03. Source Code and Command

yongzzai

Summary of Chap.03 of book [1]

1 저급, 고급 언어

* 고급언어는 개발자가 이해하기 쉽게 만든 언어.
* 컴퓨터가 이것을 실행할 때에는 저급언어로 변환되어 이해함.

1.1 저급 언어

* 저급 언어는 기계어와 어셈블리어로 나뉘어짐.
* 기계어는 이진수 또는 십육진수로 표현됨
* **어셈블리어**는 기계어를 읽기 편한 상태로 번역한 저급 언어임.

1.2 고급 언어

* 고급언어가 저급언어로 변환되는 방식은 두가지가 있음.
1. 컴파일
2. 인터프리트

1.3 컴파일 언어

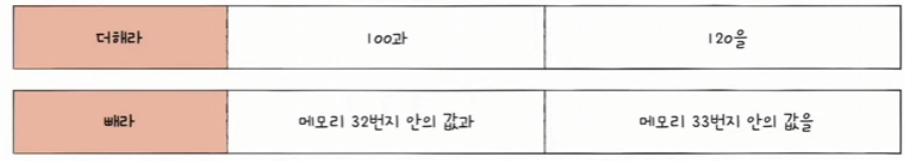
* 고급언어인 소스코드가 컴파일러를 통해 저급언어로 변환됨. 컴파일의 결과물로 저급언어인 **목적 코드**로 변환됨.

1.4 인터프리트 언어

* 인터프리터에 의해 한줄씩 실행되는 언어임.

2 명령어의 구조

* 명령어는 수행할 연산과 연산에 사용될 데이터혹은 데이터가 저장된 위치로 이루어져있음.
* 붉은 색 부분은 **연산코드**라고 부르고, 흰색부분은 **오퍼랜드**라고 부름.



2.1 오퍼랜드

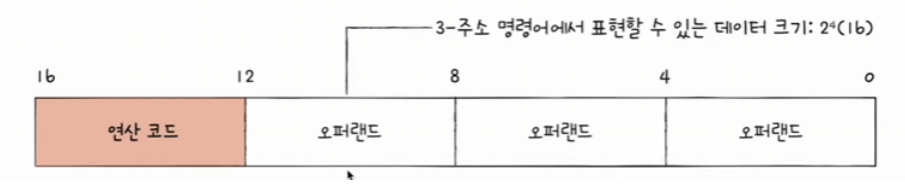
* 연산에 사용될 데이터 또는 연산에 사용될 데이터가 저장된 위치: **오퍼랜드**.
* 오퍼랜드가 담기는 공간은 **주소필드**라고 부르기도 함.

2.2 연산코드

* 연산코드가 담는 내용은 CPU마다 다름.
* 공통적으로 가지는 특징으론
1) 데이터전송: 데이터를 옮기거나, 메모리에 저장, CPU로 데이터를 이동, 등
2) 산술/논리 연산: 덧셈, 뺄셈, 곱셈, 나눗셈, AND/OR/NOT 연산, 등
3) 제어흐름 변경: 특정 메모리 주소로 실행 순서를 옮김 등
4) 입출력 제어: 특정 입출력 장치에서 데이터를 READ/WRITE 등

2.3 명령어 주소

* 명령어에 데이터를 직접적으로 배치하면 용량 부족 문제가 발생될 수 있음. 따라서, 데이터를 직접 명령어에 싣기보다는 데이터의 주소를 싣는 방식이 더 많이 사용됨.



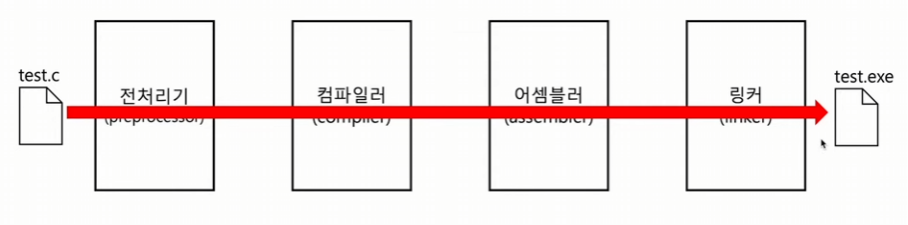
* 위와 같이 오퍼랜드가 3개 들어가있는 명령어는 포함할 수 있는 데이터의 용량이 매우 작아지는데, 데이터의 주소를 명령어에 포함하면 명령어에 포함할 수 있는 데이터의 용량이 커짐.

2.4 명령어 주소 지정 방식

* **유효주소 (effective address)**는 연산에 사용될 데이터가 저장된 위치를 뜻함.
* CPU는 명령어에 있는 주소를 보고 데이터가 어디에 저장되어있는지를 파악할 수 있어야하는데 이때 연산에 사용할 데이터가 저장된 위치를 찾는 방법을 **명령어 주소 지정 방식 (addressing modes)**라고 함.

1. 즉시 주소 지정 방식 (immediate addressing mode)
연산에 사용할 데이터를 오퍼랜드 필드에 직접 명시하는 방법으로 사용가능한 데이터의 크기가 작아지지만 빠름.
2. 직접 주소 지정 방식 (direct addressing mode)
오퍼랜드 필드에 유효 주소를 직접적으로 명시하는 방법으로 유효 주소를 표현할 수 있는 크기가 연산 코드만큼 줄어듬.
3. 간접 주소 지정 방식 (indirect addressing mode)
오퍼랜드 필드에 유효 주소의 주소를 명시하는 방식으로 속도가 느림. (유효주소를 저장한 위치의 주소를 명시하는 것임.)
4. 레지스터 주소 지정 방식 (register addressing mode)
연산에 사용할 데이터가 저장된 레지스터를 명시하는 방식으로 메모리에 접근하는 속도보다 레지스터에 접근하는 것이 더 빠름.
5. 레지스터 간접 주소 지정 방식 (register indirect addressing mode)
연산에 사용할 데이터를 메모리에 저장하고 그 주소를 저장한 레지스터를 오퍼랜드 필드에 명시하는 방식.

3 C언어의 컴파일 과정



References

[1] Minchul Kang. 혼자 공부하는 컴퓨터 구조+운영체제. Hanbit Media, 2022.