CSCE 312: Computer Organization

David Kebo Houngninou

COVID-19: Protect Yourself & Others

CDC recommendations

Get Vaccinated
Wear a mask
Stay 6 feet away from others
Avoid crowds and poorly ventilated spaces
Wash your hands often
Cover coughs and sneezes
Clean and disinfect
Monitor your health daily
https://www.cdc.gov/coronavirus/2019-ncov/prevent-getting-sick/prevention.html

Outline

1. What is computer organization?
2. The big picture
3. Computer organization vs. computer architecture
4. Hardware organization
5. Processor: execution flow
6. Memory hierarchy
7. OS

What is computer organization?

Computer organization describes the implementation, the arrangement, and the connection of the hardware components in the functional units of a computer at a low level.

What is computer architecture?

Computer architecture is a specification detailing how a set of software and hardware technology standards interact to form a computer system.

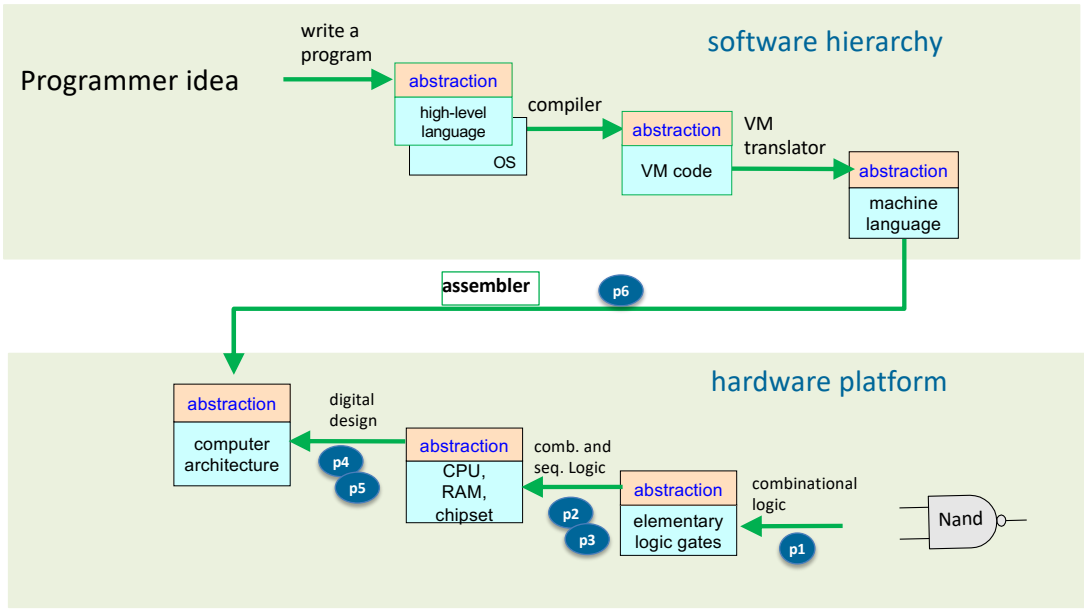
Computer organization vs. Computer architecture

Computer organization	Computer architecture
• Implementation of the visible attributes • Cache • Physical registers • Gates • Flip flops • Buses and interconnects • Low-level design	• Set of attributes visible to the programmer • Instruction set • Visible registers • Main memory • Addressing modes • Data types

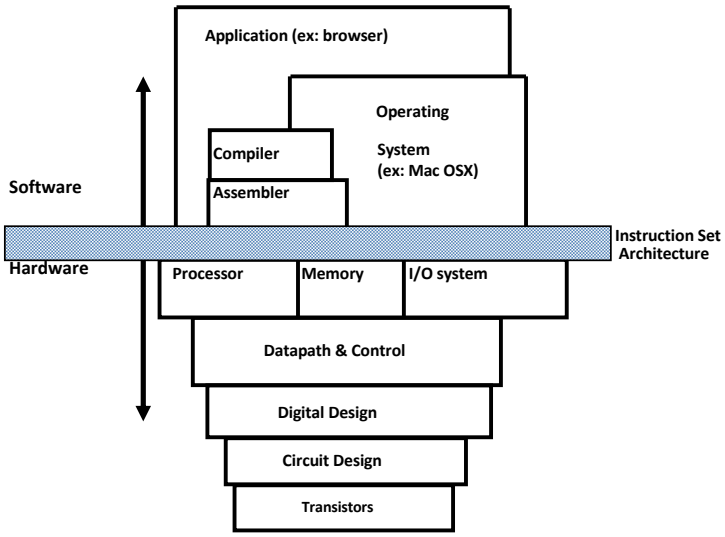
Computer organization vs. Computer architecture

- The computer architecture exposes features of a computer that the programmer needs to know in order to use it.
- The underlying details constitute a computer organization.
- Organization details are important to the designer of the computer, but not all are needed to be visible to the programmer.

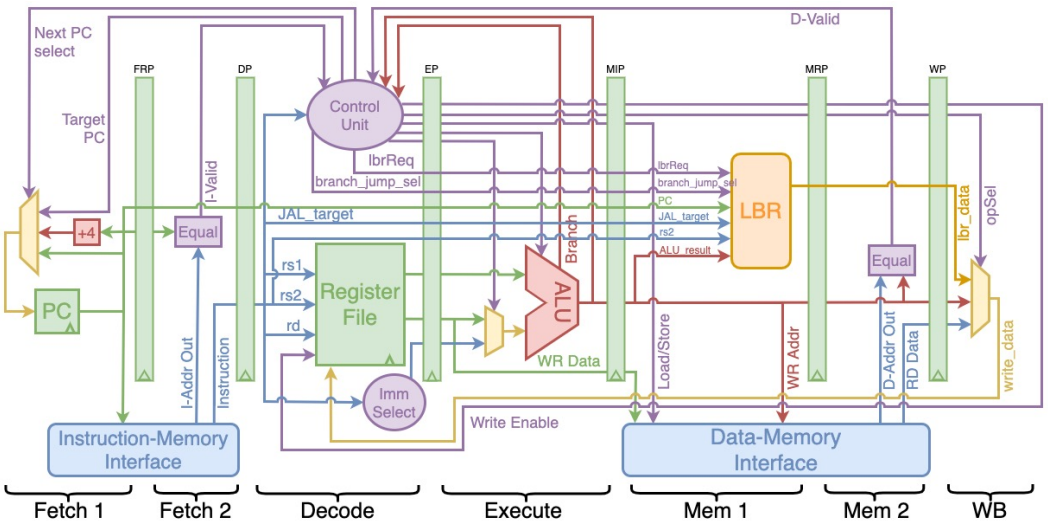
The Big Picture



Conventional Machine Structures

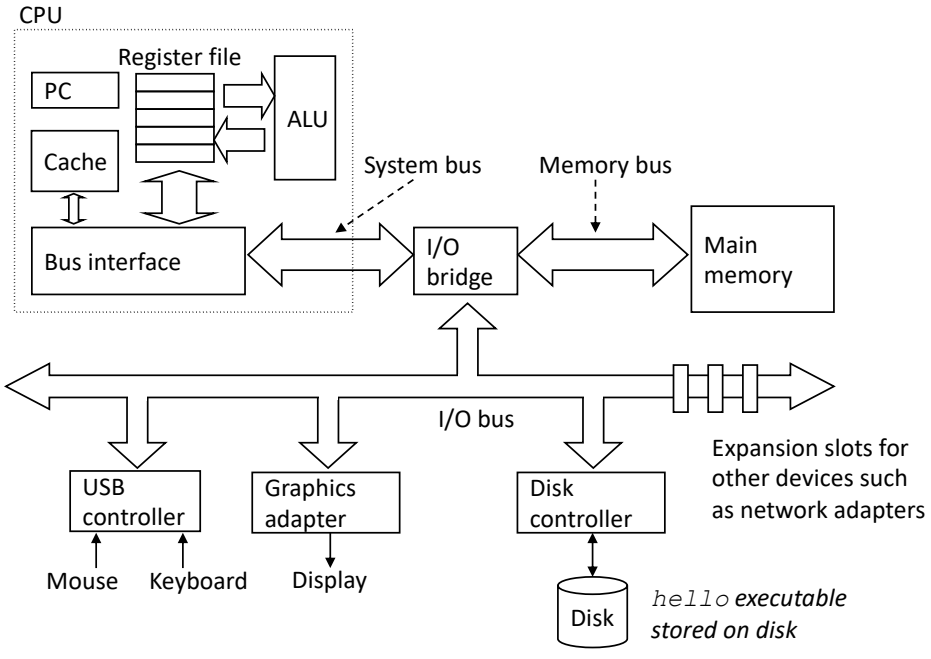


Example



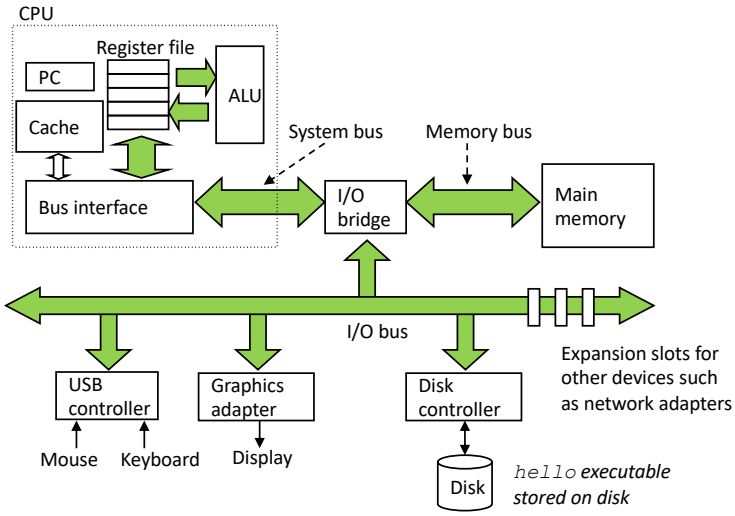
Tour of a computer system

Hardware organization



Buses

A bus is a collection of electrical wires that
connects computer components and transfers data between them.



Buses

Buses transfer fixed-size chunks of bytes know as words.

The number of bytes in a word (word size) depends on the system.

e.g.

- Intel Pentium Pro: 4 Bytes word size (32 bits bus)
- Intel Core i7: 8 Bytes word size (64 bits bus)

Types of system buses:

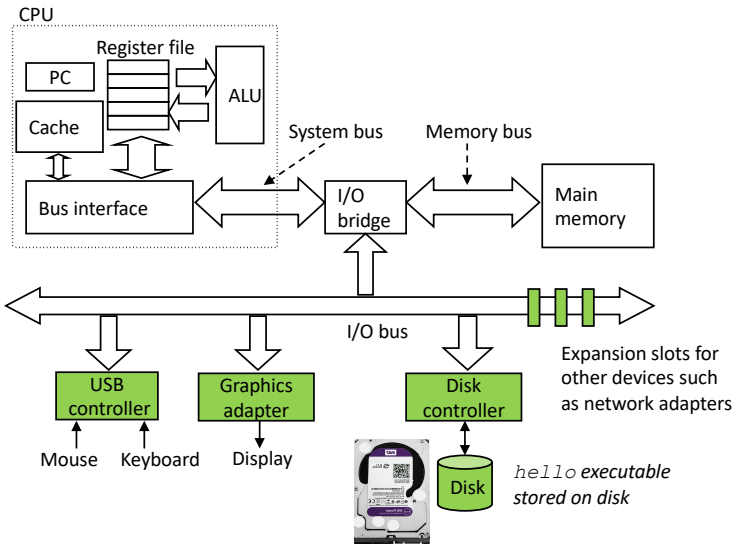
Data bus: moves data between the processor, memory and I/O devices

Address bus: moves addresses (the location of data in memory)

Control bus: moves control signals/commands between devices

I/O Devices

An I/O device is the system's connection to the user or internal devices.



I/O Devices

Inputs: signals received by the system from the user or other components.

Outputs: signals sent from the system to the user or other components.

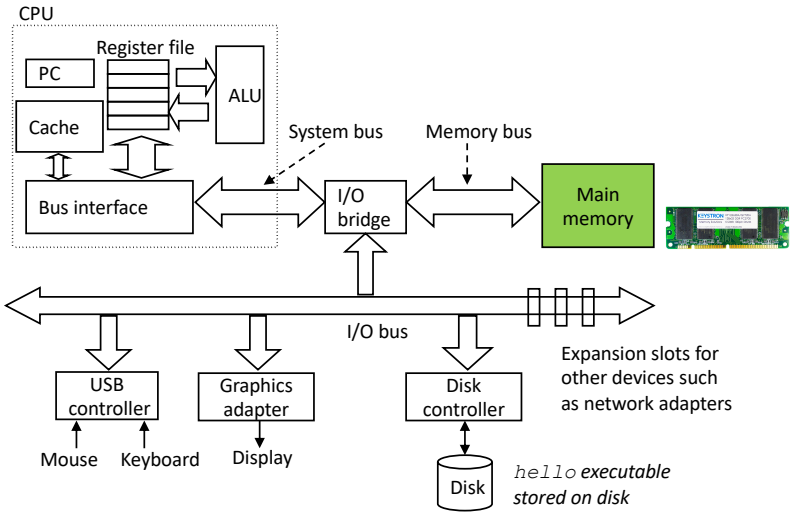
e.g.

- Input devices: keyboard, mouse, trackpad, camera, microphone, disk, network card.
- Output devices: display, printer, speaker, disk, network card.

I/O devices are connected to the I/O bus by a controller or an adapter.

Main memory

The main memory is a storage device that holds code and data while the processor is executing a program.



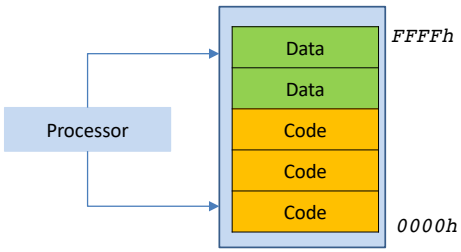
Main memory

Main memory (RAM) is organized as a linear array of bytes. Each array byte has its own address or index.

Main memory is volatile and made out of DRAM chips.

Code and data may share the same memory space.

e.g. of main memory DDR SDRAM



Examples

1. In the computer system diagram, list ALL the memory storage devices?
2. In the computer system diagram, where is the OS stored?
3. In the computer system diagram, where are running programs stored?
4. What type of memory is used to make main memory?
5. What type of memory is used to make the cache?

Examples

1. In the computer system diagram, list ALL the memory storage devices?

   Cache, Registers, RAM, Disk
2. In the computer system diagram, where is the OS stored?

   Disk
3. In the computer system diagram, where are running programs stored?

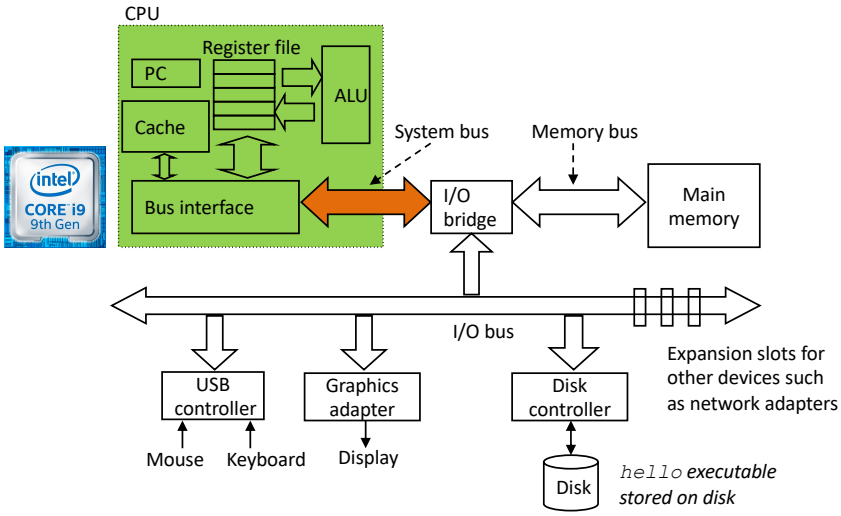
   RAM
4. What type of memory is used to make main memory?

   DRAM
5. What type of memory is used to make the cache?

   SRAM

Processor

The processor is the computational unit that executes the instructions stored in main memory.



Processor

The processor core includes registers, an Arithmetic Logic Unit and controllers.

The processor operates according to an instruction execution model: the Instruction Set Architecture (ISA).

e.g. of processors:

- Desktops / Laptops: Intel Core i9
- Gaming: AMD Radeon RX 5700 XT
- Lightweight netbooks: Intel Atom x7-z8700

e.g. of ISAs:

Intel x86, RISC-V, ARM

Processor: execution flow

The processor performs a series of steps to execute instructions:

1. The program counter (PC) points to an instruction in memory.
2. The processor reads / fetches the instruction from memory.
3. The processor decodes the bits in the instruction.
4. The processor executes the operation of the instruction.
5. The processor updates the PC to point to the next instruction.

Processor: instruction types

The processor can perform different types of instructions:

- Load: copy data from the main memory into a register.
- Store: copy data from a register to the main memory.
- Operate: copy the content of two registers to the ALU, perform an operation on the two words, and store the result in a register.
- Jump: load the PC with a new address.

Examples

What is the type (load, store, operate, jump) of each of these instructions?

```
1. ADD r1, r0, r2
2. SUB r2, r3, #10
3. LDR R1, =0x12345678
4. MUL r0, r1, r2
5. LDR r2, [r0]
6. STR r2, [r1]
7. B routine_1
8. AND r9,r2,#0xFF00
```


Examples

What is the type (load, store, operate, jump) of each of these instructions?

```
1. ADD r1, r0, r2 operate
2. SUB r2, r3, #10 operate
3. LDR R1, =0x12345678 load
4. MUL r0, r1, r2 operate
5. LDR r2, [r0] load
6. STR r2, [r1] store
7. B routine_1 jump
8. AND r9,r2,#0xFF00 operate
```

Writing your 1st program

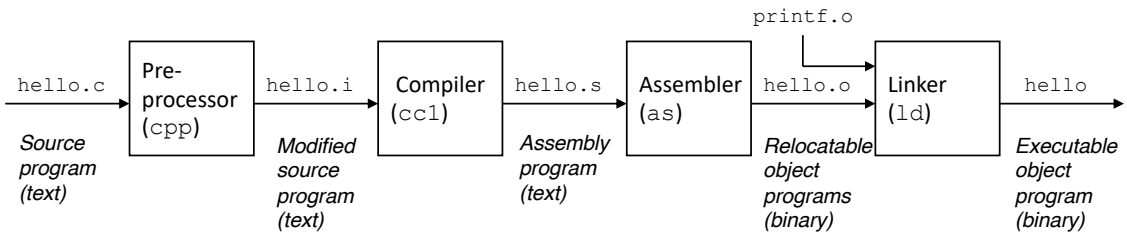
```
/* $begin hello */
#include <stdio.h>

int main()
{
    printf("hello, world\n");
    return 0;
}
/* $end hello */
```

- The hello program displays the phrase "hello, world" to the screen.
- This program is written in the C programming language.
- The computer stores the hello.c program in a file as a sequence of bytes.

Compiling your 1st program

```
linux > gcc –o hello hello.c
```



The GCC compiler reads the file hello.c and translates the file into an executable file hello

Compilation system

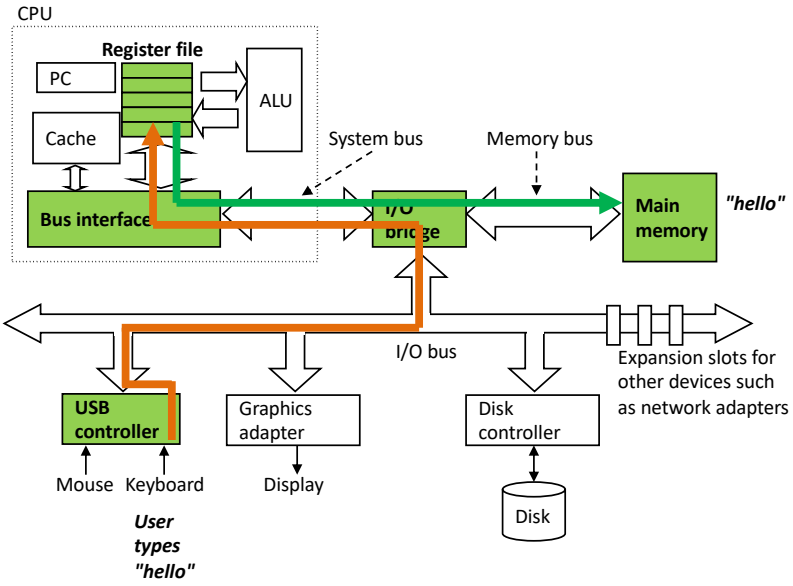
1. Preprocessor: prepares the hello.c file for compilation by including the library files that the program requires.
2. Compiler: translates the text file hello.i into the assembly language file hello.s.
3. Assembler: translates the assembly language file hello.s into an object file hello.o.
4. Linker: takes the object files generated by the compiler and combines them into one executable object file.
5. Loader: loads the program machine code in main memory.

Executing your 1st program

```
linux > gcc -o hello hello.c
linux > ./hello
hello, world
```

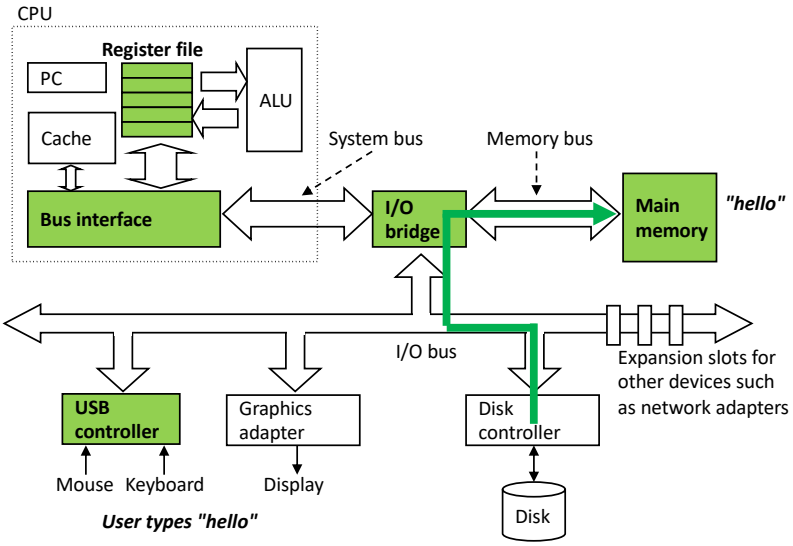
1. After compilation, the hello.c program is translated into an executable file that is stored on the computer disk.
2. The command ./hello runs the executable file by loading the program from the disk to the main memory (Direct Memory Access)
3. The processor fetches the program from the main memory, decodes, and executes it.
4. The hello program prints its message to the screen and terminates.

Executing your 1st program



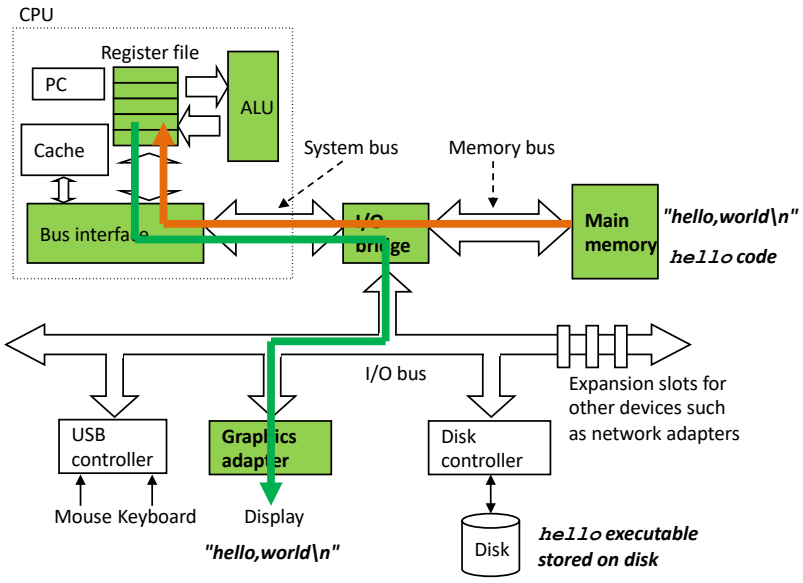
Reading the `hello` command from the keyboard

Executing your 1st program



Loading the executable from the disk into main memory

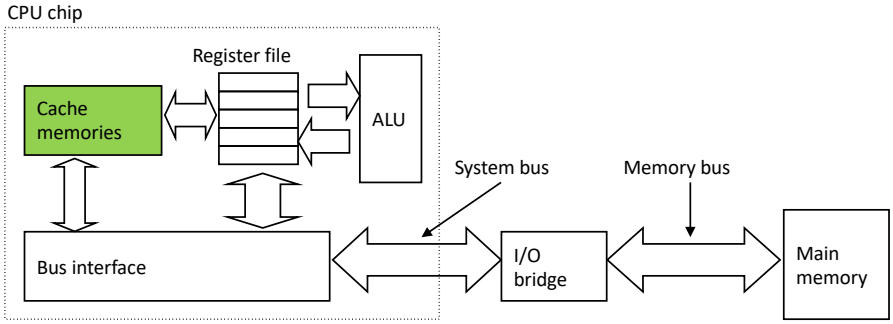
Executing your 1st program



Writing the output string from memory to the display

Cache memory

Cache memory is a small, temporary, volatile memory that provides high-speed access to the processor and stores frequently used code and data.



Cache memory

Cache is made of SRAM (Static Random Access Memory)

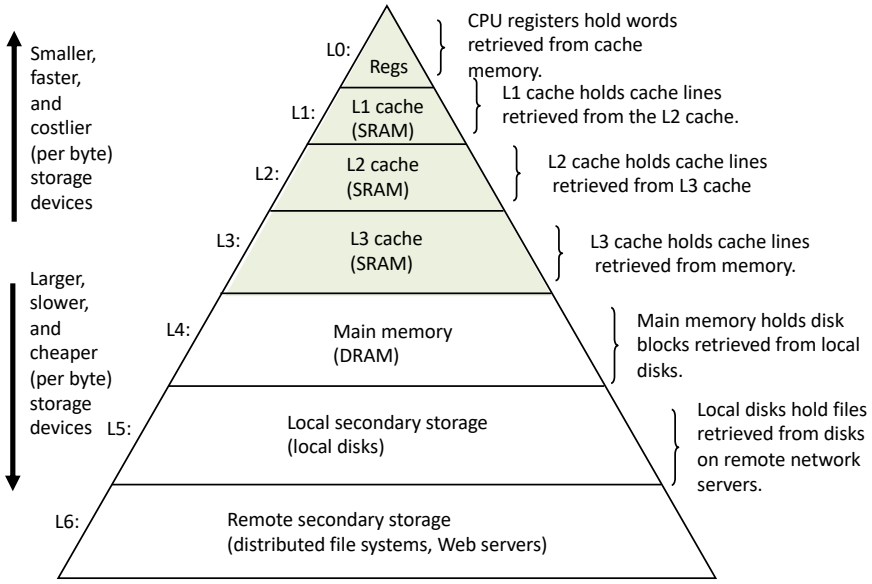
Physically it is integrated in the motherboard and embedded in the processor or main memory (RAM).

Cache has different hierarchies called levels: L1, L2, L3, L4, etc.

Cache brings performance to memory access by using the principle of locality.

Locality is the tendency of a processor to access the same set of memory locations frequently over a short period.

Memory hierarchy



An example of memory hierarchy

The operating system

The operating system (OS) is a software that manages the computer hardware and software resources.

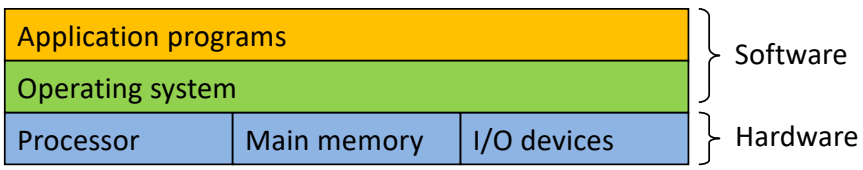
The operating system

The operating system (OS) is interposed between the application program and the hardware.

Modern OS support multitasking: multiple processes can run simultaneously.

A process is an abstraction of a running program.

A process consists of multiple execution units called threads.



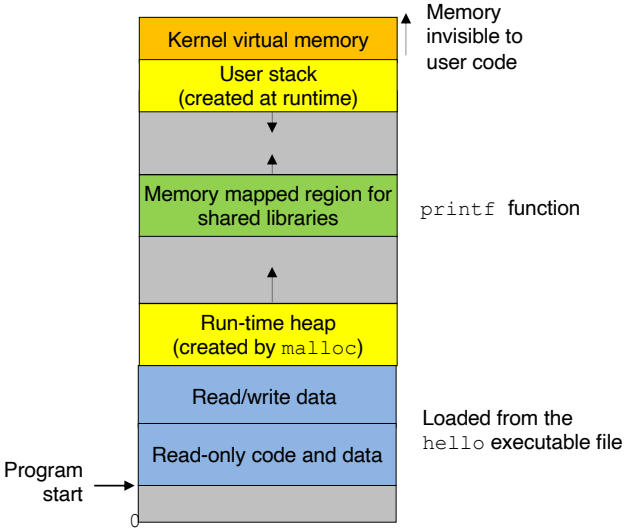
Virtual memory

Virtual memory is an abstraction that provides each process with the illusion that it has exclusive use of the main memory.

A virtual address space is a set of ranges of virtual addresses that the operating system makes available to a process.

Virtual memory

Each range of virtual addresses has a specific purpose



Process virtual address space

End