COMP 364: A Code Tasting

Carlos G. Oliver, Christopher J.F. Cameron

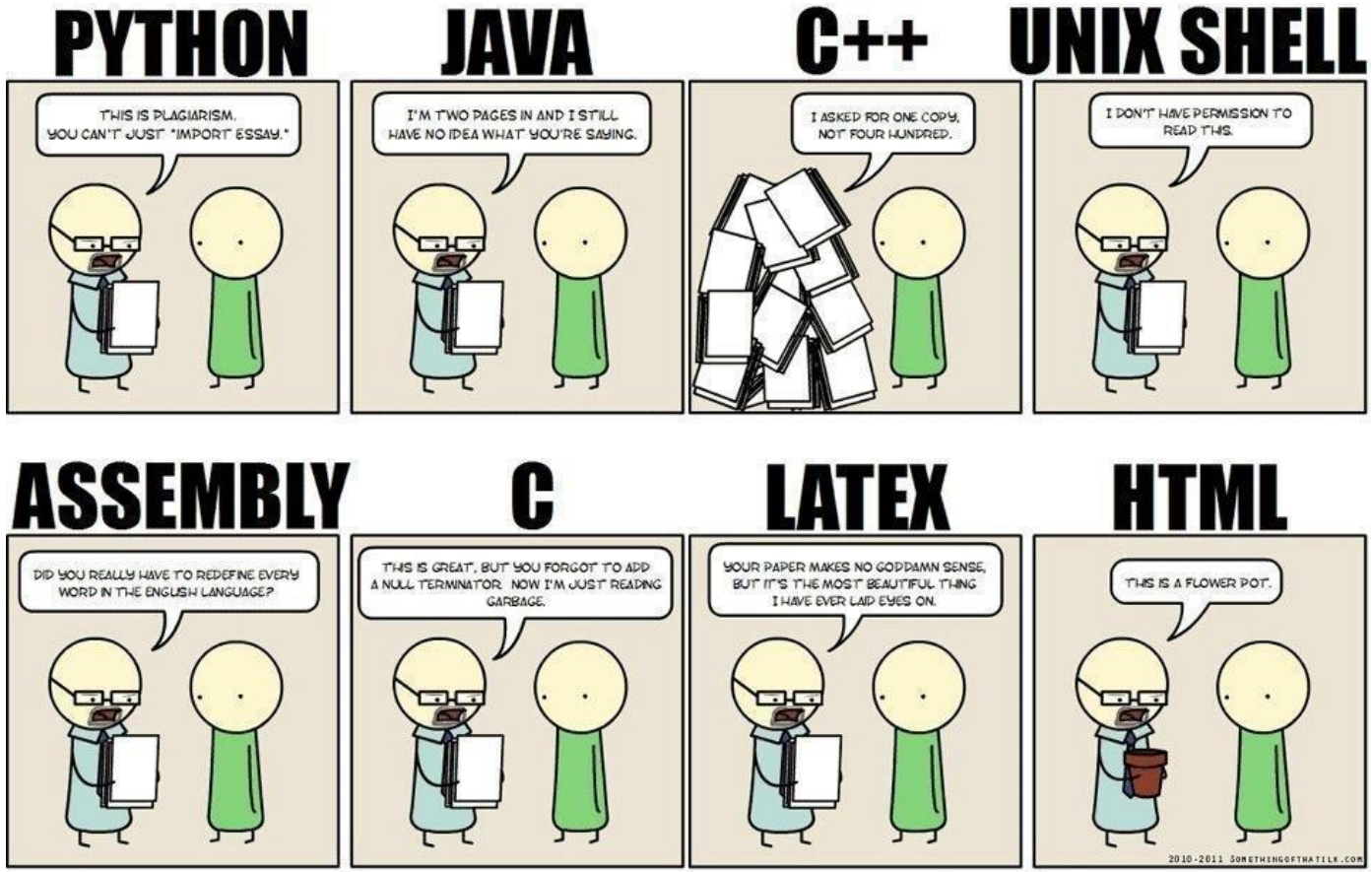
December 4, 2017

Final Exam

- Weight: 40 %
  - Multiple Choice
  - Short answer: short coding and theory questions
  - Long answer
- One double sided cheat sheet
- Review session: Wednesday Dec. 6 in class
- Material: Everything up to and including scikit-image (Lecture 33)

The world of programming languages

- There are thousands of programming languages
- Each has their strengths and weaknesses depending on the problem to be solved.



- Today we'll do a very quick tasting of some of the major languages.

Outline

We can roughly assign languages to general *families* or "programming paradigms".

1. Machine languages: e.g. Assembly
2. Procedural programming: e.g C
3. Object Oriented: e.g. Java
4. Functional Programming: e.g. Haskell
5. Just for fun, esoteric languages: e.g Brainfuck

We will see examples of "Hello world!" and Fibonacci in these and other languages today.

High vs. Low level languages

Levels of Programming Languages

High-level program

```
class Triangle {
  ...
  float surface()
    return b*h/2;
  }
```

Low-level program

```
LOAD r1,b
LOAD r2,h
MUL r1,r2
DIV r1,#2
RET
```

Executable Machine code

```
0001001001000101
0010010011101100
10101101001...
```


[1] https://cdn.educba.com/academy/wp-content/uploads/2015/11/35.jpg

Starting point: Python

- Python is a "multi-paradigm" language. It can behave as multiple programming paradigms.

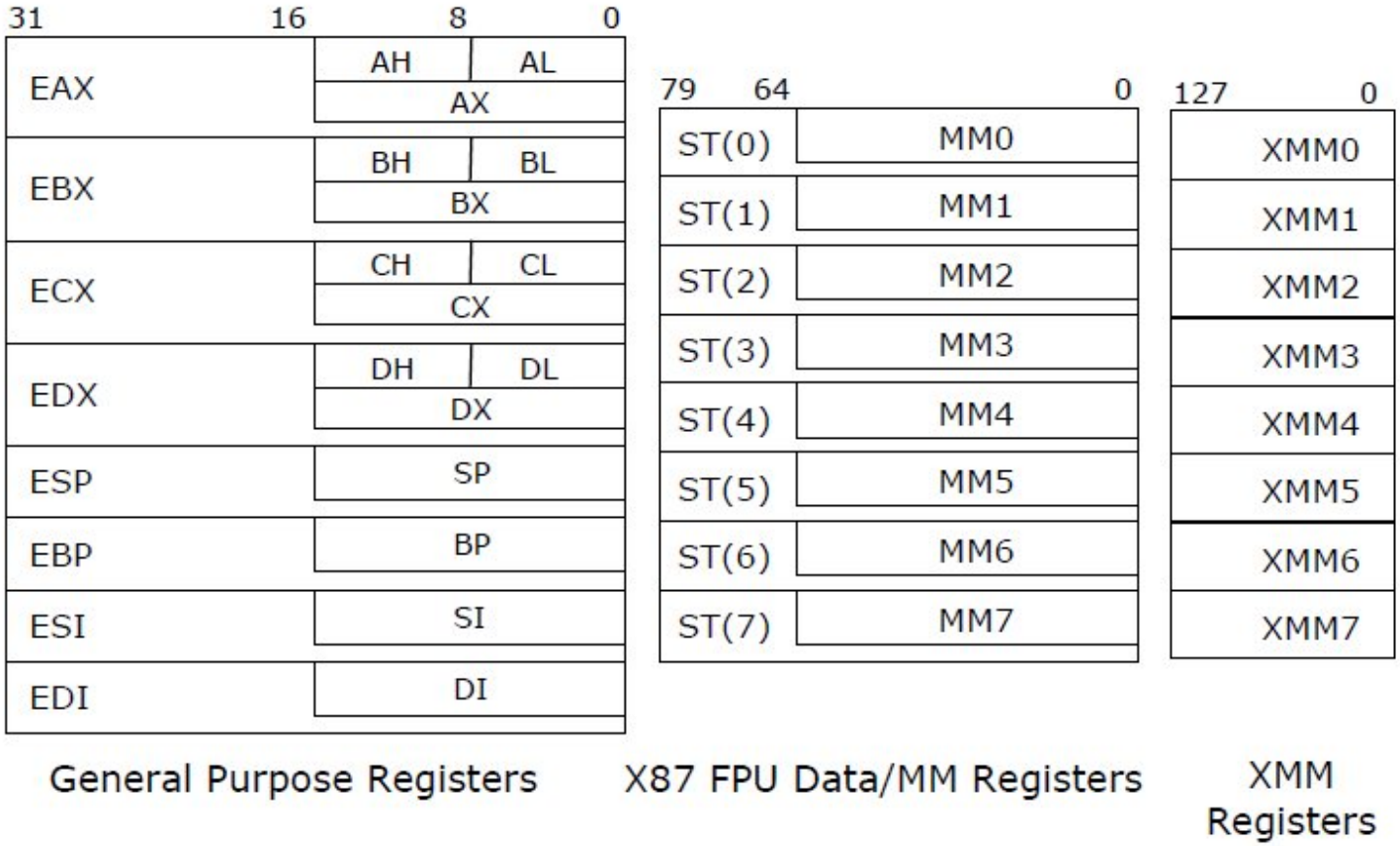
```
1  print("Hello World!")
```

Fibonacci sequence: $F_n = F_{n-1} + F_{n-2}$ where $F_0 = 0, F_1 = 1$

```
1  def fib():
2      a = 0
3      b = 1
4      while True:
5          yield a
6          a, b = b, a + b
```

Machine language: Assembly

- A small number of low level machine instructions to manipulate data stored in memory registers.
- e.g. `MOV EAX, 1h` moves the value `1h` to register `AL`
- e.g. `ADD EAX, 0x` adds the value in register `EAX` to hexadecimal `0x`



Machine language: Assembly

Usage:

- Low memory footprint, fast, compact
- Interacting directly with hardware

Drawbacks (?):

- Not very user friendly

Hello world in Assembly

```
.text
    .global _start
_start:
 # write our string to stdout.
mov    len,edx          # third arg: message length.
mov    msg,ecx          # second arg: pointer to message.
mov    1,ebx          # first arg: file handle (stdout).
mov    4,eax          # system call number (sys_write).
int    0x80            # call kernel.
# and exit.
movl    0,ebx       # first arg: exit code.
movl    1,eax       # system call number (sys_exit).
nt     0x80           # call kernel.
.data
msg:
.ascii  "Hello, world!\n"    # the string to print.
len = . - msg                  # length of the string.
```

Fibonacci in Assembly

```
main proc
        mov ecx, times          ;set loop counter to "times"
        sub ecx,2               ;loop times-2 times
top:
 cmp ecx, 0             ; test at top of loop
 je bottom              ; exit when while condition false
 xor ebx,ebx            ;Clear ebx
 mov ebx,first          ;move first into ebx
 add ebx,second         ;add ebx, [ first+second ]
 mov third,ebx          ;Copy [first+second] to third
 xor ebx,ebx            ;clear for further use
 mov ebx,second         ;move second into ebx
 mov first,ebx          ;copy ebx to second [first=second]
 xor ebx,ebx            ;clear for later use
 mov ebx,third          ;move third into ebx
 mov second,ebx         ;copy ebx to third [second=third]
 xor ebx,ebx            ;clear it
 dec ecx                ;decrement loop
```

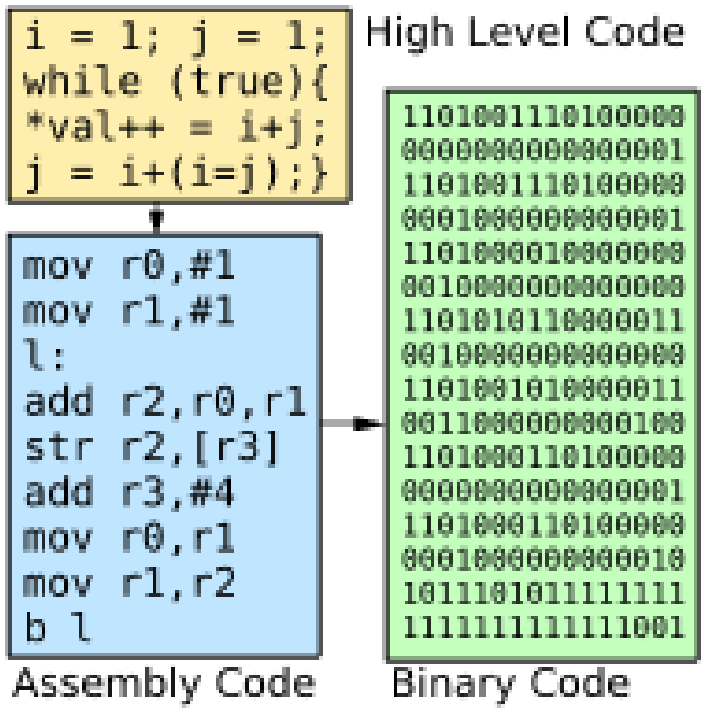
```
        bottom:
          push third
                push times
                push offset finalprint
                call printf
        push 0          ;exit gracefully
            call exit       ;exit system
    main endp

end main
```

https://stackoverflow.com/questions/35579861/fibonacci-series-in-assembly-x86

C

- High-level general purpose language with low-level access to memory.
- Source code is converted (compiled) into assembler and then to machine code.
- Very fast
- Many python built in functions and numpy are implemented in C for efficiency



Hello World in C

```
include<stdio.h>

main()
{
            printf("Hello World");

}
```

Fibonacci in C

```c
int main() //we have to declare return type of functions
{
   int n=10, first = 0, second = 1, next, c;

   for ( c = 0 ; c < n ; c++ )
   {
      if ( c <= 1 )
         next = c;
      else
      {
         next = first + second;
         first = second;
         second = next;
      }
      printf("%d\n",next);
   }
   return 0;
}
```

Pointers: Direct access to memory

```c
#include <stdio.h>
int main () {
   int  var = 20;   /* actual variable declaration */
   int  *ip;        /* pointer variable declaration */
   ip = &var;  /* store address of var in pointer*/
   printf("Address of var variable: %x\n", &var  );
   /* address stored in pointer variable */
   printf("Address stored in ip variable: %x\n", ip );
   /* access the value using the pointer */
   printf("Value of *ip variable: %d\n", *ip );
   return 0;
}
```

```
 Address of var variable:  bffd8b3c Address stored in
ip variable:  bffd8b3c Value of *ip variable:  20
```

Static vs Dynamic typing

- Languages like C are statically typed: at compile time, the data type of a variable is known according to variable declaration (e.g. `int x = 5`).
- Languages like Python are dynamically typed: interpreter figures out the type of a variable during runtime (e.g. `x = 5`).
- Static typing results in fast and memory efficient code as the compiler can optimize code with knowledge of type.
- Dynamic typing results in more flexible and easy to write code. Cons: slower and more type errors.

BASH

- Language for performing operating system tasks.
- Useful for automating command line tasks. (e.g moving files, backups, installing/executing programs, user permissions)

Hello world:

```
echo "Hello World"
```

Fibonacci

Move all `jpg` files starting with the letter a to desktop.

```
for x in `ls a*.jpg`; do mv $x ~/Desktop; done
```

Java

- Java is a high-level, strictly object oriented programming language.
- We manipulate collections of objects and their attributes.
- Python is also (kind of) an object oriented language.
- Java is a compiled and statically typed language.

Hello World in Java

```
public static void main(String [] args){
        System.out.println("Hello World!");
}
```

```
public class FibonacciSeries {

public static int fibIterative(int number) {
        if (number == 0 || number == 1)
                return number;
        int firstNumber = 0, secondNumber = 1;
        int fibNumber = 0;
        for (int series = 2; series <= number; series++) {
                fibNumber = firstNumber + secondNumber;
                firstNumber = secondNumber;
                secondNumber = fibNumber;
        }
        return fibNumber;
}
public static void main(String[] args) {
        int nextFib = fibIterative(5)
        }
}
```

Haskell

- Haskell is a fully functional programming language.
- i.e. code inside functions produces no side effects outside the function
- Results in very clean code, easy to debug.
- Makes heavy use of lazy evaluation (generators) and recursion (functions calling themselves)
- Potential con: steep learning curve.

Hello World in Haskell

```haskell
-- define function main, pass string to putStrLn function
main = putStrLn "Hello, World!"
```

Factorial: all information is thought of as inputs and outputs to functions (no variable assignment)

```haskell
-- define return input/output types
factorial :: Int -> Int
-- on input 0 function returns 1
factorial 0 = 1
factorial n = n * factorial (n-1)
```

Now we can call our function:

```haskell
factorial 42
1405006117752879898543142606244511569936384000000000
```

Fibonacci in Haskell

```haskell
fib :: Integer -> Integer
fib 0 = 0
fib 1 = 1
fib n = fib (n-1) + fib (n-2)
```

Esoteric languages: Brainfuck

- There are many esoteric programming languages
- Often these are written just for fun but can often help us think in different and useful ways about coding.
- Brainfuck visualizer
- There are only 8 commands in Brainfuck that can read/write/jump along a "tape" of memory cells. (see Turing Machine)

Hello world:

```
++++++++++[>+++++++>++++++++++>+++>+<<<<-
]>++.>+.+++++++..+++.>++.<<+++++++++++++++.>.
+++.------.--------.>+.>.
```

fibonacci

Honourable mention

- C++: C's low level access with high level constructs such as objects.
- JavaScript: web development, browser integration
- Ruby: Python alternative
- Lisp: List processing language, everything is list comprehensions
- TEX: very powerful typesetting/document preparation language

Thank you!