1주차 예비보고서

전공: 컴퓨터공학과 학년: 2학년 학번: 20231515 이름: 김다은

2.

|  |
| --- |
| **1. 전처리 단계(Preprocessing Stage)**  ‘#include’, ‘#define’, ‘#ifdef’ 같은 전처리 지시문을 처리한다. 전처리기(Preprocessor)는 소스 코드의 ‘#include’에 포함된 헤더 파일을 가져와 소스코드에 삽입한다. 이후에는 매크로 치환이 진행된다. ‘#define’ 등의 전처리 지시문으로 선언된 부분을 모두 치환한다. 전처리 과정을 거치면 .i의 확장자를 가지는 프로그램이 생성된다.  **2. 컴파일 단계(Compilation Stage)**  컴파일 과정에서는 앞선 과정에서 생성된 파일을 compiler가 어셈블리어로 변환시킨다. 이때 컴파일 과정은 또 한번 전단부(Front-end), 중단부(Middle-end), 후단부(Back-end)로 나뉜다.  2-1. 전단부  전단부에서는 프로그래밍 언어에 종속적인 부분이 처리된다. 보통 첫번째로 어휘분석이 진행된다. C소스코드를 의미가 있는 최소단위로 나눈다. 두번째로 구문분석이 진행된다. 최소단위로 트리를 만들면서 문법 오류를 찾아내는 것이다. 세번째로 의미분석이 진행된다. 문법적 오류가 아니라 의미상 오류가 있는지 검사하는 것이다 네번째로 중간 표현이 생성된다. 중단부 과정에 진입하기 위한 GIMPLE트리를 생성하는 것이다. GIMPLE 트리는 여러 프로그래밍 언어 소스를 표준화하여 공통된 표현으로 바꾼 것이다.  2-2. 중단부  중단부에서는 아키텍쳐 비종속적인 최적화가 진행된다. 전단부에서 만들어진 GIMPLE트리를 Static Single Assignment 형태로 바꾼 후, 후단부에서 사용하는 Register Transfer Language 구조로 변환한다. 이 RTL이란 고급 언어와 어셈블리 언어의 중간 구조를 말한다.  2-3. 후단부  후단부에서는 RTL Optimizer에 의해 아키텍쳐에 종속적인 최적화를 진행한다. 이러한 컴파일 과정의 결과 .s 확장자를 가진 어셈블리 코드로 이루어진 파일이 생성된다.  **3. 어셈블 단계(Assemble Stage)**  컴파일과정을 거친 어셈블리 코드는 assembler에 의해 기계어로 어셈블된다. 오브젝트 파일은 어셈블된 프로그램의 데이터와 명령어가 들어있는 ELF 바이너리 포맷 구조를 갖는다. 이 구조는 링킹 단계에서 Linker가 다수의 Binary 파일을 하나의 실행 파일로 묶기 위한 정보를 효율적으로 읽을 수 있도록 일정한 규칙에 따라 만들어진 구조이다. 결과적으로 Assembler에 의해 오브젝트 파일 .o가 생성된다.  **4. 링킹 단계(Linking Stage)**  Linker는 오브젝트 파일들과 프로그램에서 사용된 C 표준 라이브러리, 사용자 라이브러리를 연결한다. 다시 말해 여러 오브젝트 파일과 라이브러리 파일을 결합하여 하나의 실행가능한 binary 파일을 생성한다. linker는 오브젝트 파일에서의 함수 호출을 해당 함수의 정의와 연결한다. 또한 라이브러리 함수들도 이처럼 연결한다. 링크 단계가 완료되면 실행 파일이 생성된다. |