

CS250: FOUNDATIONS OF COMPUTER SYSTEMS

[COMPUTER ARCHITECTURE]

The Stored Program Concept

A limited repertoire you say?

Well ... it's just like Lego

Only with a surfeit of pieces

Mix, match and combine in ways

That are infinite

Literally and metaphorically

All leading to a machine

With limits

constrained only by creativity

SHRIDEEP PALLICKARA

Computer Science

Colorado State University

COMPUTER SCIENCE DEPARTMENT



COLORADO STATE UNIVERSITY

1

Frequently asked questions from the previous class survey

- What do general purpose registers typically hold?
- How can programs function without knowing that there is a cache?
- If a program is updating variables that are currently in the cache, does it write to the cache and main memory?
- When something is evicted from the cache is it saved in main memory?
- When something is evicted, what if it is needed soon after?
- If cache hit rates are high, why do we even need a big main memory?



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

ARCHITECTURE

L12.2

2

Topics covered in this lecture

- Caching
 - Direct mapped
 - Associative
 - N-way associativity
- Stored Program Concept
- The von Neumann architecture



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

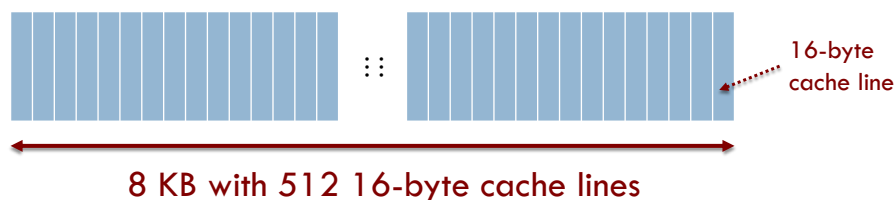
ARCHITECTURE

L12.3

3

Generally, if a cache line is n bytes long

- It will hold n bytes from main memory that fall on an **n -byte boundary**
- In our example of 16-byte cache lines, a cache line holds blocks of 16 bytes whose addresses fall on 16-byte boundaries in main memory
 - i.e., the least-significant 4 bits of the address of the first byte in the cache line are always 0



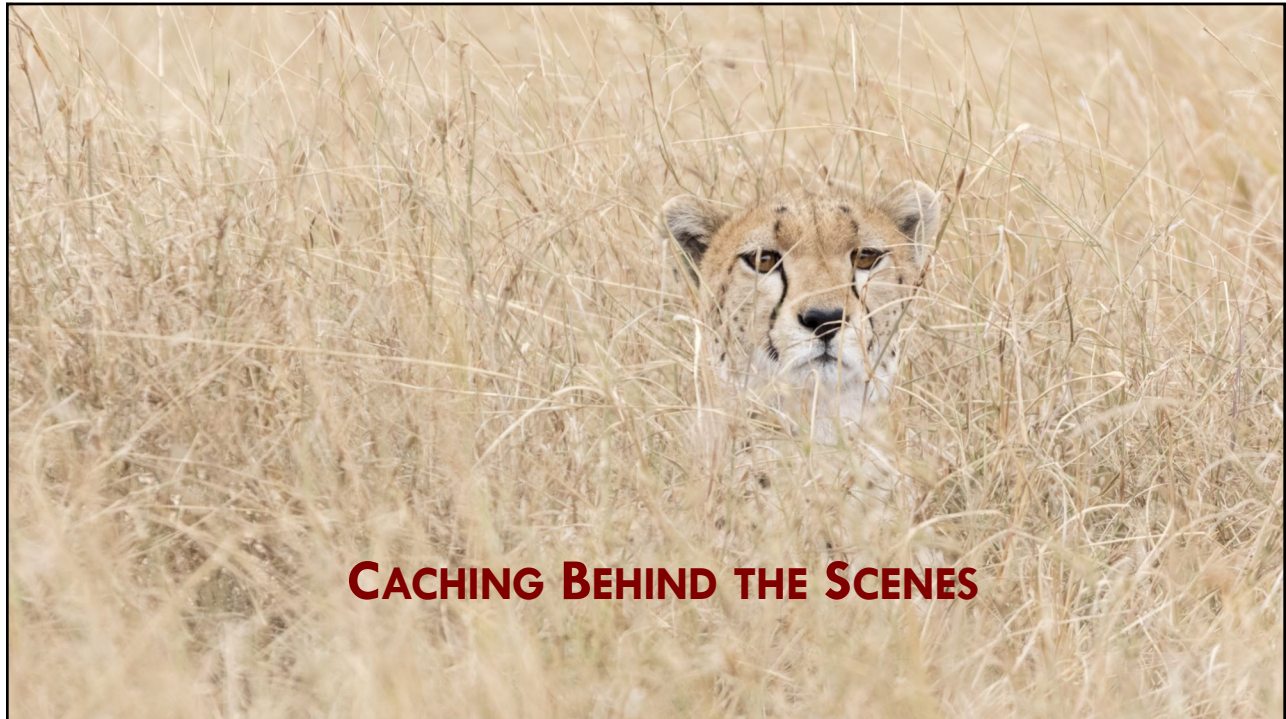
COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

ARCHITECTURE

L12.4


4



5

Types of caches

- Direct mapped caches
- Fully associative caches
- N-way associative caches

 **COLORADO STATE UNIVERSITY** Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT ARCHITECTURE L12.6

6

A **direct mapped cache** is also known as a one-way associative cache

- In a direct-mapped cache, a particular block of main memory is always loaded into—mapped to—the exact same cache line
- This mapping is **determined by a small number of bits** in the data block's memory address



COLORADO STATE UNIVERSITY

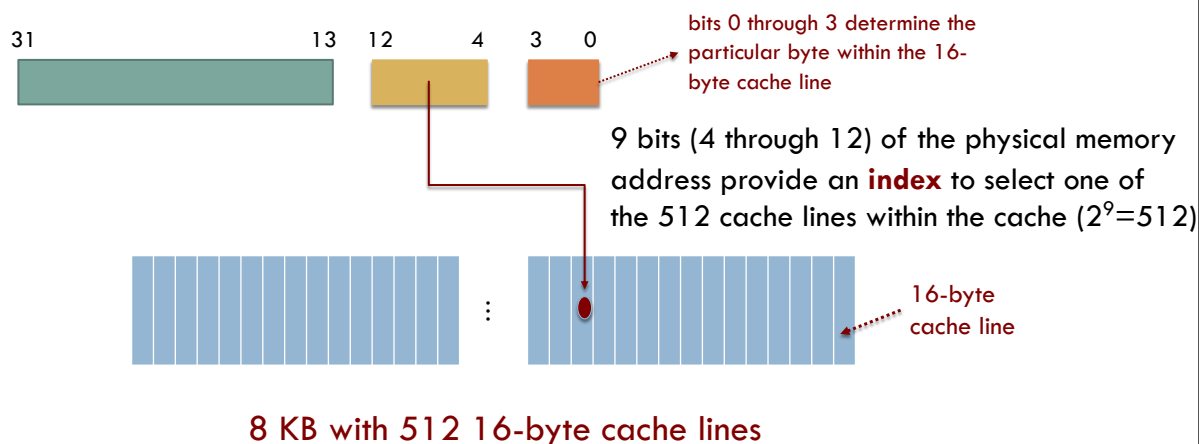
Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

ARCHITECTURE

L12.7

7

Direct-mapped Cache



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

ARCHITECTURE

L12.8

8

Problems with a direct mapped cache

- Two different memory addresses located on 8KB boundaries **cannot both appear simultaneously** in the cache
- How many such addresses exist in our 32-bit system?
 - 2^{19} 8KB blocks exist in our system
 - $2^{19} \cdot 512$ (2^9) blocks of 16-bytes (2^4) each
 - $2^{19} \cdot 2^9 \cdot 2^4 = 2^{32}$ (the size of the main memory in our example)



The ideal world: A fully **associative** cache

- The cache controller can place a block of bytes in *any one* of the cache lines present in the cache memory
- While this is the most flexible cache system, the **extra circuitry** to achieve full associativity *is expensive* and, worse, *can slow down* the memory subsystem
- Most L1 and L2 caches are not fully associative for this reason



Trade-off space

- A fully associative cache is too complex, too slow, and too expensive to implement
- But a direct-mapped cache is too inefficient



A compromise: the **n-way associative cache**

- In an n -way set associative cache, the cache is broken up into sets of n cache lines
- The CPU determines the **particular set to use** based on
 - ▣ Some subset of the memory address bits, just as in the direct-mapping scheme, and ...
 - ▣ The cache controller uses a fully associative mapping algorithm to determine which one of the n cache lines within the set to use



For example, an 8KB two-way set associative cache subsystem with 16-byte cache lines [1/2]

- Organizes the cache into **256** cache-line sets with **two cache** lines each
- Eight bits from the memory address determine which one of these 256 different sets will contain the data
 - $2^8 = 256$
- Once the cache-line set is determined, the cache controller maps the block of bytes to one of the two cache lines within the set



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

ARCHITECTURE

L12.13

13

For example, an 8KB two-way set associative cache subsystem with 16-byte cache lines [2/2]

- This means **two different memory addresses** located on 8KB boundaries (addresses having the same value in bits 4 through 11) can both appear simultaneously in the cache
- However, a **conflict** will occur if you attempt to access a **third** memory location at an address that is an even multiple of 8KB



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

ARCHITECTURE

L12.14

14

An 8 KB two-way set associative cache subsystem with 16-byte cache lines

bits 0 through 3 determine the particular byte within the 16-byte cache line


Eight bits (11 through 4) provide index to select one of 256 sets $2^8=256$

A cache line set comprising two cache lines

The cache controller chooses one of two different cache lines within the set $2^8=256$

16-byte cache line


8 KB with 2-way set associative cache with 256 sets of two (16-byte) cache lines each

 COLORADO STATE UNIVERSITY Professor: SHRIDEEP PALLICKARA COMPUTER SCIENCE DEPARTMENT ARCHITECTURE L12.15

15

What if we have a 4-way associative cache

- A four-way set associative cache puts four associative cache lines in each cache-line set
- In our example, 8KB cache, a four-way set associative caching scheme would have **128 cache-line sets** with **four cache lines** each
- This would allow the cache to maintain up to four different blocks of data without a conflict, each of which would map to the same cache line in a direct-mapped cache

 COLORADO STATE UNIVERSITY Professor: SHRIDEEP PALLICKARA COMPUTER SCIENCE DEPARTMENT ARCHITECTURE L12.16

16

2-/4-way set associative vs direct mapped

- A 2- or 4-way set associative cache is
 - ▣ Much better than a direct-mapped cache and
 - ▣ Considerably less complex than a fully associative cache



Can we keep increasing the number of lines in each cache-line set?

- The more cache lines we have in each cache-line set, the closer we come to creating a fully associative cache
 - ▣ With all the attendant problems of complexity and speed
- Most cache designs are direct-mapped, two-way set associative, or four-way set associative
 - ▣ The various members of the 80x86 family make use of all three



WHY (MAIN) MEMORY MATTERS ...



19

An analogy: You at a government office

- Some interactions can be completed using your IDs/cards (in your wallet), documents (in your backpack), and documents (at home)
 - ▣ Items can be retrieved from the wallet in 2 seconds
 - ▣ The bag needs to be searched, and it takes about 120 seconds to do so
 - ▣ The trip home and back will take 36,000 seconds (or 10 hours)
- Average time to complete transaction if your wallet suffices 95% of the time but the backpack comes into play 5% of time?
 - ▣ $0.95 * (\text{wallet_time}) + 0.05 * (\text{backpack_time})$
 - ▣ $0.95 * 2 + 0.05 * 120 = 7.9$ seconds



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

ARCHITECTURE

L12.20

20

An analogy: You at a government office

- Average time to complete transaction if your wallet suffices 95% of the time but the backpack comes into play 4% of time and you need to go home 1% of the time?
 - $0.95 * (\text{wallet_time}) + 0.04 * (\text{backpack_time}) + 0.01 (\text{home_trip})$
 - $0.95 * 2 + 0.04 * 120 + 0.01 * 36,000 = 505.9$ seconds



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

ARCHITECTURE

L12.21

21

Let's make it a little more real

- Cache access: 2 ns
- Main memory access: 50 ns
- Disk access: 8 milliseconds [8,000,000 ns]
- 97% cache and 3% main memory
 - $0.97 * 2 + 0.03 * 50 = 3.4$ ns
- 95% cache and 5% main memory
 - $0.95 * 2 + 0.05 * 50 = 4.4$ ns
- 95% cache, 4% main memory, and 1% disk
 - $0.95 * 2 + 0.04 * 50 + 0.01 * 8,000,000 = 80,003.9$ ns



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

ARCHITECTURE

L12.22

22



23

Etymology Tidbit: Program "execution"

- Execute
 - ▣ Verb
 - ▣ Carry out or put into affect a plan, order, or course of action
- Contrary to some lore
 - ▣ No bits are actually "killed" when a program "executes"



24



25

The Stored Program Concept

[1/2]

- Compared to all the machines around us, the most remarkable feature of the digital computer is its amazing **versatility**
- Here is a machine with a **finite and fixed hardware** that can perform an **infinite number of tasks**
 - From playing games to typesetting books to driving cars
- This remarkable versatility—a boon that we have come to take for granted—is the fruit of a brilliant early idea called the **stored program concept**



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

ARCHITECTURE

L12.26

26

The Stored Program Concept

[2/2]

- Formulated independently by several scientists and engineers in the **1930s**
- The stored program concept is still considered the **most profound invention** in, if not the very foundation of, modern computer science.



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

ARCHITECTURE

L12.27

27

The Stored Program Concept: Like many scientific breakthroughs, the basic idea is simple

- The computer is based on a **fixed hardware** platform capable of executing a fixed repertoire of simple instructions
- At the same time, these instructions can be **combined like building blocks**, yielding *arbitrarily* sophisticated programs



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

ARCHITECTURE

L12.28

28

Moreover, the logic of these programs is not embedded in the hardware

- As was customary in mechanical computers predating 1930
- Instead, the program's code is **temporarily stored** in the computer's memory, like data
 - Becoming what is known as **software**



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

ARCHITECTURE

L12.29

29

Since the computer's operation manifests itself to the user through the currently executing software ...

- The same hardware platform can be made to **behave completely differently** each time it is loaded **with a different program**



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

ARCHITECTURE

L12.30

30

The stored program concept in models of computing

- The stored program concept is the key element of both abstract and practical computer models
 - ▣ Notably the Turing machine (1936) and the von Neumann machine (1945)
- The Turing machine is an **abstract artifact** describing a deceptively simple computer
 - ▣ Is used mainly in *theoretical* computer science for analyzing the logical foundations of computation
- In contrast, the **von Neumann** machine is a **practical** model that informs the construction of almost all computer platforms today



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

ARCHITECTURE

L12.31

31

Make everything as simple as possible, but not simpler.
—Albert Einstein (1879–1955)

THE VON NEUMANN ARCHITECTURE

COMPUTER SCIENCE DEPARTMENT



COLORADO STATE UNIVERSITY

32

The von Neumann architecture

- The von Neumann architecture is based on a Central Processing Unit (CPU)
 - Interacting with a memory device
 - Receiving data from some input device, and
 - Emitting data to some output device
- At the heart of this architecture lies the stored program concept:
 - The computer's **memory stores** not only the **data** that the computer manipulates
 - But also, the **instructions** that tell the computer what to do



COLORADO STATE UNIVERSITY

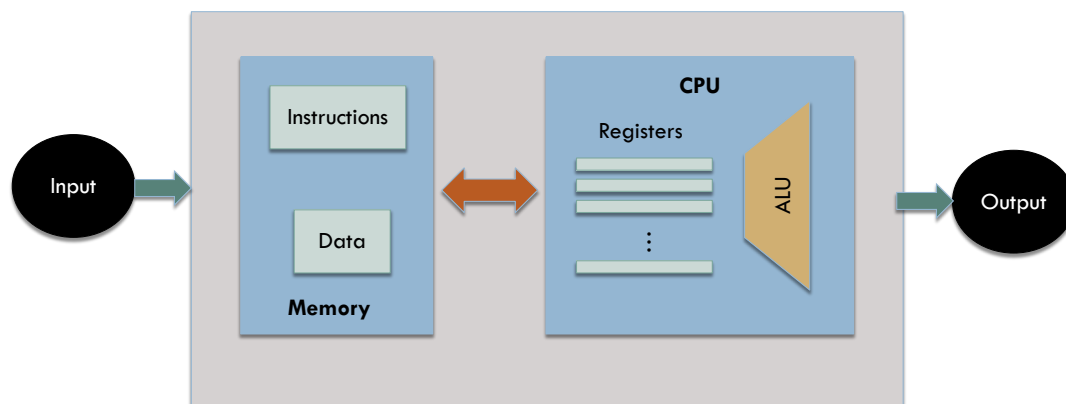
Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

ARCHITECTURE

L12.33

33

A generic von Neumann computer architecture



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

ARCHITECTURE

L12.34

34

The computer's Memory can be discussed from both physical and logical perspectives

- Physically, the memory is a **linear sequence of addressable bytes**, each having a unique address and a value
- Logically, this address space serves two purposes: **storing data and storing instructions**
 - Both the “instruction words” and the “data words” are implemented exactly the same way—as sequences of bits



Instruction Memory

[1/2]

- Before a high-level program can be executed on a target computer, it must first be **translated into the machine language** of the target computer
- Each high-level statement is translated into one or more low-level instructions
 - Written as binary values to a file called the binary, or **executable**, version of the program
- Before running a program:
 - **First load** its binary version from a mass storage device, and
 - **Serialize its instructions** into the computer's instruction memory



Instruction Memory

[2/2]

- From the pure focus of computer architecture, how a program is loaded into the computer's memory is considered an external issue
- What's important is that *when* the CPU is called upon to execute a program
 - The program's code will **already reside in the computer's memory**



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

ARCHITECTURE

L12.37

37

Data Memory

- High-level programs are designed to manipulate abstract artifacts like variables, arrays, and objects
- Yet at the hardware level, these data abstractions are realized by binary values stored in memory
- In particular, following translation to machine language
 - Abstract array processing and get/set operations on objects are reduced to reading and writing selected locations in memory



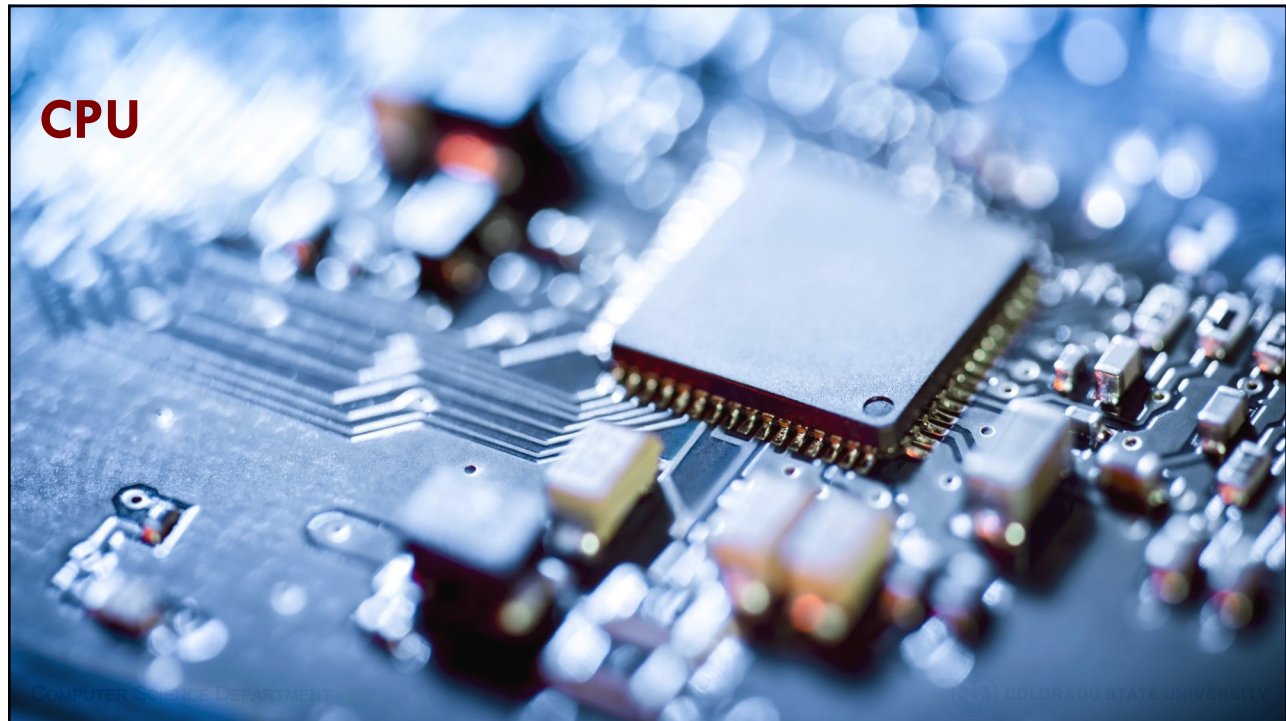
COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

ARCHITECTURE

L12.38

38



39

The CPU is the centerpiece of the computer's architecture

- The Central Processing Unit (CPU) is in charge of executing the instructions of the currently running program
- Each instruction tells the CPU which computation to perform, which registers to access, and which instruction to fetch and execute next
- The CPU executes these tasks using three main elements:
 - ▣ An Arithmetic Logic Unit (ALU)
 - ▣ A set of registers, and
 - ▣ A control unit



40

Arithmetic Logic Unit (ALU)

- The ALU chip is built to perform all the low-level arithmetic and logical operations featured by the computer
- A typical ALU can add two given values, compute their bitwise And, compare them for equality, and so on ...



How much functionality should be packed into the ALU is a design decision

- In general, any function not supported by the ALU can be *realized later*, using system software running on top of the hardware platform
- The **trade-off** is simple:
 - Hardware implementations are typically more efficient but result in more expensive hardware
 - While software implementations are inexpensive and less efficient



Why we use registers

[1/2]

- While performing computations, the CPU is often required to store **interim values** temporarily
- In theory, we could have stored these values in the RAM, but this would entail **long-distance trips** between the CPU and the RAM
 - The CPU and the RAM are two separate chips



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

ARCHITECTURE

L12.43

43

Why we use registers

[2/2]

- Long distance trips between the CPU and RAM result in **delays**
- These delays would frustrate the CPU-resident ALU, which is an ultra-fast combinational calculator
- The result will be a condition known as **starvation**
 - Happens when a fast processor depends on a **sluggish data store** for supplying its inputs and consuming its outputs



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

ARCHITECTURE

L12.44

44

To avert starvation and boost performance

- CPUs are equipped with a **small set** of high-speed (and relatively expensive) registers, acting as the processor's immediate memory
- These registers serve **various purposes**:
 - **Data registers** store interim values
 - **Address registers** store values that are used to address the RAM
 - The **program counter** stores the address of the instruction that should be fetched and executed next
 - The **instruction register** stores the current instruction



The contents of this slide-set are based on the following references

- Noam Nisan and Shimon Schocken. *The Elements of Computing Systems: Building a Modern Computer from First Principles*. 2nd Edition. ISBN-10/ ISBN-13: 0262539802 / 978-0262539807. MIT Press. [Chapter 5]
- Jonathan E. Steinhart. *The Secret Life of Programs: Understand Computers -- Craft Better Code*. ISBN-10/ ISBN-13 : 1593279701/ 978-1593279707. No Starch Press. [Chapter 5]
- Randall Hyde. *Write Great Code, Volume 1, 2nd Edition: Understanding the Machine* 2nd Edition. ASIN: B07VSC1K8Z. No Starch Press. 2020. [Chapter 11]

