Introduction

ICS 233

Computer Architecture and Assembly Language

Prof. Muhamed Mudawar

College of Computer Sciences and Engineering

King Fahd University of Petroleum and Minerals

Presentation Outline

- ❖ Welcome to ICS 233
- ❖ High-Level, Assembly-, and Machine-Languages
- ❖ Components of a Computer System
- ❖ Chip Manufacturing Process
- ❖ Technology Improvements
- ❖ Programmer's View of a Computer System

Welcome to ICS 233

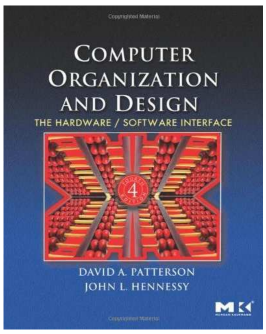
- Instructor: Dr. Muhamed F. Mudawar
- Office: Building 22, Room 328
- Office Phone: 4642
- Schedule and Office Hours:
  - http://faculty.kfupm.edu.sa/coe/mudawar/schedule/
- Course Web Page:
  - http://faculty.kfupm.edu.sa/coe/mudawar/ics233/
- Email:
  - mudawar@kfupm.edu.sa

Which Textbook will be Used?

- Computer Organization & Design:

  The Hardware/Software Interface

  - Fourth Edition
  - David Patterson and John Hennessy
  - Morgan Kaufmann Publishers, 2009



- Read the textbook in addition to slides

Grading Policy

❖ Laboratory	15%
❖ Quizzes	10%
❖ Programming Assignments	10%
❖ CPU Design Project	15%
❖ Midterm Exam I	15%
❖ Midterm Exam II	15%
❖ Final Exam	20%

Late Policy, Attendance, & Makeup

- ❖ Late assignment or project is accepted up to 2 days late
  - ✧ But will be penalized 5% for each late day
- ❖ Attendance will be taken at the beginning of each lecture
  - ✧ Official / medical excuses must be presented within one week
- ❖ Late attendance is counted as half presence
  - ✧ Two late attendances are counted as one absence
- ❖ No makeup exam will be given for missing exam or quiz

Software Tools

- MIPS Simulators
  - MARS: MIPS Assembly and Runtime Simulator
    - Runs MIPS-32 assembly language programs
    - Website: http://courses.missouristate.edu/KenVollmar/MARS/
  - SPIM
    - Also Runs MIPS-32 assembly language programs
    - Website: http://www.cs.wisc.edu/~larus/spim.html
- CPU Design and Simulation Tool
  - Logisim
    - Educational tool for designing and simulating CPUs
    - Website: http://ozark.hendrix.edu/~burch/logisim/

Course Learning Outcomes

- Towards the end of this course, you should be able to …
  - Describe the instruction set architecture of a MIPS processor
  - Analyze, write, and test MIPS assembly language programs
  - Describe organization/operation of integer & floating-point units
  - Design the datapath and control of a single-cycle CPU
  - Design the datapath/control of a pipelined CPU & handle hazards
  - Describe the organization/operation of memory and caches
  - Analyze the performance of processors and caches
- Required Background
  - Ability to program confidently in Java or C
  - Ability to design a combinational and sequential circuit

What is "Computer Architecture" ?

- ❖ Computer Architecture =

  Instruction Set Architecture +

  Computer Organization
- ❖ Instruction Set Architecture (ISA)
  - ✧ WHAT the computer does (logical view)
- ❖ Computer Organization
  - ✧ HOW the ISA is implemented (physical view)
- ❖ We will study both in this course

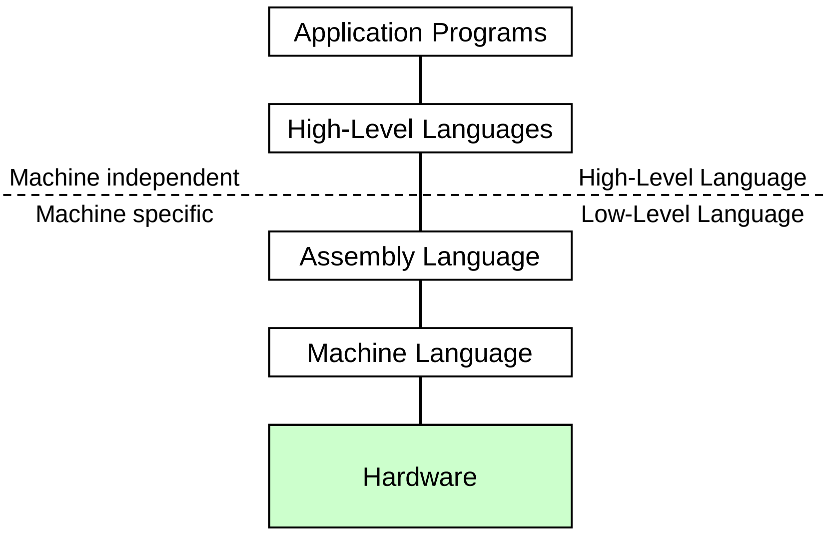
Next . . .

- ❖ Welcome to ICS 233
- ❖ High-Level, Assembly-, and Machine-Languages
- ❖ Components of a Computer System
- ❖ Chip Manufacturing Process
- ❖ Technology Improvements
- ❖ Programmer's View of a Computer System

Some Important Questions to Ask

- ❖ What is Assembly Language?
- ❖ What is Machine Language?
- ❖ How is Assembly related to a high-level language?
- ❖ Why Learn Assembly Language?
- ❖ What is an Assembler, Linker, and Debugger?

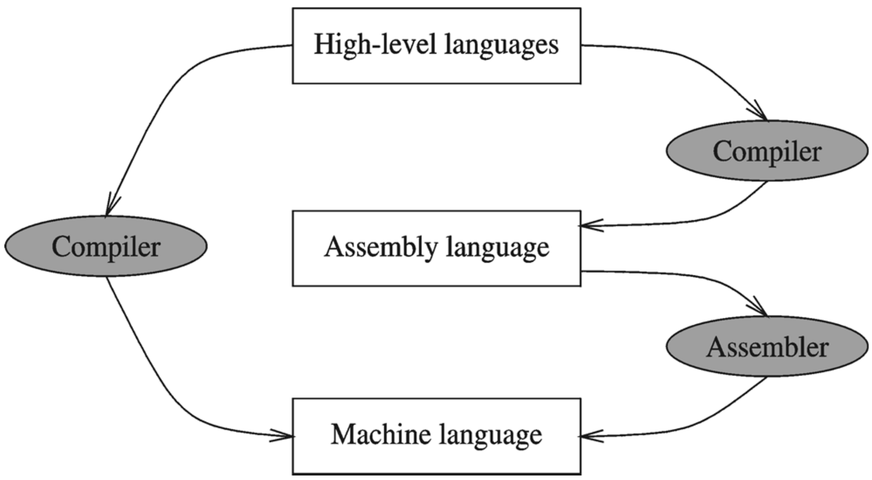
A Hierarchy of Languages



Assembly and Machine Language

- Machine language
  - Native to a processor: executed directly by hardware
  - Instructions consist of binary code: 1s and 0s
- Assembly language
  - Slightly higher-level language
  - Readability of instructions is better than machine language
  - One-to-one correspondence with machine language instructions
- Assemblers translate assembly to machine code
- Compilers translate high-level programs to machine code
  - Either directly, or
  - Indirectly via an assembler

Compiler and Assembler



Translating Languages

Program (C Language):

```
swap(int v[], int k) {
   int temp;
   temp = v[k];
   v[k] = v[k+1];
   v[k+1] = temp;
}
```

Compiler

MIPS Assembly Language:

```
sll $2,$5, 2
add $2,$4,$2
lw  $15,0($2)
lw  $16,4($2)
sw  $16,0($2)
sw  $15,4($2)
jr  $31
```

Assembler

MIPS Machine Language:

```
00051080
00821020
8C620000
8CF20004
ACF20000
AC620004
03E00008
```

A statement in a high-level language is translated typically into several machine-level instructions

Advantages of High-Level Languages

- Program development is faster
  - High-level statements: fewer instructions to code
- Program maintenance is easier
  - For the same above reasons
- Programs are portable
  - Contain few machine-dependent details
    - Can be used with little or no modifications on different machines
  - Compiler translates to the target machine language
  - However, Assembly language programs are not portable

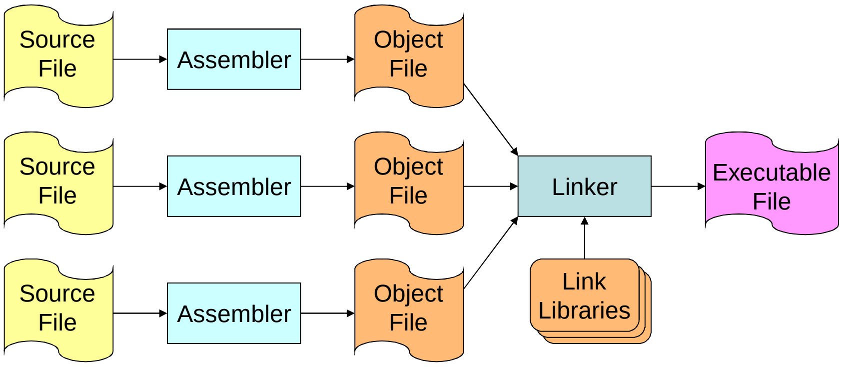
Why Learn Assembly Language?

- Many reasons:
  - Accessibility to system hardware
  - Space and time efficiency
  - Writing a compiler for a high-level language
- Accessibility to system hardware
  - Assembly Language is useful for implementing system software
  - Also useful for small embedded system applications
- Space and Time efficiency
  - Understanding sources of program inefficiency
  - Tuning program performance
  - Writing compact code

Assembly Language Programming Tools

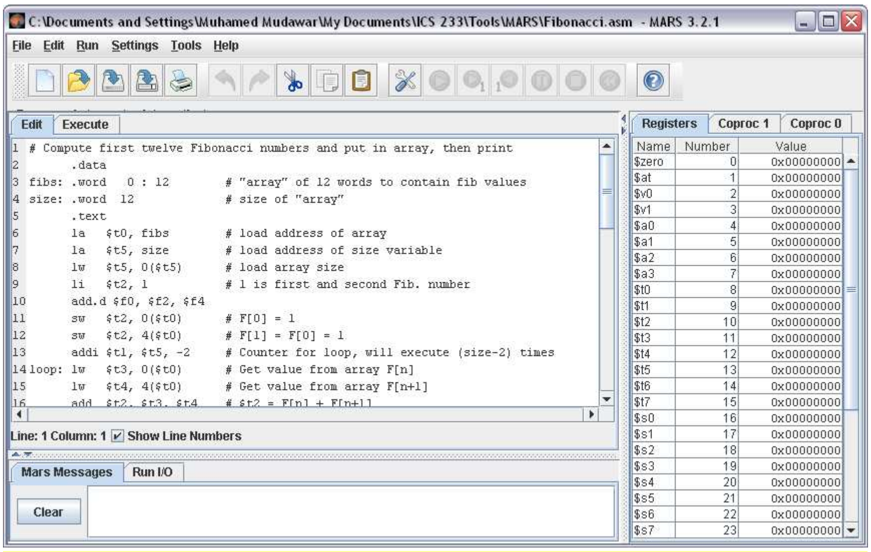
- Editor
  - Allows you to create and edit assembly language source files
- Assembler
  - Converts assembly language programs into object files
  - Object files contain the machine instructions
- Linker
  - Combines object files created by the assembler with link libraries
  - Produces a single executable program
- Debugger
  - Allows you to trace the execution of a program
  - Allows you to view machine instructions, memory, and registers

Assemble and Link Process



- A program may consist of multiple source files
- Assembler translates each source file separately into an object file
- Linker links all object files together with link libraries

MARS Assembler and Simulator Tool

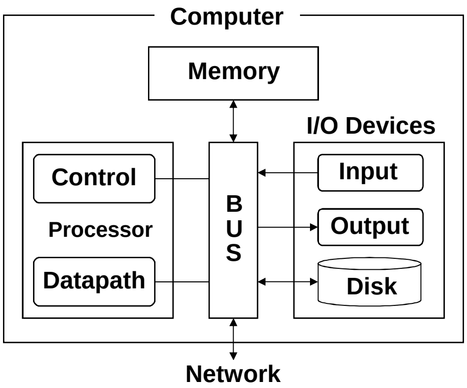


Next . . .

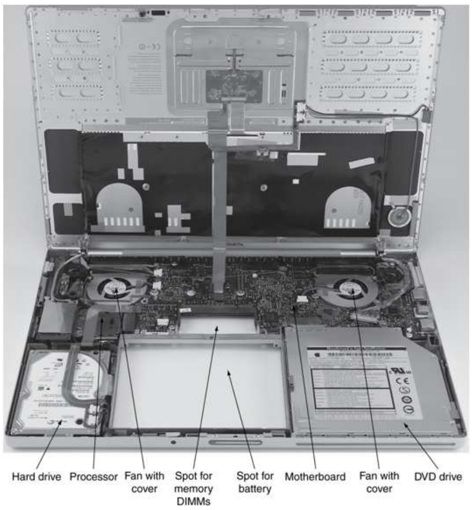
- Welcome to ICS 233
- High-Level, Assembly-, and Machine-Languages
- Components of a Computer System
- Chip Manufacturing Process
- Technology Improvements
- Programmer's View of a Computer System

Components of a Computer System

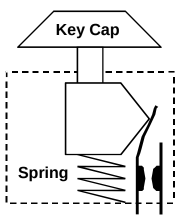
- Processor
  - Datapath
  - Control
- Memory & Storage
  - Main Memory
  - Disk Storage
- Input devices
- Output devices
- Bus: Interconnects processor to memory and I/O
- Network: newly added component for communication



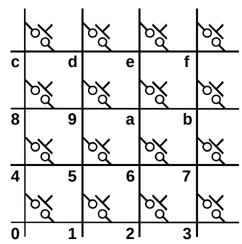
Opening the Box



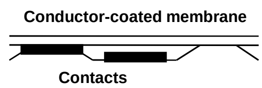
Input Devices



Mechanical switch

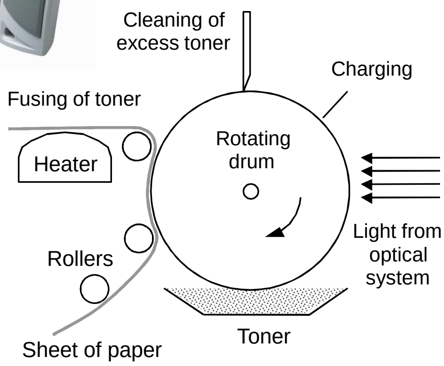


Logical arrangement of keys



Membrane switch

Output Devices



Laser printing

Memory Devices

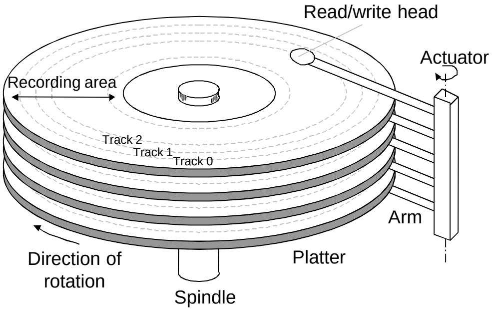
- Volatile Memory Devices
  - RAM = Random Access Memory
  - DRAM = Dynamic RAM
    - 1-Transistor cell + capacitor
    - Dense but slow, must be refreshed
    - Typical choice for main memory
  - SRAM: Static RAM
    - 6-Transistor cell, faster but less dense than DRAM
    - Typical choice for cache memory

- Non-Volatile Memory Devices
  - ROM = Read Only Memory
  - Flash Memory

Magnetic Disk Storage

A Magnetic disk consists of a collection of platters

Provides a number of recording surfaces



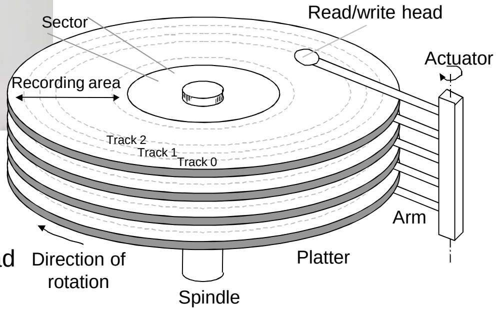
Arm provides read/write heads for all surfaces

The disk heads are connected together and move in conjunction

Magnetic Disk Storage

Disk Access Time =

- Seek Time +
- Rotation Latency +
- Transfer Time



Seek Time: head movement to the desired track (milliseconds)

Rotation Latency: disk rotation until desired sector arrives under the head

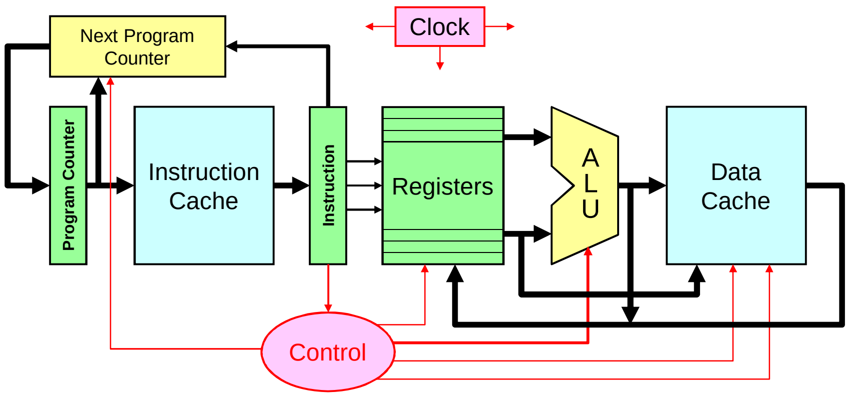
Transfer Time: to transfer data

Example on Disk Access Time

- Given a magnetic disk with the following properties
  - Rotation speed = 5400 RPM (rotations per minute)
  - Average seek = 9 ms, Sector = 512 bytes, Track = 200 sectors
- Calculate
  - Time of one rotation (in milliseconds)
  - Average time to access a block of 80 consecutive sectors
- Answer
  - Rotations per second = 5400/60 = 90 RPS
  - Rotation time in milliseconds = 1000/90 = 11.1 ms
  - Average rotational latency = time of half rotation = 5.56 ms
  - Time to transfer 80 sectors = (80/200) * 11.1 = 4.44 ms
  - Average access time = 9 + 5.56 + 4.44 = 19 ms

Inside the Processor (CPU)

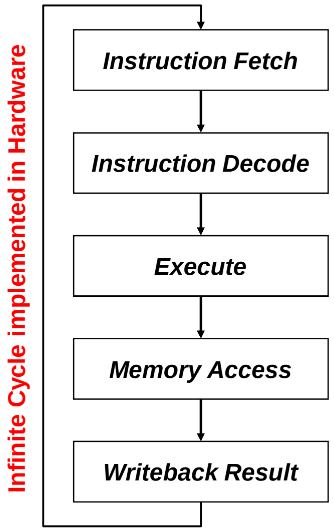
- Datapath: part of a processor that executes instructions
- Control: generates control signals for each instruction



Datapath Components

- Program Counter (PC)
  - Contains address of instruction to be fetched
  - Next Program Counter: computes address of next instruction
- Instruction and Data Caches
  - Small and fast memory containing most recent instructions/data
- Register File
  - General-purpose registers used for intermediate computations
- ALU = Arithmetic and Logic Unit
  - Executes arithmetic and logic instructions
- Buses
  - Used to wire and interconnect the various components

Fetch - Execute Cycle



Fetch instruction
Compute address of next instruction

Generate control signals for instruction
Read operands from registers

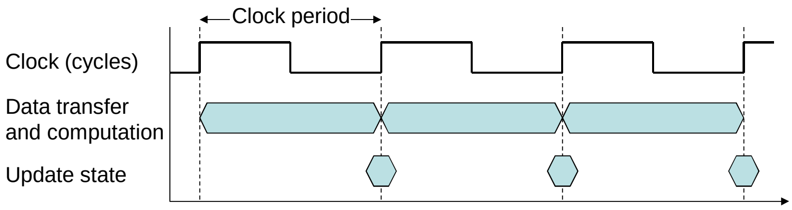
Compute result value

Read or write memory (load/store)

Writeback result in a register

Clocking

Operation of digital hardware is governed by a clock

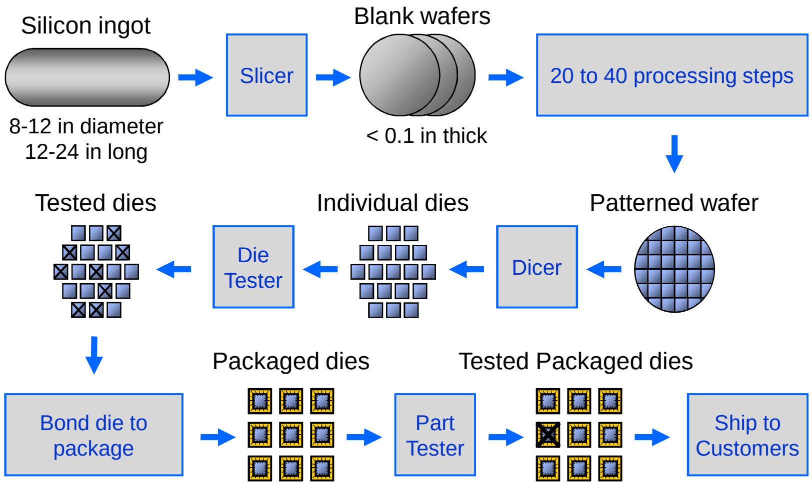


- Clock period: duration of a clock cycle
  - e.g., 250 ps = 0.25 ns = 0.25 $\times 10^{-9}$ sec
- Clock frequency (rate) = 1 / clock period
  - e.g., 1/ 0.25 $\times 10^{-9}$ sec = 4.0×10^{9} Hz = 4.0 GHz

Next . . .

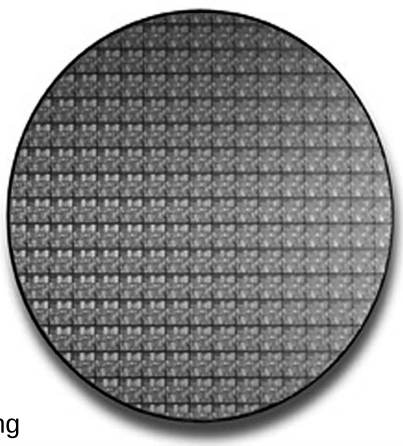
- ❖ Welcome to ICS 233
- ❖ Assembly-, Machine-, and High-Level Languages
- ❖ Components of a Computer System
- ❖ Chip Manufacturing Process
- ❖ Technology Improvements
- ❖ Programmer's View of a Computer System

Chip Manufacturing Process

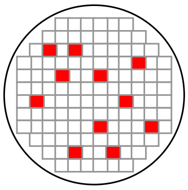


Wafer of Pentium 4 Processors

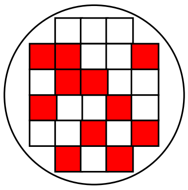
- 8 inches (20 cm) in diameter
- Die area is 250 mm^2
  - About 16 mm per side
- 55 million transistors per die
  - 0.18 μm technology
  - Size of smallest transistor
  - Improved technology uses
    - 0.13 μm and 0.09 μm
- Dies per wafer = 169
  - When yield = 100%
  - Number is reduced after testing
  - Rounded dies at boundary are useless



Effect of Die Size on Yield



120 dies, 109 good



26 dies, 15 good

☐ Good Die

■ Defective Die

Dramatic decrease in yield with larger dies

Yield = (Number of Good Dies) / (Total Number of Dies)

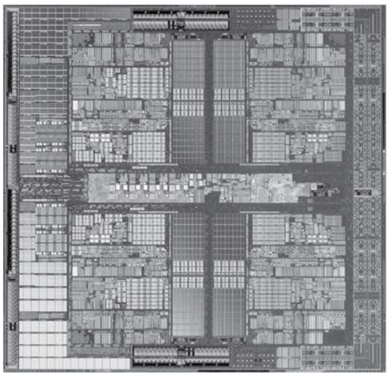
$$\text{Yield} = \frac{1}{(1 + (\text{Defect per area} \times \text{Die area} / 2))^2}$$

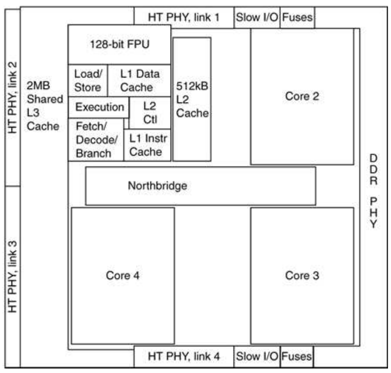
Die Cost = (Wafer Cost) / (Dies per Wafer × Yield)

Inside a Multicore Processor Chip

AMD Barcelona: 4 Processor Cores

3 Levels of Caches





Next . . .

- ❖ Welcome to ICS 233
- ❖ Assembly-, Machine-, and High-Level Languages
- ❖ Components of a Computer System
- ❖ Chip Manufacturing Process
- ❖ Technology Improvements
- ❖ Programmer's View of a Computer System

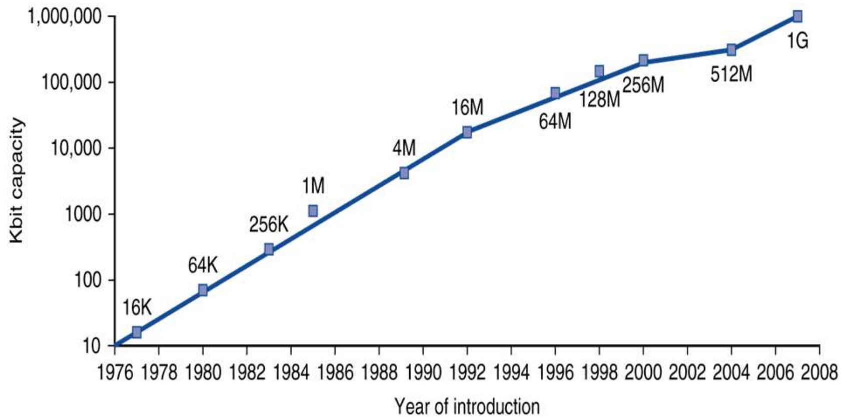
Technology Improvements

Year	Technology	Relative performance/cost
1951	Vacuum tube	1
1965	Transistor	35
1975	Integrated circuit (IC)	900
1995	Very large scale IC (VLSI)	2,400,000
2005	Ultra large scale IC	6,200,000,000

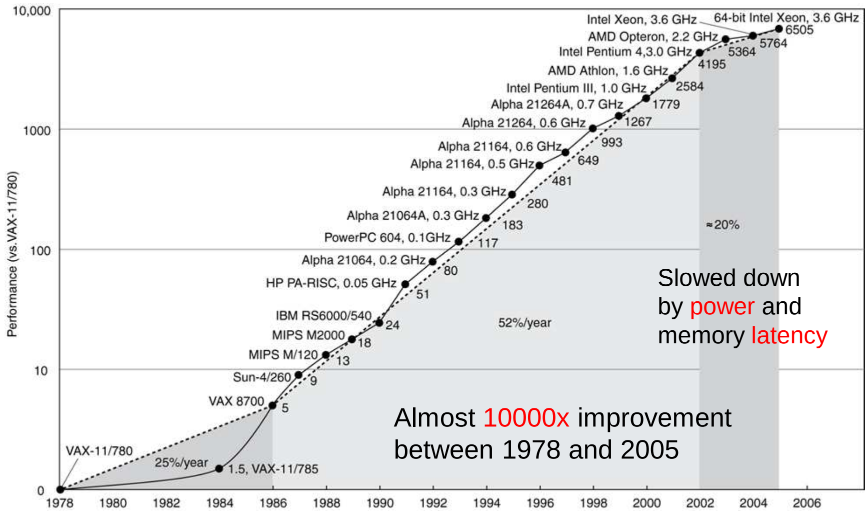
- ❖ Processor transistor count: about 30% to 40% per year
- ❖ Memory capacity: about 60% per year (4x every 3 years)
- ❖ Disk capacity: about 60% per year
- ❖ Opportunities for new applications
- ❖ Better organizations and designs

Growth of Capacity per DRAM Chip

- DRAM capacity quadrupled almost every 3 years
  - 60% increase per year, for 20 years



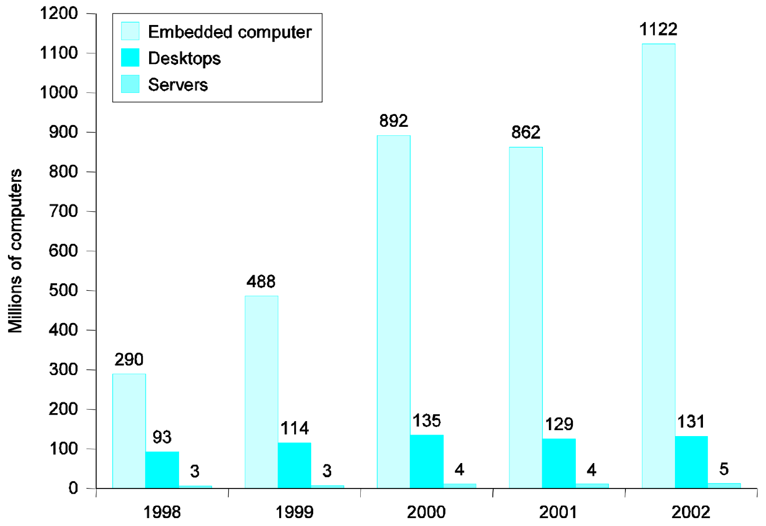
Processor Performance



Classes of Computers

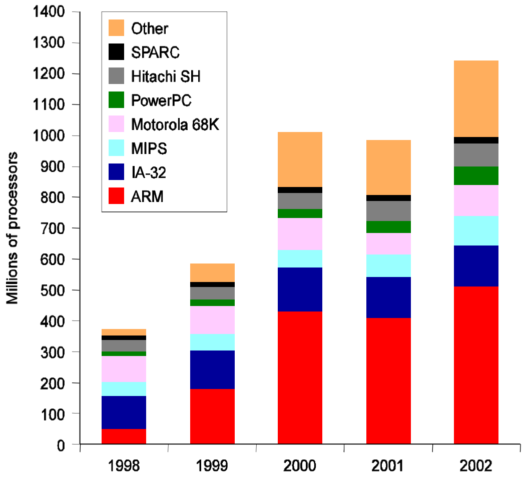
- Desktop / Notebook Computers
  - General purpose, variety of software
  - Subject to cost/performance tradeoff
- Server Computers
  - Network based
  - High capacity, performance, reliability
  - Range from small servers to building sized
- Embedded Computers
  - Hidden as components of systems
  - Stringent power/performance/cost constraints

Computer Sales

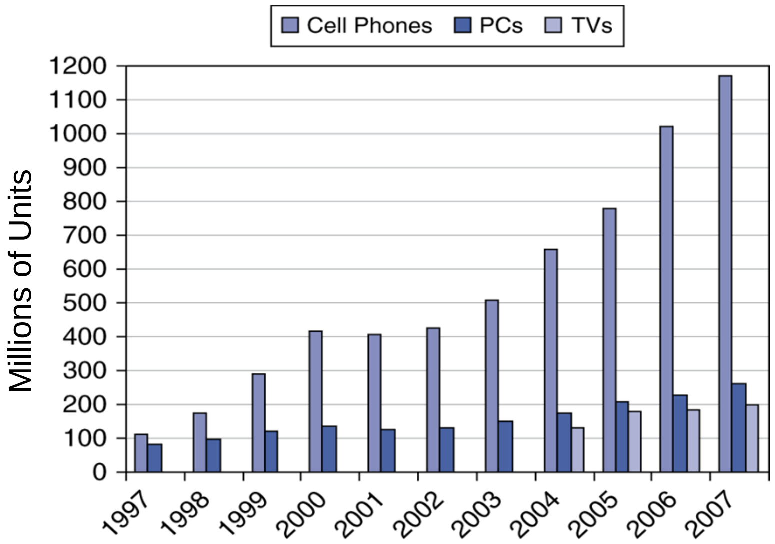


Microprocessor Sales

- ARM processor sales exceeded Intel IA-32 processors, which came second
- ARM processors are used mostly in cellular phones
- Most processors today are embedded in cell phones, digital TVs, video games, and a variety of consumer devices



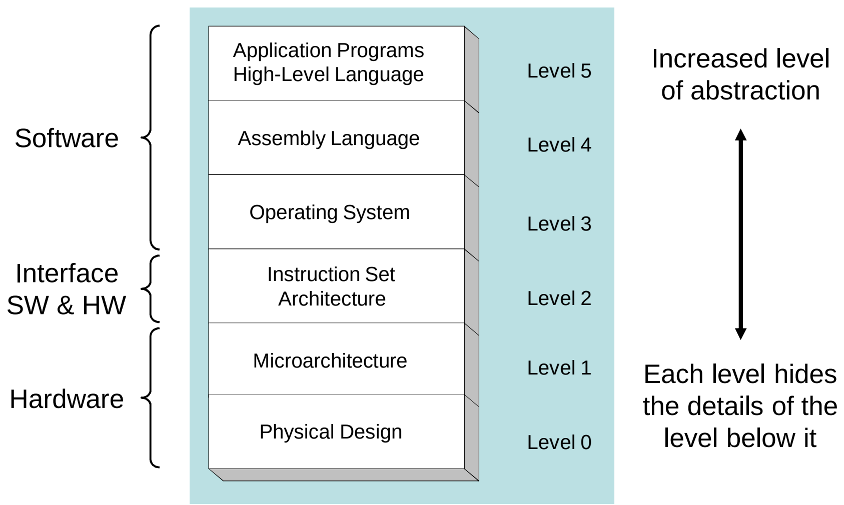
The Processor Market



Next . . .

- ❖ Welcome to ICS 233
- ❖ Assembly-, Machine-, and High-Level Languages
- ❖ Components of a Computer System
- ❖ Chip Manufacturing Process
- ❖ Technology Improvements
- ❖ Programmer's View of a Computer System

Programmer's View of a Computer System



Programmer's View - 2

- Application Programs (Level 5)
  - Written in high-level programming languages
  - Such as Java, C++, Pascal, Visual Basic . . .
  - Programs compile into assembly language level (Level 4)
- Assembly Language (Level 4)
  - Instruction mnemonics are used
  - Have one-to-one correspondence to machine language
  - Calls functions written at the operating system level (Level 3)
  - Programs are translated into machine language (Level 2)
- Operating System (Level 3)
  - Provides services to level 4 and 5 programs
  - Translated to run at the machine instruction level (Level 2)

Programmer's View - 3

- Instruction Set Architecture (Level 2)
  - Interface between software and hardware
  - Specifies how a processor functions
  - Machine instructions, registers, and memory are exposed
  - Machine language is executed by Level 1 (microarchitecture)
- Microarchitecture (Level 1)
  - Controls the execution of machine instructions (Level 2)
  - Implemented by digital logic
- Physical Design (Level 0)
  - Implements the microarchitecture
  - Physical layout of circuits on a chip