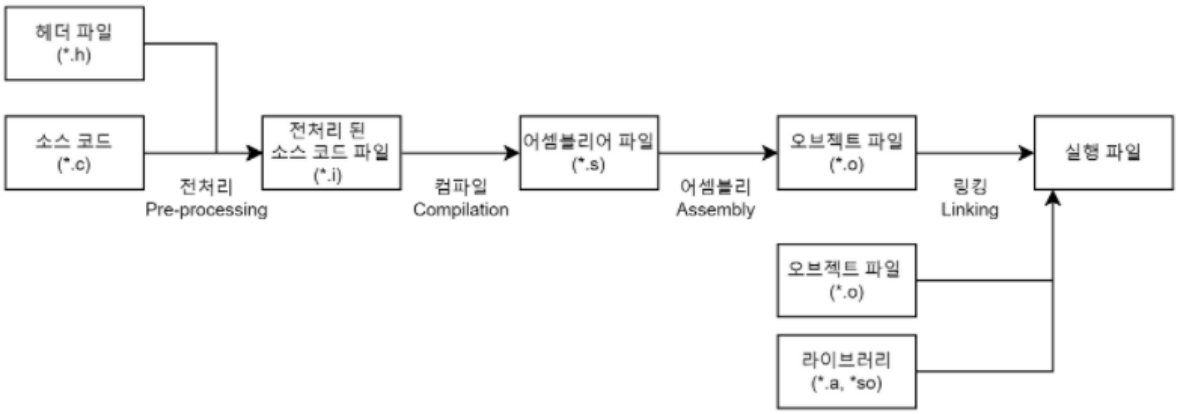
임베디드소프트웨어2 실습2

Overview

컴파일러(Compiler)

개발자가 작성한 소스 파일(프로그래밍 언어)을 컴퓨터가 해석 할 수 있도록 컴퓨터 언어로 변환시켜주는 역할을 한다.

컴파일 과정



컴파일 과정은 **4가지 단계(전처리 과정-컴파일 과정-어셈블리 과정-링킹 과정)**로 나누어 진다. 이 4가지 단계를 묶어서 컴파일 과정 또는 빌드 과정이라고 부르기도 한다.

1. 전처리(Pre-processing): 전처리기(Preprocessor)를 통해 **소스코드 파일(*.c)**을 **전처리된 파일(*.i)**으로 변환

2. 컴파일(Compilation): 컴파일러(Compiler)를 통해 **전처리된 파일(*.i)**을 **어셈블리어 파일(*.s)**로 변환

3. 어셈블리(Assembly): 어셈블러(Assembler)를 통해 **어셈블리어 파일(*.s)**을 **오브젝트 파일(*.o)**로 변환

4. 링킹(Linking): 링커(Linker)를 통해 **오브젝트 파일(*.o)**을 묶어 **실행 파일**로

GCC(GNU Compiler Collection)

GNU 프로젝트의 일환으로 개발되어 널리 쓰이고 있는 컴파일러이다.

컴파일러, GCC, GCC의 옵션에 대한 내용은 링크 참조 (출처: https://velog.io/@dhwltnoooh/gcc-컴파일러)

크로스컴파일러

컴파일러가 실행되는 플랫폼(PC)이 아닌 다른 플랫폼(ARM)에서 실행 가능한 코드를 생성할 수 있는 컴파일러이다. 예를 들어, 작성한 C 파일을 운영체제를 지원하지 않는 마이크로컨트롤러와 같이 컴파일이 실현 불가능한 임베디드 시스템에서 컴파일 하는데 사용되는 것이다.

앞에서 말한 GCC 컴파일러는 리눅스에서 사용할 수 있는 크로스컴파일러로, 실습과 같이 ARM 계열의 프로세서에서 동작하는 실행 프로그램을 만들기 위해선 'arm-linux-gnueabihf-gcc'처럼 이름을 바꾸어 컴파일하면 된다. 여기서 'gnueabihf'는 Operating System의 이름+데이터를 주고받는 인터페이스를 말한다.

툴체인(Toolchain)

대부분 크로스컴파일 환경을 뜻하며, 소프트웨어 개발에 사용되는 프로그래밍 도구의 집합이라 할 수 있다. 기본 구성으로는 소스 코드 편집을 위한 문서 편집기와 컴파일러, 컴파일한 object들을 컨트롤하기 위한 어셈블러, 링커, 운영체제의 기능을 제공하는 라이브러리로 구성된다.

크로스컴파일러(출처: https://kkhipp.tistory.com/160), 툴체인(출처: https://jayleekr.github.io/posts/01-Toolchain/)

Makefile

리눅스에서 프로그램을 빌드할 때 주로 쓰는 자동화 도구 'make'라는 툴의 전용 표준 문법을 가지고 빌드 과정을 서울하는 파일이다. 개발중인 프로그램의 작업 디렉토리에 Makefile이라는 파일명을 하는 스크립트를 저장해 두면, make가 그것을 해석해서 자동으로 빌드를 수행하도록 할 수 있다.

Makefile 개념, 작성법(출처: https://nanite.tistory.com/77)

개발 환경 테스트 실습 명령어

1. su
2. vim hello.c
3. cd /
4. mkdir 폴더명
5. cd 폴더명
6. vim hello.c
7. sudo apt-get install lib32z1

```
#include <stdio.h>
int main() {
  printf("Hello!! ARM Linux!!\n");
  return 0;
}
```

```
CC = /opt/gnueabi/opt/ext-toolchain/bin/arm-linux-gnueabihf-gcc
```

/* 공유라이브러리 파일 다운로드 */

8. /opt/gnueabi/opt/ext-toolchain/bin/arm-linux-gnueabihf-gcc -o hello hello.c
9. ./hello
10. file hello
11. vim Makefile

```
CFLAGS = -Wall
all : arm_hello
arm_hello : hello.c
  $(CC) $(CFLAGS) hello.c -o arm_hello

clean :
  rm -f *.bak *.o arm_hello
```

12. make
13. ls
14. file arm_hello