ECE232: Hardware Organization and Design

Part 1: Introduction

http://www.ecs.umass.edu/ece/ece232/

Adapted from *Computer Organization and Design*, Patterson & Hennessy, UCB

Course Administration - 1

- Instructors: Israel Koren
  koren@ecs.umass.edu
- Mani Krishna
  krishna@ecs.umass.edu
- TAs: Daniel Gomez-Prado
  Kunal Ganeshpure
  Nitin Prakash

  Office Hours & e-mail addresses posted on the course web page
- URL: http://www.ecs.umass.edu/ece/ece232/
- SPARK: http://spark.oit.umass.edu/
- Text: Required: *Computer Organization and Design, The Hardware/Software Interface* by D. A. Patterson and J. L. Hennessy, Morgan Kaufmann; 4th Edition, ISBN 978-0123744937

Course Administration - 2

- Slides will be posted in spark (https://spark.oit.umass.edu)
  - Grading Policy
    - Midterm 1 25% March 7, 2011, 4-6 pm
    - Midterm 2 25% April 14, 2011, 4-6 pm
    - Final 40% TBA
    - Homework 10%
- No Midterm make-up exams
  - Percentages adjusted for justified absence
- Homework policy
  - Students are encouraged to work in groups. Maximum group size is 4. All names must be clearly noted. Solutions are returned during discussion
  - Homework must be submitted through spark.oit.umass.edu
  - If you work in a group indicate all names at the top of the page; every student must submit homework through spark
  - Late policy: 20% deducted for homework turned in late
  - Homework must be picked up within 2 weeks

Course Content & Goals

- Content
  - Principles of computer architecture: CPU datapath and control unit design
  - Assembly language programming in MIPS
  - Memory hierarchies and design
  - I/O organization and design
  - Possible advanced topics
- Course goals
  - To learn the organizational structures that determine the capabilities and performance of computer systems
  - To understand the interactions between the computer's architecture and its software
  - To understand cost performance trade-offs

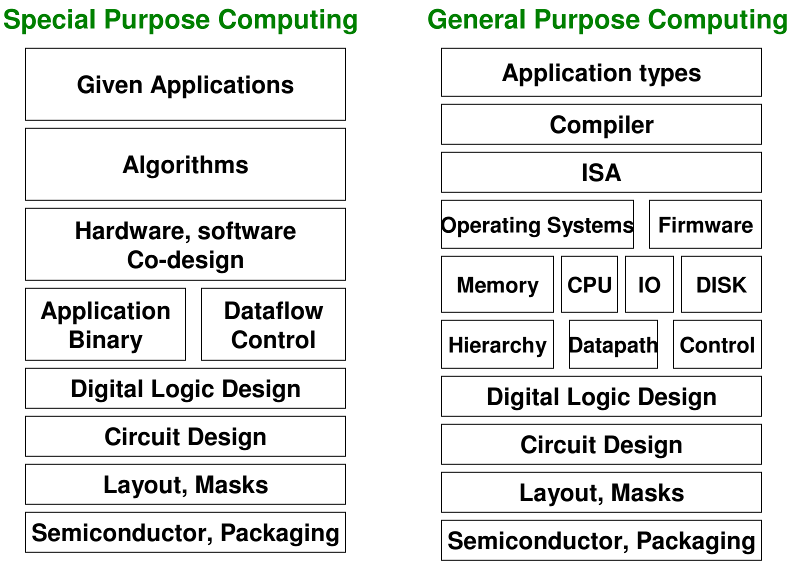
What You Should Know

- Binary numbers
- Read and write basic C/java programs
- Understand the steps in compiling and executing a program
- Basic Verliog constructs
  - To deal with HW assignments
- Logic design
  - logical equations, schematic diagrams
  - Combinational vs. sequential logic
  - Finite state machines (FSMs)

Why you should know hardware organization?

- Computer organization principles are everywhere
  - **Embedded computer** vs. **general-purpose computers**:
    - Cellphone
    - Digital Camera
    - MP3 music player
    - Industrial process control
- Complex system design
  - How to partition a problem
  - Functional Spec → Control & Datapath → Physical implementation
  - Modern CAD tools

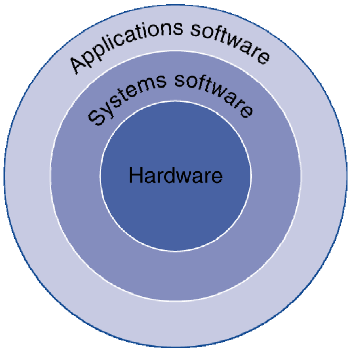
Computing Systems



Abstractions: ISA and ABI

- Abstraction helps us deal with complexity
  - Hide lower-level detail
- **Instruction set architecture** - ISA: An abstract interface between the hardware and the lowest level software of a machine
  - Encompasses all the information necessary to write a machine language program that will run correctly, including
    - instructions, registers, memory access, I/O
- ABI (**application binary interface**): The user portion of the instruction set plus the operating system interfaces used by application programmers
  - Defines a standard for binary portability across computers

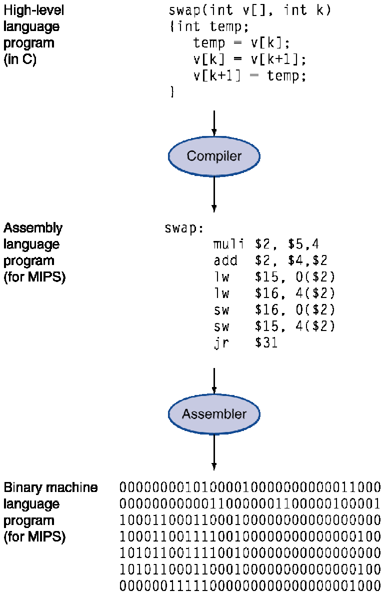
System Layers



- Application software
  - Written in high-level language
- System software
  - Compiler: translates HLL code to machine code
  - Operating System: service code
    - Handling input/output
    - Managing memory and storage
    - Scheduling tasks & sharing resources
- Hardware
  - Processor, memory, I/O controllers

Levels of Program Code

- High-level language
  - Level of abstraction closer to problem domain
  - Provides for productivity and portability
- Assembly language
  - Textual representation of instructions
- Hardware representation
  - Binary digits (bits)
  - Encoded instructions and data



High-level language program (in C)

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

Assembly language program (for MIPS)

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

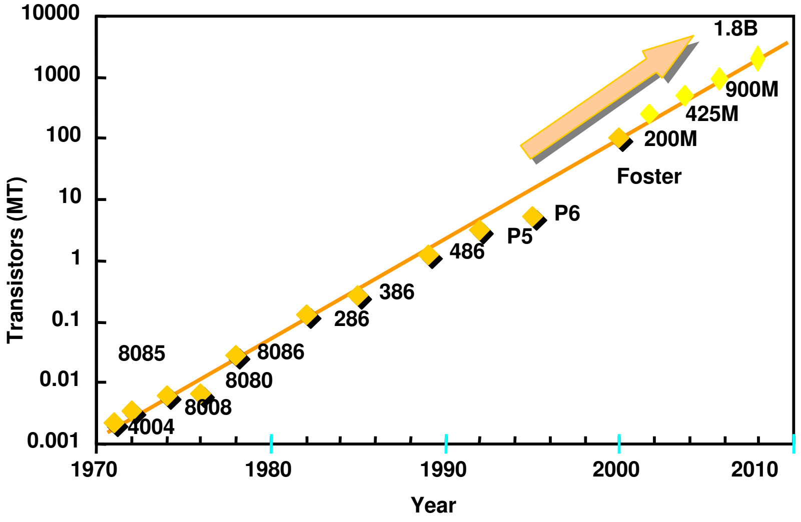
Binary machine language program (for MIPS)

```
00000000101000010000000000011000
00000000000110000001100000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
00000011111000000000000000001000
```

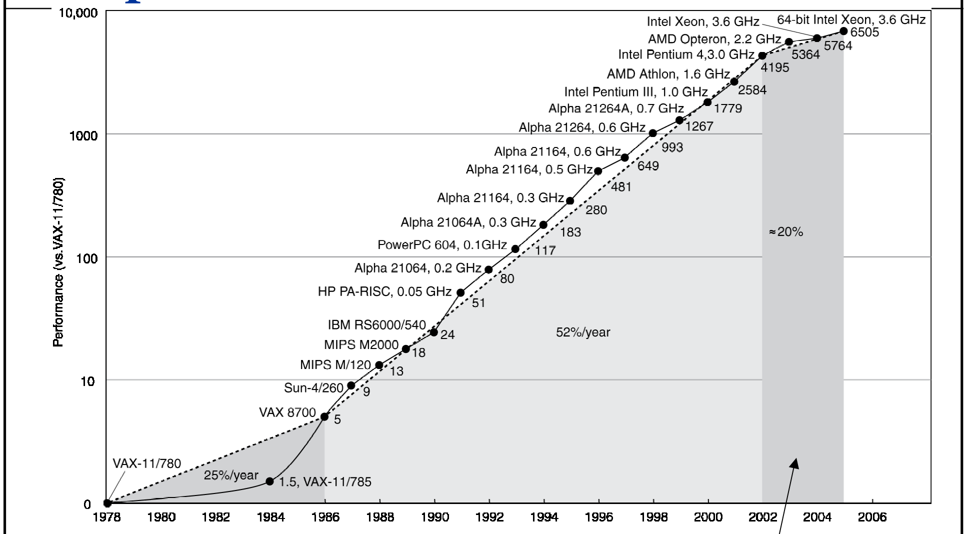
µProcessor Advances - Moore's Law

- In 1965, Gordon Moore predicted that the number of transistors that can be integrated on a die would double every 18 to 24 months (i.e., grow exponentially with time).
- Amazingly visionary – million transistor/chip barrier was crossed in the 1980's.
  - 2300 transistors, 1 MHz clock (Intel 4004) - 1971
  - 16 Million transistors (Ultra Sparc III)
  - 42 Million transistors, 2 GHz clock (Intel Xeon) – 2001
  - 55 Million transistors, 3 GHz, 130nm technology, 250mm^2 die (Intel Pentium 4) - 2004
  - 140 Million transistor (HP PA-8500)
  - 1.8 Billion transistors (Itanium II)

Moore's Law & Intel Processors

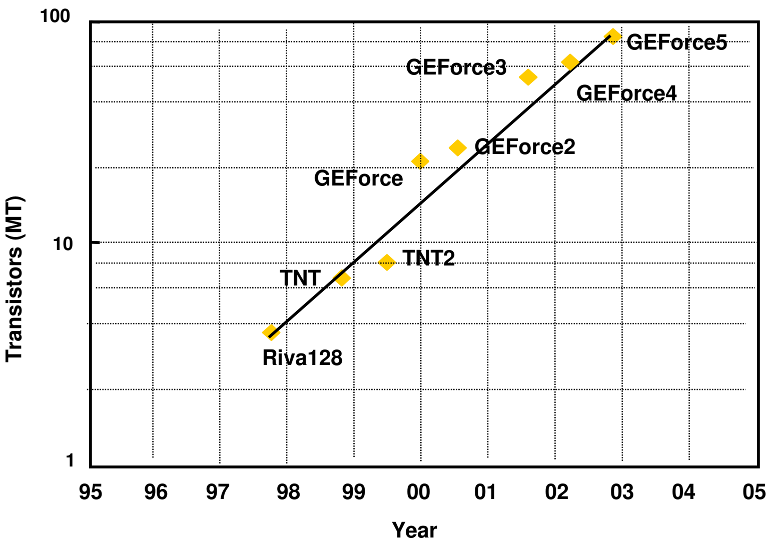


Uniprocessor Performance



Constrained by power, instruction-level parallelism, memory latency

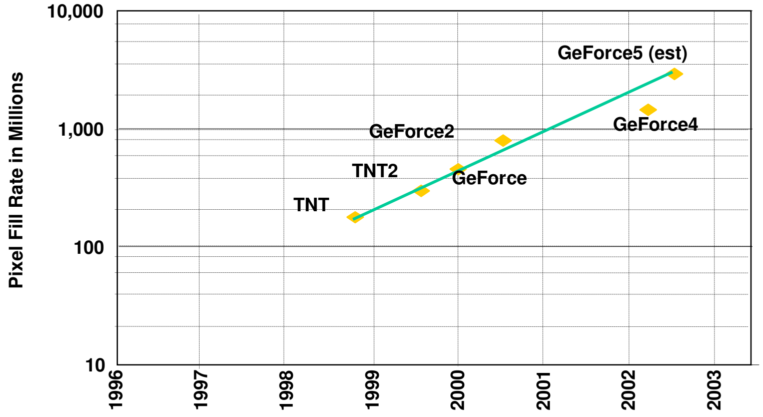
Moore’s law in GPU world



Observation

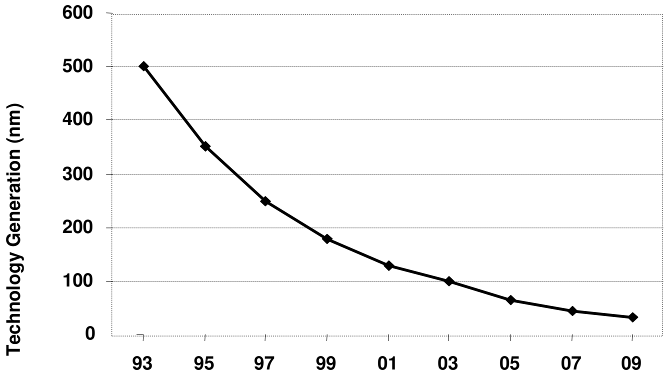
- Transistor count increases to meet demand for performance and functionality
- New applications create demand for increase in performance

GPU Pixel Fill-rates Doubling every 1 year !



How is that possible?

- Scale the transistor channel length

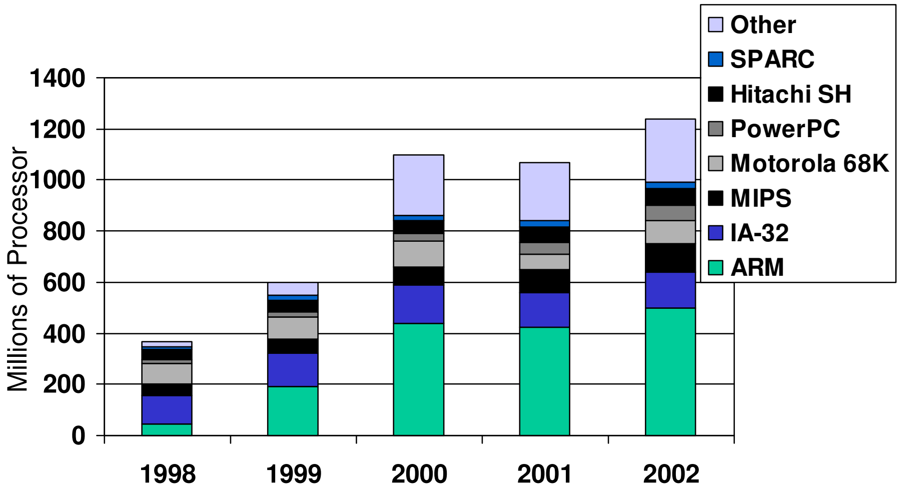


Feature size scaling to reduce die size

Impacts of Advancing Technology

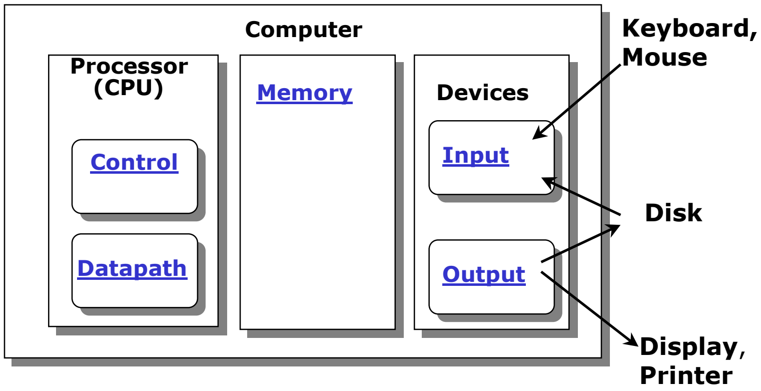
- **Processor**
  - logic capacity: increases about 30% per year
  - performance: 2x every 1.5 years
- **Memory**
  - DRAM capacity: 4x every 3 years, now 2x every 2 years
  - speed: 1.5x every 10 years
  - cost per bit: decreases about 25% per year
- **Disk**
  - capacity: increases about 60% per year

ISA Type Sales

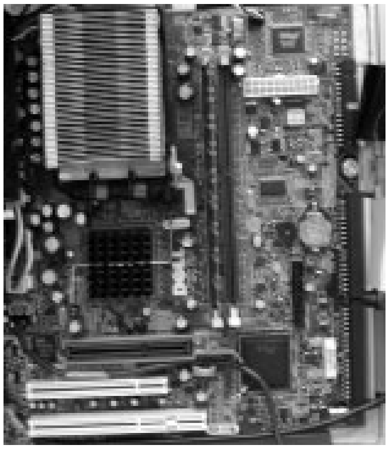


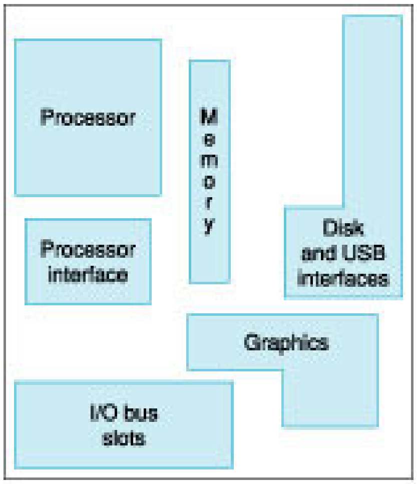
Example Machine Organization

- Workstation design target
  - 25% of cost - processor
  - 25% of cost - memory (minimum memory size)
  - Rest - I/O devices, power supplies, box

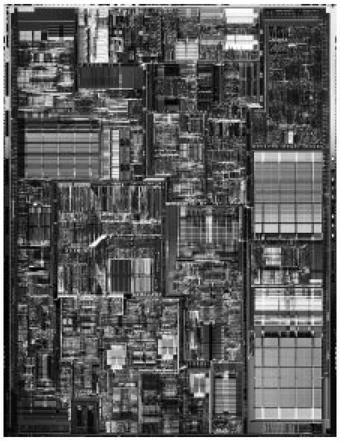


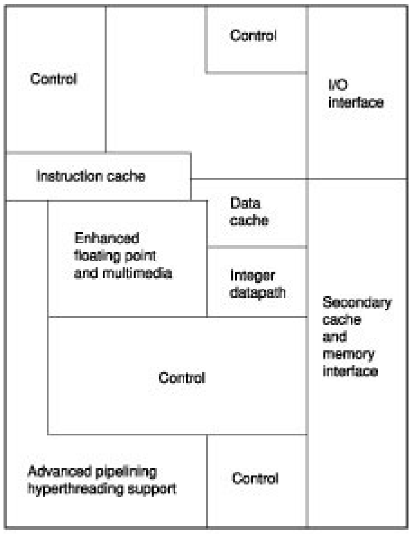
PC Motherboard Closeup





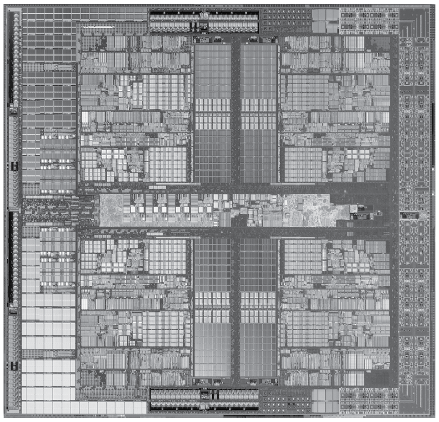
Inside the Pentium 4 Processor Chip

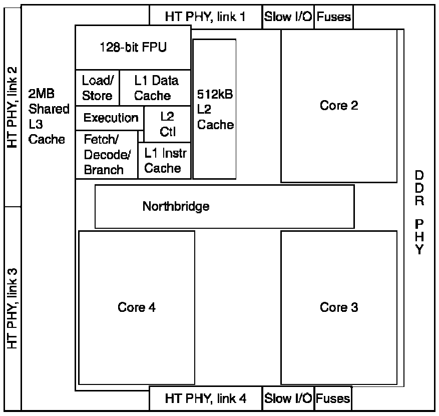



Control
Control
I/O interface
Instruction cache
Data cache
Enhanced floating point and multimedia
Integer datapath
Secondary cache and memory interface
Control
Advanced pipelining hyperthreading support
Control

Inside the Processor

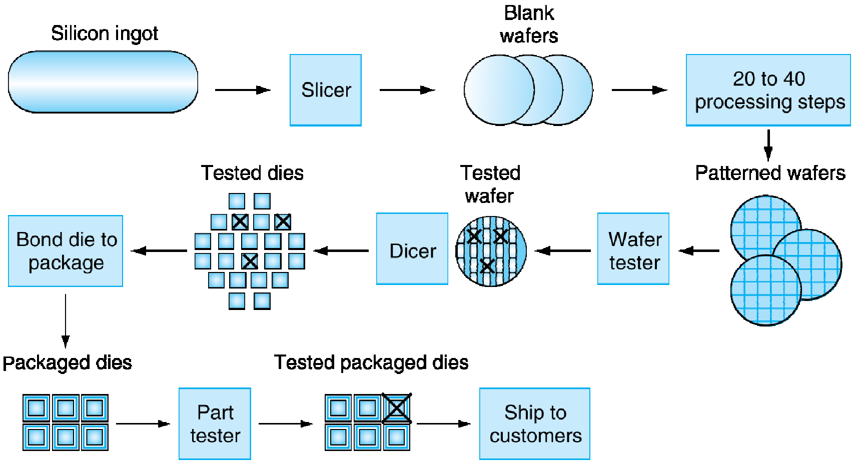
- AMD Barcelona: 4 processor cores





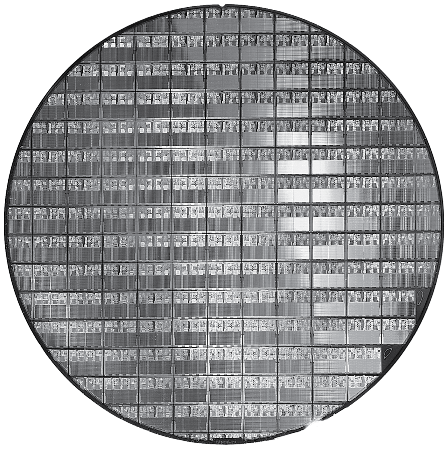
Reading assignment: Chapter 1

Manufacturing ICs



- Yield: proportion of working dies per wafer

AMD Opteron X2 Wafer



- X2: 300mm wafer, 117 chips, 90nm technology