CS 316
Intro to Computer System Organization and Programming

Kavita Bala
Fall 2007
Computer Science
Cornell University

Information

• Instructor: Kavita Bala (kb@cs.cornell.edu)

• Tu/Th 1:25-2:40 Hollister B14
Course Objective

- Bridge the gap between hardware and software
  - How a processor works
  - How a computer is organized

- Establish a foundation for building higher-level applications
  - How to understand program performance
  - How to understand where the world is going

Who am I?

- Current life
  - Graphics
  - Parallel processing in graphics

- Previous life
  - Compilers
  - Operating Systems
  - Networks
Who am I?

- Reality
- Parallel Processing
- Scalable Algorithms
- Understanding of human vision
- Games
- Time
- Slow

Course Staff

- TAs
  - Adam Arbree (arbree@cs.cornell.edu)
  - Jed Liu (liujed@cs.cornell.edu)
  - Robert Burgess (burgess@cs.cornell.edu)

- Undergraduate consultants:
  - Daniel Margo, Matt Oliveri, Ben Pu

- AA: Amy Fish (amyfish@cs.cornell.edu)

- Sections:
  - Tu/Th/Fr 2:55-4:10
  - Wednesday section cancelled
Book

- Computer Organization and Design
  - The Hardware/Software Interface

- David Patterson, John Hennessy
  - Get revised printing from summer

Course

- Programming Assignments: 6-8
  - Work in groups of 2

- Homeworks: 4-5
  - Work alone

- 2 prelims, 1 final project
Grading

• Breakdown
  – 35-50% Projects (approx. 8 assignments)
  – 30-40% Prelims (2)
  – 15-20% Homeworks (approx. 4-5)
  – 5% Flexgrade (participation, attitude, improvement and effort)

Administrivia

• http://www.cs.cornell.edu/courses/cs316/2007fa
  – Updates
  – Schedule
  – Lecture notes
  – Office hours
  – Homeworks, etc.
Communication

• Email
  – cs316-l@cs.cornell.edu
  – The email alias goes to me and the TAs, not to whole class

• Mailing list for students
  – Sign up sheet

Sections & Projects

• Sections start next week

• Projects will be done in two-person teams
  – We will pair you up if you don’t have a preferred partner
  – Start early, time management is key
  – Manage the team effort
Academic Integrity

• All submitted work must be your own
  – OK to study together
  – Cannot share solutions however
• Project groups submit joint work
  – Same restrictions apply to projects at the group level
  – Cannot be in possession of someone else’s solution
• Closed-book exams, no calculators
Transistors and Gates

![Transistor and Gate Diagram]

Truth table

<table>
<thead>
<tr>
<th>In</th>
<th>Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Logic and State

![Logic and State Diagram]
A Calculator

Basic Computer System

• A processor executes instructions
  – Processor has some internal state in storage elements (registers)
• A memory holds instructions and data
  – von Neumann architecture: combined inst and data
• A bus connects the two
Simple Processor

MIPS Processor

© Kavita Bala, Computer Science, Cornell University
Example Machine Organization

- TI SuperSPARC™ TMS390Z50 in Sun SPARCstation20

- Floating-point Unit
- Integer Unit
- Inst Cache
- Ref MMU
- Data Cache
- Store Buffer
- Bus Interface

- MBus Module
- MBus control
- M-S Adapter
- L64852

- DRAM Controller
- Serial
- Kbd
- Mouse
- Audio
- RTC
- Boot PROM
- Floppy
Why should you care?

- Bridge the gap between hardware and software
  - How a processor works
  - How a computer is organized

- Establish a foundation for building higher-level applications
  - How to understand program performance
  - How to understand where the world is going

Moore’s Law

- 1965
  - 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
  - 2300 transistors, 1 MHz clock (Intel 4004) - 1971
  - 16 Million transistors (Ultra Sparc III) – 2001
  - 42 Million transistors, 2 GHz clock (Intel Xeon) – 2001
  - 55 Million transistors, 3 GHz, 130nm technology, 250mm² die (Intel Pentium 4) – 2004
  - 290+ Million transistors, 3 GHz (Intel Core 2 Duo) – 2007
Processor Performance Increase

Where is the Market?

© Kavita Bala, Computer Science, Cornell University
GPUs: Faster than Moore’s Law

Graph courtesy of Professor John Poulton (from Eric Haines)

© Kavita Bala, Computer Science, Cornell University

Computer System Programming

© Kavita Bala, Computer Science, Cornell University
Instruction Set Architecture (ISA)

- ISA
  - abstract interface between hardware and the lowest level software
  - user portion of the instruction set plus the operating system interfaces used by application programmers
MIPS R3000 ISA

• Instruction Categories
  – Load/Store
  – Computational
  – Jump and Branch
  – Floating Point
    ▪ coprocessor
  – Memory Management

<table>
<thead>
<tr>
<th>OP</th>
<th>rs</th>
<th>rt</th>
<th>rd</th>
<th>sa</th>
<th>funct</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>jump target</td>
</tr>
</tbody>
</table>

Calling Conventions

main:
  jal mult
Laftercall1:
  add $1,$2,$3
  jal mult
Laftercall2:
  sub $3,$4,$5

mult:
  addiu sp,sp,-4
  sw $31, 0(sp)
  beq $4, $0, Lout
  ...
  jal mult
Linside:
  ...
Lout:
  lw $31, 0(sp)
  addiu sp,sp,4
  jr $31
Data Layout

```
blue() {
    pink(0,1,2,3,4,5);
}
pink() {
    orange(10,11,12,13,14);
}
```

Buffer Overflows

```
blue() {
    pink(0,1,2,3,4,5);
}
pink() {
    orange(10,11,12,13,14);
}
orange() {
    char buf[100];
    gets(buf);  // read string, no check
}
```
Parallel Processing

• Spin Locks

• Shared memory, multiple cores

• Etc.

Can answer the question.....

• A: for i = 0 to 99
  – for j = 0 to 999
    ▪ A[i][j] = Computation ()

• B: for j = 0 to 999
  – for i = 0 to 99
    ▪ A[i][j] = complexComputation ()

• Why is B 15 times slower than A?
Applications

• Distributed ray tracer
  – Multiple cores running highly parallel application
  – Great images!

• Core war
  – Corrupt your neighbors context!

Covered in this course
Nuts and Bolts: Switches, Transistors, Gates

A switch

- A switch is a simple device that can act as a conductor or isolator
- Can be used for amazing things…
Switches

- Either (OR)

- Both (AND)

- But requires mechanical force

Transistors

- Solid-state switch
  - The most amazing invention of the 1900s

- PNP and NPN
Transistors

- Semi-conductor

![Schematic of an NPN transistor]

- **NPN Transistor**
  - Connect E to C when base = 1

- **PNP Transistor**
  - Connect E to C when base = 0

P and N Transistors

- **PNP Transistor**
  - Connect E to C when base = 0

- **NPN Transistor**
  - Connect E to C when base = 1
Then and Now

- The first transistor
  - on a workbench at AT&T Bell Labs in 1947
- An Intel Pentium
  - 125 million transistors

Inverter

- Function: NOT
- Called an inverter
- Symbol:
  \[
  \text{in} \quad \rightarrow \quad \text{out}
  \]
- Truth table
  \[
  \begin{array}{cc}
  \text{In} & \text{Out} \\
  0 & 1 \\
  1 & 0 \\
  \end{array}
  \]
- Useful for taking the inverse of an input
  - CMOS: complementary-symmetry metal-oxide-semiconductor
NAND Gate

- Function: NAND
- Symbol:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>out</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

NOR Gate

- Function: NOR
- Symbol:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>out</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>