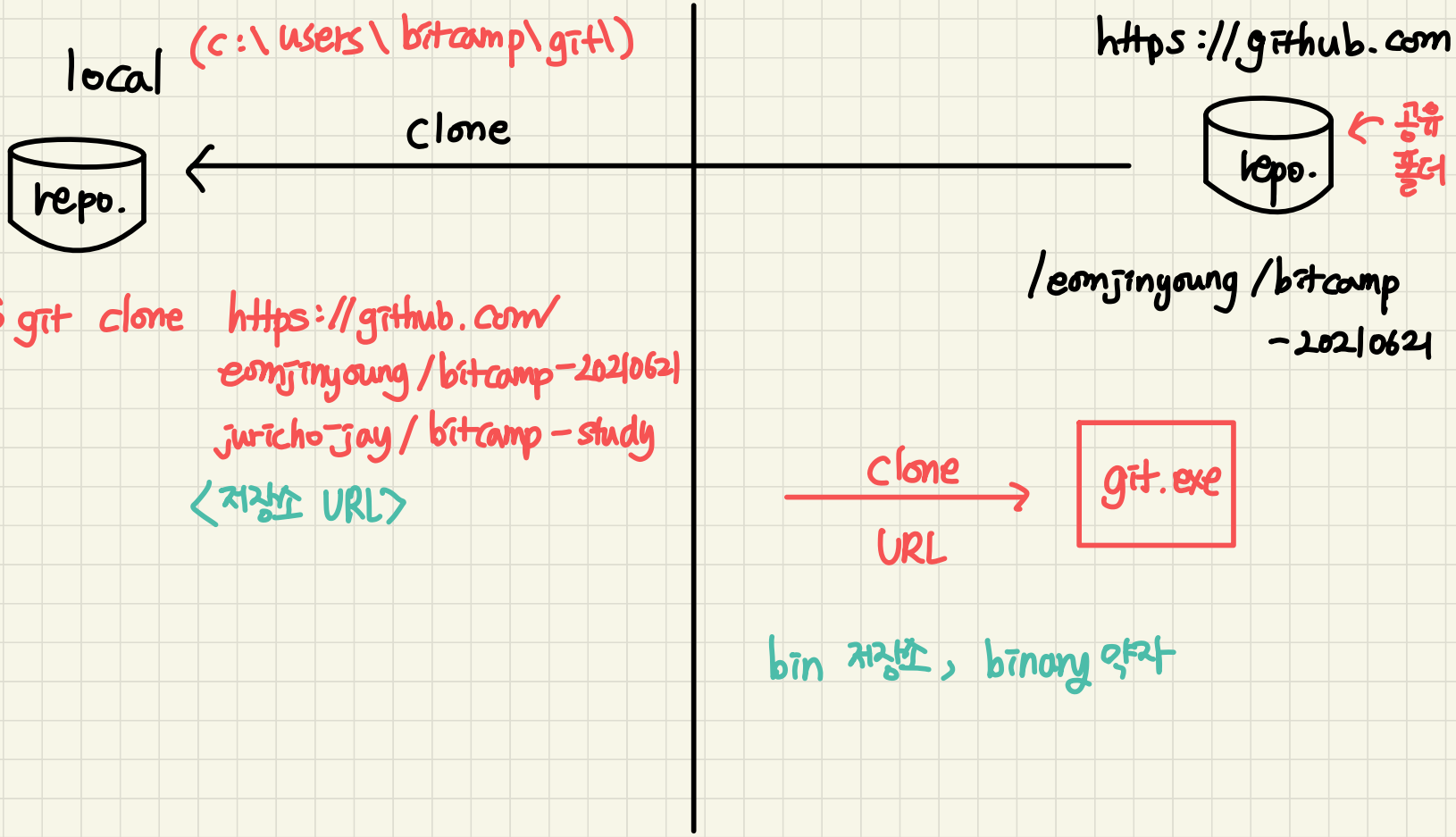
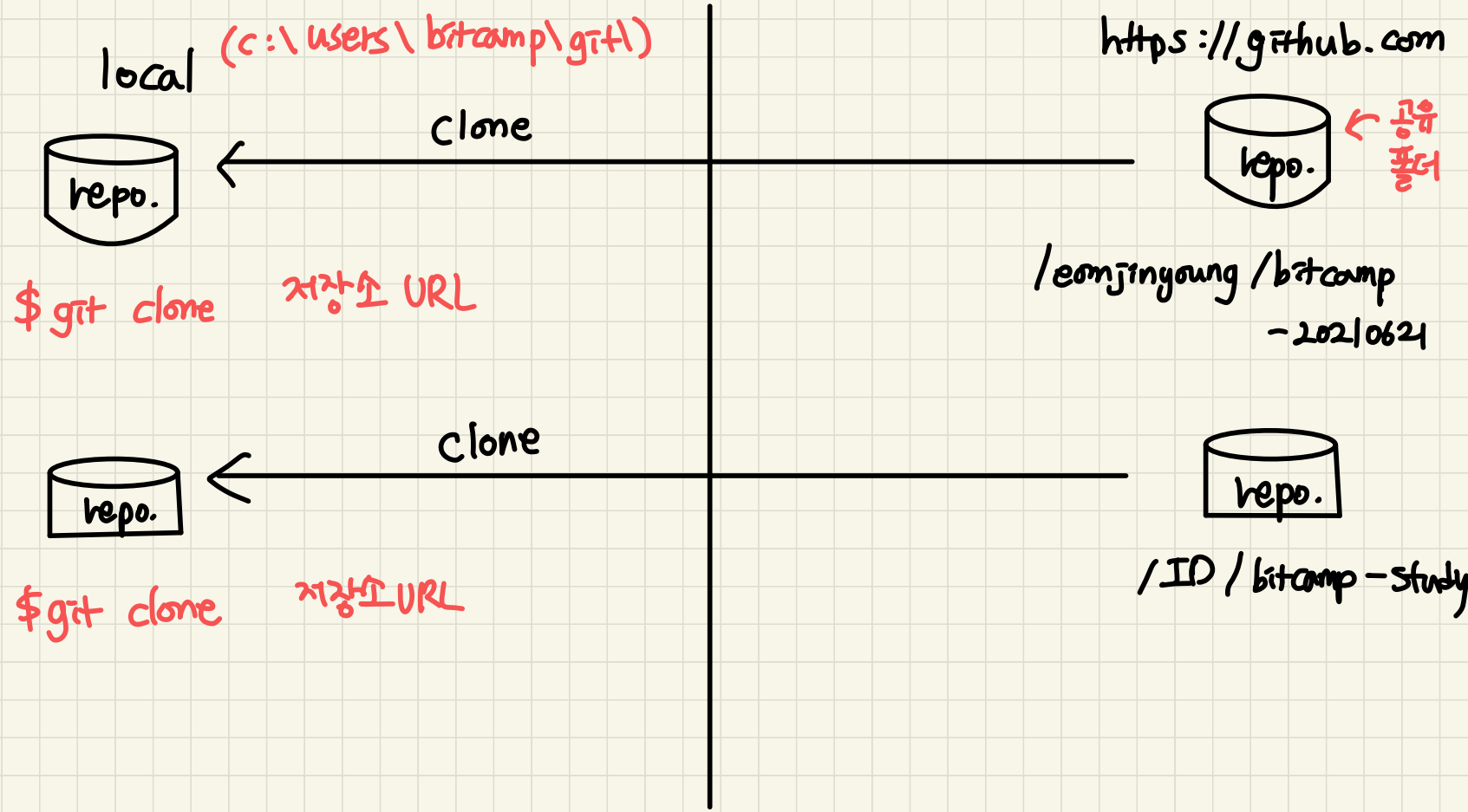
* git 원격저장소를 local로 복제하기



* git 원격저장소를 local로 복제하기



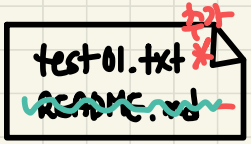
* git 로컬 저장소에 파일 백업하기

[Working Directory] 추가 / 변경 / 삭제 된

README.md
test01.txt

좋은 커밋 메시지란? 기가지 규칙?!
↳ depends on each firm

① 백업 명단 작성 ⟶ ② 백업 수행



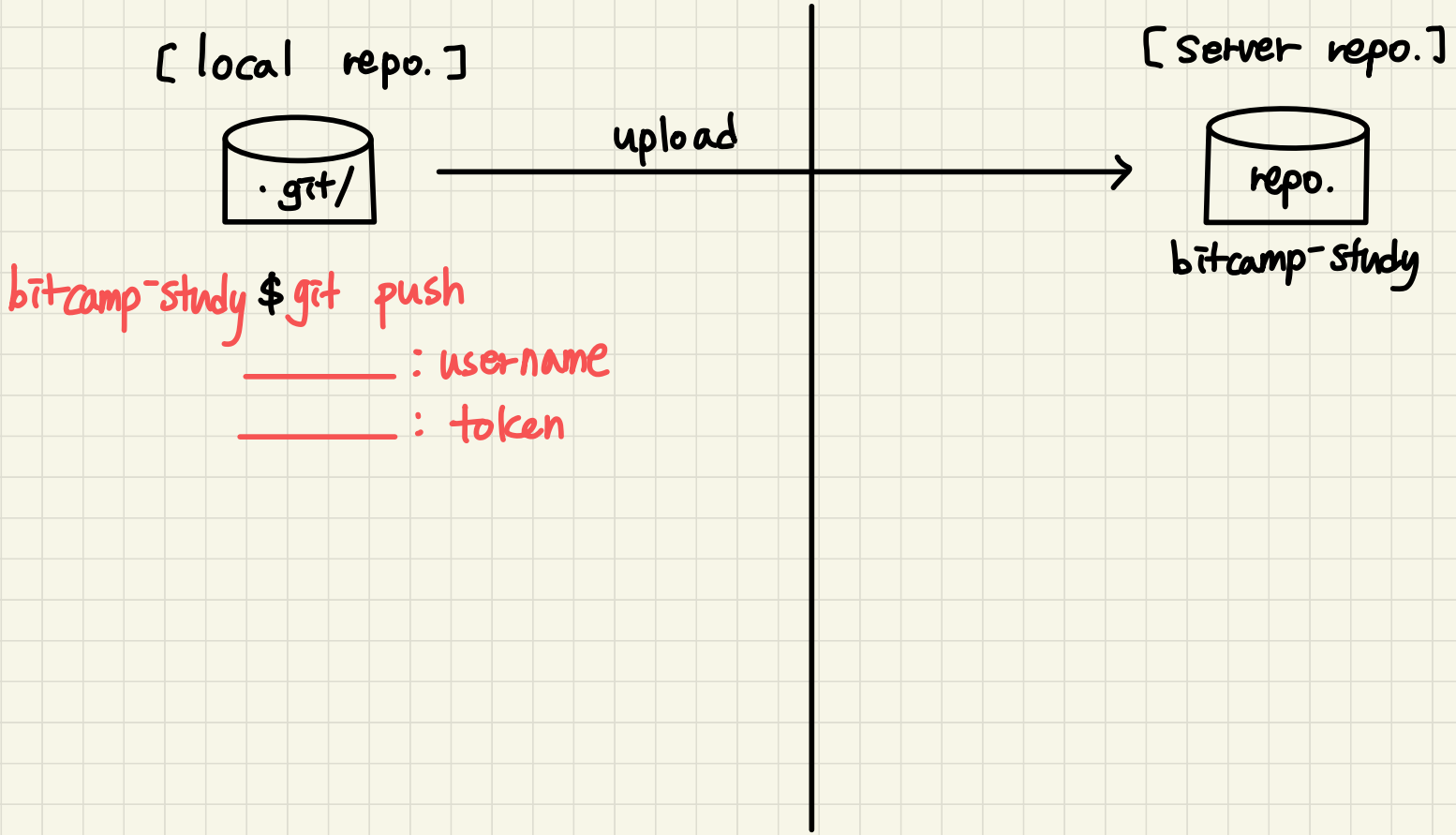
message

bitcamp-study $ git add .

변경된 파일을 등록
↳ 추가/삭제/수정

$ git commit -m "백업 내용 + 이유"

* git 로컬 저장소에 백업된 파일을 서버 저장소에 업로드



bitcamp-study $ git push

_____ : username

_____ : token

* 백업 100번하고 push 1번해도

100번 백업 기록은 모두 기록됨 (버전 관리 시스템)

* 서버 저장소에서 변경된 파일 가져오기

[local 저장소]

[server 저장소]

$ git pull

✗ 수업도구준비 (계속)

③ VS Code (편집기) 설치

↳ 소스 파일 편집

- 명령어를 작성한 파일
- 실행 파일을 만들 때 사용

↳ HTML/CSS/JavaScript 파일 편집

↳ 기타 설정 파일 편집

↳ Microsoft에서 다운로드

Pref → Settings

* font-size

Ajqk

높이를 의미

④ C/C++ 컴파일러 설치

↳ Window gcc 설치

⇒ MinGW

⇒ MinGW-w64 posix seh

→ C:\tools 에 압축 풀기

tools/mingw64/bin

↳ gcc 컴파일러가 설치된 폴더의 경로를 OS 환경 변수 PATH에 등록

⇓

gcc 컴파일러를 실행할 때 경로를 적을 필요가 없다.

↳ 폴더 등록

C:\tools\mingw64\bin\gcc --version

PAST

컴퓨터

↑ 제어

S/W ⇒ OS

사용자 = 개발자

A → OS

B → OS'

C → OS''

compiler 필요...

PRESENT

개발자 ✓

명령어 번역

실행

A ← DOS ← 실행파일 → 복사

A ← DOS → 복사

A ← DOS → 복사

UNIX → 프로그래밍 할 줄 아는 사람을 위한 것이었기에)...

(리눅스, 맥)

compiler 내장

* 수업 도구 설치 (계속)

⑤ node.js 설치

↳ javascript로 작성한 명령어를 실행

↳ nodejs.org 에서 다운로드

mac OS → $ brew install node

amd 64
Intel CPU

aarch 64
ARM CPU

(LTS)
Long-term service

windows는 ARM CPU가 없음

⑥ JDK 설치

↳ Java로 작성한 명령어를 bytecode로 번역

↓

가상의 기계어

↓

JVM이 실행

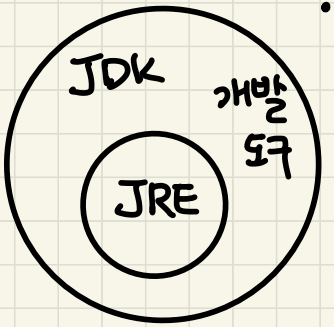
Java Virtual Machine

↳ graalvm.org 에서 다운로드

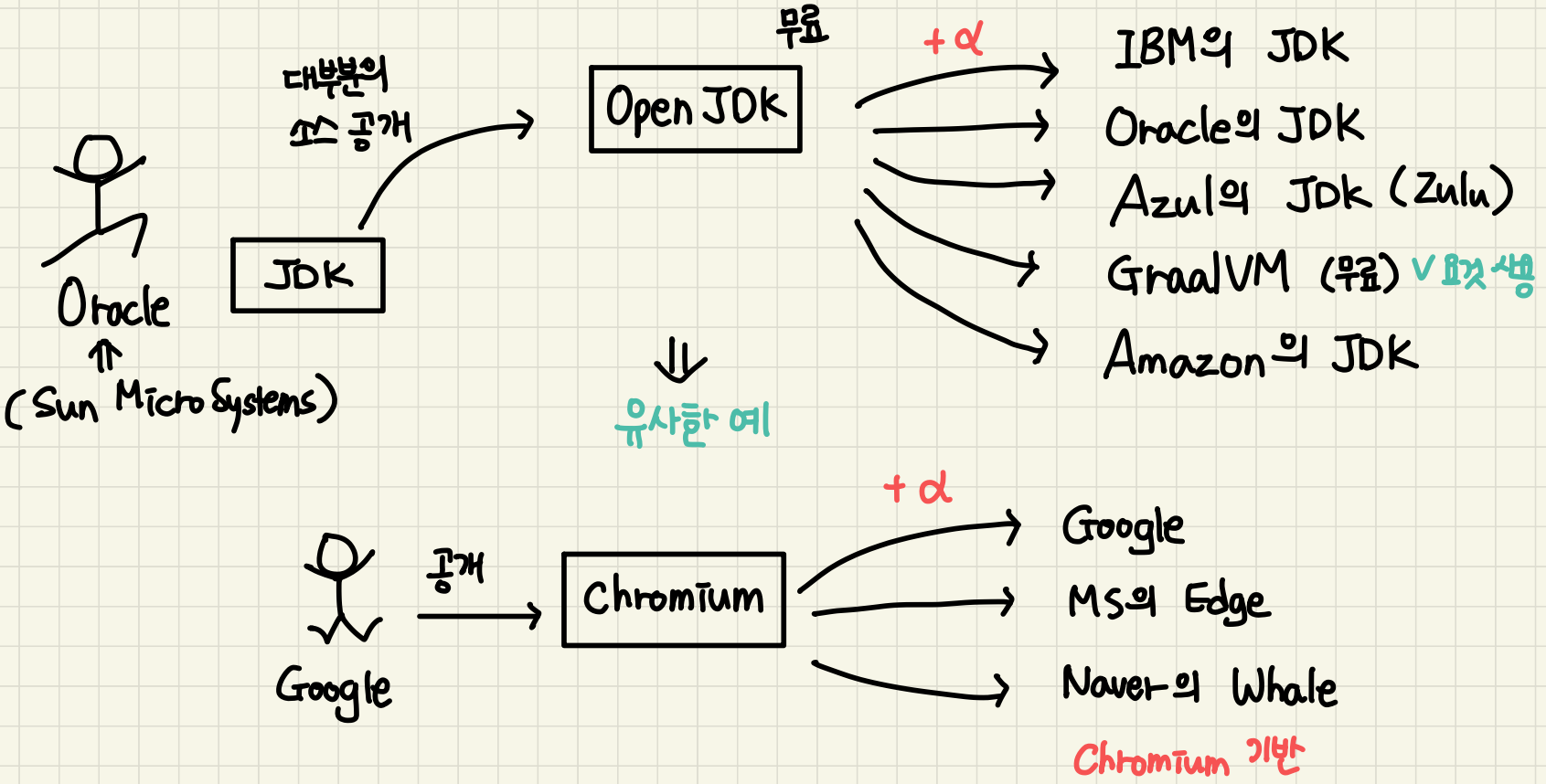
* JDK

제품분류 : Java SE (Standard Edition)

- JRE (Java Runtime Environment)
  - ↳ 자바 프로그램을 사용할 때 사용
  - ↳ bytecode 인터프리터 = JVM (Java Virtual Machine)
- JDK (Java Development kit) ← 자바 개발 도구
  - ↳ JRE + 개발도구
    - ↳ 컴파일러, 프로파일러, 디버거, 자바문서 생성기 등
  - ↳ 자바 개발자가 설치



* Open JDK


대부분의
소스 공개
무료
+α
Oracle
(Sun Micro Systems)
JDK
Open JDK
IBM의 JDK
Oracle의 JDK
Azul의 JDK (Zulu)
GraalVM (무료) V8컴파일
Amazon의 JDK
유사한 예
+α
공개
Google
Chromium
Google
MS의 Edge
Naver의 Whale
Chromium 기반

* 프로그래밍 - 프로그램 만들고 실행하기 <1

00000010 / 2

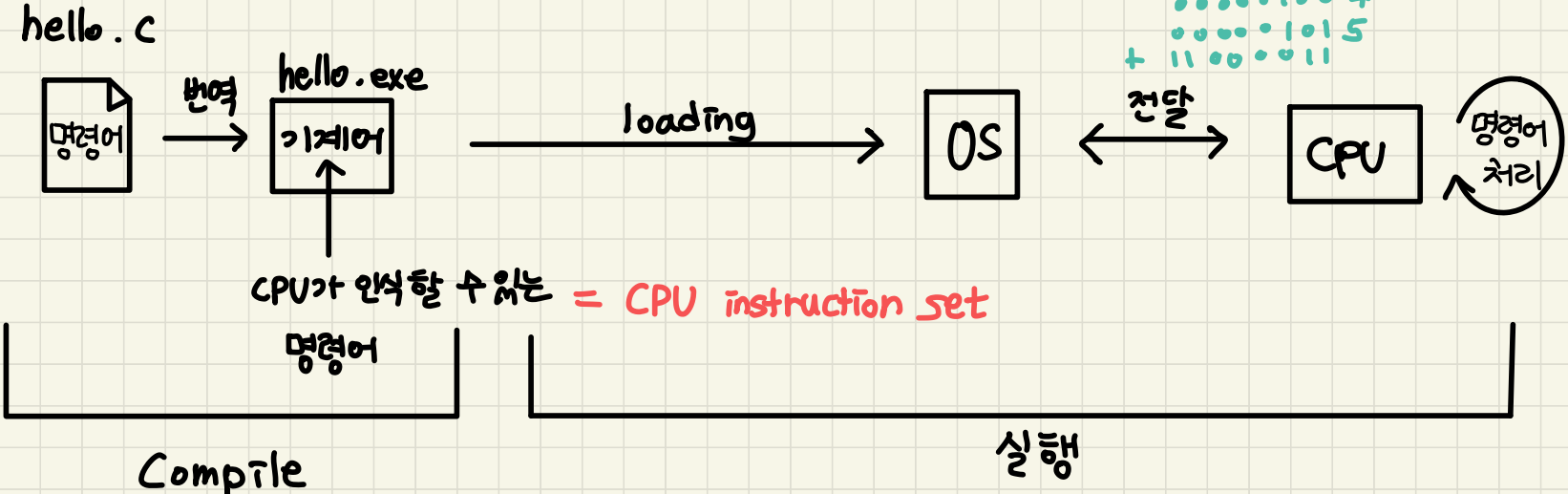
1→4
2→5
3→+ (규칙)

4?
+
5

CPU → 메모리필요

0 0 0 0 1 0 0 4
0 0 0 0 1 0 1 5
+ 1 1 0 0 0 0 1 1

① 컴파일 방식



ADD → C6 → 11000110

Instruction set
정해놓은 규칙

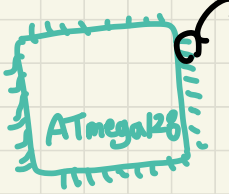
Intel CPU Instruction Set → AMD?
호환?!

* 아두이노

명령어 Instruction

0 – Low Voltage

1 – High Voltage



전기가 왔다갔다~

+, - 로 처리

10101011 8 bit CPU

1010111110001001 16 bit CPU

x2 32 bit CPU

x2 64 bit CPU

명령어는 아님

Assembly 언어!

↓

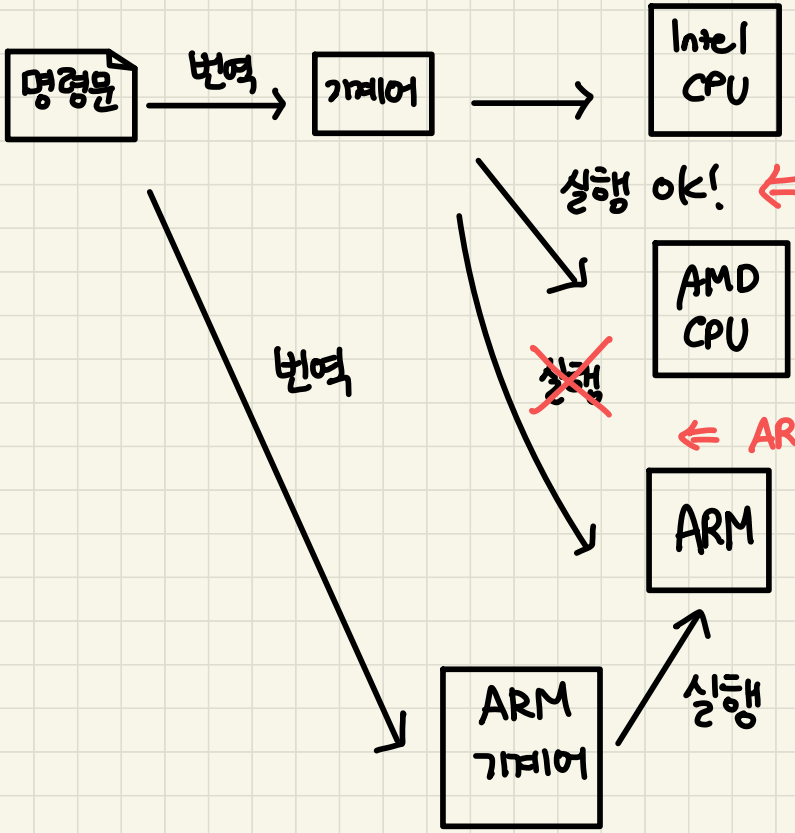
Assembly compiler 필요

AVR Instruction Set 생쌩매뉴얼

어떤 규칙인지!

✱ 기계어와 CPU

CPU가 다르면 핀에 흐르는 규칙이 다름
명령문이 다름



⇐ AMD CPU Intel CPU 명령 규칙에 따라 동작하도록 설계되었다.

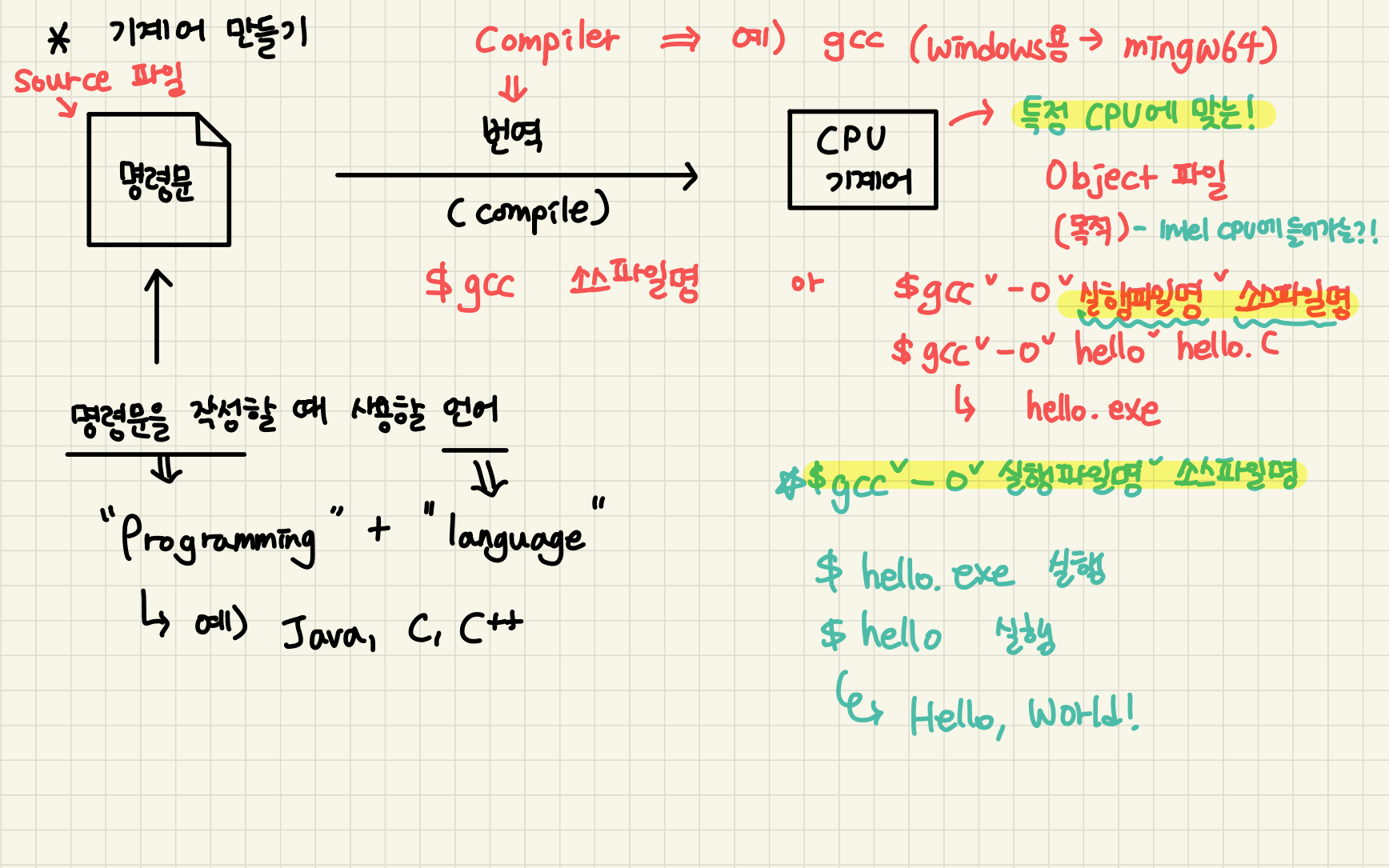
⇓

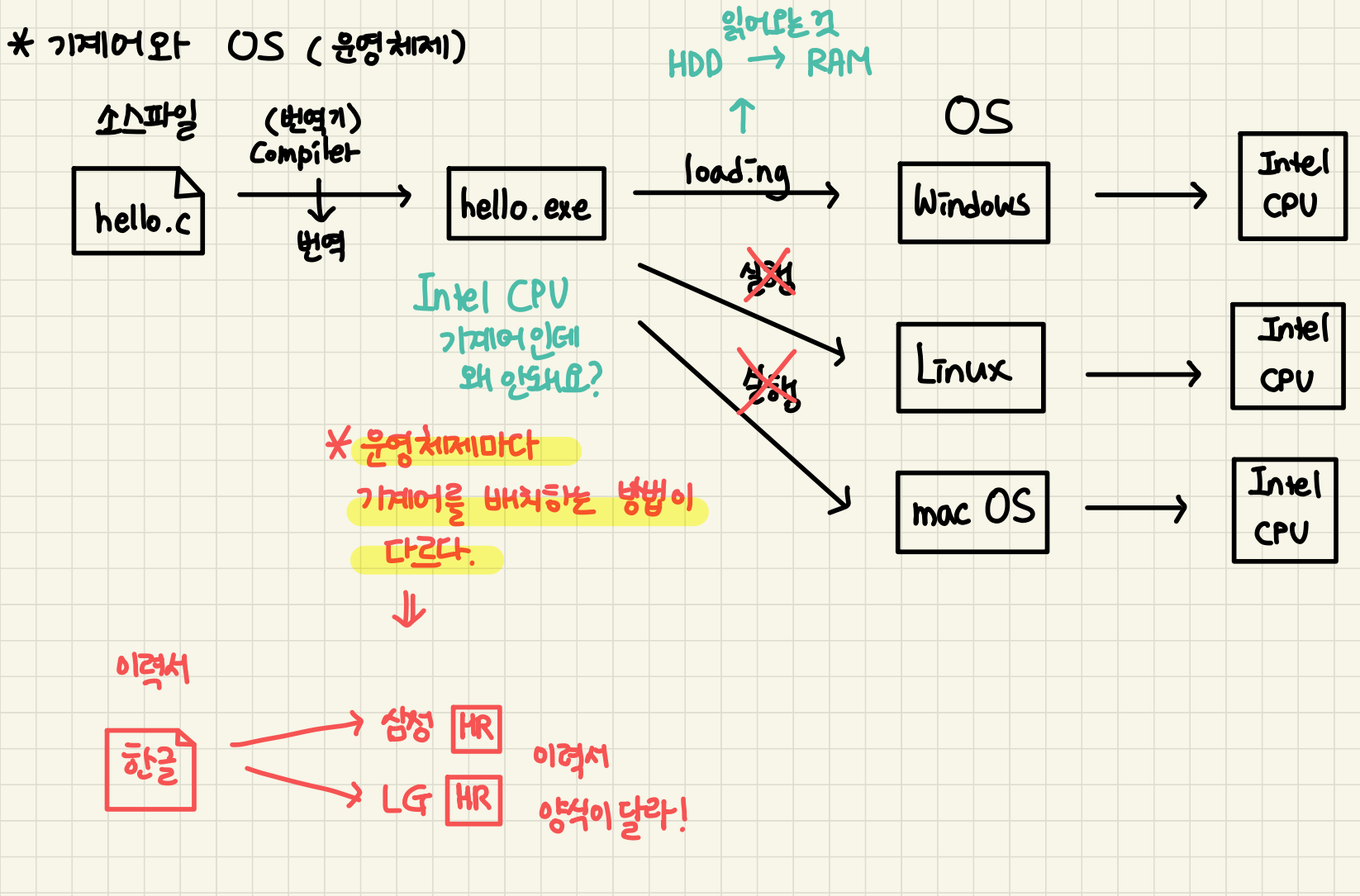
"Intel CPU compatible" (호환)

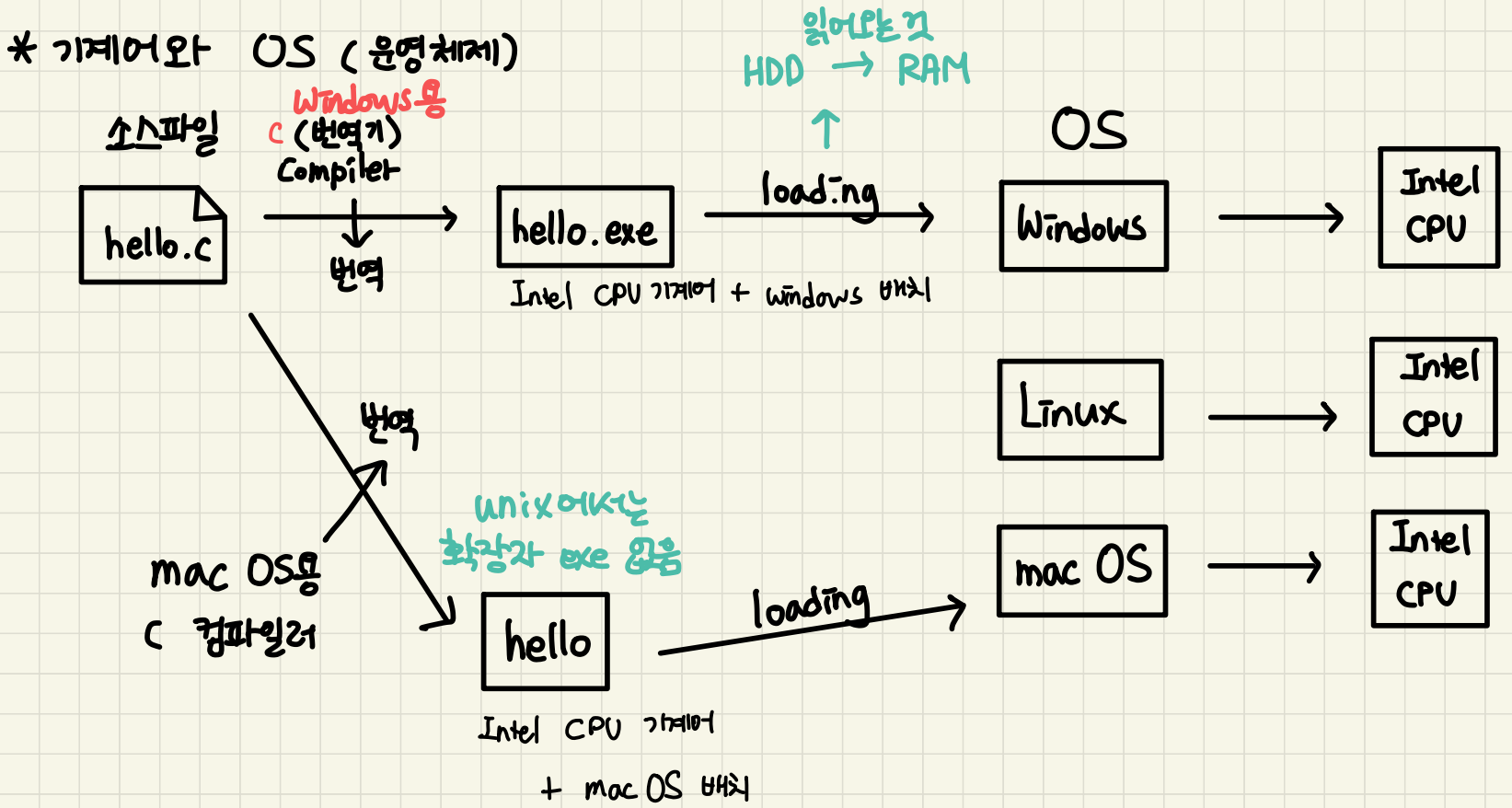
⇐ ARM CPU는 Intel CPU와 규칙이 다르다.

번역을 해주는 소프트웨어

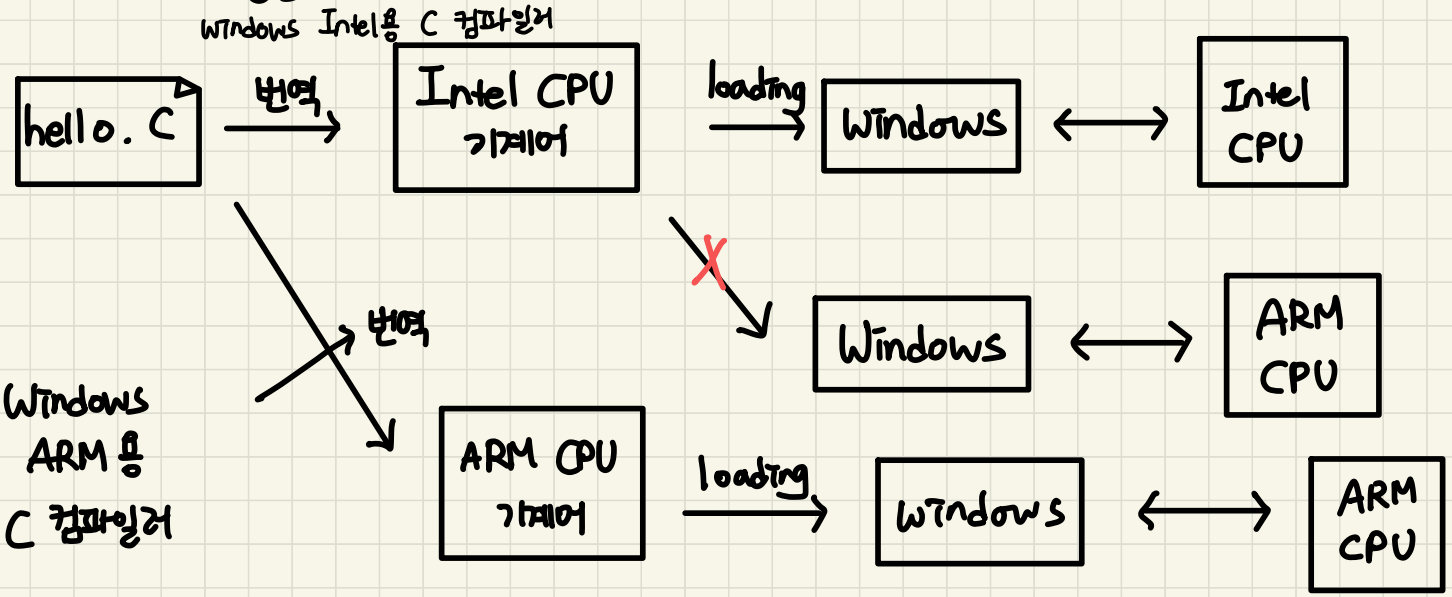
→ Compiler

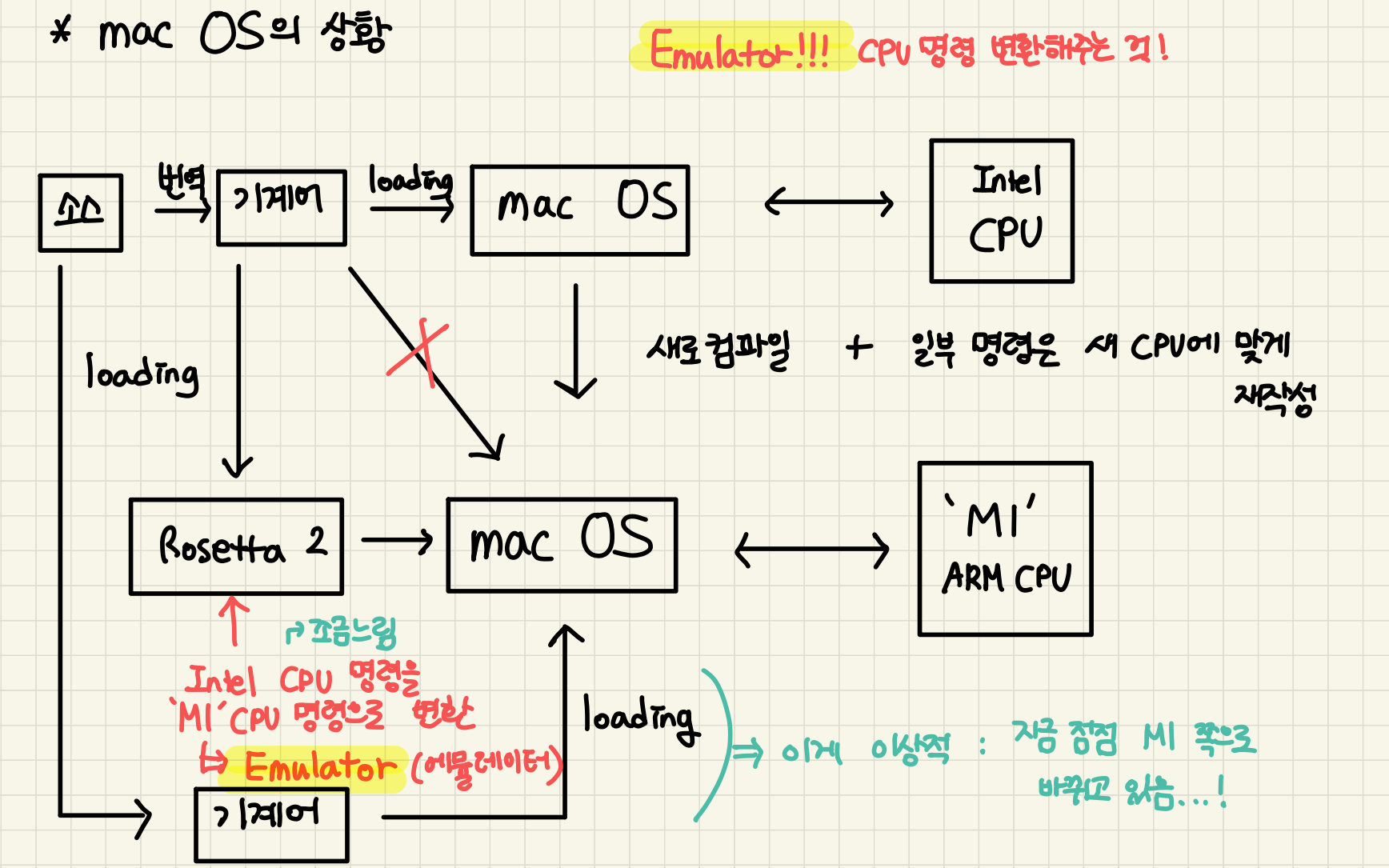

* 기계어 만들기
Source 파일
명령문
번역
(compile)
Compiler ⇒ 예) gcc (windows용 → mingw64)
CPU 기계어
특정 CPU에 맞는!
Object 파일
(목적) - Intel CPU에 들어가는?!
$gcc 소스파일명
아 $gcc -o 실행파일명 소스파일명
$gcc -o hello hello.c
↳ hello.exe
*$gcc -o 실행파일명 소스파일명
명령문을 작성할 때 사용할 언어
"Programming" + "language"
↳ 예) Java, C, C++
$ hello.exe 실행
$ hello 실행
↳ Hello, World!


* 기계어와 OS (운영체제)
소스파일
hello.c
(번역기)
Compiler
번역
hello.exe
읽어들임
HDD → RAM
loading
OS
Windows
Intel CPU
실행
Intel CPU
기계어인데
왜 안돼요?
Linux
Intel CPU
실행
mac OS
Intel CPU
* 운영체제마다
기계어를 배치하는 방법이
다르다.
이력서
한글
삼성 HR
LG HR
이력서
양식이 달라!


* 기계어와 OS (운영체제)
소스파일
Windows용
C (번역기)
Compiler
hello.c
번역
hello.exe
Intel CPU 기계어 + Windows 배치
읽어오는것
HDD → RAM
loading
OS
Windows
Intel CPU
Linux
Intel CPU
mac OS
Intel CPU
번역
mac OS용
C 컴파일러
unix에서는
확장자 exe 없음
hello
loading
Intel CPU 기계어
+ mac OS 배치

* 기계어와 OS


Windows Intel용 C 컴파일러
hello.C
번역
Intel CPU 기계어
loading
Windows
Intel CPU
번역
Windows ARM용 C 컴파일러
Windows
ARM CPU
ARM CPU 기계어
loading
Windows
ARM CPU


* mac OS의 상황
Emulator!!! CPU 명령 변환해주는 것!
소스
번역
기계어
loading
mac OS
Intel CPU
loading
새로 컴파일 + 일부 명령은 새 CPU에 맞게 재작성
Rosetta 2
mac OS
'M1' ARM CPU
↱조금느림
Intel CPU 명령을 'M1' CPU 명령으로 변환
⇔ Emulator (에뮬레이터)
loading
기계어
⇒ 이게 이상적 : 지금 점점 M1 쪽으로 바뀌고 있음...!

② 인터프리터 방식

ex) JavaScript, PHP, Phython 등

↳ 소스 파일의 명령을 도우미 프로그램을 통해 바로 실행

통역기 = interpreter

$ node 소스파일명

$ node hello.js

excel

* 기계어가 아니라서 도우미 프로그램이 필요함