```
a db 10          DS: 0A 8F 00 04 75 34 01
b dw 10001111b
⇓
008F
4 bytes ← c dd 1,34,75,04h
8 bytes ← e dq 10

c/b
c in DX:AX

mov AX, [c]      => AX: 7504
mov DX, [c+2]    => DX: 0134
div word [b]

unsigned: e+c                 signed: e+c
e -> edx:eax                  mov eax, [c]
mov eax, [e]                  cdq ; edx:eax = c
mov edx, [e+4]                add eax, [e]
add eax, [c]                  adc edx, [e+4]
adc edx, 0

edx      eax
[e+4] :  e     +
0     :  c
_______________
```

cmp a, b ; fictiv substraction a-b without storing the result.

CF, OF, SF, ZF, AF, PF

```
jump label
    mov . . . .
    - - - - .
    - - - - .
label:
```
jumps here

unsigned (a, b)		signed (g, l)	
ja → jump above	a > b	jg	(greater)
jae	a ≥ b	jge	(lower)
jb → jump below	a < b	jl	
jbe	a ≤ b	jle	
je → jump equal	a = b	je	
jne	a ≠ b	jne	

```
if a>1 then:
    a++
```

```
a db 10
cmp byte [a], 1
jbe noinc
    add byte [a], 1
noinc:
```
we skip the instruction if <= 1

```
if  a > 1  then
        a++
else
        b++
```

```
 a  db  10
 b  db  10
 comp  byte [a], 1
 jbe  incb
        add  byte [a], 1
        jmp  outif
 incb:
        add  byte [b], 1
 outif:
```

dec a; sub [a], 1
inc a; add [a], 1

repeats five times

mov ECX, 5
repeat:

loop repeat → { dec ecx
jecxz }

mov ECX, 0
repeat:

loop repeat → { dec ecx ; $2^{32}-1$ // representing a negative number
jecxz } $\overset{||}{0?}$ goes until gets to 0 again

mov ECX, [e]
jecxz overloop (we jump over the loop if ecx is 0
repeat:
— — —
— — —
loop repeat
overloop

Transform a lowercase array of letters into uppercase.

PS: 'a', 'b'

```
Segment data use 32 class=data
    s db 'a', 'b', 'c', 'd'      -> use ESi
    l equ $ - s                  (if word - divide this value by 2)
                                 (if dword - / 4)
         where we are
         in memory

    d times l db 0
Segment code use 32 class = code
    start:
        mov bl, 'a' - 'A'

        mov ESi, 0
        mov ECx, l
        jecxz end
         repeat:
         mov al, [s + ESi]
         sub al, bl
         mov [d + ESi], al
         iNC ESi
         loop repeat
         end:
```

Method II:

```
mov bl , 'a' - 'A'
mov ESi , 0  = | mov ESi , [s]
               | mov EDi , [d]
mov ECx , l
jecxz end
repeat:
 mov al , [s + ESi]  =  mov al , [ESi]
 sub al , bl
 mov [d + EDi] , al  }  =  mov [EDi] , al
 inc EDi             }     inc ESi ; s+1
 loop repeat               inc EDi ; d+1
 end:
```

An array of doublewords S of length l, obtain the array D of length l-1 such that each elem. of D is the remainder of 2 consec. elements

D(i) = S(i) % S(i+1) ; SIGNED.

```
S dd 100,26,13,5,20
len equ ($-S)/4
d times len dd 0

mov ESI, S
mov EDI, d
mov ECX, len
jecxz endloop
repeat
    mov EAX, [ESI]
    cdq ; edx:eax = first elem.
    mov EBX, [ESI+4] ; second elem.
    idiv EBX ; Edx = edx:eax % ebx
    mov [EDI], EDX
    add ESI,4
    add EDI,4
loop repeat
endloop
```

```
S    dw    10, 11, 12, 13, 14, 15
len EQU ($ - S) \ 2
two db 2
; compute the sum of odd numbers
  mov bx, 0 ; initialize the sum
  mov ESi, S
  mov ecx, len
  jecxz endloop
  Sumloop
        mov  ax, [ESi]
        div byte [two] ; AL = ax / 2 ; AH = AX % 2
        cmp ah, 1
        jne dontadd
              add bx, [ESi]
        dontadd
        add Esi, 2
  loop sumloop
  endloop.
```