CS 24

Introduction to Computing Systems

Introduction: Perspectives on Computing Systems

Outline

Outline

Two Important Perspectives

The Programmer's Perspective

You are a $\{$Java, Python, C++, $\ldots\}$ programmer, and you want to write fast, safe programs that might interact with your OS.

Two Important Perspectives

The Programmer's Perspective

You are a {Java, Python, C++, ...} programmer, and you want to write fast, safe programs that might interact with your OS.

The System Builder's Perspective

You are a "system's person" and you want to understand how your entire computer works by building the pieces one by one.

Two Important Perspectives

The Programmer's Perspective

You are a {Java, Python, C++, . . . } programmer, and you want to write fast, safe programs that might interact with your OS.

The System Builder's Perspective

You are a "system's person" and you want to understand how your entire computer works by building the pieces one by one.

Poll (M2)

Which perspective(s) do you feel are most applicable to you?

a The Programmer's Perspective
b The System Builder's Perspective
c Both
d Neither

CS 24 Works From Both Perspectives

For The Programmer

We will discuss:

- important system realities that will change how you program

CS 24 Works From Both Perspectives

For The Programmer

We will discuss:

- important system realities that will change how you program
- how to interact with the OS

CS 24 Works From Both Perspectives

For The Programmer

We will discuss:

- important system realities that will change how you program
- how to interact with the OS
- how to write concurrent programs

CS 24 Works From Both Perspectives

For The Programmer

We will discuss:

- important system realities that will change how you program
- how to interact with the OS
- how to write concurrent programs
- how to write code with a security mindset

CS 24 Works From Both Perspectives

For The Programmer

We will discuss:

- important system realities that will change how you program
- how to interact with the OS
- how to write concurrent programs
- how to write code with a security mindset

For The System Builder

You will build:

CS 24 Works From Both Perspectives

For The Programmer

We will discuss:

- important system realities that will change how you program
- how to interact with the OS
- how to write concurrent programs
- how to write code with a security mindset

For The System Builder

You will build:

- a virtual machine

CS 24 Works From Both Perspectives

For The Programmer

We will discuss:

- important system realities that will change how you program
- how to interact with the OS
- how to write concurrent programs
- how to write code with a security mindset

For The System Builder

You will build:

- a virtual machine
- a small compiler

CS 24 Works From Both Perspectives

For The Programmer

We will discuss:

- important system realities that will change how you program
- how to interact with the OS
- how to write concurrent programs
- how to write code with a security mindset

For The System Builder

You will build:

- a virtual machine
- a small compiler
- a memory allocator

CS 24 Works From Both Perspectives

For The Programmer

We will discuss:

- important system realities that will change how you program
- how to interact with the OS
- how to write concurrent programs
- how to write code with a security mindset

For The System Builder

You will build:

- a virtual machine
- a small compiler
- a memory allocator
- a garbage collector

CS 24 Prepares You For Future Systems Courses

Computer Architecture	Graphics	Databases
Parallel Computing	Networks	Distributed Systems
Operating Systems	Compilers	Security

Adam's Perspective

I like to geek out on the stuff I'm teaching. So, I've inserted a bit of what I think is cool into this course.

Adam's Perspective

I like to geek out on the stuff I'm teaching. So, I've inserted a bit of what I think is cool into this course.

- I think **compilers** and **interpreters** are cool. So, we'll spend some time thinking about the low-level parts of a compiler.

Adam's Perspective

I like to geek out on the stuff I'm teaching. So, I've inserted a bit of what I think is cool into this course.

- I think **compilers** and **interpreters** are cool. So, we'll spend some time thinking about the low-level parts of a compiler.
- I think **security** is cool. So, we'll spend some time thinking about the "security mindset" and how to break things.

Adam's Perspective

I like to geek out on the stuff I'm teaching. So, I've inserted a bit of what I think is cool into this course.

- I think **compilers** and **interpreters** are cool. So, we'll spend some time thinking about the low-level parts of a compiler.
- I think **security** is cool. So, we'll spend some time thinking about the "security mindset" and how to break things.
- I think **real-world software** is cool. So, we'll spend some time re-writing core pieces of the system.

The Plan

Overview

We're going to work our way up the "abstraction ladder" emphasizing things that **all** programmers need to know.

The Plan

Overview

We're going to work our way up the "abstraction ladder" emphasizing things that **all** programmers need to know.

We'll pay special attention to uncovering the truth behind the lies.

The Plan

Overview

We're going to work our way up the "abstraction ladder" emphasizing things that **all** programmers need to know.

We'll pay special attention to uncovering the truth behind the lies.

Lies? What lies? Here's a few important ones we hope to dispel throughout the term:

The Plan

Overview

We're going to work our way up the “abstraction ladder” emphasizing things that **all** programmers need to know.

We'll pay special attention to uncovering the truth behind the lies.

Lies? What lies? Here's a few important ones we hope to dispel throughout the term:

- Lie 1: We compute with integers

The Plan

Overview

We're going to work our way up the "abstraction ladder" emphasizing things that **all** programmers need to know.

We'll pay special attention to uncovering the truth behind the lies.

Lies? What lies? Here's a few important ones we hope to dispel throughout the term:

- Lie 1: We compute with integers
- Lie 2: You only need Python

The Plan

Overview

We’re going to work our way up the “abstraction ladder” emphasizing things that **all** programmers need to know.

We’ll pay special attention to uncovering the truth behind the lies.

Lies? What lies? Here’s a few important ones we hope to dispel throughout the term:

- Lie 1: We compute with integers
- Lie 2: You only need Python
- Lie 3: Memory is like an array

The Plan

Overview

We're going to work our way up the "abstraction ladder" emphasizing things that **all** programmers need to know.

We'll pay special attention to uncovering the truth behind the lies.

Lies? What lies? Here's a few important ones we hope to dispel throughout the term:

- Lie 1: We compute with integers
- Lie 2: You only need Python
- Lie 3: Memory is like an array
- Lie 4: Constants don't matter

The Plan

Overview

We're going to work our way up the "abstraction ladder" emphasizing things that **all** programmers need to know.

We'll pay special attention to uncovering the truth behind the lies.

Lies? What lies? Here's a few important ones we hope to dispel throughout the term:

- Lie 1: We compute with integers
- Lie 2: You only need Python
- Lie 3: Memory is like an array
- Lie 4: Constants don't matter
- Lie 5: Your computer runs all your programs at the same time

Outline

Our Expectations

Our Expectations

- you know data structures at the level of CS 2

Our Expectations

- you know data structures at the level of CS 2
- you know C at the level of CS 3

Our Expectations

- you know data structures at the level of CS 2
- you know C at the level of CS 3
- you understand that correctness is critical in system's programming

Our Expectations

- you know data structures at the level of CS 2
- you know C at the level of CS 3
- you understand that correctness is critical in system's programming
- The Pre-Test takes no more than **eleven hours** to complete

Your Expectations

Your Expectations

- course staff will do our best to help you when you get stuck

Your Expectations

- course staff will do our best to help you when you get stuck
- course staff will grade each assignment within two weeks of when you turn it in

Your Expectations

- course staff will do our best to help you when you get stuck
- course staff will grade each assignment within two weeks of when you turn it in
- course staff will hold an **obscene** number of office hours

Your Expectations

- course staff will do our best to help you when you get stuck
- course staff will grade each assignment within two weeks of when you turn it in
- course staff will hold an **obscene** number of office hours
- Prof. Blank's "door" is always open

Aside: What this course is NOT

Aside: What this course is **NOT**

- This is **not** a C course

Aside: What this course is **NOT**

- This is **not** a C course

- This is **not** an EE course

Aside: What this course is **NOT**

- This is **not** a C course
- This is **not** an EE course
- This is **not** a hardware course

Grading

Grading Breakdown

- pretest $= 5\%$
- projects $=$ varying percentages
- lecturcises $= 30\%$
- final $= 10\%$

Warnings

Grading

Grading Breakdown

- pretest $= 5\%$
- projects $=$ varying percentages
- lecturcises $= 30\%$
- final $= 10\%$

Warnings

- Code that fails the correctness tests that we provide will receive no credit

Grading

Grading Breakdown

- pretest $= 5\%$
- projects = varying percentages
- lecturcises $= 30\%$
- final $= 10\%$

Warnings

- Code that fails the correctness tests that we provide will receive no credit
- We reserve the right to have private tests. Passing all the tests we give you does not mean your code is perfect.

Grading

Grading Breakdown

- pretest $= 5\%$
- projects $=$ varying percentages
- lecturcises $= 30\%$
- final $= 10\%$

Warnings

- Code that fails the correctness tests that we provide will receive no credit
- We reserve the right to have private tests. Passing all the tests we give you does not mean your code is perfect.
- We will not accept projects late due to misuse of `git`.

Grading

Grading Breakdown

- pretest $= 5\%$
- projects $=$ varying percentages
- lecturcises $= 30\%$
- final $= 10\%$

Warnings

- Code that fails the correctness tests that we provide will receive no credit
- We reserve the right to have private tests. Passing all the tests we give you does not mean your code is perfect.
- We will not accept projects late due to misuse of `git`.
- If you do not get any credit on the coding portion, you will not get any credit on the written portion.

Regrade Requests

To maintain consistency, all regrade requests should go directly to Prof. Blank via e-mail. Do not attempt to contact TAs about grading questions.

Some Changes From CS 2/3...

- Office Hours!
  - OH are now in ANB 106 (which is called $(\text{CS})^2$ for Computer Science Collaboration Support).
  - OH The schedule has also changed a bit–we've removed office hours from days that were unpopulated and started them earlier at 3pm!
  - $(\text{CS})^2$ is a **new** dedicated space for undergraduates taking CS courses! If it's not in use for a course, you can just walk in and use it as a collaboration space! It has power, monitors to connect to, chargers, and dry-erase tables!
- Lecturcises! tl;dr: Some of the exercises in lecture will now be turned in (later in the week). See syllabus for full details.

Outline

Bits, Bytes, and Binary

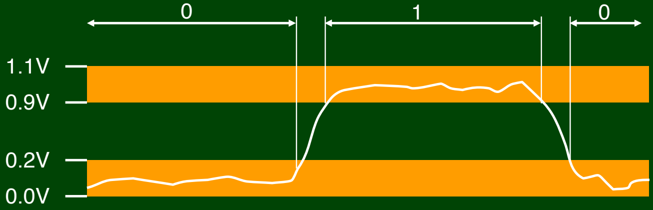
Bits are the “digits” of binary. Every bit is 0 or 1.

Bits, Bytes, and Binary

Bits are the "digits" of binary. Every bit is 0 or 1.

Why Binary?

- Easy to store with bistable elements
- Reliably transmitted on noisy and inaccurate wires

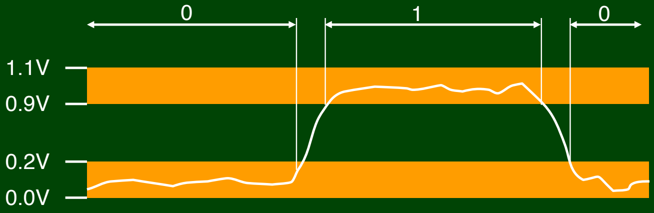


Bits, Bytes, and Binary

Bits are the "digits" of binary. Every bit is 0 or 1.

Why Binary?

- Easy to store with bistable elements
- Reliably transmitted on noisy and inaccurate wires



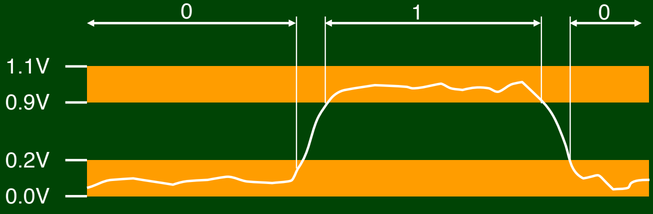
A **byte** is a group of 8 bits.

Bits, Bytes, and Binary

Bits are the "digits" of binary. Every bit is 0 or 1.

Why Binary?

- Easy to store with bistable elements
- Reliably transmitted on noisy and inaccurate wires



A **byte** is a group of 8 bits.

Computers are made entirely of circuits acting only on bits. **Everything** is represented as a series of bits.

Binary as Numbers

Read a binary number the same way as a decimal number.

Binary as Numbers

Read a binary number the same way as a decimal number.

$$1234 = 1000 + 200 + 30 + 4 = 1 \times 10^3 + 2 \times 10^2 + 3 \times 10^1 + 4 \times 10^0$$

Binary as Numbers

Read a binary number the same way as a decimal number.

$$1234 = 1000 + 200 + 30 + 4 = 1 \times 10^3 + 2 \times 10^2 + 3 \times 10^1 + 4 \times 10^0$$

In general, a decimal number with digits $d_{n-1} d_{n-2} \cdots d_0$ is $\sum_{k=0}^{n-1} d_k \times 10^k$.

Binary as Numbers

Read a binary number the same way as a decimal number.

$$1234 = 1000 + 200 + 30 + 4 = 1 \times 10^3 + 2 \times 10^2 + 3 \times 10^1 + 4 \times 10^0$$

In general, a decimal number with digits $d_{n-1}d_{n-2}\cdots d_0$ is $\sum_{k=0}^{n-1} d_k \times 10^k$.

Binary just replaces the 10 with a 2. That is:

In general, a binary number with **bits** $b_{n-1}b_{n-2}\cdots b_0$ is $\sum_{k=0}^{n-1} b_k \times 2^k$.

Binary as Numbers

Read a binary number the same way as a decimal number.

$$1234 = 1000 + 200 + 30 + 4 = 1 \times 10^3 + 2 \times 10^2 + 3 \times 10^1 + 4 \times 10^0$$

In general, a decimal number with digits $d_{n-1}d_{n-2}\cdots d_0$ is $\sum_{k=0}^{n-1} d_k \times 10^k$.

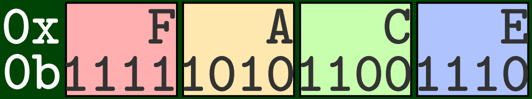
Binary just replaces the 10 with a 2. That is:

In general, a binary number with **bits** $b_{n-1}b_{n-2}\cdots b_0$ is $\sum_{k=0}^{n-1} b_k \times 2^k$.

It quickly becomes annoying to use binary, because there are so many digits to express even small numbers. So, we often write things (numbers, addresses, instructions) in **base 16**.

Hexadecimal

Base 16 is called **hexadecimal** and it uses the symbols $\{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F\}$, where A through F represent 10 through 15.



Hexadecimal

Base 16 is called **hexadecimal** and it uses the symbols $\{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F\}$, where A through F represent 10 through 15.

```
0x   F   A   C   E
0b1111101011001110
```

Note that 4 bits / hex digit (why?)

Hexadecimal

Base 16 is called **hexadecimal** and it uses the symbols $\{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F\}$, where A through F represent 10 through 15.

```
0x   F   A   C   E
0b1111101011001110
```

Note that 4 bits / hex digit (why?)

Poll

What is $(1337)_{10}$ in hexadecimal?

Data Representation

What does "`0xe282ac`" mean?

Data Representation

What does “`0xe282ac`” mean?

It’s an integer (14844588)...right?

Strings

Or is it an ASCII string?

Letter	Base-10	Binary
A	65	01000001
B	66	01000010
D	?	?
a	97	?
_	?	01011111
!	33	?

â‚¬

Strings

Or is it an ASCII string?

Letter	Base-10	Binary
A	65	01000001
B	66	01000010
D	?	?
a	97	?
_	?	01011111
!	33	?

â‚¬

Or a unicode code point (€)?

Color

Or a color?

Instructions

Or x86-64 instructions?

```
0:  e2 82                       loop   -124
2:  ac                          lodsb
```

Which Computer Systems?

In this course, we will limit our discussions in the following ways:

- We will only cover the **x86-64** architecture (not ARM or RISC-V)
- We will assume we're working with **Linux**
- Our case studies will be limited to the **Intel Nehalem** microarchitecture

We have a reference machine set up which you should use for **all of the projects**. The Pre-Test includes a section on getting this environment set up.

If you do not use `labradoodle`, we are not responsible for according deductions in your grades.