IOS 입문 강의

제 1강 컴퓨터의 기본 원리와 IOS

1교시. 컴퓨터의 기본 원리

1. 하드웨어

1) CPU (중앙처리장치)
- 컴퓨터의 두뇌 역할 -> 데이터를 처리하는 역할
- 1,0,1,0 의 이진법 숫자의 전기신호를 cpu가 분석
- 전기신호가 오고 사라지듯이 cpu는 분석만 함 —> 처리의 결과값은 메모리로

2) Memory (주 기억장치) = 렘
- 데이터 값에 대한 보관소
- 그러나, 전기가 끊기면 날아감. 휘발성 —-> 보조기억장치에 강제로 저장해서 흔적을 남겨놓음

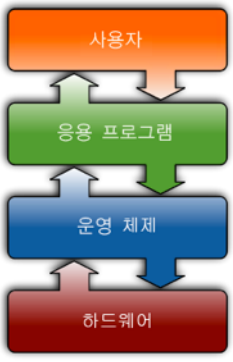
※ CPU와 Memory는 속도가 빠름

3) 보조기억장치 (SSD, HDD 등)
- 데이터 영구 저장소
- 속도가 느림

4) Mainborad
- 전체 하드웨어들을 연결함

2. 운영체제

* 운영체게 = OS
  * 오퍼레이팅 시스템 (operating system)



- 운영 체제(運營體制, 문화어: 조작체계) 또는 오퍼레이팅 시스템(영어: Operating System, OS)은 시스템 하드웨어를 관리할뿐 아니라 응용 소프트웨어를 실행하기 위하여 하드웨어 추상화 플랫폼과 공통 시스템 서비스를 제공하는 시스템 소프트웨어이다. 최근에는 가상화 기술의 발전에 힘입어 실제 하드웨어가 아닌 하이퍼바이저 위에서 실행되기도 한다.

IOS 입문 5기 화이팅! 😊

* **운영체제의 종류**
  * 윈도우
  * IOS
  * OSX
  * 리눅스
  * 안드로이드
  * 등등

* **IOS 란?**
  * **아이폰 안에 들어가는 기본 OS**
  * 아이폰 안에도 하드웨어들이 컴퓨터와 동일하게 있고
  * 하드웨어가 IOS와 통신한다.
  * IOS 위에서 응용 프로그램 (어플리케이션)이 돌아가는것

3. 응용프로그램

* 응용 소프트웨어 (application software)
  * 운영 체제 위에서 실행이 되는 소프트웨어

4. 작동 원리

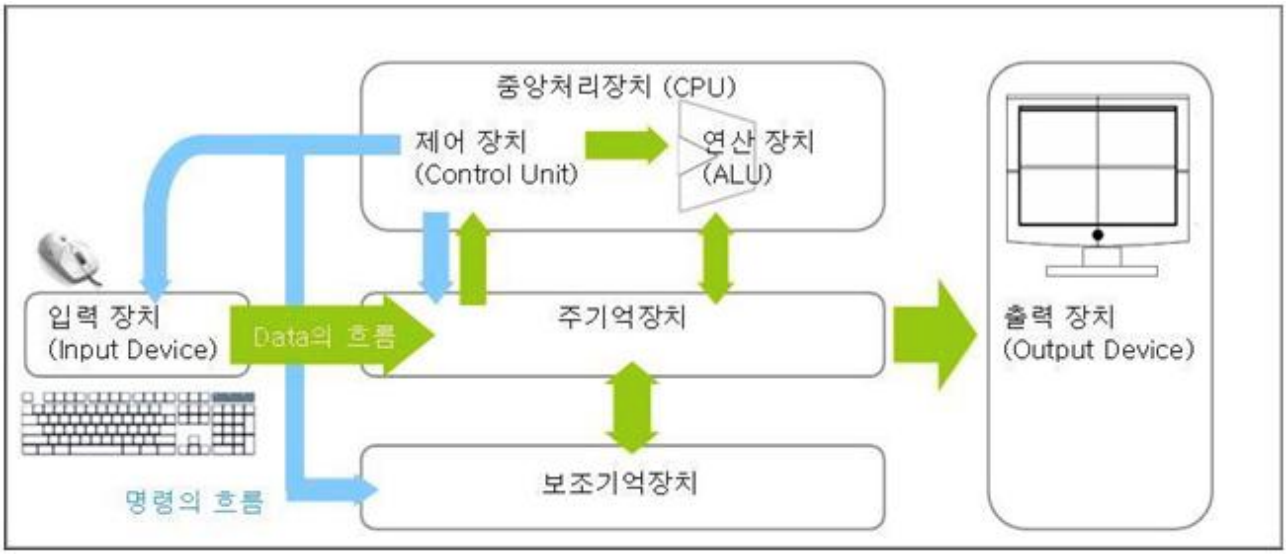
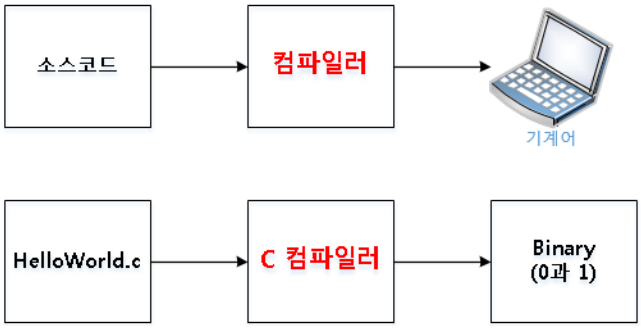


그림 출처 - http://kylog.tistory.com/11

2교시. 왜 Swift인가?

1. 컴파일언어와 스크립트 언어

* 컴파일언어



  * **컴파일**: 사람이 이해할 수 있는 언어를 기계어로 변환하는 작업
  * 고급언어 —-> 기계어(저급언어)
  * 예시) C, java, objective C ……
    * 실행속도가 빠름
    * 버그 발견이 빠름
    * 코딩이 어렵고, 미리 컴파일러를 통해 컴파일을 해야한다.

* **스크립트 언어**
  * 인터프린터 : 명령어들을 한 줄씩 읽어들여서 실행하는 프로그램
  * 예시) python, javascript, php……
    * 코딩이 쉽고, 파일로 저장된다.
    * 컴파일 언어보다는 상대적으로 실행속도가 느리다.
    * 컴파일 과정이 없기 때문에 실제로 돌려보지 않을 때에는 버그를 알아내기 어렵다.

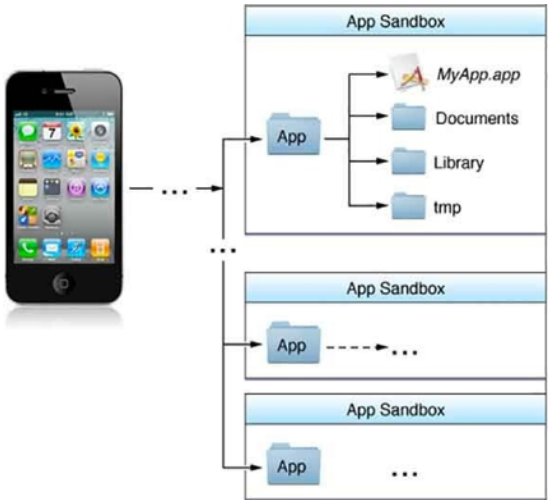
2. 응용프로그램 동작 순서

* 일반 응용프로그램 - 컴파일 언어로 프로그래밍
* 웹브라우저 - 스크립트 언어로 작성

3. Swift란?

* **컴파일 언어이지만, 문법은 스크립트 형식**
  * 스크립트 언어의 장점인 코딩이 쉽고 + 컴파일 언어이기에 속도가 빠름

4. IOS의 구조 파악하기



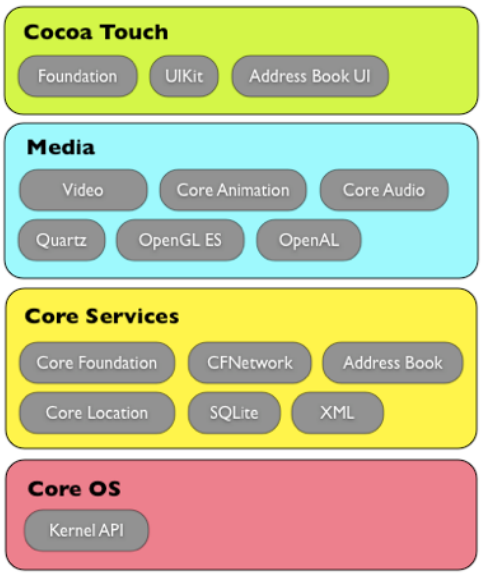
* Sandbox구조
  * 각각의 어플리케이션을 각각의 샌드박스(Sandbox)라고 함
  * 앱 안에 폴더 및 응용파일이 패키지 형식으로 묶여있음
  * 앱은 OS와만 통신 가능
    * 때문에 앱과 앱들 간에는 통신이 불가능
    * 만약 앱과 앱이 전환이 된다면, 그것은 URL로 전환하는 것
      * But, 동시에 앱을 써야하는 경우, OS가 background영역으로 이용 (ex. 음악, 간단한 네트워킹 등)

* sandbox의 특징
  * app간의 독립성 유지 = 보안 강화
    * app간의 교류, 해킹, 침투 같은 것이 불가능..
  * OS에서 모든 app 관리. = 관리가 쉬워짐 (app은 OS를 못 건드림)
    * <-> 반대로 안드로이드는 os에서도 app을 건드리고, app에서도 os를 건드릴 수 있음.
  * app들간의 정보 교환이 어렵다.

5. Framework 계층구조

* Framework
  * 프로그램을 만들기 위한 패키지 (소스 모음)
  * 개발자가 어느정도 만들어져있는 재료를 가지고, 앱을 만들 수 있도록 제공되어있다.
  * appple에서는 쉽게 UI를 표시할 수 있도록 제공.

* 계층구조



*우리가 주로 건드리는 것은 **Foundation, UIKit**

*Foundation Framework
* APP내 오브젝트를 관리하는 틀을 제공.
* 메모리에 할당, 반환 등 기본 규칙이 정의되어있다.

*UIKit framework
* IOS 인터페이스 관련 오브젝트 제공

*우리는 계층구조들 중에 Cocoa Touch를 건드리게 될 것

6. 앱 실행 순서

* IOS에서는 프로그램들이 **Application Delegate**로부터 시작
* Application Delegate : OS에서 명령을 받는 역할
* 지금은 스토리보드도 같이 불리도록 자동으로 되어있다.
  (xcode화면 보면 알 수 있음)

7. 화면구조

* UIWindow에서 루트뷰 컨트롤러를 통해 뷰 컨트롤러로 가고, 그것이 뷰를 띄워서 우리가 화면을 본다.?
* 윈도우는 우리가 보는 핸드폰 화면이라고 생각하면 된다. 액자같은 이 윈도우를 통해 일정 화면을 보고, 뷰가 실제 화면크기보다 크면 스크롤해서 보는 구조.