cmp op1, op2 ---> op2 cond op1
j<cond> et

```
for(int i = 0; i < n; i++){
   //instr
}
```

```
movl $0, %ecx  # i = 0
et_loop:
cmp n, %ecx  # if i == n
je et_exit  # exit
//  instructiuni

add $1, %ecx  # i++
jmp et_loop  # loop

et_exit:
   ....
```

+++ TABLOURI UNIDIMENSIONALE DE DATE (ARRAY-URI)

array-urile au o zona bine delimitata in memorie start----------end
accesarea unui element la array-uri: O(1)
listele nu au o zona bine delimitata start----alte chestii---- element------alte chestii----element----end
accesarea unui element la liste: O(n)

+ Tablouri de elemente de tip long.
+ Declararea array-urilor
   v: .long 3, 5, 17, 27, 11, 10, 4 // long v[7] = {3, 5, 17, 27, 11, 10, 4};
   sau
   n: .long 15
   v: .space 60

+ Accesarea elementelor
v = adresa primului element
&3 = v + 0B
&5 = v + 4B
&17 = v + 8B
&(i) = v + i * 4B

a(b, c, d) reprezinta continutul aflat la adresa de memorie:
   a + b + c*d
   a NU poate fi registru
   b si c doar registru
   d doar constanta numerica
   aici totul e fara $
Daca a = 0, notam (b, c, d) si inseamna continutul de la adresa b + c * d
folosim esi si edi pt elemente, ecx pt index
ex1.s

cmp (%edi, %ecx, 4), ... nu o sa mearga, trb sa bag (edi, ecx, 4) intr-un registru

+++ APELUL PRINTF

printf("Am citit %d", x);
    %d = int
    %s = siruri
    %f = float
ex2.s

printf ia din stiva:
[top] printf, x, y
push y
push x
push $formatStr
call printf
pop %ebx
pop %ebx
pop %ebx

+ Fie v: .long 15, 3, 27, 10, 19, 2, 8 un array de dimensiune n: .long 7, declarat in .data
Sa se determine minimul din array, si sa se afiseze printr-un apel la printf