

NEIL KLINGENSMITH

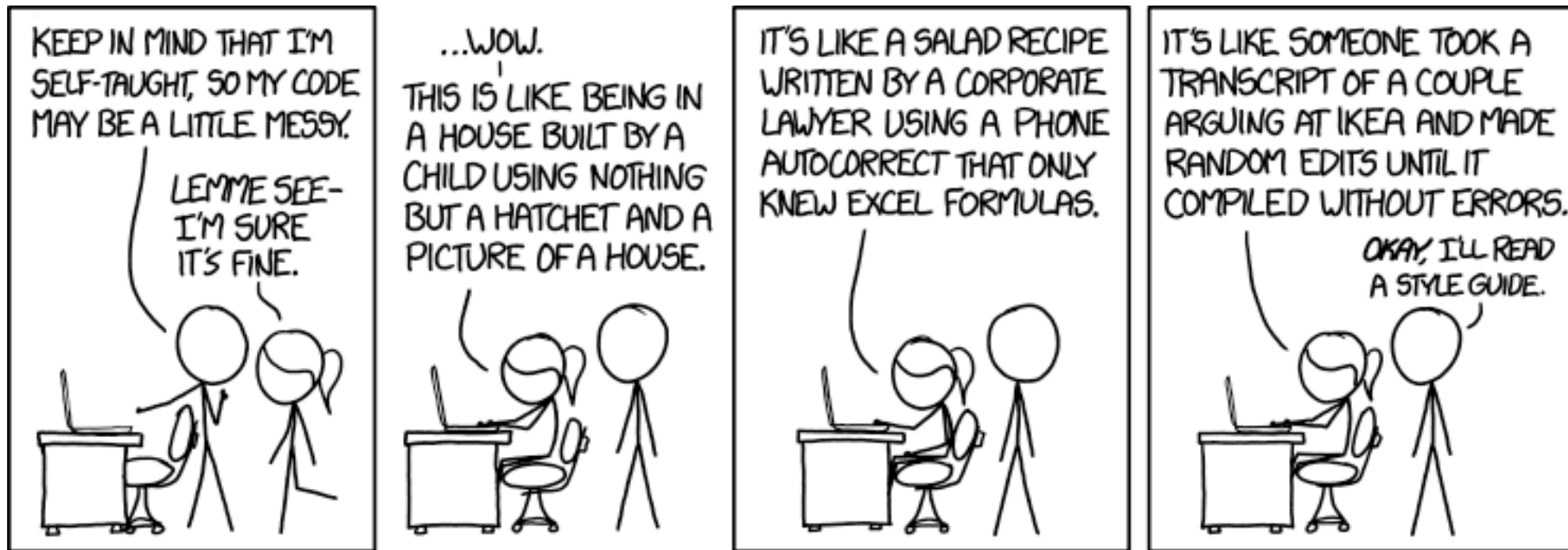
CS 264: INTRO TO SYSTEMS

<https://comp264.org>



WHY DO YOU HAVE TO TAKE THIS STUPID CLASS

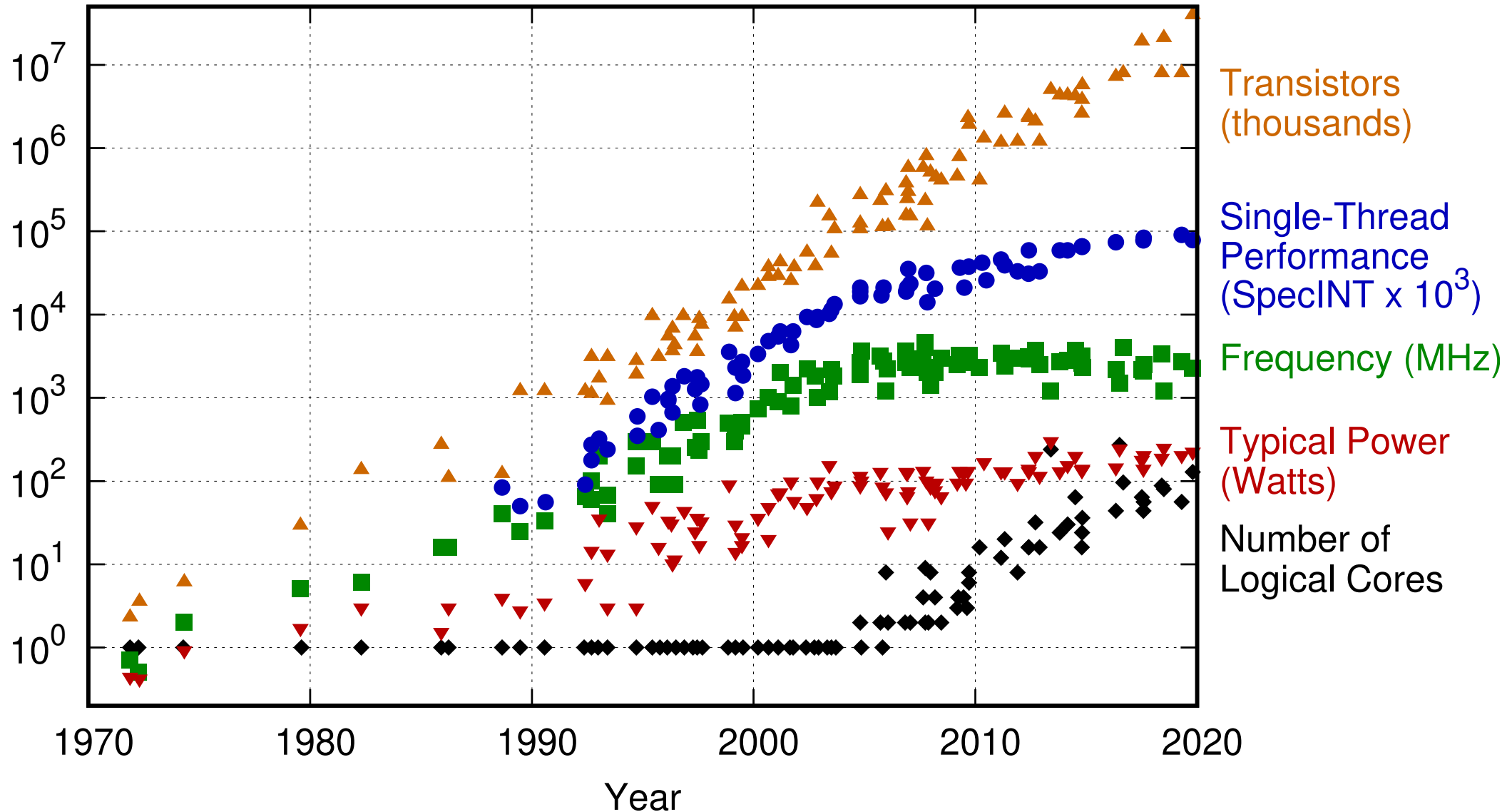
- **Abstraction is good, but don't forget reality:**
 - **Most CS classes emphasize abstraction. Not this one.**



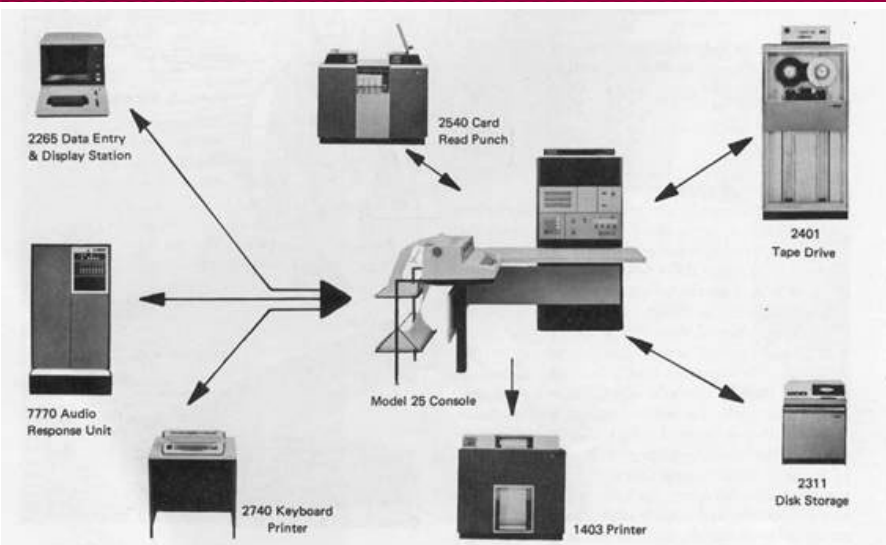
WHY DO YOU HAVE TO TAKE THIS STUPID CLASS

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

48 Years of Microprocessor Trend Data



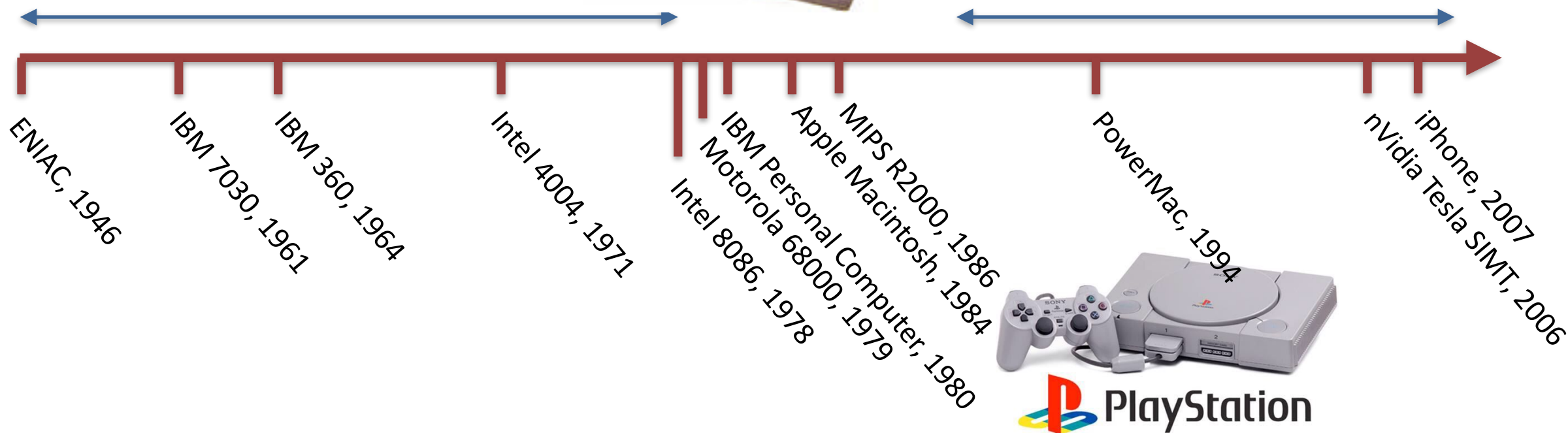
Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2019 by K. Rupp



Many/most programs
written in assembly language



Most programs written in
higher level languages



WHAT IS THIS CLASS GOING TO BE LIKE?

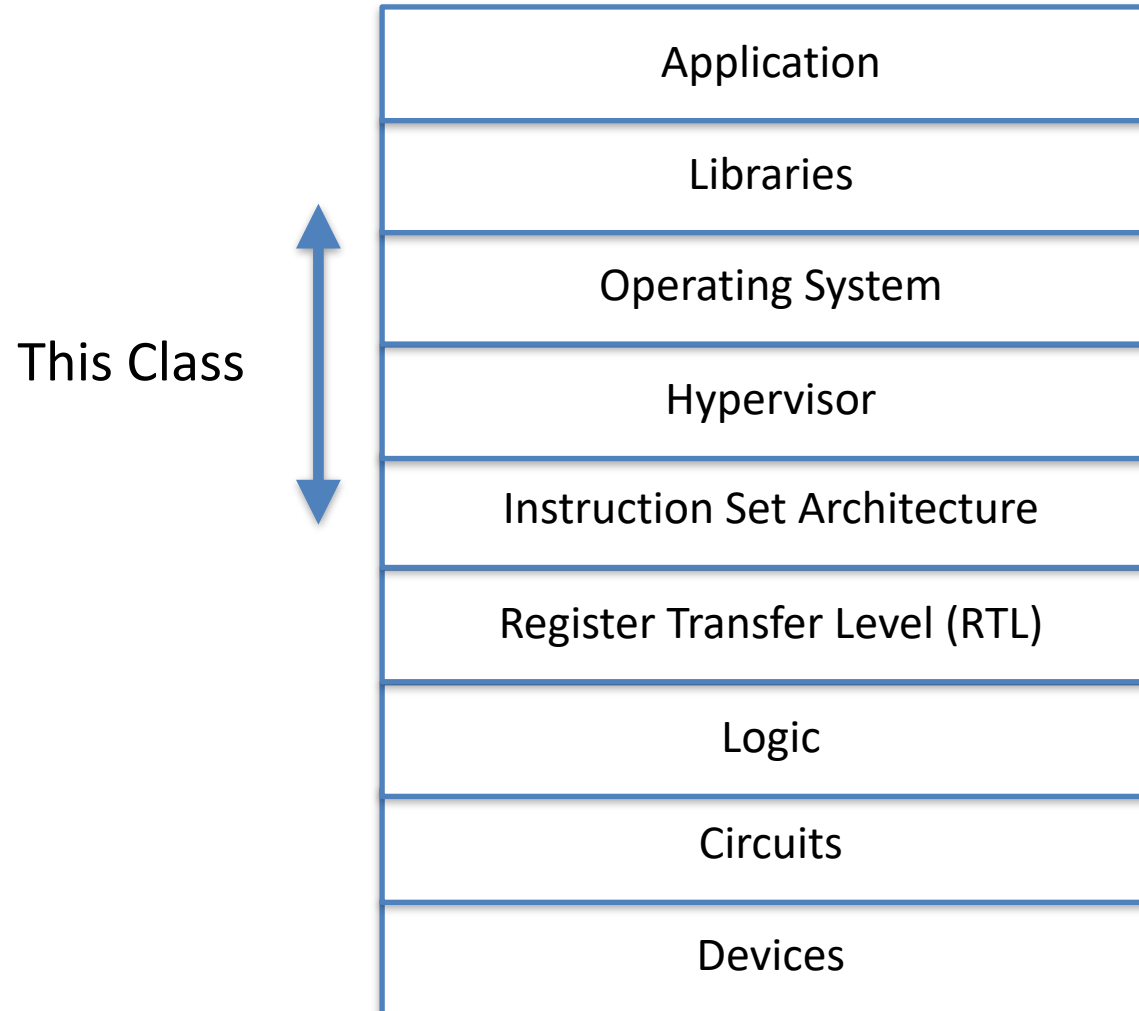
MY GOALS FOR YOU

- 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

- **Weeks 1-3: Hardware Basics**
- **Weeks 4-9: Assembly Language Programming**
- **Weeks 10-15: C Programming**

ABSTRACTIONS IN A COMPUTER



COURSE WEBSITE: `http://comp264.org`

- **Video lectures on the course schedule. Watch them on your own and take notes.**
- **Weekly homework and quizzes on the course schedule.**

LIVE ZOOM SESSIONS

- **Clarification of questions and activities**
- **Questions**
- **You may record, but I won't**

LABS

- **Lab is a time when you can do your homework (with help from TA Jack).**
- **Lab sessions will be held Thursdays from 4:30-6PM online. (Same Zoom link as class)**

GRADING

- **No partial credit for code that doesn't compile.**
- **No extended due dates.**
- **Follow stack overflow forum rules:**
 - **Need to show that you've attempted the assignment before asking for help.**
 - **Need a specific question.**

Category	Weight
Homework	55%
Participation	10%
Progress	10%
Quizzes & Checkins	25%

DOING YOUR OWN WORK

- **You're allowed to use Internet as a source and modify code you find.**
- **Don't copy-paste code verbatim.**
- **We may ask you questions about how your code works.**

SLOP DAYS

- **Each students gets five slop days to use during the semester.**
- **Slop day allows you to turn in homework up to 24 hrs late.**
- **Can't use more than two slop days on one assignment.**
- **Tell Kyle that you're going to use slop days before the due date.**

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; NOOOOOOO!!!!!!!  
    }  
}
```



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

Data Registers

AX

BX

CX

DX

Address Registers

SI

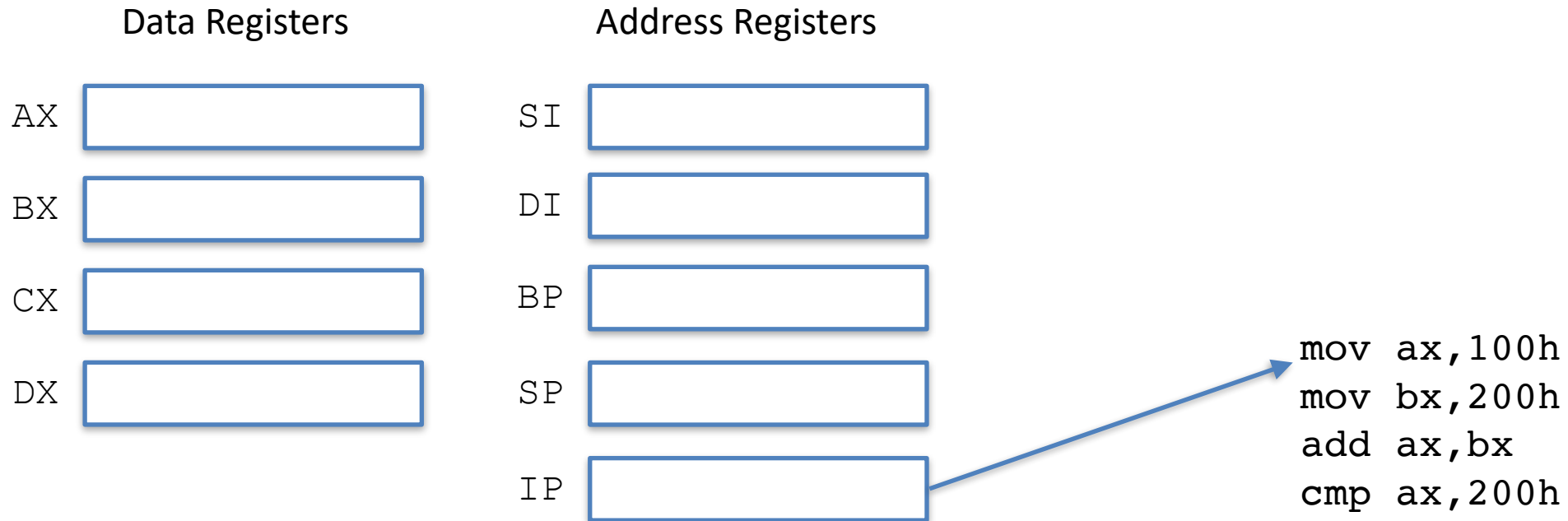
DI

BP

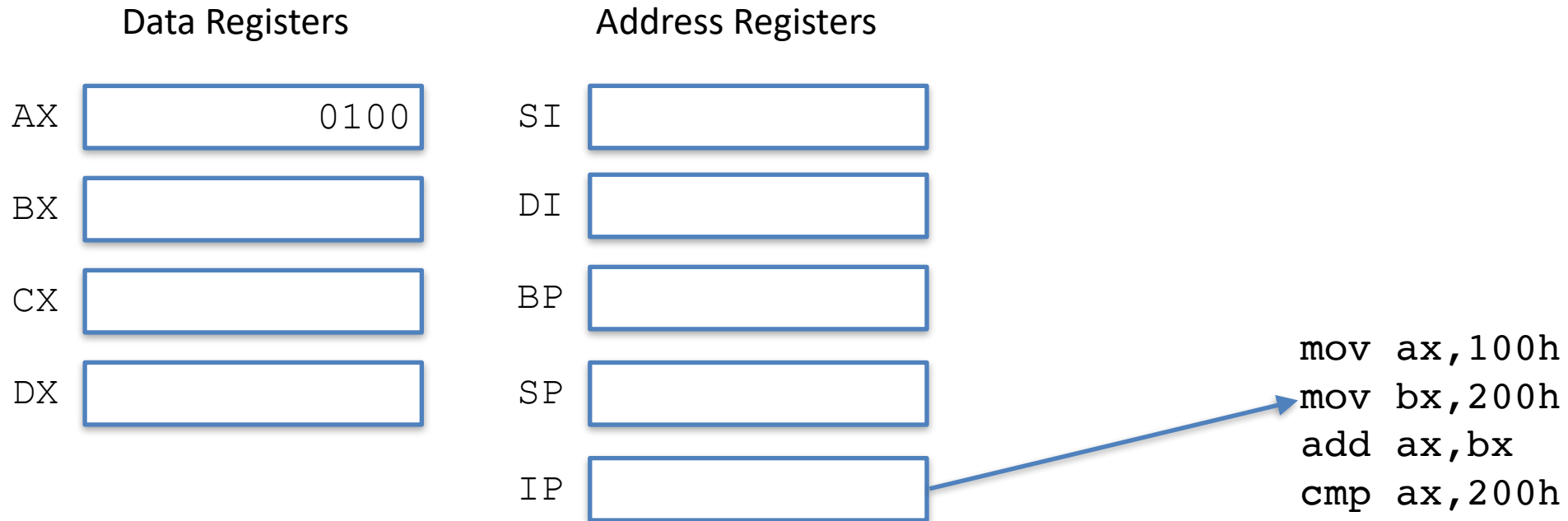
SP

IP

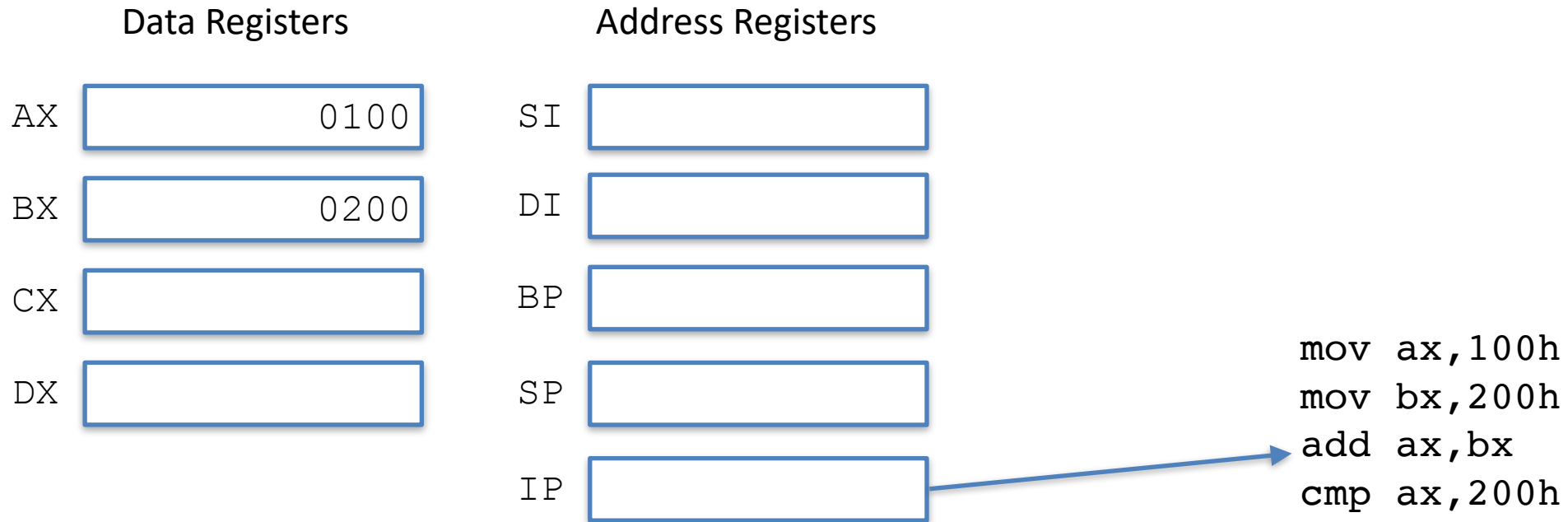
PROGRAMMER'S MODEL OF X86: INSIDE THE CPU



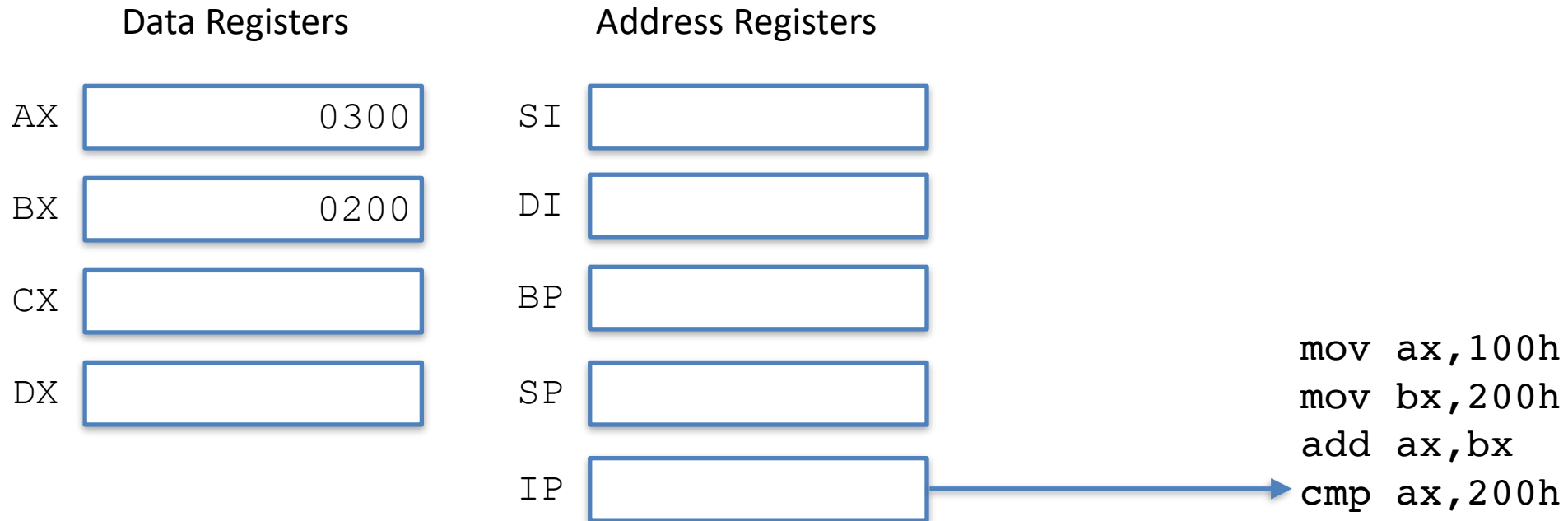
PROGRAMMER'S MODEL OF X86: INSIDE THE CPU



PROGRAMMER'S MODEL OF X86: INSIDE THE CPU



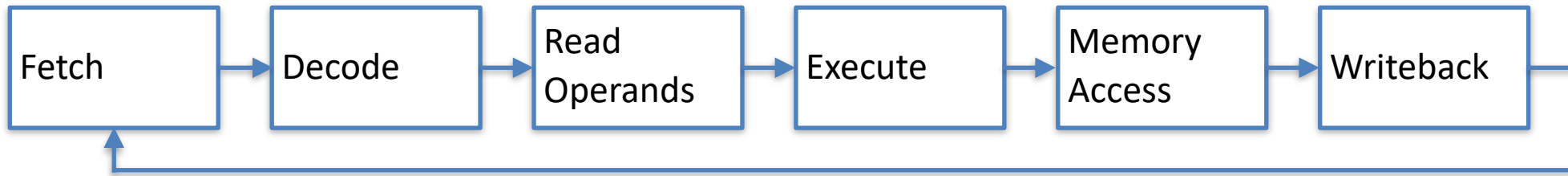
PROGRAMMER'S MODEL OF X86: INSIDE THE CPU



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 your GitHub username.** `neil@cs.luc.edu`