

Introduction to Scientific Computing I

Lecture 4

Amir Farbin

Announcements

- Lab 1 reminder
- Clinic reminder
- Laptops

Programming

- Each CPU understands its own instruction sets.
- Low level operations:
 - Copy in/out data from memory into registers.
 - Perform operations on registers.
 - Conditional (if/then) and Control flow (jump)
 - Manage a stack (where small sets of data can be shared between different blocks of code)
- Each instruction are each assigned a specific binary value and are not human readable...
- Assembly is a human-read language that most closely mirrors Machine Language.

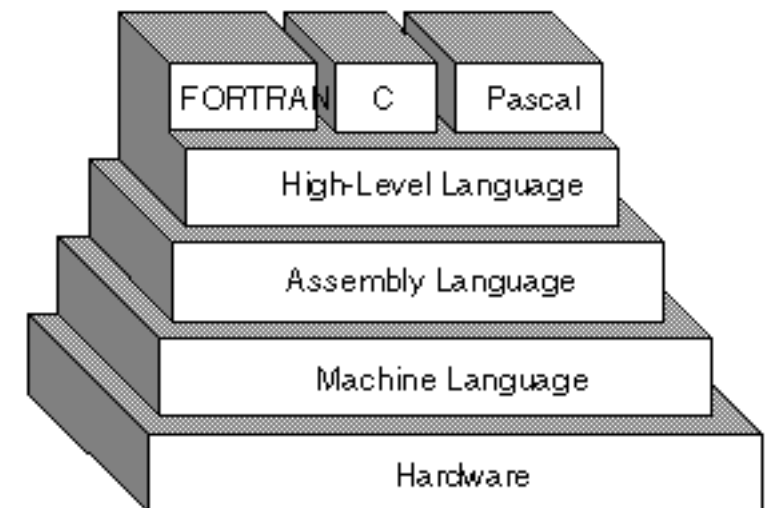
```
x3000 LD R1, x006           ; load data pointer
x3001 LDR R2, R1, #0         ; load data value
x3002 BRz x005             ; branch to end if zero
;
; repeating statements go here
;
x3003 ADD R1, R1, #1         ; increment data pointer
x3004 BRnzp x001           ; branch back to top
x3005 HALT
;
; data section
;
x3006 x4000 ; address of data

; The translation into LC-2 Machine Code
0011 0000 0000 0000        ; load at x3000
X3000 0010 001 0 0000 0110 ; LD R1, x006
x3001 0110 010 001 000000   ; LDR R2, R1, #0
x3002 0000 010 0 0000 0101   ; BRz x005
;
; repeating statements go here
;
x3003 0001 001 001 1 00001   ; ADD R1, R1, #1
x3004 0000 111 0 0000 0001   ; BRnzp x001
x3005 1111 0000 0010 0101    ; HALT
;
; data section
;
x3006 0100 0000 0000 0000    ; x4000
```

from: <http://www.eecs.umich.edu/courses/eecs284/example1.htm>

Programming Languages

- Writing Assembly (Machine) code requires
 - knowledge of the specific CPU,
 - working at level of very small operations and
 - awareness of registers, memory, and hardware.
- High-level languages provide high level abstractions and human friendlier syntax for programming.
- Two types:
 - **Compiled**: The text of the high level language is converted by a *compiler* into machine code. The machine code is run subsequently.
 - **Interpreted**: A program runs that reads the text of the high level language and performs the operations.



High-level Language

- 3 Fundamental Elements of any programming language:

1. Primitives:

- Numbers, Characters, ...
- Mathematical Operations: +, -, *, /, ...
- Logical Operations: and, or, ...
- Conditionals: if-then-else, ...

2. Means of Combination: e.g. list

3. Means of Abstraction:

- Assignment: `x = 1`
 - Definition: `def x: 1`
 - Function: `def f(x): x`
- Beyond these, there are universal programming concepts and patterns (e.g. object oriented programming) that enable or facilitate building sophisticated software.
 - While we will learn these in python, look beyond the syntax and specifics of the programming language.

Why Python?

- Interpreted: no compilation time. Multi-platform.
 - Expense of speed, but the time consuming code are often in compiled libraries.
- Large library: almost any package out there has a python API.
- Easy to read. Convenient syntax.
- Convenient data structures that are simple to build.
 - Advanced data structures: list, dictionaries, sets...
 - Dynamic typing: no need to declare the type of a variable.
 - Built-in memory management (reference counting + garbage collection): No need to worry about memory addresses or allocating/freeing memory.
 - Dynamic name resolution (late binding): same code can be reused for different data.
- Multi-paradigm:
 - structured programming: functions, sub-routines, etc...
 - functional programming: filter, map, reduce, lambda, generators, ...
 - object-oriented programming: class, inheritance, ...
 - ...
- Language of choice for Data Science.