Practice 1: for-loop

```
        .data
a:      .byte   -21, 20, 30, 1, -200, -60, 0, 30, -9, 23
b:      .space   10
```

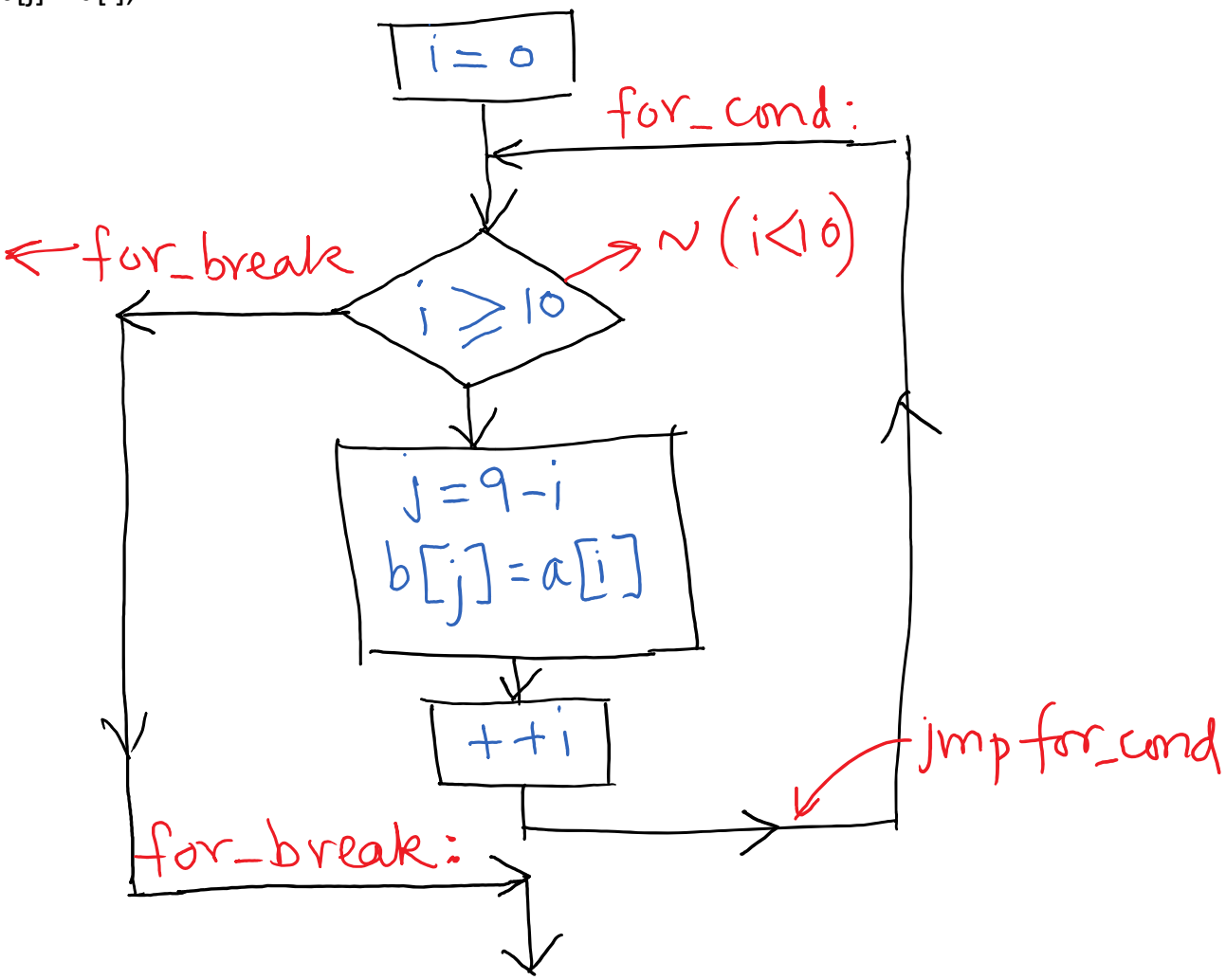
Copy the elements of array a into array b in reverse order

Pseudo-code:

```
for( i = 0;  i < 10;  ++i)
{
    j = 9 - i;
    b[j] = a[i];
}
```

alternate implementation

$\sim(i<10) = i \geq 10$



Program: InverseCopy

```
        .data
a:              .byte       -21, 20, 30, 1, -200, -60, 0, 30, -9, 23
b:              .space      10
;-------------------------------------------------------------------------------
        .text
                        …

                mov.w       #0, R5   ; i = 0
for_cond:
                cmp.w   #10, R5      ; (i >= 10) = ~(i<10)
                jge         for_break

                mov.w   #9, R6       ; j = 9
                sub.w    R5, R6      ; j = 9 - i

                mov.ba(R5), b(R6)    ; b[j] = a[i]

                inc.w  R5
                jmp    for_cond
for_break:

loop:           jmp    loop
```

Practice 3: if-structure inside a for-loop

```
        .data
a:      .byte   10, 20, 30, 40, 50, 60, 70, 80, 90, 100
count:  .space  2
```

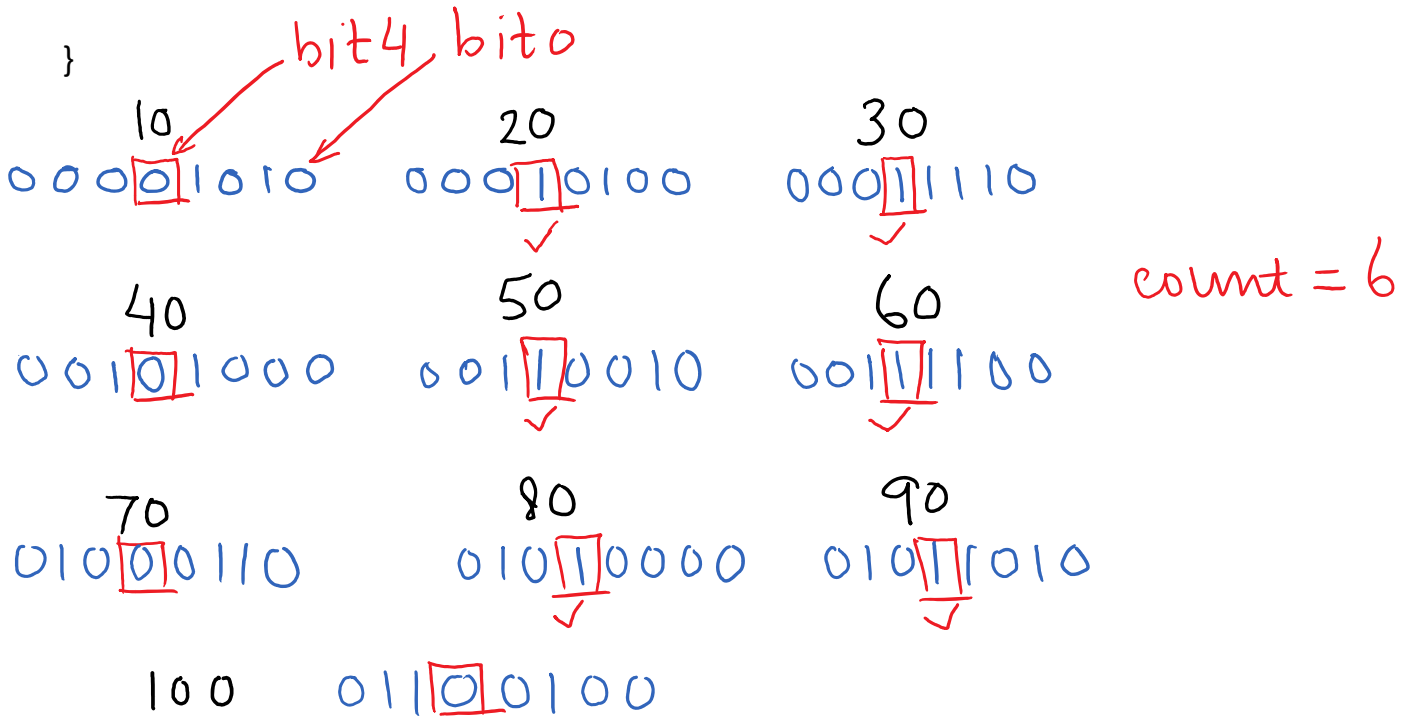
Count the number of elements of array a whose 4th bit is set

Pseudo-code:

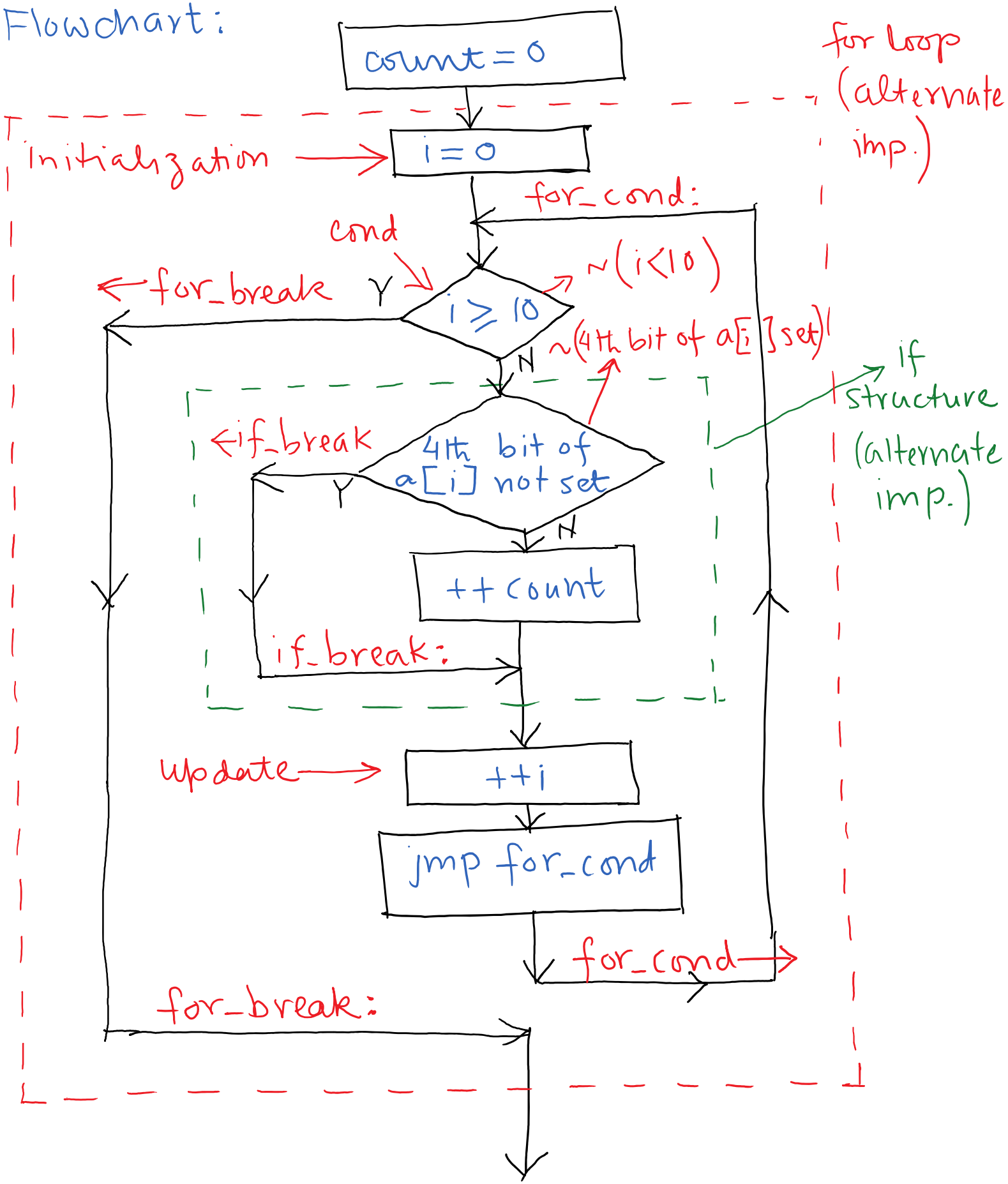
```
count = 0
for( i = 0;   i < 10;   ++i)
{

    If( 4th bit of a[i] is set )
    {
        ++count
    }

}
```



Flowchart:


count = 0
for loop (alternate imp.)
initialization
i = 0
for_cond:
cond
for_break
Y
i >= 10
~(i<10)
N
~(4th bit of a[i] set)
if structure (alternate imp.)
if_break
4th bit of a[i] not set
Y
N
++count
if_break:
update
++i
jmp for_cond
for_cond
for_break:

Program: FourthBitSet

```
        .data
a:              .byte  10, 20, 30, 40, 50, 60, 70, 80, 90, 100
count:          .space2
;-------------------------------------------------------------------------------
        .text
                  …

        mov.w     #0, &count

                  ; for-loop (alternate implementation)
                  mov.w     #0, R5        ;initialize for-loop
for_cond:
                  cmp.w     #10, R5
                  jge       for_break     ; (R5 >= 10) = ~(R5 < 10)
                      ; if-structure (alternate implementation)
                      bit.b   #BIT4, a(R5)
                      jnc     if_break      ; (bit4 of a(R5) not set) = ~(bit4 of a(R5) set)
                      inc.w   count
if_break:
                      ; end if-structure

                  inc.w  R5     ; update for-loop
                  jmp    for_cond
for_break:
                  ;  end for-loop

loop:       jmp   loop
```