2주차 예비보고서

전공: 컴퓨터공학과 학년: 2학년 학번: 20191619 이름: 이동석

**1. 목적**

UNIX 상에서 제공하는 C/C++ 관련 도구를 미리 사용해 봄으로써, 수업시간에 실습이 원활히 진행될 수 있도록한다.

**2. 예비 학습**

C/C++ 프로그램의 컴파일 과정은 프리프로세서(Preprocessor 전처리기), 컴파일러(Compiler), 어셈블러(Assembler), 링커(Linker) 총 4가지 단계로 구성된다. 컴파일 과정에서의 임시파일을 보기위해 gcc --save-temps 를 이용했다.

1 #include<stdio.h>

2 #define f(x) (x\*x+5\*x+4)

3 #define a -4

4 #define b -1

5 void main()

6 {

7 int sol\_1=a;

8 int sol\_2=b;

9

10 printf("%d\n",f(sol\_1));

11 printf("%d\n",f(sol\_2));

12 }

**2-1. 프리프로세서**

프리프로세서에 의해 위의 코드는 다음과 같이 바뀐다. 전처리기는 매크로 치환과 확장 #으로 시작되는 전처리 구문 해석과 주석을 처리하는 등의 역할을 한다. 이때 생성되는 임시파일은 파일명.i 이다.

853 void main()

854 {

855 int sol\_1=-4;

856 int sol\_2=-1;

857

858 printf("%d\n",(sol\_1\*sol\_1+5\*sol\_1+4));

859 printf("%d\n",(sol\_2\*sol\_2+5\*sol\_2+4));

860 }

모두 치환된 것을 볼 수 있다.

**2-2. 컴파일러**

컴파일러는 전처리기가 생성한 파일명.i를 컴파일 하여 aseembler 가 읽을 수 있는 파일명.s 의 소스코드를 생성한다. 현재 우리가 사용하고 있는 gcc(GNU Compiler Collection)은 대표적인 C 컴파일러다.

C의 컴파일 과정은 전단부, 중단부, 후단부로 나뉜다. 전단부에서는 언어의 종속적인 부분을 처리한다. Lexical analysis(어휘분석) 과정으로 토큰을 생성하고 Parser가 이를 AST(Abstract Syntax Tree) 구조로 나타내고 검증한다. 중단부에서는 GIMPLE Tree를 SSA형태로 변환하고, 최적화를 수행한다. 이후 후단부에서는 아키텍처 최적화를 수행한다.

**2-3. 어셈블러**

컴파일러 단계가 끝나면, 파일명.s가 생성된다. 어셈블러는 이 파일을 명령어와 데이터가 들어있는 ELF 바이너리 포맷 구조를 가진 목적코드 파일명.o을 생성한다. 이 파일은 링커가 바이너리 파일을 하나의 실행 파일로 효율적으로 묶기 위해 정형화 되어있다. 어셈블러는 mnemonic 코드를 기계어로 변환하고, symbol lables에 주소를 할당한다. 대부분의 어셈블러는 2개의 pass를 가지고있다.

**2-4. 링커**

링커는 목적파일들과 C 라이브러리 등을 링크한다. 링크에는 static linking(정적 링킹)과 dynamic linking(동적 링킹) 이 있다. 일반적으로 우리는 정적링킹을 사용한다. 링킹 과정을 거치면 실행파일이 생성된다.