EE 4720—Computer Architecture

01-1

URL: https://www.ece.lsu.edu/ee4720/

RSS: https://www.ece.lsu.edu/ee4720/rss_home.xml

Offered by:

David M. Koppelman

3316R P. F. Taylor Hall, 578-5482, koppel@ece.lsu.edu, https://www.ece.lsu.edu/koppel

Office Hours: Monday - Friday, 15:00-16:00.

Should already know:

How to design a computer.

Will learn:

How to design a *good* computer.

01-2

Prerequisites By Course:

EE 3752, Microprocessor Systems, and EE 3755, Computer Organization.

Prerequisites By Topic:

- Logic design.
- Computer organization.
- Assembly-language programming.

Optional Text

"Computer architecture, a quantitative approach," John L. Hennessy & David A. Patterson, or "Computer organization & design," David A. Patterson & John L. Hennessy.

01-3

Course Content

Topics that cover modern general-purpose microprocessors:

- CPU pipelined implementations.
- Instruction set families: RISC (⟵ normal), CISC (⟵ old-fashioned), VLIW.
- Code optimization and performance.
- Benchmarks, and when to believe them.
- Implementing interrupts and floating point operations.
- Single-core parallelism: pipelining, superscalar, vector instructions.
- Basic parallel computer organizations: multi-core, cluster.
- Branch prediction and related techniques.
- Dynamic scheduling.
- Caches and memory.

Graded Material

01-4

Midterm Exam, 40%

Fifty minutes, closed book.

Final Exam, 40%

Two hours, closed book.

Yes, it's cumulative.

Homework, 20%

Written and computer assignments.

Lowest grade or unsubmitted assignment dropped.

Course Usefulness

01-5

Material in Course Needed For:

- General-purpose processor designers and testers.
- Special-purpose processor designers and testers.
- Compiler writers.
- Programmers of high-performance systems, including games.
- Developing exploits for low-level vulnerabilities.
- Answering job interview questions.

01-6

Course Resources

- Slides and other material via `https://www.ece.lsu.edu/ee4720/`
- Web site also has homework assignments, exams, grades, and other material.
- Announcements are on course home page and available as a Web (RSS) Feed.

Instruction Set Architecture (ISA) Concept History

01-7

Key Concept: Split computer design into *Architecture Design* and *Implementation*.

Quick (and Temporary) Definitions

Instruction Set Architecture (ISA):
The Machine Language
Examples: Intel 64, Itanium, PowerPC.

Implementation:
The Chip
Examples: Intel Core, Athlon, G5.

An ISA can have multiple implementations . . .
. . . sometimes developed decades apart.

ISA Concept History

01-8

Early Computer Design

ISA and Implementation not considered totally separate things.

Start with requirements. (*E.g.*, Need division instructions.)

Develop processor hardware.

ISA is documentation of "implementation" (developed hardware).

Problems Faced by Early Computer Manufacturers

Realization that software development expensive & time-consuming.

Would like to offer past customers better machine that runs old software.

Engineers say a really better machine would break old software.

Marketing people rather not test customers' loyalty, so don't break sw.

ISA Concept History

01-9

ISA and IBM System/360

Concept of ISA crystallized by Amdahl, Blaauw, & Brooks.

Engineers designed ISA and implementation as separate entities.

ISA would ease development of new implementations.

System/360 developed in 1964 and its successors are still available.

Instruction Set Architecture & Implementation

01-10

Instruction Set Architecture (ISA):
Precise definition of computer's instructions and their effects.

It's all that programmer needs to program machine.

It's all that hardware designer needs to design machine.

Implementation: [of an ISA] (noun)
Hardware that executes instructions defined by the ISA.

Microarchitecture:
Organization and features used for an implementation.

ISA and implementation descriptions at
`https://www.ece.lsu.edu/ee4720/reference.html`.

What a Typical ISA Defines

Instructions. (Operations, encoding, etc.)

Data Formats (Integer, Floating Point, Vector/Packed)

Registers and Memory Organization.

Interrupts, exceptions, and traps.

Implementation-Dependent Features. (Memory control, custom features.)

ISA Design Goals

- Define what program is. (Instructions and their encodings.).
- Define what program will do. (Data types, operations.)
- Clearly distinguish defined, undefined, and implementation-dependent behavior.
- Enable good first implementation.

  Easy.

- Enable good future implementations.

  Requires foresight: See future needs and technology (*e.g.*, gates).

  Requires discipline: Sacrifice first-implementation performance.

  And maybe foolishness: Insufficient first-implementation sales.

How Damaging (to profit) Are Bad ISAs?

Examples of Bad (now) ISA Features

- Undefined FP rounding behavior. System/360
- Base + 16-Bit Offset Addressing. (IA-32)
- Not enough registers. (IA-32)
- Overly complex instructions. (VAX)

Bad features can be overcome by implementation...
... especially if engineering budget is very large.

Some Architectures and their Implementations

01-14

Architecture: IBM *System/360*

Developed in 1964 for large business computers.

Designers appreciated and popularized the difference between ISA and implementation.

First planned family of computers.

Very successful, successor machines still in use under name z/Architecture.

First Implementations: Model 30, Model 75.

Some Architectures and their Implementations

01-15

Architecture: DEC *VAX*

Developed by the Digital Equipment Corporation.

First implementation: VAX 11/780, introduced in 1978.

Came to dominate *minicomputer* market in 1980s.

Single-chip implementations were developed but not successful.

Later DEC developed the Alpha ISA, more suitable to 1990s technology.

Some Architectures and their Implementations

01-16

Architecture: Intel *IA-32* and *Intel 64*

Initially developed in 1978 for small systems.

First processor: 8086, implements small part of IA-32.

Major improvements in amount of memory addressable by subsequent chips, 80186, 80286 (1982).

The 80386 (1985) could host a modern 32-bit operating system.

Later chips implemented ISA extensions for multimedia and data movement . . .
. . . and continued to incorporate microarchitectural innovations.

> 80486 (1989), Pentium (1992), Pentium Pro (1995), Pentium II (1997), Pentium III (1999), Pentium 4 (2000), Core Duo (2006), Core i7 6gen (2015), Core i9-9980XE (2018). . .

Unlike System/360, the way it would be used was not forseen.

Includes unpopular features, such as small memory segments.

Nevertheless, implementations have competed well with modern ISAs.

Some Architectures and their Implementations

01-17

Architecture: DEC (then Compaq, now HP) *Alpha*

An example of a *RISC* processor.

Designed for easy programming.

Designed for easy implementation.

RISC programs are larger than others, but run faster.

Developed for a 25-year lifetime.

First implementation: DECchip 21064 (1992).

Later implementations: 21264, 21364.

Implementations are usually the fastest processors.

Alas, Compaq plans to discontinue it.

Some Architectures and their Implementations

01-18

Architecture: *Itanium* (née IA-64)

First general purpose *VLIW* ISA.

ISA helps processor overcome problems in turn-of-the-century processors.

First implementation: Itanium (same name as architecture) (2000).

Radically different from other processors.

Did not succeed.

This Course

01-19

This Course

ISA Features and Families

- Reference ISA: MIPS 32
- Families: RISC, CISC, VLIW
- Features: Common to the semi-exotic.
- Specific: SPARC, IA-32/Intel 64, Itanium, VAX, PowerPC, ARM A64, etc.

Microarchitecture (Implementation Techniques)

- Basic Pipelining
- Deep Pipelining, Superscalar, VLIW, Multicore.
- Branch Prediction
- Dynamic Scheduling (Out-of-order execution.)
- Memory and Caches