# Lecture 1: Introduction to Computer Systems

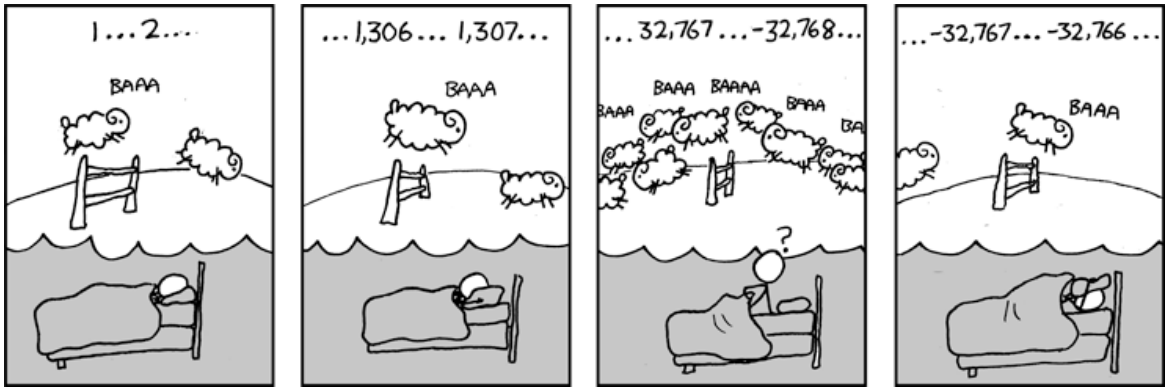
CS 105

January 22, 2020

# Abstraction

# Correctness

- **Example 1: Is $x^2 \geq 0$?**
  - Floats: Yes!




  - Ints:
    - 40000 * 40000 → 1600000000
    - 50000 * 50000 → ??
- **Example 2: Is (x + y) + z = x + (y + z)?**
  - Ints: Yes!
  - Floats:
    - (2^30 + -2^30) + 3.14 → 3.14
    - 2^30 + (-2^30 + 3.14) → ??

# Performance

```
void copyij(int src[2048][2048],
            int dst[2048][2048])
{
  int i,j;
  for (i = 0; i < 2048; i++)
    for (j = 0; j < 2048; j++)
      dst[i][j] = src[i][j];
}
```

4.3ms

```
void copyji(int src[2048][2048],
            int dst[2048][2048])
{
  int i,j;
  for (j = 0; j < 2048; j++)
    for (i = 0; i < 2048; i++)
      dst[i][j] = src[i][j];
}
```

81.8ms

- Hierarchical memory organization
- Performance depends on access patterns
  - Including how step through multi-dimensional array

# Security

```
void admin_stuff(int authenticated){
  if(authenticated){
    // do admin stuff
  }
}

int dontTryThisAtHome(char * user_input, int size) {
  char data[size];
  int ret = memcpy(*user_input, data);
  return ret;
}
```

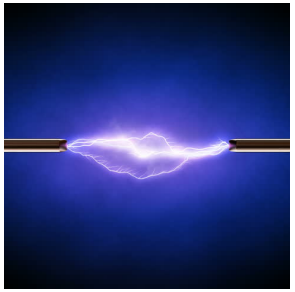
# Bits

- a **bit** is a binary digit that can have two possible values

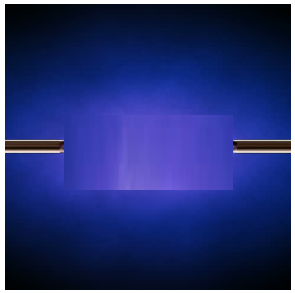
- can be physically represented with a two state device









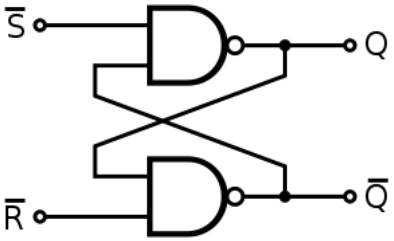



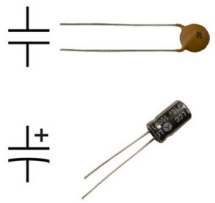
# Bits


A'LA'IH, DO'NEH'LINI, DO'NEH'LINI, A'LA'IH, A'LA'IH, DO'NEH'LINI, DO'NEH'LINI, DO'NEH'LINI, A'LA'IH, A'LA'IH, DO'NEH'LINI, A'LA'IH, DO'NEH'LINI, DO'NEH'LINI, DO'NEH'LINI . . .
FOR ADDED SECURITY, AFTER WE ENCRYPT THE DATA STREAM, WE SEND IT THROUGH OUR NAVAJO CODE TALKER.
... IS HE JUST USING NAVAJO WORDS FOR "ZERO" AND "ONE"?
WHOA, HEY, KEEP YOUR VOICE DOWN!

# Storing bits

- Static random access memory (SRAM): stores each bit of data in a flip-flop, a circuit with two stable states
- Dynamic Memory (DRAM): stores each bit of data in a capacitor, which stores energy in an electric field (or not)
- Magnetic Disk: regions of the platter are magnetized with either N-S polarity or S-N polarity
- Optical Disk: stores bits as tiny indentations (pits) or not (lands) that reflect light differently
- Flash Disk: electrons are stored in one of two gates separated by oxide layers



# Bytes and Memory

- A **byte** is a unit of eight bits
- **Memory** is an array of bytes

- An index into the array is an *address*, *location*, or *pointer*
  - Often expressed in hexadecimal

- We speak of the *value* in memory at an address
  - The value may be a single byte …
  - … or a multi-byte quantity starting at that address

# Binary Numbers

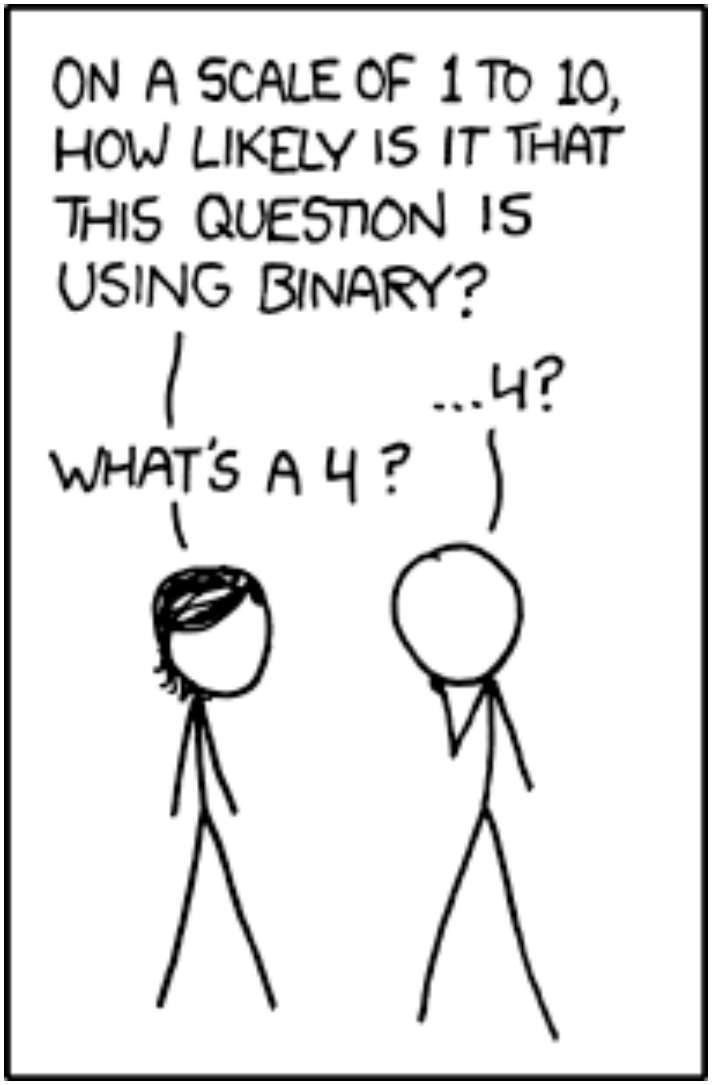
## 4211

$$= 4 \cdot 10^3 + 2 \cdot 10^2 + 1 \cdot 10^1 + 1 \cdot 10^0$$
$$= 4211$$

## 1011

$$= 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$$
$$= 11$$

# Binary Numbers



# Hexidecimal Numbers

- Use digits 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
- Compute numbers base 16

$$1011$$

$$= 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$$
$$= 11$$

$$= 1 \cdot 10^3 + 0 \cdot 10^2 + 1 \cdot 10^1 + 1 \cdot 10^0$$
$$= 1011$$

$$= 1 \cdot 16^3 + 0 \cdot 16^2 + 1 \cdot 16^1 + 1 \cdot 16^0$$
$$= 4113$$

- one byte is two digits in hex

# ASCII characters

| Char | Dec | Oct | Hex | Char | Dec | Oct | Hex | Char | Dec | Oct | Hex |
|---|---|---|---|---|---|---|---|---|---|---|---|
| (sp) | 32 | 0040 | 0x20 | @ | 64 | 0100 | 0x40 | ` | 96 | 0140 | 0x60 |
| ! | 33 | 0041 | 0x21 | A | 65 | 0101 | 0x41 | a | 97 | 0141 | 0x61 |
| " | 34 | 0042 | 0x22 | B | 66 | 0102 | 0x42 | b | 98 | 0142 | 0x62 |
| # | 35 | 0043 | 0x23 | C | 67 | 0103 | 0x43 | c | 99 | 0143 | 0x63 |
| $ | 36 | 0044 | 0x24 | D | 68 | 0104 | 0x44 | d | 100 | 0144 | 0x64 |
| % | 37 | 0045 | 0x25 | E | 69 | 0105 | 0x45 | e | 101 | 0145 | 0x65 |
| & | 38 | 0046 | 0x26 | F | 70 | 0106 | 0x46 | f | 102 | 0146 | 0x66 |
| ' | 39 | 0047 | 0x27 | G | 71 | 0107 | 0x47 | g | 103 | 0147 | 0x67 |
| ( | 40 | 0050 | 0x28 | H | 72 | 0110 | 0x48 | h | 104 | 0150 | 0x68 |
| ) | 41 | 0051 | 0x29 | I | 73 | 0111 | 0x49 | i | 105 | 0151 | 0x69 |
| * | 42 | 0052 | 0x2a | J | 74 | 0112 | 0x4a | j | 106 | 0152 | 0x6a |
| + | 43 | 0053 | 0x2b | K | 75 | 0113 | 0x4b | k | 107 | 0153 | 0x6b |
| , | 44 | 0054 | 0x2c | L | 76 | 0114 | 0x4c | l | 108 | 0154 | 0x6c |
| - | 45 | 0055 | 0x2d | M | 77 | 0115 | 0x4d | m | 109 | 0155 | 0x6d |
| . | 46 | 0056 | 0x2e | N | 78 | 0116 | 0x4e | n | 110 | 0156 | 0x6e |
| / | 47 | 0057 | 0x2f | O | 79 | 0117 | 0x4f | o | 111 | 0157 | 0x6f |
| 0 | 48 | 0060 | 0x30 | P | 80 | 0120 | 0x50 | p | 112 | 0160 | 0x70 |
| 1 | 49 | 0061 | 0x31 | Q | 81 | 0121 | 0x51 | q | 113 | 0161 | 0x71 |
| 2 | 50 | 0062 | 0x32 | R | 82 | 0122 | 0x52 | r | 114 | 0162 | 0x72 |
| 3 | 51 | 0063 | 0x33 | S | 83 | 0123 | 0x53 | s | 115 | 0163 | 0x73 |
| 4 | 52 | 0064 | 0x34 | T | 84 | 0124 | 0x54 | t | 116 | 0164 | 0x74 |
| 5 | 53 | 0065 | 0x35 | U | 85 | 0125 | 0x55 | u | 117 | 0165 | 0x75 |
| 6 | 54 | 0066 | 0x36 | V | 86 | 0126 | 0x56 | v | 118 | 0166 | 0x76 |
| 7 | 55 | 0067 | 0x37 | W | 87 | 0127 | 0x57 | w | 119 | 0167 | 0x77 |
| 8 | 56 | 0070 | 0x38 | X | 88 | 0130 | 0x58 | x | 120 | 0170 | 0x78 |
| 9 | 57 | 0071 | 0x39 | Y | 89 | 0131 | 0x59 | y | 121 | 0171 | 0x79 |
| : | 58 | 0072 | 0x3a | Z | 90 | 0132 | 0x5a | z | 122 | 0172 | 0x7a |
| ; | 59 | 0073 | 0x3b | [ | 91 | 0133 | 0x5b | { | 123 | 0173 | 0x7b |
| < | 60 | 0074 | 0x3c | \ | 92 | 0134 | 0x5c | \| | 124 | 0174 | 0x7c |
| = | 61 | 0075 | 0x3d | ] | 93 | 0135 | 0x5d | } | 125 | 0175 | 0x7d |
| > | 62 | 0076 | 0x3e | ^ | 94 | 0136 | 0x5e | ~ | 126 | 0176 | 0x7e |
| ? | 63 | 0077 | 0x3f | _ | 95 | 0137 | 0x5f | | | | |

# x86 instructions

Machine code bytes

```
B8 22 11 00 FF
01 CA
31 F6
53
8B 5C 24 04
8D 34 48
39 C3
72 EB
C3
```

Assembly

```
foo:
    movl $0xFF001122, %eax
    addl %ecx, %edx
    xorl %es1, %es1
    pushl %ebx
    movl 4(%esp), %ebx
    leal (%eax,%ecx,2), %exi
    cmpl %eax, %ebx
    jnae foo
    retl
```

# Bits and Bytes Require Interpretation

00000000 00110101 00110000 00110001 (or 0x00353031) might be interpreted as

- The integer $3{,}485{,}745_{10}$
- A floating point number close to $4.884569 \times 10^{-39}$
- The string “105”
- A portion of an image or video
- An address in memory

# Information is Bits + Context

# C

```
#include<stdio.h>

int main(int argc, char** argv){
  printf("Hello world!\n");
  return 0;
}
```

# Example Data Representations

| C Data Type | x86-64 |
|---|---|
| **char** | 1 |
| **short** | 2 |
| **int** | 4 |
| **long** | 8 |
| **float** | 4 |
| **double** | 8 |
| **pointer** | 8 |

# Memory Access in C

```
int x;    // an integer
int *p;   // a pointer to an integer

// normal initialization:
x = 0;

// silly, but illustrative:
p = &x;   // & means "address of"
*p = 0;   // * means "memory at address"
```

- & and * are inverses of one another
- prefix vs infix operators
- `x` occupies 4 bytes in memory; `p` occupies 8

# Arrays

- Contiguous block of memory
- Pointer to start, then indexed by element size
  - Indices start at zero

- **`ary[k]`** is the same as **`*(ary+k)`**
  - Location of **`ary+k`** depends on the type of array elements

# Arrays and Pointers Combined

```
int *p[47];
```

- Array of pointers … or … pointer to an array??

- It's an array of **`47`** pointers
  - **`p[3]`** is the fourth pointer in the array **`p`**
  - **`p[3]`** is the base of an array
  - **`p[3][6]`** is the integer at position **`6`** in the array **`p[3]`**

# What is printed?

```
int a[100];
int *p[47];

p[3] = a+12;
for (int i = 0; i < 100; i++){
    a[i] = i;
}

printf("%d\n", p[3][4]);
```

# Structs

- Heterogeneous records, like objects
- Typical linked list declaration:

```
typedef struct cell {
  int value;
  struct cell *next;
} cell_t;
```

- Usage:

```
cell_t c;
c.value = 42;
c.next = NULL;
```

How many bytes are allocated for `c`? for `p`?

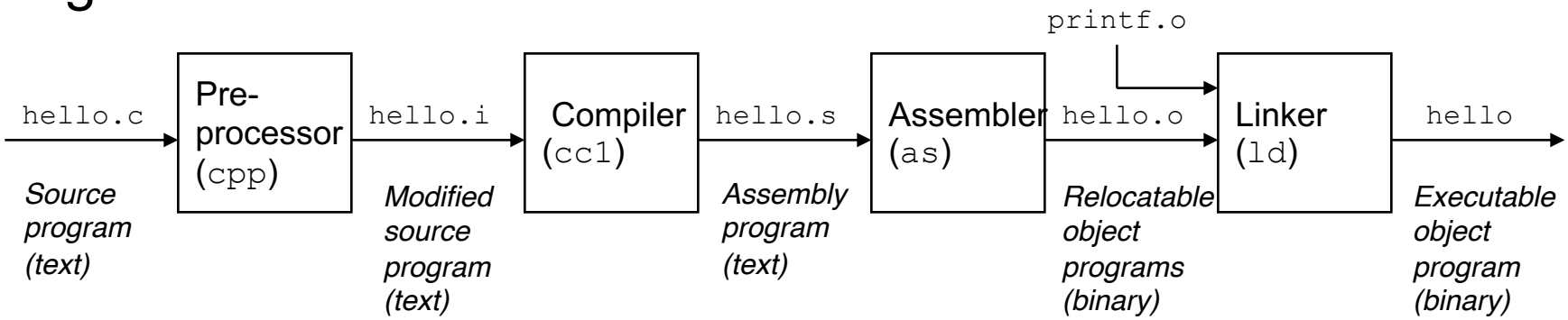
- Usage with pointers:

```
cell_t *p;
p->value = 42;
p->next = NULL;
```

`p->next` is an bbreviation for `(*p).next`

# Compilation

- gcc –o hello hello.c




```
#include<stdio.h>

int main(int argc,
         char ** argv){

  printf("Hello
          world!\n");
  return 0;
}
```

```
…
int printf(const char *
           restrict,
           ...)
__attribute__((__format_
_ (__printf__, 1, 2)));
…
int main(int argc,
         char ** argv){

  printf("Hello
          world!\n");
  return 0;
}
```

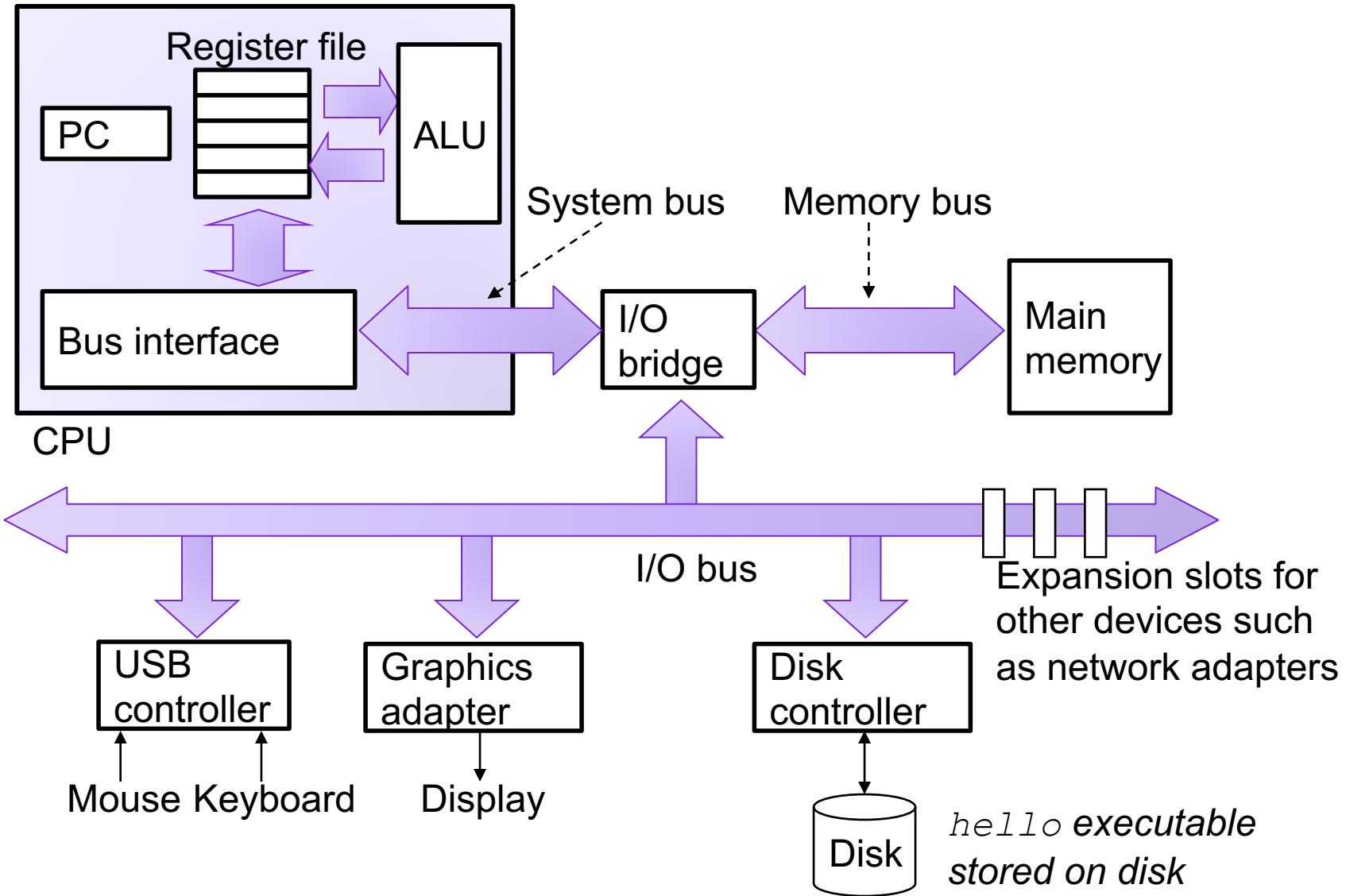
```
pushq   %rbp
movq    %rsp, %rbp
subq  $32, %rsp
leaq  L_.str(%rip), %rax
movl  $0, -4(%rbp)
movl  %edi, -8(%rbp)
movq  %rsi, -16(%rbp)
movq  %rax, %rdi
movb  $0, %al
callq _printf
xorl  %ecx, %ecx
movl  %eax, -20(%rbp)
movl  %ecx, %eax
addq  $32, %rsp
popq  %rbp
retq
```

```
55
48 89 e5
48 83 ec 20
48 8d 05 25 00 00 00
c7 45 fc 00 00 00 00
89 7d f8
48 89 75 f0
48 89 c7
b0 00
e8 00 00 00 00
31 c9
89 45 ec
89 c8
48 83 c4 20
5d
c3
```

# Running a Program

- ./hello

# A Computer System



# LOGISTICS

# Course staff

Prof. Eleanor Birrell
Edmunds 221

Research in security and privacy
OH: T 1:30-3:30pm, W 7-9pm

Jenna Brandt

Joe Brennan

Gabriel da Motta

Adam Lininger-White

Douglas Webster

# The Course in a Nutshell

- Textbook
  - Bryant and O'Halloran, *Computer Systems: A Programmer's Perspective*, **third edition,** Pearson, 2016 (Recommended)
- Classes
  - Monday and Wednesday, 1:15-2:30 or 2:45-4pm in Edmunds 101
- Labs
  - Mondays 7-8:15 in Edmunds 229/219
  - **Start Monday!** Be sure to have an account and password

| Mentor Session Schedule (Edmunds 227) | | | | | | |
|---|---|---|---|---|---|---|
| Monday | Tuesday | Wednesday | Thursday | Friday | Saturday | Sunday |
| LAB | 7-9pm | 7-9pm | 7-9pm | 2-4pm | 2-4pm | 2-4pm |

# Grading

- Assignments
  - Introduced during labs, Due Fridays at 5pm
  - Tremendous fun, work in pairs
  - 45% of the grade
  - Seven late days
- Midterm exam
  - March 11
  - 20% of the grade each
- Final exam
  - Thursday, May 7 or Tuesday, May 12 or Friday, May 15 (2-5pm)
  - 30% of the grade
- Participation
  - 5% of the grade

# Course website

https://www.cs.pomona.edu/classes/cs105

- All information is on the course website
- All course materials get posted on the course website
- Links from the course page:
  - Piazza, for questions and discussion
  - Lab assistants and mentors, schedule
  - Submission site
  - Grading site