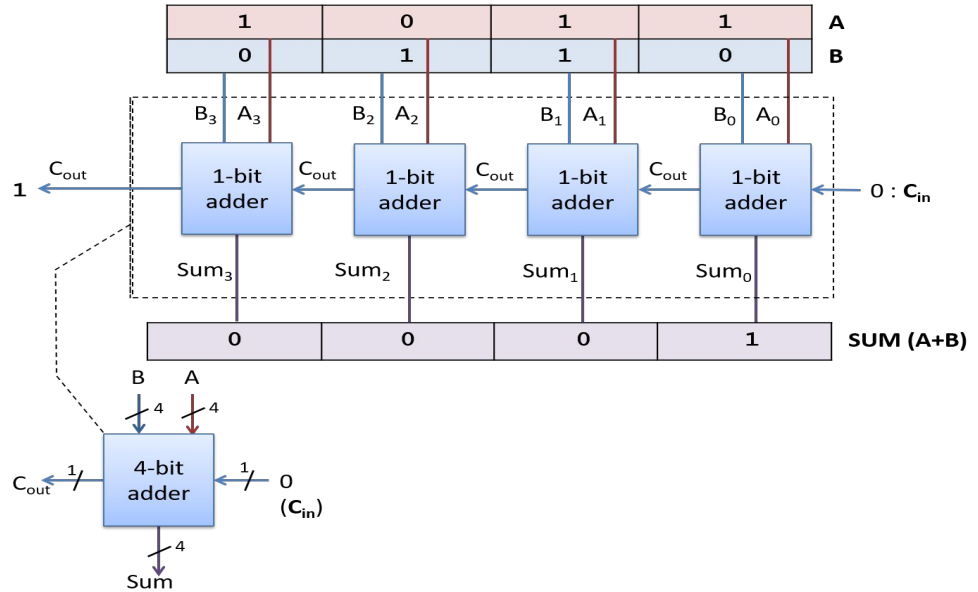
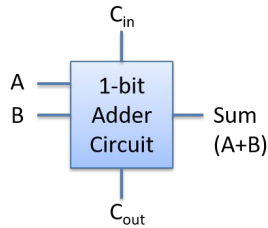
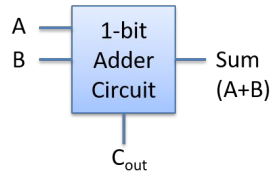
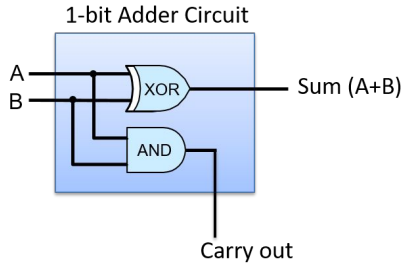


Storage



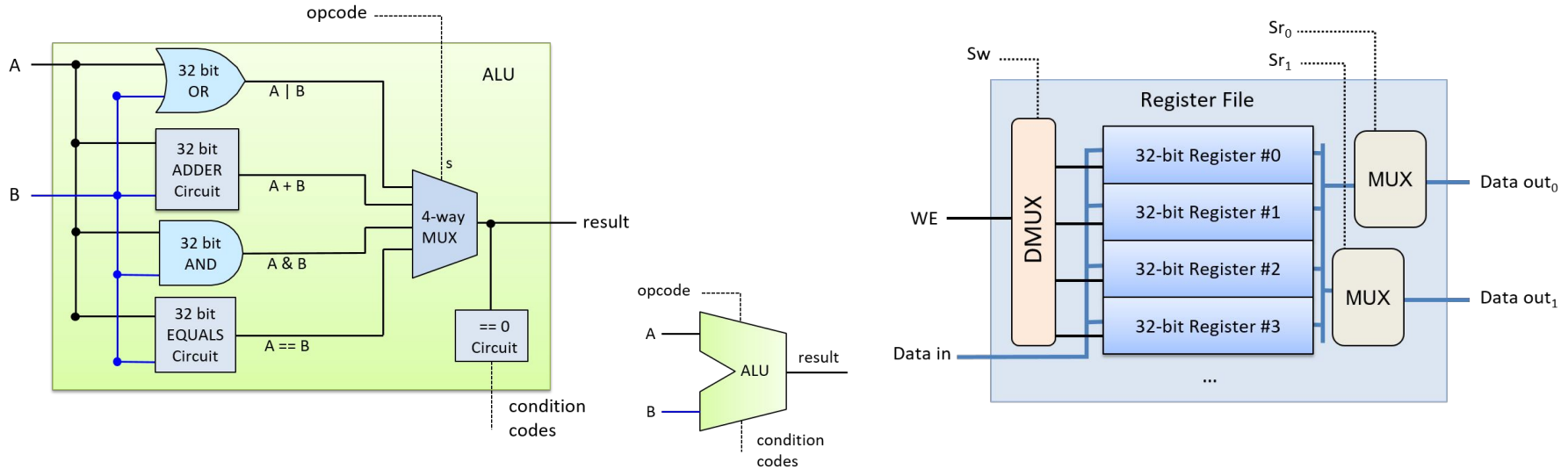
Abstraction and building up complexity

1-bit version of circuits is building block to create multi-bit versions, which in turn can be building blocks for larger units, ...



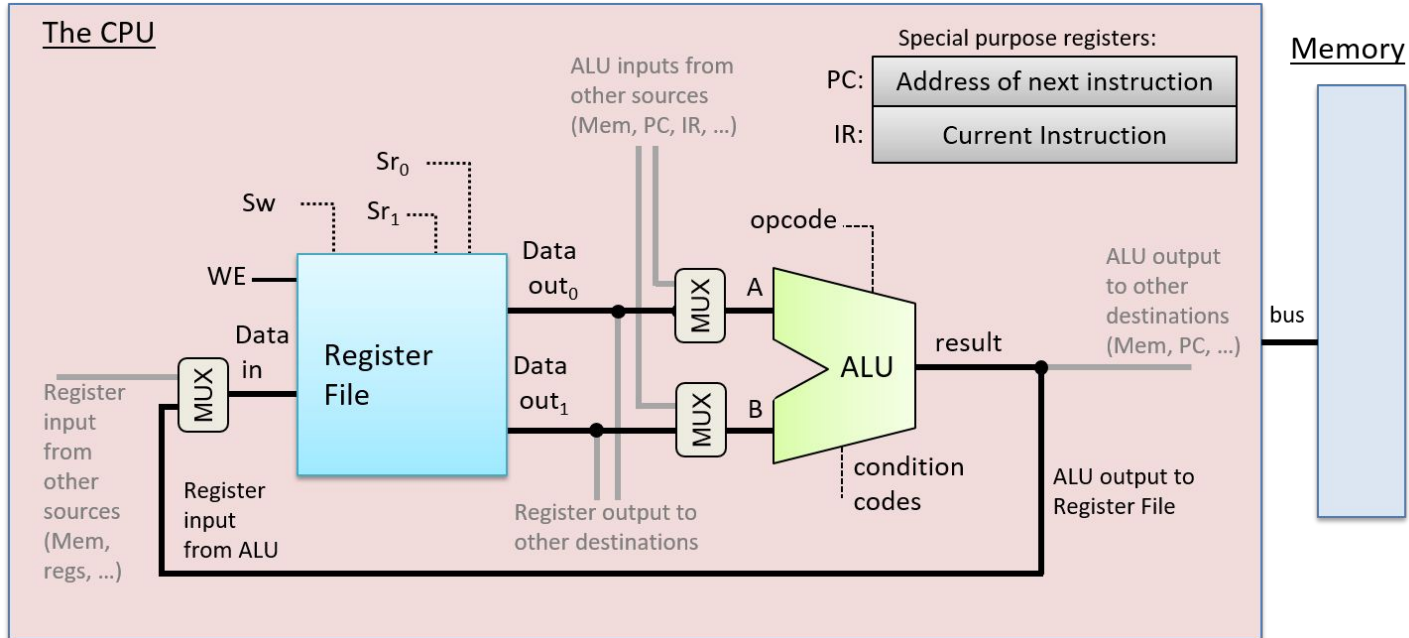
Build up large functional units

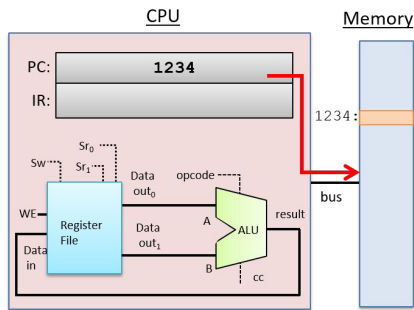
From simple arithmetic/logic, control, and storage circuits



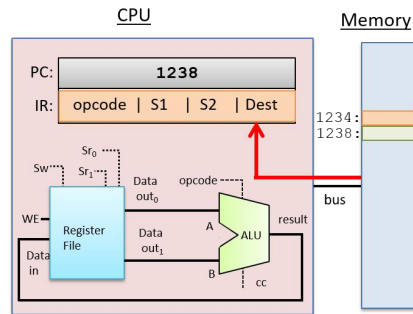
Put it all together:

Clock Driven Execution, IR, PC,
Step through 4 Stages of Execution (for instructions with all register operands)

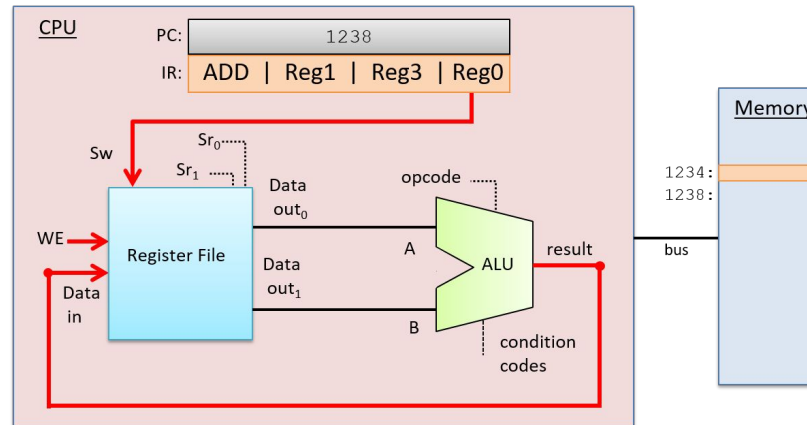
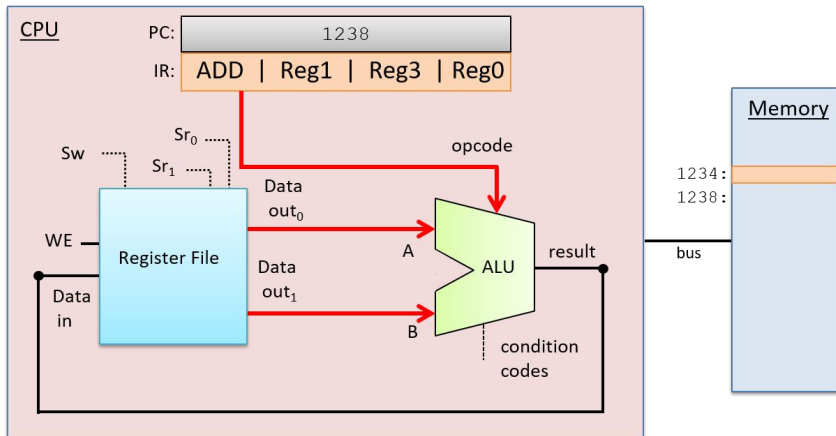
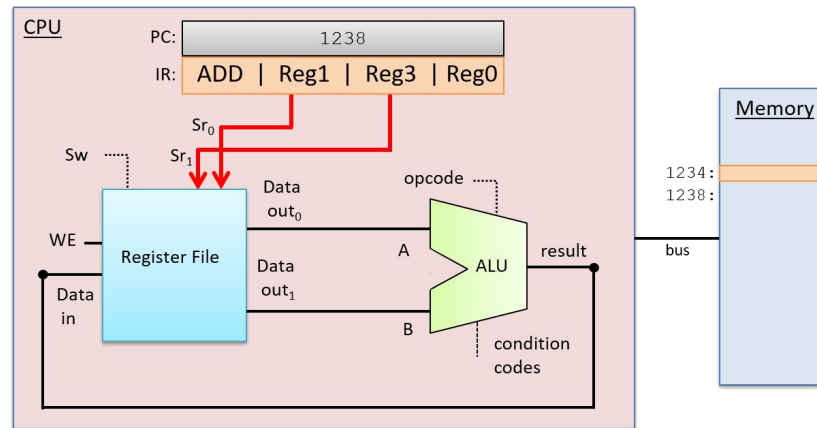




A. Issue read request to memory using the memory address in PC.



B. Store instruction data in IR and increment PC.



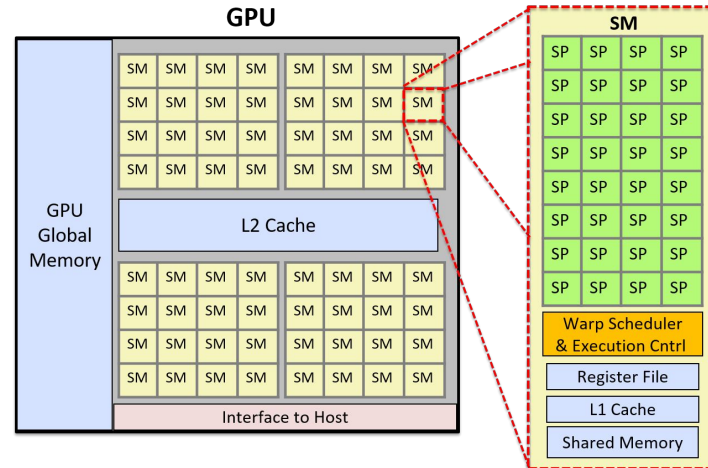
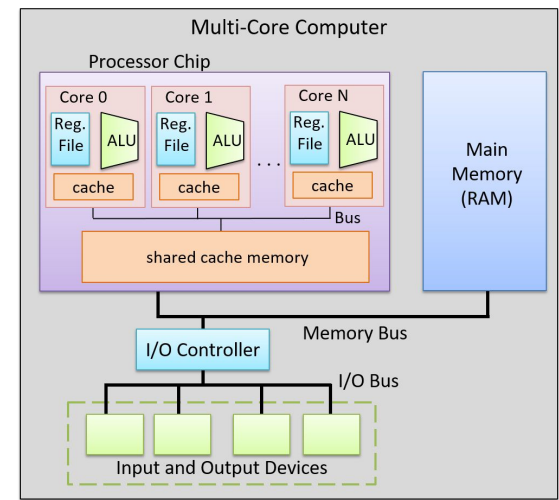
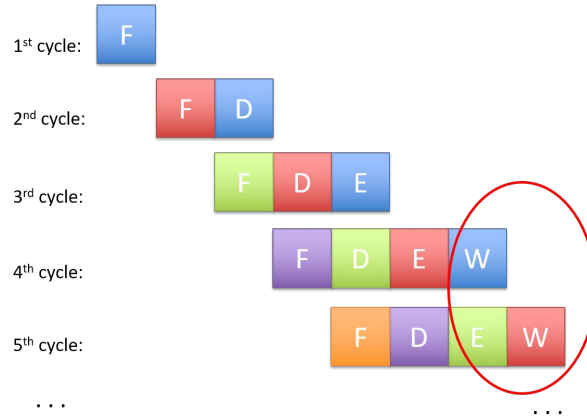
Parallel Architecture

- In more Detail

- Pipelining
- Multicore

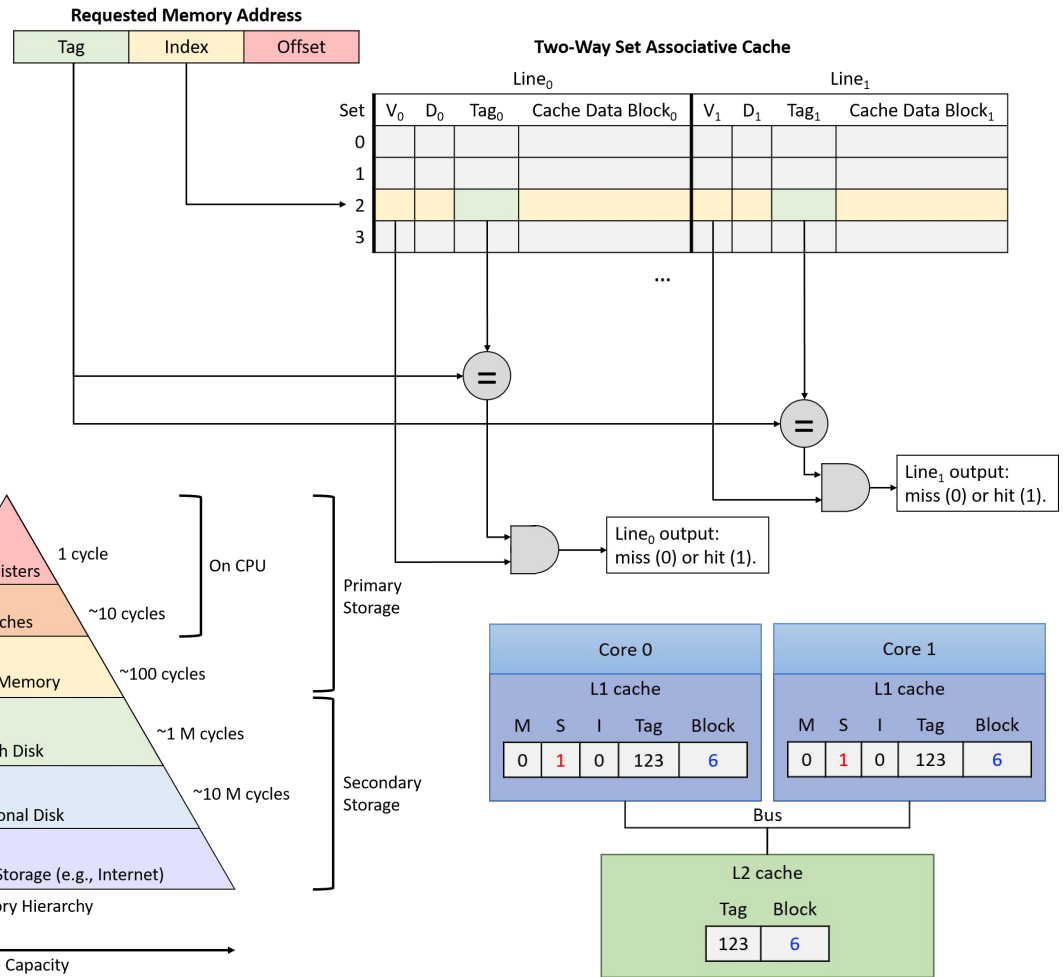
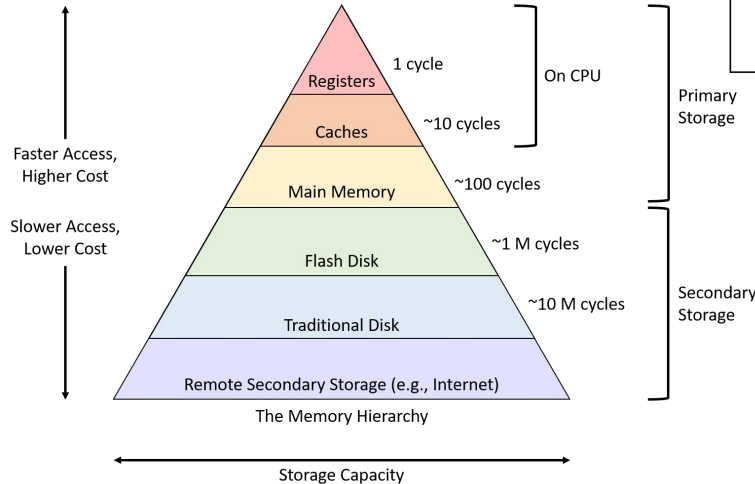
- Also high-level overview of others (chapt 5, 14, 15)

- ILP, Superscalar, Vector Processors
- Hardware Multithreading
- Accelerators, GPU as example
- Flynn's Taxonomy
- Moore's Law, Power Wall
- Performance metrics



Memory Hierarchy & Caching (Chapt. 11)

- Devices
- Locality
- Caching
- direct mapped
- set associative
- coherency
- MSI example
- false sharing (chapt 14)



Book Development: History of community help

External Reviewers of Every Chapter from Experts in our Field (mostly faculty):

- Volunteers, multiple for each chapter
- Strengthened content and presentation
- Helped ensure broad applicability of our textbook

2019-20: Early Adopters Program: Beta Version of our textbook (SIGCSE'20)

- Required textbook at 19 different institutions
- Small stipend (\$100) to faculty from SIGCSE Special Projects Grant
- Feedback on its use in different courses
- Helped further refine topic coverage and presentation

People are eager to volunteer for resource filling need, and free online

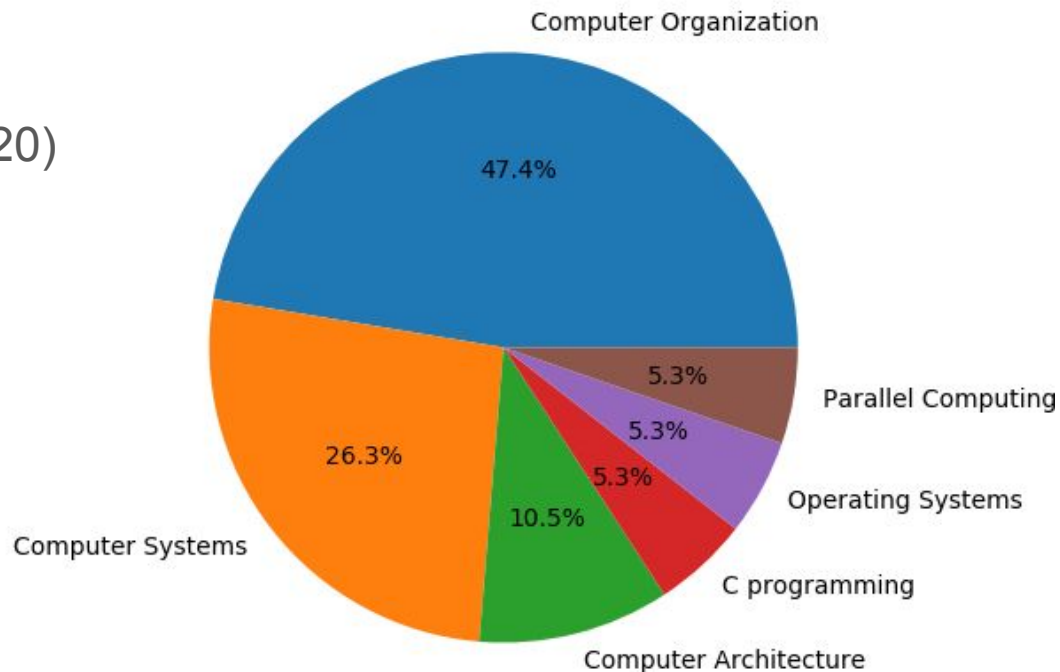
Book Use

We know of ~50 different institutions using it in their courses

19 Early Adopter Institutions (2019-20)

- Most as primary textbook in intro to computer systems or computer organization courses
- Some using in Architecture, OS, Parallel Computing as primary or supplementary textbook

2020 Survey of Early Adopters Types of Courses using *Dive into Systems* as a required text



Our Current Effort (NSF funded)

- **Primary: Adding Interactive Content to Dive into Systems**
 - Online format: ideal for adding other types of content to promote student learning
 - Develop interactive exercises for book chapters
 - Also adding videos of worked examples/solutions
- **Secondary: Developing Instructor Portal Content**
 - exercises, programming/lab assignments, links to example curricula using Dive into Systems, ...

Adding Interactive Exercises

- Seeking Exercise Developers from larger CS community
 - Use the expertise and help from our larger community!
 - Diversity of uses/ideas/school type/participants
 - NSF funding to provide stipends (\$1,000) to some, also volunteers
 - Groups develop interactive exercises for book chapters
- Students at our institutions
 - Develop tools, implement exercises in Runestone
- 4 Year plan for topic groups:
 - Year 1: 2022-23: C programming, Assembly Programming
 - Year 2: 2023-24: Binary, Memory Hierarchy & Caching
 - Year 3: 2024-25: OS, Shared Memory Parallel Computing
 - Year 4: 2025-26: Architecture

*All contributors
acknowledged
for their work!*

Our Current Interactive Tool Development

Tool Demos

[ASM Visualizer](#): assembly code tracing

["Ask me another"](#) new functionality added to Runestone*

*Runestone (by Brad Miller) is the tool we are using as our main tool, and interface to, our interactive exercises

ASM Visualizer

Welcome! You are using ASMVisualizer in function mode. In this mode you can write multiple functions to be called by our_main. Please type your assembly code below and click submit.

```
1 .text
   .globl  our_main
   .type   our_main, @function
```

```
our_main:
    push %rbp
    mov  %rsp, %rbp

    # Add your code for the our_main function here:
    mov  $10, %rax
    add  $30, %rax

    pop  %rbp
    retq

    .size  our_main, .-our_main
```

Submit

1. type in assembly code & submit

2. trace its execution: next/prev
show reg, stack, instr

Instructions

6	0x401117	push %rbp
7	0x401118	mov %rsp, %rbp
8		
9		# Add your code for the our_main function here:
10	0x40111B	mov \$10, %rax
11	0x401122	add \$30, %rax
12		
13	0x401126	pop %rbp
14	0x401127	retq
15		.size our main.

Autoscroll to current instruction

➡ line that just executed

➡ next line to execute

Step 4 of 6

First

Prev

Next

Last

Program Output

Stack Content

	Address	Value
RSP	0x1FFF000200	0x1FFF000210
RBP	0x1FFF000208	0x401110
	0x1FFF000210	0x401130

Autoscroll to stack pointer

Register Contents

Register	Value
RAX	0xA
RSP	0x1FFF000200
RBP	0x1FFF000200
RFLAG	0x44

Show more values on click

"Ask me another" question like the current one

cache organization, size and address bits

Trace through sequence of addresses, answer questions about effects on cache: direct mapped or 2-way set associative

Cache Organization: 2-Way Set Associative Address Length: 8 bits

address: 0b10011010
tag: 2 index: 4 offset: 2

block size (in bytes) = 4
number of lines = 32
number of sets = 16

Generate an Address Check me

Correct. Good job!
Correct. Good job!
Correct. Good job!

Activity: 2 Cache System (test_caching_info)

Cache Organization: 2-Way Set Associative Address Length: 8 bits

block size: 8 total number of lines: 8

Usage: Select a range of bits, and then click its corresponding button below.

address: 0b 1 1 1 1 1 1 0 0

Your current tag bits: 3 Your current index bits: 2 Your current offset bits: 3

Set to Tag Set to Index Set to Offset Reset selection

Generate an Address Check me

Correct. Good job!

Activity: 3 Cache Partition (test_caching_partition)

Index	V	D	Tag
0	0	0	
1	0	0	111
2	0	0	
3	0	0	
4	0	0	110
5	0	0	
6	1	0	101
7	1	0	010

Ref	Address	R/W
0	10111000	R
1	01011100	R
2	10111000	W

Hit?	Miss?	Index	V	D	Tag
<input type="radio"/>	<input checked="" type="radio"/>	6	1	0	101
<input type="radio"/>	<input checked="" type="radio"/>	7	1	0	010
<input type="radio"/>	<input type="radio"/>				

Generate another Check me

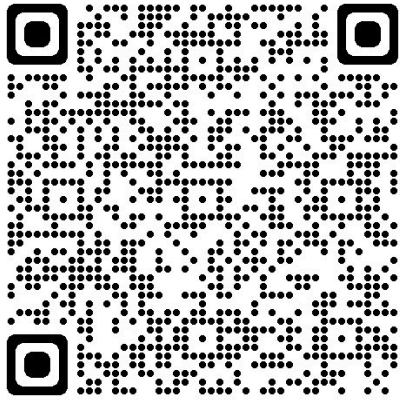
Correct. Good job!

Activity: 4 Cache Table (test_caching_table)

Interested in Participating?

- Join the *Dive into Systems* [mailing list](#) (off diveintosystems.org)
- Look for announcements posted to SIGCSE mailing list
- Can Sign-up now: <https://forms.gle/sHUnEsjSVWLptrMo8>
 - Link also available as a QR code (right).
 - we will send emails with yearly deadlines
- Timeline:
 - Year 2: 2023-24: [Binary Representation](#)
[Memory Hierarchy and Caching](#)
 - Year 3: 2024-25: OS, Shared Memory Parallel Computing
 - Year 4: 2025-26: Architecture





Do you use our book?
Please let us know!



Thank you!

Questions?



Interested in
participating
in our new effort?



<https://forms.gle/sHUnEsjSVWLptrMo8>

Read our book/mailling list: diveintosystems.org