CS/ECE 552: Introduction to Computer Architecture

Instructor: Mark D. Hill
T.A.: Brandon Schwartz

Section 2 Fall 2000
University of Wisconsin-Madison

Lecture notes originally created by Mark D. Hill
Updated by T.N. Vijaykumar (Purdue) & Mark D. Hill

Computer Architecture =

Instruction Set Architecture

- ... the attributes of a [computing] system as seen by the programmer. i.e., the conceptual structure and functional behavior, as distinct from the organization of the data flows and controls, the logic design, and the physical implementation. -- Amdahl, Blaaw, & Brooks, 1964
- E.g., Intel x86, Sun SPARC, IBM PowerPC, SGI MIPS

+ Machine Organization

- ALUs, Buses, Caches, Memories, etc.

552 In Context

Prerequisites

- 352 - gates up to multiplexors and adders
- 354 - high-level language down to machine language interface or *instruction set architecture* (ISA)

This course -- 552 -- *puts it all together*

- Implement the logic that provides ISA interface
- Must do datapath and control, but *no magic*
- Manage tremendous complexity with abstraction

Follow-on courses explore trade-offs: e.g., 752, 755, 757

Why Study Computer Architecture?

(1) To become a *computer designer*

- Many who have many this class design the computers you use

(2) To learn what is *under the hood* of your computer

- To better understand when things break
- To see better how to design high-performance applications
- To aid design of system software (OS, compilers, libraries)

(3) Because it is *intellectually fascinating*!

Why is Computer Architecture so Dynamic?

Everything is changing

- At non-uniform rates
- With inflection points

Technology Push

Application Pull

Technology Push

What do these two intervals have in common?

- 1776-1999 (224 years)
- 2000-2001 (2 years)

Answer: Equal progress in absolute processor speed
(and more doublings in 2002-3, and 2004-5)

Consider salary doubling

Driven by *Moore's Law*

- Device per chip doubles every 18 months to 2 years

Computer architects work to turn the additional resources into speed!

Some Intel Numbers

Date	What	Comments
1947	1st transistor	Bell Labs
1958	1st Integrated Circuit	Texas Instruments
1971	1st microprocessor	Intel
1974	Intel 4004	2300 transistors
1978	Intel 8086	29K transistors
1989	Intel 80486	1.2M transistors
1995	Intel Pentium Pro	5.5M transistors
2006	*Intel Estimate*	*350M transistors*
2011	*Intel Estimate*	*1G transistors*

Technology Push, cont.

Technology advances at varying rates

- E.g., DRAM capacity: about 60% per year
- But DRAM speed: about 10% per year
- Need more caches!

Inflection Points

- Crossover causes rapid change
- E.g. Enough devices for processor on chip: microprocessor
- Soon: system on a chip & chip multiprocessors

Application Pull

Corollary to Moore's Law: *Cost halves every two years*

- In a decade you can buy a computer for less than its sales tax today. --Jim Gray

Computers cost-effective for

- National security -- weapons design
- Enterprise computing -- banking
- Departmental computing -- computer aided design
- Personal computing -- speadsheets, email, web
- Embedded computing -- microcode in electric shaver

Application Pull, cont.

What about the future?

Must dream up applications that are not cost-effective today

- Virtual reality
- Mass customization
- Web agents
- Wireless
- Proactive (beyond interactive) w/ sensors
    - e.g., mattress adjust to save your back

This is your job!

Abstraction

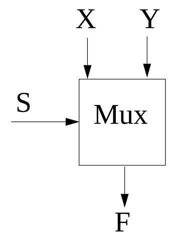
Difference between interface and implementation

- Interface - *WHAT* something does
- Implementation - *HOW* it does so

Abstraction - E.g.,

2-to-1 Mux

Interface:



S	F
0	X
1	Y

Implementations

- gates (fast or slow), pass transistors

What's the Big Deal?

E.g., a processor interface book

Worse for computers, in general - *a tower of abstraction*

- Application software
- System software (OS and compiler/assembler/linker)
- Hardware (CPU, memory, I/O)

Each interface is complex and implemented with layer below

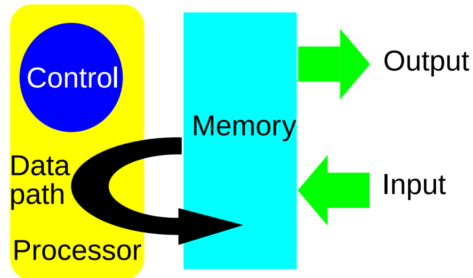
Abstraction keeps unnecessary details hidden

Hundreds of engineers to build one product

Basic Division of Hardware

In space and time

In space



Basic Division of Hardware

In time

• Fetch the instruction from memory	add r1, r2, r3
• Decode the instruction - what does this mean?	
• Read input operands	read r2, r3
• *Perform operation*	*add*
• Write results	write to r1
• Determine next instruction	pc := pc + 4

Classes of Computers

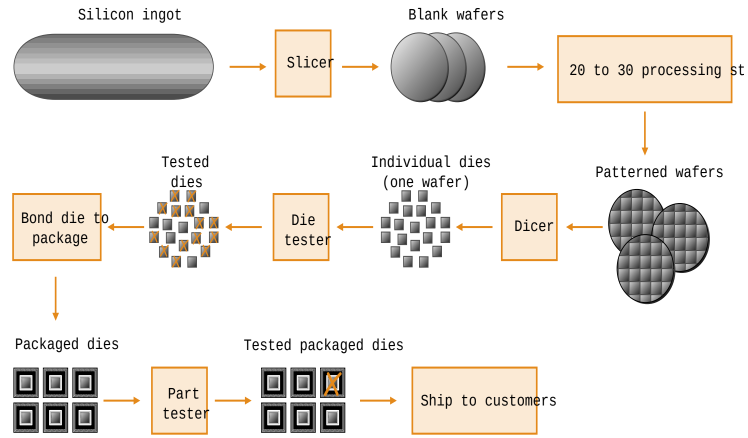
- Supercomputer $5 million - 20 million
- Mainframe $0.5 million - 4 million
- Server $10 thousand - 200 thousand
- PC/Workstation $1 thousand - 10 thousand
- Network Computer/WebTV $300 - 1000
- Embedded computer $1 - 10 ("invisible" like electric motor)
- Future Disposables 1-100 cents!

Building computer chips

Complex multi-step process

- slice ingots -> wafers
- process wafers (many steps) -> patterned wafers
- dice patterned wafers -> dies
- test dies -> good dies
- bond good die to package -> packaged dies (parts)
- test parts -> good parts
- ship to customers -> make money!

Building computer chips



Performance vs. Design Time

Time to deliver product is important

E.g., a new design will take 3 years to complete

- will be 3 times faster
- but if technology improves 50% per year
- in 3 years $1.5^3 = 3.38$
- so new design is worse!

Bottom Line

Designers must know BOTH software and hardware

Both contribute to layers of abstraction of computers

IC costs and performance

Compilers and Operating Systems