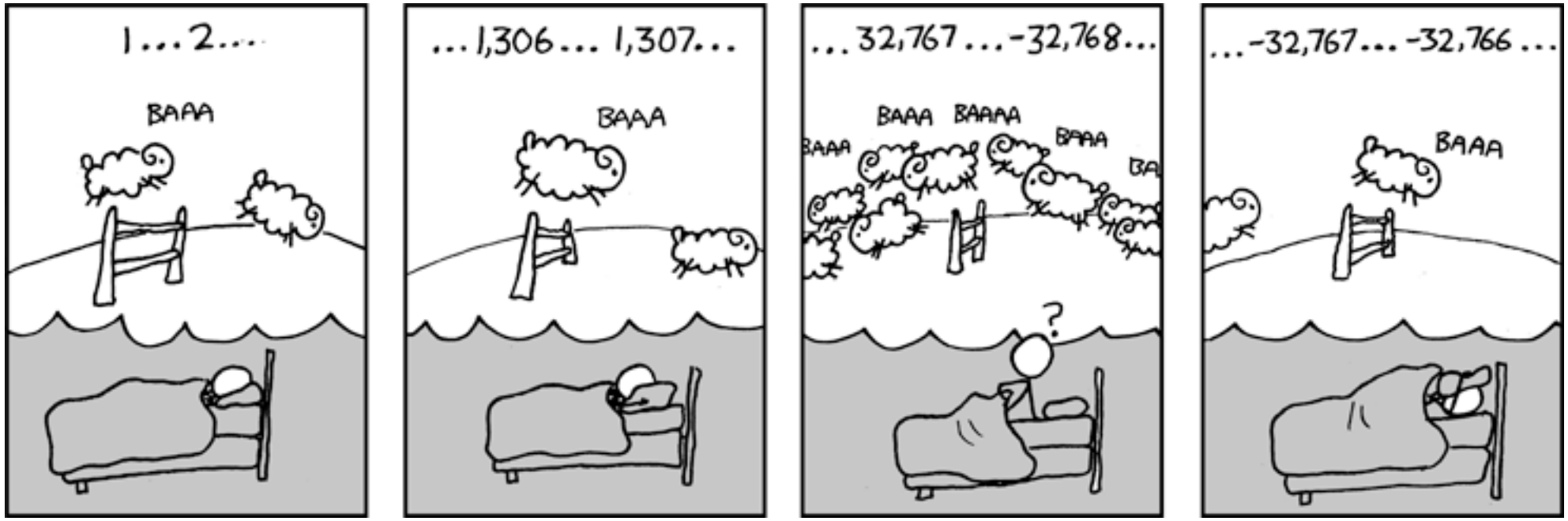
NEIL KLINGENSMITH

CS 264: INTRO TO SYSTEMS

https://neilklingensmith.com/teaching/loyola/cs264-s2020/

WHY DO YOU HAVE TO TAKE THIS STUPID CLASS

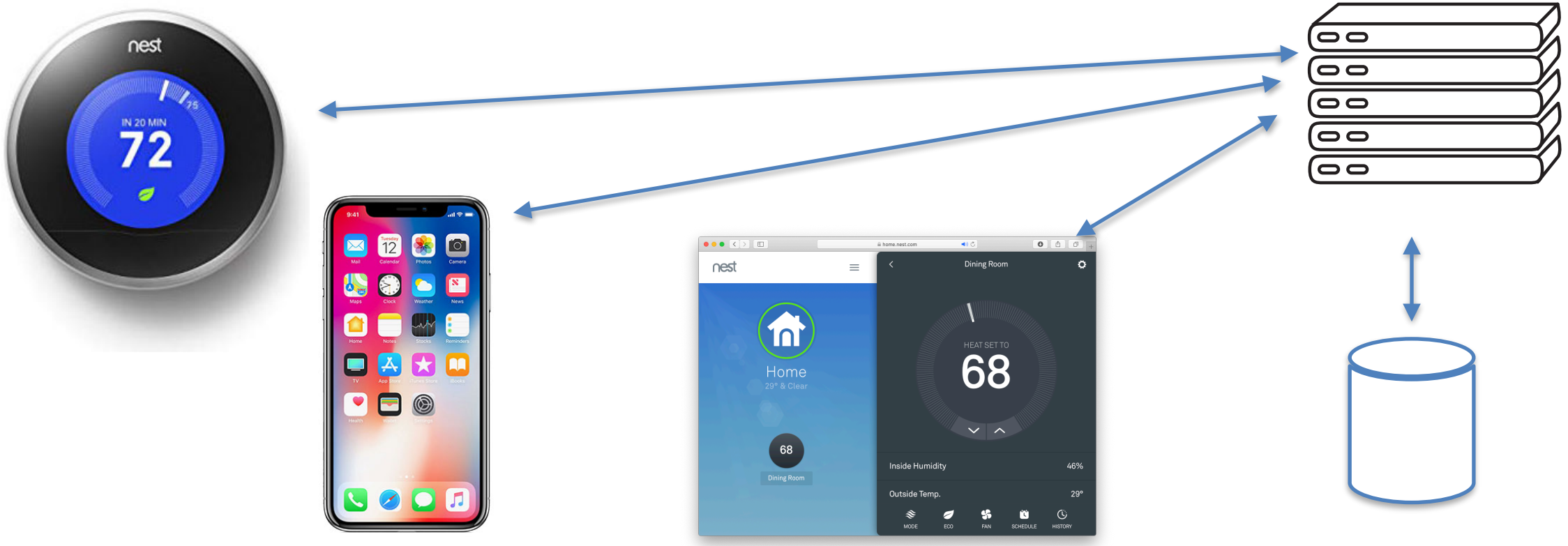
- **Abstraction is good, but don't forget reality:**
  - **Most CS classes emphasize abstraction. Not this one.**



WHY DO YOU HAVE TO TAKE THIS STUPID CLASS

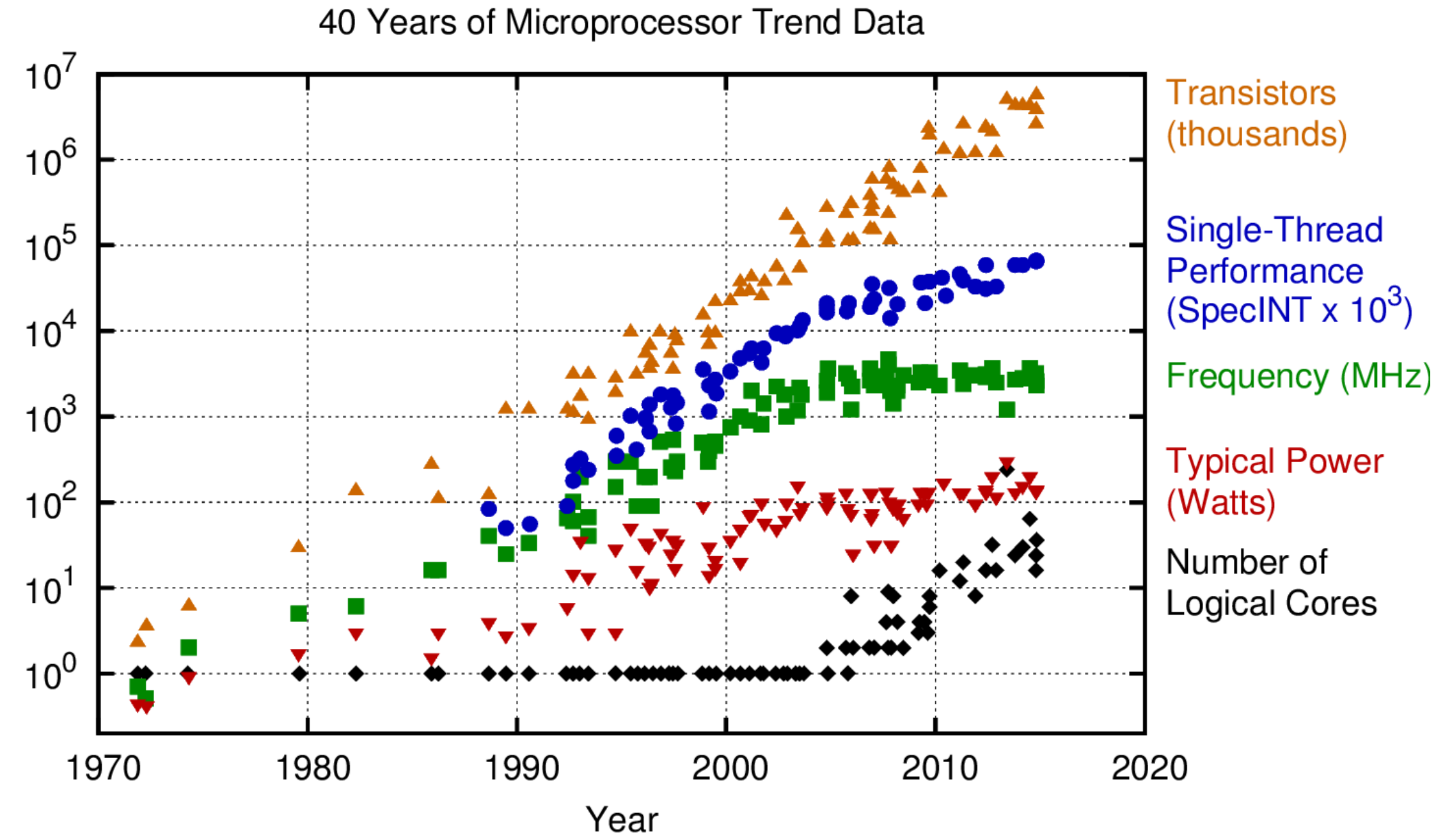
- **People don’t just write programs in one language for one platform anymore. Real projects have lots of parts.**

WHY DO YOU HAVE TO TAKE THIS STUPID CLASS



WHY DO YOU HAVE TO TAKE THIS STUPID CLASS

- People don't just write programs in one language for one platform anymore. Real projects have lots of parts.
- Computers are changing: parallelism is much more important today than it was in the 90s.
- Stuff you learn here will be used in security, OS, compilers, architecture, IoT, etc.


40 Years of Microprocessor Trend Data
10^7
10^6
10^5
10^4
10^3
10^2
10^1
10^0
1970
1980
1990
2000
2010
2020
Year
Transistors (thousands)
Single-Thread Performance (SpecINT x 10^3)
Frequency (MHz)
Typical Power (Watts)
Number of Logical Cores

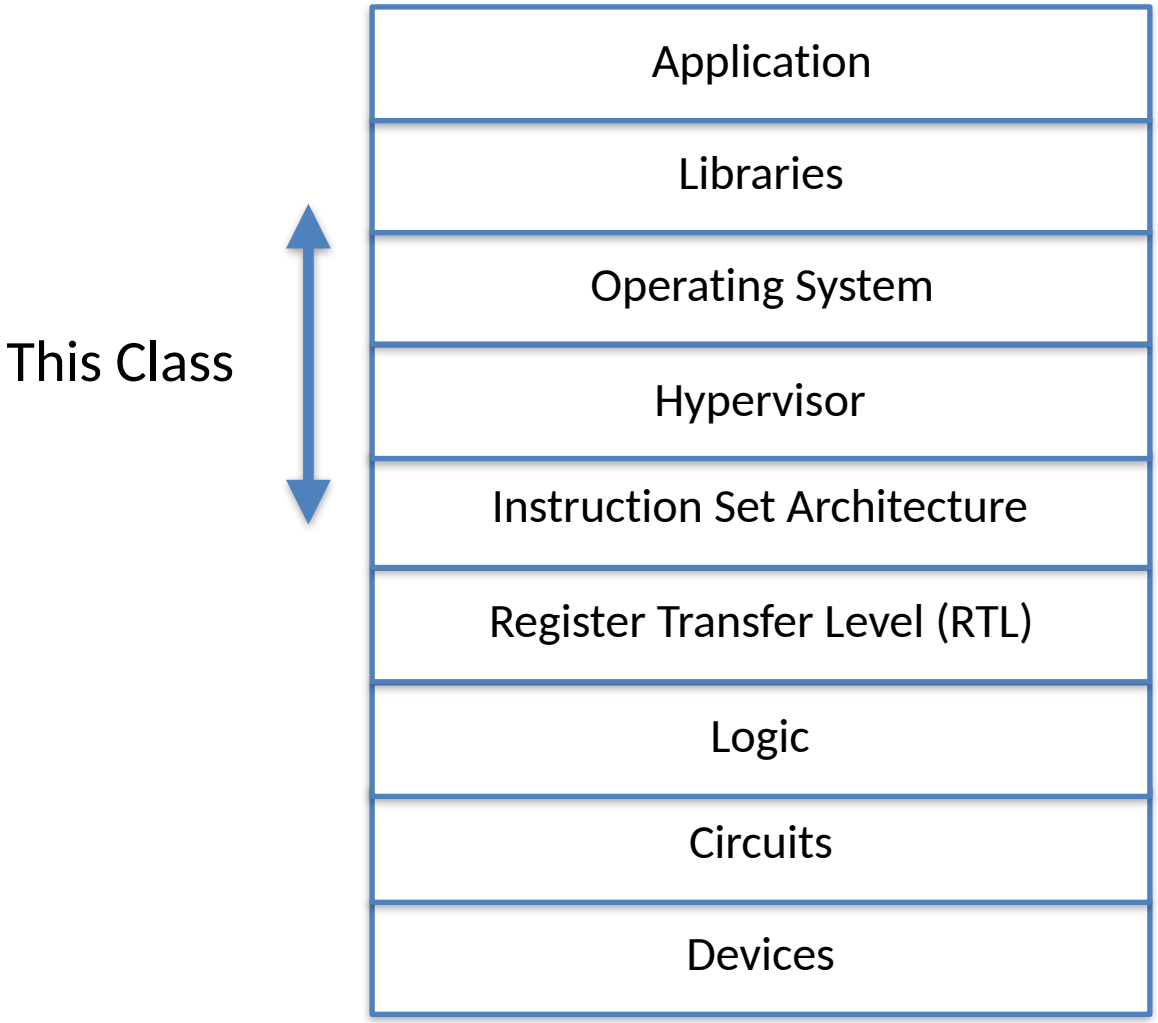
MY GOALS FOR YOU

1. Have a gut feeling for what memory is.
2. Write a few bare metal programs that aren't constrained by an OS.
3. Understand how the computer runs your program.

COURSE OUTLINE

- 1st Five Weeks: Assembly Language Programming
- 2nd Five Weeks: C Programming
- Last Five Weeks: Final Project

ABSTRACTIONS IN A COMPUTER



LABS

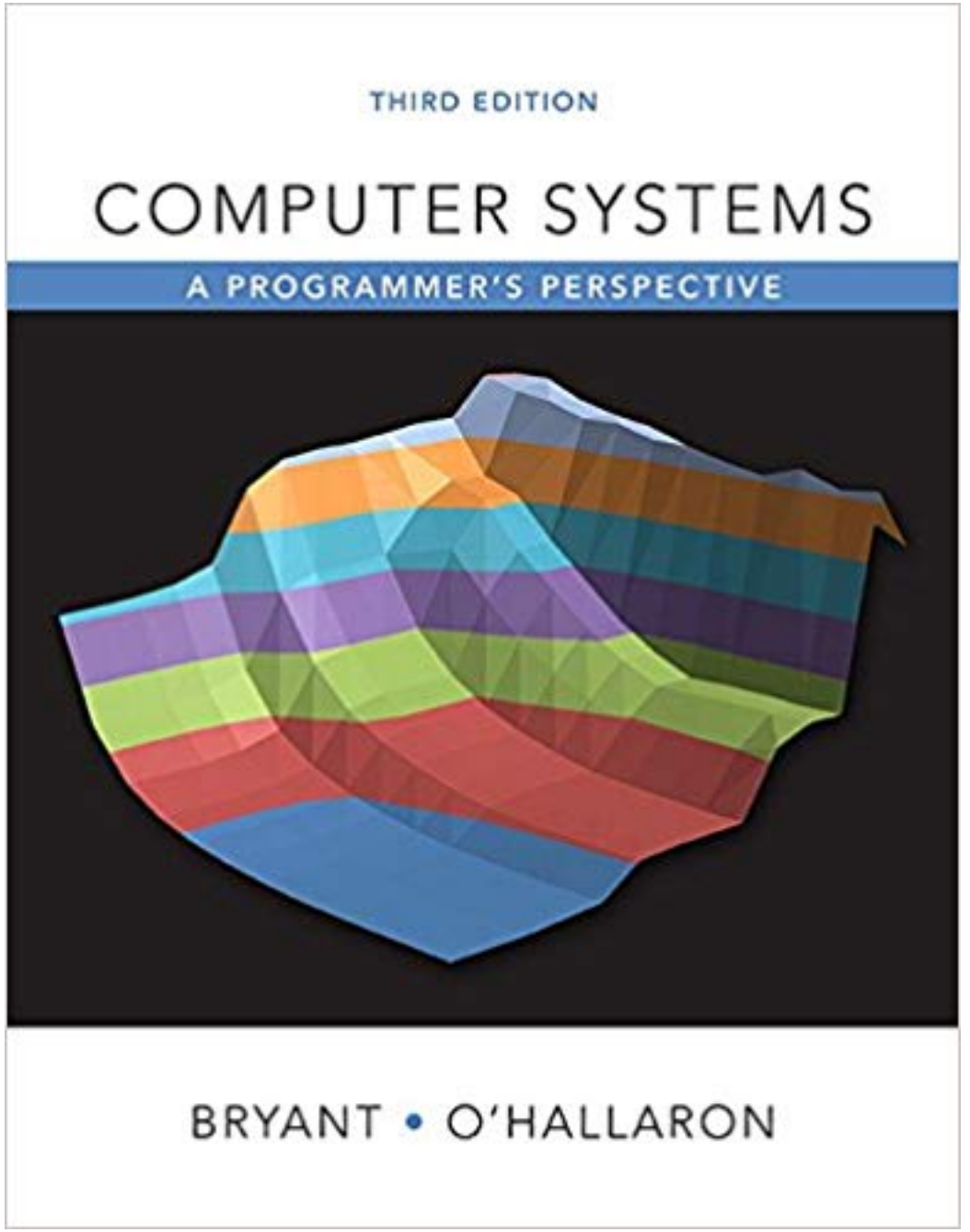
- Lab is a time when you can do your homework (with help from Neil and others).
- Lab sessions will be held Thursdays from 4-6 PM in Doyle 314.

REQUIRED MATERIALS

- **Book: Computer Systems: A Programmer's Perspective**
- **You need a laptop with at least 8 GB RAM to run VMWare.**
- **Download VMWare (or VirtualBox), link on course website.**

THE BOOK

- **Not Required.**
- **Buy it if you like books.**
- **I have a PDF version.**



GRADING

- **No quizzes or exams. Your whole grade is based on homework and final project.**
- **No partial credit for code that doesn't compile.**
- **Start homework on Tuesday/ Wednesday so you can get help on Thursday in lab if you get stuck.**

Category	Weight
Homework	30%
Participation	10%
Progress	10%
Final Project	40%

DOING YOUR OWN WORK

- **Do not share code.**
- **Do not copy code from the internet.**
- **You might want to save them for the end of the semester.**

SLOP DAYS

- **Each students gets five slop days to use during the semester.**
- **Can’t use more than two slop days on one assignment.**

CODING STYLE

1. Every function should have a header explaining what it does. For example:

```
/*
 * memcpy()
 *
 * Copies count bytes from src to dest. Returns
 * the number of bytes copied or a negative number
 * in case of error.
 */
int memcpy(void *dest, void *src, unsigned int count) {
```

CODING STYLE

1. Every function should have a header explaining what it does.

2. Functions written in assembly language also need a stack frame diagram. For example:

```
; memcpy
;  -------------
; | count       | 2 bytes
;  -------------
; | src         | 2 bytes
;  -------------
; | dest        | 2 bytes
;  -------------
; | Ret Addr    | 2 bytes
;  -------------
; | Caller's BP | 2 bytes
;  -------------
; Copies count bytes from src to dest. Returns...
memcpy:
```

CODING STYLE

1. **Every function should have a header explaining what it does.**
2. **Functions written in assembly language also need a stack frame diagram. For example:**
3. **Indent properly.**

```
for(k = 0; k < PAGE_SIZE; k++){
    if(page->next != NULL){
page = page->next;
    }
}
```

NOOOOOO!!!!!!

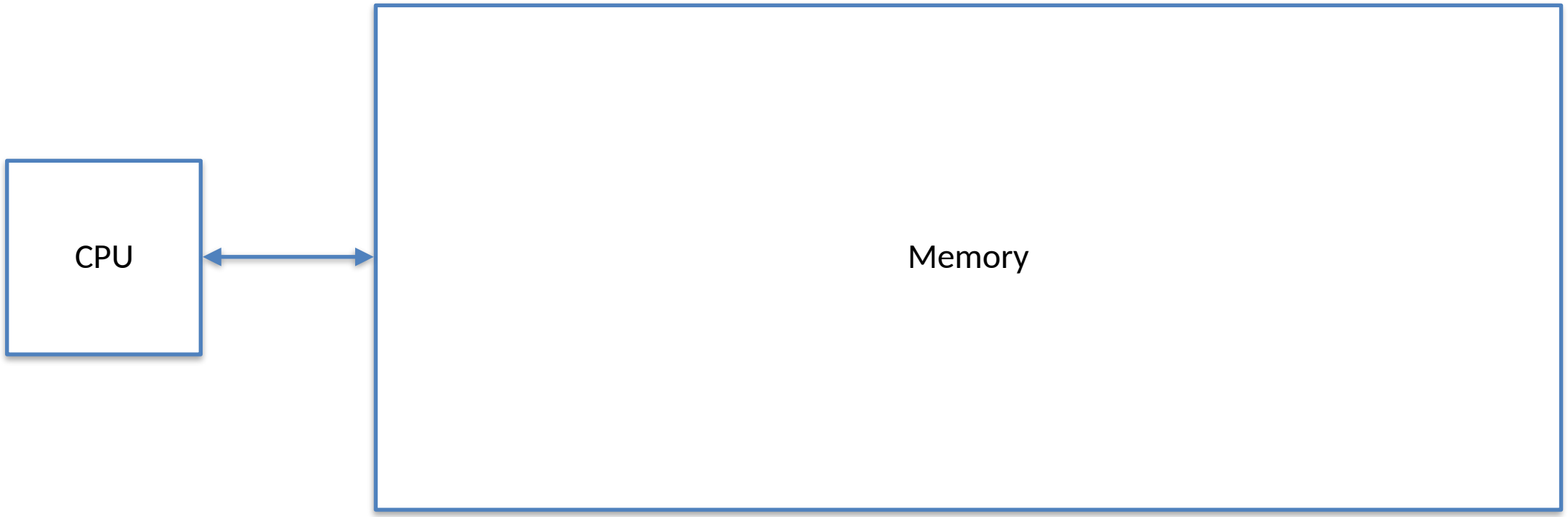
CODING STYLE

1. **Every function should have a header explaining what it does.**
2. **Functions written in assembly language also need a stack frame diagram. For example:**
3. **Indent properly.**
4. **Comment your code**

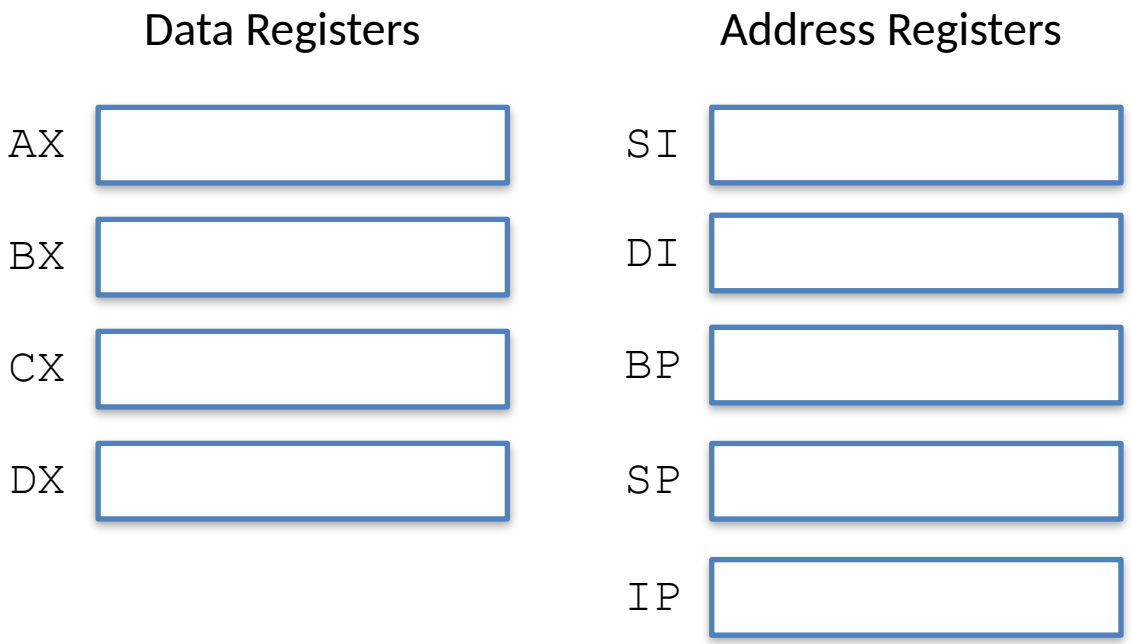
```
for(k = 0; k < PAGE_SIZE; k++){ // Loop thru each page...
    if(page->next != NULL){ // Don't dereference NULL ptr.
        page = page->next; // Get next element of list
    }
}
```

INTRO…

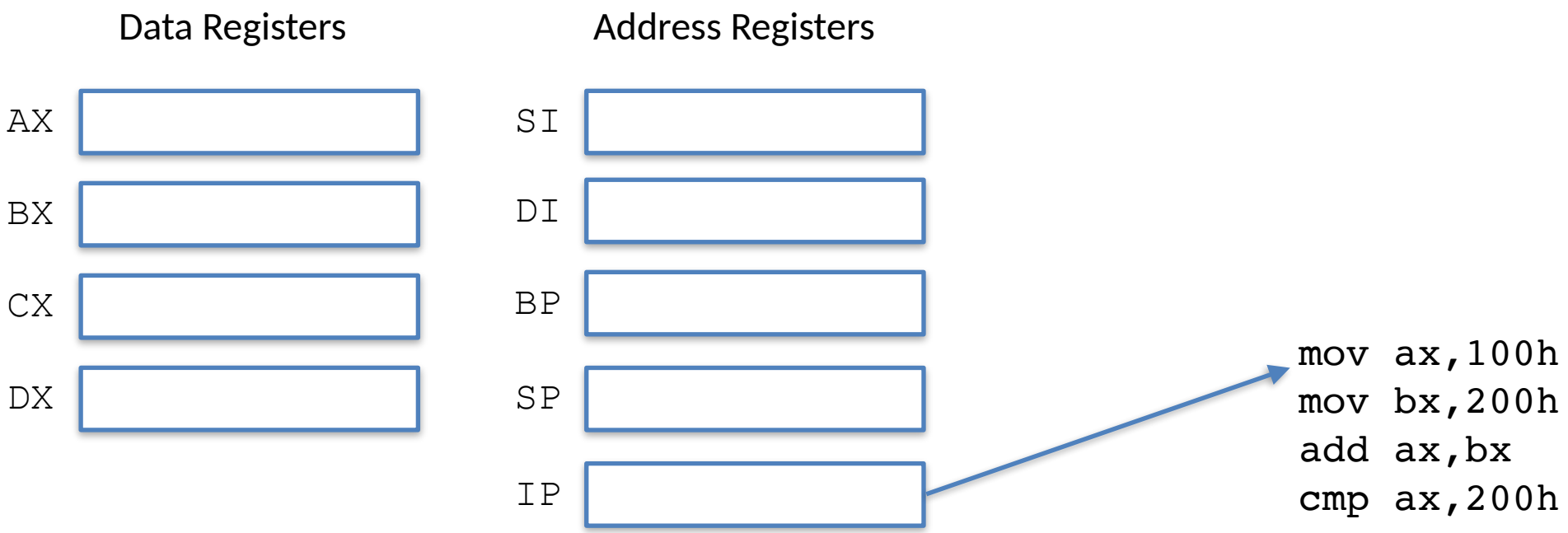
PROGRAMMER'S MODEL OF X86



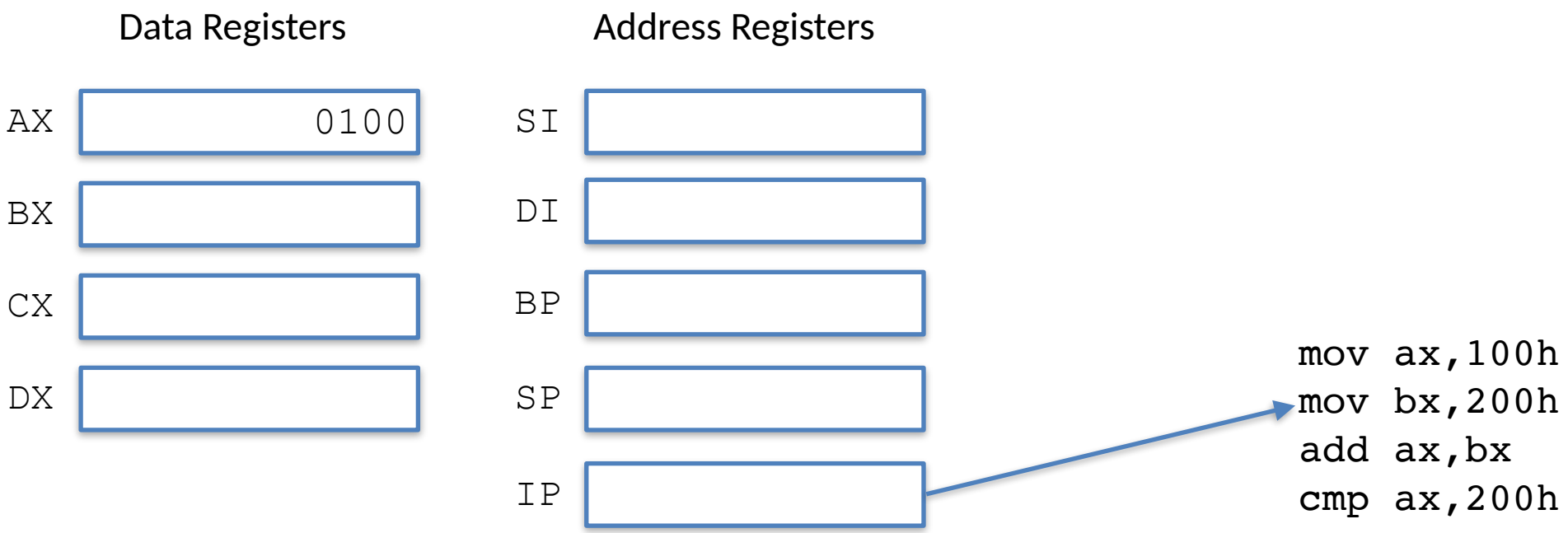
PROGRAMMER'S MODEL OF X86: INSIDE THE CPU



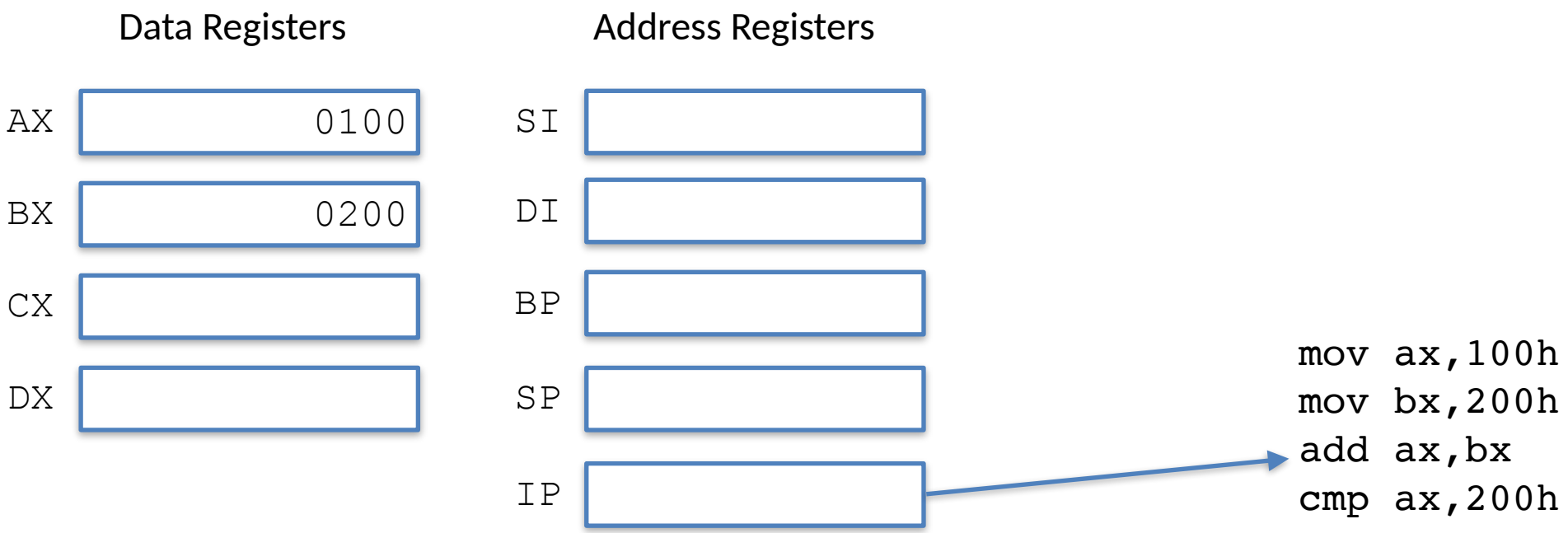
PROGRAMMER'S MODEL OF X86: INSIDE THE CPU



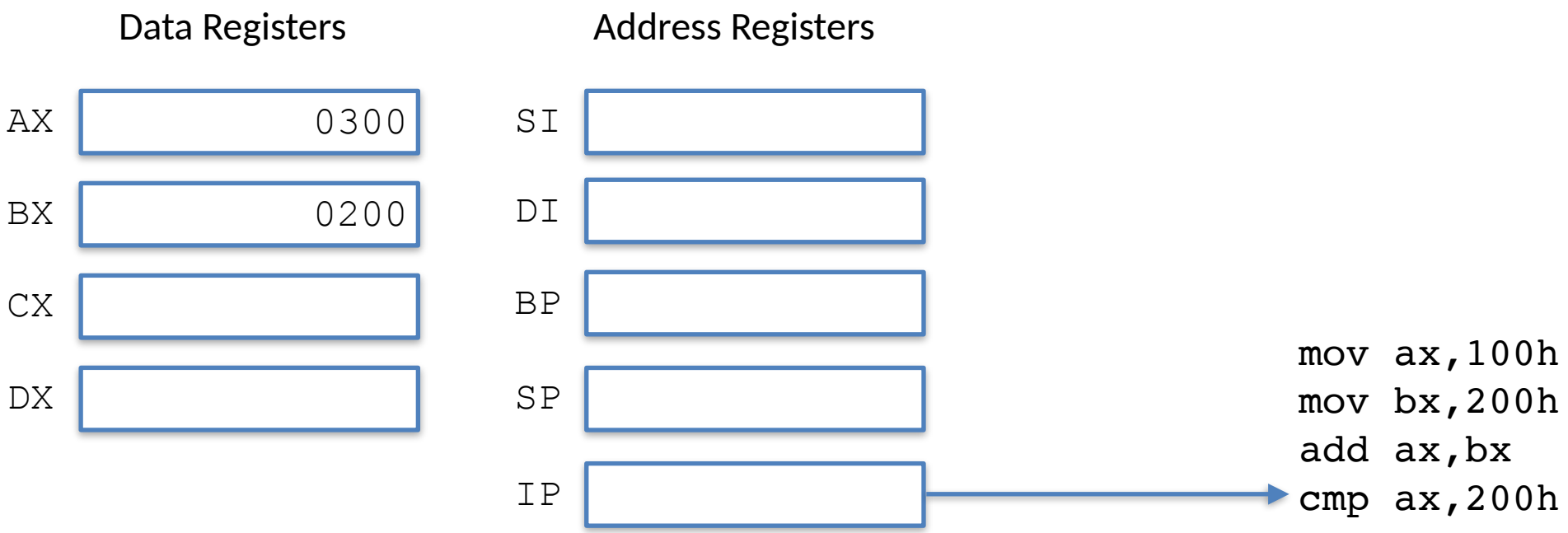
PROGRAMMER'S MODEL OF X86: INSIDE THE CPU



PROGRAMMER'S MODEL OF X86: INSIDE THE CPU



PROGRAMMER'S MODEL OF X86: INSIDE THE CPU



THE ONLY THING A COMPUTER KNOWS HOW TO DO IS EXECUTE INSTRUCTIONS.

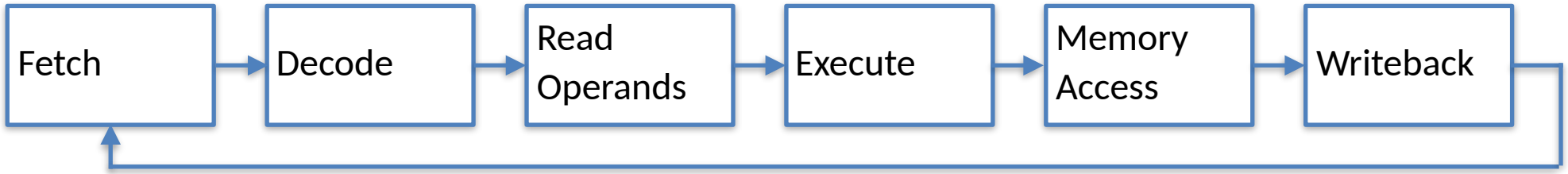
```
if( a < 5 ) {
  b += a;
  a++;
}
```

```
cmp ax,5
jge .not_less_than

add bx,ax

inc ax

.not_less_than:
   ...
```

KINDS OF INSTRUCTIONS

- Arithmetic
  - Add, subtract, multiply, divide
- Logic
  - AND, OR, NOT, XOR
- Shifts
  - Left shift, right shift, rotate, etc.
- Control
  - Branch/Jump
  - Procedure calls
- Memory Accesses
  - Load/store

THE ONLY THING A COMPUTER KNOWS HOW TO DO IS EXECUTE INSTRUCTIONS.



HOMEWORK

- **Download and install emu8086.**
  - **You need Windows: use VMWare if you have a mac.**
  - **If you need help, come to lab on Thursday.**
- **Sign up for GitHub if you don't have an account.**
- **Send me you GitHub username.** `neil@cs.luc.edu`