Machine Organization

Objectives

At the end of the meeting, students should be able to:

- Describe the computer architecture; and
- Explain how the computer handles machine instructions.

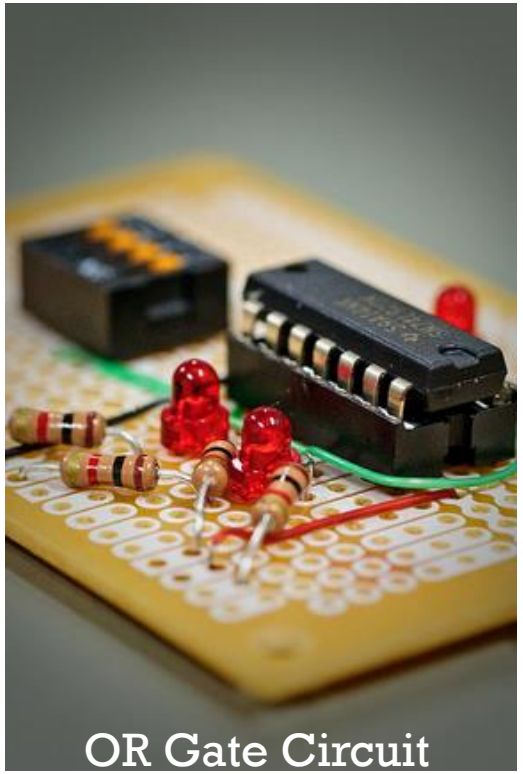
CPU

- Central Processing Unit
  - The heart of a computer
  - The circuitry that controls the manipulation of data and execution of instructions

Gates

- The CPU is amazingly small given the immense amount of circuitry it contains.

Circuits of a computer are made of **gates.**



OR Gate Circuit

Transistor

- The CPU is amazingly small given the immense amount of circuitry it contains.

  Gates, however are also made of another tiny component called a **transistor**, and a modern CPU has millions of transistors in its circuitry.

Moore's Law

- According to Moore's Law, the transistor count of the integrated circuits **doubles every two years**.

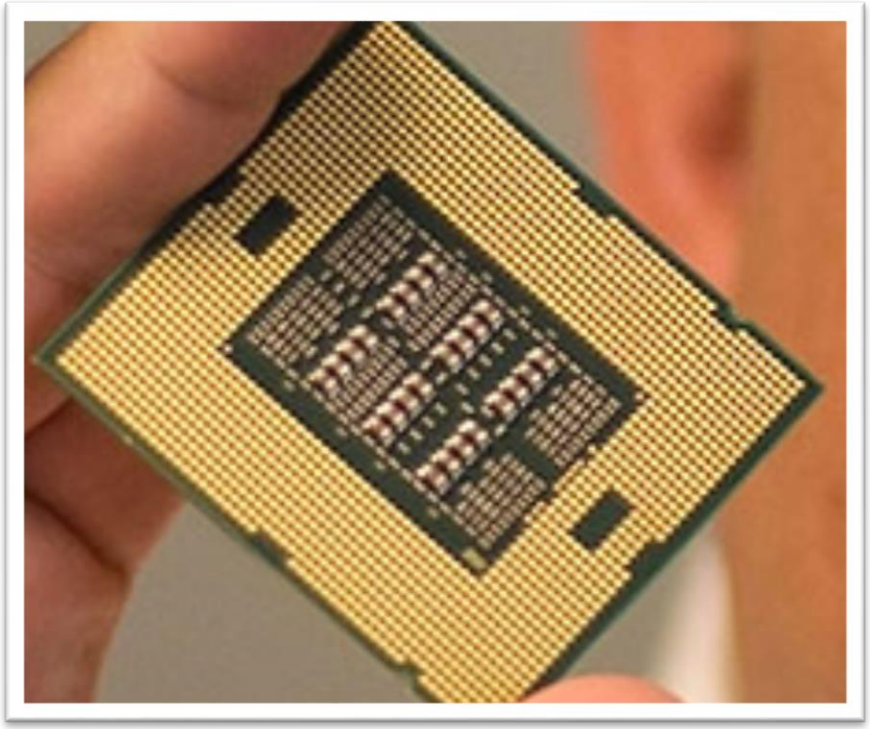
Transistor count

Processor	Transistor count	Date of introduction	Manufacturer
16-Core SPARC T3	1,000,000,000	2010	Sun/Oracle
Six-Core Core i7	1,170,000,000	2010	Intel
8-core POWER7	1,200,000,000	2010	IBM
Quad-core z196	1,400,000,000	2010	IBM
Dual-Core Itanium 2	1,700,000,000	2006	Intel
Quad-Core Itanium Tukwila	2,000,000,000	2010	Intel
8-Core Xeon Nehalem-EX	2,300,000,000	2010	Intel
10-Core Xeon Westmere-EX	2,600,000,000	2011	Intel

Source: http://en.wikipedia.org/wiki/Transistor_count

Intel's 10-Core Xeon Processor

Shipping Soon



ALU and Control Unit

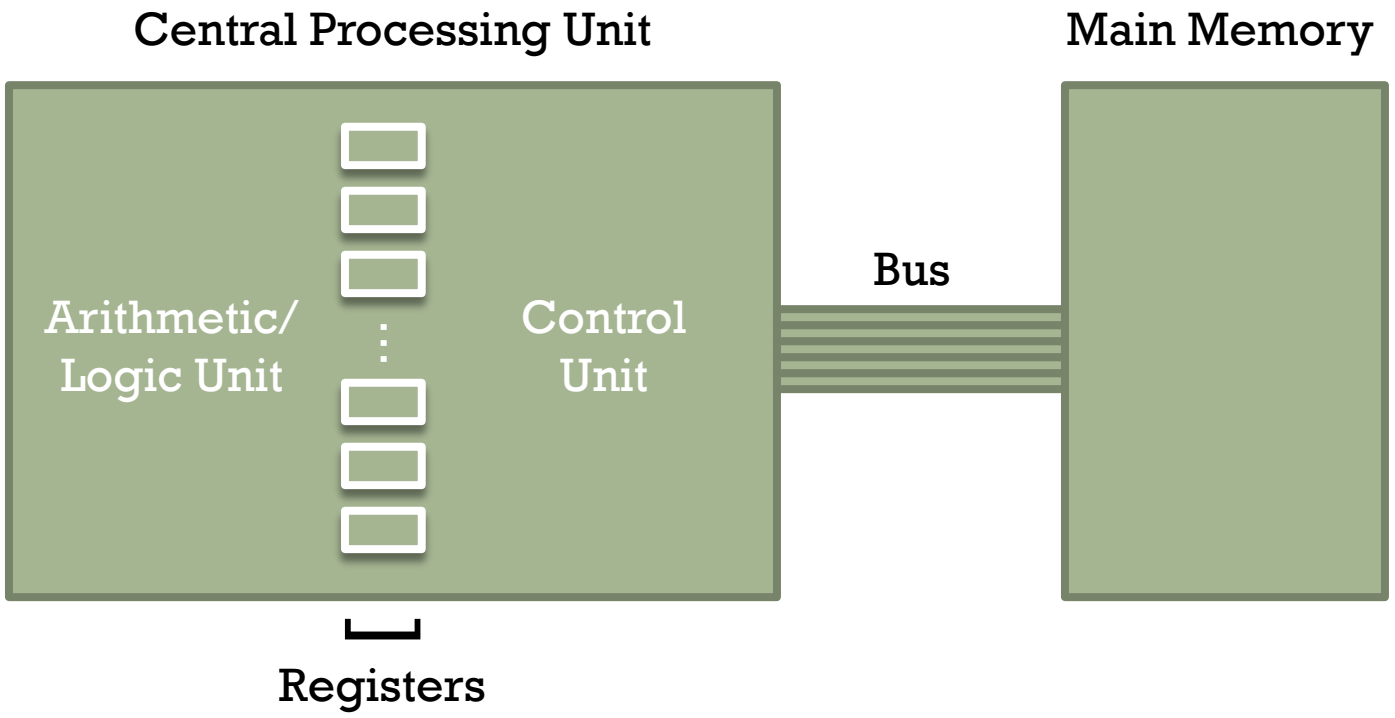
The CPU consists of two parts:

- **Arithmetic Logic Unit**, which contains the circuitry that performs operations on data (+,-,/,*).
- **Control Unit**, contains the circuitry for coordinating the computer's activities.

Registers and Buses

- **Registers**- temporary storage inside the CPU
  - General Purpose Registers
    - Serves as temporary holding places for data being manipulated by the CPU.
  - Special Purpose Registers
- **Bus**
  - A collection of wires that connects the CPU to the main memory

Computer Architecture



CPU and main memory connected via a bus

Adding two numbers

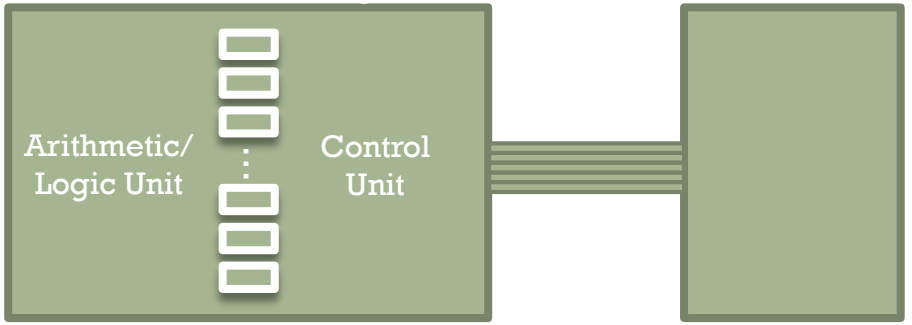
Step 1: Get one of the values to be added from memory and place it in a register.

Step 2: Get the other value to be added from memory and place it in another register.

Step 3: Activate the addition circuitry with the registers used in Steps 1 and 2 as inputs and another register designated to hold the result.

Step 4: Store the result in memory.

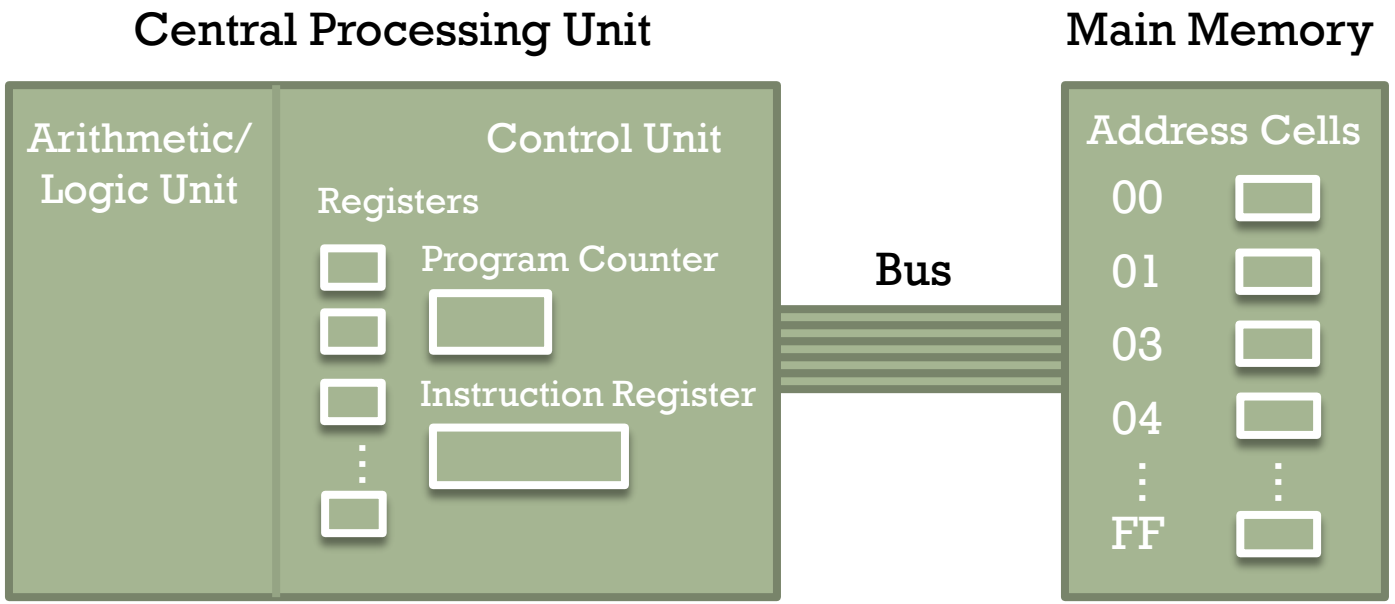
Step 5: Stop



Special Purpose Registers

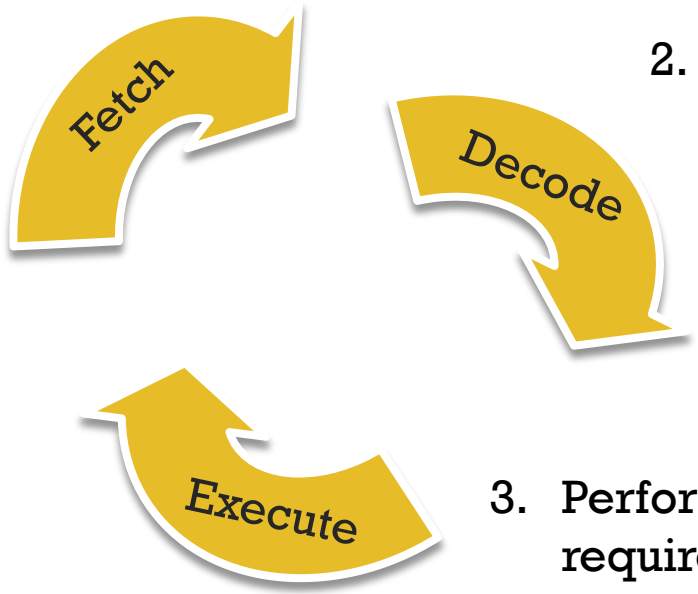
- Inside the computer are two special purpose registers
  - **Instruction Register**
    - Used to hold the instruction being executed
  - **Program Counter**
    - Contains the address of the next instruction to be executed.

Computer Architecture



The Machine Cycle

1. Retrieve the next instruction from memory (as indicated by the program counter) and then increment the program counter.

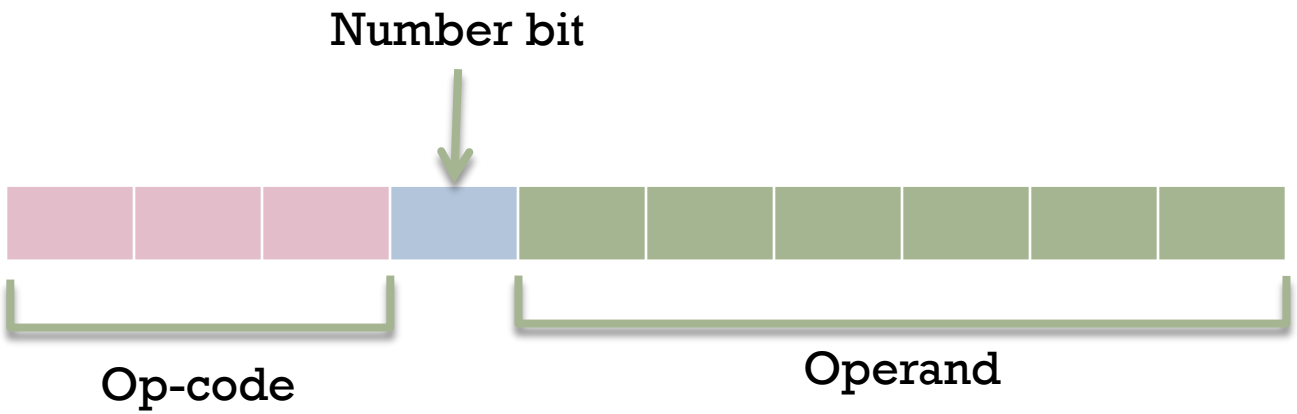


2. Decode the bit pattern in the instruction register.

3. Perform the instruction required by the instruction in the instruction register.

Machine Instruction

- Each machine instruction is composed of two parts: the **op-code** and the **operand**.



Machine Instruction

- **Op-code** field indicates which of the elementary operations, such as STORE or JUMP, is requested by the instruction
- **Operand** field provide detailed information about the operation specified by the op-code
- The **middle bit** distinguishes between operands that are memory addresses and operands that are numbers. When the bit is set to '1', the operand represents a number.

Assembly Language

- The op-codes are given an English **mnemonic** to simplify programming. Together these mnemonics are called an assembly language.

A Simple Machine Language

Op-code	Mnemonic	Function	Example
001	LOAD	Load the value of the operand into the Accumulator	LOAD 10
010	STORE	Store the value of the Accumulator at the address specified by the operand	STORE 8
011	ADD	Add the value of the operand to the Accumulator	ADD #5
100	SUB	Subtract the value of the operand from the Accumulator	SUB #1
101	EQUAL	If the value of the operand equals the value of the Accumulator, skip the next instruction	EQUAL #20
110	JUMP	Jump to a specified instruction by setting the Program Counter to the value of the operand	JUMP 6
111	HALT	Stop execution	HALT

Sum Program

#	Machine code	Assembly code	Description
0	001 1 000010	LOAD #2	Load the value 2 into the Accumulator
1	010 0 001101	STORE 13	Store the value of the Accumulator in memory location 13
2	001 1 000101	LOAD #5	Load the value 5 into the Accumulator
3	010 0 001110	STORE 14	Store the value of the Accumulator in memory location 14
4	001 0 001101	LOAD 13	Load the value of memory location 13 into the Accumulator
5	011 0 001110	ADD 14	Add the value of memory location 14 to the Accumulator
6	010 0 001111	STORE 15	Store the value of the Accumulator in memory location 15
7	111 0 000000	HALT	Stop execution