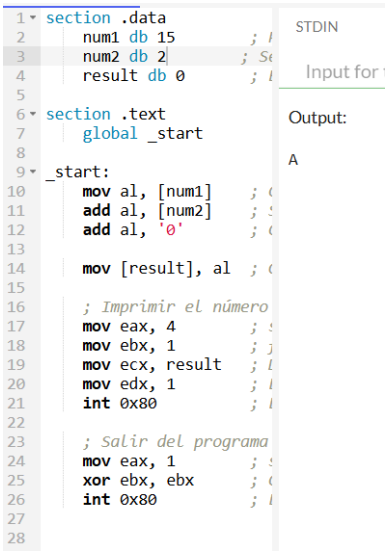
6. De acuerdo al código ensamblador anexo (también lo puedes encontrar aqui:

a. Modifica el código para que imprima los siguientes caracteres utilizando solo sumas:

i. A



```
section .data
    num1 db 15
    num2 db 2
    result db 0

section .text
    global _start

_start:
    mov al, [num1]
    add al, [num2]
    add al, '0'

    mov [result], al

    ; Imprimir el número
    mov eax, 4
    mov ebx, 1
    mov ecx, result
    mov edx, 1
    int 0x80

    ; Salir del programa
    mov eax, 1
    xor ebx, ebx
    int 0x80
```

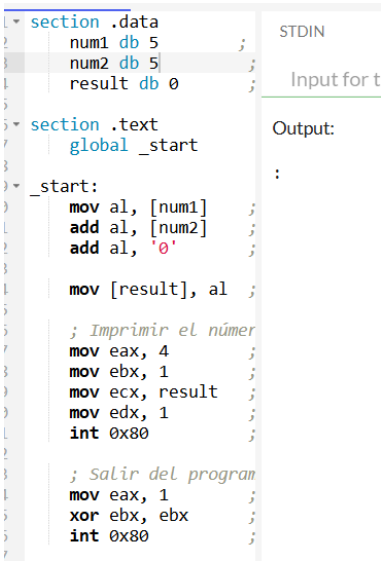
STDIN

Input for

Output:

A

ii. :



```
section .data
    num1 db 5
    num2 db 5
    result db 0

section .text
    global _start

_start:
    mov al, [num1]
    add al, [num2]
    add al, '0'

    mov [result], al

    ; Imprimir el número
    mov eax, 4
    mov ebx, 1
    mov ecx, result
    mov edx, 1
    int 0x80

    ; Salir del programa
    mov eax, 1
    xor ebx, ebx
    int 0x80
```

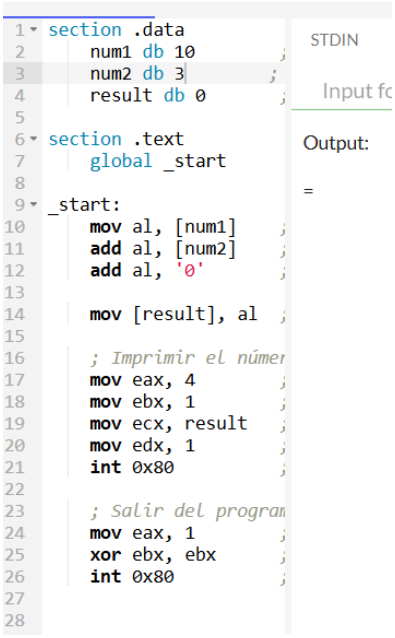
STDIN

Input for t

Output:

:

iii. =



```
section .data
    num1 db 10
    num2 db 3
    result db 0

section .text
    global _start

_start:
    mov al, [num1]
    add al, [num2]
    add al, '0'

    mov [result], al

    ; Imprimir el número
    mov eax, 4
    mov ebx, 1
    mov ecx, result
    mov edx, 1
    int 0x80

    ; Salir del programa
    mov eax, 1
    xor ebx, ebx
    int 0x80
```

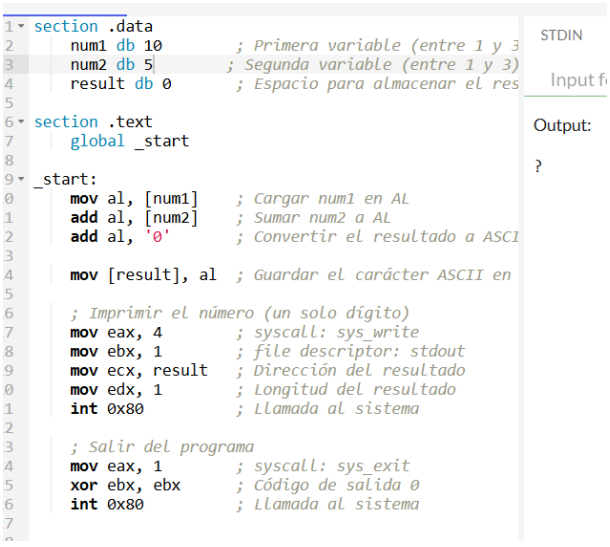
STDIN

Input fo

Output:

=

iv. ?



```
section .data
    num1 db 10          ; Primera variable (entre 1 y 3
    num2 db 5           ; Segunda variable (entre 1 y 3)
    result db 0         ; Espacio para almacenar el res

section .text
    global _start

_start:
    mov al, [num1]      ; Cargar num1 en AL
    add al, [num2]      ; Sumar num2 a AL
    add al, '0'         ; Convertir el resultado a ASCI

    mov [result], al    ; Guardar el carácter ASCII en

    ; Imprimir el número (un solo dígito)
    mov eax, 4          ; syscall: sys_write
    mov ebx, 1          ; file descriptor: stdout
    mov ecx, result     ; Dirección del resultado
    mov edx, 1          ; Longitud del resultado
    int 0x80            ; Llamada al sistema

    ; Salir del programa
    mov eax, 1          ; syscall: sys_exit
    xor ebx, ebx        ; Código de salida 0
    int 0x80            ; Llamada al sistema
```

STDIN

Input f

Output:

?

V. _

```
section .data
    num1 db 10
    num2 db 37
    result db 0

section .text
    global _start

_start:
    mov al, [num1]
    add al, [num2]
    add al, '0'

    mov [result], al

    ; Imprimir el núm
    mov eax, 4
    mov ebx, 1
    mov ecx, result
    mov edx, 1
    int 0x80

    ; Salir del progr
    mov eax, 1
    xor ebx, ebx
    int 0x80
```

STDIN

Input for the

Output:

_

b. Ahora modificarlo para imprima los siguientes caracteres utilizando al menos una resta dentro del código:

i. B

```
section .data
    num1 db 48
    num2 db 30
    result db 0

section .text
    global _start

_start:

    mov al, [num1]
    sub al, [num2]
    add al, '0'

    mov [result], al

    ; Imprimir el número
    mov eax, 4
    mov ebx, 1
    mov ecx, result
    mov edx, 1
    int 0x80

    ; Salir del programa
    mov eax, 1
    xor ebx, ebx
    int 0x80
```

STDIN

Input for the progr

Output:

B

ii. x

```
section .data
    num1 db 122
    num2 db 2
    num3 db 48
    result db 0

section .text
    global _start

_start:
    mov al, [num1]
    sub al, [num2]
    sub al, [num3]
    add al, '0'

    mov [result], al

    ; Imprimir el núme
    mov eax, 4
    mov ebx, 1
    mov ecx, result
    mov edx, 1
    int 0x80

    ; Salir del progra
    mov eax, 1
    xor ebx, ebx
    int 0x80
```

STDIN

Input

Output:

x

iii. +

```
section .data
    num1 db 15
    num2 db 20
    result db 0

section .text
    global _start

_start:

    mov al, [num1]
    sub al, [num2]
    add al, '0'

    mov [result], al

    ; Imprimir el núme
    mov eax, 4
    mov ebx, 1
    mov ecx, result
    mov edx, 1
    int 0x80

    ; Salir del progra
    mov eax, 1
    xor ebx, ebx
    int 0x80
```

STDIN

Input f

Output:

+

iv. ‘

```
section .data
    num1 db 40
    num2 db 1
    num3 db 48
    result db 0

section .text
    global _start

_start:
    mov al, [num1]
    sub al, [num2]
    sub al, [num3]
    add al, '0'

    mov [result], al

    ; Imprimir el núme
    mov eax, 4
    mov ebx, 1
    mov ecx, result
    mov edx, 1
    int 0x80

    ; Salir del progra
    mov eax, 1
    xor ebx, ebx
    int 0x80
```

STDIN

Input for th

Output:

‘

v. {

```
1  section .data
2      num1 db 124
3      num2 db 1
4      num3 db 48
5      result db 0
6
7  section .text
8      global _start
9
10 _start:
11     mov al, [num1]
12     sub al, [num2]
13     sub al, [num3]
14     add al, '0'
15
16     mov [result], al
17
18     ; Imprimir el núme
19     mov eax, 4
20     mov ebx, 1
21     mov ecx, result
22     mov edx, 1
23     int 0x80
24
25     ; Salir del progra
26     mov eax, 1
27     xor ebx, ebx
28     int 0x80
```

STDIN

Input for

Output:

{