Lecture 1:
Course Introduction

CSE 29: Systems Programming and Software Tools

Aaron Schulman (Shalev)

Lecture 1 Overview

- Introduce Staff
- Class overview
- Jumping right into Systems Programming

Personnel

- Instructor: Aaron Schulman
  - Office hours on course website

  Co-taught with: Joe Politz

- **Eight TAs and four tutors shared across both classes:**
  - Janet Vorobyeva
  - Andrew Pan
  - Elena Tomson
  - Rachel Lim
  - Kruti Dharanipathi
  - Reese Whitlock
  - Alex Yen
  - Savitha Ravi
  - Anya Chernova
  - Samuel Gonzalez
  - Miles Davis
  - Travis Henry

Who am I?


Former skateboarder
Networking Researcher
Dad
Hacker
Swimmer

Course Website / Syllabus

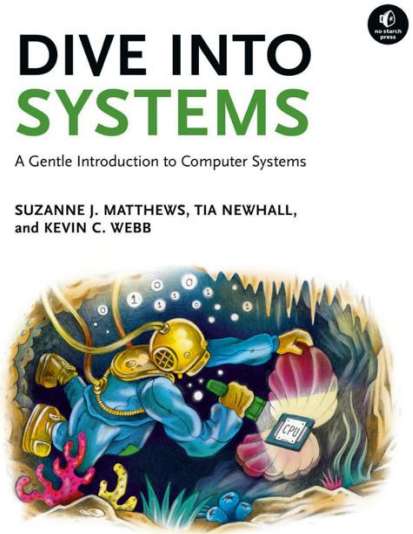
https://ucsd-cse29.github.io/fa25/

Podcasting and async learning

- Class **will not** be podcast
  - Lecture slides posted to website immediately after class
  - Lecture attendance is strongly encouraged, especially for insight into projects

- Readings will be assigned from "Dive into Systems" Textbook

HTML freely available at:
https://diveintosystems.org/book/



Expected Outcomes

Students who complete this course will be able to:

- **Learn how each function call makes the world's computers come alive!**

- Describe how a single C program runs on a computer
- Read, write, debug, and test C programs
- Use software tools to work with C programs (e.g., GDB - GNU Debugger)
- Control a computer from the terminal
- Use effective programming practices like incremental development, debugging, testing
- Describe how multiple programs can run at the same time on a computer
- Learn how to avoid basic security vulnerabilities in low-level code

Logistics

- *Big Assignments* Due on even weeks [2, 4, 6, 8, 10]
  - Mon – Problem set due
  - Thu – Programming assignment due
- Labs *every week*
- Exams weeks [3, 6, 9] (45min)
  - Taken at the computer-based testing facility
- Multiple tries!
  - Assignments: Resubmit ~2 weeks after deadline to improve score
  - Exams: Finals week exam makeups (2hr)
- Discussion
  - Go over problem sets

How *to* Pass CSE 29

- Attend lecture / discussion
  - But the material is in the book anyway, and the slides are posted online
  - Lecture is a great dedicated time slot to learn, and I will present concepts in an accessible way and connect to practical CS programming!
  - If you can not attend a lecture, please make a friend and ask for their notes
- Do the projects
  - But maybe I can just have ChatGPT do them for me?
  - Excellent practice for the exams, and assignments help with the project
  - Senior CS folks in industry report that the skills you get from this course will follow you throughout your career. LLMs can write code, but not be a CS!

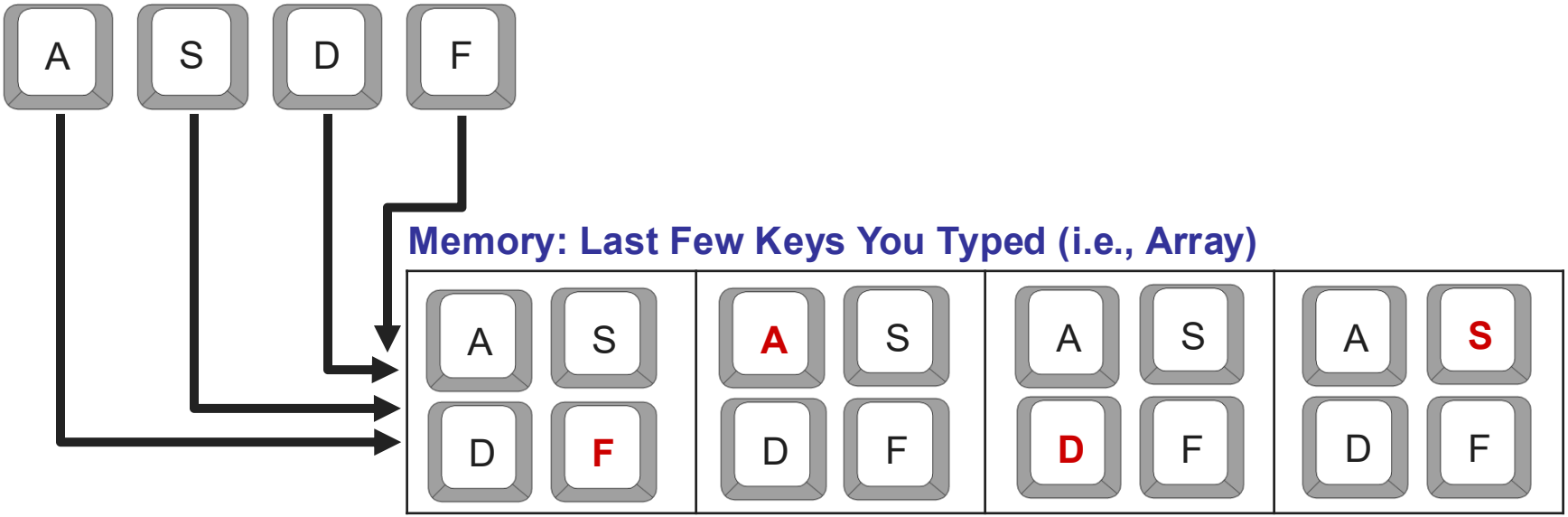
Week 0 Announcements

- Lab attendance is **required** and teaching **terminal-based tooling** happens there, make sure to go to lab
- Submit the welcome survey before lab on Tuesday of week 1
- Assignments, quizzes, and other things with deadlines will start in week 1
- Try to sign into PrarieLearn (https://us.prairielearn.com/)
  - Quizzes and Exams will use the PrarieLearn platform

Demo

How do we store a string of characters inside a computer?

What happens when you press a key?



How do computers store characters?

SRAM Cell: Stores state of a single bit [1 or 0]

Cheap!

- Made of sand (silicon)

Fast!

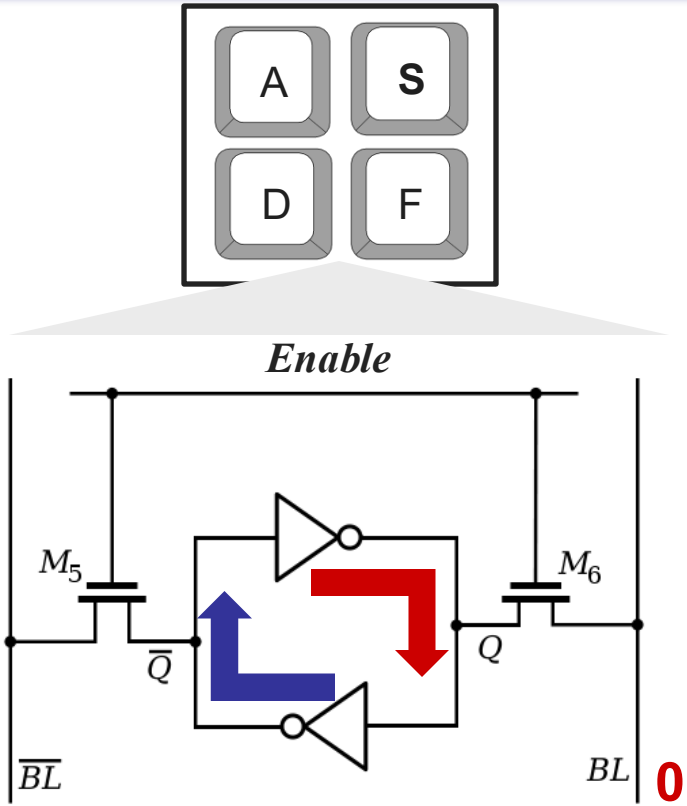
- Read/write billions of times per second

Accurate!

- Nearly always correctly returns the stored bit

Small!

- We can fit billions of them inside of a computer chip



How do we represent characters as bits?

1. Each character is assigned a number that represents it
   - ASCII Encoding: [**0-127]** represents all English Characters

2. Each number is assigned a set of bits that represent it
   - Binary numbering system

ASCII Table

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Logically, how is a number represented?

Two (Binary) electrical states:

High	Low
on	*off*
1	**0**

A number is an *array* of RAM cells with binary states:

1 byte = 8 bits

0	1	0	1	1	0	0	1

MSB **LSB**

Converting integers to/from binary arrays

1 bit

0 =0

1 =1

2 values: 2^1

2 bits

0	**0**	**=0**
0	1	**=1**
1	0	**=2**
1	1	**=3**

4 values: 2^2

3 bits

0	**0**	**0**	**=0**
0	0	1	**=1**
0	1	0	**=2**
0	1	1	**=3**
1	0	0	**=4**
1	0	1	**=5**
1	1	0	**=6**
1	1	1	**=7**

8 values: 2^3

What does each index of a bit array mean?

1 byte = 8 bits

128	64	32	16	8	4	2	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

MSB **LSB**

Bit array -> Integer conversions

$$\mathbf{1\ 0\ 1\ 0} = 1x2^3 + 0x2^2 + 1x2^1 + 0x2^0$$
$$= 8 + 0 + 2 + 0 = 10$$

$$\mathbf{1\ 1\ 1\ 0} = 1x2^3 + 1x2^2 + 1x2^1 + 0x2^0$$
$$= 8 + 4 + 2 + 0 = 14$$