CSE 378
Machine Organization
and Assembly Language Programming

Winter 2006

John Zahorjan
Lucas Kreger-Stickles

Today

- Part 1: Course Mechanics
- Part 2: Course Overview

Mechanics: Course Goals / Orientation

- For 97% of us, computer architecture is “hardware”
  - It’s what’s above CSE 370 and below CSE 451
- One focus of CSE 378 is how this software is organized, and how to make it fast
- We’re also going to be interested in the “hardware/software interface”
  - What does a compiler do?
  - What does an OS do?
  - What support does the hardware provide?

Mechanics: Prerequisites

- CSE 370
  - Binary / hex integers
  - Basic machine organization: memory, registers, ALU, control, clock-cycle
  - (378 is logical organization, not logic / physical characterisitcs)
- Programming
  - Java – not so much Java programming, as running Java programs
    - javac, java, classpath, jar, an editor
  - C – we’ll be using C--, a C subset, but we won’t be doing much programming in it.
  - Unix – we’ll be using a Unix shell (cygwin, at least) in very modest ways. (We’ll also be using Windows.)

Mechanics: Homework

- Some problems from the book
- The majority of the work will be building a working machine
  - Three incremental projects
  - Working in pairs if you like
    - Dividing the workload isn't easy
  - The final result will be a working processor that runs an operating system and a simple shell (plus applications)
- The challenge is mastering breadth (rather than depth)

Mechanics: Exams

- Two midterms
  - Wednesday, January 25
  - Friday, February 24 (subject to change)
- One final
  - Wednesday, March 15 (8:30-10:20)

Mechanics: Grading

- 45% homeworks
- 10% first midterm
- 15% second midterm
- 25% final (covers entire quarter)
- 5% other

Mechanics: Late Policy

- Assignments due beginning of lecture on due date
  - Mostly electronic turn-in
  - We *could* be very rigid about the exact turn-in time…
- 20% / day late penalty
- 2 free extension days (at your discretion)
  - Make sure to clearly notify the TA

Mechanics: Academic Misconduct

- *"In general, any activity you engage in for the purpose of earning credit while avoiding learning, or to help others do so, is likely to be an act of Academic Misconduct."*
- Different people learn best in different ways.
- It's never cheating to interact with course staff.

Mechanics: Interacting with Live Course Staff

- Lectures
  - Speaking up is good (for everyone, but especially me).
- Sections
  - Oriented towards clarifying issues with lectures / homeworks, rather than providing additional information.
- Office hours:
  - Me: Tuesdays, 2:00-3:00 (Sieg 534), by appointment, whenever
  - Lucas: TBD

Mechanics: Interacting with Course Staff

- E-mail
- Anonymous feedback
  - Link off course home page to provide it
    - Go faster / go slower
    - Can we have an extension?
    - More / less homework
  - Link off course home page to read it
    - All submitted anonymous feedback that has "permission to post publicly" checked, minus anything libelous
- Course wiki
  - User-editable web
- Class mailing list
  - Your @cs.washington.edu account is already enrolled
  - Mostly one-way communication

Brief Intermission

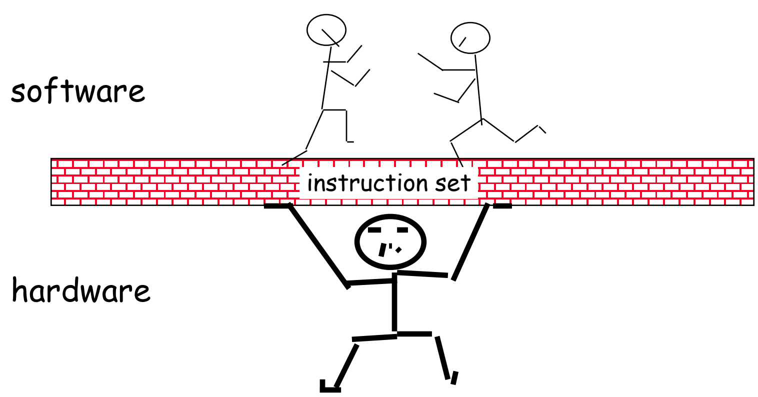
((More) Questions?)

What is "Computer Architecture"?

Computer Architecture =

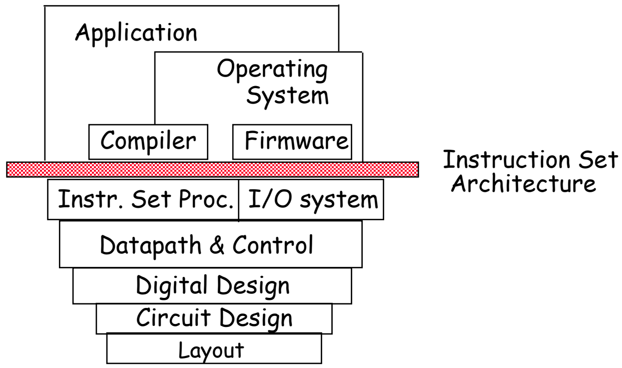
- Instruction Set Architecture (ISA) +
- Machine Organization + …

The Instruction Set: a Critical Interface



Lesson from history:
Push complex functionality into software –
it's more flexible, and it ends up being faster.

What is "Computer Architecture"?

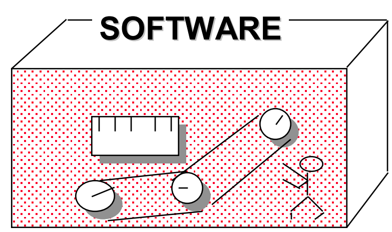


Instruction Set Architecture

(subset of Computer Architecture)

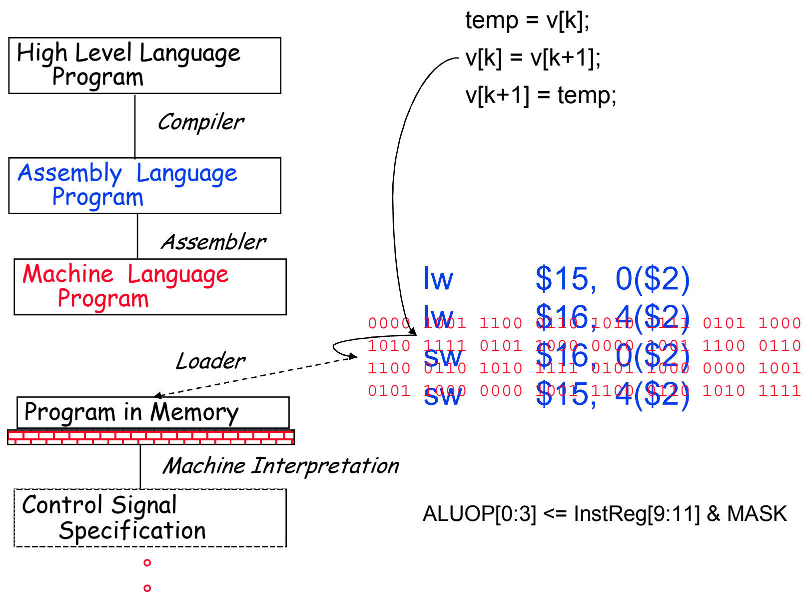
"... the attributes of a [computing] system as seen by the programmer, *i.e.*, the conceptual structure and functional behavior, as distinct from the organization of the data flows and controls the logic design, and the physical implementation."

– Amdahl, Blaaw, and Brooks, 1964



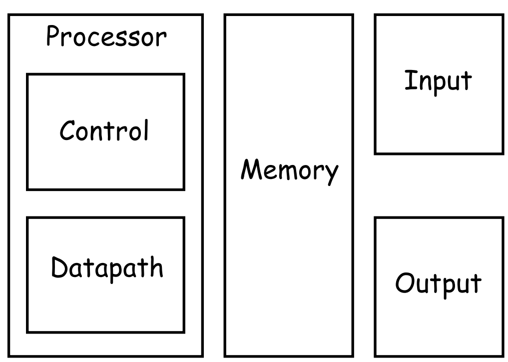
- Organization of Programmable Storage
- Data Types & Data Structures: Encodings & Representations
- Instruction Set
- Instruction Formats
- Modes of Addressing and Accessing Data Items and Instructions
- Exceptional Conditions

Levels of Representation

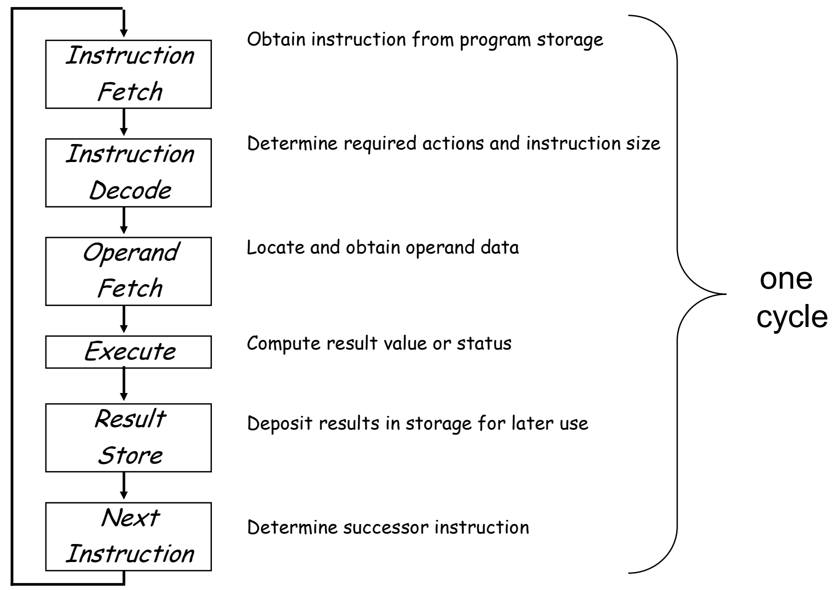


Machine Organization

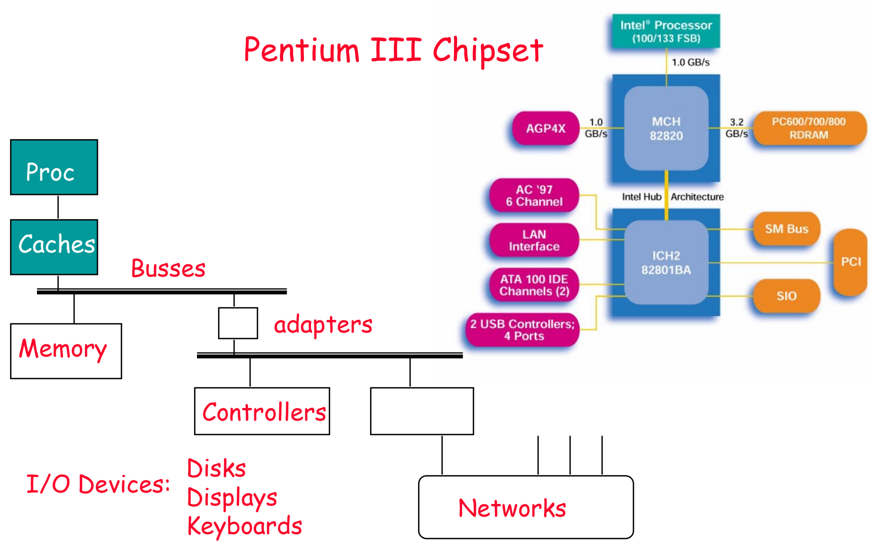
- Since 1946 all computers have had 5 components



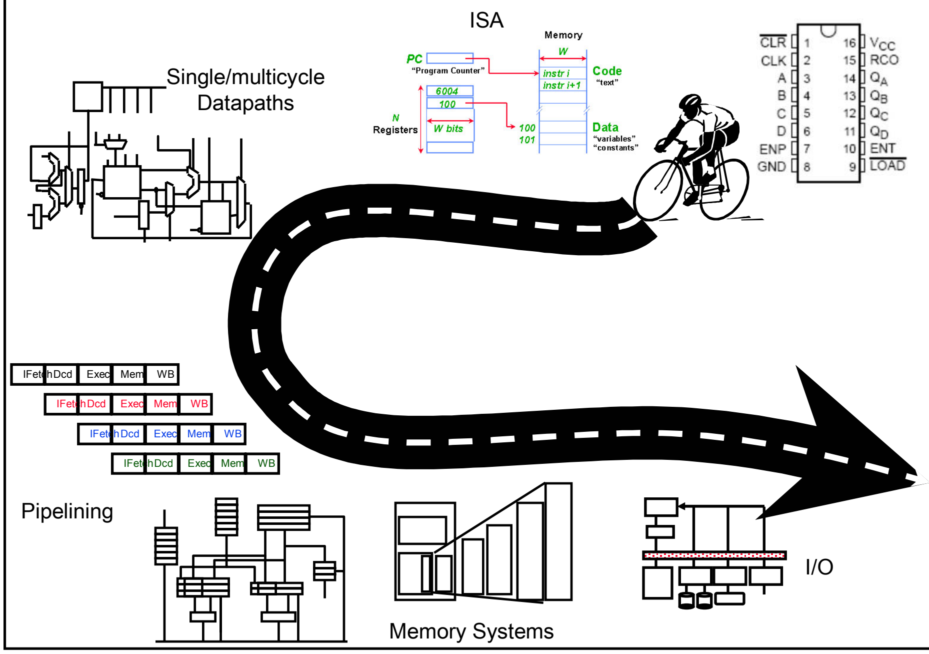
Basic Execution Cycle



A Machine (is not just a CPU)



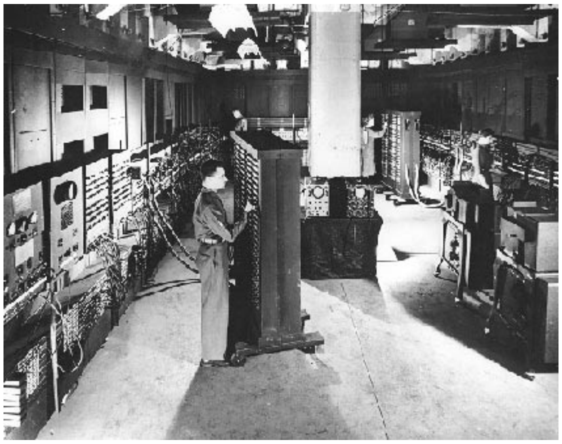
Where are We Going?



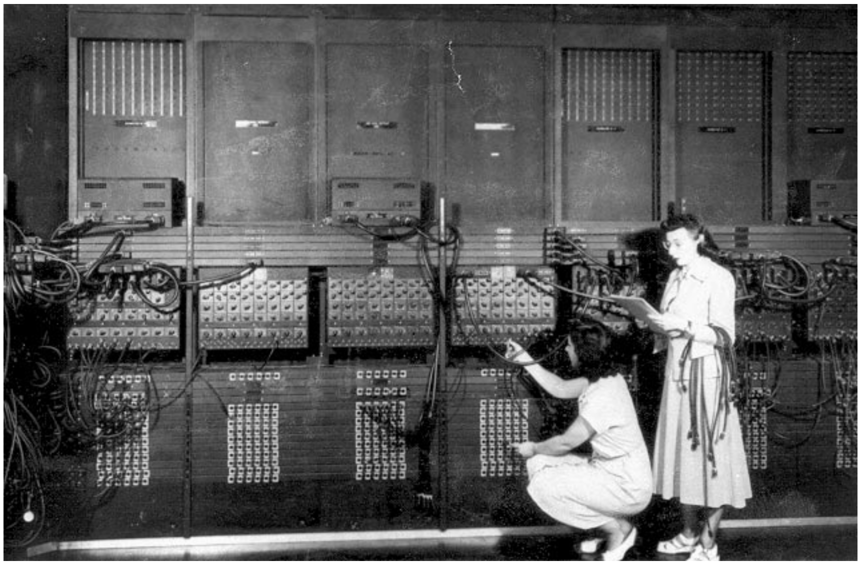
A Bit of History
(And What is Moore’s Law?)

ENIAC: 1946

Cost to build: $486,804.22
17,468 vacuum tubes, 5,000 additions/second (5 Kips)
30 feet x 50 feet, 30 tons
Cost to operate (electricity): $650/hr. (idling)



ENIAC Programming



IBM S360/67: 1967

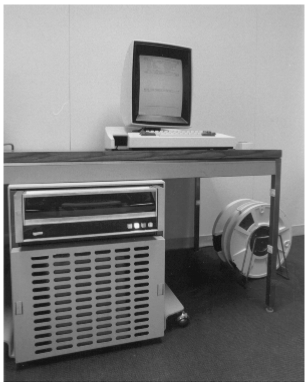
Cost: $3,000,000
1,000,000 instructions/sec. (1 Mip)
512KB “core” memory ($1,000,000/MB)
352MB disk

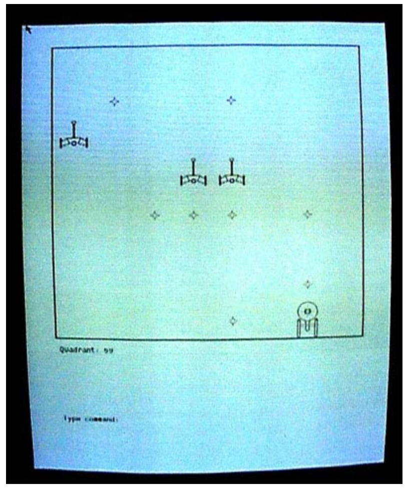
X 11/780: circa 1980

Cost: $150,000
1 “VAX Mip”
1MB Ram

erox Alto: 1973

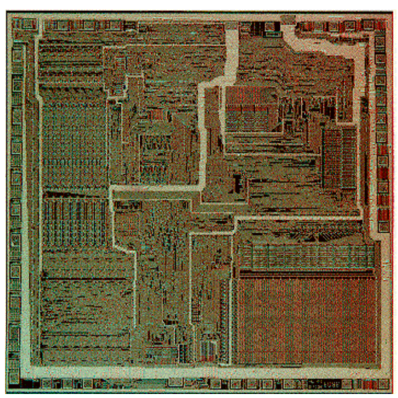
Cost: $32,000 (research)
1 Mip
Bitmap display
Mouse
“Microsoft Word”





el 8086 (x86): 1978

Cost: ~$350
5-10 MHz (~1Mip)
29,000 transistors



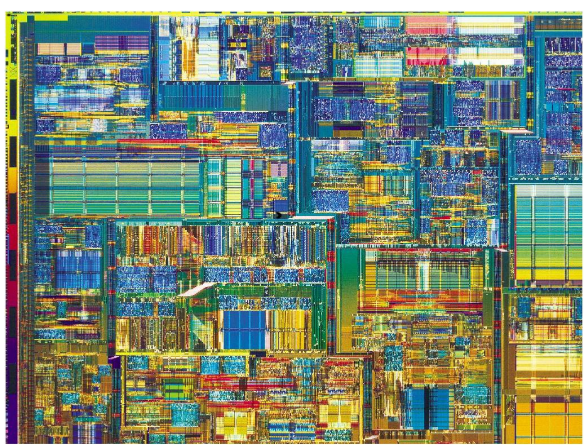
cessors + Workstation Concept

8/12/1981 IBM introduces its Personal Computer, which uses Microsoft's 16-bit operating system, Microsoft® MS-DOS® version 1.0, plus Microsoft BASIC, Microsoft COBOL, Microsoft Pascal, and other Microsoft products.

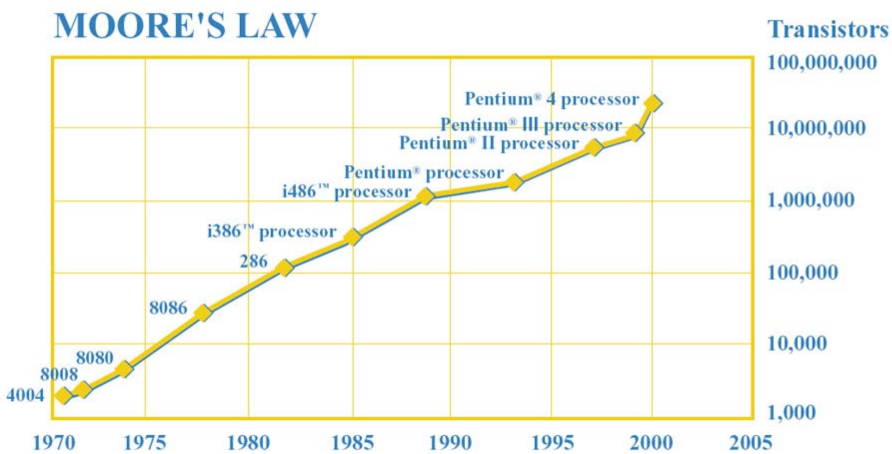
1984: Original Mac
Cost: $3,500
8 MHz
64KB RAM
No disk (400KB floppy)

Pentium 4: 2000’s

Cost: $100’s
2 GHz
42,000,000 transistors



Moore's Law: 1975



One Way to View Architecture as a Topic

What are we going to do with all those transistors?

or

How can we make *programs* run faster at the rate processor speeds are improving?

A Remark About the Weight of History

A *computing system* is more than just hardware – there is an enormous base of software required (e.g., OS, compilers, applications).

Architectures tend to undergo evolution, rather than revolution, since *backward compatibility* is required to gain adoption.

On the other hand, the *machine organization* (implementation of the ISA) is free to change as dramatically as the designer thinks is beneficial.