

CH04.review Question and Exercise

1. A- FFFF8002h / B - 00004321h
2. 10030000h
3. 3001FFFFh
4. 10020001h
5. AI = 4
6. Sign flag = 0
7. 결과가 127이지만 부호있는 바이트의 범위를 넘었고 그걸 알려주는 역할
8. 0000 0000 4444 5555h
9. 0FFF FFFF 8432 6732h
10. 0003 5678h
11. Eax : 1234 1237h
12. Yes
13. No
14. Yes
15. No
16. X, o, x, x, x, x, o, x
17. A : FCh / B : 01h
18. A : 1000h / b : 3000h / c : FFF0h / d : 4000h
19. A : 0000 0001h / B : 0000 1000h / C : 0000 0002h / D : FFFF FFFCh

Algorithm Workbench

1) Exchange upper/lower words in DWORD variable 'three':

```
mov ax, WORD PTR three
mov dx, WORD PTR three+2
mov WORD PTR three, dx
mov WORD PTR three+2, ax
```

2) Reorder A,B,C,D -> B,C,D,A using XCHG (max 3 times):

```
xchg al, bl
xchg bl, cl
xchg cl, dl
```

3) Parity check using PF:

```
add al, 0
jpe EVEN_PARITY
jpo ODD_PARITY
```

4) Add two negatives, cause Overflow:

```
mov al, -100
add al, -60
```

5) Set Zero & Carry flags via addition:

```
mov al, 0FFh
add al, 1
```

6) Set Carry flag via subtraction:

```
mov al, 0
sub al, 1
```

7) EAX = -val2 + 7 - val3 + val1:

```
mov eax, 7
add eax, val1
sub eax, val2
```

sub eax, val3

8) Loop sum of DWORD array (scaled indexed):

xor eax, eax

xor edi, edi

mov ecx, LENGTHOF arrDwords

L1:

add eax, DWORD PTR arrDwords[edi*4]

inc edi
loop L1

9) $AX = (val2 + BX) - val4$:

mov ax, val2

add ax, bx

sub ax, val4

10) Set Carry & Overflow flags at same time:

mov al, 80h

add al, 80h

11) Use Zero flag to detect unsigned overflow after INC/DEC:

inc ax

jz UnsignedOverflow_INC

dec ax

inc ax

jz UnsignedOverflow_DEC

12) Align myBytes to even address:

ALIGN 2

13) TYPE/LENGTHOF/SIZEOF results:

a) 1

b) 4

c) 4

d) 2

e) 4

f) 8

g) 5

14) Move first 2 bytes of myBytes to DX (result 2010h):

mov dx, WORD PTR myBytes

15) Move second byte in myWords to AL:

mov al, BYTE PTR myWords+1

16) Move all 4 bytes in myBytes to EAX:

mov eax, DWORD PTR myBytes

17) LABEL for myWords as DWORD:

myWords32 LABEL DWORD

mov eax, myWords32

18) LABEL for myBytes as WORD:

myBytes16 LABEL WORD
mov ax, myBytes16