상명대학교 컴퓨터과학과 "EA0003: 시스템소프트웨어" 실습

- 실습 번호: lab01

- 실습 디렉터리: ~/sys\_sw/lab01 이름: 이경헌

- 실습 날짜: 2017년 월 일

- 실습 제목: NASM 사용 학번: 201311229

- 실습 내용:

1. (실습준비) 자신의 홈 디렉터리 아래에 실습 디렉터리 (~/sys\_sw/lab01)를 만든다. 이번 실습에서 만들어지는 모든 파일들은 이 디렉터리에 두도록 한다.
2. (nasm 명령) nasm 명령의 사용법에 대해 알아보자. “man nasm” 명령을 입력하면 nasm 사용법에 대한 매뉴얼을 볼 수 있다.
   1. nasm 명령줄의 형식을 적으시오.

nasm [-@ response file] [-o outfile] [-f format] [-l listfile] [options...] [--] filename

* 1. nasm 명령 중 자주 사용하게 될 –f, –o, –l 옵션에 대해 조사하여 아래에 적으시오.

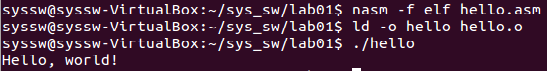
-f 출력할 파일의 format을 지정할수 있게 하는 옵션.

-o 출력할 파일의 이름을 지정할 수 있게 하는 옵션.

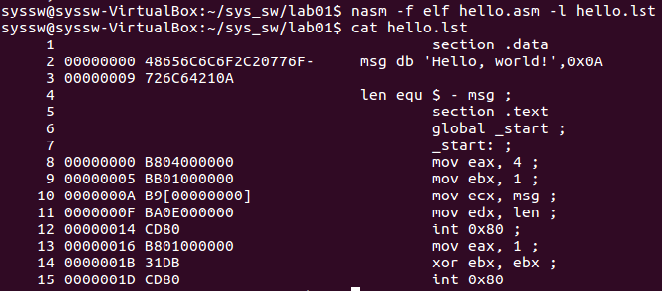
* 1. “nasm –hf” 를 입력하면 nasm 이 지원하는 오브젝트 파일 형식을 알 수 있다. 이를 실행해 보고, 디폴트 오브젝트 파일 형식과 우리가 앞으로 사용해야 할 오브젝트 파일 형식이 무엇인지 적으시오.

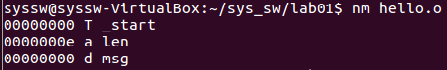
디폴트 오브젝트 파일 형식 : Bin 형식 , 앞으로 사용해야 할 오브젝트 파일 형식 :elf

1. (nasm 실행) nasm 을 사용하여 간단한 어셈블리 프로그램을 실행시켜 보자.
   1. (간단한 어셈블리 프로그램의 실행) vi 에디터 등을 사용하여 강의 자료에 있는 hello.asm 파일을 작성하고, 이를 어셈블, 링크하여 실행 파일 hello 를 생성한 후, 제대로 실행되는지 확인하시오 (이 과정은 수업 시간에 다루었음). 이를 확인하기 위해 어셈블, 링크, 실행하는 과정을 수행한 터미널 창을 캡쳐하여 아래에 첨부하시오.



* 1. (어셈블리 리스트 파일) Nasm은 어셈블 과정에서 오브젝트 파일 외에 리스트 파일도 만들 수 있다. 리스트 파일을 만들면, 어셈블러가 생성하는 16진수 형태의 코드를 확인할 수 있다. 3.1의 컴파일 과정에 –l 옵션을 사용하여 hello.lst 라는 이름으로 리스트 파일을 만들어 보자 (이 파일의 내용은 수업 시간에 다루었음). 확인을 위해 이 과정을 수행한 터미널 창을 캡쳐하여 아래에 첨부하시오.



* 1. (오브젝트 파일과 심볼) 리눅스의 nm 명령을 사용하면 어셈블리 프로그램에서 정의된 심볼들에 대한 정보를 얻을 수 있다. 위 3.1 에서 생성된 오브젝트 파일 hello.o 에 대해 명령 “nm hello.o” 를 수행해 보자. 이 결과 출력된 내용을 아래에 적고 출력된 내용에 대해 설명하시오 (nm 명령의 출력에 대한 설명은 “man nm” 명령을 참조하시오). 

00000000T \_start

텍스트 코드 섹션 00000000 에 있는 (시작점) \_start 다.

0000000e a len

링크를 해도 변하지 않는 값을 지닌 0000000e 에 있는 길이를 나타내는 len이다.

00000000 d msg

링크를 해도 변하지 않는 값을 지닌 00000000에 글을 쓰라는 것을 나타내는 msg이다.

* 1. (프로그램의 entry point) 위의 hello.asm 에 사용된 심볼 중 \_start 라는 심볼을 \_mystart 로 변경한 후, 이를 hello\_mystart.asm으로 저장하시오. 3.1과 동일한 방법으로 hello\_mystart.asm을 어셈블, 링크하여 실행 파일 hello\_mystart 를 생성한 후, (1) hello\_mystart 가 링크되는 과정에서 어떤 문제가 발생하는지 아래에 적으시오. (2) 왜 이와 같은 문제가 발생하는지 이유를 적고, 이 문제를 해결하기 위해서는 ld 명령을 사용할 때 어떤 옵션이 필요한지 적으시오 (힌트: “man ld” 명령을 통해 ld 에 대해 조사해 볼 것). (3) 확인을 위해 문제를 해결한 과정을 수행한 터미널 창을 캡쳐하여 아래에 첨부하시오.



* 1. (gcc를 이용한 링크) 위의 hello.asm 에 사용된 심볼 중 \_start 라는 심볼을 main 으로 변경한 후, 이를 hello\_main.asm으로 저장하시오. 위와 동일한 방법으로 hello\_main.asm을 어셈블 하되, 링크 시에는 ld 대신 gcc 를 사용하시오. 명령 “gcc hello\_main.o –o hello\_main” 과 같이 실행 파일을 만든 후 제대로 실행되는지 확인하는 과정을 수행한 터미널 창을 캡쳐하여 아래에 첨부하시오.

1. (nasm 의사명령/지시자) E-Camous 게시판의 nasm 어셈블러 매뉴얼(nasmdoc.pdf)을 읽고, 다음 내용들을 조사/요약하시오.
   1. nasm의 의사명령어들 (pseudo instructions) (매뉴얼 3.2절 참고)

실제 명령어가 아니지만 명령 필드에서 사용된다.가장 사용하기 편리한 곳이기 때문이다.

실제 cpu가 알아들을수 있는 명령어가 아니라 기계어로 변환하는 어셈블러를 위한 명령어 이다.

* 1. nasm에서의 상수 (매뉴얼 3.4절 참고)

nasm 은 정수와, 문자, 문자열, 소수점 숫자 까지 총 네 가지 타입의 상수들을 다룰 수 있다.

* 1. 어셈블러 지시자 section 또는 segment (매뉴얼 5.2절 참고)

Section과 segment 는 완전하게 같다.\

작성한 코드가 출력파일의 어느섹션에 어셈블 될지를 변경할수 있다.

section의 수와 이름이 고정되어있는 경우도 있고, 사용자가 원하는데로 만들수있는 경우도 있다.

존재하지 않는 section으로 switch 하려고 하면 에러를 발생하거나 새 section을 정의 할수 있다.

Section 또는 segment 파일의 포맷과, 숫자, section 의 이름 을 바꿀 수 있다.

* 1. 어셈블러 지시자 global (매뉴얼 5.5절 참고)

외부 링커에서 접근할 때 오류를 방지하기 위해 GLOBAL로 선언한다.

Global로 정의된 심볼은 외부 링커에게 프로그램 내부의 심볼에 접근가능하게 만들어준다.

한번에 하나씩의 argument를 사용 할 수 있다.

끝.