CSE 378
Machine Organization
and Assembly Language Programming

Spring 2009

John Zahorjan
Ivayla Dermendjieva
David St. Hilaire

Today

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

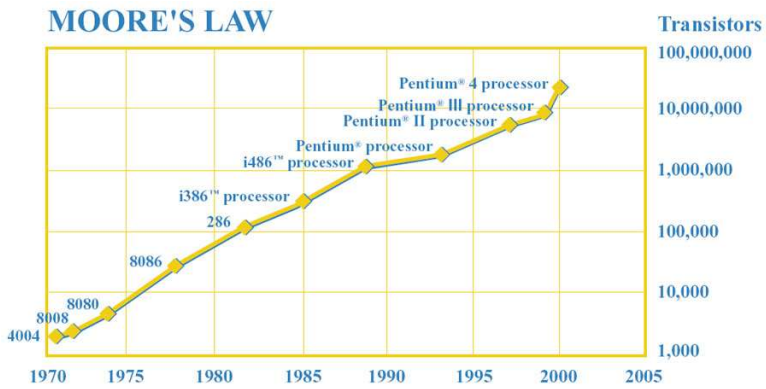
Mechanics: Course Goals: Part 1

- For 97% of us, computer architecture is “hardware”
  - It’s what’s above CSE 370 and below CSE 451 (and 341 and142/3 and 341 and 401 and ...)
- 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?

This Version of 378

- Software simulators, rather than hardware
- Pros:
  - Greater breadth (by which I mean depth)
    - Compiler, OS, shell, and applications
  - “Easier debugging”
  - Fewer hidden gotchas
  - “Work where you want, when you want”
- Cons
  - New set of tools (Cebollita / SMOK)
  - Not as useful a pre-req for CSE 466 / 471
    - But a better pre-req for a lot of other courses, and non-courses

Moore's Law (1975)



Course Goals, Part II: Parallelism

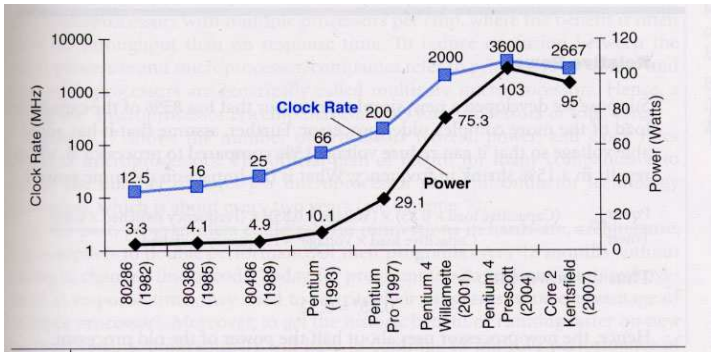


FIGURE 1.15 Clock rate and Power for Intel x86 microprocessors over eight generations and 25 years. The Pentium 4 made a dramatic jump in clock rate and power but less so in performance. The Prescott thermal problems led to the abandonment of the Pentium 4 line. The Core 2 line reverts to a simpler pipeline with lower clock rates and multiple processors per chip.

Parallel Execution of Sequential Code

- "Sequential code"
  - c=0;
    if (a<0) c=1;
- Parallel Execution Approaches
  - Pipelining
  - Multiple issue
  - Multi-threading
  - Multi-core
- All these approaches have something to teach us about turning sequential code into "explicitly parallel code"

Multicores (Explicitly Parallel Code)

Parallelism has always been critical to performance in computing, but it was often hidden. Chapter 4 will explain pipelining, an elegant technique that runs programs faster by overlapping the execution of instructions. This is one example of *instruction-level parallelism*, where the parallel nature of the hardware is abstracted away so the programmer and compiler can think of the hardware as executing instructions sequentially.

Forcing programmers to be aware of the parallel hardware and to explicitly rewrite their programs to be parallel had been the "third rail" of computer architecture, for companies in the past that depended on such a change in behavior failed (see **Section 7.14** on the CD). From this historical perspective, it's startling that the whole IT industry has bet its future that programmers will finally successfully switch to explicitly parallel programming.

Mechanics: Prerequisites

- CSE 370
  - Binary / hex integers
  - Basic machine organization: memory, registers, ALU, control, clock-cycle (?)
  - (378 is logical organization, not logic)
- CSE 303 / 143
  - 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.)
  - Shell and *make*

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

What the assignments show you

- What you see using a computer is created by layering
  - Software using software (libraries)
  - Software using hardware
- Very impressive things are ultimately achieved, even though each layer is doing something extremely simple
  - So long as they can do it fast enough...
- Simple Is Good
  - Simple is fast
  - Simple is flexible

An example: Java

- You probably think of computers as “Java” (and vice versa)
- A problem people often have in this course is forgetting about that
  - The processor isn't trying to execute Java
  - It's very primitive:
    - add 0100010000111101 and 010111101000111
  - Java is a whole lot of adding (plus some other things, like conditionals)

Mechanics: Exams

- Two midterms
  - Monday, April 20 (subject to change)
  - Monday, May 18 (subject to change)
- One final
  - Monday, June 8 (8:30-10:20)

Mechanics: Grading

- 40% homeworks
- 10% first midterm
- 15% second midterm
- 30% final (covers entire quarter)
- 5% other

Mechanics: Late Policy

- Assignments:
  - 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
  - TAs: TBD

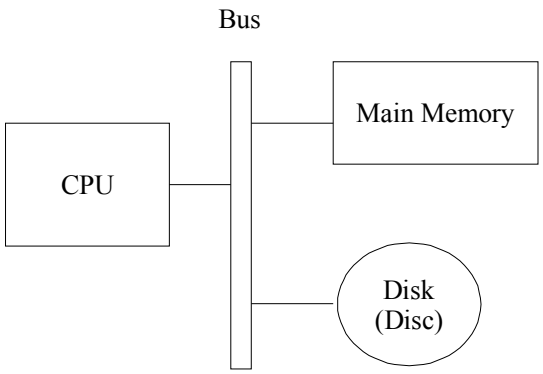
Mechanics: Interacting with Course Staff

- 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
  - You should subscribe (directions on course home page)
  - Mostly one-way communication

Brief Intermission

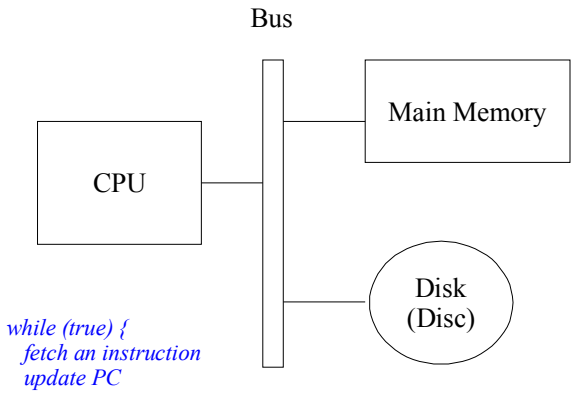
((More) Questions?)

One View of the Hardware (Simplified)



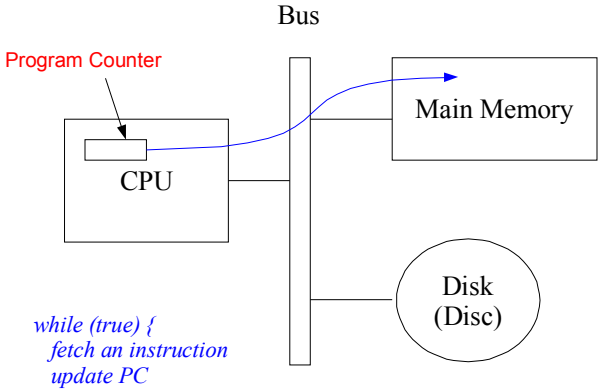
Where are your program's variables?
Where is your program?

Basic Operation of CPU



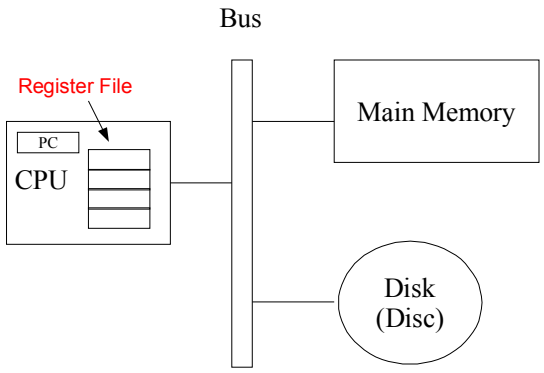

```
while (true) {
  fetch an instruction
  update PC
  execute instruction
}
```

How Does the CPU "Name" the Next Instruction?




```
while (true) {
  fetch an instruction
  update PC
  execute instruction
}
```

Performance: Accessing Memory Is Slow



The register file holds data (not instructions).
Registers are hardware – your program can't create them.

What Does an Instruction Look Like?

- 00100011?
- Instruction Format:
  - Defines which bits mean what

This is just an example!

OP	Operand 1	Operand 2

  - OP: 4 bits
    - "opcode": What the operation is
      - Add (0000)
      - Subtract (0001)
      - Or (0010)
      - Etc.
  - Operands: 2 bits each
    - What to operate on
      - Register 0 (00), 1 (01), 2 (10) or 3 (11)

What Does an Instruction Look Like?

- 00100111?

Or contents of register 3 with contents of register 1 and put result in register 1

- Instruction Format:
  - Defines which bits mean what

OP	Operand 1	Operand 2

  - OP: 4 bits
    - "opcode": What the operation is
      - Add (0000)
      - Subtract (0001)
      - Or (0010)
      - Etc.
  - Operands: 2 bits each
    - What to operate on
      - Register 0 (00), 1 (01), 2 (10) or 3 (11)

or $1,$3

From C to Machine Language

High-level language (C)

```
a = b + c;
```

Compiler

Assembly Language (MIPS)

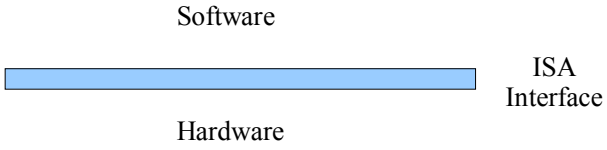
```
add $16, $17, $18
```

Assembler

Binary Machine Language (MIPS)

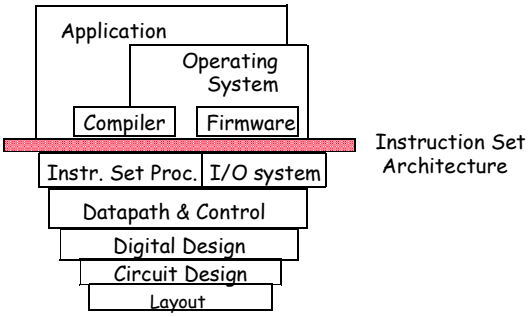
```
01010111010101101...
```

Another View: Instruction Set Architecture



- *What instructions are there?*
- *How are they encoded?*
- *How many registers are there?*
- *How do you "name" data in memory?*
- *Etc.*

What is "Computer Architecture"?



MIPS R2000 ISA

- 32 registers (in register file)
- Each register is 32 bits wide
- “load-store” architecture
  - Source operands must be in registers
  - Result goes into a register
- MIPS is a “reduced instruction set” (RISC) architecture
  - Designed for parallelism (pipelining)
- mips.com sells architectures (not hardware)