프로그래밍(2)

Programming(2)

(0002027002)

Introduction to Programming

2024년 2학기

정보기술대학
정보통신공학과
김 영 필
(ypkim@inu.ac.kr)

INU 인천대학교
INCHEON NATIONAL UNIVERSITY

Programming이란 무엇인가?

- **Program**을 만드는 것
- **누가?**
  - **인간 개발자**
  - 혹은, …

MAKE BUILD

COMPILE COMPOSE

(Computer) Program은 무엇인가?

- (명령어) 들의 집합

- 어디에 있는가?
  - 저장장치 (e.g. HDD, SSD)?
  - 메인 메모리 (e.g. RAM)?

stored image

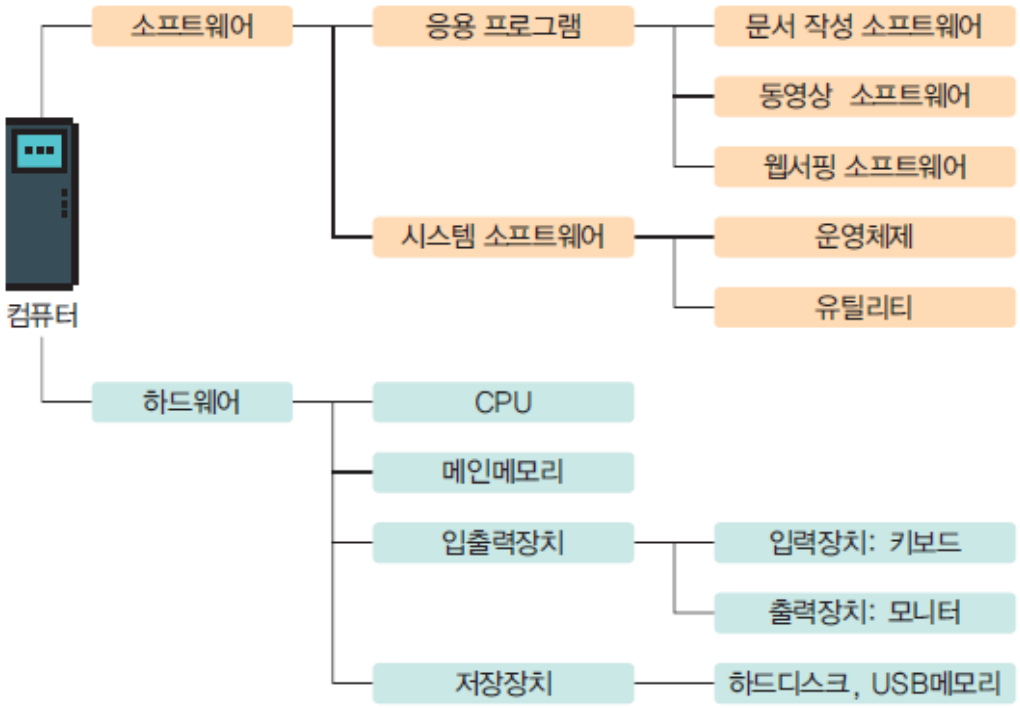
loaded image

```
int main() {
        int a, b, c;
        c = a + b;
        return c;
}
```

```
112d:   55                      push   %rbp
112e:   48 89 e5                mov    %rsp,%rbp
1131:   8b 55 f4                mov    -0xc(%rbp),%edx
1134:   8b 45 f8                mov    -0x8(%rbp),%eax
1137:   01 d0                   add    %edx,%eax
1139:   89 45 fc                mov    %eax,-0x4(%rbp)
113c:   8b 45 fc                mov    -0x4(%rbp),%eax
113f:   5d                      pop    %rbp
1140:   c3                      retq
```

Program은 어떻게 실행 되는가?

- 전제조건
  - 일단 **컴퓨터 하드웨어(H/W)**가 있어야 한다!
  - 또한 **소프트웨어(S/W)**가 있어야 한다!



소프트웨어의 종류

- 응용프로그램
  - 웹브라우저, 채팅프로그램, 동영상 재생기, 화상회의 프로그램, 게임, ...
- 시스템 소프트웨어
  - 운영체제, 컴파일러, 데이터베이스관리시스템, 하이퍼바이저, 펌웨어

하드웨어의 종류

- CPU
  - Intel, AMD, ARM, Alpha, MIPS… X86_64, AMD64, ARMv8, AArch64
- Memory
  - RAM (DDR..)
- Chipset
  - Northbridge,
    Southbridge,
    others
- I/O bus
  - IDE, SATA, AMBA,..
- Peripheral devices
  - HDD, SSD, CD-ROM, Printers, monitors, keyboard, mouse, tablet, Wired or wireless NIC,

하드웨어와 소프트웨어의 관계

- 주어진 **컴퓨터 하드웨어**를 통해 **데이터**를 어떻게 처리하는지가 중요
- 그 핵심 역할은 소프트웨어 가 담당
- 요리로 치면, 다양한 요리를 만들어 내는 **레시피**의 역할

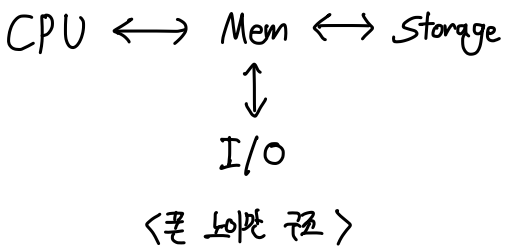
프로그램 실행과 관련된 하드웨어의 특징

- CPU
  - 컴퓨터에서 주방장 역할을 담당하는 부분
  - 가장 중요한 일은 메모리에 있는 프로그램 **을 실행하는 것**
- 메모리
  - 컴퓨터의 유일한 **작업 공간**
  - 요리에 필요한 재료들은 창고에 보관되지만, 재료의 손질과 조리 작업은 작업대에서
  - **휘발성** ⟷ Non-Violate , DRAM ⟷ NVRAM
- 입출력 장치
  - 보조 저장장치
    - **오래 보관**해야 하거나 (**비휘발성**) **용량이 큰 데이터** (HDD, SSD, USB)
  - 보조 처리장치
    - CPU **의 부담을 줄여 주며 데이터 처리**가 가능한 장치들 (GPU, DMA) 또는 네트워크(NIC)나 입출력 (Mouse, Tablet, Printer, HMD) 등의 다양한 목적의 장치들

Direct Memory Access

메인메모리와 프로그램의 관계

- CPU는 메인 메모리 에서 **명령어**를 가져와 수행
- 새로운 명령어가 필요하면?
  - 메인메모리의 내용을 업데이트! ➔ **폰 노이만 구조**로 인한 컴퓨터 활용의 다양성



다시 프로그래밍으로!

- 프로그래밍(programming) 이란 프로그램을 만드는 작업
- 어떻게 만들어야 하나?
  - 컴퓨터에게 작업을 시키려면 구체적으로 설명하여 기록된 형태로 기술
  - 컴퓨터에게 알려줄 작업을 하나로 모으면 컴퓨터 프로그램 (program) 이 됨
  - 프로그램에는 컴퓨터에게 지시할 명령들과 명령의 결과를 받는 방법들, 데이터가 모여 있음

컴퓨터와 인간 사이에서 소통의 수단의 필요

프로그래밍 언어

- 저급 언어 (low-level language)
  - 기계어(machine language)
  - 어셈블리어(assembly language)
- 고급 언어 (high-level language)
  - C, **Java**, Python, C++, C#, R, Go, Rust, ...
    - 절차적(Procedual), 객체지향(Object-oriented), 함수형(Functional) 등

참고: Programming paradigms (1)

- 명령형 (Imperative) 방식
  - 프로그램 == 내부 상태를 변경시키는 문장들로 구성
  - 특징: 직접 할당문, 공통 자료 구조들, 전역 변수
  - 예) **C**, C++, Java, Python

- 구조적 (Structured) 방식
  - 논리적인 프로그램 구조를 강화한 명령형 방식
  - 특징: 블록, 인텐트, 제한된 goto 사용
  - 예) **C**, C++, Java, Python

- 절차적 (Procedural) 방식
  - 구조적 방식에 모듈화된 프로시저 호출을 추가한 방식
  - 특징: 지역변수, 반복 및 선택구조, 모듈화
  - 예) **C**, C++, Python

참고: Programming paradigms (2)

- 함수형 (Functional) 방식
  - 프로그램 연산을 마치 수학 함수를 평가하듯 처리. 프로그램 상태를 유지하지 않음
  - 특징: 람다 계산, 재귀, 공식화
  - 예) C++, C#, Java ver. 8+, Python, JavaScript

f(x, y) = a·x + b

(x, y) -> { return a*x + b; }

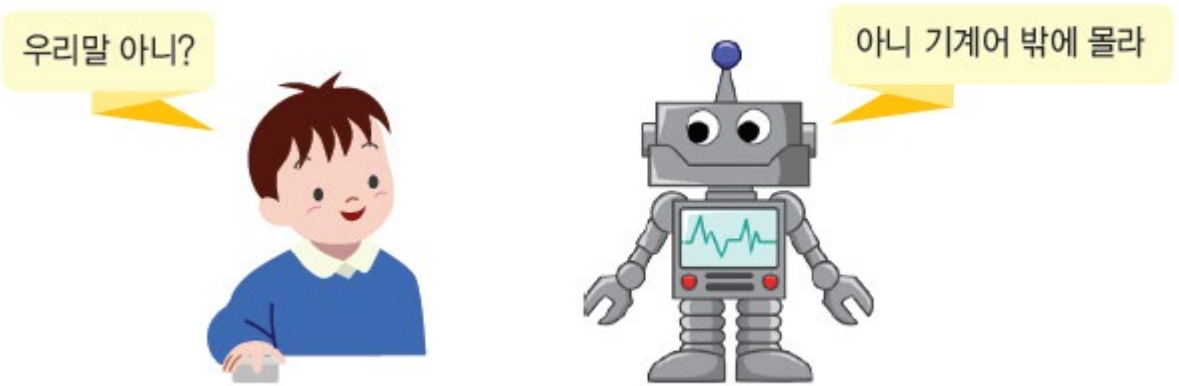
- 객체지향 (Object-oriented) 방식
  - 객체 단위로 데이터와 그 사전 정의된 연산들로 묶어 처리
  - 특징: 객체, 메소드, 정보 은닉, 상속 등.
  - 예) C++, C#, Java, Python, JavaScript

Method는 Procedure와 다르게 제약이 부가됨

프로그래밍 언어의 필요성

Q) 그렇다면 인간이 기계어를 사용하면 어떤가?

- 기계어를 사용할 수는 있으나 이진수로 프로그램을 작성하여야 하기 때문에 아주 불편하다.
- 프로그래밍 언어는 자연어와 기계어 중간쯤에 위치
- 컴파일러가 프로그래밍 언어를 기계어로 통역



컴파일러 (Compiler)

보통 소스 파일과 컴파일 된 파일의 확장자는 다르게 하여 구분
- 자바 : hello**.java** -> hello**.class**
- C : hello**.c** -> hello**.obj** -> hello**.exe**
- C++ : **.cpp** -> **.obj** -> **.exe**

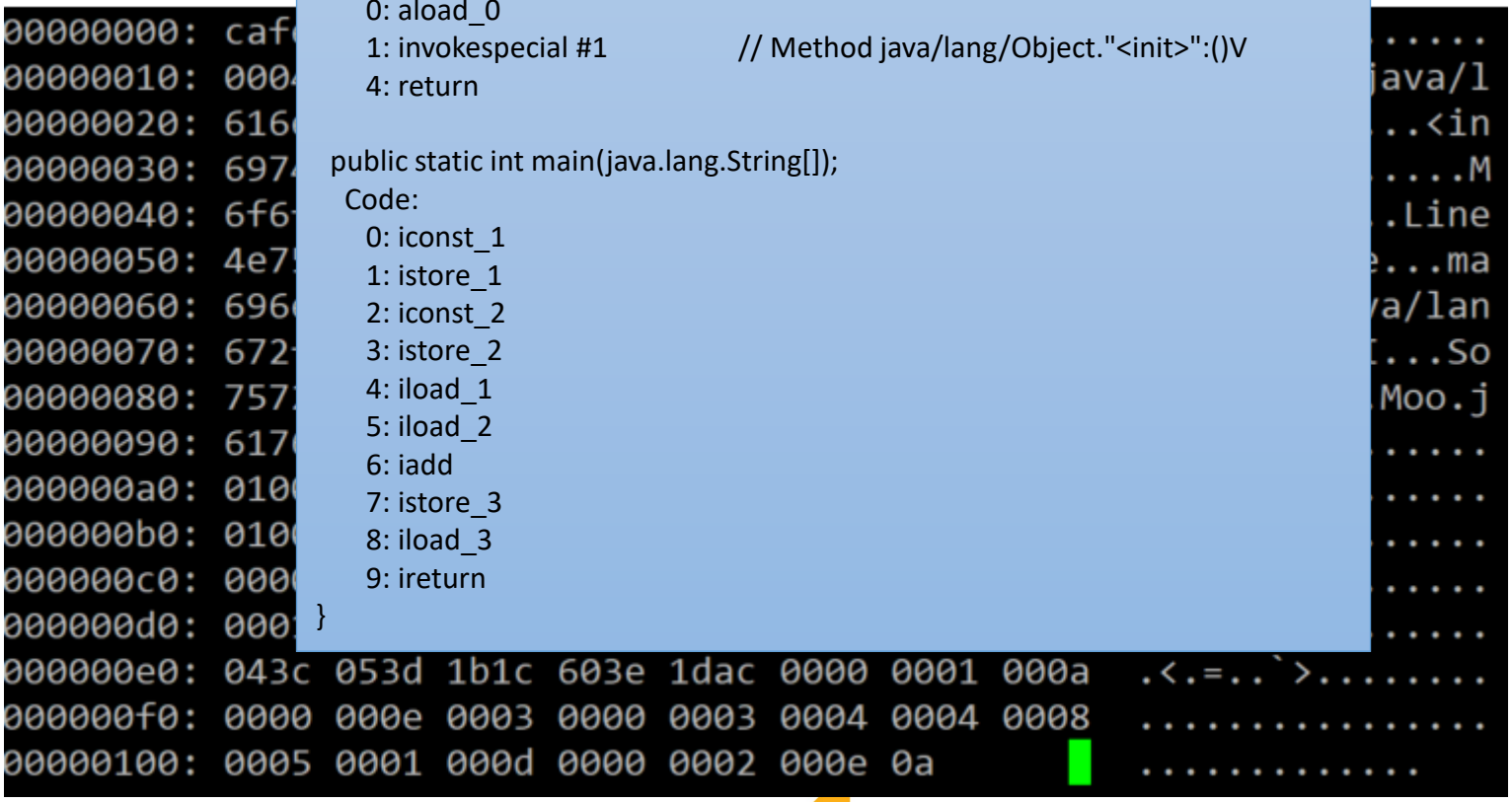
- 컴퓨터는 기계어만 인식
- 고급 언어를 기계어 로 번역하는 과정이 필요 (Compile)
- 컴파일러는 소스코드를 번역하여 기계어로 이루어진 실행파일을 생성
- C언어, C++, Java, Rust와 같은 많은 고급 언어가 컴파일러 방식을 사용

참고) Java 컴파일 결과물

```
public class Moo {
        public static int main (String [] argv) {
                int a = 1, b = 2, c;
                c = a + b;
```

```
Compiled from "Moo.java"
public class Moo {
  public Moo();
    Code:
      0: aload_0
      1: invokespecial #1                  // Method java/lang/Object."<init>":()V
      4: return

  public static int main(java.lang.String[]);
    Code:
      0: iconst_1
      1: istore_1
      2: iconst_2
      3: istore_2
      4: iload_1
      5: iload_2
      6: iadd
      7: istore_3
      8: iload_3
      9: ireturn
}
```

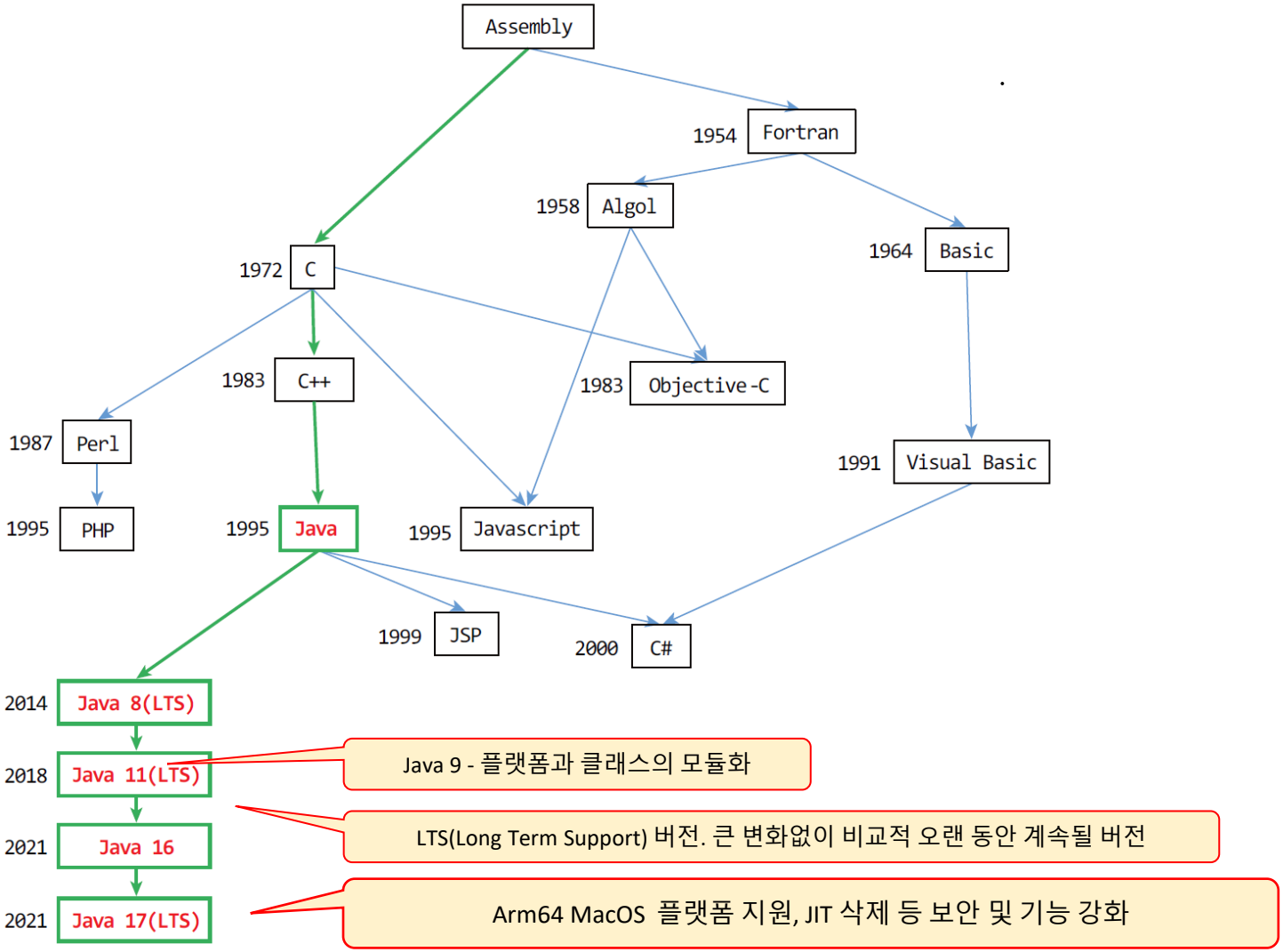


인터프리터(Interpreter)

인터프리터는 소스 파일만 있으면 끝
- Python : hello**.py**
- Bash : script**.sh**
- JavaScript: foo**.js**

- **소스코드**를 한 번에 한 문장씩 번역하여 실행하는 방식
- 인터프리터 방식은 한 줄씩 실행하기 때문에 전체 재료를 파악하기 어려움
- 또한 오류 검증이나, 중복 제거 등의 최적화가 어려움
- 컴파일러 방식이 성능면에서 우수

잘 알려진 프로그래밍 언어들의 출현 순서



학습활동 (LMS 과제로 제출)

- 과거 수강했던 프로그래밍 기초 수업에서 **재미있었고 흥미로웠던** 주제나 개념이 있었다면 최대 **3개만 이유와 함께** 적어보자.
- 과거 수강했던 프로그래밍 기초 수업에서 **어려웠던 내용이나 잘 이해가 되지 않았던** 주제나 개념이 있었다면 최대 **3개만 이유와 함께** 적어보자.
- 프로그래밍을 배워서 가장 이루고자 하는 **목표나 결과물**이 있다면 무엇인지 자유롭게 글로 설명해 보자. (분량 자유)