실습 1. 간단한 어셈블리언어 프로그램

1. 간단한 어셈블리언어 프로그램

(1) 자료실에서 asm.bat, print.inc, print.lib파일을 다운로드 받아서 어셈블리 소스코드를 작성하는 디렉토리에 저장한다.

(2) 강의자료의 first2.asm, first3.asm, first4.asm를 실행하여 결과를 적고, 여기에서 사용한 레지스터 값의 출력 방법을 말하시오.

(3) 강의자료의 second.asm, second2.asm, second3.asm를 실행하여 결과를 적고, 여기에서 사용한 메모리 변수 값을 출력하는 방법과 문자열 출력 방법을 말하시오.

(4) 리틀엔디언 순서가 무엇인지 설명하시오. 그리고 강의자료의 endian.asm을 실행하여 결과를 적고, 리틀엔디언 순서를 확인해보시오.

(5) 기호상수를 정의하는 3가지 디렉티브를 적으시오. 그리고 강의자료의 equ.asm 프로그램에서 정의된 기호상수는 어떤 값으로 변환되는 지 말해보고, 이 프로그램을 실행하여 변환된 결과를 확인하시오.

2. 데이터 전송 명령어

(1) 별도로 주어진 mov.asm을 실행하여 결과를 적고, 여기서 사용한 데이터 전송 명령어들 3가지 유형으로 분류하여 보시오.

(2) mov 명령어의 피연산자 사용 제한을 적어보시오.

(3) 다음과 같이 메모리 데이터 배치를 바꾸는 어셈블리언어 프로그램을 작성하고, 배열 데이터를 출력하여 결과를 확인하시오.

```
.data
array       dword    10h, 20h, 30h, 40h              (실행 전)

array       dword    40h, 10h, 20h, 30h              (실행 후)
```

데이터 전송 명령어 사용

(예) 여러 가지 데이터 전송 명령어들을 사용하는 프로그램

mov.asm

```
; mov.asm
include print.inc

.data
dvardword   ?, ?
wvarword    8000h

.code
main proc
    ; dvar[0] <- wvar (signed)
    movsx   eax, wvar       ; sign extension
    mov     dvar, eax
    ; dvar[1] <- wvar (unsigned)
    movzx   eax, wvar       ; zero extension
    mov     dvar+4, eax
    ; print dvar array
    mov     esi, offset dvar
    mov     ebx, 4          ; 32-bit unit size
    mov     ecx, 2          ; two units
    call    DumpMem

    ; dvar1 <=> dvar2 (exchange)
    mov     eax, dvar
    xchg    eax, dvar+4
    mov     dvar, eax
    ; print dvar array
    mov     esi, offset dvar
    mov     ebx, 4          ; 32-bit unit size
    mov     ecx, 2          ; two units
    call    DumpMem

    mov     eax, 0          ; exit 0
    call    ExitProcess
main endp
end main
```

실행결과

```
C:\asm> mov

Dump of offset 01314000
------------------------------
FFFF8000  00008000                    ... 8000h의 두 가지 확장된 결과

Dump of offset 01314000
------------------------------
00008000  FFFF8000                    ... 교환된 결과

C:\asm>
```