09/30/19

lnxsrv0(6/7/9).seas.ucla.edu

cs31.seas.ucla.edu

g31 -> complier + visual c++

UCLA Web or eduroam

Due 11 pm

Start early 😉

Develop incrementally

Read what we write (specification for each assignment) – match all

https://rds.seaslab.seas.ucla.edu/RDWeb/Feed/webfeed.aspx

seaslab\rayh

Kpy60701

Continued:

* Audio: sample points -> subset of actually -> storage -> loss in data
* Colored Picture: division of the picture -> pixel (picture element) -> darkest + RGB
* Videos: both concepts -> compression: similarity between each frame

Machine Language:

* Memory

1. 21321
2. 12343
3. 59548
4. 49485
5. 94844
6. 34024

999:

CPU:

* Accumulator: 43454 (5 digits #)
* Instruction counter: 000 (3 digits #)

Instruction -> Memory -> CPU

21 001

Operation code & Address

21: copy the number at the indicated memory address -> accumulator

12343 – execution -> increase instruction counter (arithmetic logic unit, a part of CPU)

10/02/19

Interference -> characteristics of transistor (two possible states)

* Ex: low & high voltages – gate (and & or)
  + Combinations: 2 bits -> addition -> building CPU & memory (collection of number)

CS 51 – Logic Circuit

* Memory

0. 1. 2….

CPU: break codes -> operation code + address -> operation code runs from a list (ex. 99. Stop)

CPU:

* Accumulator: 43454 (5 digits #)
* Instruction counter: 000 (3 digits #)

Instruction -> Memory -> CPU

21 001

Operation code & Address

21: copy the number at the indicated memory address -> accumulator

Why separate?

CPU – fast memory (small amount) to access memory

RAM – bulk ~ available

Hard drive – even slower (faster = more expensive) -> archive storage

Data wiped out when replaced ~ variables

Common mistakes – hard to spot

Error – undefined behaviors

Ex. Prices + Fees -> algorithm: load price, add fees, store total, halt (relocate each to memory)

Conversion of typed codes/ instructions -> machine languages

Ex: load price, add fees, store total, alt

* Price data 42, fee data 13 -> total data

Assembly Language (translator language) – assembled by the assembler

New models supporting older machines (upward comp) -> vendor lock-in (unable to switch coding/ machine language support)

FORTRAN (IBM) -> Higher-level Language

Ex.

Integer price = 42

Integer fees = 13

Integer total = price + fees

Outside translator for FORTRAN -> prevent lock-in

* Higher-level language: a program is compiled by the compiler (~assembler)

Developments: BCPL

C: Early – popular for smaller systems

C++ -> Objective-C -> Java (C++ w/o complexity?) -> C#, Swift,

[Scripting languages: Python, Perl, Ruby] - Javascript

(Apple -> Objective-C -> Swift) (Microsoft: Java -> C#)

C++ Bjarne Stroustrup

1980

1985 -> released

1998 ISO C++ Standard (C++ 98)

2011 revision of the C++ standard (C++ 11)

2014 revision (C++14)

2017 revision (C++17)

2020 revision (C++20)

C++ Experience with Pointer (better to know the root)

2-phase processes -> errors (2 places)

1. Unable to translate – incorrect instruction (coding error -> compilation/ syntax errors)
2. Machine languages unable to run (logic/ runtime error)
   1. Difficult to determine

Print -> (cout = standard output designation)

#include <isostream>

using namespace std;

int main () – one only, execution

{

cout << “Text” << end];

}