The great divide: from hardware to software

What is software exactly ?

A sequence of operations for the hardware to execute.

Encoded as binary and stored in memory

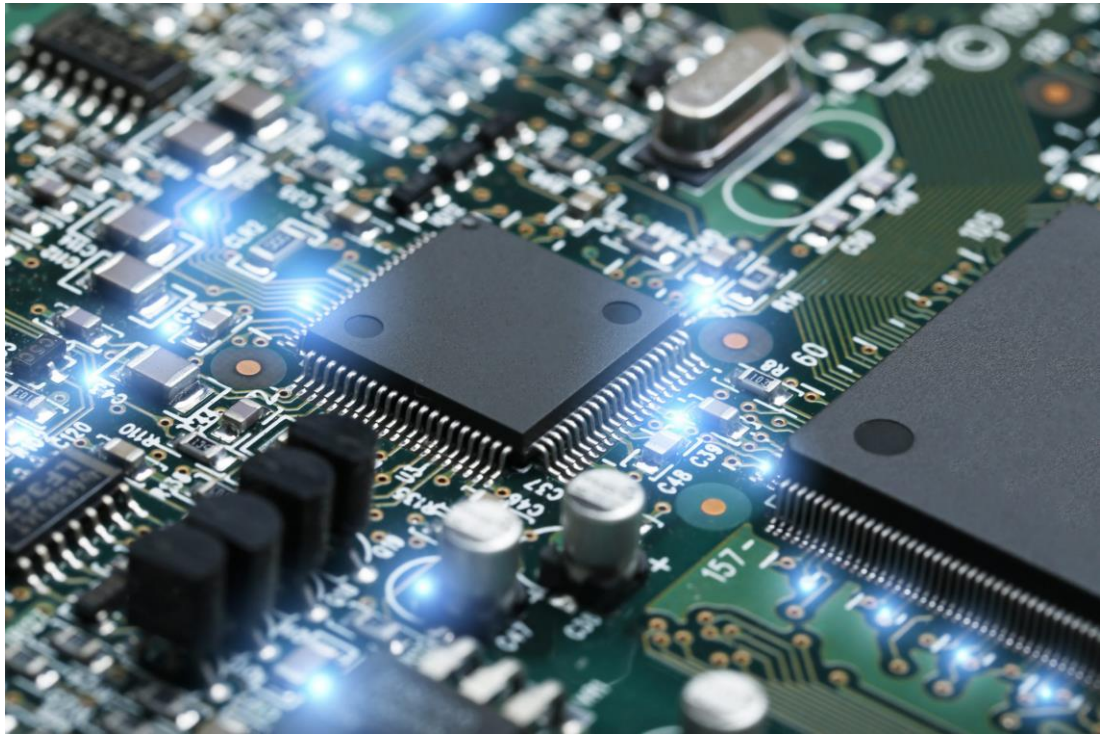
So its just binary – like any other kind of data

What is a programming language ?

A set of pre-defined human-readable instructions representing binary encodings

Also - a set of rules to guide how instructions are expressed by the programmer

Programming languages are less about computers, and more about people



Img src: https://railgallery.wongm.com/sunbury-line-disruption-ginifer-2019/F133_6745.jpg.html

Programming languages

Abstraction

Programming languages are an interface to the computer's capability

We can think of languages as residing at some level of abstraction from the underlying hardware

In fact, we've been climbing these these levels of abstraction the whole time in this unit

High level: languages which are increasingly human readable, and abstract away (or even deny access to) the underlying hardware executing it

Low level: languages typically offer more access to the underlying system

e.g., allowing direct access to memory locations, or ability to manipulate individual bits in a register

Image src: www.industryconnect.org/common-programming-languages/

Programming languages

Abstraction

Programming languages are an interface to the computer's capability

We can think of languages as residing at some level of abstraction from the underlying hardware

In fact, we've been climbing these these levels of abstraction the whole time in this unit

High level: languages which are increasingly human readable, and abstract away (or even deny access to) the underlying hardware executing it

Low level: languages typically offer more access to the underlying system

e.g., allowing direct access to memory locations, or ability to manipulate individual bits in a register

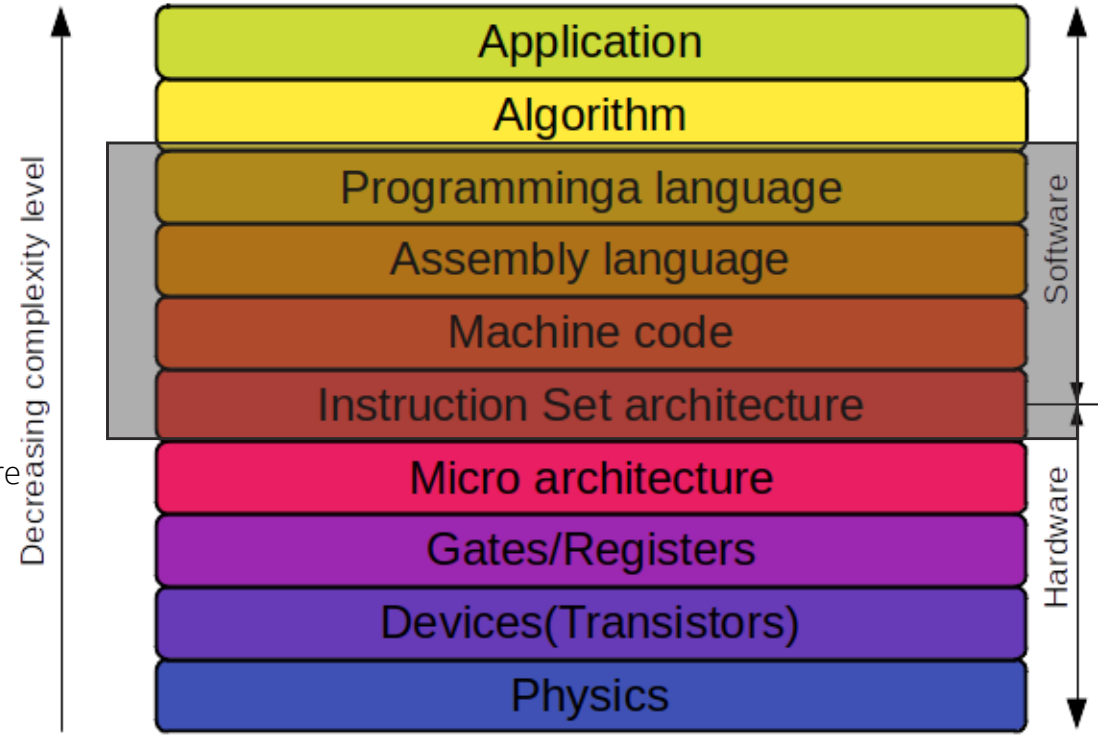


Image src: https://scimos.com/wp-content/uploads/2020/07/ABSTRACTION-LAYERS-e1594137453801.png

Programming languages

High Level Programming Language Examples

High level instructions

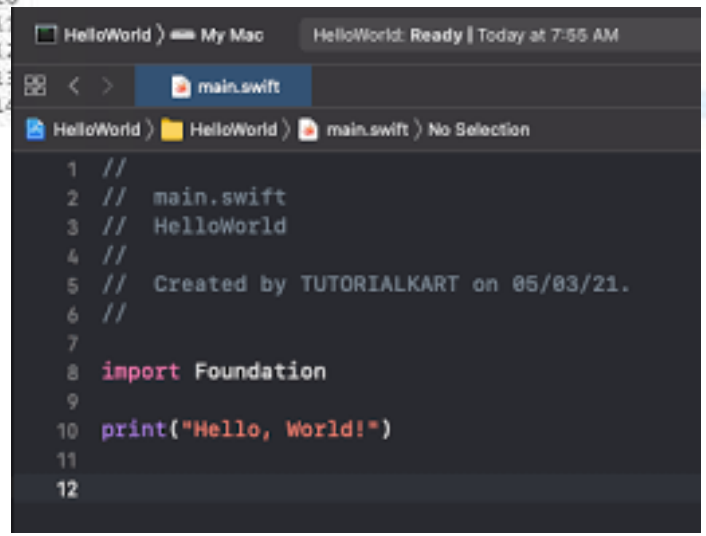
Hello World !

```java
package com.example.helloworld;

public class HelloWorld {

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

}
```



Programming languages

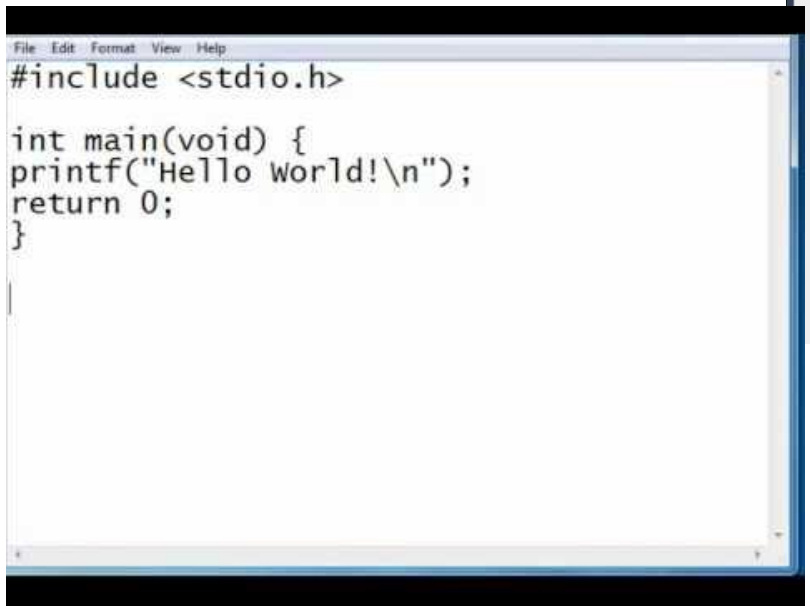
Low(er) level examples

ARM Assembly (e.g., Raspberry Pi, most mobile devices)

X86 Assembly (e.g., , Intel processors)

Low level instructions

Hello world!



Hello.cpp*

(Global Scope)

```cpp
#include <iostream>
using namespace std;

int main()
{
    //say hello
    cout << "Hello C++" << endl;

    system("PAUSE");
    return 0;
}

```

```
    .equ    LAST_RAM_WORD,   0x007FFFFC
    .equ    JTAG_UART_BASE,  0x10001000
    .equ    DATA_OFFSET,     0
    .equ    STATUS_OFFSET,   4
    .equ    WSPACE_MASK,     0xFFFF

    .text
    .global _start
    .org    0x00000000

_start:
    movia       sp, LAST_RAM_WORD
    movi        r2, '\n'
    call        PrintChar
    movia       r2, MSG
    call        PrintString
_end:
    br          _end

PrintChar:
    subi        sp, sp, 8
    stw         r3, 4(sp)
    stw         r4, 0(sp)
    movia       r3, JTAG_UART_BASE
pc_loop:
    ldwio       r4, STATUS_OFFSET(r3)
    andhi       r4, r4, WSPACE_MASK
    beq         r4, r0, pc_loop
    stwio       r2, DATA_OFFSET(r3)
    ldw         r3, 4(sp)
    ldw         r4, 0(sp)
    addi        sp, sp, 8
    ret

PrintString:
    subi        sp, sp, 12
    stw         ra, 8(sp)
    stw         r3, 4(sp)
    stw         r2, 0(sp)
    mov         r3, r2
ps_loop:
    ldb         r2, 0(r3)
    beq         r2, r0, end_ps_loop
    call        PrintChar
    addi        r3, r3, 1
    br          ps_loop
end_ps_loop:
    ldw         ra, 8(sp)
    ldw         r3, 4(sp)
    ldw         r2, 0(sp)
    addi        sp, sp, 12
    ret

        .org        0x1000
MSG:    .asciz      "Hello World\n"
        .end
```

SWINBURNE UNIVERSITY OF TECHNOLOGY

Programming Languages

Why so many different options and layers of abstraction ?

Remember programming languages are all about people and problems to solve

Some programs are big and complex, and are maintained by potentially hundreds of developers (*think financial software, enterprise systems, operating systems*)

Some programs are smaller, but need to run in real-time, (think vehicle braking control, temperature control systems)

Some programs need to run on many different operating systems/hardware (think mobile apps)

Some programs need to run on hardware with limited power and/or computation (think wearables)

Some programs run on the cloud, in a completely virtualised environment

Some programs were written decades ago and simply need to be maintained

Some programs need to control custom hardware, and so need to have access to its inner workings (think device drivers!)

The point ? Context is everything, and the needs of computation are highly diverse

SWINBURNE UNIVERSITY OF TECHNOLOGY