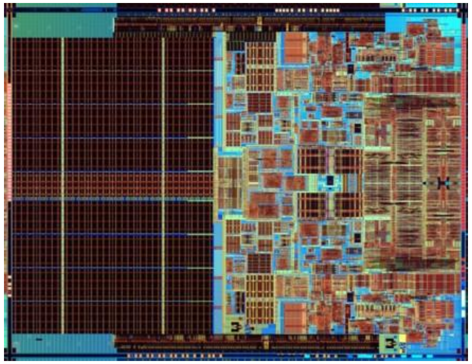
CSI 312 - Computer Architecture

Lecture 1 - Course Overview

Fall 2011

Reading: 1.1-1.3



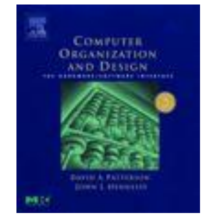
Intel Core 2 Duo Processor
Image courtesy Intel Corporation

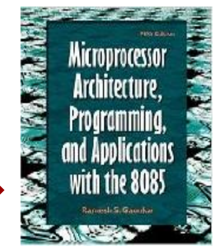
Outline - Course Overview

- Administrative Details
- Computer Systems Overview
  - Types of Computer Systems
  - High-Level Organization - The “5 classic components”
  - High-Level Operations - the “fetch/execute cycle”
  - Common Abstractions
  - “Under the Hood” of some example computer systems
- Course Overview
  - Roadmap - subjects to be covered
  - Course Objectives

Textbook and References

- **Textbooks:**
  - **David A. Patterson and John L. Hennessy, *Computer Organization and Design, 4th Edition,* Morgan-Kafumann.**
  - **R. S. Gaonkar. Microprocessor architecture (Programming, and Applications), 5th edition, Prentice Hall 2003.**





- **References:**
  - **John L. Hennessy and David A Patterson, *Computer Architecture-A Quantitative Approach*, *3rd Ed.*, Morgan-Kaufmann, 2002.**

Administrative Details

- **Grading**

Attendance	05%
Problem sets	10%
Quizzes and Participation	10%
Projects	15%
Exam 1	10%
Exam 2	10%
Midterm Exam	15%
Final Exam	25%

- **My Schedule**
  - **Classes**
    - CSI 312 TTH 08:00-09:15
    - CSI 212 TTH 02:00-3:15
    - CSI 319 TTH 3:30-4:45
    - CSI 250 TTH 5:00 – 6:15
  - **Office Hours: TTH 1:00 – 2:00 & by appointment**

“Official” Prerequisite – CSI 211

- **Introduction to Object Oriented Programming**
  - **Variables, Primitive Data Types, and Expressions**
  - **Assignments, Conditionals, Loops, etc.**
  - **Classes, Objects, & Methods**
  - **Basic Input/Output**
  - **Arrays & Basic Data Structures**

Course Objectives

- Students should be able to...
  - Describe high-level organization of computer systems
  - Understand representation of instructions in memory
  - Understand the fetch/execute cycle
  - Understand the concept of Instruction Set Architecture
  - Understand how computers represent data
  - Understand memory organization
  - Understand input/output
  - implement assembly language programming

Roadmap for the Term: Major Topics

- **Computer Systems Overview** ◂
- **Technology Trends**
- **Instruction Sets (and Software)**
- **Logic and Arithmetic**
- **Performance**
- **Processor Implementation**
- **Memory Systems**
- **Input/Output**

Outline - Course Overview

- **Administrative Details**
- **Computer Systems Overview**
  - **Classes of Computer Systems**
  - **High-Level Organization - The “5 classic components”**
  - **High-Level Operations - the “fetch/execute cycle”**
  - **Common Abstractions**
  - **“Under the Hood” of some example computer systems**
- **Course Overview**
  - **Roadmap - subjects to be covered**
  - **Course Objectives**
  - **Project Details**

Classes of Computer Systems

Desktop

Server

Embedded

Image sources:
Dell Computer www.dell.com
Rackable Systems www.rackablecom
Apple Computer www.apple.com

Desktop Computer Systems

- For "General-Purpose" Use
  - Word-Processing, Web surfing, Multimedia, etc.
  - Computation and Programming
- What's in the box
  - Microprocessor
  - Memory - Synchronous DRAM
  - Hard disk(s), CDROM/DVD, etc.
  - I/O - mouse, keyboard, video card, monitor, network, etc.

- Important Issues:
  - Performance - how fast is "fast enough"?
  - Basic capabilities (and expandability)
  - Cost

Server Computer Systems

- Large-Scale Services
  - File storage
  - Computation (e.g., supercomputers)
  - Transaction Processing, Web
- What's in the Box(es)
  - Microprocessor(s)
  - Hard disks
  - Network Interface(s)
- Important issues:
  - Performance
  - Reliability, availability
  - Cost

One Rack-Mount PC Unit
(Google uses ~ 10,000)

Embedded Computer Systems

- Computer as part of larger system
  - Consumer electronics, appliances
  - Networking, telecommunications
  - Automotive / aircraft control

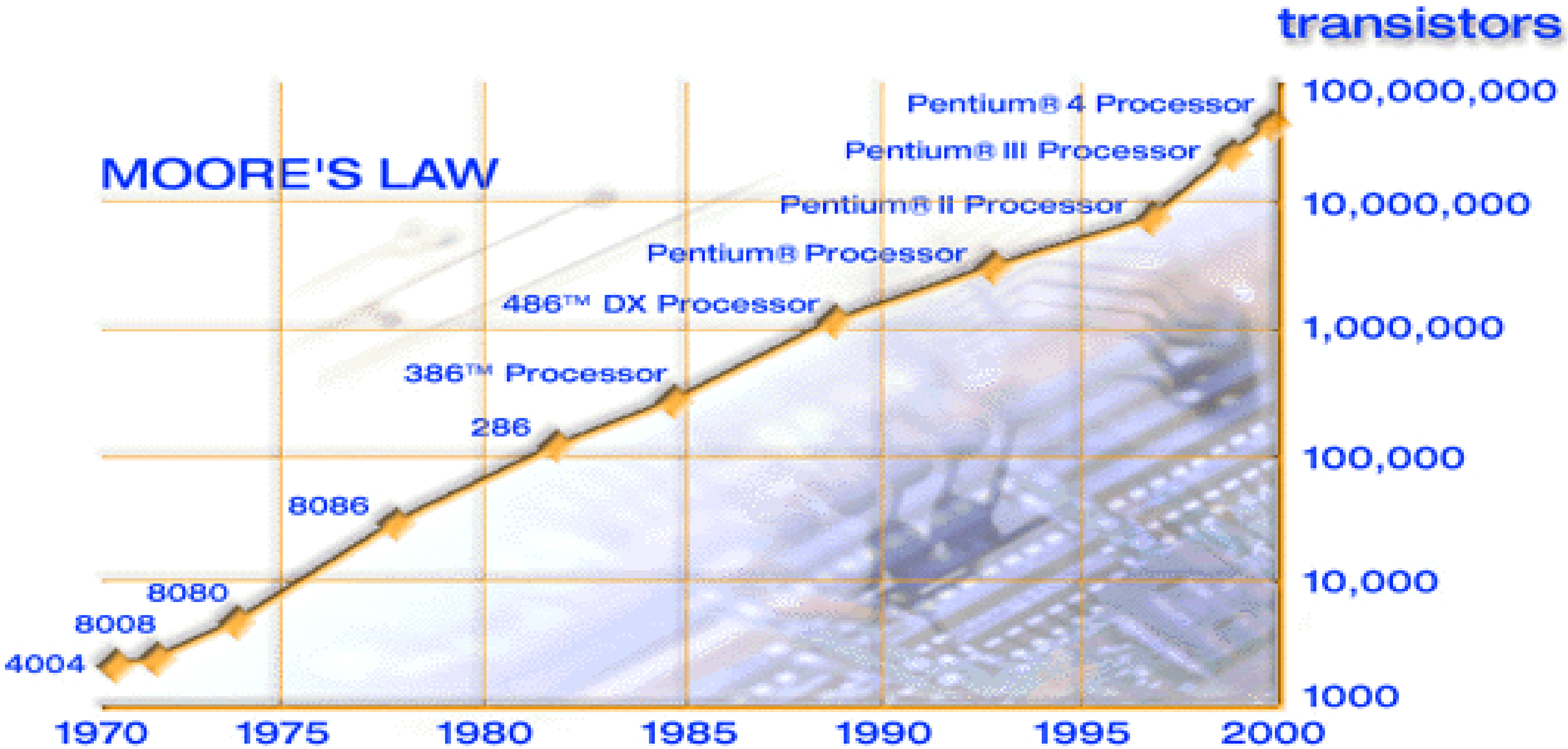
- What's in the box
  - Microcontroller / Microprocessor / System on Chip (SOC)
  - Memory: RAM, ROM; Disk
  - Special-purpose I/O (including analog stuff)
- Important issues
  - Cost, Power Consumption
  - Performance (against real-time constraints)
  - Reliability and Safety

Therefore We Find a Computer/Processor in

Computer Technology - Dramatic Change!

- Processor
  - 2X in speed every 1.5 years (since ‘85); 100X performance increase in last decade.
- Memory
  - DRAM capacity: 2x / 2 years (since ‘96); 64x size improvement in last decade.
- Disk
  - Capacity: 2X / 1 year (since ‘97) 250X size increase in last decade.

Tech. Trends: Microprocessor Complexity



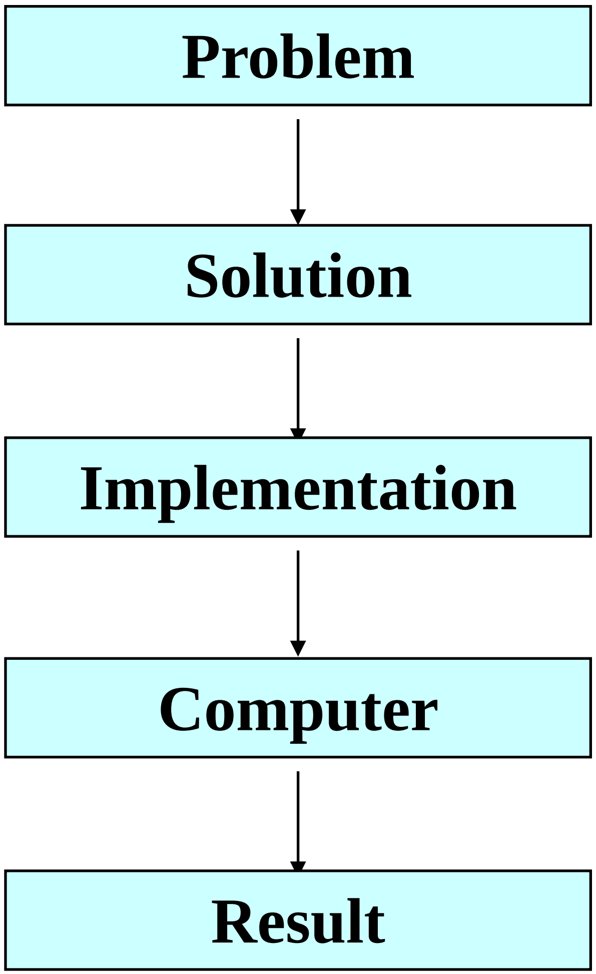
2 * transistors/Chip Every 1.5 to 2.0 years

Called "<u>Moore's Law</u>"

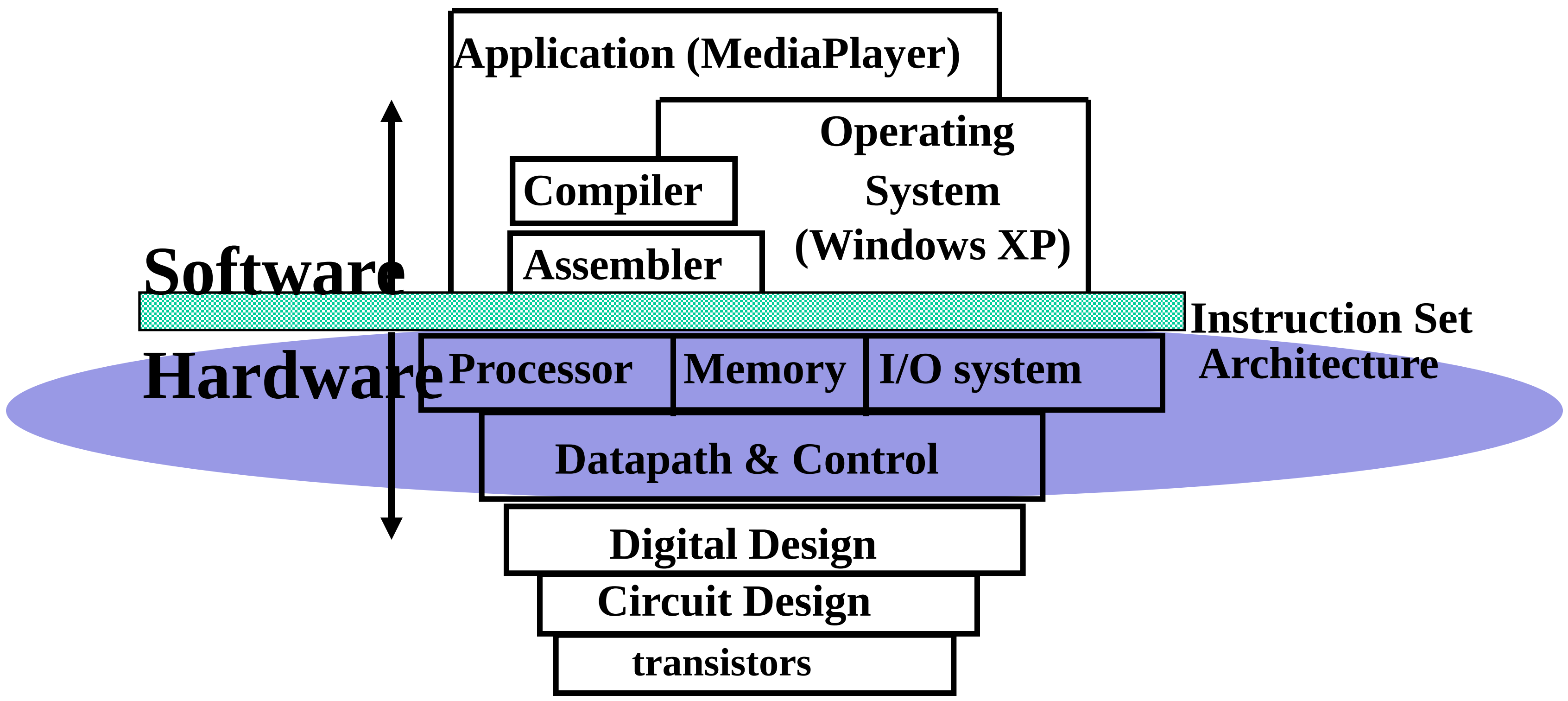
We use computers to find solutions to problems

While thinking of a solution, think about:

- **Cost $$$**
- **Speed**
- **Energy/Power**
- **Size**
- **Efficiency**
- **etc…**

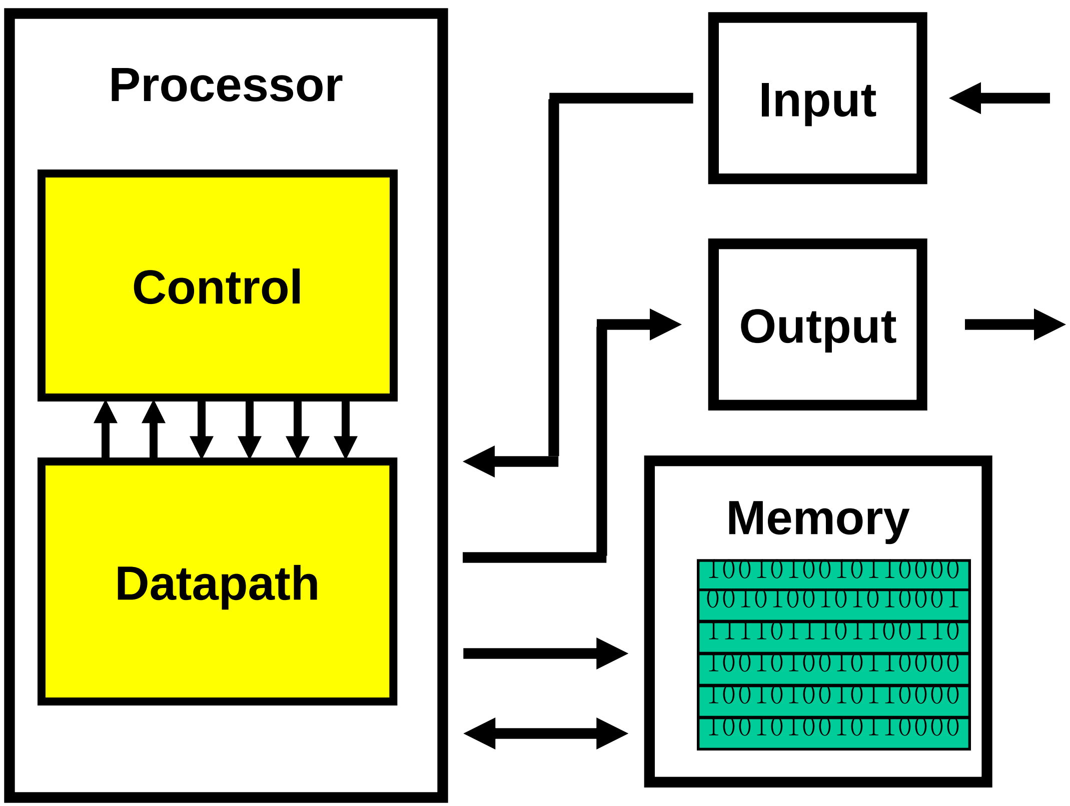


Where is "Computer Architecture"?



Computer System Organization

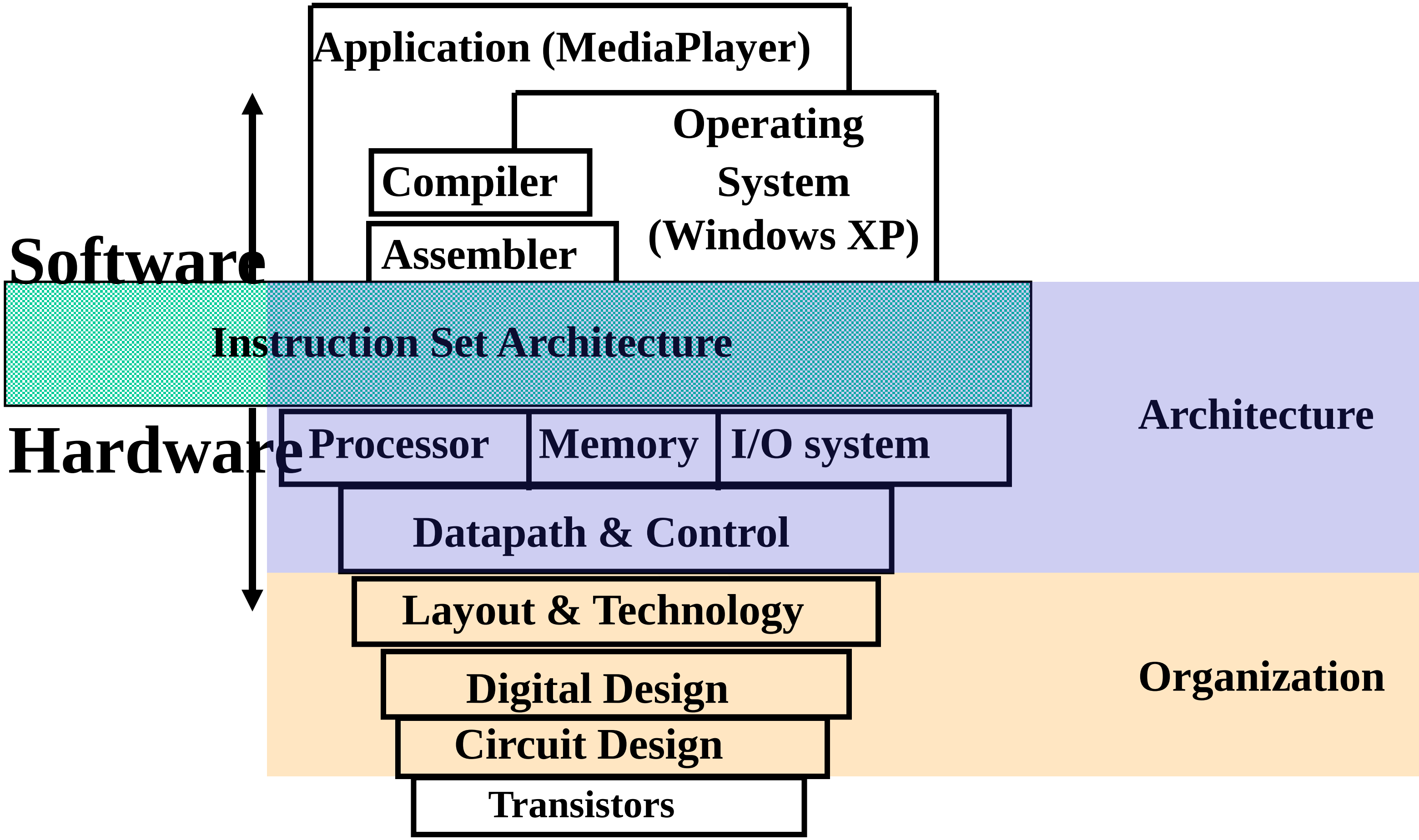
- "Five classic components"



Computer Architecture & Organization

- Computer Architecture
  - What the “low level” programmer sees
    - Types of Instructions
    - Number of Registers
    - Types of Operations
- Computer Organization
  - How the designer implements the Design
    - Layout
    - Interconnection (wires)

Computer Architecture & Organization



Architecture & Organization

- **Architecture is those attributes visible to the programmer**
  - **Instruction set, number of bits used for data representation, I/O mechanisms, addressing techniques.**
  - **e.g. Is there a multiply instruction?**

- **Organization is how features are implemented**
  - **Control signals, interfaces, memory technology.**
  - **e.g. Is there a hardware multiply unit or is it done by repeated addition?**

Architecture vs. Organization

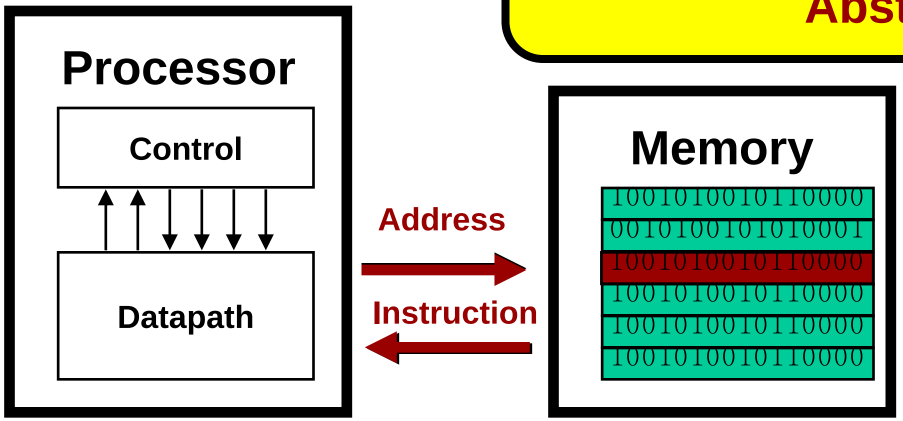
- **Architecture: features visible to programmer**
  - **Registers and memory model**
  - **Data types**
  - **Instructions**
- **Organization: system implementation**
  - **Processor design: Datapath, Control, "microarchitecture"**
  - **System design: Processor + Memory, I/O**
  - **The interaction between components (wires)**

Computer System Operation

- Executing Programs - the "fetch/execute" cycle
  - Processor **fetches** instruction from memory
  - Processor **executes** "machine language" instruction
    - Load Data
    - Perform Calculation
    - Store Results

next instr

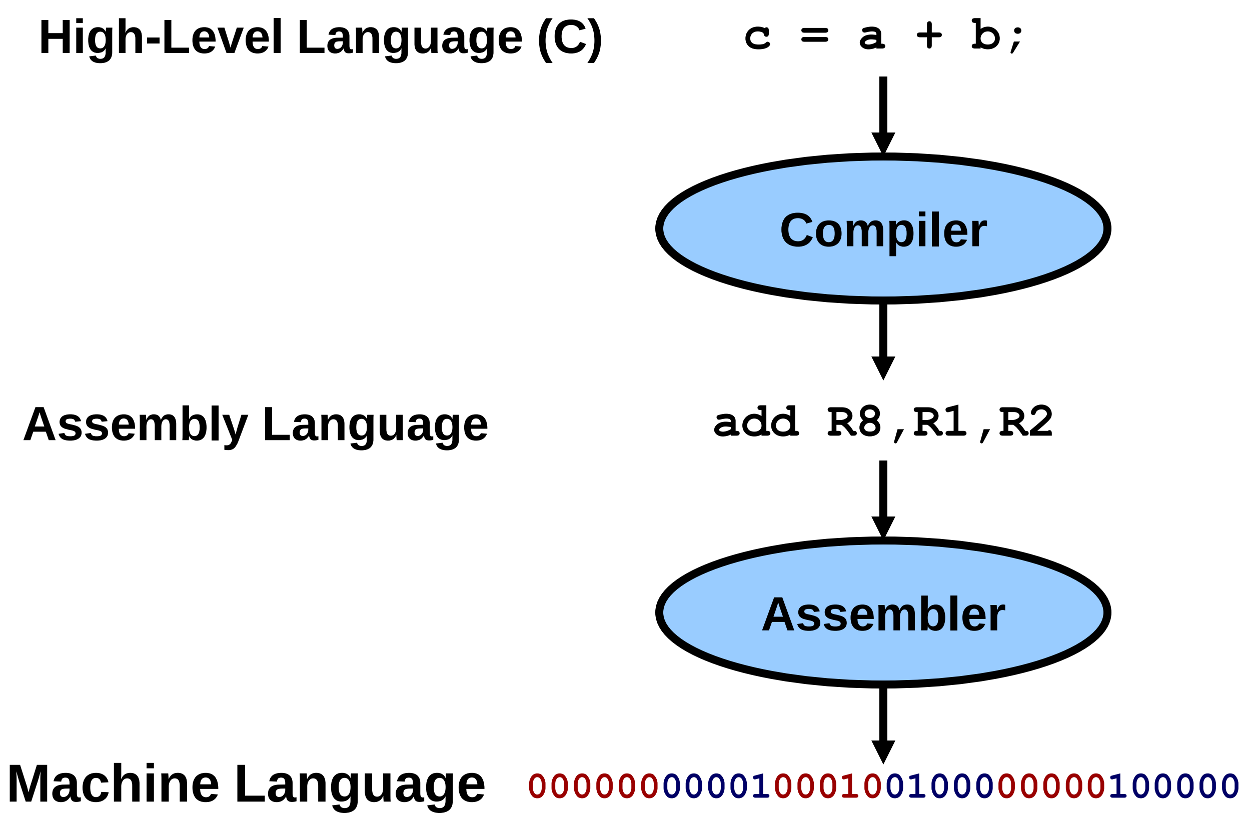
OK, but how do we write useful programs using these instructions? Abstraction!



Abstractions in Computer Systems

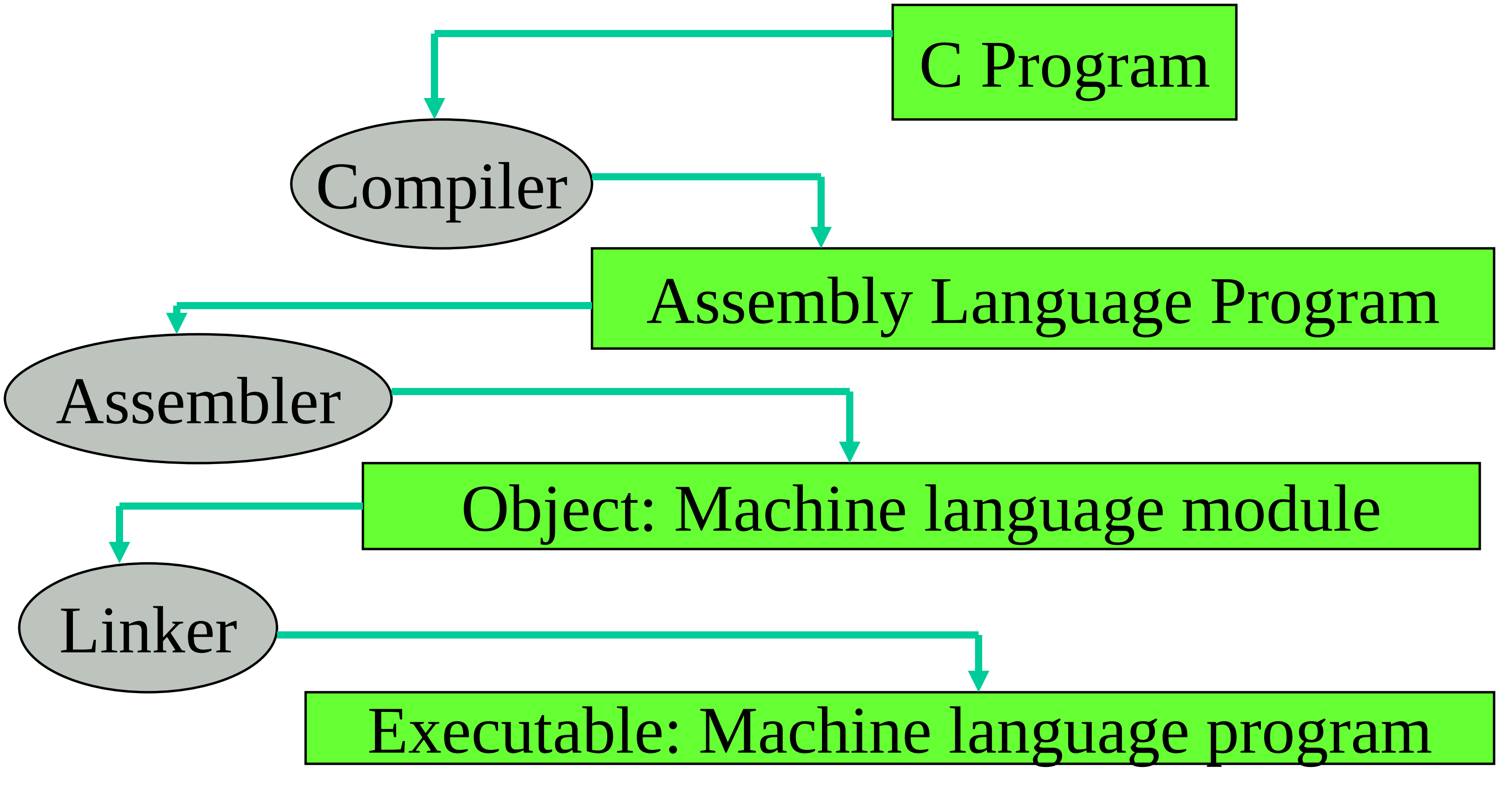
- Designers use abstraction to manage complexity
  - Focus on relevant information
  - Suppress unnecessary detail

Software Abstractions - Languages



Software Abstractions - Languages

Translating & Starting a Program



Translating & Optimizing a Program: *The Compiler*

The *Compiler* transforms the C program into an assembly language program, a symbolic form of what the machine understands.

Translating a Program: *The Assembler*

The *Assembler* transforms the Assembly program into a machine language module.

Stitching a Program: *The Linker*

The ***linker*** or ***link editor*** **takes all the independently assembled machine language programs and "stitches" them together.**

There are three steps for the linker:

1) Place code and data modules symbolically in memory

2) Determine the addresses of data and instruction labels

3) Patch both the internal and external references

<u>Example 1</u> (using C, or C++)

a = b + c; // found in .C files

The C-compiler (x) will translate this c-statement to an assembly language instruction:

Add a, b, c ; found in .ASM files
; (a = b + c)

<u>N:B</u>: Each microprocessor has its own compiler since each has its own instruction set.

Add	a	b	c
Opcode	Operand	Operand	Operand

Example 1.1

If the processor we are using does not support instructions with 3 operands (but only 2), the C-compiler (Y) will translate the C-statement

a = b + c;

to different assembly language instructions:

Add b, c ; to add c to b and put result in b
; (b = b + c)

Load a, b ; to load b into a (a = b)

Example 2:

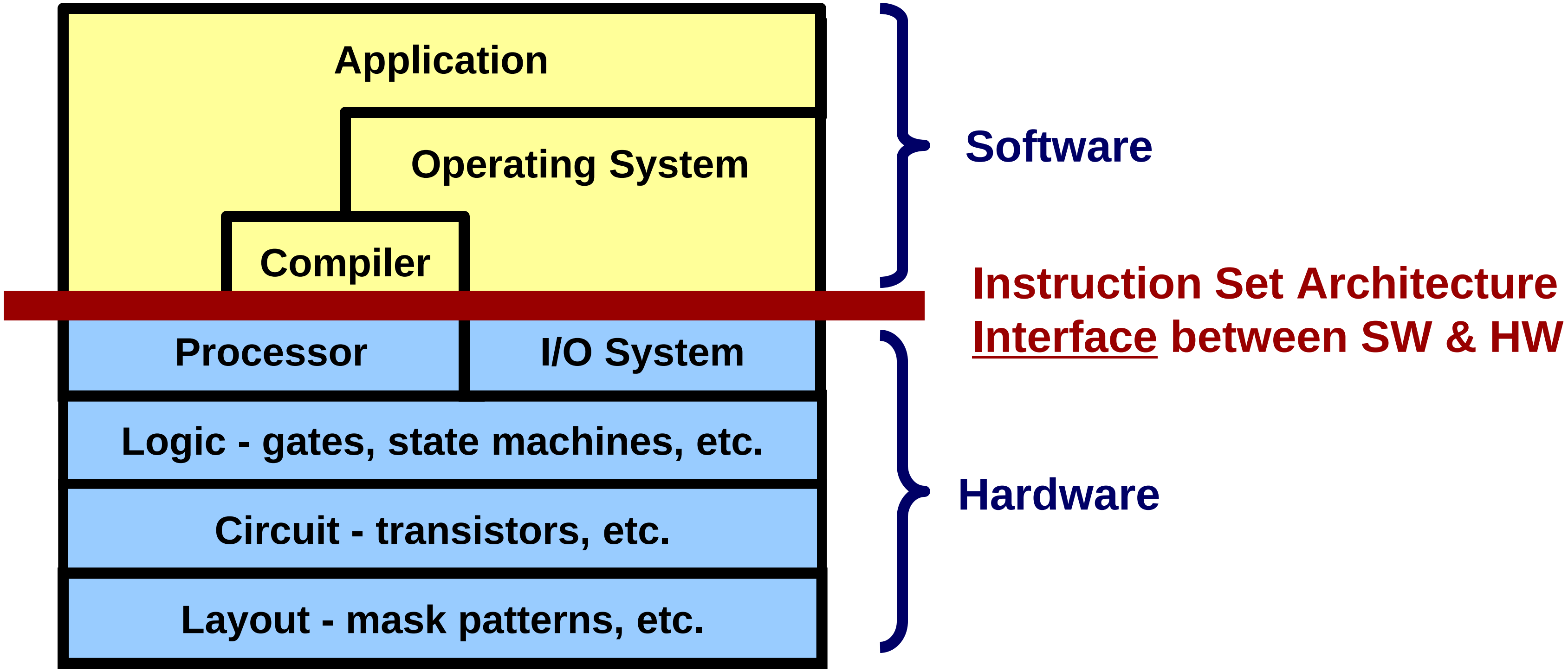
f = (g + h) – (i + j);

The compiler will generate:

```
Add t0, g, h   ; the compiler uses t0 as a
               ;  temporary storage location
Add t1, i, j   ;  the compiler uses t1 as a
               ; second temporary storage location
Sub f, t0, t1  ; finally subtract t1 from t0 and
               ; put the result in f
```

Instruction Set Architecture (ISA) - The Hardware-Software Interface

- The **most important** abstraction of computer design



For an operation to be performed, components should

be Structured in a way to perform a specific Function.

Further, Components should be organized under a specific Architecture.

Components Functions & Structure

- **Function** is the operation of individual components as part of the structure

- **Structure** is the way in which components relate to each other

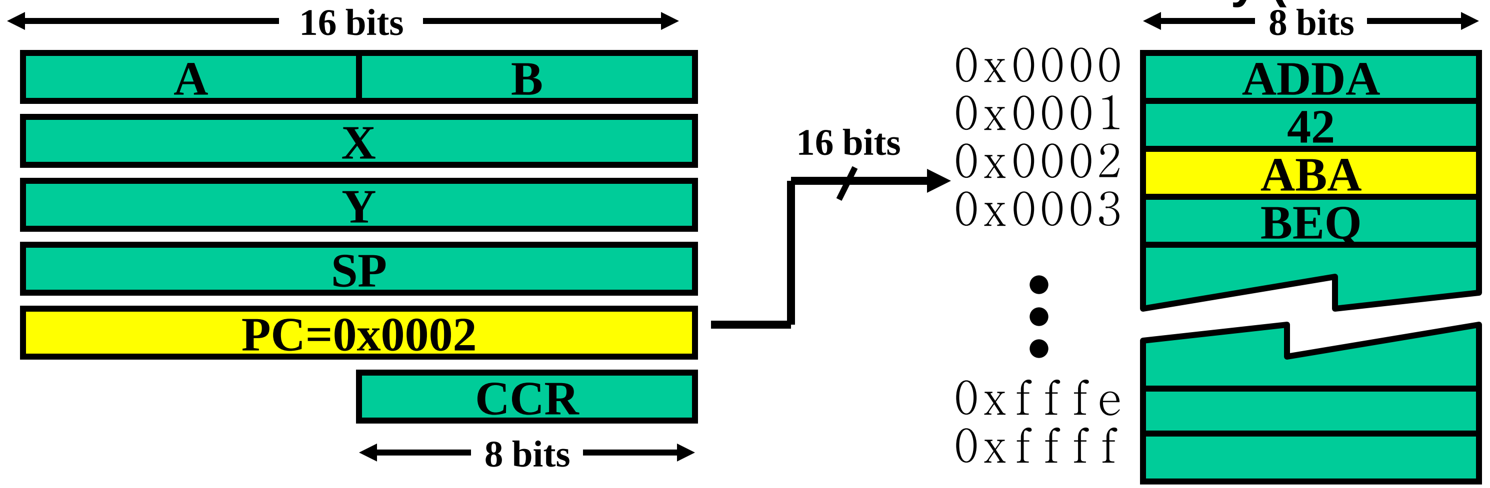
Function

- **All computer functions are:**
  - **Data processing**
  - **Data storage**
  - **Data movement**
  - **Control**

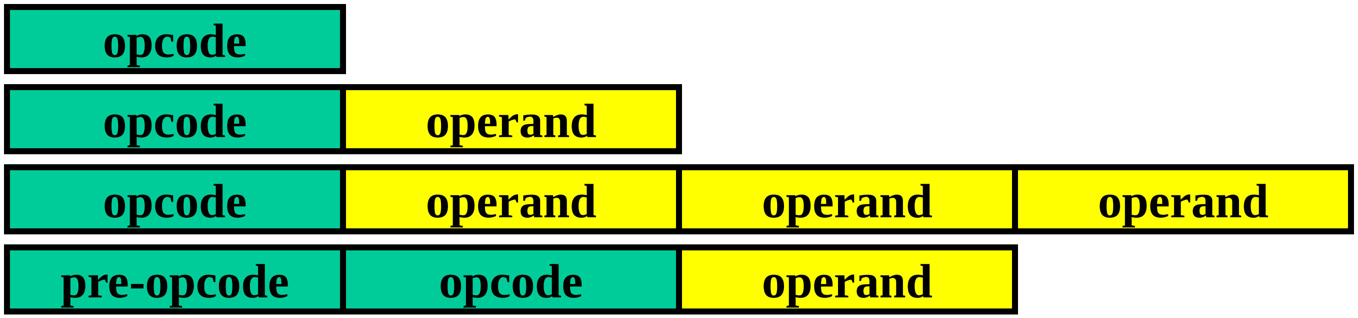
Example Architecture: MC68HC11

Registers

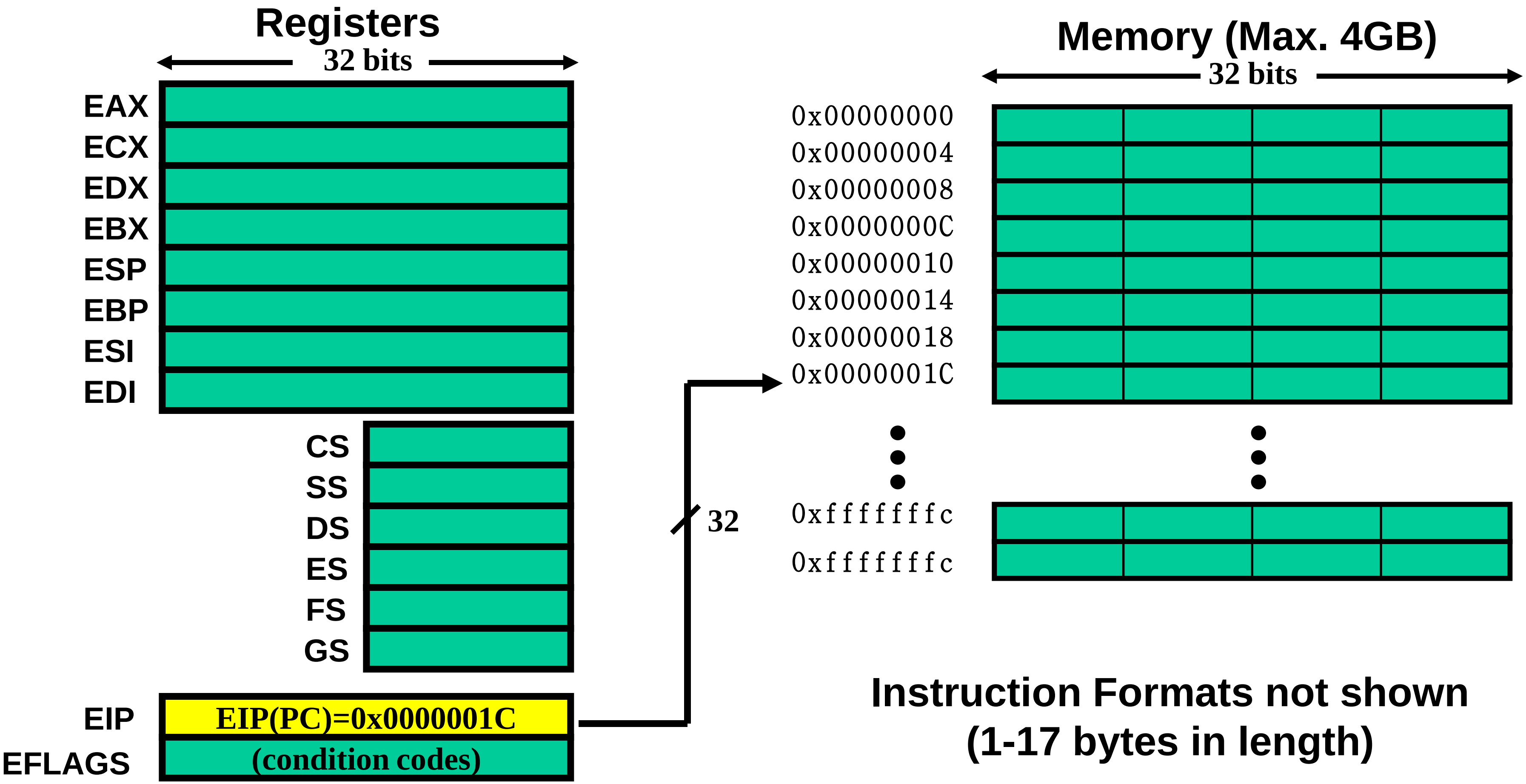
Memory (Max 65KB)



Instruction Formats

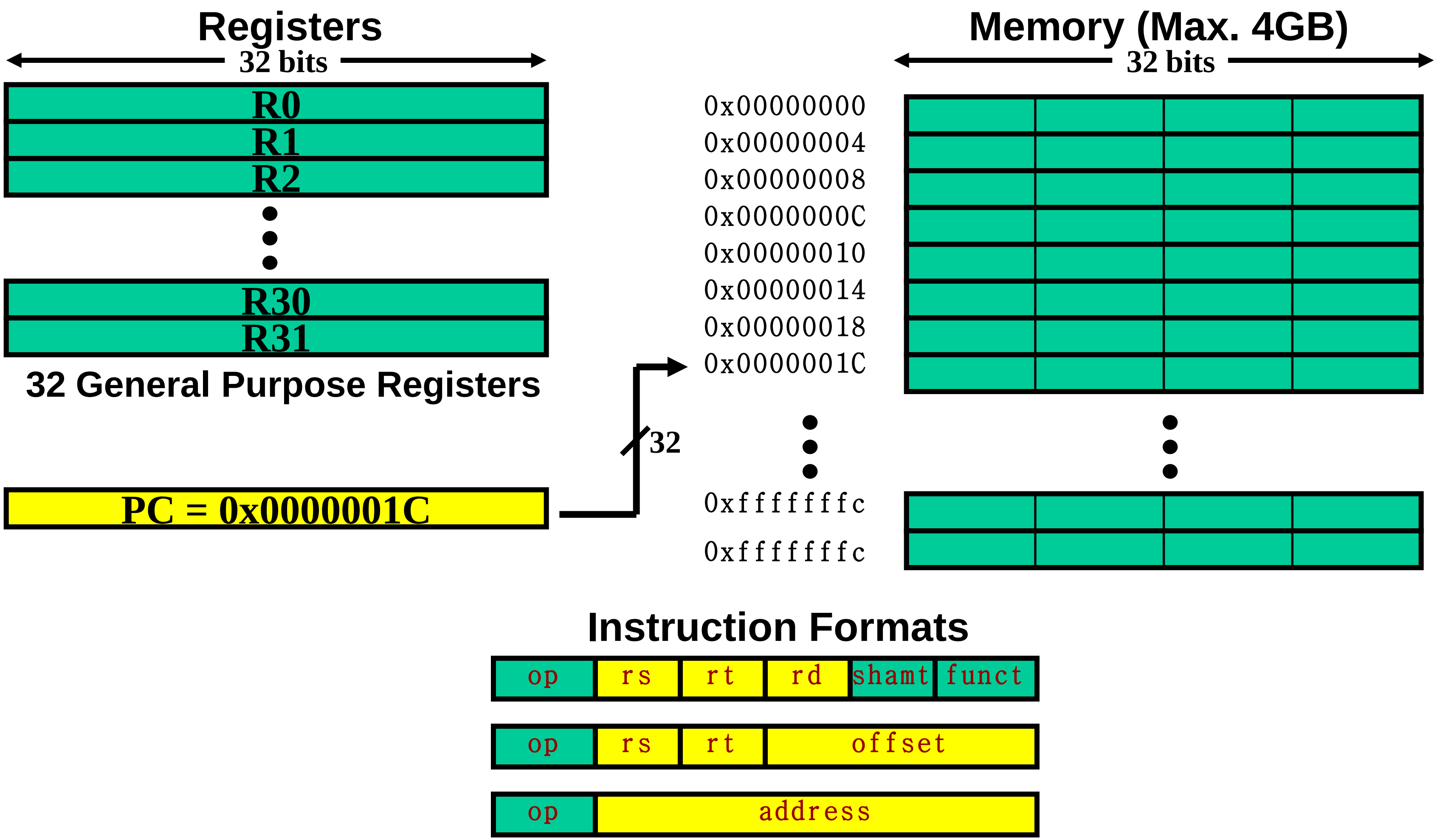


Example Architecture: 80x86 (IA-32)



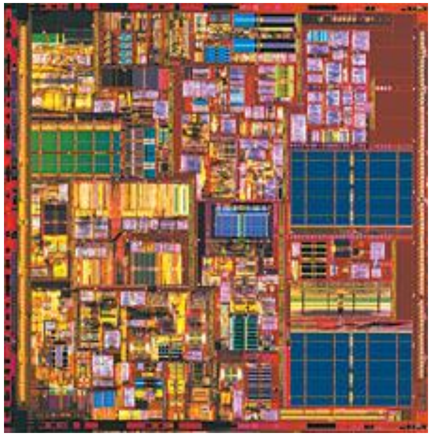
Instruction Formats not shown (1-17 bytes in length)

Example Architecture: MIPS



Under the Hood: The Pentium 4

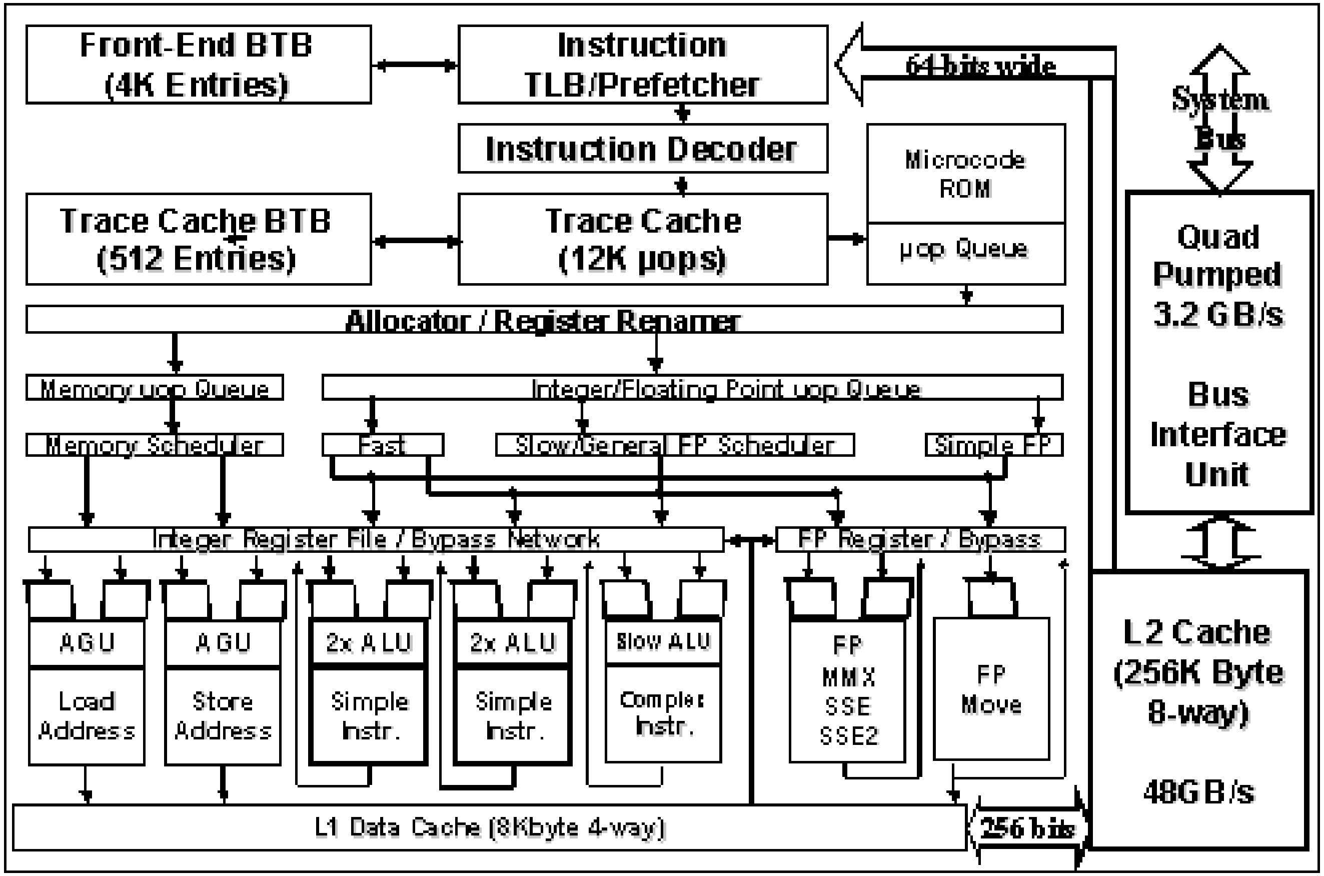
Package



Die Photo

Image sources:
Intel Corporation www.intel.com

Pentium 4 Microarchitecture

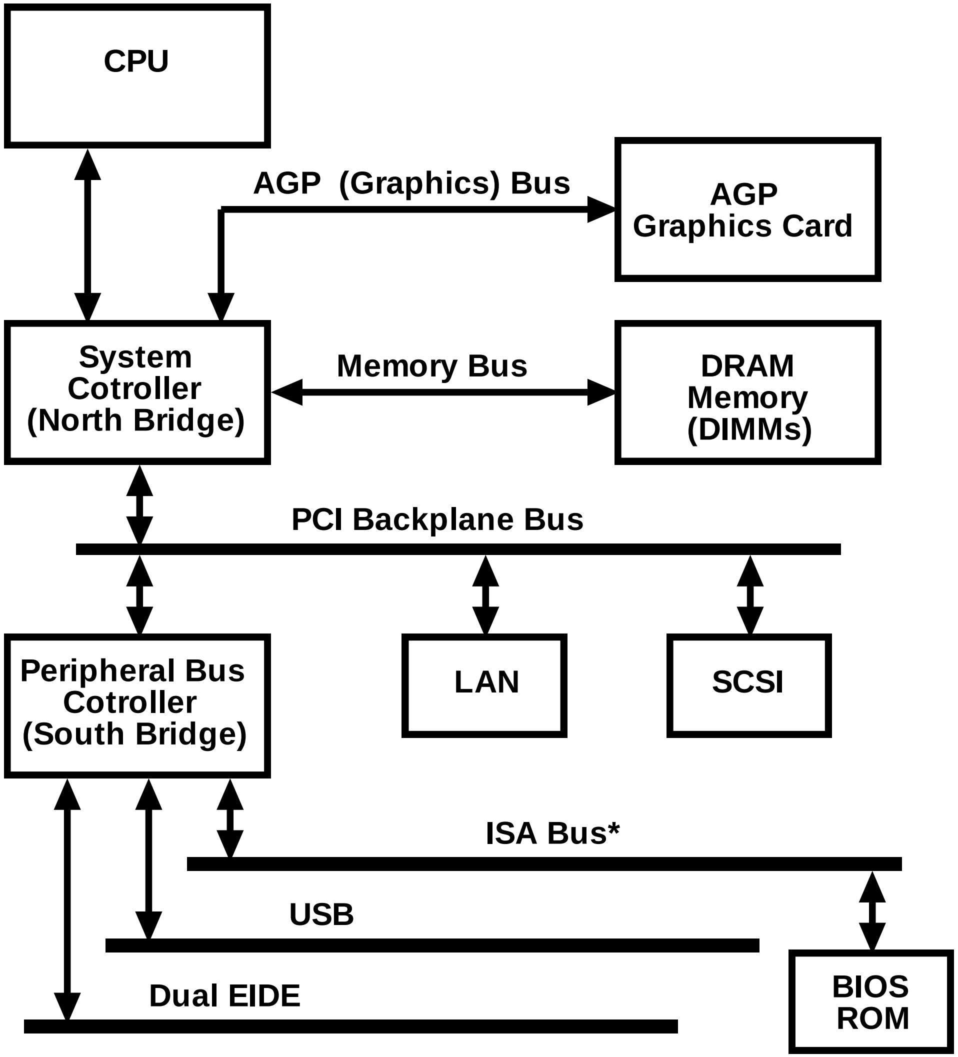


Source: "The Microarchitecture of the Pentium® 4 Processor", *Intel Technology Journal*, First Quarter 2001
`http://developer.intel.com/technology/itj/q12001/articles/art_2.htm.`

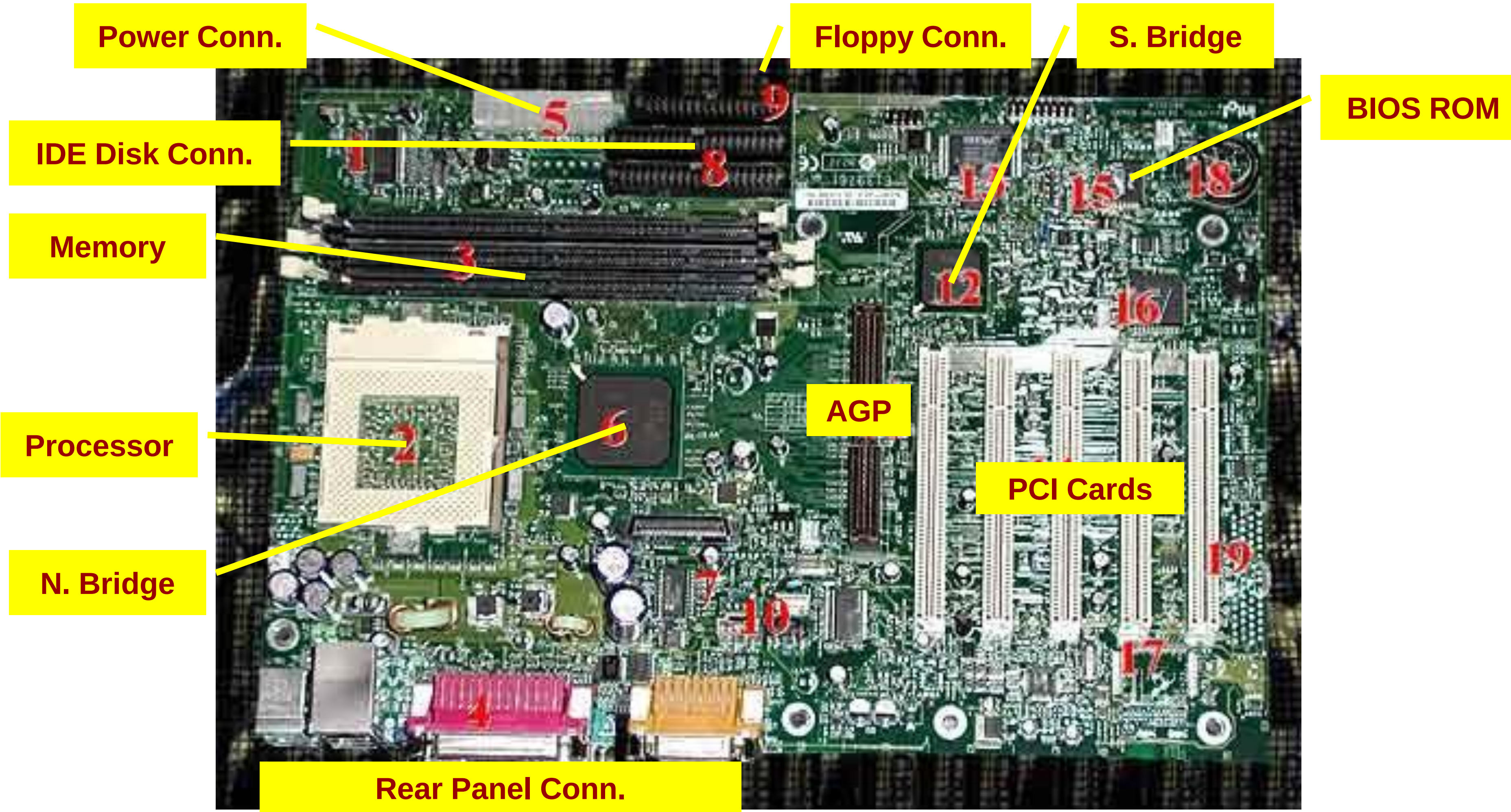
Under the Hood: A Desktop PC

- Display (CRT or LCD)
- Keyboard, Mouse
- "The Box"
  - Power Supply
  - Motherboard (see next slide)
    - Memory
    - Graphics card
    - Standard bus card slots (e.g. PCI)
    - Standard I/O connectors (e.g. USB, Parallel Port, etc)
    - Disks, CDRW, etc.

Organization of a Desktop PC



Typical Motherboard (Pentium III)



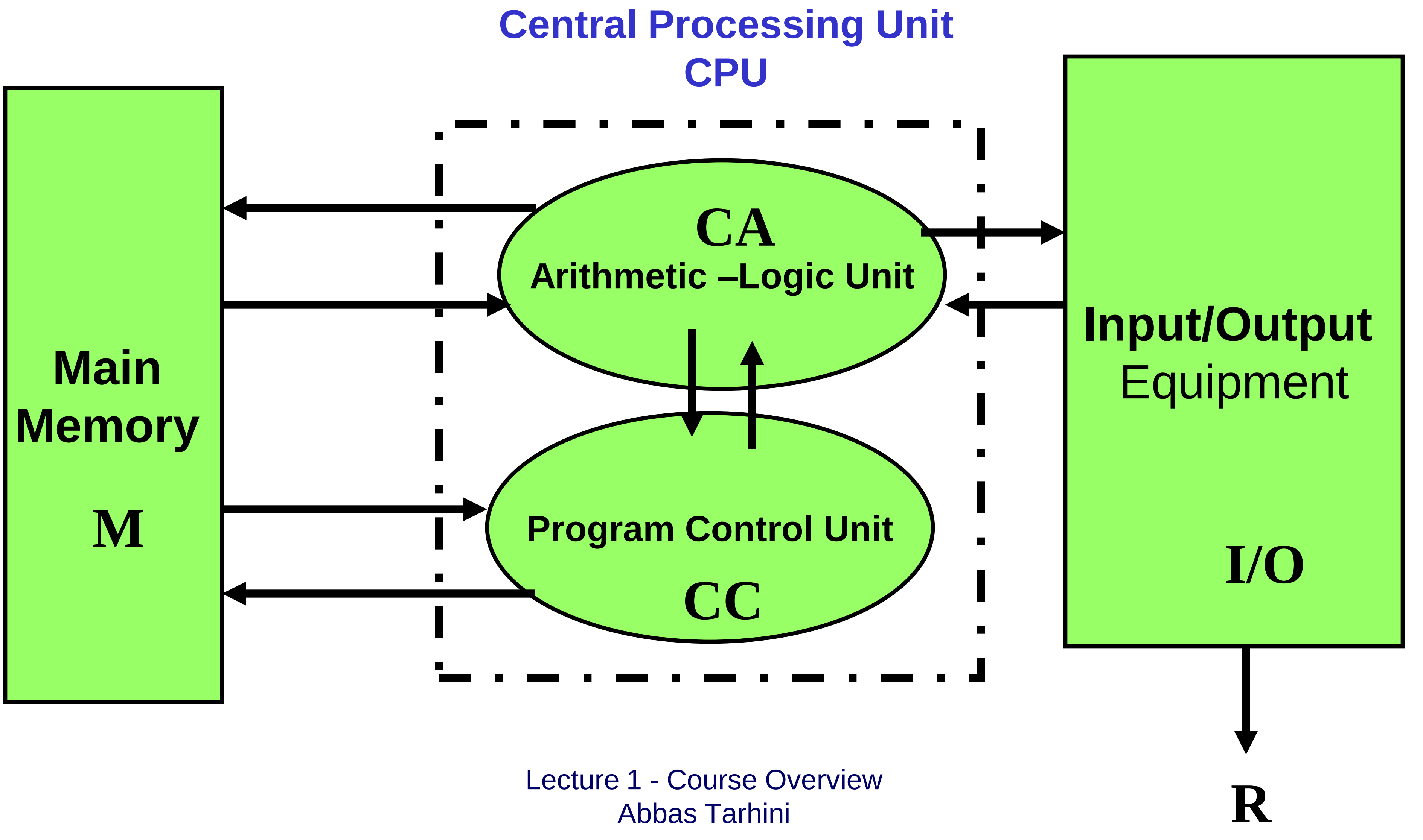
The Quantum Leap:

The von Neumann machine - Completed in 1952

Scientist at the Institute of Advanced Studies (IAS)

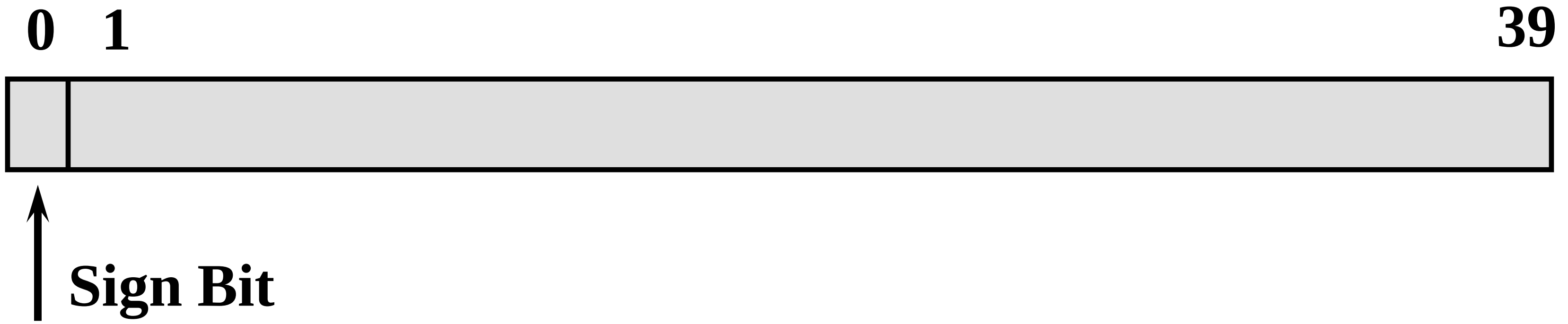
- **Stored Program concept**
- **Main memory storing programs and data**
- **ALU operating on binary data**
- **Control unit interpreting instructions from memory and executing them**
- **Input and Output equipment operated by control unit**

Structure of von Neumann Machine



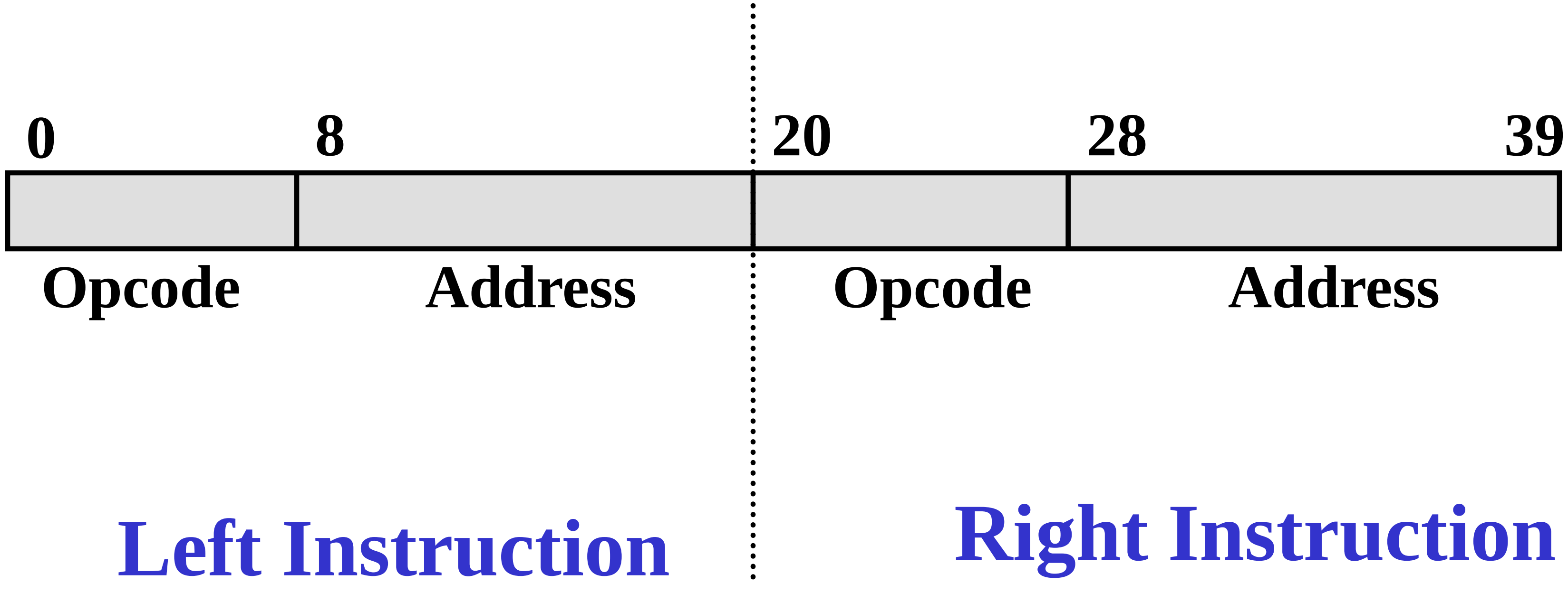
Structure of WORD (1)

- 1000 Storage locations, WORDS
  - 40 Binary Digits (bits)
  - Number word:

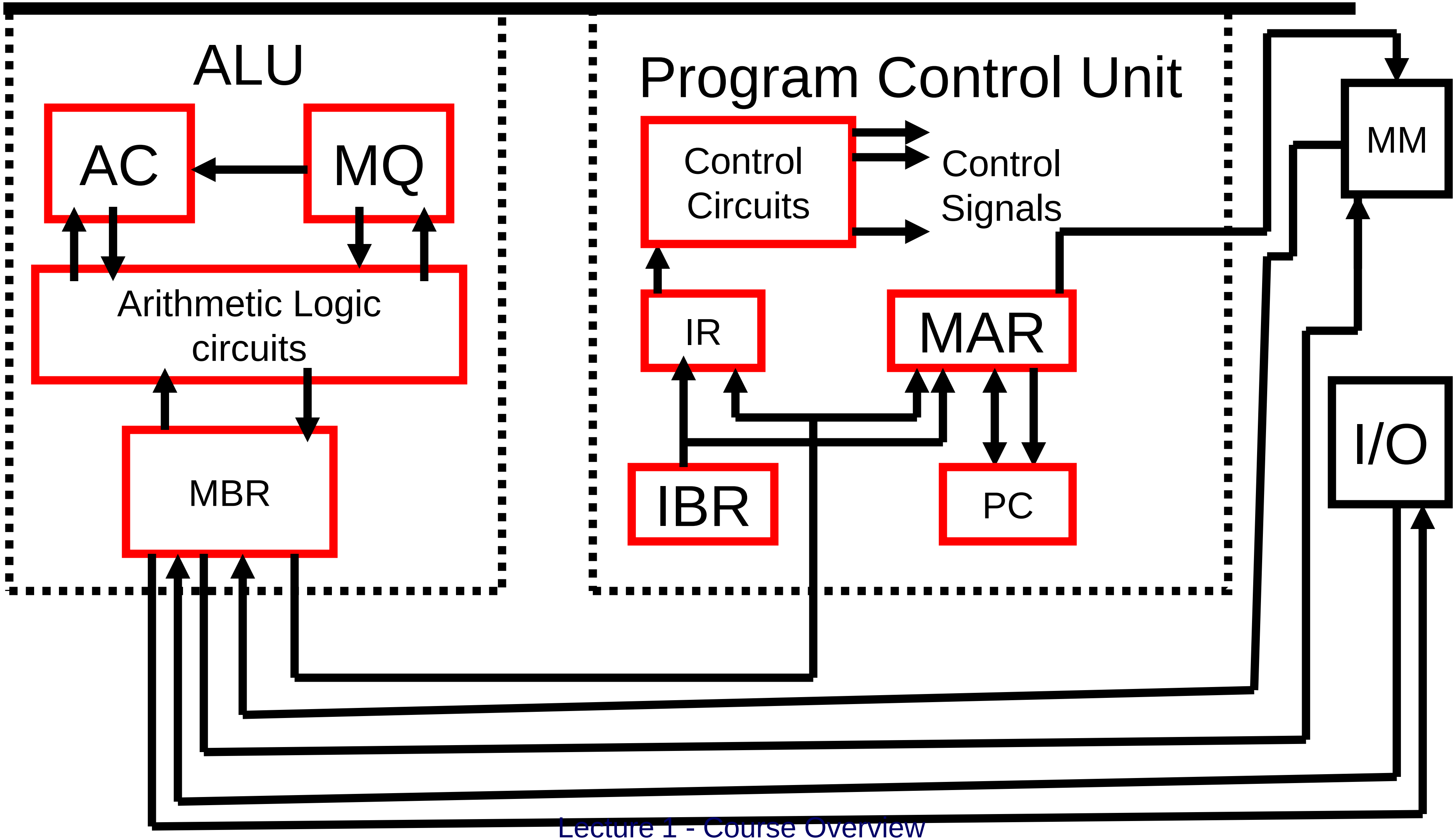


Structure of WORD (2)

- 1000 Storage locations, WORDS
  - Instruction word
  - Two 20-bit



Detailed Structure of IAS Computer



Components of the IAS Computer

- Memory Buffer Register (MBR)
- Memory Address Register (MAR)
- Instruction Register (IR)
- Instruction Buffer Register (IBR)
- Program Counter (PC)
- Accumulator (AC) & Multiplier Quotient (MQ)

IAS: 21 instructions grouped in 5 sets

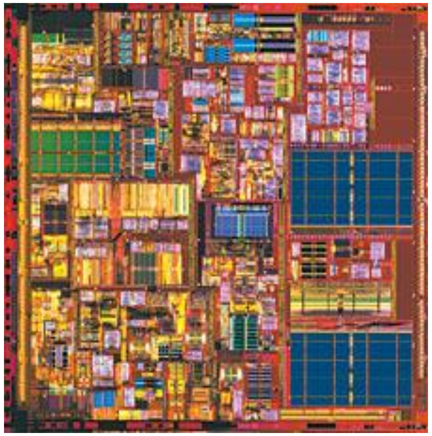
1. **Data Transfer**: move data between ALU & memory or between ALU and registers
2. **Unconditional Branch**: change the sequential execution of instructions
3. **Conditional Branch**
4. **Arithmetic**: operations performed by ALU
5. **Address Modify**: compute (in ALU) & insert (in memory) addresses => addressing flexibility

Roadmap for the Term: Major Topics

- **Computer Systems Overview**
- **Technology Trends**
- **Instruction Sets (and Software)**
- **Logic and Arithmetic**
- **Performance**
- **Processor Implementation**
- **Memory Systems**
- **Input/Output**

Computer Systems Overview

- Types of Computer Systems
- Abstractions used in Computer Systems
- Architecture vs. Organization
- Common Architectures
- “Under the Hood” - chips and systems



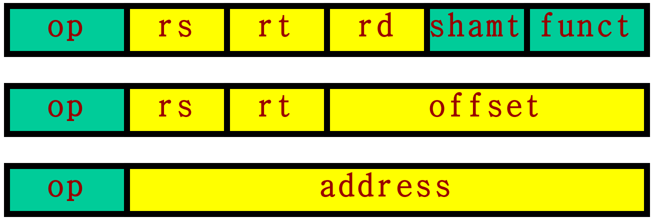
Technology Trends

- Historical Notes
- Current Technology (CMOS VLSI)
- Trends (Moore's Law)

Image Source:
Intel Corporation
`www.intel.com`

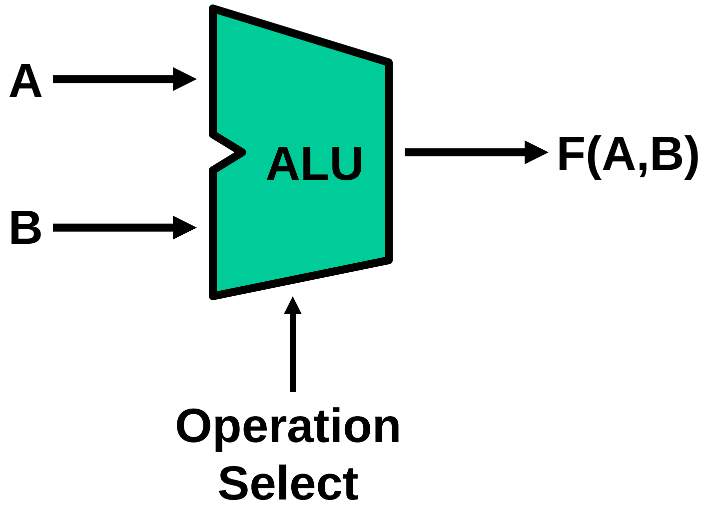
Instruction Sets (and Software)

- General principles of instruction set design
- The MIPS instruction set
- Software concerns: procedures, stacks, etc.



Logic & Arithmetic

- Quick review: binary numbers and arithmetic
- Adder & ALUs; multiplication & division
- Floating Point

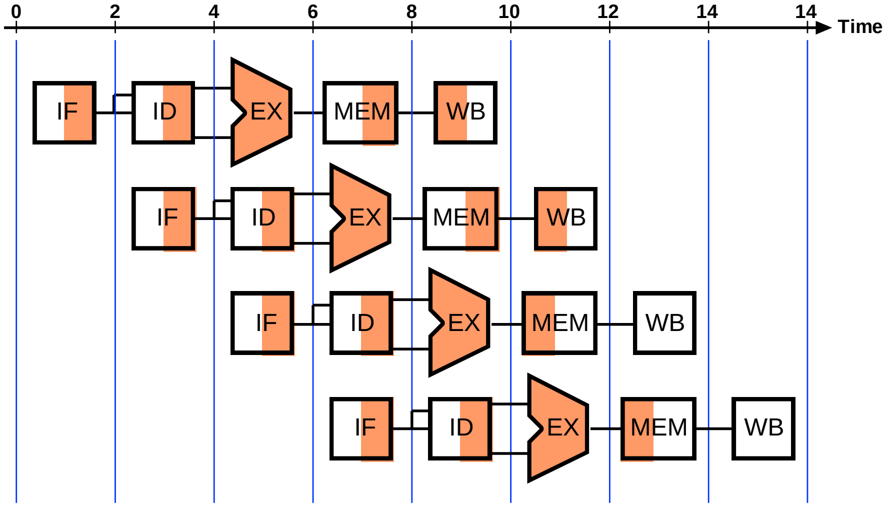


Performance

- Response Time vs. Throughput
- Measuring performance using individual programs
- Combining measurements
- Benchmarks

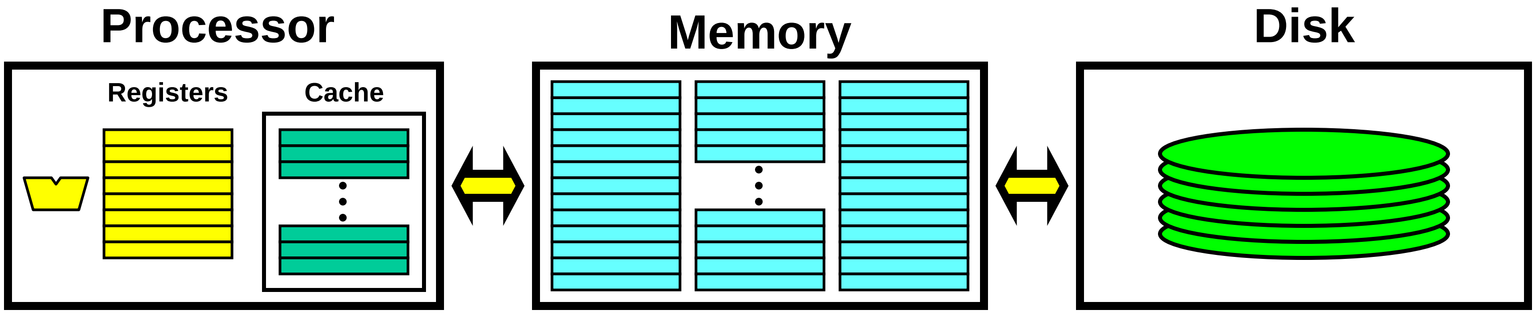
Processor Implementation

- Basic implementation
  - Single-Cycle
  - Multicycle
- Pipelined implementation
- Advanced techniques



Memory Systems

- Memory Technology Overview
- Memory Hierarchy
  - Cache Memories - making access faster
  - Virtual Memory - making memory larger using disk



Input/Output

- I/O Overview
- Impact of I/O on Performance
- Buses
- Interfacing

Image Source:
Seagate Technolgy LLC
www.seagate.com