Computer Architecture

Yipeng Huang

Rutgers University

January 16, 2024
Table of contents

Curiosity
  - Computer science
  - Computer systems

Community
  - The students
  - The instructors

Learning
  - Preview of syllabus

Expectations
  - Accessing the class & resources
  - Lecture and short quizzes
  - Midterm and final exams
  - Programming assignments
  - Academic honesty and integrity

A New Golden Age for Computer Architecture

Accessing iLab Linux machines
Warm up questions

1. What are some fundamental concepts in computer science?
2. How does a computer work?
What this class is about: abstractions

Intermediate courses in computer science:

**CS 112: Data structures**
learned about how to store data and manipulate data with algorithms

**CS 205/206: Discrete structures**
learn about the discrete and continuous mathematics governing computer science

**CS 211: Computer architecture**
learn about the abstractions that make programs run on computer building blocks

**CS 213: Software methodology**
learn how to organize complex programs

**CS 214: Systems programming**
learn how to interact with the operating system and network
What are the parts of a computer?

1. What are the parts of a computer?
What are the parts of a computer?

- Central Processing Unit
- Registers
- Caches
- Memory
- File Systems
- Network
- Screen
- User interfaces
- GPU
- FPGA
- ASIC
- Circuit boards
- Chips
- Integrated circuits
- Logic gates
- Transistors
What are desirable properties for computers?

1. What are desirable properties for computers?
What are desirable properties for computers?

- Correctness
- Performance
- Efficiency
- Security and privacy
- Fairness
- Positively impacts human condition
Important computing abstractions

Abstractions
A way to hide the details of an underlying system so you (users & programmers) can be more creative.

Low-level programming
C, assembly language, machine code, instruction set architecture

The memory hierarchy
File system, main memory, caches, data representations

Digital logic
Pipelines, registers, flip flops, arithmetic units, gates
Table of contents

Curiosity
   Computer science
   Computer systems

Community
   The students
   The instructors

Learning
   Preview of syllabus

Expectations
   Accessing the class & resources
   Lecture and short quizzes
   Midterm and final exams
   Programming assignments
   Academic honesty and integrity

A New Golden Age for Computer Architecture

Accessing iLab Linux machines
The students

Take a look around to meet your fellow students.

As of today, 244 registered students. From prior year distributions:

- ≈ 10 seniors, ≈ 55 juniors, ≈ 100 sophomores, ≈ 55 first-years
- ≈ 10 BAIT, ≈ 40 CS, ≈ 10 finance, ≈ 5 ITI, ≈ 10 mathematics, ≈ 130 undeclared, ≈ 10 prebusiness
Welcome all to class

- We welcome in this class diverse backgrounds and viewpoints spanning various dimensions: race, national origin, gender, sexuality, disability status, class, religious beliefs.
- We will treat each other with respect and strive to create a safe environment to exchange questions and ideas.
The instructors

Prof. Yipeng Huang
yipeng.huang@rutgers.edu

Teaching assistants

- Khyati Doshi (MS recitation PTL)
- Nate Blum (undergraduate recitation PTL)
- Pedro Torres (undergraduate recitation PTL)
My research is in abstractions that allow us to use novel computer architectures such as quantum and analog computers.

- I am looking for students who want to pursue research projects.
- I also teach CS 558—Quantum Computing: Programs and Systems
- Worked with DARPA to investigate feasibility of using analog electronic circuits for scientific computation.
Table of contents

Curiosity
   Computer science
   Computer systems

Community
   The students
   The instructors

Learning
   Preview of syllabus

Expectations
   Accessing the class & resources
   Lecture and short quizzes
   Midterm and final exams
   Programming assignments
   Academic honesty and integrity

A New Golden Age for Computer Architecture

Accessing iLab Linux machines
What this class is about

Course objective
Sustain and enhance your (the student’s) interest and confidence in computer science.

Specific learning goals
Throughout the course, students will learn about important computing abstractions such as low-level programming, data representations, the memory hierarchy, and digital logic via case studies that are representative of real-world computer systems.
Low-level programming: C

Learn a new and foundational programming language.

```c
#include <stdio.h>

int main() {
    printf("Hello, World!");
    return 0;
}

https://www.tiobe.com/tiobe-index/
```
Data representations

```c
void show_squares()
{
    int x;
    for (x = 5; x <= 5000000; x+=10)
        printf("x = %d x^2 = %d\n", x, x*x);
}
```

- Numbers are represented using a finite word size
- Operations can overflow when values too large
  - But behavior still has clear, mathematical properties

**Figure:** Credit: Computer Systems: A Programmer’s Perspective
Low-level programming: assembly

Study the interface between software and hardware.

MOV EAX, [EBX] ; Move the 4 bytes in memory at the address contained in EBX into EAX

MOV [ESI+EAX], CL ; Move the contents of CL into the byte at address ESI+EAX

MOV DS, DX ; Move the contents of DX into segment register DS
The memory hierarchy

Computer Memory Hierarchy

- Processor registers: very fast, very expensive
- Processor cache: very fast, very expensive
- Random access memory: fast, affordable
- Flash/USB memory: slower, cheap
- Hard drives: slow, very cheap
- Tape backup: very slow, affordable

Figure: Credit: wikimedia
The memory hierarchy

```c
void copyij(int src[2048][2048],
            int dst[2048][2048])
{
    int i,j;
    for (i = 0; i < 2048; i++)
        for (j = 0; j < 2048; j++)
            dst[i][j] = src[i][j];
}
```

```c
void copyji(int src[2048][2048],
            int dst[2048][2048])
{
    int i,j;
    for (j = 0; j < 2048; j++)
        for (i = 0; i < 2048; i++)
            dst[i][j] = src[i][j];
}
```

4.3 ms

81.8 ms

19 times slower!

(Measured on 2 GHz Intel Core i7 Haswell)

- Hierarchical memory organization
- Performance depends on access patterns
  - Including how step through multi-dimensional array

Figure: Credit: Computer Systems: A Programmer’s Perspective
Table of contents
Curiosity
   Computer science
   Computer systems
Community
   The students
   The instructors
Learning
   Preview of syllabus
Expectations
   Accessing the class & resources
   Lecture and short quizzes
   Midterm and final exams
   Programming assignments
   Academic honesty and integrity
A New Golden Age for Computer Architecture
Accessing iLab Linux machines
Accessing the class & resources

Canvas
Announcements, lecture slides, videos, quizzes, assignments, submissions.
https://rutgers.instructure.com/courses/261416

Long-range syllabus
Reading and programming assignment plan.
https://yipenghuang.com/teaching/2024-spring-211/

Textbooks
▶ Modern C: https://gustedt.gitlabpages.inria.fr/modern-c/
Lecture and short quizzes

Lectures
▶ In-person attendance here in SERC Room 111 is expected. If you are sick or have an emergency, use the Zoom livestream and recording.
▶ It benefits your learning to attend live, ask questions, and keep up.
▶ Videos will be posted within one day, access link on Canvas.

Short quizzes (5% of course grade)
▶ Ensure that you keep up with the class and check on basic concepts from previous week, and to collect feedback.
▶ 30 minutes for quiz, max two attempts, open for set time window.
Midterm and final exams

Midterm exam (10% of course grade)

- Tuesday after spring break, March 19, in class session time.
- Will be on C programming and data representations.
- Best way to prepare will be to master the programming assignments and quizzes.

Final exam (15% of course grade)

- Tuesday, May 7 12:00pm to 3:00pm.
- Standard final exam block time assigned by the registrar.
- Cumulative of whole course.
- Best way to prepare will be to master the programming assignments and quizzes.
Programming assignments (70% of course grade)

Students will apply essential knowledge about computer systems to modify and create new low-level software and hardware implementations via hands-on programming exercises.

0. Setting up your programming environment (1.25% of course grade)
1. A new language: C (6.25% of course grade)
2. Review of data structures and algorithms in C (12.5% of course grade)
3. Everything is a number, numbers are bits and bytes (12.5% of course grade)
4. How programs are represented and run in computers (12.5% of course grade)
5. How to create the illusion of fast and big memory (12.5% of course grade)
6. How to build computers from simple logic (12.5% of course grade)
Programming assignments

ilab

▶ All students need access to ilab to compile, run, and test programs in Linux.
▶ If you do not have access, sign up immediately: https://services.cs.rutgers.edu/accounts/activate/activate

Piazza

▶ Ask all questions if possible on Piazza.
▶ If you send the instructor or any of the TAs an email that is better addressed on Piazza, we will kindly ask you to repost your question on Piazza and we will answer it there.
▶ Sign up now:
  https://piazza.com/rutgers/spring2024/011982110508
Programming assignments

Automatic compiling, testing, and grading
  ▶ It is important that you carefully follow the specified output formats so that the testing framework can validate your program.

Submit on Canvas
  ▶ Start early.
  ▶ You can submit as many times as you wish.
  ▶ We will not accept late assignments; deadline will be enforced by Canvas.
Recitation code review study groups

Goals

- Give stylistic code review and feedback (avoid coding to satisfy autograder).
- Boost recitation attendance and give structure to recitation sections.

Mechanics

- Teams of \( \approx 5 \) students.
- Review and discuss code from previous assignment.
- As a team, present findings in 5 minute short summary.
Importance of writing your own code

### Ineffective Sorts

#### DEFINE HAILEYHEARTEDMERGESORT(LIST):
- IF LENGTH(LIST) < 2:
  - RETURN LIST
- Pivot = INT((LENGTH(LIST) / 2)
- A = HAILEYHEARTEDMERGESORT(LIST[:Pivot])
- B = HAILEYHEARTEDMERGESORT(LIST[Pivot:])
  \[\text{// UHHHHHHHHHHH} \]
- RETURN [A, B] // HOPE. SORRY.

#### DEFINE FASTBOGOSORT(LIST):
- // AN OPTIMIZED BOGOSORT
- // RUNS IN O(N LOG N)
- FOR N FROM 1 TO LOG(LENGTH(LIST)):
  - SHUFFLE(LIST);
    \[\text{// IF ISORTED(LIST):} \]
  - RETURN LIST
  - RETURN "KERNEL PAGE FAULT (ERROR CODE: 2)"

#### DEFINE JOINIERANDOQuickSort(LIST):
- OK SO YOU CHOOSE A PIVOT
- THEN DIVIDE THE LIST IN HALF
- FOR EACH HALF:
  - CHECK TO SEE IF IT'S SORTED
    NO WAY IT DOESN'T MATTER
  - COMPARE EACH ELEMENT TO THE PIVOT:
    THE BIGGER ONES GO IN A NEW LIST
    THE SMALLER ONES GO INTO UM
    THE SECOND LIST FROM BEFORE
    HANG ON, LET ME NAME THE LISTS
    THIS IS LIST A
    THE NEW ONE IS LIST B
    PUT THE BIG ONES INTO LIST B
    NOW TAKE THE SECOND LIST
    CALL IT LIST, UH, A2
    WHICH WAS THE PIVOT IN?
    SCRATCH ALL THAT
    IT JUST RECURSIVELY CALLS ITSELF
    UNTIL BOTH LISTS ARE EMPTY
    RIGHT?
    NOT EMPTY, BUT YOU KNOW WHAT I MEAN
    AM I ALLOWED TO USE THE STANDARD LIBRARIES?

#### DEFINE PANICSORT(LIST):
- IF ISORTED(LIST):
  - RETURN LIST
- FOR N FROM 1 TO 10000:
  - Pivot = RANDOM(0, LENGTH(LIST))
  - LIST = LIST[Pivot:] + LIST[:Pivot]
    // THIS CAN'T BE HAPPENING
    RETURN LIST
- IF ISORTED(LIST):
  - RETURN LIST
  - // OH JEEZ
  - // I'M GONNA BE IN SO MUCH TROUBLE
  LIST = [ ]
  SYSTEM("SHUTDOWN -H +5")
  SYSTEM("RM -RF /")
  SYSTEM("RM -RF ~*")
  SYSTEM("RM -RF /")
  SYSTEM("RD /S X: C:\*") // PORTABILITY
  RETURN [1, 2, 3, 4, 5]
Academic honesty and integrity

Study and practice programming to learn

- We adopt the collaboration guidelines table set by CS 112 Data Structures
  https://ds.cs.rutgers.edu/
- You are encouraged to discuss the homework with your classmates on Piazza.
- You are encouraged to research and study concepts online.

Importance of writing your own code

- But, you must not disclose your code or see your classmates’ code.
- You cannot look at answers online that are obviously specific to this class.
- Finding your own solution and writing and debugging your own code is vital to your learning. Copying someone else’s code short-circuits this process.
- We will use automatic tools to detect identical or similar submissions.

Rutgers Academic Integrity Policy

- https://nbprovost.rutgers.edu/academic-integrity-students
- Every offense will be reported to office of student conduct.
Table of contents

Curiosity
  Computer science
  Computer systems

Community
  The students
  The instructors

Learning
  Preview of syllabus

Expectations
  Accessing the class & resources
  Lecture and short quizzes
  Midterm and final exams
  Programming assignments
  Academic honesty and integrity

A New Golden Age for Computer Architecture
Accessing iLab Linux machines
Learning goal
At the end of this course, students should have the preliminary skills to design and evaluate solutions involving the computer software-hardware interface to address new problems.

1https://cacm.acm.org/magazines/2019/2/234352-a-new-golden-age-for-computer-architecture/fulltext
<table>
<thead>
<tr>
<th>Time Period</th>
<th>Analog Continuous-time Computing</th>
<th>Digital Discrete-time Computing</th>
</tr>
</thead>
<tbody>
<tr>
<td>1940s</td>
<td>Analog computers for rocket and artillery controllers.</td>
<td>Turing’s Bomba.</td>
</tr>
<tr>
<td>1950s</td>
<td>Analog computers for field problems.</td>
<td>1st transistorized digital computer.</td>
</tr>
<tr>
<td>1960s</td>
<td>1st transistorized analog computer.</td>
<td>Moore’s law projection for transistor scaling.</td>
</tr>
<tr>
<td>1980s</td>
<td>...</td>
<td>VLSI democratized.</td>
</tr>
<tr>
<td>1990s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2000s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2010s</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Architecture abstraction milestones**
- Stored program computer.
- Microprogramming.
- Instruction set architecture.
- Reduced instruction set computers.

**Figure:** Emerging Architectures for Humanity’s Grand Challenges, Yipeng Huang
History: computer architecture abstractions drove digital revolution

Figure: Credit: A New Golden Age for Computer Architecture
Present: power constraints driving diverse computer architectures

Figure: Credit: A New Golden Age for Computer Architecture
Present: power constraints driving diverse computer architectures

<table>
<thead>
<tr>
<th>Year</th>
<th>1940s</th>
<th>1950s</th>
<th>1960s</th>
<th>1970s</th>
<th>1980s</th>
<th>1990s</th>
<th>2000s</th>
<th>2010s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analog continuous-time computing</td>
<td>Analog computers for rocket and artillery controllers.</td>
<td>Analog computers for field problems.</td>
<td>1st transistorized analog computer.</td>
<td>Analog-digital hybrid computers.</td>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Stored program computer.</td>
<td>Microprogramming.</td>
<td>Instruction set architecture.</td>
<td>Reduced instruction set computers.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Present: a rapidly evolving and influential field of study

Heterogeneity
Multicore CPUS, GPUS, FPGAs, ASICs, TPUs

Energy conservation
Laptop and phone battery life, datacenter energy consumption

Security
Spectre / Meltdown

Virtualization
Docker, Amazon AWS
Present: a rapidly evolving and influential field of study

CS 211 lays foundations for many areas of computer systems and science

- Internet technology
- Security
- Distributed systems
- Database implementation
- Parallel programming
- Operating systems
- Systems programming
- Programming languages and compilers
Future: post-Moore’s Law computer architectures

- Quantum chemistry & high energy physics
  - Quantum circuit computers

- High dimensional nonlinear optimization
  - Quantum annealers

- Computational neuroscience & pattern recognition
  - Analog neuromorphic networks

- Fluid dynamics & plasma physics
  - Analog continuous-time accelerators

Image sources: Wikimedia.org
Future: post-Moore’s Law computer architectures

New and extreme workload challenges

Multicore CPUs, GPUs, FPGAs, ASICs, analog, quantum, etc.

Open challenges in emerging architectures:

Problem abstractions
- How do you accurately solve big problems?

Programming abstractions
- Can you borrow ideas from conventional computing?

Architecture abstractions
- How to interface with the unconventional hardware?
Table of contents

Curiosity
  Computer science
  Computer systems

Community
  The students
  The instructors

Learning
  Preview of syllabus

Expectations
  Accessing the class & resources
  Lecture and short quizzes
  Midterm and final exams
  Programming assignments
  Academic honesty and integrity

A New Golden Age for Computer Architecture

Accessing iLab Linux machines
Why use Linux?

Do you have Linux? Trick question...
Why use Linux?

- Stable
- Open source
- Flexible: all form factors (wearables, IoT, Raspberry Pi, Roku, Android, laptops, iLab, web hosting, warehouse-scale datacenters)
- A critical piece of infrastructure for practicing computer science
Key steps to get going

1. **Activate account:** [https://services.cs.rutgers.edu/accounts/](https://services.cs.rutgers.edu/accounts/)
2. **Familiarize yourself with CS department infrastructure:** [https://resources.cs.rutgers.edu/docs/new-users/beginners-info/](https://resources.cs.rutgers.edu/docs/new-users/beginners-info/)
3. **Use what you are familiar with to log onto iLab remotely. Command line:** Windows command line, macOS, terminal, PuTTy. **Graphical:** X2Go... [https://resources.cs.rutgers.edu/docs/other/working-at-home/](https://resources.cs.rutgers.edu/docs/other/working-at-home/)
4. **Use what you are familiar with to move files.** SCP, Filezilla, Cyberduck... [https://resources.cs.rutgers.edu/docs/file-storage/accessing-files-remotely/](https://resources.cs.rutgers.edu/docs/file-storage/accessing-files-remotely/)
5. **Use what you are familiar with to edit files.** Vim, Emacs, other text editors, VS Code...