Logistics

Primary Textbook

- Randal E. Bryant and David R. O'Hallaron,
  - *Computer Systems: A Programmer's Perspective*, **Third Edition** (CS:APP3e), Pearson, 2016
  - https://csapp.cs.cmu.edu
  - This book really matters for the course!
    - How to solve labs
    - Practice problems typical of exam problems
  - Electronic editions available
- Teams will be used for all communications
  - Available on the course plan page; please join

Recommended reading

- Brian Kernighan and Dennis Ritchie,
  - *The C Programming Language*, Second Edition, Prentice Hall, 1988
  - Guide to C by the designers of the language
  - Well-written, concise
  - A little dated
    - Doesn’t cover additions to C since 1988
(that’s thirty years ago…)
    - Casual about issues we consider serious problems now

Course Components

- Programming Assignments (~8)
  - 1-2+ weeks each
  - Provide in-depth understanding of an aspect of systems
  - Programming and measurement
  - Best **n-1** out of **n** assignments for your grade
- Examinations
  - Test your understanding of concepts & mathematical principles
  - Cover content until that period
- Class Activities (Bonus points)
  - Will intimate on the bonus later
- Project
  - May have one; will inform next week
  - If so, the weightage is 25%
  - If not, it will be equally distributed in the above components

Module 1 : Programs and Data

- Topics
  - Bit operations, arithmetic, assembly language programs
  - Representation of C control and data structures
  - Includes aspects of architecture and compilers

Module 2 : Memory Hierarchy

- Topics
  - Memory technology, memory hierarchy, caches, disks, locality
  - Includes aspects of architecture and OS

Module 3 : Virtual Memory

- Topics
  - Virtual memory, address translation, dynamic storage allocation
  - Includes aspects of architecture and OS

Module 4: Exceptional Flows

- Topics
  - Hardware exceptions, processes, process control, Unix signals, nonlocal jumps
  - Includes aspects of compilers, OS, and architecture

Module 5 : Networking, and Concurrency

- Topics
  - High level and low-level I/O, network programming
  - Internet services, Web servers
  - concurrency, concurrent server design, threads
  - I/O multiplexing with select
  - Includes aspects of networking, OS, and architecture

Assignment 0: C Programming

- We will add this today
- Not completing this assignment would automatically earn you 0 in future assignments
- It should all be review:
  - Basic C control flow, syntax, etc.
  - Explicit memory management, as required in C.
  - Creating and manipulating pointer-based data structures.
  - Implementing robust code that operates correctly with invalid arguments, including NULL pointers.
  - Creating rules in a Makefile
- If this assignment takes you more than 10 hours, you need to invest some more time for refreshing C

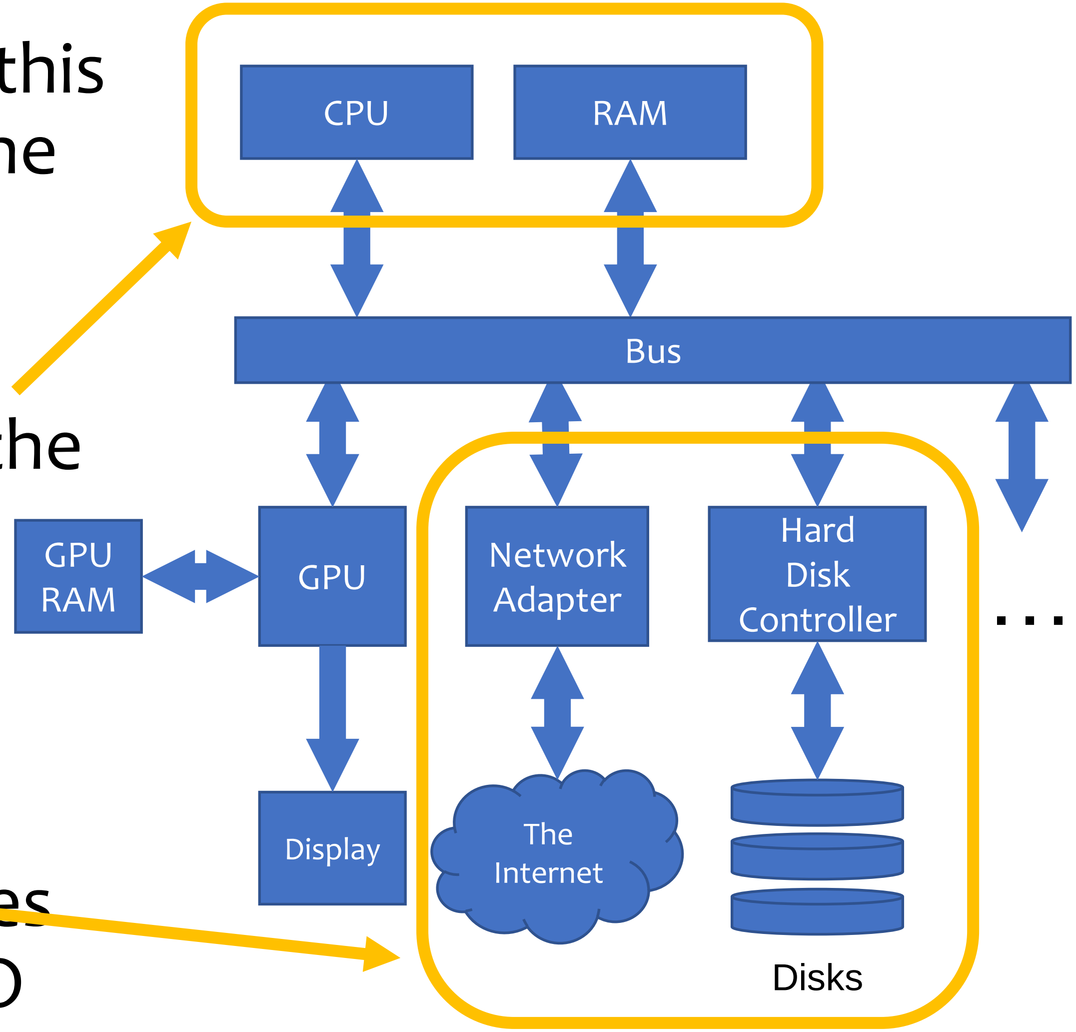
Policies

- Work groups
  - You must work alone on all assignments
- Grace days
  - **5 grace days for the semester**
  - **Limit of 0, 1, or 2 grace days per lab used automatically**
  - Covers scheduling crunch, out-of-town trips, illnesses, minor setbacks
  - Once grace day(s) used up, get penalized **25% per day**
  - No handins later than **3 days after due date**
- Catastrophic events
  - Major illness, death in family, …: please write to us
- Advice
  - Once you start running late, it's **really hard** to catch up
  - Try to save your grace days until the last few labs

Bits, Bytes and Integers

Roadmap – Inside a Computer

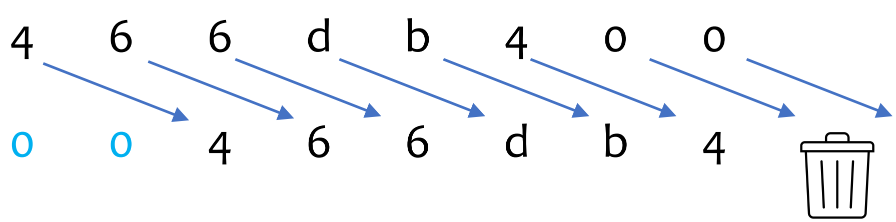
- You may have seen this block diagram, or one like it, before.
- For the first half of the course we'll be concentrating on the CPU and RAM
- Second half discusses disk and network I/O



Things You May Know Already

What value do you get if you arithmetic right-shift the hexadecimal number 0x466db400 by eight bits?

- Hexadecimal is just like decimal … if you have sixteen fingers.
- “Eight bits” is two hex digits.
- “Arithmetic right shift” is division by a power of two.



Things You May Know Already

On an x86-64 machine, how much space does this C struct take? That is, what is the value of sizeof(struct S)?

```
struct S {
    char c[2];    2 bytes
                  6 bytes wasted space
    char *p;      8 bytes
    int z;        4 bytes
                  4 bytes wasted space
}
```

24 bytes total

Things You May Know Already

```
/* a.c */
#include <stdio.h>
extern long x;
int main(void)
{
  printf("%ld\n", x);
  return 0;
}
```

```
/* b.c */

double x = 3.14159;
```

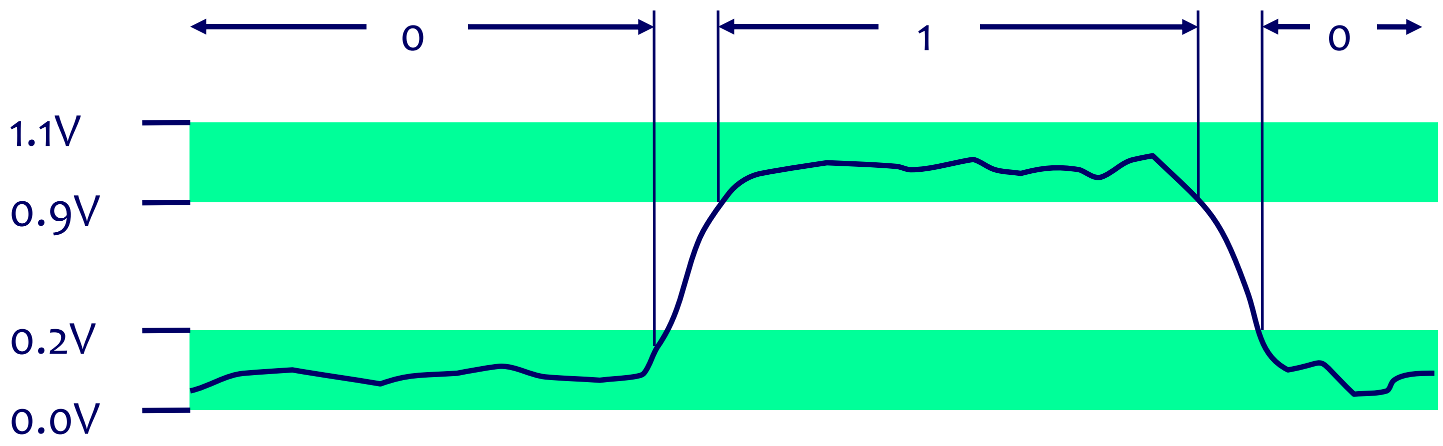
What's wrong with this program?

Today: Bits, Bytes, and Integers

- **Representing information as bits**
- **Bit-level manipulations**
- **Integers**
  - Representation: unsigned and signed
  - Conversion, casting
  - Expanding, truncating
  - Addition, negation, multiplication, shifting
  - Summary
- **Representations in memory, pointers, strings**

Everything is bits

- **Each bit is 0 or 1**
- **By encoding/interpreting sets of bits in various ways**
  - Computers determine what to do (instructions)
  - … and represent and manipulate numbers, sets, strings, etc…
- **Why bits? Electronic Implementation**
  - Easy to store with bistable elements
  - Reliably transmitted on noisy and inaccurate wires



For example, can count in binary

- **Base 2 Number Representation**
  - Represent 15213_{10} as 11101101101101_2
  - Represent 1.20_{10} as $1.0011001100110011[0011]\ldots_2$
  - Represent 1.5213×10^4 as $1.1101101101101_2 \times 2^{13}$

Encoding Byte Values

- **Byte = 8 bits**
  - Binary 00000000_2 to 11111111_2
  - Decimal: 0_{10} to 255_{10}
  - Hexadecimal 00_{16} to FF_{16}
    - Base 16 number representation
    - Use characters ‘0’ to ‘9’ and ‘A’ to ‘F’
    - Write $FA1D37B_{16}$ in C as
      - 0xFA1D37B
      - 0xfa1d37b

Hex	Decimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Preview: Combining bytes…

C Data Type	Typical 32-bit	Typical 64-bit
`char`	1	1
`short`	2	2
`int`	4	4
`long`	4	8
`float`	4	4
`double`	8	8
pointer	4	8

Preview: … to make integers

	W			
	8	**16**	**32**	**64**
UMax	255	65,535	4,294,967,295	18,446,744,073,709,551,615
TMax	127	32,767	2,147,483,647	9,223,372,036,854,775,807
TMin	-128	-32,768	-2,147,483,648	-9,223,372,036,854,775,808

- UMax = $\mathbf{2^{w} - 1}$ where *w* is the number of bits ("word size")
- UMin = 0
- TMax = $\mathbf{2^{w-1} - 1}$
- TMin = $\mathbf{-2^{w-1}}$
  - Asymmetric!
  - Because of zero