

```

; find if no. if +,-,0

section .data
    msg1 db "Enter no.: ", 0xa, 0xd
    len1 equ $-msg1
    msg2 db "The no. is +ve", 0xa, 0xd
    len2 equ $-msg2
    msg3 db "The no. is -ve", 0xa, 0xd
    len3 equ $-msg3
    msg4 db "The no. is invalid", 0xa, 0xd
    len4 equ $-msg4
    msg5 db "The no. is zero", 0xa, 0xd
    len5 equ $-msg5

section .bss
    num resb 10

section .text
global _start
_start:
    ; display msg1
    mov eax, 4
    mov ebx, 1
    mov ecx, msg1
    mov edx, len1
    int 80h

    ; taking input from user
    mov eax, 3
    mov ebx, 0
    mov ecx, num
    mov edx, 10
    int 80h

    mov eax, [num]
    cmp al, '+'; compare if no. is +ve
    je positive ; jump if no. is +ve

    cmp al, '-'; compare if the no. is -ve
    je negative; jump if no. is negative

    cmp al, '0'

```

```
je zero
```

```
cmp al, 39h ; compare if no. is valid
```

```
jae invalid; jump if no. is invalid
```

```
positive:
```

```
    mov eax, 4
```

```
    mov ebx, 1
```

```
    mov ecx, msg2
```

```
    mov edx, len2
```

```
    int 80h
```

```
    jmp exit;
```

```
negative:
```

```
    mov eax,4
```

```
    mov ebx, 1
```

```
    mov ecx, msg3
```

```
    mov edx, len3
```

```
    int 80h
```

```
    jmp exit
```

```
invalid:
```

```
    mov eax, 4
```

```
    mov ebx, 1
```

```
    mov ecx, msg4
```

```
    mov edx, len4
```

```
    int 80h
```

```
    jmp exit
```

```
zero:
```

```
    mov eax, 4
```

```
    mov ebx, 1
```

```
    mov ecx, msg5
```

```
    mov edx, len5
```

```
    int 80h
```

```
    jmp exit
```

```
exit:
```

```
    mov eax, 1
```

```
    mov ebx, 0
```

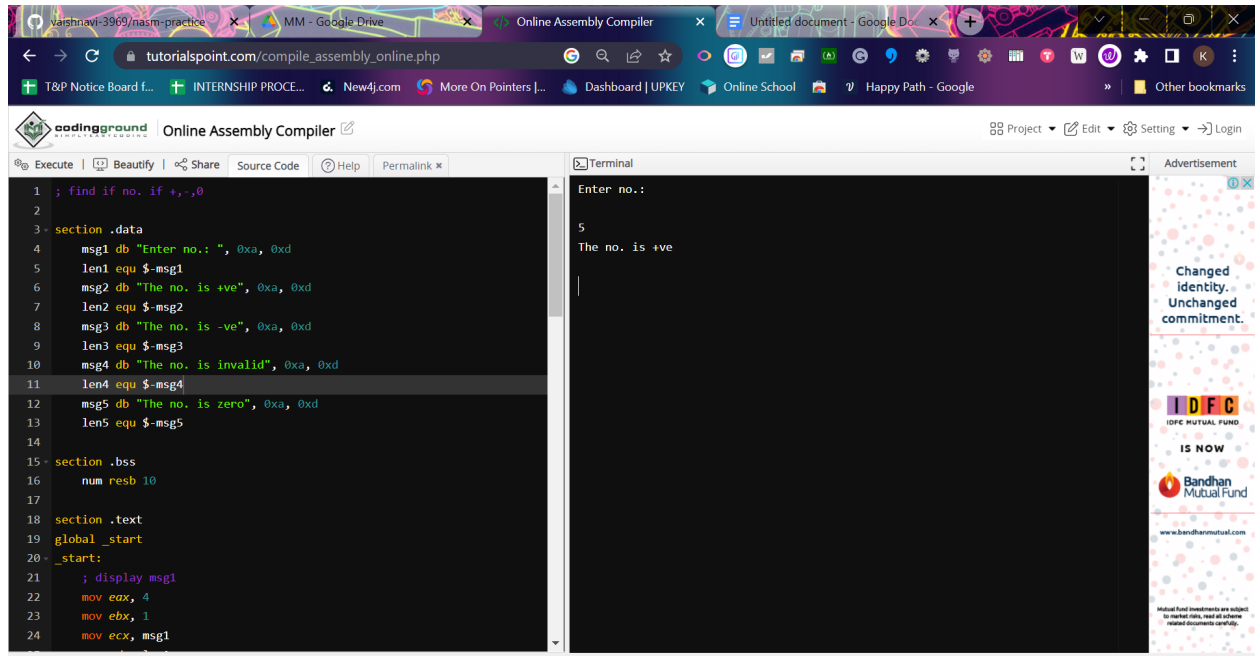
```
    int 80h
```

```
; -----
```

```
mov eax, 1
```

```
mov ebx, 0
```

```
int 80h
```



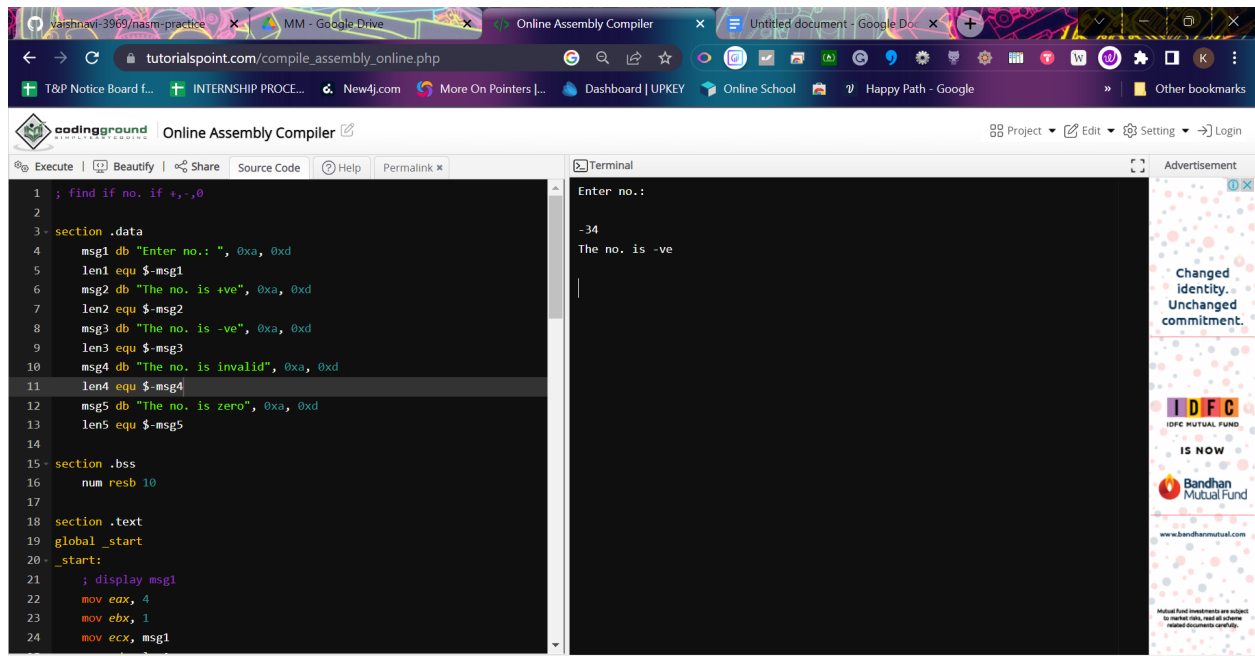
The screenshot shows the Online Assembly Compiler interface. The code editor contains the following assembly code:

```
1 ; find if no. if +,-,0
2
3 section .data
4 msg1 db "Enter no.: ", 0xa, 0xd
5 len1 equ $-msg1
6 msg2 db "The no. is +ve", 0xa, 0xd
7 len2 equ $-msg2
8 msg3 db "The no. is -ve", 0xa, 0xd
9 len3 equ $-msg3
10 msg4 db "The no. is invalid", 0xa, 0xd
11 len4 equ $-msg4
12 msg5 db "The no. is zero", 0xa, 0xd
13 len5 equ $-msg5
14
15 section .bss
16 num resb 10
17
18 section .text
19 global _start
20 _start:
21 ; display msg1
22 mov eax, 4
23 mov ebx, 1
24 mov ecx, msg1
```

The terminal window shows the following output:

```
Enter no.:
5
The no. is +ve
```

The interface includes a top navigation bar with links like 'Execute', 'Beautify', 'Share', 'Source Code', 'Help', and 'Permalink'. The right sidebar contains an advertisement for IDFC Mutual Fund and Bandhan Mutual Fund.



The screenshot shows the Online Assembly Compiler interface with the same assembly code as the previous image. The terminal window shows the following output:

```
Enter no.:
-34
The no. is -ve
```

The interface is identical to the previous screenshot, showing the same code editor, terminal, and navigation elements.

The screenshot shows the Online Assembly Compiler interface. The assembly code on the left includes a message box for "Enter no.:", a loop to take input from the user, and logic to check if the input is zero. The terminal on the right shows the prompt "Enter no.:" followed by the user input "0" and the output "The no. is zero".

```
21 ; display msg1
22 mov eax, 4
23 mov ebx, 1
24 mov ecx, msg1
25 mov edx, len1
26 int 80h
27
28 ; taking input from user
29 mov eax, 3
30 mov ebx, 0
31 mov ecx, num
32 mov edx, 10
33 int 80h
34
35 mov eax, [num]
36 cmp al, '+' ; compare if no. is +ve
37 je positive ; jump is no. is +ve
38
39 cmp al, '-' ; compare if the no. is -ve
40 je negative ; jump if no. is negative
41
42 cmp al, '0'
43 je zero
44
45 cmp al, 39h ; compare if no. is valid
```

Terminal output:

```
Enter no.:
0
The no. is zero
```

The screenshot shows the Online Assembly Compiler interface. The assembly code on the left is identical to the previous one. The terminal on the right shows the prompt "Enter no.:" followed by the user input "adfad" and the output "The no. is invalid".

```
21 ; display msg1
22 mov eax, 4
23 mov ebx, 1
24 mov ecx, msg1
25 mov edx, len1
26 int 80h
27
28 ; taking input from user
29 mov eax, 3
30 mov ebx, 0
31 mov ecx, num
32 mov edx, 10
33 int 80h
34
35 mov eax, [num]
36 cmp al, '+' ; compare if no. is +ve
37 je positive ; jump is no. is +ve
38
39 cmp al, '-' ; compare if the no. is -ve
40 je negative ; jump if no. is negative
41
42 cmp al, '0'
43 je zero
44
45 cmp al, 39h ; compare if no. is valid
```

Terminal output:

```
Enter no.:
adfad
The no. is invalid
```

section .data

msg_input db 'Enter no.: ', 10

len_input equ \$-msg_input

msg_positive db "The no. is positive", 10

len_positive equ \$-msg_positive

msg_negative db "The no. is negative", 10

len_negative equ \$-msg_negative

```
msg_zero db "The no. is zero", 10
len_zero equ $-msg_zero
msg_invalid db "The no. is invalid", 10
len_invalid equ $-msg_invalid
```

```
section .bss
    num resb 10
```

```
section .data
```

```
global _start
```

```
_start:
```

```
    ; displaying msg
    mov eax, 4
    mov ebx, 1
    mov ecx, msg_input
    mov edx, len_input
    int 80h
```

```
    ; taking input
    mov eax, 3
    mov ebx, 0
    mov ecx, num
    mov edx, 10
    int 80h
```

```
    ; performing operations
    mov eax, [num]
```

```
    cmp al, '+'
    je positive
```

```
    cmp al, '-'
    je negative
```

```
    cmp al, '0'
    je zero
```

```
    cmp al, '9'
    jae invalid
```

```
positive:
```

```
    mov eax, 4
    mov ebx, 1
    mov ecx, msg_positive
```

```
    mov edx, len_positive
    int 80h
    jmp exit
```

negative:

```
    mov eax, 4
    mov ebx, 1
    mov ecx, msg_negative
    mov edx, len_negative
    int 80h
    jmp exit
```

zero:

```
    mov eax, 4
    mov ebx, 1
    mov ecx, msg_zero
    mov edx, len_zero
    int 80h
    jmp exit
```

invalid:

```
    mov eax, 4
    mov ebx, 1
    mov ecx, msg_invalid
    mov edx, len_invalid
    int 80h
    jmp exit
```

exit:

```
    mov eax, 1
    mov ebx, 0
    int 80h
```

section .data

```
prompt db "Enter a number: ", 10, 0
len_prompt equ $-prompt
positive db "The number is positive.", 10, 0
len_positive equ $-positive
negative db "The number is negative.", 10, 0
len_negative equ $-negative
zero db "The number is zero.", 10, 0
len_zero equ $-zero
```

```
invalid db "Invalid input.", 10, 0
len_invalid equ $-invalid
```

```
section .bss
input resb 10
```

```
section .text
global _start
```

```
_start:
```

```
; Prompt the user for input
mov eax, 4
mov ebx, 1
mov ecx, prompt
mov edx, len_prompt
int 80h
```

```
; Read the input
mov eax, 3
mov ebx, 0
mov ecx, input
mov edx, 2
int 80h
```

```
; Convert input to a signed integer
mov eax,[input]
cmp al, '+'
je positive_result
cmp al, '-'
je negative_result
cmp al, '0'
je zero_result
cmp al, '9'
jae invalid_result
```

```
positive_result:
```

```
mov eax, 4
mov ebx, 1
mov ecx, positive
mov edx, len_positive
int 80h
jmp exit_program
```

```
negative_result:
```

```
mov eax, 4
mov ebx, 1
mov ecx, negative
mov edx, len_negative
int 80h
jmp exit_program
```

```
zero_result:
mov eax, 4
mov ebx, 1
mov ecx, zero
mov edx, len_zero
int 80h
jmp exit_program
```

```
invalid_result:
mov eax, 4
mov ebx, 1
mov ecx, invalid
mov edx, len_invalid
int 80h
jmp exit_program
```

```
exit_program:
; Exit program
mov eax, 1
mov ebx, 0
int 80h
```

```
mov eax, 1
mov ebx, 0
int 80h
```