```
; Q 5>
;   a>  Arrange a set of given numbers in ascending order using bubble
;       sort algorithm
    SW      2000h                           ; Signed Array A[]
            [2000h]     =    00006h         ; A.length = 6
            [2002h]     =    0FFFFh         ; A[0] = -1
            [2004h]     =    00003h         ; A[1] = 3
            [2006h]     =    0FFFDh         ; A[2] = -3
            [2008h]     =    0FFFEh         ; A[3] = -2
            [200Ah]     =    00002h         ; A[4] = 2
            [200Ch]     =    00001h         ; A[5] = 1
            .

    A
    1000h
        mov     bx, 2000h
        mov     cx, [bx]
        dec     cx
        add     bx, 2
        mov     si, 0
        loop_pri:
        cmp     si, cx
        jae     loop_pri_end
        mov     di, cx
        loop_sec:
        cmp     di, si
        jbe     loop_sec_end
        shl     di, 1
        mov     ax, [bx + di]
        cmp     ax, [bx + di - 2]
        jge     dont_swap
        xchg    ax, [bx + di - 2]
        mov     [bx + di], ax
        dont_swap:
        shr     di, 1
        dec     di
        jmp     loop_sec
        loop_sec_end:
        inc     si
        jmp     loop_pri
        loop_pri_end:
        hlt
        .

    GO      1000h
    INT                         ;(try '.' here)

    SW      2002h                           ; ans = 0FFFDh => A[0] = -3
    SW      2004h                           ; ans = 0FFFEh => A[1] = -2
    SW      2006h                           ; ans = 0FFFFh => A[2] = -1
    SW      2008h                           ; ans = 00001h => A[3] = 1
    SW      200Ah                           ; ans = 00002h => A[4] = 2
    SW      200Ch                           ; ans = 00003h => A[5] = 3
    .



;   b>  Arrange a set of given numbers in ascending order using stack
    SW      2000h                       ; Signed Array A[]
            [2000h]     =   00006h      ; A.length = 6
            [2002h]     =   0FFFFh      ; A[0] = -1
            [2004h]     =   00003h      ; A[1] = 3
            [2006h]     =   0FFFDh      ; A[2] = -3
            [2008h]     =   0FFFEh      ; A[3] = -2
            [200Ah]     =   00002h      ; A[4] = 2
            [200Ch]     =   00001h      ; A[5] = 1
            .

    A
    1000h
        mov     bx, 2000h
        mov     cx, [bx]
        dec     cx
        add     bx, 2
        mov     si, 0
        loop_pri:
        cmp     si, cx
        jae     loop_pri_end
        mov     di, cx
        loop_sec:
        cmp     di, si
        jbe     loop_sec_end
        shl     di, 1
        mov     ax, [bx + di]
        cmp     ax, [bx + di - 2]
        jle     dont_swap
        xchg    ax, [bx + di - 2]
        mov     [bx + di], ax
        dont_swap:
        shr     di, 1
        dec     di
        jmp     loop_sec
        loop_sec_end:
        inc     si
        jmp     loop_pri
        loop_pri_end:
        mov     si, 0
        loop_ter_1:
        shl     si, 1
        push    [bx + si]
        shr     si, 1
        inc     si
        cmp     si, cx
        jle     loop_ter_1
        mov     si, 0
        loop_ter_2:
```

```
        shl     si, 1
        pop     [bx + si]
        shr     si, 1
        inc     si
        cmp     si, cx
        jle     loop_ter_2
        hlt
        .

    GO      1000h
    INT                     ;(try '.' here)

    SW      2002h                       ; ans = 0FFFDh => A[0] = -3
    SW      2004h                       ; ans = 0FFFEh => A[1] = -2
    SW      2006h                       ; ans = 0FFFFh => A[2] = -1
    SW      2008h                       ; ans = 00001h => A[3] = 1
    SW      200Ah                       ; ans = 00002h => A[4] = 2
    SW      200Ch                       ; ans = 00003h => A[5] = 3
    .



;   c>  Determine the bit positions containing '1' in a 16bit number
    SW      2000h
            [2000h]     =   1249h       ; 16bit number => 0001001001001001b
            [2002h]     =   2004h       ; Address to Bit Position Array B[]
            .

    A
    1000h
        mov     ax, [2000h]
        mov     di, [2002h]
        mov     cx, 0000h
        mov     dx, 0000h
        add     di, 0002h
        loop_label:
        test    ax, 0001h
        jz      bit_is_0
        mov     [di], dx
        add     di, 0002h
        inc     cx
        bit_is_0:
        inc     dx
        shr     ax, 1
        jnz     loop_label
        mov     di, [2002h]
        mov     [di], cx
        hlt
        .

    GO      1000h
    INT                     ;(try '.' here)
    SW      2004h                       ; ans = 0005h => B.length = 5
    SW      2006h                       ; ans = 0000h => b0
    SW      2008h                       ; ans = 0003h => b3
    SW      200Ah                       ; ans = 0006h => b6
    SW      200Ch                       ; ans = 0009h => b9
    SW      200Eh                       ; ans = 000Ch => b12
    .
```