CPSC-240 Computer Organization and Assembly Language

Chapter 2

Architecture Overview

Instructor: Yitsen Ku, Ph.D.
Department of Computer Science,
California State University, Fullerton, USA

CALIFORNIA STATE UNIVERSITY
FULLERTON™

Outline

- Architecture Overview
- Data Storage Sizes
- Central Processing Unit
- Main Memory
- Memory Layout
- Memory Hierarchy

Architecture Overview

Architecture Overview

- The basic components of a computer include a Central Processing Unit (CPU), Primary Storage or Random Access Memory (RAM), Secondary Storage, Input/Output devices (e.g., screen, keyboard, mouse), and an interconnection referred to as the Bus.
- Programs and data are typically stored on secondary storage (e.g., disk drive or solid state drive). When a program is executed, it must be copied from secondary storage into the primary storage or main memory (RAM). The CPU executes the program from primary storage or RAM.

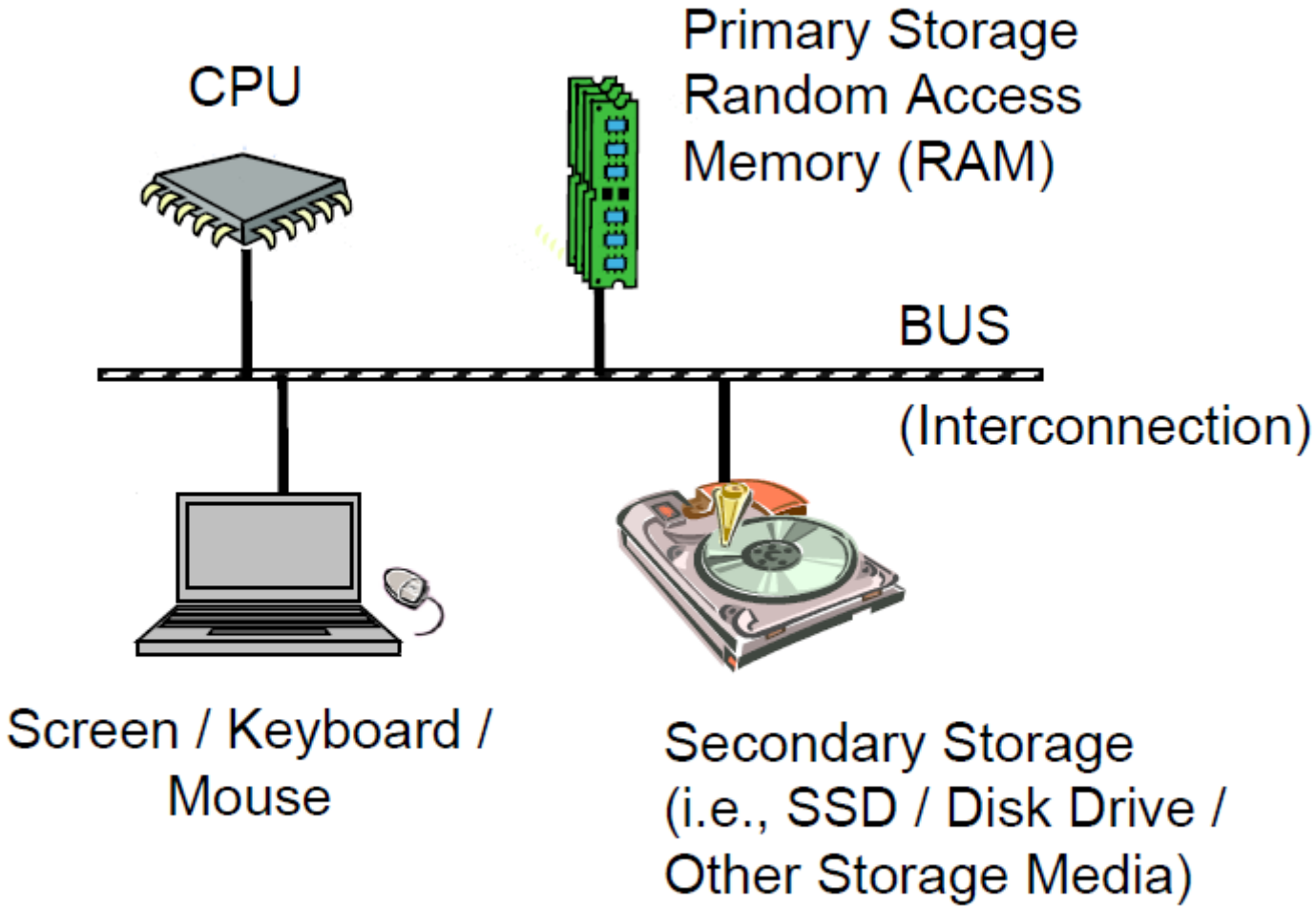


Illustration 1: Computer Architecture

Data Storage Sizes

x86-64 Data Storage Sizes

- The x86-64 architecture supports a specific set of data storage size elements, all based on powers of two. The supported storage sizes are as follows:

Storage	Size (bits)	Size (bytes)
Byte	8-bits	1 byte
Word	16-bits	2 bytes
Double-word	32-bits	4 bytes
Quadword	64-bits	8 bytes
Double quadword	128-bits	16 bytes

C/C++ Data Storage Sizes

C/C++ Declaration	Storage	Size (bits)	Size (bytes)
char	Byte	8-bits	1 byte
short	Word	16-bits	2 bytes
int	Double-word	32-bits	4 bytes
unsigned int	Double-word	32-bits	4 bytes
long[5]	Quadword	64-bits	8 bytes
long long	Quadword	64-bits	8 bytes
char *	Quadword	64-bits	8 bytes
int *	Quadword	64-bits	8 bytes
float	Double-word	32-bits	4 bytes
double	Quadword	64-bits	8 bytes

Central Processing Unit

Central Processing Unit

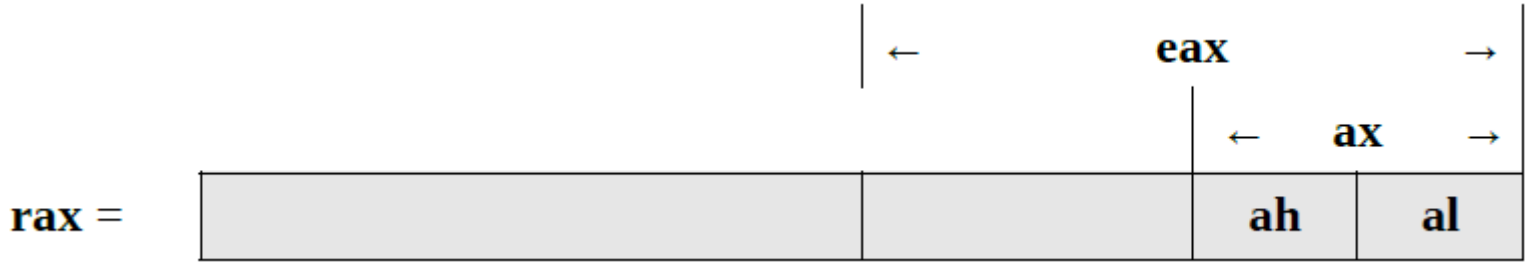
- The Central Processing Unit (CPU) is typically referred to as the "brains" of the computer since that is where the actual calculations are performed.
- The CPU chip includes a number of functional units, including the Arithmetic Logic Unit (ALU). It should be noted that the internal design of a modern processor is quite complex.

CPU Registers

- In order to support the ALU, processor registers and cache memory are also included “on the chip”.
- A CPU register, or just register, is a temporary storage or working location built into the CPU itself (separate from memory). Computations are typically performed by the CPU using registers.

General Purpose Registers (GPRs)

- When using data element sizes less than 64-bits (i.e., 32-bit, 16-bit, or 8-bit), the lower portion of the register can be accessed by using a different register name as shown in the following table.
- For example, when accessing the lower portions of the 64-bit **rax** register, the layout is as follows:



64-bit register	Lowest 32-bits	Lowest 16-bits	Lowest 8-bits
rax	eax	ax	al
rbx	ebx	bx	bl
rcx	ecx	cx	cl
rdx	edx	dx	dl
rsi	esi	si	sil
rdi	edi	di	dil
rbp	ebp	bp	bpl
rsp	esp	sp	spl
r8	r8d	r8w	r8b
r9	r9d	r9w	r9b
r10	r10d	r10w	r10b
r11	r11d	r11w	r11b
r12	r12d	r12w	r12b
r13	r13d	r13w	r13b
r14	r14d	r14w	r14b
r15	r15d	r15w	r15b

General Purpose Registers (GPRs)

- The ability to access portions of the register means that, if the quadword **rax** register is set to $50{,}000{,}000{,}000_{10}$ (fifty billion), the **rax** register would contain the following value in hex.

rax = 0000 000B A43B 7400

- If a subsequent operation sets the word **ax** register to $50{,}000_{10}$ (fifty thousand, which is $C350_{16}$), the **rax** register would contain the following value in hex.

rax = 0000 000B A43B C350

- If a subsequent operation sets the byte sized **al** register to 50_{10} (fifty, which is 32_{16}), the **rax** register would contain the following value in hex.

rax = 0000 000B A43B C332

Stack Pointer Register (RSP)

- One of the CPU registers, **rsp**, is used to point to the current top of the stack. The **rsp** register should not be used for data or other uses. Additional information regarding the stack and stack operations is provided in Chapter 9, Process Stack.

Base Pointer Register (RBP)

- One of the CPU registers, **rbp**, is used as a base pointer during function calls. The **rbp** register should not be used for data or other uses. Additional information regarding the functions and function calls is provided in Chapter 12, Functions.

Instruction Pointer Register (RIP)

- In addition to the GPRs, there is a special register, **rip**, which is used by the CPU to point to the ***next instruction to be executed***.
- Specifically, since the **rip** points to the next instruction, that means the instruction being pointed to by **rip**, and shown in the debugger, has not yet been executed. This is an important distinction which can be confusing when reviewing code in a debugger.

Flag Register (rFlags)

- The flag register, **rFlags**, is used for status and CPU control information. The **rFlag** register is updated by the CPU after each instruction and not directly accessible by programs. This register stores status information about the instruction that was just executed. Of the 64-bits in the **rFlag** register, many are reserved for future use.

Flag Register (rFlags)

Name	Symbol	Bit	Use
Carry	CF	0	Used to indicate if the previous operation resulted in a carry.
Parity	PF	2	Used to indicate if the last byte has an even number of 1's (i.e., even parity).
Adjust	AF	4	Used to support Binary Coded Decimal operations.
Zero	ZF	6	Used to indicate if the previous operation resulted in a zero result.
Sign	SF	7	Used to indicate if the result of the previous operation resulted in a 1 in the most significant bit (indicating negative in the context of signed data).
Direction	DF	10	Used to specify the direction (increment or decrement) for some string operations.
Overflow	OF	11	Used to indicate if the previous operation resulted in an overflow.

XMM Registers

- There are a set of dedicated registers used to support 64-bit and 32-bit floating-point operations and Single Instruction Multiple Data (SIMD) instructions. The SIMD instructions allow a single instruction to be applied simultaneously to multiple data items. Used effectively, this can result in a significant performance increase. Typical applications include some graphics processing and digital signal processing.
- Additionally, the XMM registers are used to support the Streaming SIMD Extensions (SSE). The SSE instructions are out of the scope of this text.

XMM Registers

128-bit Registers
xmm0
xmm1
xmm2
xmm3
xmm4
xmm5
xmm6
xmm7
xmm8
xmm9
xmm10
xmm11
xmm12
xmm13
xmm14
xmm15

Cache Memory

- Cache memory is a small subset of the primary storage or RAM located in the CPU chip. If a memory location is accessed, a copy of the value is placed in the cache.
- Subsequent accesses to that memory location that occur in quick succession are retrieved from the cache location (internal to the CPU chip).
- A cache hit occurs when the requested data can be found in a cache, while a cache miss occurs when it cannot. Cache hits are served by reading data from the cache, which is faster than reading from main memory.

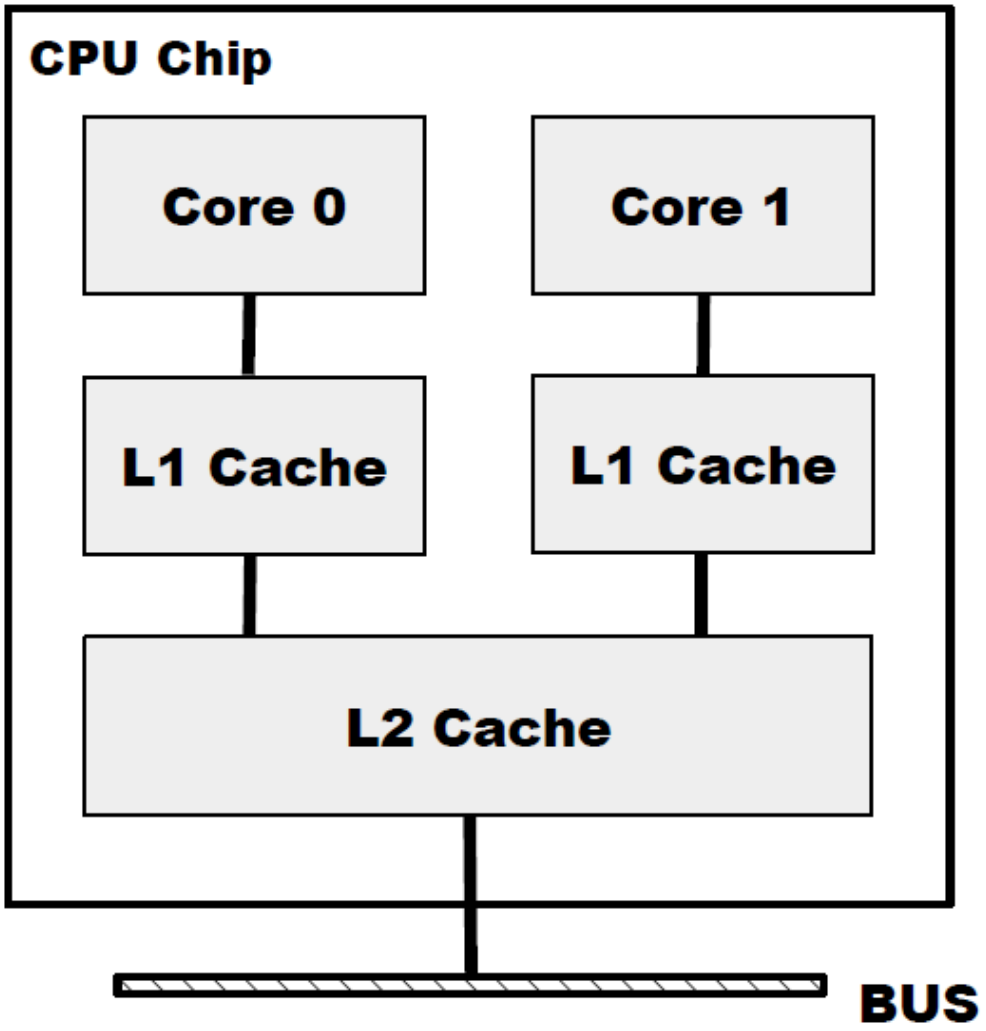
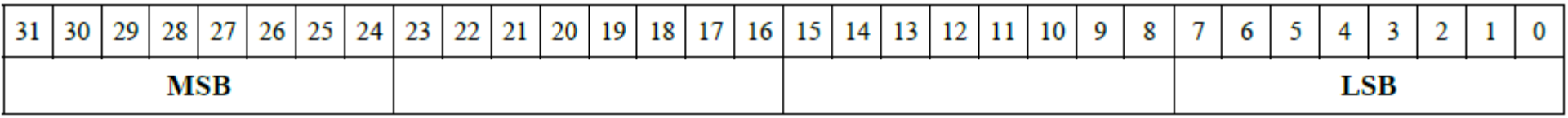


Illustration 2: CPU Block Diagram

Main Memory

Main Memory

- Memory can be viewed as a series of bytes, one after another. That is, memory is *byte addressable*. This means each memory address holds one byte of information. To store a double-word, four bytes are required which use four memory addresses.
- The Least Significant Byte (LSB) is stored in the lowest memory address. The Most Significant Byte (MSB) is stored in the highest memory location.
- For a double-word (32-bits), the MSB and LSB are allocated as shown below.



Little-Endian Data Layout

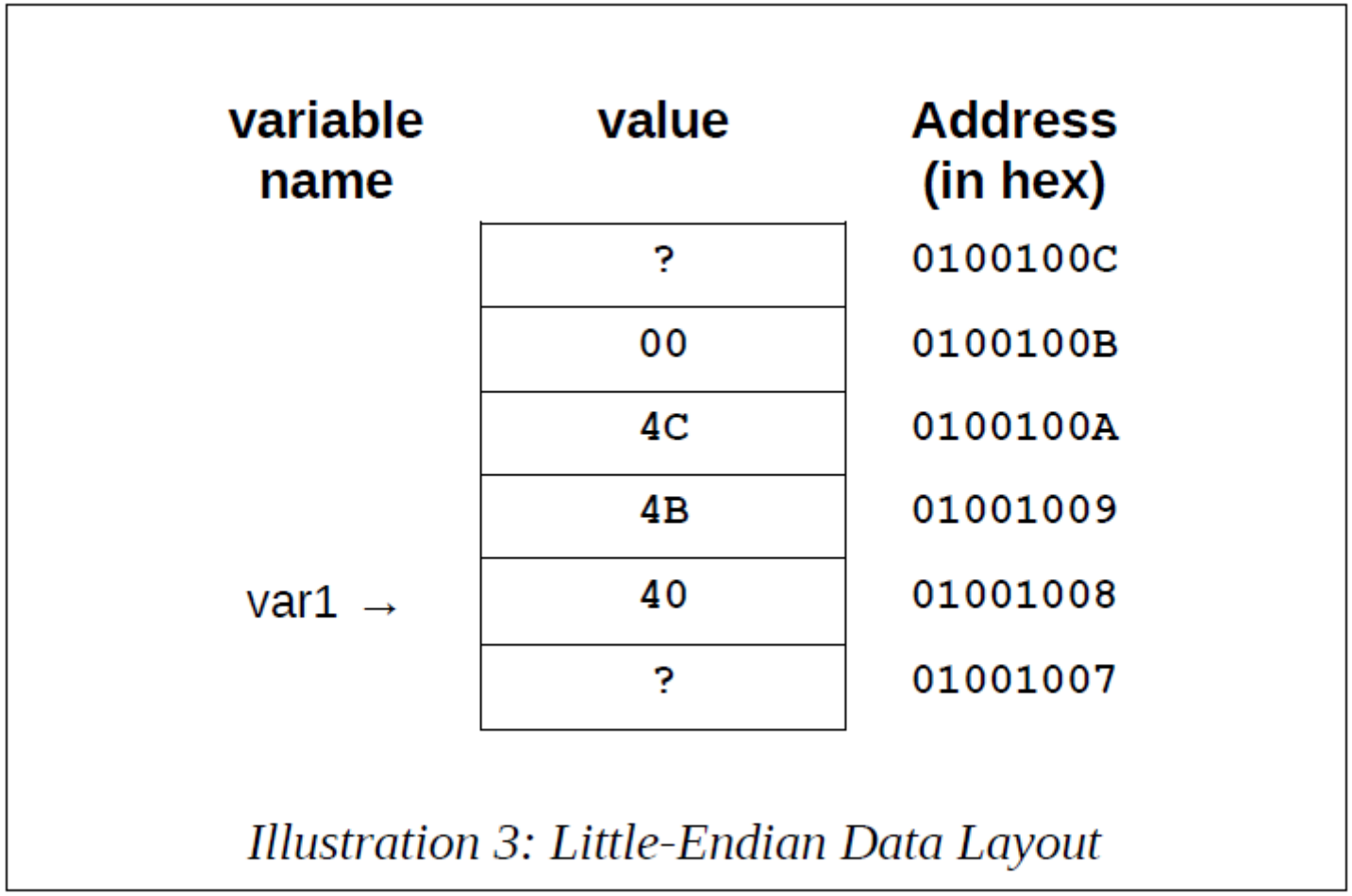


Illustration 3: Little-Endian Data Layout

Memory Layout

General Memory Layout

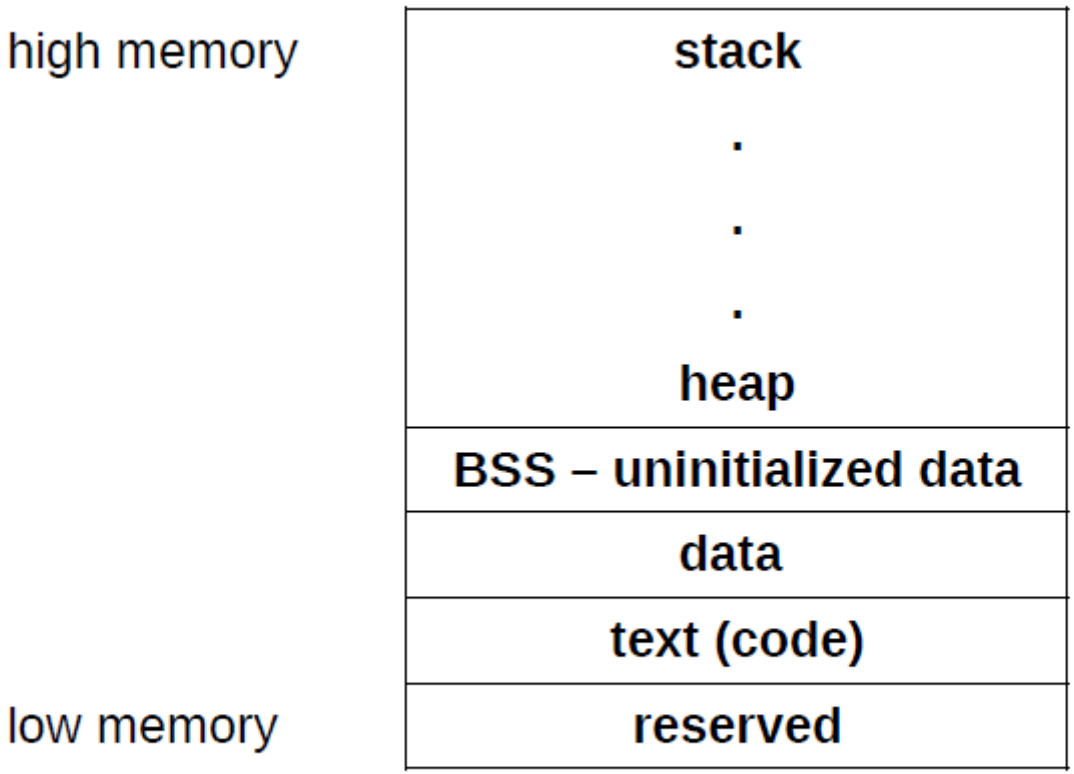


Illustration 4: General Memory Layout

General Memory Layout

- The reserved section is not available to user programs.
- The text (or code) section is where the machine language (i.e., the 1's and 0's that represent the code) is stored.
- The data section is where the initialized data is stored.
- The uninitialized data section, typically called BSS section, is where declared variables that have not been provided an initial value are stored.
- The heap is where dynamically allocated data will be stored (if requested). The stack starts in high memory and grows downward.

Memory Hierarchy

Memory Hierarchy Overview

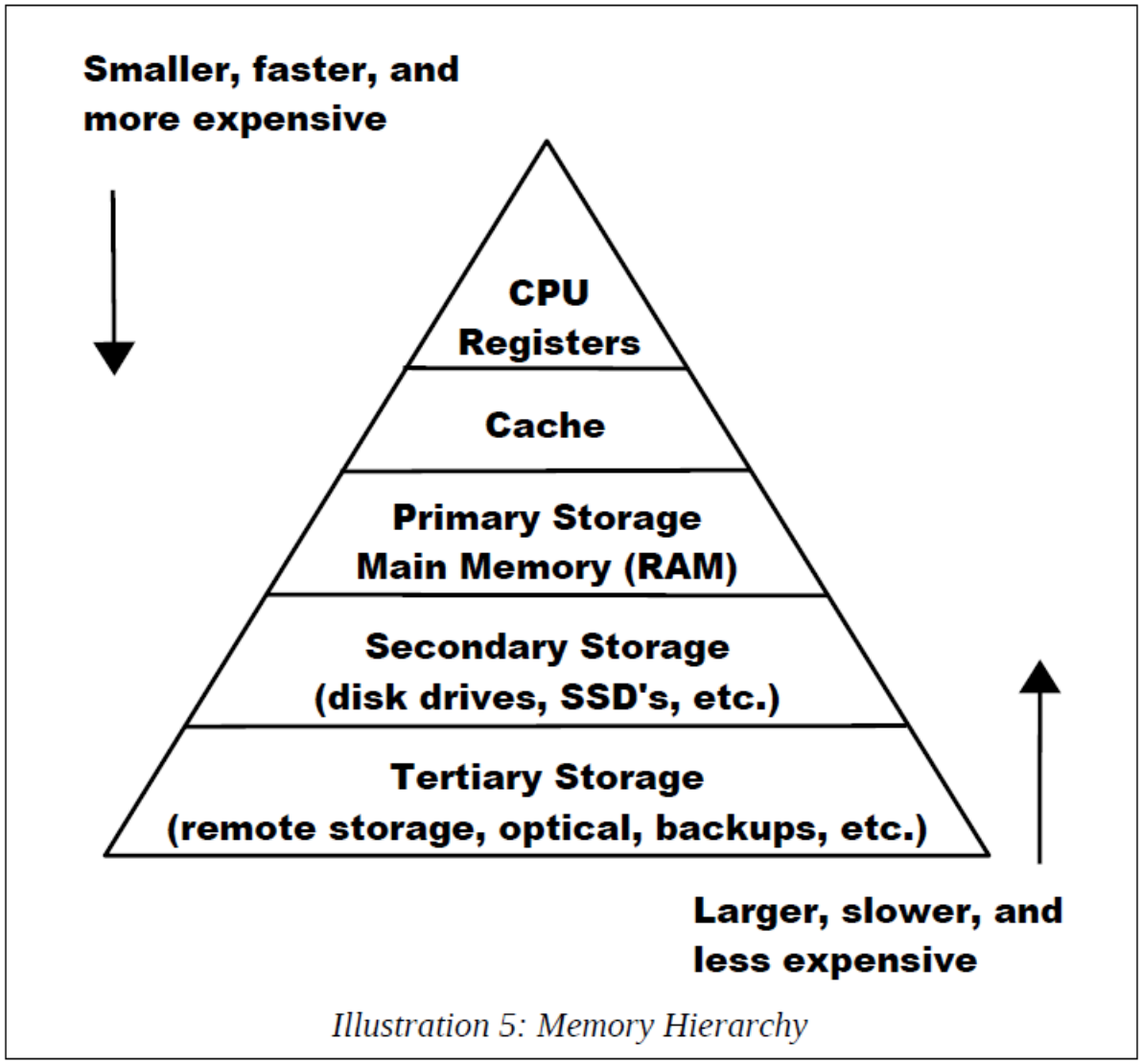


Illustration 5: Memory Hierarchy

General Memory Layout

- In general terms, faster memory is more expensive and slower memory blocks are less expensive. The CPU registers are small, fast, and expensive. Secondary storage devices such as disk drives and Solid State Drives (SSD's) are larger, slower, and less expensive.
- Some typical performance and size characteristics are as follows:

Memory Unit	Example Size	Typical Speed
Registers	16, 64-bit registers	~1 nanoseconds[13]
Cache Memory	4 - 8+ Megabytes[14] (L1 and L2)	~5-60 nanoseconds
Primary Storage (i.e., main memory)	2 – 32+ Gigabytes[15]	~100-150 nanoseconds
Secondary Storage (i.e., disk, SSD's, etc.)	500 Gigabytes – 4+ Terabytes[16]	~3-15 milliseconds[17]

End of Chapter 2