# Lecture 9: file i/o & `struct` your stuff

CSE 29: Systems Programming and Software Tools

Olivia Weng

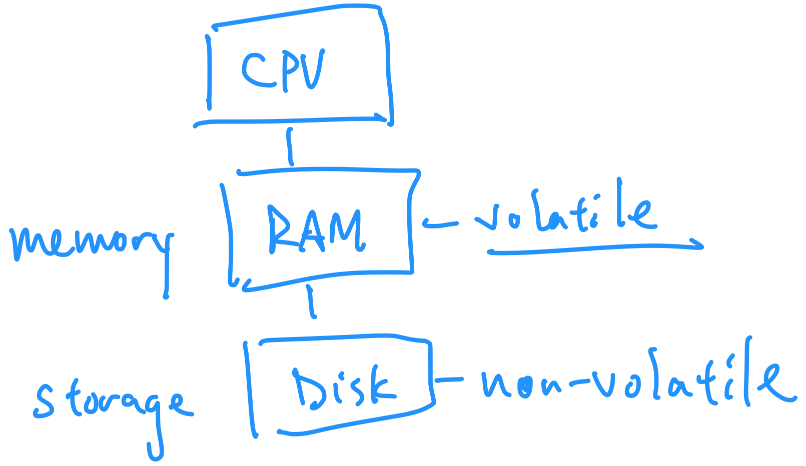
## What is file I/O?

- I/O = input/output
- What can we do with files?
  - `fopen()`: Open a file
  - `fclose()`: Close a file
  - `fgets()`: Read a string from a file
  - `fprintf()`: Write to a file
  - and more!

- We take input from STDIN and output to STDOUT…
  - In Unix, everything is a file!
  - `STDIN` and `STDOUT` are files

## What if I don't close a file?

- Usually the OS will close it for you, but weird things can happen…
    - The file doesn't get written to disk and just sits in RAM
    - Opening files will start to fail

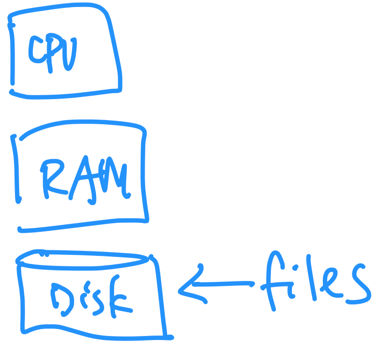


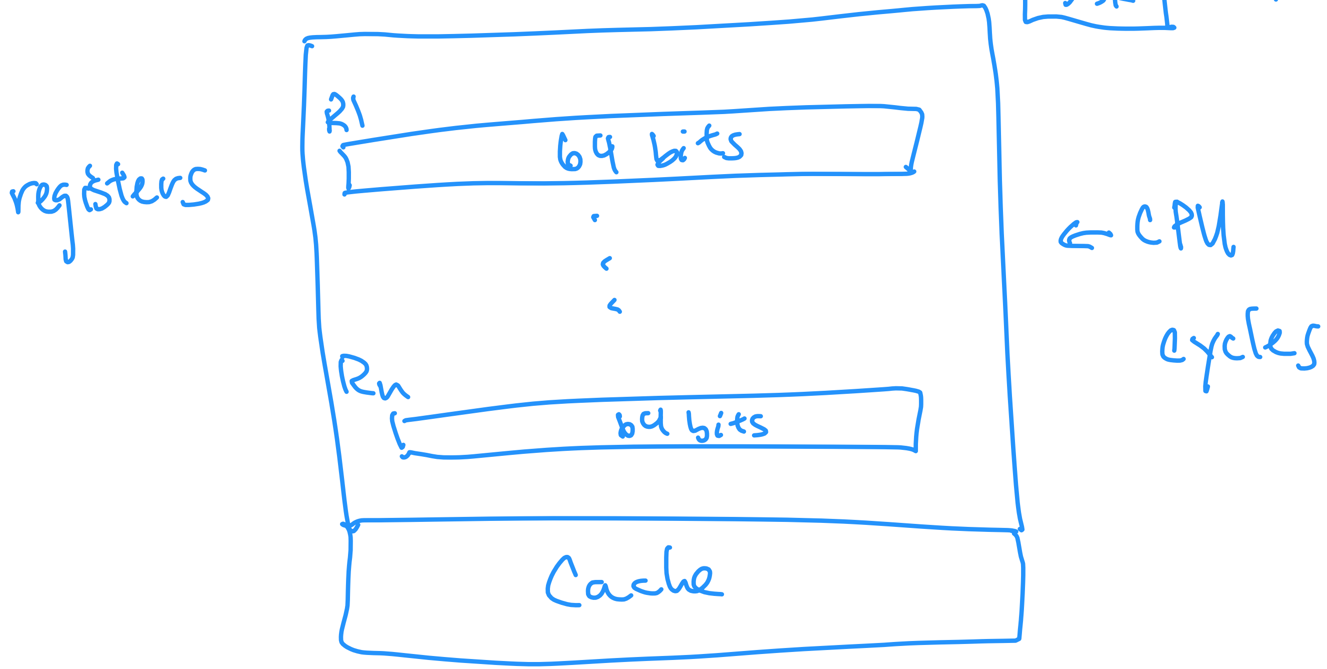
## Demo

- `read_and_write_a_file()`

## Why not store everything in files?

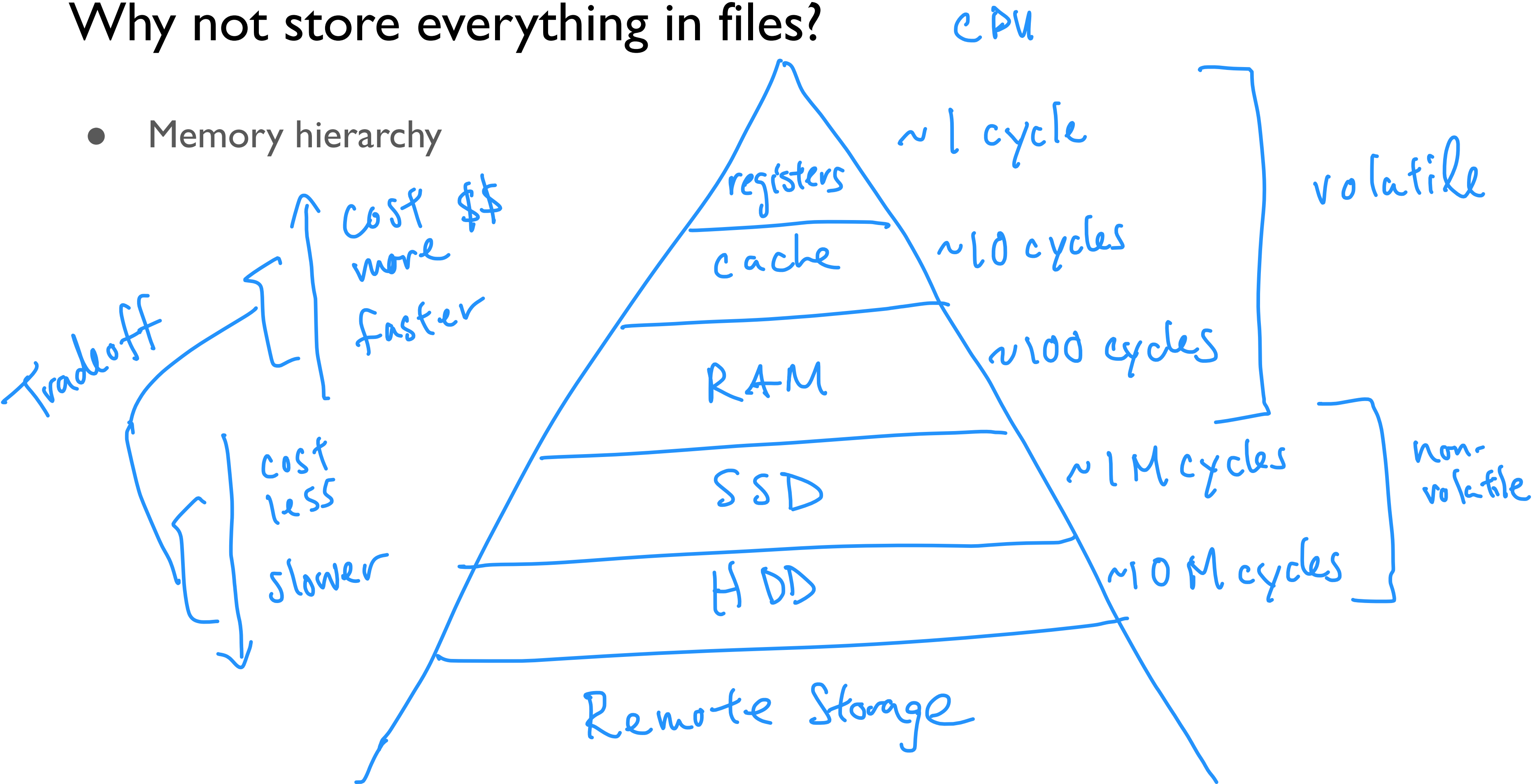
- Computer has many kinds of memory and storage



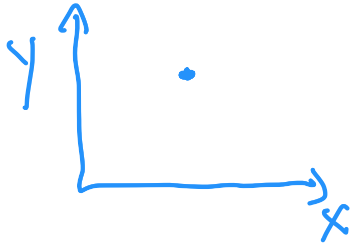


## Why not store everything in files?

- Memory hierarchy



## How can I represent Cartesian coordinates?



C

```
int point[2];
int point[1][1]; ← double check
```

Java

```
class Point
    int x;
    int y
```

## Introducing `struct` datatype

- Up until now, all variables have been either single elements or arrays
  - Example: Cartesian coordinates

```
int x;

int y;
```

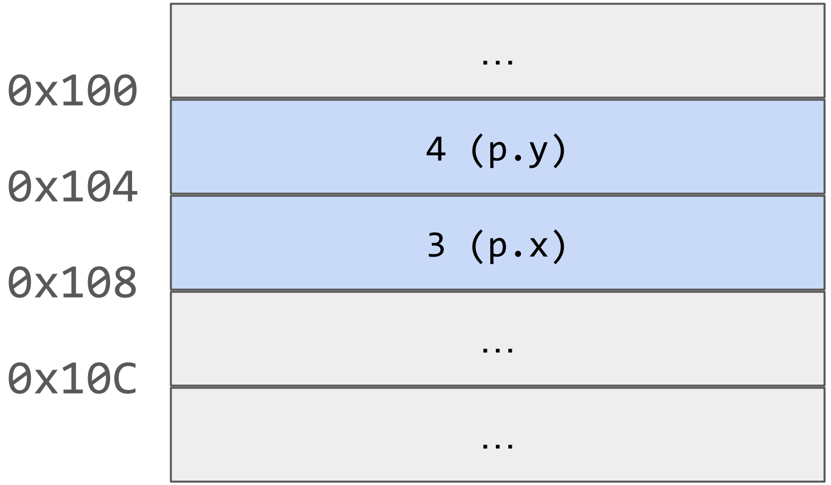
- The `struct` datatype can combine elements into one variable

```
struct point {

    int x;

    int y;

};
```

## Using struct datatypes

```
struct point {
    int x; // data member
    int y;
};
int main() {
    struct point p;
    p.x = 3;
    p.y = 4;
    // or struct point p = {3, 4};
}
```

(3, 4)



## Demo

```
struct point new_point(int x, int y);
```

## Passing a struct to a function

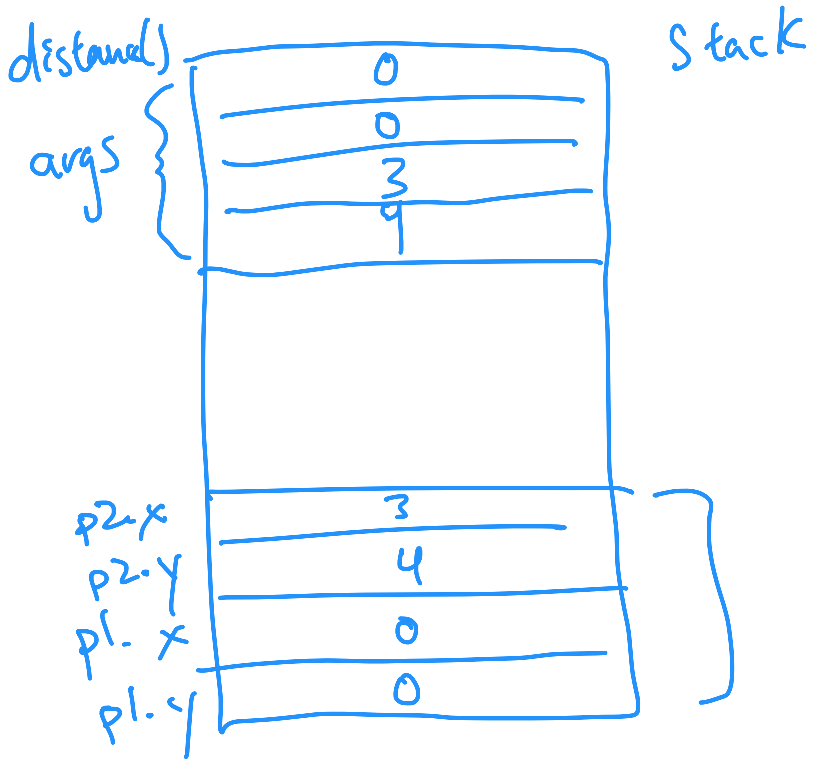
```
double distance(struct point p1, struct point p2) {
    // demo
}
int main() {
    struct point p1 = new_point(0, 0);
    struct point p2 = new_point(3, 4);
    double dist = distance(p1, p2); // structs passed by value
}
```

copied into the function

## structs are copied when passed as arguments to function

- Any issues with this?

Wasting memory



How can we avoid copying structs and wasting memory?

## How can we avoid copying structs?

- Pointers!

```
int main() {
    struct point p1;
    struct point *pp1 = &p1;
    ~~p->x = 0;~~
    ~~p->y = 0;~~
}
```

pp1->x = 0;
pp1->y = 0;

→ operator

(*pp1).x = 0;
(*pp1).y = 0;

## Passing struct to a function using pointers

```
double distance_v2(struct point *p1, struct point *p2) {
    // demo
}
```