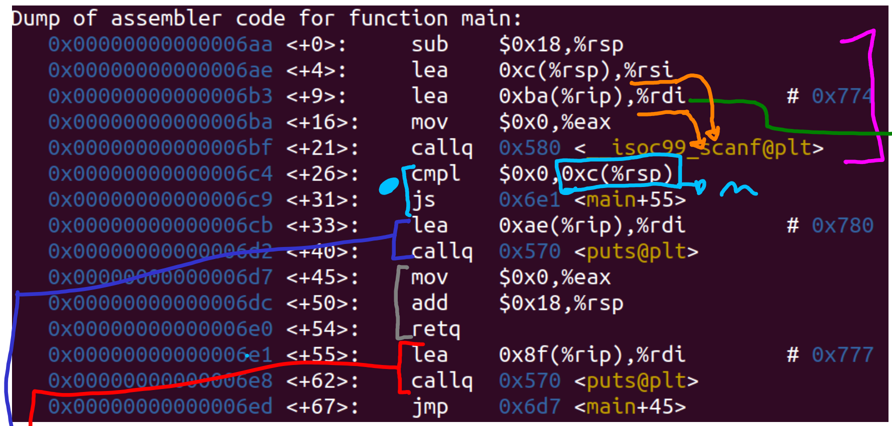


```
Dump of assembler code for function main:
   0x00000000000006aa <+0>:     sub    $0x18,%rsp
   0x00000000000006ae <+4>:     lea    0xc(%rsp),%rsi
   0x00000000000006b3 <+9>:     lea    0xba(%rip),%rdi        # 0x774
   0x00000000000006ba <+16>:    mov    $0x0,%eax
   0x00000000000006bf <+21>:    callq  0x580 <__isoc99_scanf@plt>
   0x00000000000006c4 <+26>:    cmpl   $0x0,0xc(%rsp)
   0x00000000000006c9 <+31>:    js     0x6e1 <main+55>
   0x00000000000006cb <+33>:    lea    0xae(%rip),%rdi        # 0x780
   0x00000000000006d2 <+40>:    callq  0x570 <puts@plt>
   0x00000000000006d7 <+45>:    mov    $0x0,%eax
   0x00000000000006dc <+50>:    add    $0x18,%rsp
   0x00000000000006e0 <+54>:    retq
   0x00000000000006e1 <+55>:    lea    0x8f(%rip),%rdi        # 0x777
   0x00000000000006e8 <+62>:    callq  0x570 <puts@plt>
   0x00000000000006ed <+67>:    jmp    0x6d7 <main+45>
```

→ n

int n;
SCANF("%d", &n);

CHAR FORMAT[3]="%d"

→ & FORMAT[0]

CPU

RSI

0x10C

8 bytes

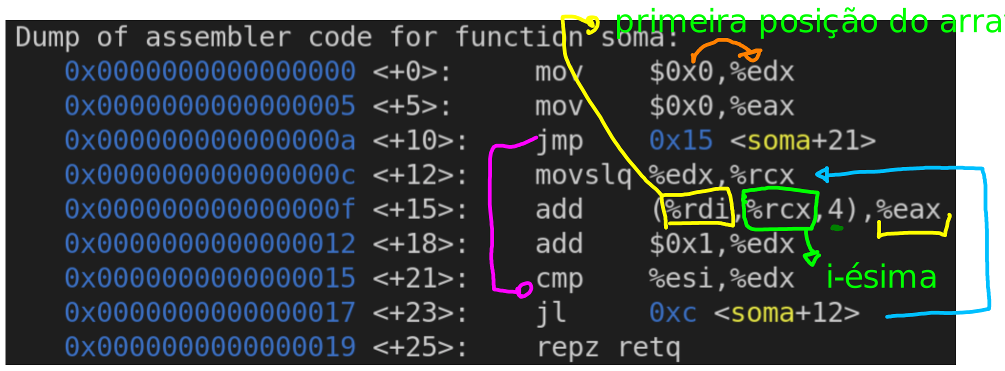
RSP = 0x100 + 0xC = 0x10C

```
int n;
SCANF("%d", &n);
• if ( n-0 < 0) {
      GOTO NEG;
  }
PRINTF("Positivo");
FINAL:
    RETURN 0;
NEG:
    PRINTF("NEGATIVO");
    GOTO FINAL;
```

0x774

%	d	\0

```
int main() {
   int n;
   scanf("%d", &n);
   if (n > 0) {
      printf("Positivo");
   } else {
      printf("Negativo");
   }
    return 0;
}
```



```
Dump of assembler code for function soma:
   0x0000000000000000 <+0>:     mov    $0x0,%edx
   0x0000000000000005 <+5>:     mov    $0x0,%eax
   0x000000000000000a <+10>:    jmp    0x15 <soma+21>
   0x000000000000000c <+12>:    movslq %edx,%rcx
   0x000000000000000f <+15>:    add    (%rdi,%rcx,4),%eax
   0x0000000000000012 <+18>:    add    $0x1,%edx
   0x0000000000000015 <+21>:    cmp    %esi,%edx
   0x0000000000000017 <+23>:    jl     0xc <soma+12>
   0x0000000000000019 <+25>:    repz retq
```

EDX → INT a; RCX
EAX → INT RES;
} LOCAIS

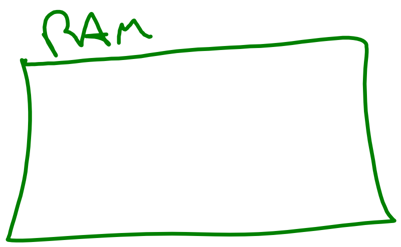
ESI → INT b; (PARAM)

RDI → INT * VEC

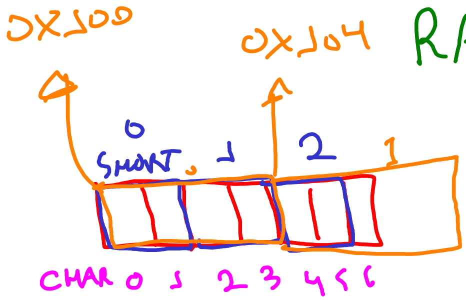
```
int soma(int *vec, int b) {
    int a = 0;
    int res = 0;

    goto verifica;

    faz_soma:
      res += vec[a];
      a ++;

     verifica:

         if (a < b) {
            goto faz_soma;
         }
         return res;

  }
```

RDI = 0x100
RCX = 0x1

EAX += RDI [RCX]



RAM [RDI + 4*RCX]



```
int soma(int *vec, int n) {
    int sum = 0;
    for (int i = 0; i < n; i++) {
        sum += vec[i];
    }
     return sum;
}
```

```
int soma(int *vec, int n) {
    int sum = 0;
    int i = 0;
    while (i < n) {
        sum += vec[i];
        i++;
    }
     return sum;
}
```