Welcome to CS 340: Introduction to Computer Systems
Course Website: https://courses.grainger.illinois.edu/cs340/

Description: Basics of computer systems. Number representations, assembly/machine language, abstract models of processors (fetch/execute, memory hierarchy), processes/process control, simple memory management, file I/O and directories, network programming, usage of cloud services. 3 credit hours.

Staff:
Prof. Wade Fagen-Ulmschneider <waf@>, Course Instructor
Teaching Associate Professor of Computer Science, Grainger College of Engineering

TAs: Ramya Bygari (rbygari2) and Daixuan Li (daixuan2)

Coursework and Grading
A total of 1,000 points are available in CS 340, along with many opportunities to earn extra credit. The points are broken down in the following way:

- **140 points**: Homeworks (1-2 /week)
  - Points even divided between the homeworks
  - Usually on PrairieLearn, but occasionally another platform
- **200 points**: Midterm Exams in CBTF (2 × 100 points)
  - Midterm 1 Exam (CBTF): Thurs, March 2 - Sat, March 4
  - Midterm 2 Exam (CBTF): Thurs, April 27 - Sat, April 29
- **440 points**: Machine Projects (11 weeks × 40 points)
  - Weekly machine problems, released every Tuesday and due the following Tuesday with a Wednesday grace period.
  - Extra credit for completing early milestones and completion.
- **220 points**: Final Project
  - Multi-week Final Project, presented during the final exam period instead of a final exam (no final exam!)

We never curve individual exam or assignment scores. Instead, if necessary, we may lower the points required for each grade cutoff to be lower than the stated cutoff. In no case will we raise the stated cutoff, so having 930 points will always earn you an “A” in the course.

<table>
<thead>
<tr>
<th>Points Range</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>[930, 1070)</td>
<td>A+</td>
</tr>
<tr>
<td>(870, 900)</td>
<td>B+</td>
</tr>
<tr>
<td>(770, 800)</td>
<td>C+</td>
</tr>
<tr>
<td>(670, 700)</td>
<td>D+</td>
</tr>
<tr>
<td>(600, 0]</td>
<td>F</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Points Range</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>(900, 930)</td>
<td>A-</td>
</tr>
<tr>
<td>(800, 830)</td>
<td>B-</td>
</tr>
<tr>
<td>(700, 770)</td>
<td>C-</td>
</tr>
<tr>
<td>(600, 670)</td>
<td>D-</td>
</tr>
<tr>
<td>(600, 0]</td>
<td>F</td>
</tr>
</tbody>
</table>

Foundations of Computer Systems
There are six major components to a computer, which we will refer to as the “foundations” of a computer system:

1. System-level Abstractions
   After covering the “foundations”, we will begin to abstract the entire system as single node and explore more complex topics:

2. 
3. 
4. 
5. 
6. 

System-level Abstractions
After covering the “foundations”, we will begin to abstract the entire system as single node and explore more complex topics:

1. 
2. 
3. 
Representing Data: Binary
All data within a computer is _______________; either 0 or 1.

Converting between base-2 and base-10:
- \(1_2 = 1_{10}\)
- \(10_2 = 2_{10}\)
- \(11_2 = 3_{10}\)
- \(100_2 = 4_{10}\)

Just like every digit has a “place value” in decimal (base-10), every digit has a “place value” in binary:

<table>
<thead>
<tr>
<th>Binary Number:</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x) Place Value:</td>
<td>2^7</td>
<td>2^6</td>
<td>2^5</td>
<td>2^4</td>
<td>2^3</td>
<td>2^2</td>
<td>2^1</td>
<td>2^0</td>
</tr>
<tr>
<td>Decimal Place Value:</td>
<td>128</td>
<td>64</td>
<td>32</td>
<td>16</td>
<td>8</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

Using this system, we can calculate more complex numbers:
- \(1011000_2 = 16_{10}\)
- \(1101000_2 = 102_{10}\)

Any value can be represented in binary by writing it in base-2, which be written in C by prefixing the number with \(0b\):
- \(4_{10} = 0b100_{2}\)
- \(7_{10} = 0b111_{2}\)
- \(18_{10} = 0b10010_{2}\)

Representing Data: Hexadecimal
Binary data gets really long, really fast! The number of students enrolled at University of Illinois is \(0b1100 1100 0110 1011\)
- To represent binary data in a compact way, we often will use hexadecimal -- or “base-16” -- denoted by the prefix \(0x\).

Hexadecimal Digits:

<table>
<thead>
<tr>
<th>Hex Number:</th>
<th>c</th>
<th>0</th>
<th>f</th>
<th>f</th>
<th>e</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>Place Value:</td>
<td>(16^5)</td>
<td>(16^4)</td>
<td>(16^3)</td>
<td>(16^2)</td>
<td>(16^1)</td>
<td>(16^0)</td>
</tr>
<tr>
<td>Decimal Place Value:</td>
<td>1848576</td>
<td>65536</td>
<td>4096</td>
<td>256</td>
<td>16</td>
<td>1</td>
</tr>
<tr>
<td>SUM:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Translation from Decimal to Hexadecimal:
- \(11_{10} = 0x0\)
- \(87_{10} = 0xf\)
- \(34_{10} = 0x22\)
- \(255_{10} = 0xff\)

Hexadecimal is particularly useful as it ________________:

University of Illinois student population last Fall (52,331):
- \(0b1100 1100 0110 1011\)

Number of people following Taylor Swift on Instagram (240,825,376):
- \(0b0000 1110 0101 1010 1011 0100 0010 0000\)

\(0x\)

\(01/hex.c\)

```c
4 int v1 = 0b10010;
5 int v2 = 0b11001;
6 int v3 = v1 + v2;
7 printf("%d\n", v3);
```

\(01/binary.c\)

```c
4 int v1 = 0b10010;
5 int v2 = 0b11001;
6 int v3 = v1 + v2;
7 printf("%d\n", v3);
```

\(01/hex.c\)

```c
4 int h1 = 0xc0ffee;
5 int h2 = 0xf00d;
6 printf("%x\n", h1 + h2);
```