NASM x86 방식의 어셈블리로 구현한 Hello, World! 출력 코드.



1. 개요  
   기계어와 1:1 대응이 되는 저급 프로그래밍 언어
2. 특징  
   컴퓨터 구조에 따라 사용하는 기계어가 달라지며 거기서 기계어와 대응되어 만들어지는 어셈블리어도 각각 다르게 된다.  
   CPU마다 지원하는 오퍼레이션의 타입과 개수는 제각각 이며, 레지스터의 크기와 개수, 저장된 데이터 형의 표현도 각기 다르다.
   1. 장단점
      1. 장점  
         기계어에 비해 가독성이 현저히 좋아진다. 또한 프로그래머가 따로 주석을 달 수 있게 되어 코드의 명령어가 어떤 기능을 하는지 알 수 있다.

기계어와 같은 저수준 언어이기 때문에 바이너리 파일크기가 작고 작동속도가 빠르다. 때문에 잘 짜여진 어셈블리 코드로 최적화 작업을 하면 고급 언어로 만든 프로그램보다 작동속도가 빠를 수 있다.

* + 1. 단점   
       저급언어의 특성상 고급 언어에 비해 생산성이 매우 떨어진다.

또한 작동 속도가 빠른 장점이 무색하게 어셈블리를 완전하게 이해하지 못하면 고급 언어로 작성한 코드보다 기능이 떨어질 수도 있다.  
예를 들어 고급 언어의 경우 특정 명령어를 지정해 줄 때 간단하게 코드를 작성해도 잘 작동되는 것과 달리 어셈블리의 경우에는 하나하나 일일이 세세하게 지정해 줘야만 하기 때문에 시간 대비 효율이 현저하게 떨어진다.

* 1. 쓰임새
     1. 과거  
        1900년대 후반 대다수의 프로그램들은 어셈블리어로 작성되어왔었다.  
        컴퓨터의 성능 자체가 낮았던 시기에는 최적화가 중요했었다. CPU의 구조가 매우 단순했고 RAM의 용량도 작았기 때문에 8비트 시절까지만 하더라도 어셈블리가 필수로 여겨져왔다.

또한 게임 프로그래밍에서는 필수로 요구되었던 언여였다.  
아타리, 패미콤과 같은 콘솔 게임기가 저성능이였기 때문에 게임의 성능을 최대한으로 끌어내기 위해선 어셈블리가 적절한 선택이었을 것이다. 이러한 어셈블리로 만들어진 대표적인 게임이 포켓몬스터 금/은, 롤러코스터 타이쿤 등이 있다.

* + 1. 현재

오늘날에는 일반적인 프로그램을 개발할 땐 그냥 고급 언어를 사용한다.  
최적화 대비 성능을 최대로 끌어내야 했었던 과거와는 달리 컴퓨터의 성능이 좋아졌기 때문에 생산성의 가치가 더욱 높아졌기 때문이다.

오늘날 어셈블리를 사용하는 경우는 매우 특수한 경우이며 보통 최적화가 매우 중요한 임베디드 프로그래밍에서 쓰인다.