Hardware

- Input/Output Devices
- Secondary Storage
- Memory
- CPU
- Buses

HARDWARE

- Refers to the physical components of the computer
  - Anything you can physically touch inside the box or outside
  - Outside
    - Mouse
    - Keyboard
    - Monitor
  - Inside
    - Motherboard
    - Memory
    - Hardrive
    - Etc..

- Inside your Computer

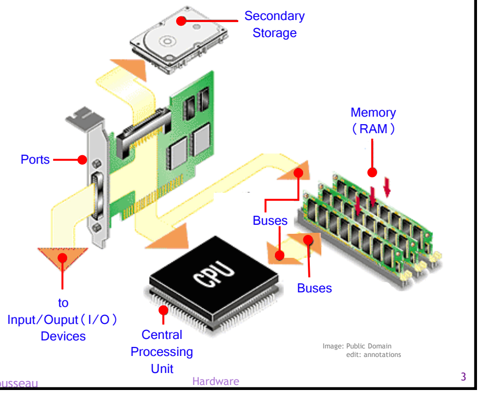


Image: Public Domain
edit: annotations

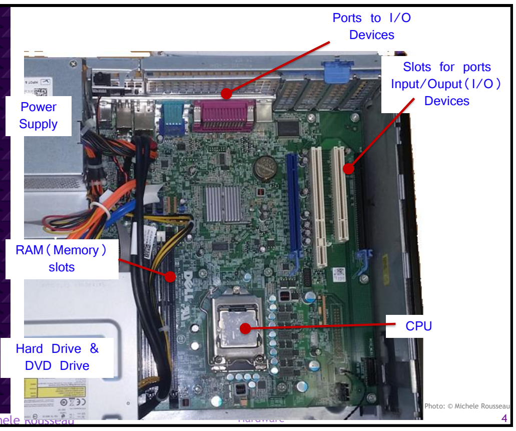


Photo: © Michele Rousseau

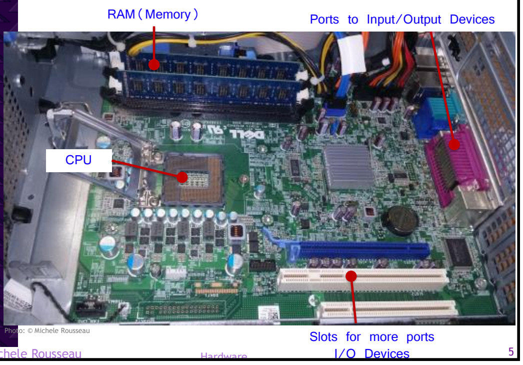

RAM (Memory)
Ports to Input/Output Devices
CPU
Slots for more ports I/O Devices

Photo: © Michele Rousseau

Non-storage – I/O Devices

- Allow the computer to communicate with the outside world - Do not store data
  - Keyboard
  - Monitor
  - Printer
  - Mouse
  - Microphone
  - speakers
  - Etc…

Illustration: Public Domain

Image: Public Domain

Illustration: Public Domain

I/O Storage Devices

(AKA auxiliary storage devic

- Cheaper than memory
- Non-volatile

Image: Public Domain

Image: Public Domain

Examples

- Magnetic disk (typical hard disk drive)
  - Platters
    - Constructed of light aluminum alloy
    - OR → glass or ceramic (more resistant to heat)
    - Coated with magnetizable material (ferrite compound)
      - On both sides
  - Can have several platters
  - Typically store in the high GBs and TBs

Image: CC - BlickPixel @ pixabay

More External Storage

- Compact Discs (CDs) - Digital Versatile Discs(DVDs) & Blu-ray Discs (BDs)
  - ROM - Read only
  - R – WORM (Write Once Read Many)
  - R/W – Read/Write (more expensive
  - How much do they typically hold
  - CD → 650 MB, DVD → 4.7 GB, BD → 25 GB
  - DVDs and BDs can be dual layer 2 (record on both sides)

Image: Public Domain

- USB flash drives → flash drives, USB drives, jump drives, pen drives, thumb drives, key drives, tokens
  - Flash RAM or Flash Memory (Can R/W)
    - Special type of Electronically Erasable Programmable Read Only Memory
  - Non-volatile
  - Chip

Image: Public Domain

Main Memory

- Collection of storage locations
  - Not the same as a hard drive this is internal to the system – located on the mother board
  - MUCH faster
- Each has its own address
  - Like a street address – but always unique
  - The address is in binary (1s and 0s)
- Remember bytes?
  - More useful for storing information than 1 bit
  - Used to represent a character in ASCII
    - American Standard Code for Information Interchange
    - Used to be 128 chars – extended is 256 chars

3 types of Memory

- RAM
  - *Random Access Memory*
    - Read / Write
    - ALL programs and data must be in RAM to be processed
    - Volatile
    - Most of main memory

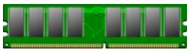


Image: Public Domain

Image: CC - Laserlicht
Edit - removed bkground

- ROM
  - *Read Only Memory*
    - Contents written my computer manufacturer
    - Read only → can't write to it
    - "Bootstrap" is on the ROM chip
    - Non-volatile

Types of Memory (2)

- Cache
  - Faster than RAM slower than CPU registers
  - Between registers and primary memory
  - Cheaper and more plentiful than registers
  - Relatively small amount of memory
    - Compared to RAM
  - Contains a copy of a portion of main memory
    - CPU - checks to see if requested portion is in cache
    - If so, it retrieves it
    - If not, it has to go to main → replaces cache with new data retrieved
    - Most processing is performed with a small portion of data → so mostly will be in cache

How we measure Memory

- We measure memory & external storage in terms of bytes

Unit	Number of Bytes	Decimal Approximation
kilobyte	2^{10}	10^3
megabyte	2^{20}	10^6
gigabyte	2^{30}	10^9
terabyte	2^{40}	10^{12}
petabyte	2^{50}	10^{15}
exabyte	2^{60}	10^{18}
zetta	2^{70}	10^{21}
yotta	2^{80}	10^{24}

BUSES

- Electrical pathways (wires)
  - Each wire can transmit 1 bit of information
- These connect all the components to the CPU
- System Bus
  - Internal Bus
  - CPU and Memory
- Expansion Bus
  - External Bus
  - CPU and I/O Devices

Image: Public Domain

System Buses

- The CPU transfers data, addresses and instructions to/from main Memory via the system bus

Von Neumann's paper proposed a single bus, but this created a bottleneck so…

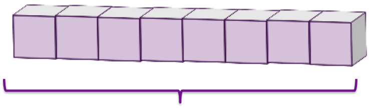
- 3 types of Buses
  - Data Bus → moves data between main memory and the CPU registers
  - Address Bus → holds the address of the data that the data bus is accessing
  - Control Bus → carries the instructions that specify how the information transfer is to take place

System/Internal Buses

- Word Size
  - The amount of data that can be handled as a unit at one time
- Data Bus
  - → moves data from the main memory to the CPU and back
    - 16 bit → 16 wires
    - 32 bit → 32 wires… etc
  - 1 word is transmitted at a time
  - Size dictates how the systems word size
- Address Bus
  - → holds the address of the data that the data bus is currently accessing
  - Used to access a specific word in memory
  - # of wires is determines the # of addressable locations
  - Typically the word size or a multiple or fraction thereof
- Control Bus
  - → Indicates whether or not a read or write is to be performed

Data Bus, Address Bus & Ram

- This size of the Data Bus and the Address Bus dictate how much RAM the system can manage
- Think of RAM as a group of boxes – each representing a memory location.
  - The size of that box or the amount that can be stored in each box is determined by the data bus
- Each Memory location in RAM has a unique address



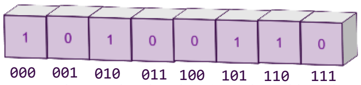
How many bits/wires do we need to represent these 8 addresses? 3 bits or wires

- How may bits/wires would my address bus need to represent 32 addresses? 5 bits or wires → $2^5 = 32$

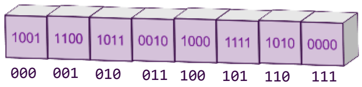
000 001 010 011 100 101 110 111

Data Bus, Address Bus & Ram

- This size of the Data Bus and the Address Bus Dictate how much RAM the system can manage
  - $2^{\text{\# of bits in the Address Bus}}$ X # of bits in the data bus = Max RAM
- If a system has a 1-bit data bus & a 3-bit address bus
  - How much RAM can it manage?



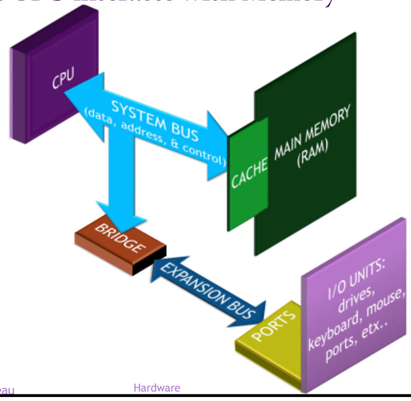
- If a system has a 4-bit data bus & a 3-bit address bus
  - How much RAM can it manage?



Expansion Bus

- → I/O devices communicate with the CPU/Memory through ports
- → These ports are connected to an Expansion or PCI bus
- → The Expansion or PCI bus communicates with the System Bus through a bridge
- → The bridge manages traffic between the Expansion bus and the System bus
- → The system bus is much faster than the Expansion or PCI bus

The CPU interacts with Memory



Mauchly & Eckert

- Built the ENIAC & UNIVAC

… while building the ENIAC
  - Came up with a way to store programs
  - To create a new program on the ENIAC wires had to be unplugged and plugged into new sockets

Image: Public Domain

- John Von Neumann published the idea of storing programs and data internally
- We call this the Von Neumann architecture

The CPU

- Central Processing Unit
  - The “brain” of the computer
  - Divided into 3 parts (Registers, CU, ALU)

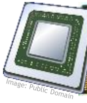


Image: Public Domain

Image: © Michele Rousseau

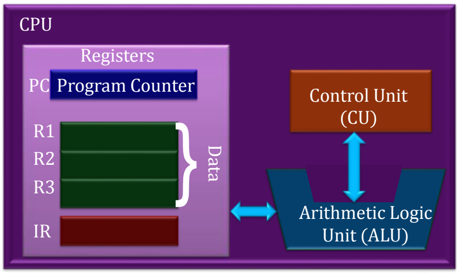


Image: © Michele Rousseau

Elements of the CPU

Control Unit (CU)
- Transfers data to and from memory
- Calls the Arithmetic Logic Unit when necessary
- Fetches instructions
- Interprets instructions
- Executes instructions in order

Arithmetic Logic Unit (ALU)
- Performs all arithmetic & logical operations
- Arithmetic operations
  - Increment & Decrement (unary operations – 1 input/operand
  - Addition & Subtraction(binary operations – 2 inputs/operands
  - Multiplication & Division
- Logical operations
  - NOT, AND, OR, and XOR

Registers

Registers are very fast temporary locations used to store data on the CPU
- Data to be processed is not in memory
  - it is moved to the CPU (registers)
- Extremely fast - speeds execution time
- Registers hold partial results of calculations before they can be stored back into memory

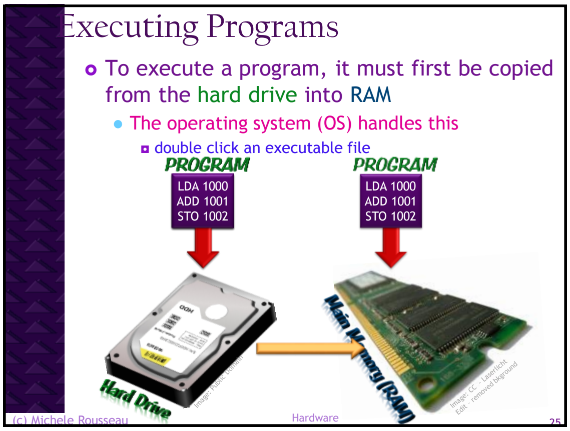
- Two basic types of registers
  - General purpose
    - (for data and partial calculations)
  - Special purpose registers

Special Purpose Registers

The CPU contains a number of Special Purpose Registers

Two basic Special Purpose Register
- Program Counter (PC)
  - Keeps track of which statement is currently being executed
  - When a statement completes its execution the PC is incremented
    - (gets the memory address of the next instruction)

- Instruction Register(IR)
  - Contains the current instruction



Executing Programs

- To execute a program, it must first be copied from the hard drive into RAM
  - The operating system (OS) handles this
    - double click an executable file

PROGRAM

LDA 1000
ADD 1001
STO 1002

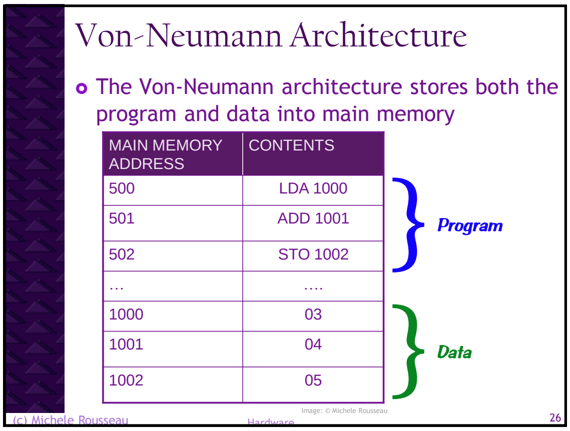
PROGRAM

LDA 1000
ADD 1001
STO 1002

Image: Public Domain

Image: CC - Laserlicht
Edit - removed bkground

(c) Michele Rousseau Hardware 25



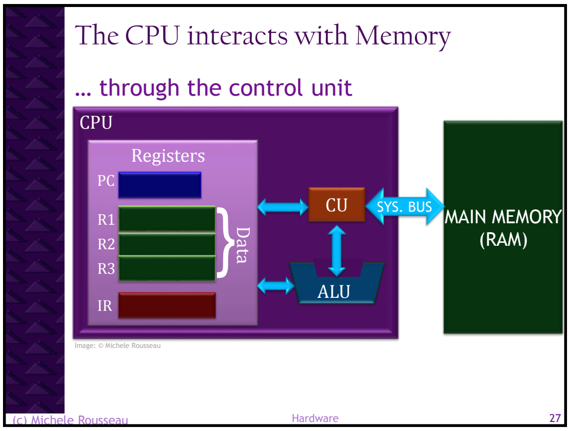
Von-Neumann Architecture

- The Von-Neumann architecture stores both the program and data into main memory

MAIN MEMORY ADDRESS	CONTENTS
500	LDA 1000
501	ADD 1001
502	STO 1002
…	….
1000	03
1001	04
1002	05

Image: © Michele Rousseau

(c) Michele Rousseau Hardware 26

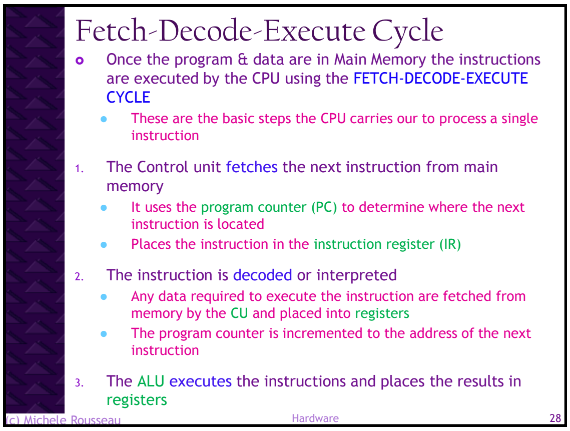


The CPU interacts with Memory

… through the control unit

Image: © Michele Rousseau

(c) Michele Rousseau Hardware 27

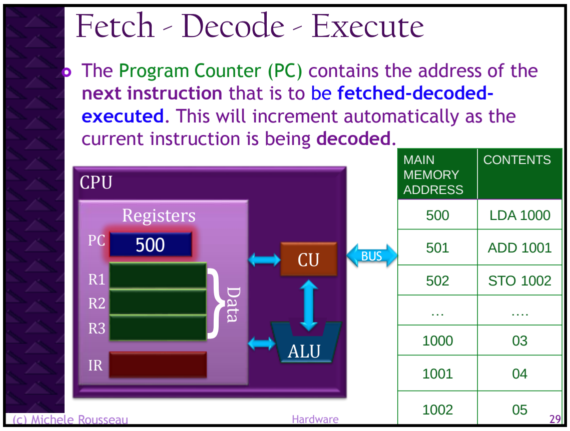


Fetch-Decode-Execute Cycle

- Once the program & data are in Main Memory the instructions are executed by the CPU using the FETCH-DECODE-EXECUTE CYCLE
  - These are the basic steps the CPU carries our to process a single instruction

1. The Control unit fetches the next instruction from main memory
   - It uses the program counter (PC) to determine where the next instruction is located
   - Places the instruction in the instruction register (IR)
2. The instruction is decoded or interpreted
   - Any data required to execute the instruction are fetched from memory by the CU and placed into registers
   - The program counter is incremented to the address of the next instruction
3. The ALU executes the instructions and places the results in registers

(c) Michele Rousseau Hardware 28

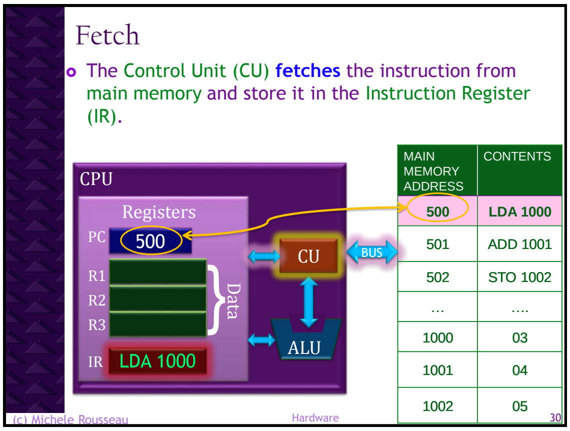


Fetch - Decode - Execute

- The Program Counter (PC) contains the address of the **next instruction** that is to be **fetched-decoded-executed**. This will increment automatically as the current instruction is being **decoded**.

MAIN MEMORY ADDRESS	CONTENTS
500	LDA 1000
501	ADD 1001
502	STO 1002
…	….
1000	03
1001	04
1002	05

(c) Michele Rousseau Hardware 29

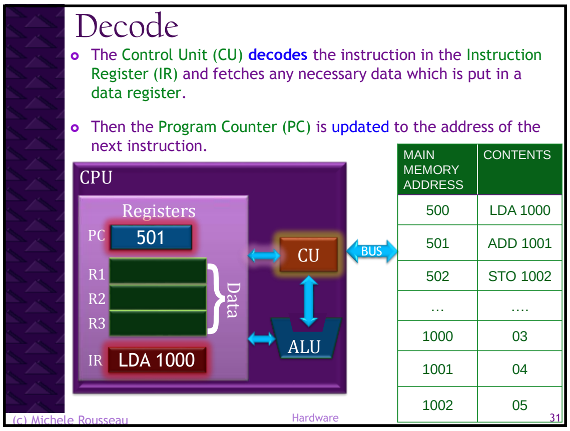


Fetch

- The Control Unit (CU) **fetches** the instruction from main memory and store it in the Instruction Register (IR).

MAIN MEMORY ADDRESS	CONTENTS
500	LDA 1000
501	ADD 1001
502	STO 1002
…	….
1000	03
1001	04
1002	05

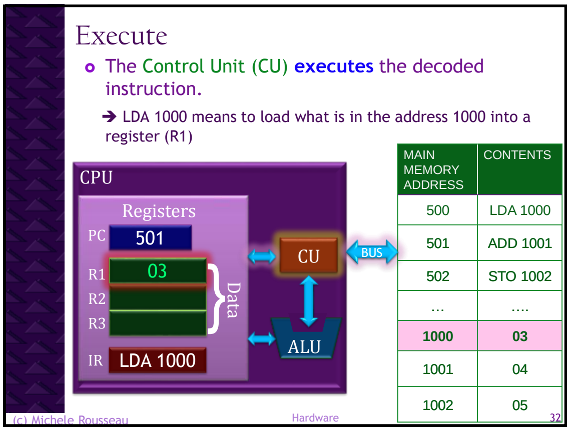
(c) Michele Rousseau Hardware 30



Decode

- The Control Unit (CU) **decodes** the instruction in the Instruction Register (IR) and fetches any necessary data which is put in a data register.
- Then the Program Counter (PC) is updated to the address of the next instruction.

MAIN MEMORY ADDRESS	CONTENTS
500	LDA 1000
501	ADD 1001
502	STO 1002
…	….
1000	03
1001	04
1002	05

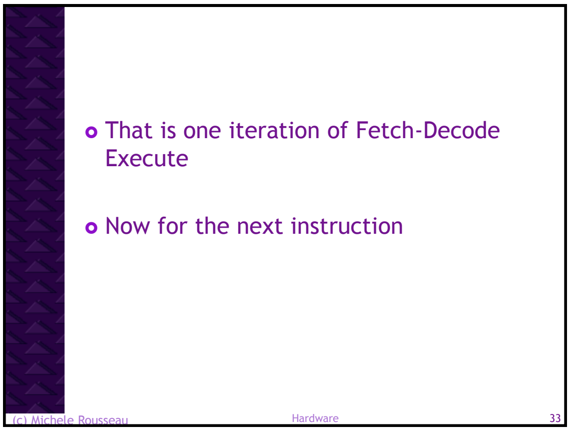


Execute

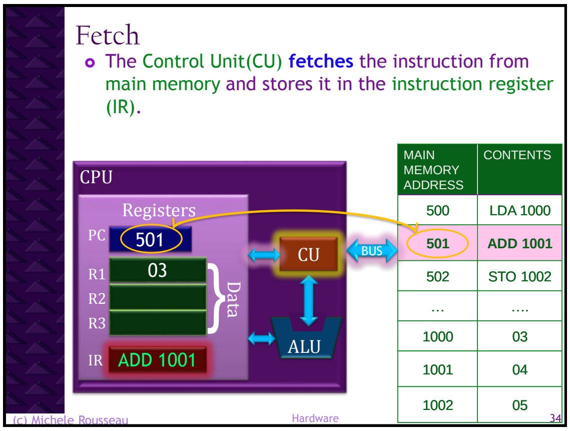
- The Control Unit (CU) **executes** the decoded instruction.

➔ LDA 1000 means to load what is in the address 1000 into a register (R1)

MAIN MEMORY ADDRESS	CONTENTS
500	LDA 1000
501	ADD 1001
502	STO 1002
…	….
1000	**03**
1001	04
1002	05



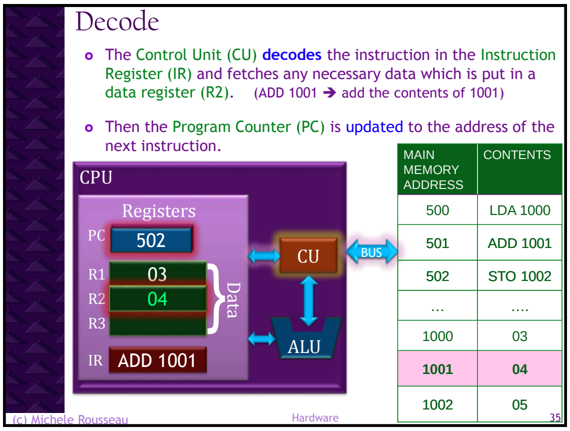
- That is one iteration of Fetch-Decode Execute
- Now for the next instruction



Fetch

- The Control Unit(CU) **fetches** the instruction from main memory and stores it in the instruction register (IR).

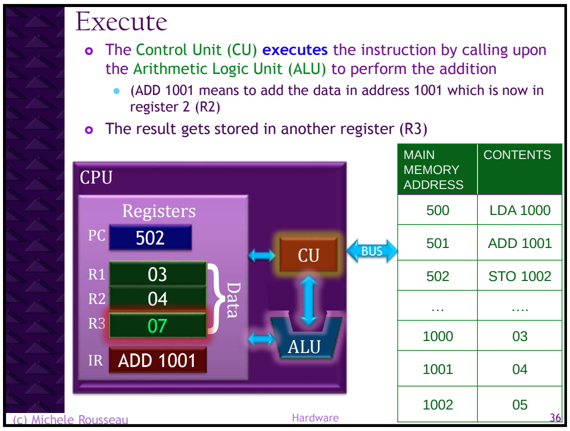
MAIN MEMORY ADDRESS	CONTENTS
500	LDA 1000
501	**ADD 1001**
502	STO 1002
…	….
1000	03
1001	04
1002	05



Decode

- The Control Unit (CU) **decodes** the instruction in the Instruction Register (IR) and fetches any necessary data which is put in a data register (R2). (ADD 1001 ➔ add the contents of 1001)
- Then the Program Counter (PC) is updated to the address of the next instruction.

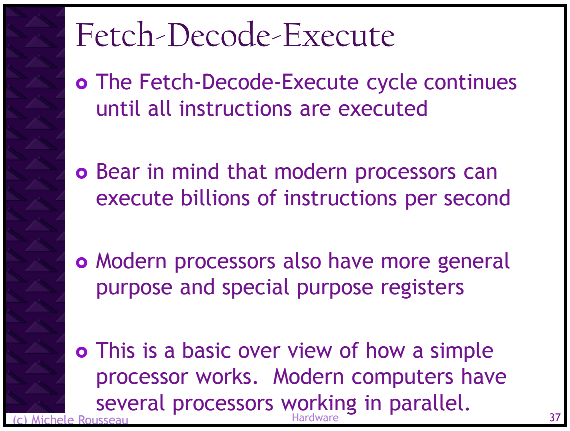
MAIN MEMORY ADDRESS	CONTENTS
500	LDA 1000
501	ADD 1001
502	STO 1002
…	….
1000	03
1001	**04**
1002	05



Execute

- The Control Unit (CU) **executes** the instruction by calling upon the Arithmetic Logic Unit (ALU) to perform the addition
  - (ADD 1001 means to add the data in address 1001 which is now in register 2 (R2)
- The result gets stored in another register (R3)

MAIN MEMORY ADDRESS	CONTENTS
500	LDA 1000
501	ADD 1001
502	STO 1002
…	….
1000	03
1001	04
1002	05



Fetch-Decode-Execute

- The Fetch-Decode-Execute cycle continues until all instructions are executed
- Bear in mind that modern processors can execute billions of instructions per second
- Modern processors also have more general purpose and special purpose registers
- This is a basic over view of how a simple processor works. Modern computers have several processors working in parallel.