Informatics

Computer Operation Principles

Claudio Sartori
Department of Computer Science and Engineering
claudio.sartori@unibo.it
https://www.unibo.it/sitoweb/claudio.sartori/

Learning Objectives

- Explain what a software stack represents and how it is used
- Describe how the Fetch/Execute Cycle works, listing the five steps
- Understand the function of the memory, control unit, arithmetic/logic unit (ALU), input unit and output unit, and a program counter
- Discuss the purpose of an operating system
- Explain the purpose of a compiler
- Describe how large tasks are performed with simple instructions
- Explain why integration and photolithography are important in integrated circuits

Computer Overview

- Computers are used throughout your day
  - we are continually using computation
- They are laptops, cameras, tablets, iPods, desktops, GPS navigators, TV's and all the other electronic devices that we use

Cast of Characters

- **Processor**: follows the program's instructions
- **Operating System**: program that performs common operations, and makes your computer a useful device
- **Software**: programs
- **Instructions**: tell the processor what to do
- **Fetch/Execute Cycle**: executes the instructions
- **Memory**: stores the data
- **Hardware**: the physical parts of the computer

Software

- When you get a new app it is really a long sequence of bits
  - A team of programmers created the bits but did not type them one by one
  - The bits were produced from a program as discussed later in the chapter
- The program is a series of instructions expressing an algorithm
  - Lines are sparse, with a few symbols each
  - Some regular English words name available facilities
  - Sometime youSeeWordsRunTogether

Software

- Program's meaning may not usually be clear without specialized training
- Even so, you might be able to guess that this instruction in a splash image display program

```
if (this.Opacity < 1.0) {
    this.Opacity += 0.02;
}
```

  will make something more opaque if it's at least partially transparent
- The smallest typo can create a bug

Software Layers

- System software is arranged in layers
- Each layer uses services from the layer below
- Each layer provides services to the layer above
  - Shared effort: new programs reuse existing work
  - Once written, a feature can be used in many applications.
  - Consistent behavior
  - To fix a bug or make an improvement, only one place to change

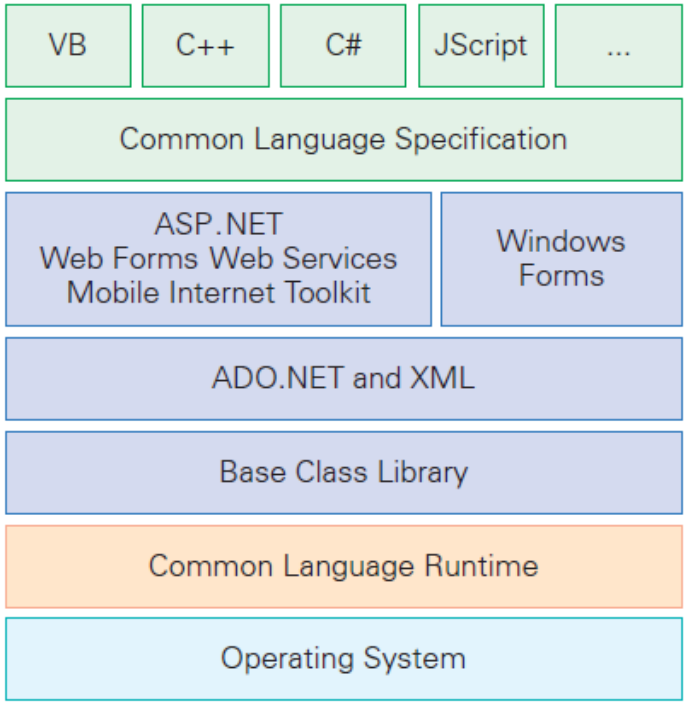


Figure 9.2 Software stack for Windows.NET.

Instruction Execution Engine

- A computer constantly executes instructions
- It does this by cycling through a series of operations
- Series is called: Fetch/Execute Cycle
  - Get the next instruction
  - Figure out what to do
  - Gather the data needed to do it
  - Do it
  - Save the result, and
  - Repeat (billions of times/second)!

A Five-Step Cycle

- These operations are repeated in a never-ending sequence
- The step names suggest the operations described in the previous slide

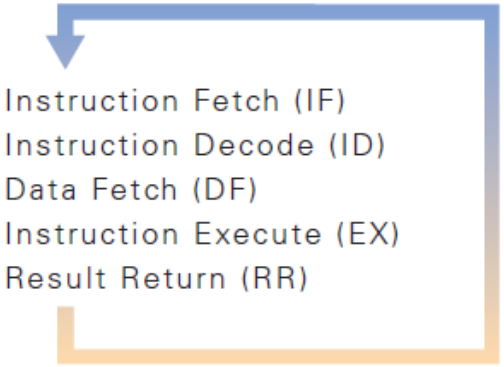


Figure 9.3 The Fetch/Execute Cycle.

Anatomy of a Computer

- All computers, regardless of their implementing technology, have five basic parts or subsystems:
  1. Memory,
  2. Control unit,
  3. Arithmetic/logic unit (ALU),
  4. Input unit, and
  5. Output unit

Principal Subsystems of a Computer

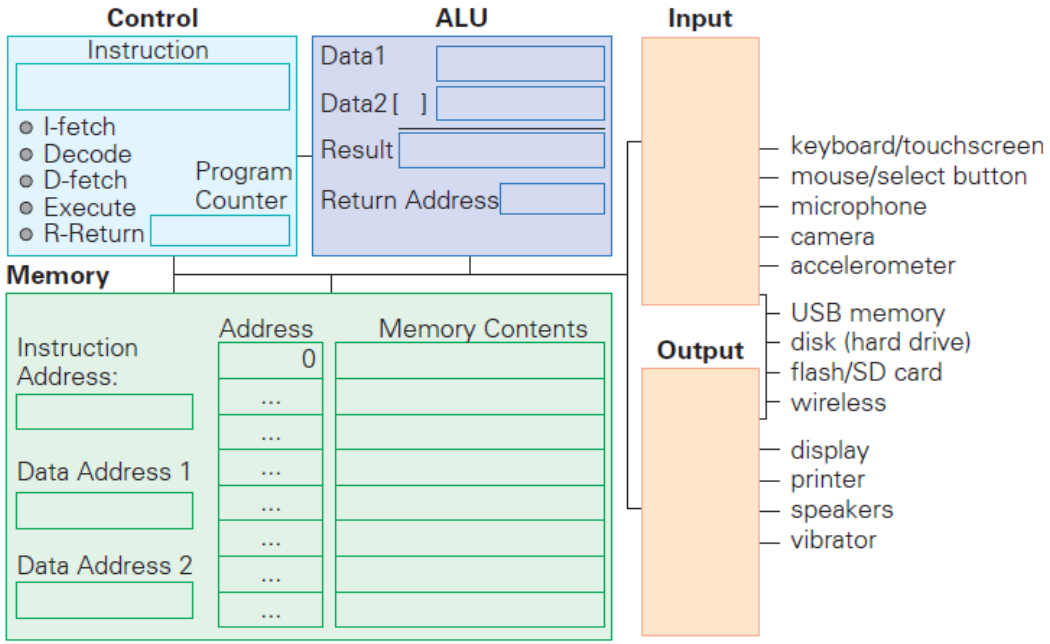


Figure 9.4 The five principal subsystems of a computer—the control unit, memory, ALU, input unit, and output unit—with typical input and output devices shown.

Principal Subsystems of a Computer

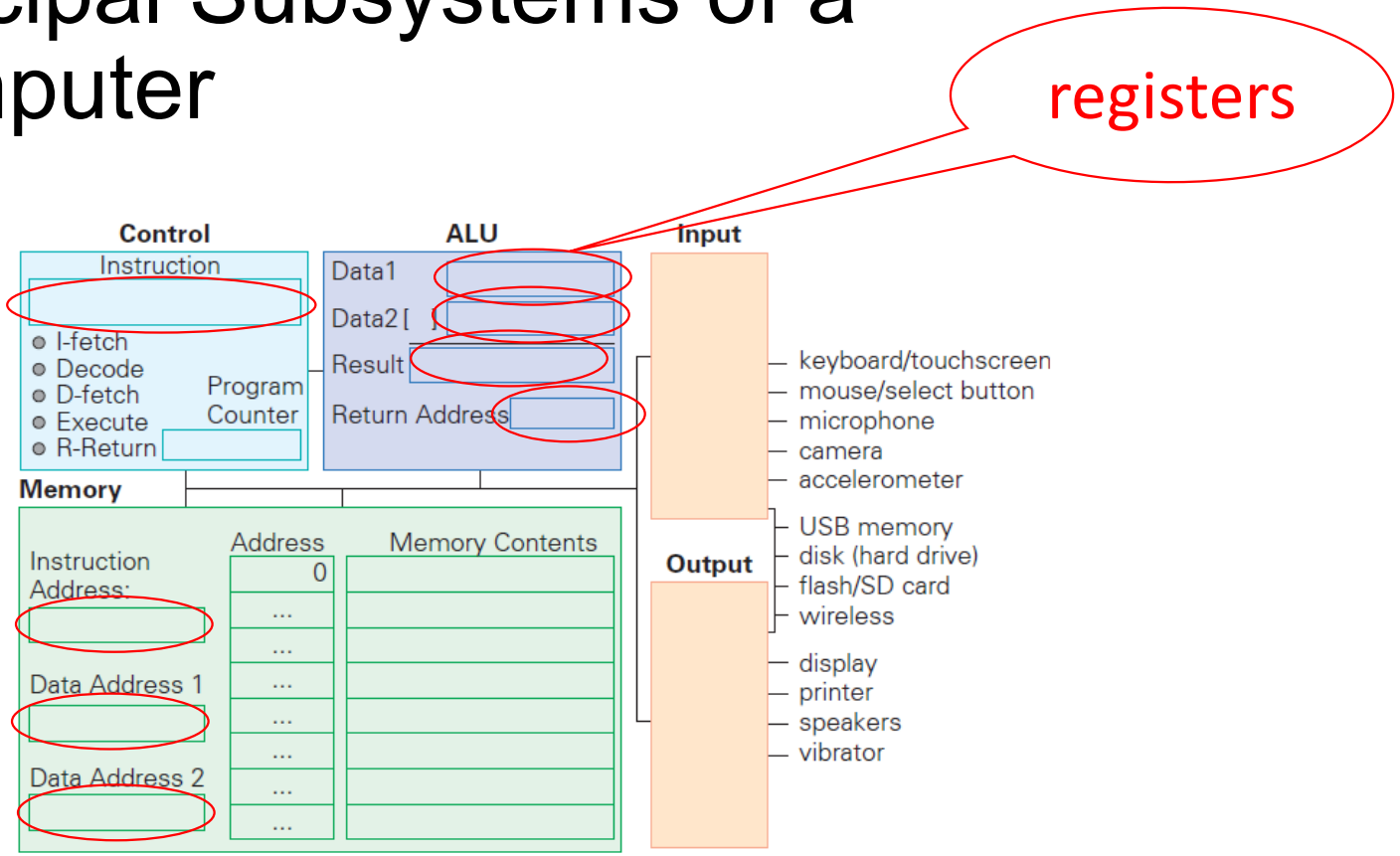


Figure 9.4 The five principal subsystems of a computer—the control unit, memory, ALU, input unit, and output unit—with typical input and output devices shown.

Registers

Memory areas where data elements and signle

istructions are stored during computer operations

1. Memory

- Memory stores both the program while it is running and the data on which the program operates
- Properties of memory:
  - **Discrete locations**
    - Memory is organized as a sequence of discrete locations
    - In modern memory, each location is composed of 1 byte (8 bits)

1. Memory

- **Addresses**
  - Every memory location has an address, whole numbers starting at 0
- **Values**
  - Memory locations record or store values
- **Finite capacity**
  - Memory locations have a finite capacity (limited size),
  - Data may not “fit” in the memory location

Byte-Size Memory Location

- Common visualization of computer memory
- Discrete locations are shown as boxes holding 1-byte each

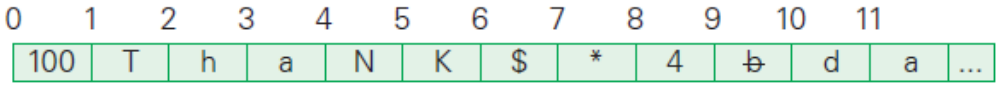


Figure 9.5 Diagram of computer memory illustrating its key properties: Discrete locations of byte-size memory, each with an address and each *containing* a value.

  - Address of location is displayed above the box and the contents of locations are shown in the box

Byte-Size Memory Location

- That 1-byte memory location can store
  - one ASCII character
  - part of a number
  - part of an instruction
- Blocks of four bytes are used as a unit so often that they are called memory **words**
- The software decides how to use the content of a memory location

Random Access Memory

- Computer memory is called **random access memory** (RAM)
  - "Random access" is out-of-date and simply means that the computer can refer to the memory locations in any order
- RAM is measured in megabytes (MB) or gigabytes (GB)
- Lots of memory is needed to provide the space required for programs and data

2. Control Unit

- The control unit of a computer is where the Fetch/Execute Cycle occurs
- Its circuitry *fetches* an instruction from memory and performs the other operations of the Fetch/Execute Cycle on it

3. Arithmetic/Logic Unit (ALU)

- “Does the math”
- A circuit in the ALU can add two numbers
- The circuit uses **logic gates** or simpler circuits that implement operations like AND and OR
- There are also circuits for multiplying, for comparing two numbers, etc.
- The ALU carries out each machine instruction with a separate circuit

4. And 5. Input and Output Units

- These two components are the wires and circuits through which information moves into and out of a computer
- A computer without input or output is useless

The Peripherals

- Peripherals connect to the computer input/output (I/O) ports
- They provide input or receiving its output
- They are not considered part of the processor:
  - They are only specialized gadgets that encode or decode information between the processor and the physical world

The Peripherals

- The keyboard encodes our keystrokes into binary form for the computer
- The monitor decodes information from the computer's memory and displays it on a screen
- The peripherals handle the physical part of the operation

Portable Memory & Hard Drives

- Some peripherals are used by computers for both input and output:
  - USB memory
  - Hard disks/drives
- They are storage devices
- The hard disk is the *alpha-peripheral,* being the most tightly linked device to the computer

A Device Driver for Every Peripheral

- Most peripheral devices are “dumb”
  - They provide only basic physical translation to or from binary signals.
- Additional information from the computer is needed to make it operate “intelligently”
- Added processing by software called a device driver gives the peripheral its standard meaning and behavior
- Every device needs a device driver

Machine Instructions

- The computer's instructions are more primitive than what *high level* programmers type
  - ADD 4000, 2000, 2080
  - Commands the computer to add the numbers stored in memory locations 2000 and 2080 and then store that in the memory location 4000
  - Computer instructions encode the memory addresses, not the numbers themselves
  - **Indirect reference**: referring to a value by referring to the address in memory

Machine Instructions

- ADD 4000, 2000, 2080
  - Looks like those three numbers should be added together
  - What it really means is that whatever numbers are stored in memory locations 2000 and 2080 be added together, and the result be stored in location 4000

Illustration of a single instruction

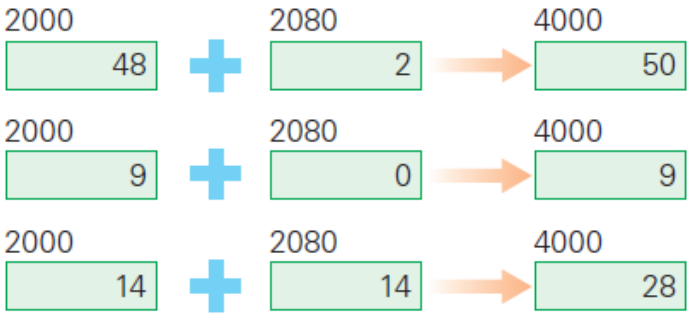


Figure 9.7 Illustration of a single **ADD** instruction producing different results depending on the contents of the memory locations referenced in the instruction.

The Program Counter: The PC's PC

- How does the computer determine what instruction it should execute next?
- Address of the Next Instruction
  - The instruction is stored in memory and the computer has its address
  - Computers use the address (known as the *program counter* or *PC*) to keep track of the next instruction

The Program Counter: The PC's PC

- After one instruction, the computer must get ready for the next
- It assumes that the next instruction is the next instruction in sequence
- Because instructions use 4 bytes of memory, the next instruction must be at the memory address PC + 4, so the processor increases the PC by 4

Branch and Jump Instructions

- Not all instructions are in a strict sequence
- The instruction may include a memory location (address) to go to next
- This changes the PC, so instead of going to PC+4 automatically, the computer "jumps" or "branches" to the specified location to continue execution

The Fetch/Execute Cycle

- A five-step cycle:
  1. Instruction Fetch (IF)
  2. Instruction Decode (ID)
  3. Data Fetch (DF) / Operand Fetch (OF)
  4. Instruction Execution (EX)
  5. Result Return (RR) / Store (ST)

ADD 4000, 2000, 2080

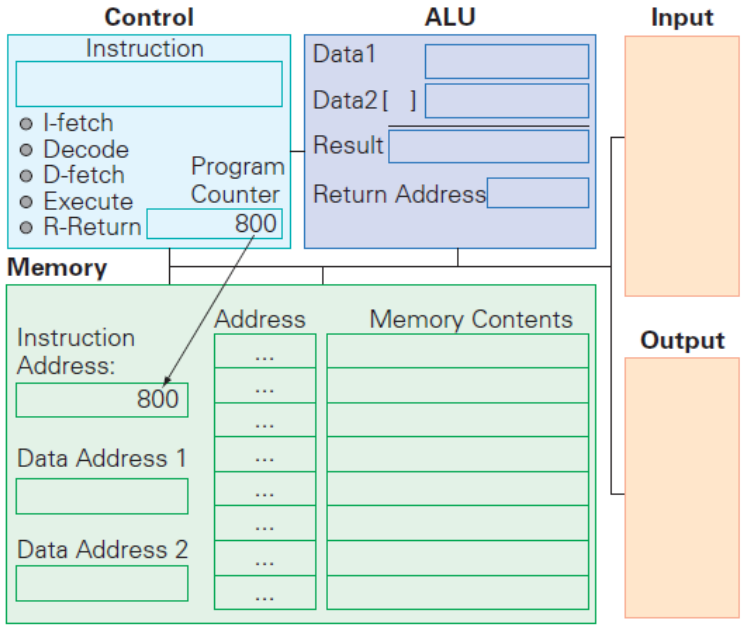


Figure 9.8 The processor before executing the instruction in memory location 800.

ADD the values found in memory locations 428 and 884 and store the result in location 800

Instruction Fetch

- Execution begins by moving the instruction at the address given by the PC (*PC 800*) from memory to the control unit
- Bits of instruction are placed into the decoder circuit of the CU
- Once instruction is fetched, the PC can be readied for fetching the next instruction

Instruction Fetch – internal view

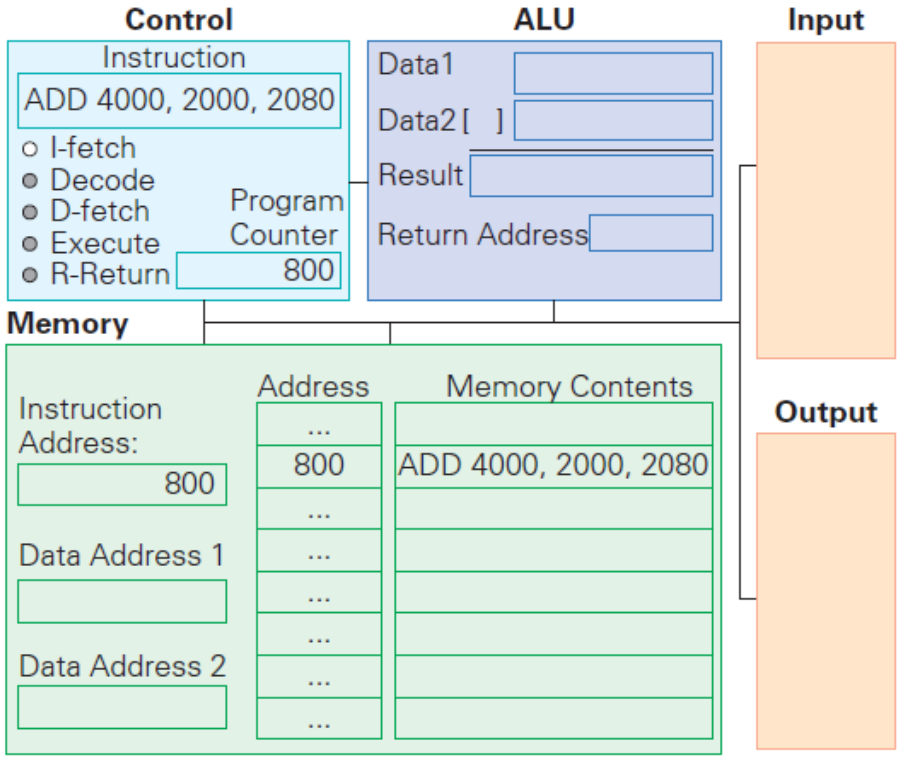


Figure 9.9 Instruction Fetch: The instruction addressed by the PC is moved from memory to the control unit.

Instruction Decode (ID)

- ALU is set up for the operation
- Decoder finds the memory address of the instruction's data (*source operands*)
  - Most instructions operate on two data values stored in memory (like ADD), so most instructions have addresses for two source operands
  - These addresses are passed to the circuit that fetches them from memory during the next step

Instruction Decode (ID)

- Decoder finds the *destination address* for the Result Return step and places the address in the RR circuit
- Decoder determines what operation the ALU will perform (ADD), and sets up the ALU

Instruction Decode

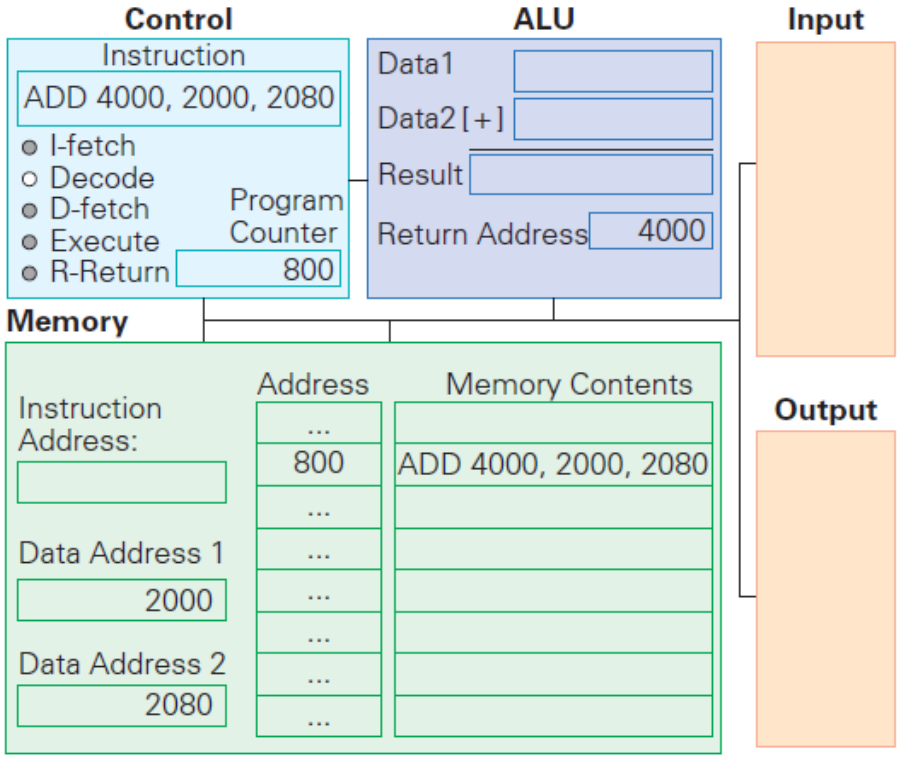


Figure 9.10 Decode: The instruction is analyzed and the processor is configured for later steps: the data addresses are sent to the Memory, the operation (+) is set in the ALU, and the result return address is set.

Data Fetch (DF)

- The data values to be operated on are retrieved from memory
- Bits at specified memory locations are copied into locations in the ALU circuitry
- Data values remain in memory (they are not destroyed)

Data Fetch

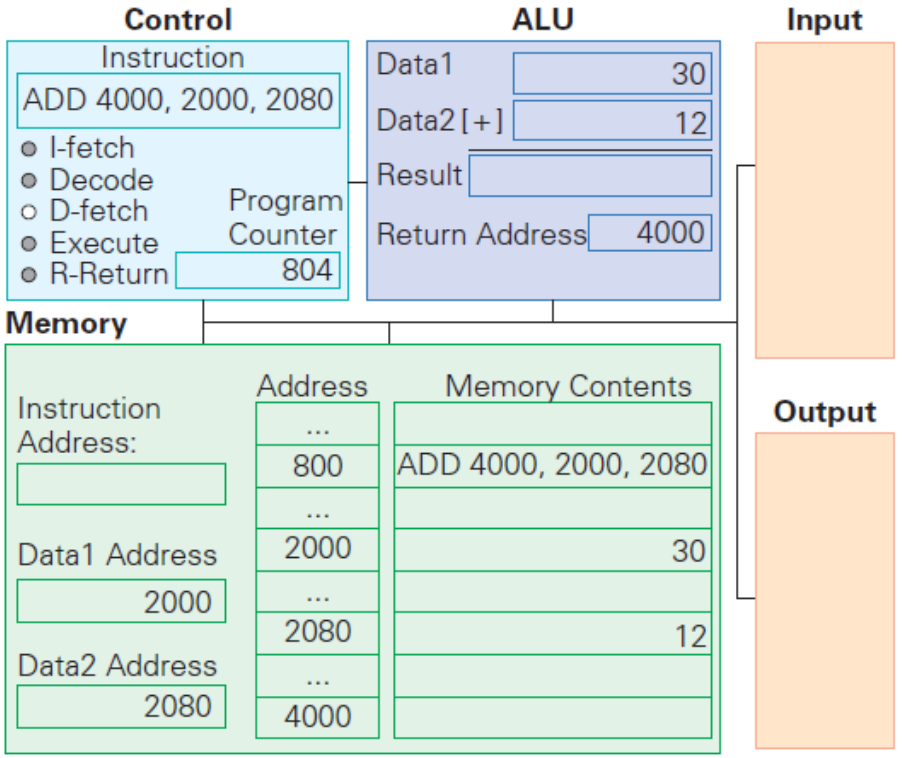


Figure 9.11 Data Fetch: The values for the two operands are fetched from memory and stored in the ALU.

Instruction Execution (EX)

- For this ADD instruction, the addition circuit adds the two source operands together to produce their sum
- Sum is held in the ALU circuitry
- This is the actual computation

EXecution

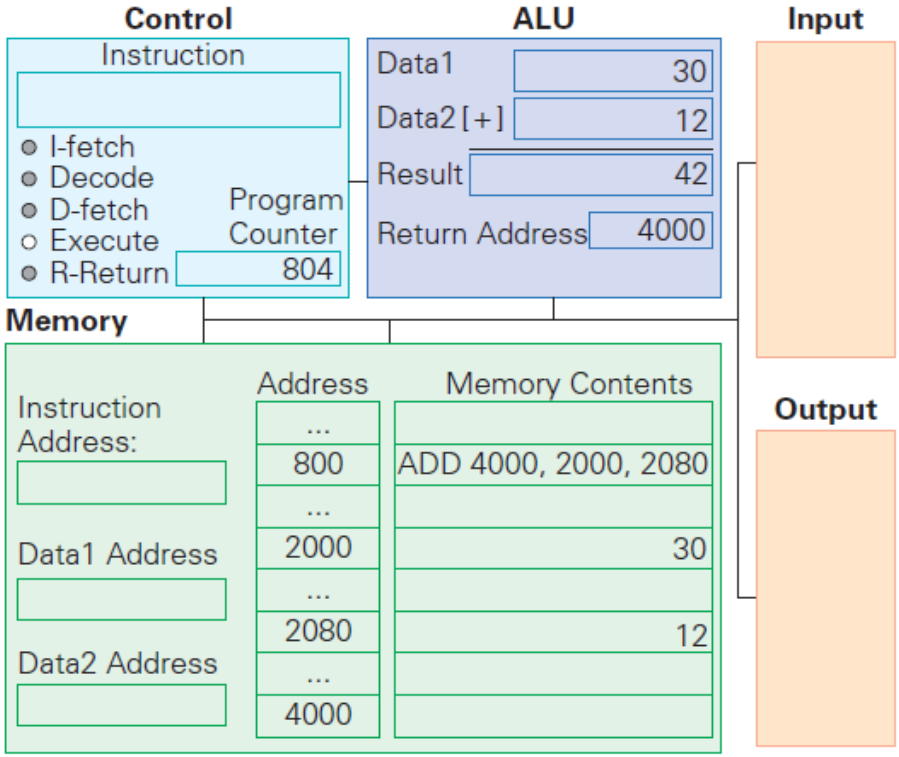


Figure 9.12 Instruction Execute: The addition operation is performed.

Return Result (RR)

- RR returns the result of EX to the memory location specified by the destination address.
- Once the result is stored, the cycle begins again

Return Result

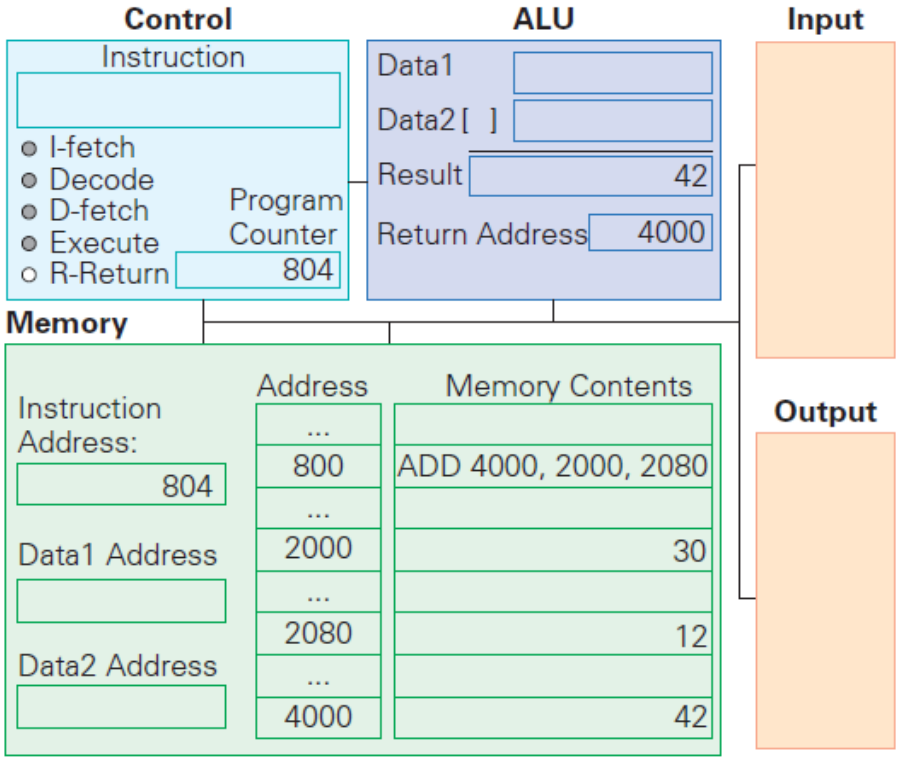


Figure 9.13 Result Return: The answer is returned to the memory, and the program counter's updated value is sent to the memory in preparation for the next fetch.

The Computer Clock

- Computers are instruction execution engines.
- Since the computer executes one instruction per cycle in principle, the speed of a computer depends on the number of Fetch/Execute Cycles it completes per second.

The Computer Clock

- The rate of the Fetch/Execute Cycle is determined by the computer's clock, and it is measured in megahertz, or millions (mega) of cycles per second (hertz).
- A 1,000 MHz clock ticks a billion (in American English) times per second, which is one gigahertz (1 GHz)

One Cycle per Clock Tick

- A computer with a 1 GHz clock has one billionth of a second—one nanosecond—between clock ticks to run the Fetch/Execute Cycle.
- In that amount of time, light travels about one foot (~30 cm).
- A simple processor might use five ticks to complete one instruction (five steps)

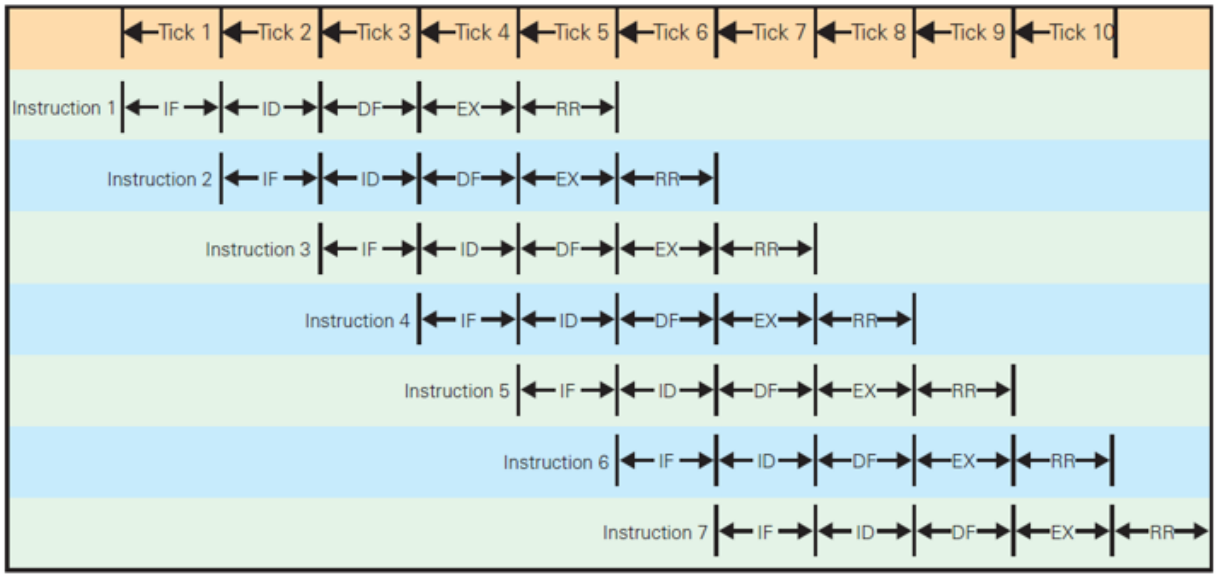
One Cycle per Clock Tick

- Modern computers try to start a new instruction each clock tick
- This is done using a *pipeline*
- A pipeline is similar to an automobile assembly line
- The CPU fetches an instruction and passes it along the pipeline
- It is then free to fetch the next instruction on the next tick

One Cycle per Clock Tick

- This is more complicated than car assembly
- Some instructions must wait for results from previous instructions
- Various other complications
- Therefore, it is not quite true that 1,000 instructions are executed in 1,000 ticks

Schematic Fetch/Execute Cycle



Many, Many Simple Operations

- Computers “know” very few instructions
- The decoder hardware in the controller recognizes, and the ALU performs, only about 100 different instructions (with a lot of duplication)
- There are only about 20 different kinds of operations.
- Everything that computers do must be reduced to some combination of these primitive, hardwired instructions

Cycling the Fetch/Execute Cycle

- ADD is representative of the complexity of computer instructions…some are slightly simpler, some slightly more complex
- Computers achieve success at what they can do with speed.
- They show their impressive capabilities by executing many simple instructions per second

Translation

- A programmer writes **source code**, such as: **this.Opacity += 0.02**
- The bits the processor needs are known as **object code, binary code,** or just **binary**
- Source code is translated into assembly code, then into binary

Assembly language

- A primitive programming language
  - Uses words instead of 0s and 1s
- ADD Opacity, TwoCths, Opacity
- To convert source code into assembly, the source code must be **compiled** by a **compiler**

Assembly language

- A compiler is a computer program that translates another computer program into assembler language
- Each language requires its own compiler
- The assembly language is converted to machine language by yet another program called an **assembler**

Integrated Circuits (ICs)

- Miniaturization
  - Computer clocks can run at GHz rates because their processor chips are so tiny
  - Electrical signals can travel one foot (30 cm) in a nanosecond
  - Early computers (the size of whole rooms) could never have run as fast because their components were farther apart than one foot
  - Making everything smaller has made computers faster

Integration

- Early computers were made from separate parts (discrete components) wired together by hand
- There were three wires coming out of each transistor, the two wires from each resistor, the two wires from each capacitor, and so on
- Each had to be connected to the wires of another transistor, resistor, or capacitor

Integration

- Active components and the wires that connect them are manufactured together from similar materials by a single (multi-step) process
- IC technology places transistors side by side in the silicon, along with the wire(s) connecting them
- Result is small and reliable

Photolithography

- ICs are made with a printing process called ***photolithography***:
  1. Begin by depositing a layer of material (like aluminum) on the silicon
  2. Cover that layer with a light-sensitive material called ***photoresist*****,** and place a mask over it
  3. The mask has a pattern corresponding to the features being constructed
  4. Exposure to **uv** light causes open areas to harden
  5. Unexposed areas do not and can be washed away leaving the pattern
  6. Hot gases etch the original layer
  7. When the remaining photoresist is removed, the pattern from the remains

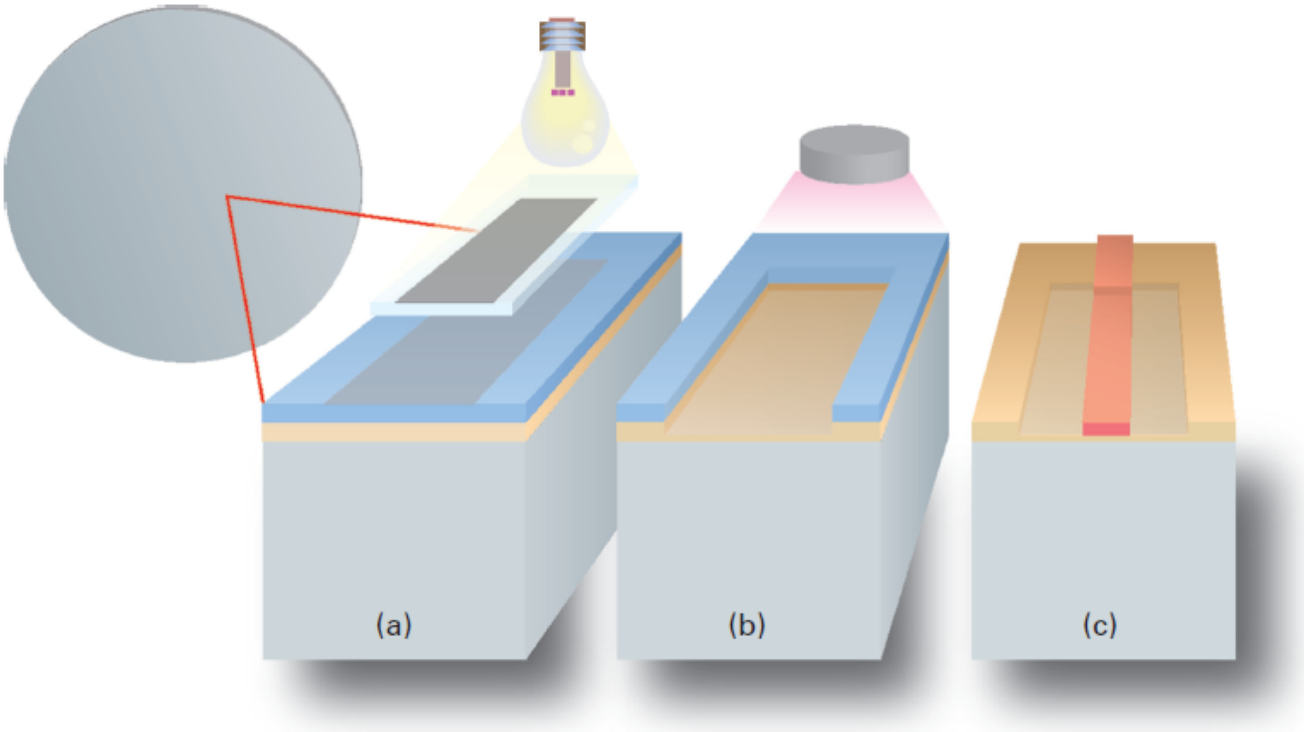
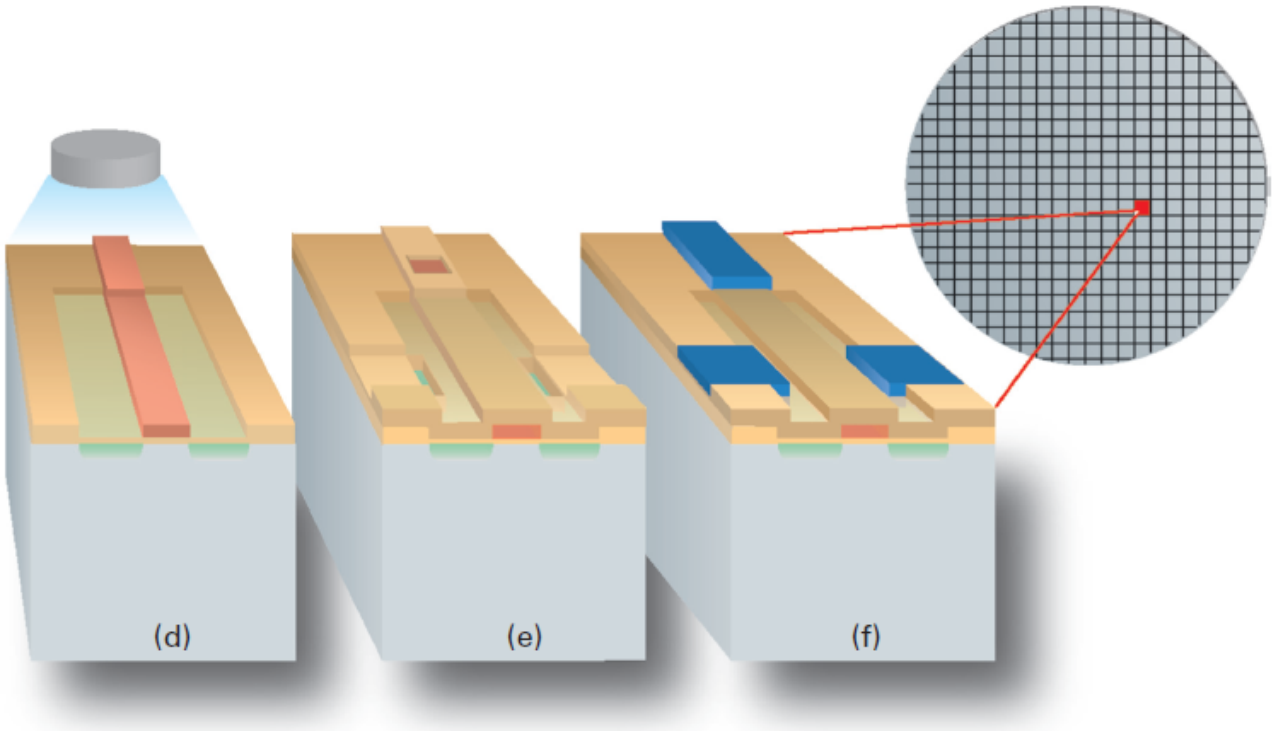


Figure 9.14 Early steps in the fabrication process. (a) A layer of photoresist (blue) is exposed to UV light through a pattern mask (light blue), hardening the exposed areas; (b) after washing away the unexposed photoresist, hot gases etch away (nearly all of) the exposed layer; and (c) the remaining resist is washed away and other layers are created by repeating the patterning and etching processes. In later stages of the fabrication process, (d) "impurities" (green) such



as boron are diffused into the silicon surface in a process called doping, which improves the availability of electrons in this region of the silicon. (e) After additional layering, etching exposes contact points for metal wires, and (f) a metal (dark blue) such as aluminum is deposited, creating "wires" to connect to other transistors. Millions of such transistors form a processor chip occupying a small square on the final fabricated wafer.

Transistors

- A ***transistor*** is a connector between two wires that can be controlled to allow a charge to flow between the wires (conduct) or not
- The transistor is a ***MOS*** (Metal Oxide Semiconductor) ***transistor***

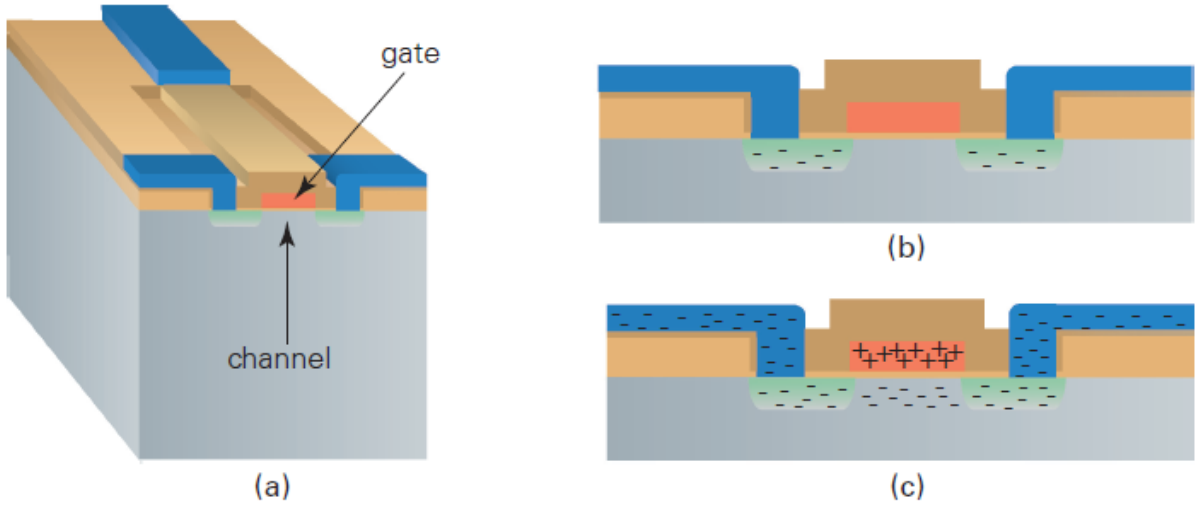


Figure 9.15 Operation of a field effect transistor. (a) Cross-section of the transistor of Figure 9.14(f). (b) The gate (red) is neutral and the channel, the region in the silicon below the gate, does not conduct, isolating the wires (blue); (c) positively charging the gate attracts electrons into the channel by the field effect, causing the channel to conduct and connecting the wires.

How Semiconductor Technology Works

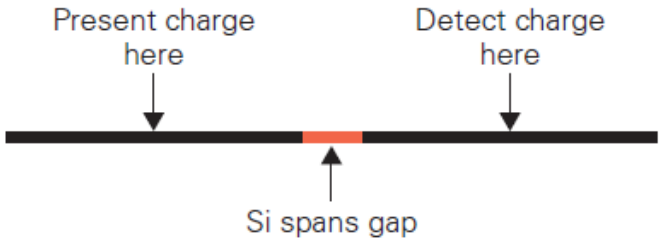
- Silicon is a ***semiconductor*****:** sometimes it conducts electricity and sometimes it does not
- The ability to control when semiconductors do and don’t conduct electricity is the main process used in computer construction

The Field Effect

- The silicon in the channel can conduct electricity when it is in a charged field
- Charging the gate positively creates a field over the channel
- Electrons are then attracted from the silicon into the channel, causing it to conduct
- If the field is removed, the electrons disperse into the silicon, the channel doesn’t conduct

On-Again, Off-Again

- The transistor controls signals in the computer



On-Again, Off-Again

- Transistors can be combined to compute A AND B

Implementing ALU Operations

- Logical operations such as AND and OR can be built from transistors
- More complicated combinations can be made from these
- Arithmetic, memory and control units can be made combining these parts
- Eventually, you have an ALU, and then a full processor

What's important about *memory*

- Access time
  - Is the time necessary to read or write the content of a memory location
- Unit cost
  - Is the cost per stored byte
- Volatile memory
  - The stored information is lost when the computer is switched off
- Persistent memory
  - The stored information is kept safe when the computer is switched off
- Maximum size of memory for a given technology

The perfect memory

- Very short access time
- Very low unit cost
- Very, very large maximum size
- Persistent

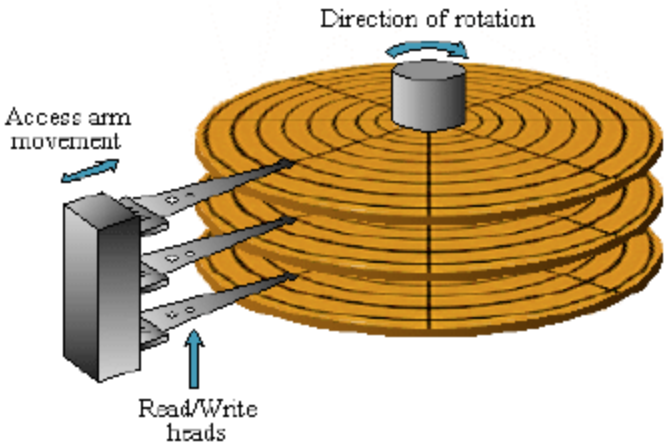
A hierarchy of memories

Memory type	Access time (sec)	Unit Cost	Volatile/ Persistent	Maximum size
CPU register	10^{0}←Increases 10^{-10}	10^{-10}Increases→	V	10^{12}←Increases
Internal cache			V	
External cache			V	
RAM			V	
Hard Disk			P	
Removable device			P	

More about memory

- For a better tradeoff, modern computer contain the complete hierarchy of memories
  - The amount of memory of each type is optimized for best tradeoff between performance and cost
- Cache memories allow optimal transfer time between ram and cpu registers
- Hard disks are the standard safe storage
- Removable memories allow best portability

Hard disk

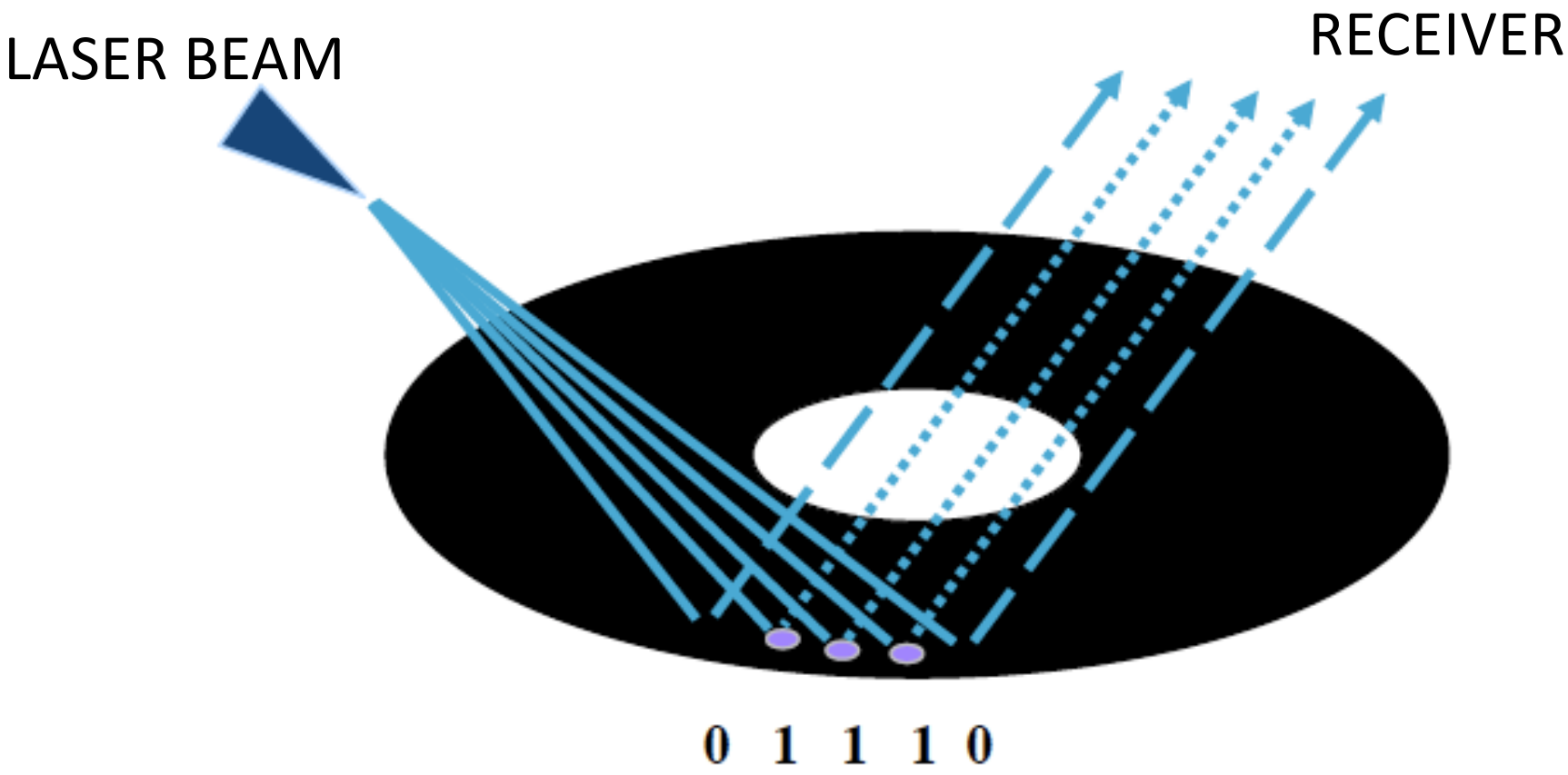

Direction of rotation
Access arm movement
Read/Write heads


Maxtor

Optical memory peripheral units

- Two components
  - Drive, with electronic, optical and mechanical devices, connected to the machine
  - Optical support, inserted in the drive and easily transported
- Laser technology
  - To write the reflectivity of the surface is phisically modified
  - To read the reflectivity is detected with a laser beam

Optical memory peripheral (ii)



Flash memories

- Electronic devices without moving parts
- Fast
- Permanent
- Easily rewritable
- Not really expensive
- Removable
- Much slower than RAM
- Much faster than magnetic HD

I/O peripherals

- Implement the HCI (human/computer interface)
  - But also machine/machine interfaces
- Input
  - Data enter into the computer
- Output
  - Data flow from the computer to the external world
- Everything is controlled by the cpu
- Dedicated cards implement the details
- Cards connected to the main BUS
  - A kind of *data highway* for exchanging data between computer components

I/O units

Input

- Keyboard
- Mouse
- Trackpad
- Scanner
- Camera
- Microphone
- …

Output

- Monitor
- Printer
- Audio
- Actuators
  - Devices designed to control other machines
    - E.g. cars, robots, …
- ...

Keyboard

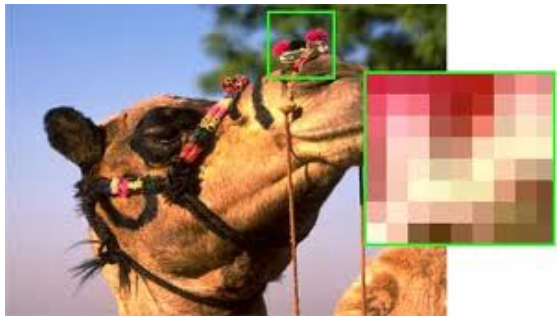
- Character input
- Sends characters to a *buffer memory*
- Software reads characters from the buffer memory

Mouse

- Transforms position/movement into digital signal
- Technology
  - Mechanical (obsolete)
  - Optical
  - Magnetic
  - Touch

Video (i)

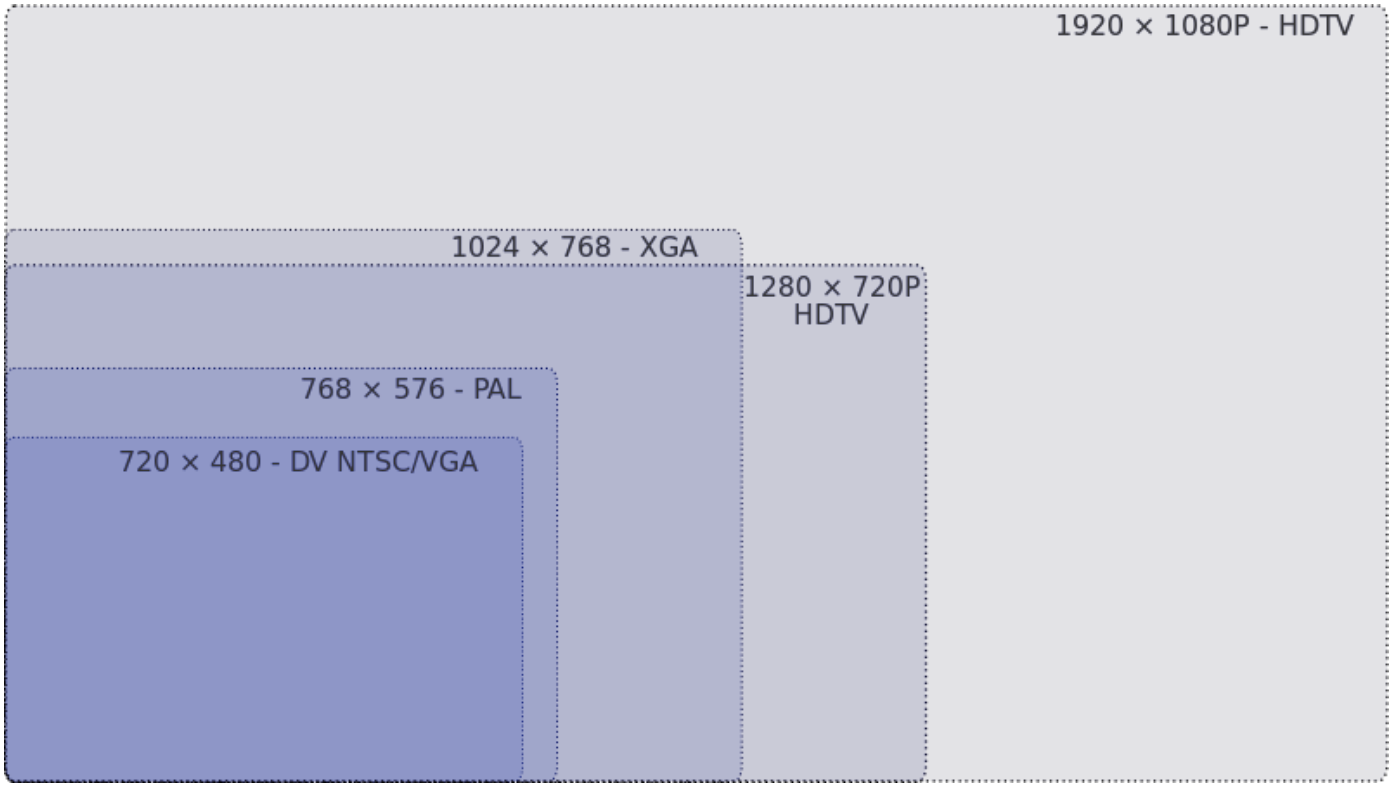
- Array of colored light dots
  - Pixel = picture element
- Resolution = number of pixels
- Diagonal size (inch)
- Color depth = number of possible different pixel colors
  - 24 bits = 2^{24} colors
- Dot pitch = pixel per square inch
- Form factor = L/H ratio
  - 4:3, 16:9, …



Monitor sizes

Resolution	Megapixel	Screen size	Pixel size (mm)	Pixel per Inch (DPI)
1024x768	0.78	15"	0.297	85.5
1440x900	1.29	15.4"	0.230	110.4
1280x720	0.92	32"	0.553	46.4
1920x1080	2.07	32"	0.369	69.7
1920x1200	2.30	17"	0.191	134.8

Screen sizes for equal dot pitches



Video controller

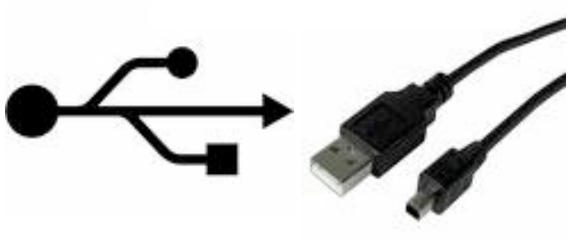
- Connected to the bus, contains the *video port*
- Has computing power on its own
  - To lighten the CPU work
  - Very important for global computer performance
- Video ports
  - VGA (analog, soon obsolete)
  - DVI (Digital Video Interface)
  - HDMI (last standard, tv compatible, includes audio)

Printer

- Transfers text and image on paper
- Quality characteristics
  - DPI
  - Pages per minute
  - Technology
    - Laser
    - Ink jet

Communication ports

- Allow connection of peripherals to the computer
- Transfer speed
- USB (3.1)
  - Data + power supply
  - Up to 10 Gb
  - Normal, mini, micro, USB-C
- HDMI
- …



HDMI™
HIGH DEFINITION MULTIMEDIA INTERFACE

Computer categories

- Microprocessor
  - Included in more complex devices
- Personal computer
  - Desktop, laptop, notebook, tablet, smartphone
- Workstation
- Departmental server
- Mainframe
- Supercomputer

Software

Who's in charge?

A bridge between the user and the machine

- The computer deals only with binary data and instructions
  - Impossible to use it directly
  - Strictly related to the specific computer
- Abstraction w.r.t. the physical structure
- Use human-friendly interaction
- Use human-friendly techniques to program the computer
- Needed specialized software to perform common tasks

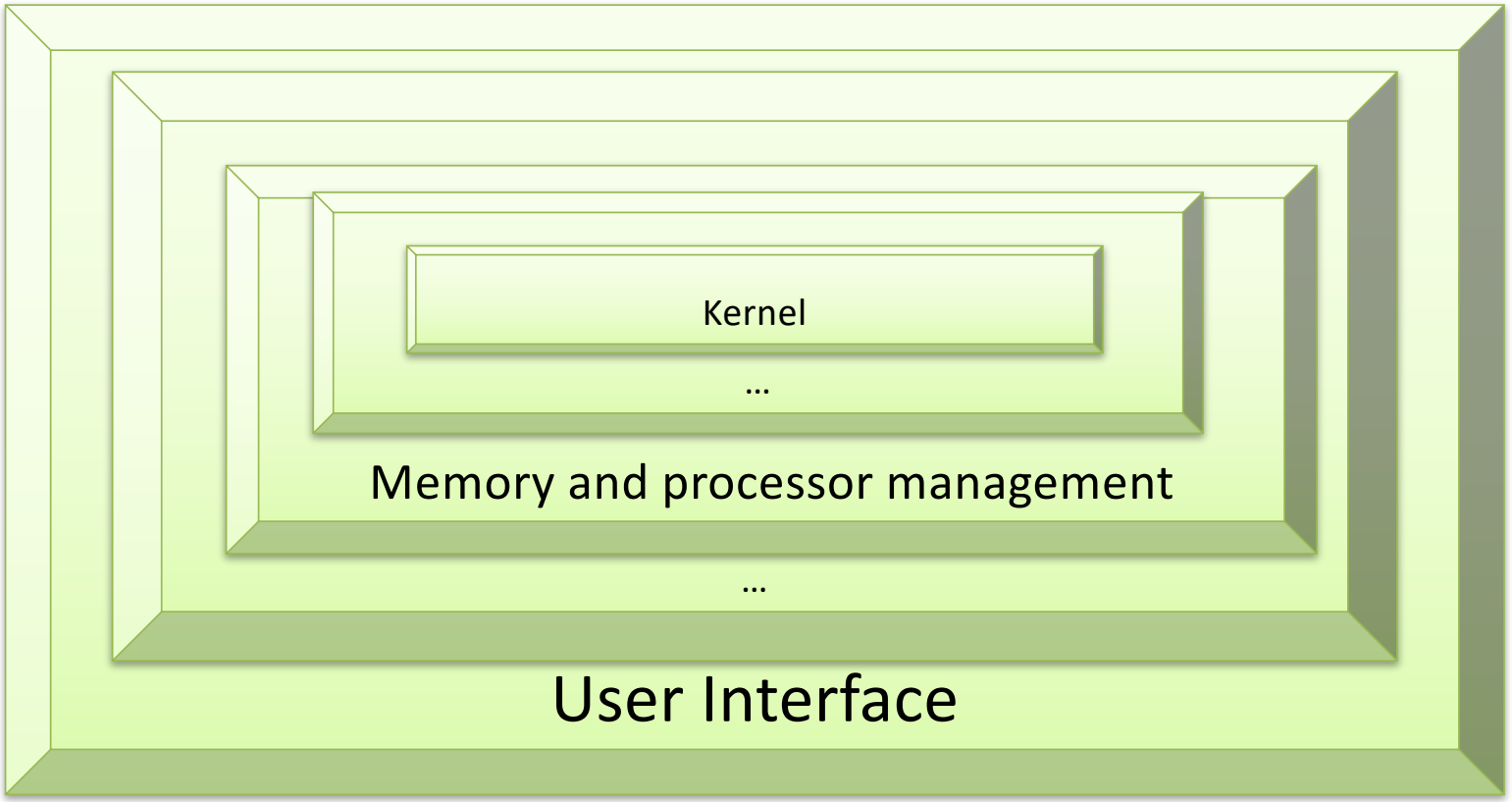
Operating system

- A set of software programs, cooperating to:
  - Effectively manage the computer and all the hardware components
  - Make available an effective environment for the Human/Computer interaction
- The OS controls completely all the computer resources
- The OS translates and implements the user requests

OS tasks

- Check and startup the machine
- Manage the processor
- Manage main memory
- Manage secondary memory
  - File system
- Manage peripherals connected trough the communication ports
- Manage the HCI
- Supports the execution of user programs

Logical structure of the OS



Bootstrap

- Check-up of hw resources and initialization of management programs
- Diagnosis
- Look for *boot sector* in secondary memory
- Load the OS in memory, as indicated in the boot sector
- Start of all the OS services
- Wait for user requests or for peripheral activity

BIOS (Basic Input-Output System)

- The innermost part of the OS
- Stored in the Read-Only Memory (ROM)
  - Provided by the computer manufacturer
- Manages hw resources
- Manages the search for the boot sector and OS load and start

CPU characteristics

- Architecture
  - Lots of technical aspects, among them
    - Set of instructions
  - E.g. Intel i5, i7, …
- Clock speed (GHz = 10^9 cycles per second)
- Number of cores
  - Several CPUs working in parallel

Processor and processes

- The processor is a piece of hw able to execute programs
- A process is a program loaded in main memory and being executed
- Process states
  - Executing
  - Waiting for some events
    - E.g. user input
  - Ready to use the processor
    - But the processor is being used by another process
- Process life is completely controlled by the OS

Multitask

- Modern OS allow simultaneous execution of many processes
- As a matter of fact, a single processor executes one process at a time
- The OS assigns slots of processor usage alternatively to the *living* processes
  - In this way the resources can be better exploited
    - E.g. when a process is waiting for something, the processor can be used by another process

Management of main memory

- Because of multitasking, main memory must be shared by processes
- Interference *must be* avoided
- The amount of memory necessary for a process depends on program complexity and on the amount of data to be processed
- A bigger main memory allows the execution of a greater number of simultaneous processes and the management of a larger amount of data

Virtual memory

- The OS is able to use part of the secondary memory as a *virtual extension* of main memory
- When main memory is full, some of the ready processes are stored in secondary memory and deleted from the main
- They well be read again in main memory later
- This behavior affects only the speed, not the functionalities
  - Writing out and reading in a process requires additional time

Operating sistems (2020, continuously changing)

- PC
  - Microsoft windows
    - 7
    - 8
    - 10
  - Apple Mac OSX
    - 10.15
  - Linux
    - MX, Manjaro, Mint, Debian, Ubuntu, ...
- Mobiles
  - Android (73%)
  - iOS (25%)
- Mainframes
  - Unix
  - IBM
  - …

Combining the Ideas

- Start with an information-processing task.
  - Task is performed by an application implemented as a large program
    - The program's commands are compiled into many simple assembly language instructions
      - The assembly instructions are then translated into a more primitive binary form
        - Fetch/Execute Cycle executes the instructions
          - The processor uses logic built of transistors to perform the execution

Summary

- You learned the following:
  - Modern software is written in a language using familiar terms and operations, though they are expressed very briefly; the code relies heavily on the software stack
  - The repeating process fetches each instruction (indicated by the PC), decodes the operation, retrieves the data, performs the operation, and stores the result back into the memory

Summary

- You learned the following:
  - This process is hardwired into the control subsystem, one of the five components of a processor
  - The memory, a very long sequence of bytes, each with an address, stores the program and data while the program is running
  - The ALU does the actual computing

Summary

- You learned the following:
  - The input and output units are the interfaces for the peripheral devices connected to the computer
  - Machine instructions do not refer to the data (operands) directly, but rather indirectly. Thus, different computations can be done with an instruction, just by changing the data in the referenced memory locations each time the instruction is executed

Summary

- You learned the following:
  - Programmers use sophisticated programming languages to create operating systems as well as complex applications software
  - The basic ideas of integrated circuits are integrating active and connective components, fabrication by photolithography, and controlling conductivity through the field effect