20180917 마프

ISA : instruction set architecture

ISA에 포함되어야 하는것은? Instruction이 일단 포함되어야 할 것이고 register도 포함되어야 할 것이다. Data type도 마찬가지로 ISA에 포함되어야 한다(integer, float, etc…)

Machine code는 01로 구성된 진짜배기 그냥 컴퓨터가 알아먹는 code이다. 그에 반해서 assembly code는 그런 instruction을 조금 더 text로 represent한 것이다. 물론 c코드 같은거보다는 조금 더 abstraction이 덜하긴 하다!

Microarchitecture를 몰라도 프로그램을 짤수 있으나 이걸 잘 이용하지 않는다면 poor performance를 보게 될 것이다..

ISA는 결국 cpu에서 사용하는 시스템 룰 같은 느낌인듯

PC : program counter에서는 현재 실행하고 있는 instruction이 무엇인지 주소를 저장한다. 물론 next instruction의 주소도 마찬가지

Register는 자주 사용되는 program data를 저장하는 듯 하다.

Condition codes의 경우 가장 자주 사용된 arithmetic operation을 저장한다고 하는데 정확히 왜 그러는지는 잘 모르겠음(설명부분에서 깜빡 정신놓음)

Memory는 code,user data, os data등 다양한 것을 저장한다.

C 파일을 object code로 변환하는 과정

C code를 먼저 컴파일러를 이용해서 assembly program(\*.s)으로 변환하고 assembler를 이용하여 object program으로 변환한다.(\*.o)

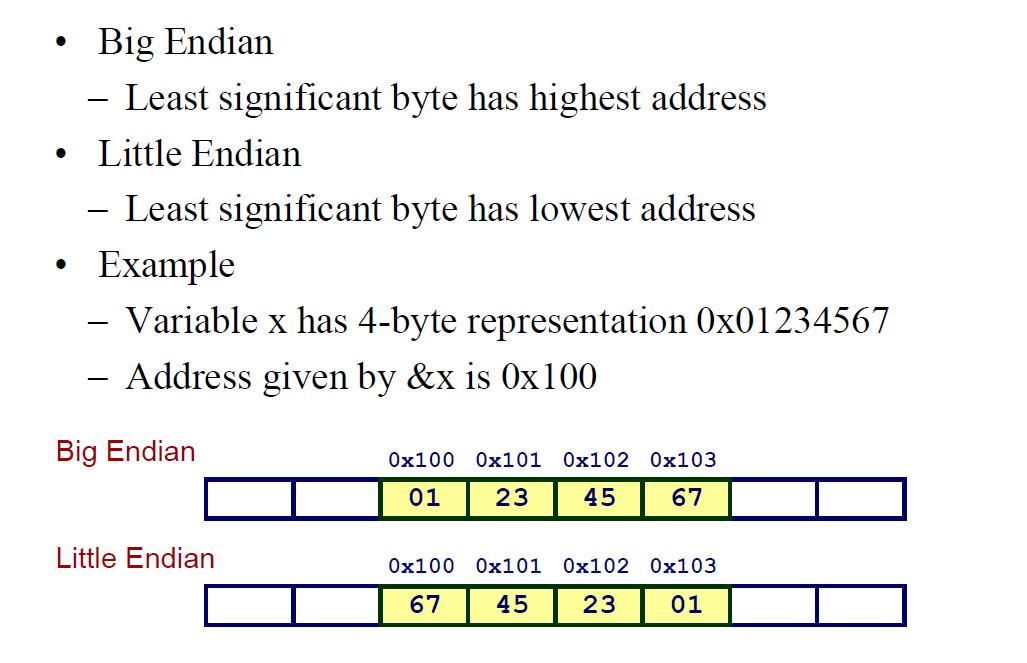
그런 다음 linker를 이용하여 executable program(\*.exe)으로 변환한다.

-------일단 내가 이해한건 이런데 아닐수도 있음!!-------

위쪽에서 object program으로 바꾸어놓은것들 여러 개를 이어주는 것이 linker인듯하다. 보통 c코드 하나마다 object 프로그램이 있으니 이것들이 서로를 참조하게 만들어 놓은 상황에서 이것들을 이어주면 executable program이 되는것임!!

Register 메모리에 저장하고 싶으면 괄호를 쳐라? 메모리의 값에 관심이 있으면 괄호를 치라고 했나? 정확히 기억안남. 녹음파일 43~44분 사이에 언급됨.

Intel은 little endian을 쓰는가? Big endian을 쓰는가?? -> little endian인 듯 하다.



어셈블리언어는 참 어렵다… 21/50페이지에서 쏼라쏼라하는데 먼소린지 1도 모르겠음 ㅎ….

Gdb debugger란게 있는데 이번주에 또 배울 듯…. 무엇일까 이건또 ㅜㅜ

disassemble하는 방법인듯하다…

일단 x86-64 register의 경우 16가지가 있는듯하다…

%e로 시작하는건 32비트고 %r로 시작하면 64비트라고 하는거같은데 그러면 뒤쪽에 있는 r 여러개는 또 뭐임?

이게 레지스터는 16개밖에 없는데 프로그램에선 당연히 16개가 넘는 variable을 사용한다. 이럴때 모든 variable이 레지스터에 등록되는 것이 아니라 메모리에 저장되어 있다가 이것들 중 필요한 것들을 레지스터에 불러와서 사용하는 듯 하다.

32비트에서는 %e로 시작하는 8개밖에 사용하지 않는다고 하는듯하다. 그러나 64비트로 넘어오면 16종류가 되는듯하다.(r로시작함)

그런데 이런 %eax류의 register들도 %eax->%ax->%ah,%al 등 32비트->16비트->8비트로 쪼갤수 있다!!

Immediate operand가 여기에 나오네 ㅡㅡ constant integer type이라는듯하다. C언어 constant와 비슷하지만 앞에 $가 붙어있다. $0x400같이