CS 340
Computer Systems

#1: Course Introduction, Binary, and Hex
Jan. 17, 2023 · Wade Fagen-Ulmschneider

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!)
  - **Must be present on Monday, May 8, 2023.**

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.

Final Course Grades

Your course grade is determined by the number of points you earn:

Points	Grade	Points	Grade	Points	Grade
Exceptional	A+	[930, 1070)	A	[900, 930)	A-
[870, 900)	B+	[830, 870)	B	[800, 830)	B-
[770, 800)	C+	[730, 770)	C	[700, 730)	C-
[670, 700)	D+	[630, 670)	D	[600, 630)	D-
		(600, 0]	F		

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]:

[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:

$1_2 =$ $\_\_\_\__{10}$

$10_2 =$ $\_\_\_\__{10}$

$11_2 =$ $\_\_\_\__{10}$

$100_2 =$ $\_\_\_\__{10}$

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

Binary Number:	0	1	0	1	1	0	0	0
(x) Place Value:	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Decimal Place Value:	128	64	32	16	8	4	2	1
SUM:								

Using this system, we can calculate more complex numbers:

$101\ 1000_2 =$ $\_\_\_\__{10}$

$11\ 0100_2 =$ $\_\_\_\__{10}$

Any value can be represented in binary by writing it in base-2, which be written in C by prefixing the number with **0b**:

$4_{10} =$ $\_\_\_\__2$ = 0b

$7_{10} =$ $\_\_\_\__2$ = 0b

$18_{10} =$ $\_\_\_\__2$ = 0b

01/binary.c

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

Note: *All source code will always be available in the lecture repository! You are highly encouraged to run it yourself – they're minimally working examples of the topic at hand! :)*

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:

Place of Hexadecimal Numbers:

Hex Number:	c	0	f	f	e	e
Place Value:	16^5	16^4	16^3	16^2	16^1	16^0
Decimal Place Value:	1048576	65536	4096	256	16	1
SUM:						

Translation from Decimal to Hexadecimal:

$11_{10} =$ 0x

$34_{10} =$ 0x

$87_{10} =$ 0x

$255_{10} =$ 0x

Hexadecimal is particularly useful as it ____________________:

University of Illinois student population last Fall (52,331):				
0b	1100	1100	0110	1011
0x				

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

01/hex.c

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