**4.1.10**

1. 피연산자의 세 가지의 기본 유형은 무엇인가?

답: 즉시값, 레지스터, 메모리

5. 인텔이 사용하는 연산 표기에서 reg/mem32는 무엇을 나타내는가?

답: 32비트 피연산자, 32비트 범용 레지스터나 메모리

6. 인텔이 사용하는 연산 표기에서 imm16은 무엇을 나타내는가?

답: 16비트 워드 즉시값

7. 다음 각 문장에 대해서 명령어가 올바른지를 말하시오,

a. mov ax, var1 부적합하다. 두 피연산자의 크기가 다르다.

b. mov ax, var2 적합하다.

c. mov eax, var3 부적합하다. 32비트 레지스터에 16비트 메모리 피연산자를 지정

d. mov var2, var3 부적합하다. 두 피연산자가 모두 메모리 피연산자여서

e. movzx ax, var2 부적합하다. 소스 피연산자가 바이트 크기의 메모리나 레지스터여야함

f. movzx var, al 부적합하다. 도착점은 레지스터여야만 한다.

g. mov ds, ax 적합

h. mov ds, 1000h 부적합하다. 즉시값은 세그먼트 레지스터로 이동될 수 없다.

9. 다음 각 명령어가 순서대로 실행된 후에 목적지 피연산자의 값은 무엇인가?

mov ax, var2 ; a.

mov ax, [var2+4] ; b.

mov ax, var3 ; c.

mov ax, [var3-2] ; d.

답: a. AX = 1000h

b. AX = 3000h

c. AX = FFF0h

d. AX = 4000h

10. 다음 각 명령어가 순서대로 실행된 후에 목적지 피연산자의 값은 무엇인가?

mov edx, var4 ; a. EDX = 00000001h

movzx edx, var2 ; b. EDX = 00001000h

mov edx, [var4+4] ; c. EDX = 00000002h

movsx edx, var1 ; d. EDX = FFFFFFFCh

**4.2.8**

6. 지시된 곳에 각 명령어가 실행된 후의 Carry, Sign, Zero, Overflow 플래그의 값을 적으시오.

mov ax, 7FF0h

add al,10h ; a. CF =1 SF= 0 ZF= 1 OF= 0

add ah, 1 ; b. CF = 0 SF= 1 ZF= 0 OF= 1

add ah, 2 ; c. CF = 0 SF= 1 ZF= 0 OF= 0

12. Carry와 Overflow 플래그가 동시에 1로 설정되는 연속되는 두 개의 명령어를 적으시오.

답: mov ax, val2

neg ax

13. INC 와 DEC 명령어를 수행한 후에 부호없는 오버플로를 나타내기 위해서 Zero 플래그가 어떻게 사용될 수 있는지를 보여주는 명령어들을 적으시오.

답: move cx, 1

Sub ecx, 1

mov eax, 0FFFFFFFFh

inc eax

inc eax

dec eax

**4.3.8**

**7.** 다음의 각 명령어가 실행된 후에 EAX 값은 무엇일까?

mov eax, TYPE myBytes - 1

mov eax, LENGTHOF myBytes - 4

mov eax, SIZEOF myBytes - 4

mov eax, TYPE myWords - 2

mov eax, LENGTHOF myWords - 3+1

mov eax, SIZEOF myWords - 8

mov eax, SIZEOF myString - 5

9. myWords 에 있는 두번째 바이트를 AL 레지스터로 이동시키는 명령어를 적으시오.

답: mov al, byte ptr [mywords+1]

10. myByte 에 있는 네 바이트 모두를 EAX 레지스터로 이동시키는 명령어를 적으시오.

답: move ax, dword ptr myBytes

**4.4.5**

7. 다음 순서의 명령어에서 오른쪽에 있는 요청한 레지스터의 값을 채우시오.

mov esi, OFFSET myBytes

mov al, [esi] ; a. AL = 10h

mov al, [esi+3] ; b. AL = 40h

mov esi, OFFSET myWords + 2

mov ax, [esi] ; c. AX = 003Bh

mov edi, 8

mov edx, [myDoubles + edi] ; d. EDX = 3

mov edx, myDoubles [edi] ; e. EDX = 3

mov edx, myPointer

8. 다음 순서의 명령어에서 오른쪽에 있는 요청한 레지스터의 값을 채우시오.

mov esi, OFFSET myBytes

mov ax, [esi] ; a. AX = 2010h

mov eax, DWORD PTR myWords ; b. EAX = 003B008Ah

mov esi, myPointer

mov ax, [esi+2] ; c. AX = 0000h

mov ax, [esi+6] ; d. AX = 0000h

mov ax, [esi-4] ; e. AX = 0044h

**4.5.5**

9. 다음 예에서 EAX의 마지막 값을 무엇인가?

mov eax, 0

mov ecx, 10 ; outer loop counter

L1:

mov eax, 3

mov ecx, 5 ; inner loop counter

L2:

add eax, 5

loop L2 ; repeat inner loop

loop L1 ; repeat outer loop

답: 무한 루프에 빠지므로 마지막 EAX값을 구할 수 없다.

10. 안쪽 루프가 시작할 때에 바깥의 루프 카운터가 지워지지 않도록 앞 문제의 코드를 수정하시오.

답: .data

count DWORD ?

.code

mov eax, 0

mov ecx, 10

L1:

mov eax, 3

mov count, eax

mov ecx, 5

L2:

add eax, 5

loop L2

mov ecx, count

loop L1