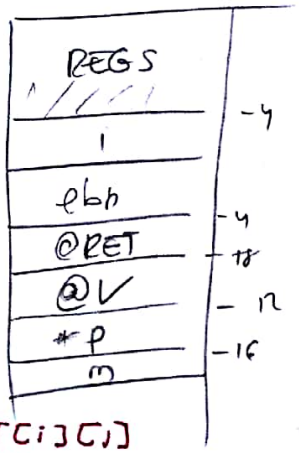
```
int Exa (int V[], int x)

int XProb(int V[], int *p, int m){
    int i;
    for (i=0; i<1000000, i++){
     V[i] += Exa (V, *p);
    return *p + m;
{
```

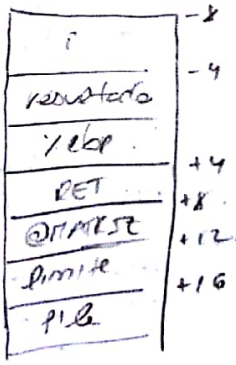


MAT[i][j]
@M + (i*NC+j)*4

BCC -m32 -O xxx xxx.c xxx.S

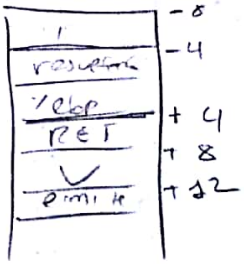
```
Xprob3: pushl %ebp
        movl %esp,%ebp
        subl $4, %esp TAM. V. locales
        pushl %esi  (si se usan se salvan
        pushl %ebx
        movl 8(%ebp), %ebx @V
        movl $0, %esi   #i=0
for:    cmpl $1000000, %esi
        jz endfor
        movl 12(%ebp), %eax # *p
        pushl (%eax)  valor p en pila   | llamada
        pushl %ebx    @V a pila         | Exa
        call Exa
        addl $8, %esp num param subrutina
        addl %eax, (%ebx, %esi, 4)  @V + i*4
        incl %esi  ++i
        jmp for
endfor: movl $12(%ebp), %eax  #*p
        movl (%eax), %eax  # valor p
        addl 16(%ebp), %eax  *p + m   %eax resultado
        pop %ebx
        pop %esi
        movl %ebp, %esp
        pop %ebp
        ret.
```

```
int Rutina(int Matriz[3][20], int limite, int pila){
    int i, resultado;
    resultado = 0;
    for(i=1; i<=limite; i++)
        if (matriz[pila][i]<=5)
            resultado++;

    return resultado;
{
```




```
rutina: pushl %ebp
        movl %esp, %ebp
        subl $8, %esp
        pushl %esi
        pushl %ebx

        movl 8(%ebp), %ebx #@MAT
        movl $1, %ecx #i
        movl $0, -4(%ebp) #resultado = 0
for:    cmpl 12(%ebp), %ecx
        jge fifor
        movl 16(%ebp) %edx   i<limite
        imull $20, %edx, %edx   i*NC
        addl %ecx, %edx   i*NC+j
        movl (%ebx, %edx, 4), %edx  @MAT+(i*NC+j)*T
if:     cmpl $5, %edx   mat[pila][i] <=5
        jg endif
        incl -4(%ebp)  # ++resultado
endif:  incl %ecx ++i
        jmp for
fifor:  movl -4(%ebp), %eax  guardo resultado en %eax
        pop %ebx
        pop %esi
        movl %ebp, %esp
        pop %ebp
        ret
```

```
int Rutina(V[20], limite){
    int i, resultado;
    i = 1;
    resultado = 0;
    while (i < limite){
        if (V[i] != 5)
            calle(&resultado);
        {
    {   ++i;
    return resultado;
{
```




```
rutina: pushl %ebp
        movl %esp, %ebp
        subl $8, %esp
        pushl %esi
        pushl %ebx

        movl $1, %ecx  i
        movl $0, -4(%ebp)  #res=0
        movl 8(%ebp), %ebx
while:
        cmpl 12(%ebp), %ecx  #i<limite
        jge endwhile
if:     movl (%ebx, %ecx, 4), %edx  #V[i]
        cmpl $5, %edx  #V[i] != 5 ?
        je endif
        pushl -4(%ebp)  # resultado
        call calle
        addl $4, %esp   #num param subrutina
endif:  incl %ecx
        jmp while  #++i

endwhile:
        movl -4(%ebp) %eax
        pop %ebx
        pop %esi
        movl %ebp, %esi
        pop %ebp
        ret.
```

```
int calcula (int M[][10], int m, int n){
    int i, suma, fila;
    suma = 0;
    fila = 0;
    for(i = m; i < n; i++){
        suma = suma + Normaliza(M[Alo][i], &fila);
    return (suma+1);
}
```

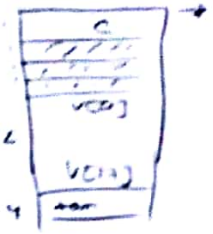
```
calcula:
    pushl %ebp
    movl %esp, %ebp
    subl $12, %esp          var. locales
    pushl %ebx
    movl $0, -8(%ebp)       suma = 0
    movl $0, -4(%ebp)       fila = 0
    movl 12(%ebp), %ebx     ebx = i = m
for:
    cmpl 16(%ebp), %ebx
    jge fifor
    leal -4(%ebp), %eax     eax = &fila
    pushl %eax              por hacer call
    movl -4(%ebp), %edx     fila
    imull $10, %edx         fila * 10(DEC)
    addl %ebx, %edx         (fila * 10) + i
    movl 8(%ebp), %ecx      @M
    movl (%ecx,%edx,4), %eax    @M + (fila*10+i)*4
    pushl %edx              por call
    call Normaliza
    addl $8, %esp           # num pasan pila
fifor: addl %eax, -8(%ebp), %eax    suma + Normaliza(M[Alo][i])
    incl %ebx               ++i;
    jmp for

    movl -8(%ebp), %eax     suma
    incl %eax               suma+1
    popl %ebx
    movl %ebp, %esp
    popl %ebp
    ret
```

32: eax, ebx, ecx, edx, edi, esi
16: ax, bx, cx, dx, di, si
8: ah, al, bh, bl, ch, cl, dh, dl

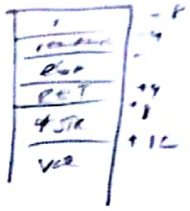
Guardamos ebx, esi, edi
No hace falta: eax, ecx, edx
Resultado: eax

tam: variable: múltiplo 4.

```
typedef struct {
    char c;
    int v[18];
    int tam;
} S;
```



```
int Rutina (S *str, int val){
    int i, resultado;
    resultado = 0; i = 0;
    do{
        if(str->v[i] >= val - 14){
            ++resultado; }
        ++i
    }while(i < str->tam);
    return resultado;
}
```



```
Rutina:
        pushl %ebp
        movl %esp, %ebp
        subl $8, %esp
        pushl %ebx
        pushl %esi
        movl $0, -4(%ebp)       res = 0
        movl $0, %ecx           i = 0
do
if:
        movl 8(%ebp), %ebx      @str
        movl 4(%ebp,%ecx,4), %ebx   str->v[i]
        movl $-14, %esi
        addl 12(%ebp), %esi     val-14
        cmpl %esi, %ebx
        jg endif

        incl -4(%ebp)
endif:  incl %ecx
while:
        movl 8(%ebp), %ebx      @str
        cmpl $76(%ebx), %ecx    str->tam
        jl do
        movl -4(%ebp), %ecx     # en resul
        popl %esi
        popl %ebx
        movl %ebp, %esp
        pop %ebp
        ret
```

je	jne	js (negativo)	jns
jg	jge	jl	jle
ja	jae	jb	jbe
> / >=		< / <=	
sin signo		sin signo	