CS 211 RECITATIONS WEEK 2

Wenjie Qiu
Teaching Assistant
Office Hour: Thursday noon to 1pm
wenjie.qiu@rutgers.edu
https://github.com/Roadsong/rutgers-cs211-recitations

Before we get start it…

- A bit about myself.
  - *Wenjie Qiu*
  - *CCNU, WUSTL, Rutgers…*
- TA for the first time.
  - *Nervous!*
  - *Interactive recitations.*
  - *Time?*
- What should you prepare?
  - *Watch the Jeff's (last week's) videos and gets your hands dirty.*
- How to ask a good question?
  - *We are engineering students, take sometimes to have a try!*
  - *Google is our friends. (Stackoverflow, Github, blogs, books, etc.)*
  - *Ask a specific questions. Which part confused you?*
  - *Learning by doing, you'll learn more if the problem is finally solved by yourself.*
  - *Instructors are not 100% correct, learn from various resources.*

Content

- Set up environments.
- Know about your computers.
- C programming language.

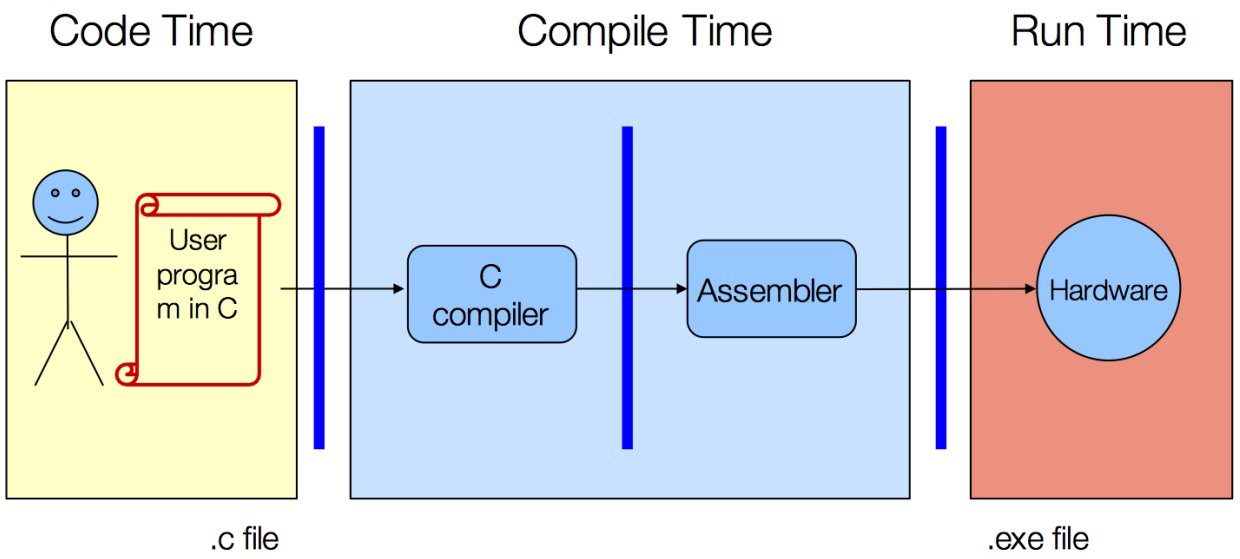
Set up environments

- Environment settings
  - *Jeff's instructions: this link.*
  - *ilab introduction: https://resources.cs.rutgers.edu/docs/instructional-lab/*
- Linux Operating Systems
  - *For us beginners:*
    - https://resources.cs.rutgers.edu/docs/new-users/beginners-info/
  - *We are using Linux*
    - Ubuntu actually, and advanced package tool (apt) is your friend
  - *What is a shell?*
  - *gcc, clang(?), compiler matters*
  - *command with options*
  - *32-bits executables*
  - *you will not destroy ilab machine (easily)*
  - *vim editors*

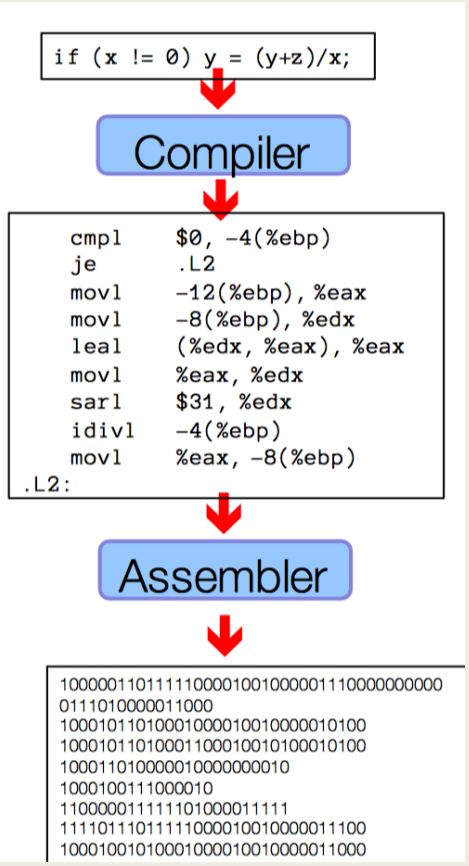
Know about your computers

HW/SW Interface:

Code / Compile / Run Times

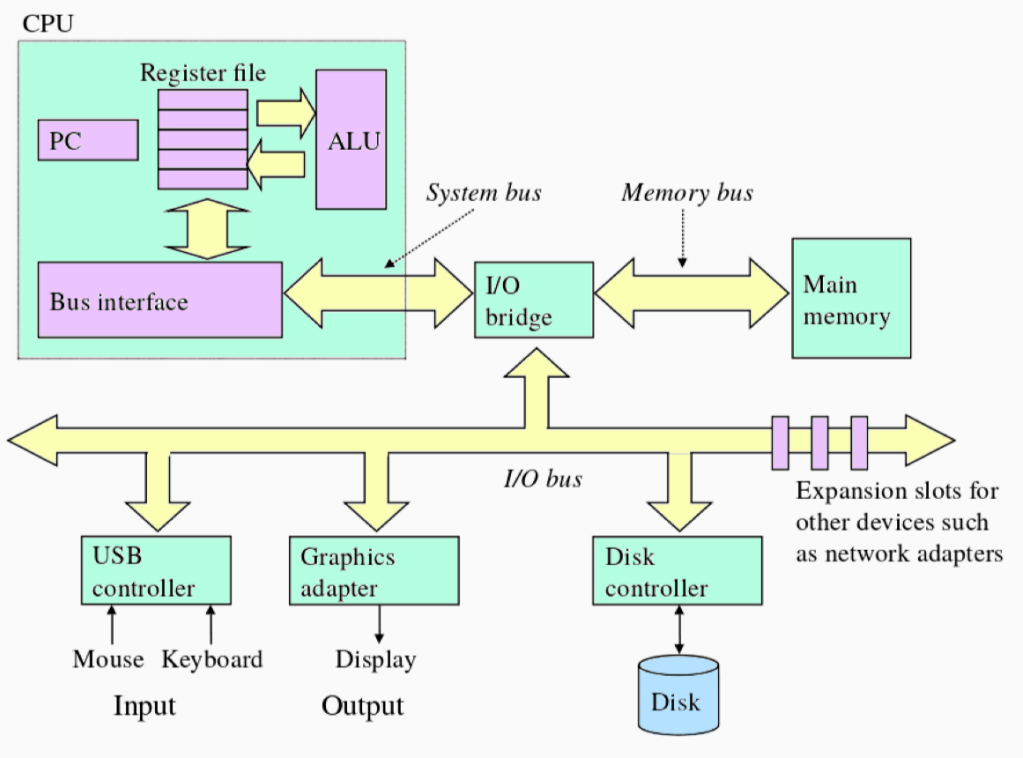


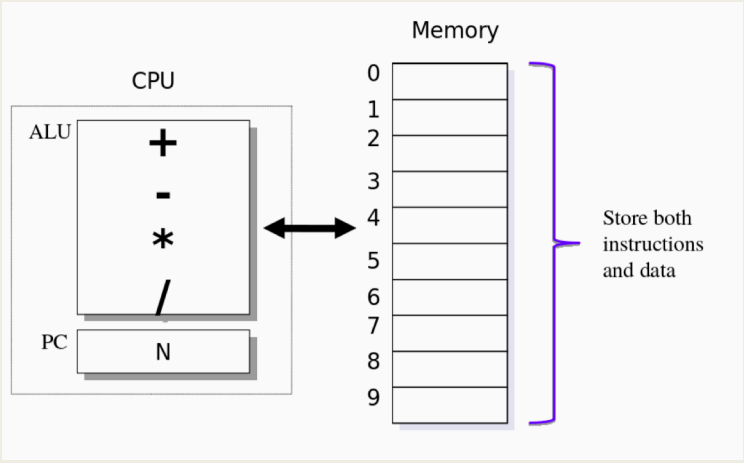
Note: The compiler and assembler are just programs, developed using this same process.



https://courses.cs.washington.edu/courses/cse351/18sp/schedule.html

Von Neuman Model

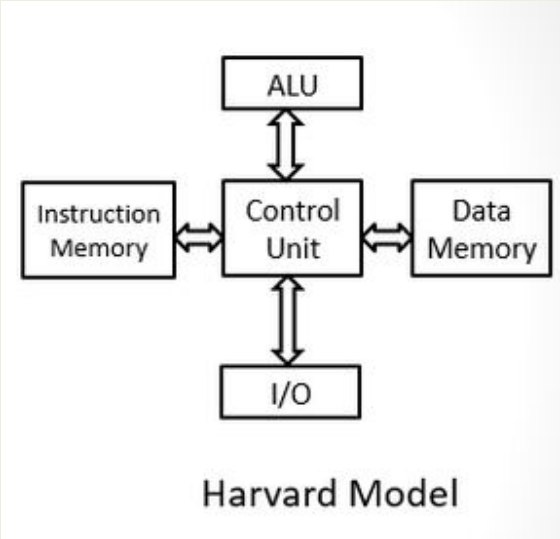




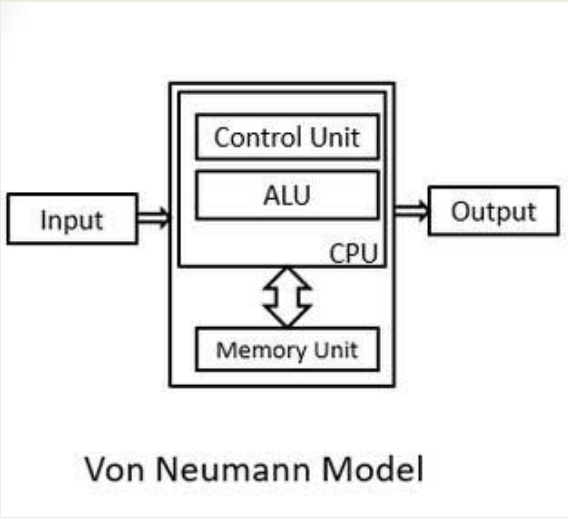
ALU: arithmetic logic unit
Responsible for do the actually math computations.

PC: program counter
Sometimes called instructions pointer (IP)

Any other models?



separate data and program memory



No separate data and program memory

Comparison with Harvard model: https://www.thecrazyprogrammer.com/2019/02/difference-between-von-neumann-and-harvard-architecture.html

C Programming language

- C standard (ANSI C (C89), C99)
  - *It matters*
- Comments
  - */* this is a comment */ (multi-line)*
  - *// This is also a comment (single line)*
- Return value
  - *void function -> return;*
  - *main function -> return 0;*
    - EXIT_SUCCESS
    - EXIT_FAILURE
  - *char function -> return ‘a’;*

Variables and Types

- Each variable has a type, which tells the compiler how to interpret it.
  - *Different interpretations lead to different results, sometimes error.*
  - *Don't play with type casting.*
- Basic data types
  - *char, int, float, double*
  - *short, long, long long (I don't like it)*
  - *signed, unsigned*
- Special number notations
  - *0x1010, 10e2*
- Scopes
  - *local (This is tricky!)*
  - *global*

scope.c

```
1  #include <stdio.h>
2
3  int i = 1;
4
5  int test(int i)
6  {
7      return i * i;
8  }
9
10 int main(int argc, char const *argv[])
11 {
12     int i = 2;
13
14     {
15         int i = 3;
16     }
17
18     for (int i = 4; i < 10; ++i)
19     {
20         ;
21     }
22
23     printf("%d\n", test(i));
24
25     return 0;
26 }
```

Example #1 Local variables

Variable i and their scopes?

What is the result?

What is the result if line 5 is

int test(int n)

Conditional Statements

- if else is easy
- switch
  - *A switch statement can always be replaced by if (then) else, just like in Python.*
  - *default is optional but highly recommended.*
  - *switch case fall through.*

```
switch(expression) {
  case const-1: statements-1;
  case const-2: statements-2;
  default: statements-n;
}
```

```
int fork;
…
switch(fork) {
     case 1:
         printf(“take left’’);
     case 2:
         printf(“take right”);
         break;
     case 3:
         printf(“make U turn”);
         break;
     default:
         printf(“go straight”);
}
```

Loops

Statement	Repeats set of statements
while (expression) {…}	zero or more times, while expression != 0, compute expression before each iteration
do {...} while (expression)	one or more times, while expression != 0 compute expression after each iteration
for (start-expression; cond-expression; update-expression) {…}	zero or more times while cond-expression != 0 compute start-expression before 1st iteration compute update-expression after each iteration

break & continue & goto

- break
  - *exit current switch or loop (what about multi-level loop?)*
  - *forget about it, go to the statements right after the switch or loop.*
- continue
  - *skip the rest of computation of current iteration of loop.*
  - *go back to evaluate the expression for the next iteration.*
- goto
  - *Dangerous but you will see it (and probably will use it!)*
  - *Linux kernel and embedded system developers tend to use it.*
  - *Sometimes not a bad choice.*

The syntax for a **goto** statement in C is as follows

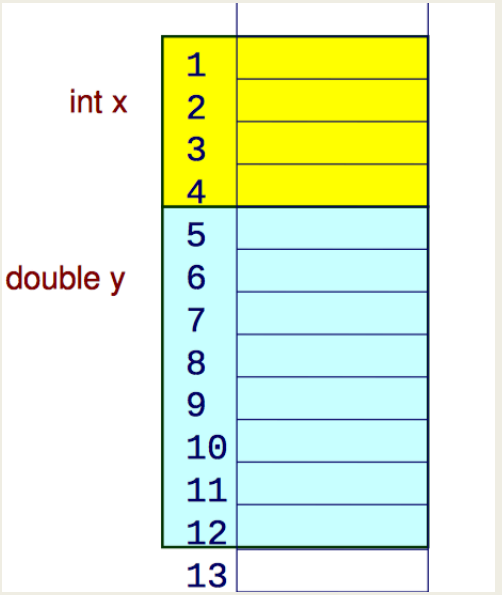
```
goto label;
..
.
label: statement;
```

Functions

```
int factorial(int n)
{
    int i;
    int result = 1;
    for (i = 1; i <= n; i++)
        result *= i;
    return result;
}
```

- Components: name, return type, parameters, body.
- Function call as part of an expression.
  - *use the return value*
- Function call as a statement. (return value will be discarded)
  - *return value will be discarded*
  - *but it will compute anyway*
  - *side effects*
- Function prototypes
  - *to declare I have a function “int factorial(int n);”*
  - *implementation can be done later.*

Memory and Pointers



- Underlying virtual memory system.
- An integer needs 4 bytes.
- A double needs 8 bytes.
- Typically maps to **smallest address**. (so what about x and y?)

- A pointer is just **an address**.
- int n = 123;
- int *p = &n; (initialize an integer pointer p, feed with the **address of integer n**, make it points to integer n)

- *p gives the value of item it points to, i.e., 123.

- What about the memory layout of an integer array?