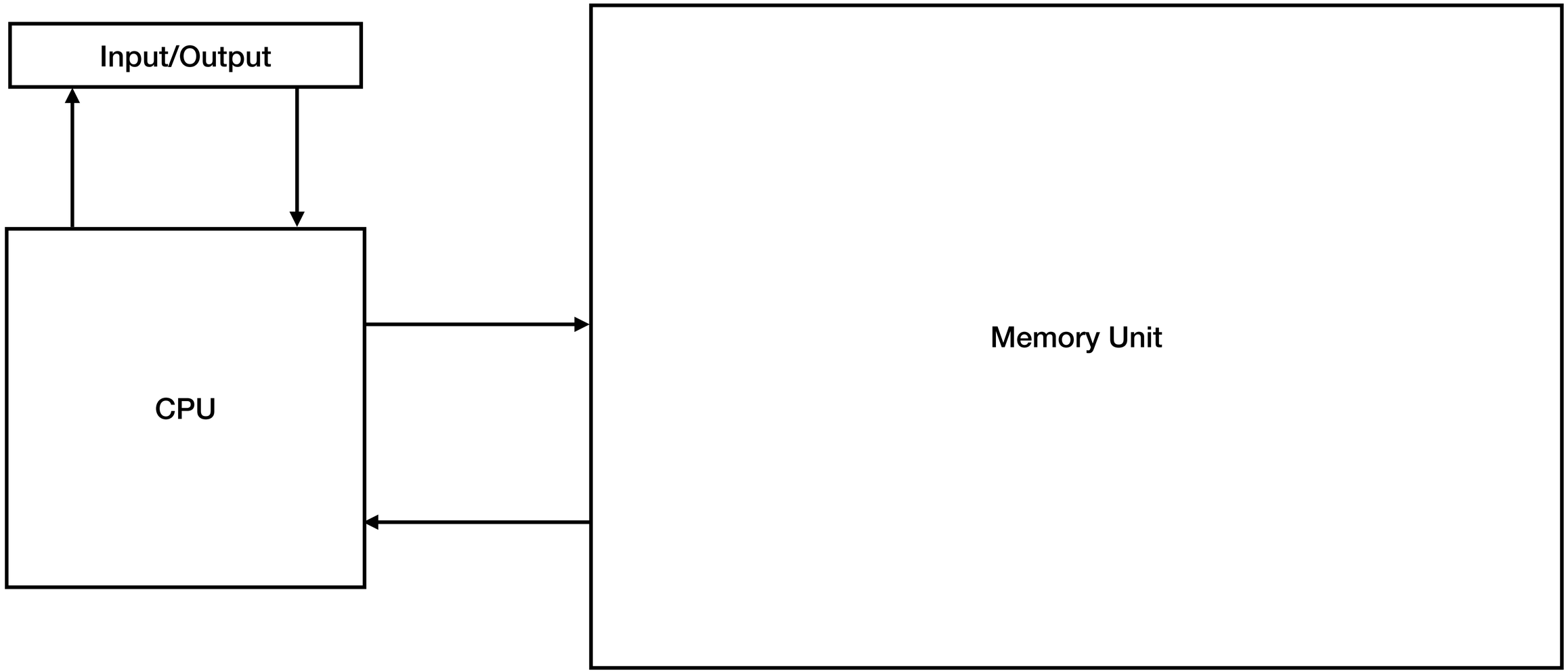
Wrap Up and Look Forward

CS143: lecture 20

Byron Zhong, August 1

A Von Neumann Machine



A new perspective...

- Variables (data) and functions (code) live in memory
  - Memory is a contiguous storage of bytes
  - Each byte has an address -- variables and functions have addresses
- When executing a program, CPU fetches an instruction from memory and performs actions:
  - Read (n bytes) from an address to a register, write (n bytes) to an address to a register
  - Manipulate the bits in registers -- computation
  - Jump to another instruction, check for conditions, ...
- The compiler `clang` translates your C program into these instructions

A new perspective...

- A process's memory is partitioned into
  - The stack: the compiler uses this to manage local variables. Stack frames come and go as functions are called and return
  - The heap: you use this to store data with complicated lifetime
    - `ptr = malloc(n);`
    - `free(ptr);`
    - One malloc, one free
  - Code, global variables, string literals ...
- Virtual memory: OS gives each process its own memory address space (0 -- FFFFFFF...)

A new perspective...

- Data and code are just bits
  - A bit answers a yes/no question -- we specify what the questions are by agreeing on an encoding
  - Unsigned integer encoding -- each bit indicates the presence of a power of 2
  - Signed integer encoding (2's complement) -- the highest bit is negative
  - We can come up with our own encodings (e.g. student record)
- Types are used to keep track of the encodings

A new perspective...

- Statically, we can organize data...
  - ... of different types into a `struct`
    - to represent a real-world object
    - to group variables that are dependent (invariants)
  - ... of the same type into an array
    - to represent multiple instances of the same thing
    - to apply the same action repeatedly
- Compiler translates structs and arrays access into direct memory access

A new perspective...

- Dynamically, we can organize data as:
  - `list`: an ordered sequence
    - If we use pointers to keep track of the order -- linked list
      - Easy to reorder, insert, delete, ...
    - If we use relative memory position to keep track of the order -- arraylist
      - Easy to access specific element
  - `map`: a collection of key-value pairs
    - BST -- if the keys are ordered
    - Hash Table -- if the keys can be converted to an integer -- need to handle collision

HW0: C basics

- Syntax, types, conditionals, compiling, ASCII...
- `printf("%c %6.2f\n", 'A' + i, freq * 100);`

HW1: `readline`, `split` and `join`

- C string: `'\0'`
- Allocation, reallocation, deallocation, ...
- **`char`** `**line_p`
- Pointers, pointers, pointers

HW2, HW3, HW4

- Data Structures:
  - Linked list
  - BST
  - Hash table
- lib directory

HW5: Groups

- File I/O
- Using map and list
- Memory cleaning
- Higher-order functions

HW6: Compress

- Bits!

Topics Covered

Memory:

- Variables and types
- Array
- Types
- Pointers
- Pass by reference
- Function frames
- Stack and Heap

Data structure:

- Array List
- Linked List
- Tree & BST
- Hash Table
- Min Heap
- Selection, insertion, bubble sort
- Tree sort, heap sort,
- Counting sort

Bits:

- Bitwise operations
- Integer representation
- Bit-packing
- Masks
- Binary and hex
- Endianness

Other:

- Threads
- Virtual memory
- Dynamic dispatch
- Terminal
- Git
- Compiler
- Makefile
- Valgrind
- Machine structure

What next?

- Data structure, complexity, sorting:
  - CMSC 27200. Theory of Algorithms
- File, permanent storage, bits:
  - CMSC 23500. Introduction to Database Systems
- Memory, instructions, language:
  - CMSC 14400 Systems Programming II
  - CMSC 22200. Computer Architecture
  - CMSC 22600. Compilers for Computer Languages
- Communication, bits, systems:
  - CMSC 23300. Networks and Distributed Systems
- Concurrency, threads, scheduling:
  - CMSC 23000. Operating Systems
  - CMSC 23010. Parallel Computing

... and many more!

Study for Final

- Binary, hex, decimal conversion (both signed and unsigned)
  - Write a C function that prints all the bits of a 64-bit integer in binary and in hex
  - Randomly pick a number, convert it to hex, binary, decimal
  - Review bit-packing and bit flags
- Hash table
  - What is a good hash function? What is a *problematic* hash function?
  - How collisions happen?
  - Chaining
  - Probing -- why do we need tombstones?
- Tagged union
  - Write a tagged union called Car with variants SUV, Sedan, Truck