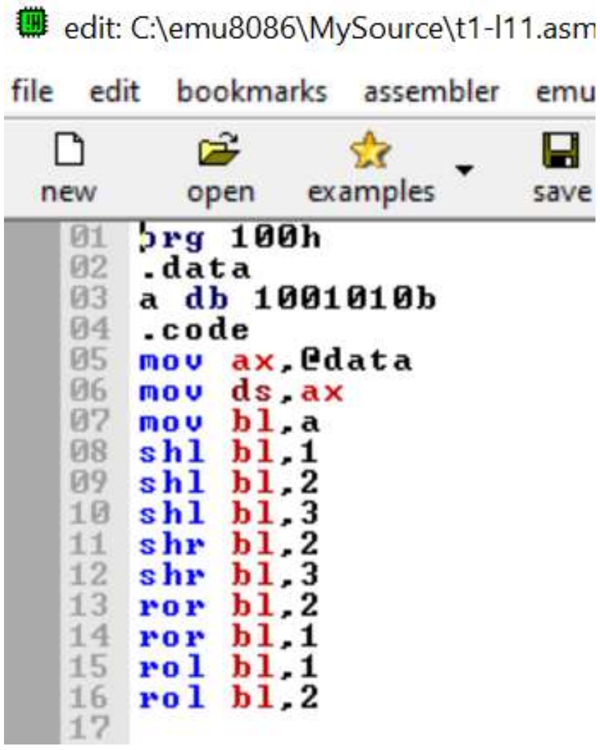
Task 1:

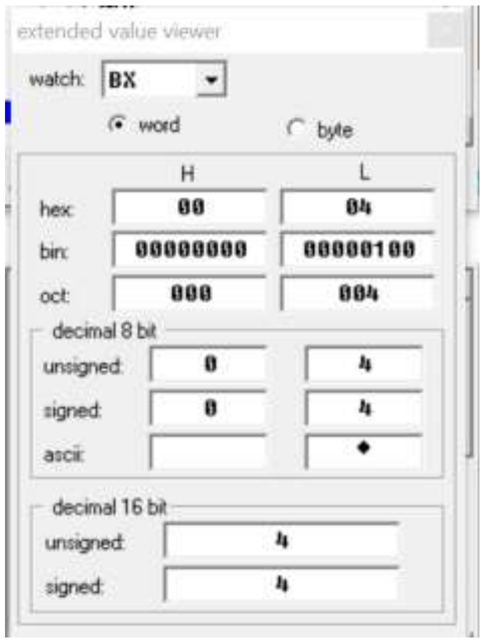


```
org 100h
.data
a db 1001010b
.code
mov ax,@data
mov ds,ax
mov bl,a
shl bl,1
shl bl,2
shl bl,3
shr bl,2
shr bl,3
ror bl,2
ror bl,1
rol bl,1
rol bl,2
```

Explanation:

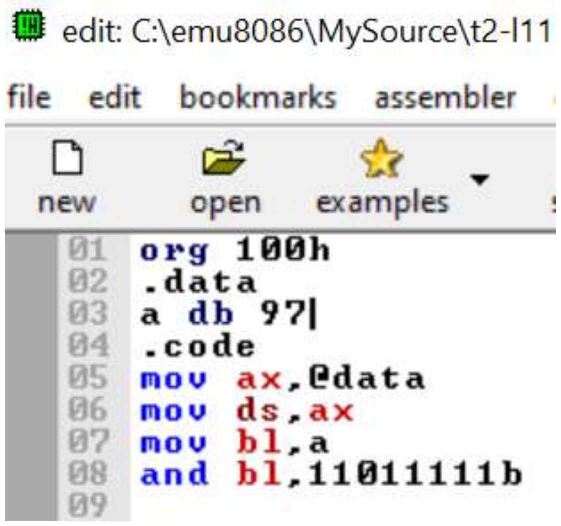
This program demonstrates the difference between **logical shifts** (SHL, SHR) and **rotations** (ROL, ROR).

Output:



```
        H          L
hex:    00         04
bin:    00000000   00000100
oct:    000        004
decimal 8 bit
unsigned:  0       4
signed:    0       4
ascii:             ♦
decimal 16 bit
unsigned:  4
signed:    4
```

Task 2:

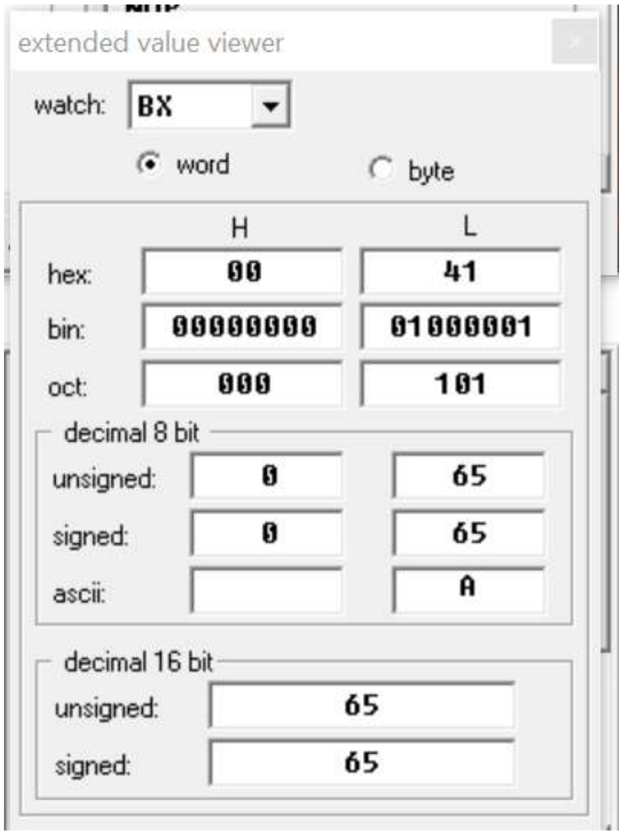


```
org 100h
.data
a db 97
.code
mov ax,@data
mov ds,ax
mov bl,a
and bl,11011111b
```

Explanation:

This short program takes the lowercase letter 'a' (ASCII 97) and converts it to uppercase 'A' (ASCII 65) using **bit masking**.

Output:



watch: BX

[✓ word] byte

	H	L
hex:	00	41
bin:	00000000	01000001
oct:	000	101

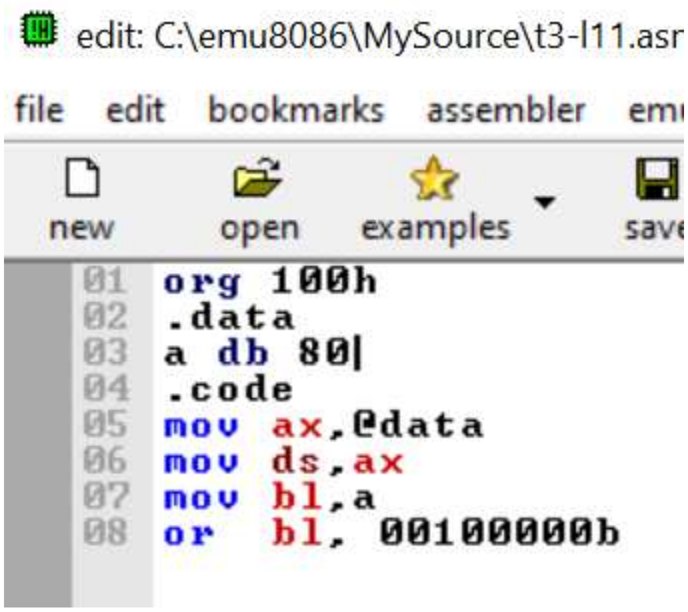
decimal 8 bit

unsigned:	0	65
signed:	0	65
ascii:		A

decimal 16 bit

unsigned:	65
signed:	65

Task 3:

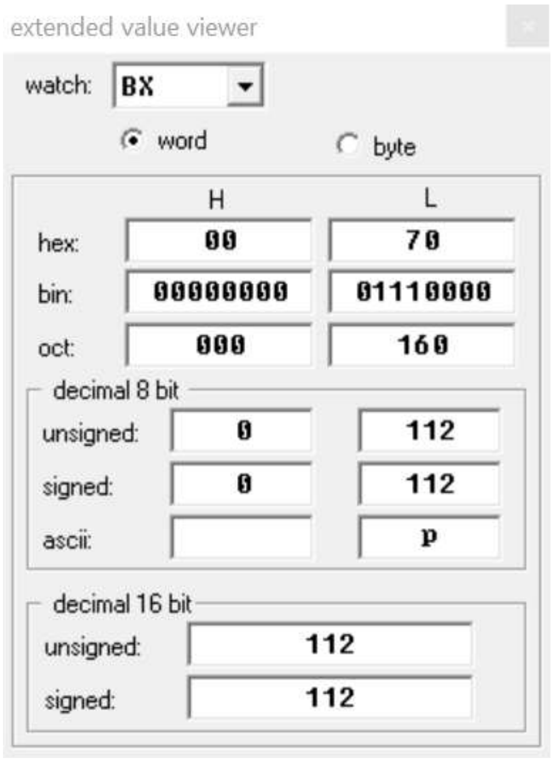

edit: C:\emu8086\MySource\t3-l11.asr
file edit bookmarks assembler emı
new open examples save

```
org 100h
.data
a db 80
.code
mov ax,@data
mov ds,ax
mov bl,a
or  bl, 00100000b
```

Explanation:

This program takes an uppercase letter 'P' and converts it to lowercase 'p' by **setting bit 5** with the OR instruction.

Output:



watch: BX

[✓ word] byte

	H	L
hex:	00	70
bin:	00000000	01110000
oct:	000	160

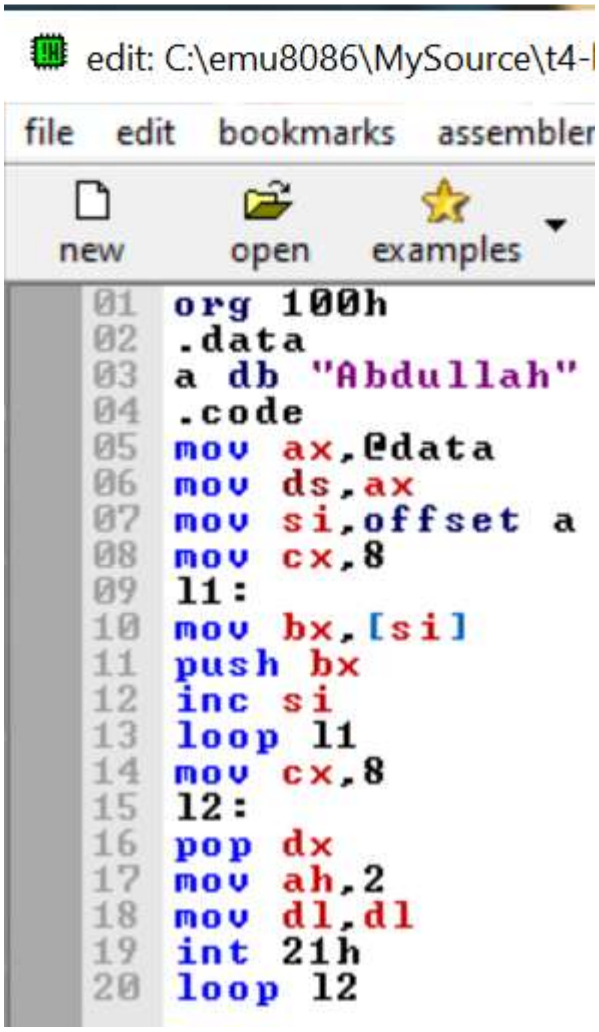
decimal 8 bit

unsigned:	0	112
signed:	0	112
ascii:		p

decimal 16 bit

unsigned:	112
signed:	112

Task 4:

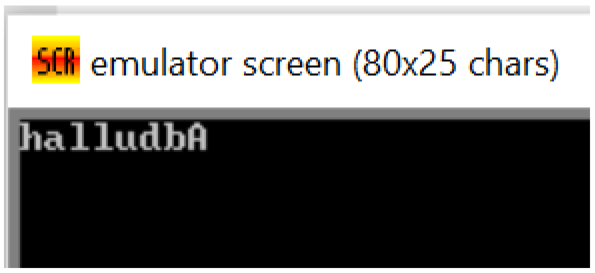


```
org 100h
.data
a db "Abdullah"
.code
mov ax,@data
mov ds,ax
mov si,offset a
mov cx,8
l1:
mov bx,[si]
push bx
inc si
loop l1
mov cx,8
l2:
pop dx
mov ah,2
mov dl,dl
int 21h
loop l2
```

Explanation:

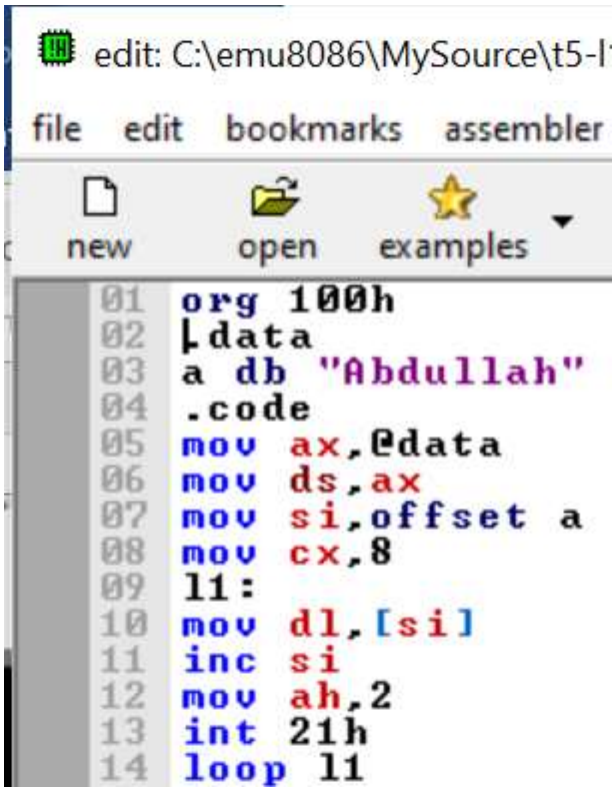
It’s a simple **string reversal and display** program using the **stack** and **DOS interrupt 21h (function 2)** for character output.

Output:



```
halludbA
```

Task 5:

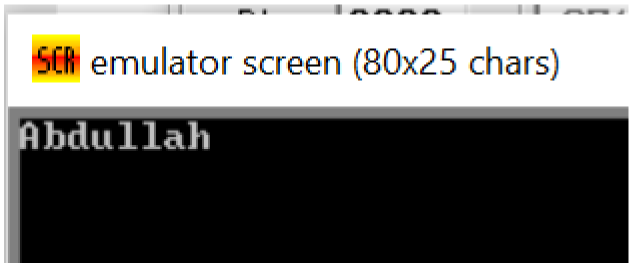


```
org 100h
.data
a db "Abdullah"
.code
mov ax,@data
mov ds,ax
mov si,offset a
mov cx,8
l1:
mov dl,[si]
inc si
mov ah,2
int 21h
loop l1
```

Explanation:

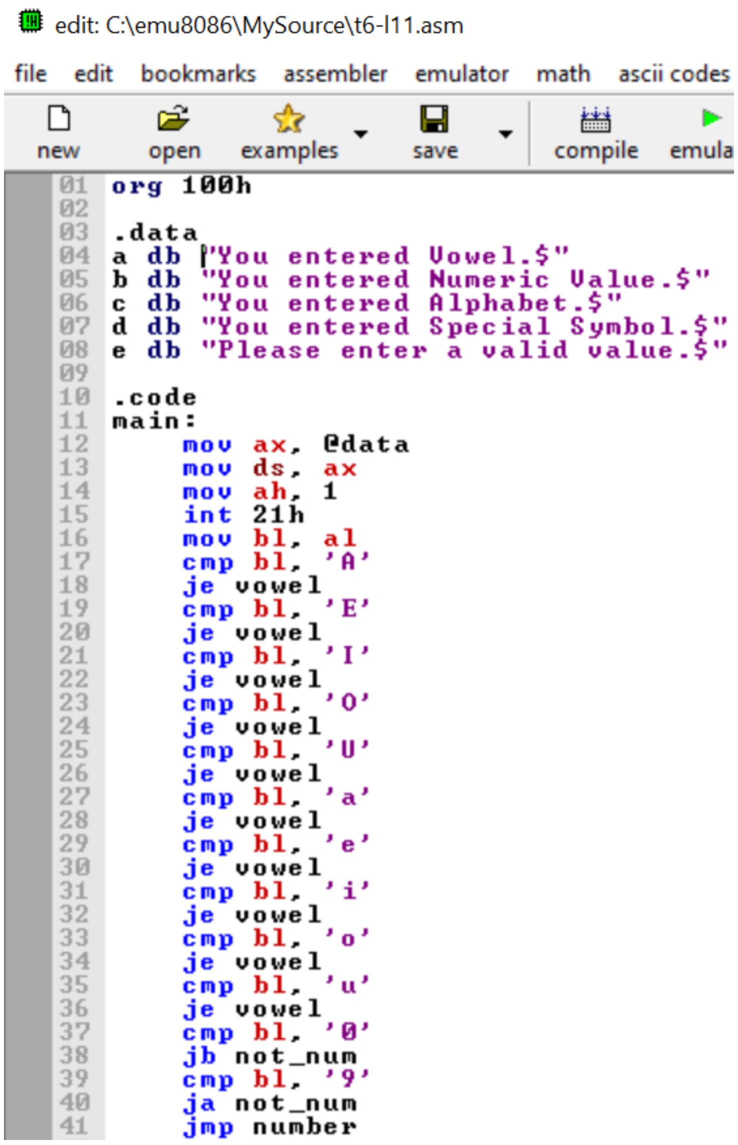
- Points to the start of the string.
- Loops four times (for each character).
- Loads each character into `DL`.
- Calls **INT 21h / AH=2** to display it.

Output:

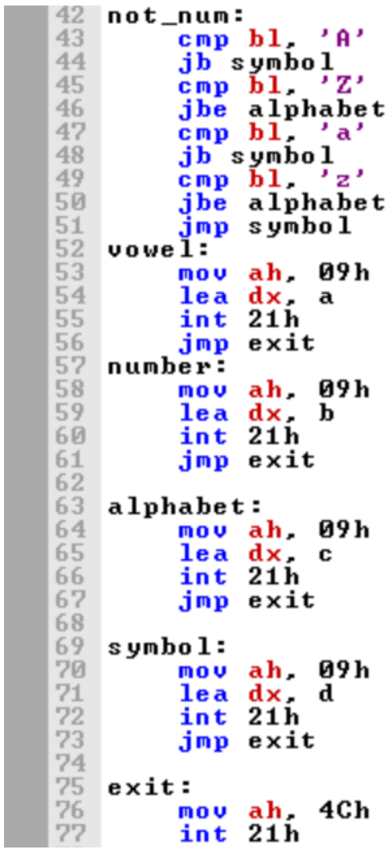


```
Abdullah
```

Task 6:


edit: C:\emu8086\MySource\t6-l11.asm
file edit bookmarks assembler emulator math ascii codes
new open examples save compile emula

```
org 100h

.data
a db "You entered Vowel.$"
b db "You entered Numeric Value.$"
c db "You entered Alphabet.$"
d db "You entered Special Symbol.$"
e db "Please enter a valid value.$"

.code
main:
        mov ax, @data
        mov ds, ax
        mov ah, 1
        int 21h
        mov bl, al
        cmp bl, 'A'
        je vowel
        cmp bl, 'E'
        je vowel
        cmp bl, 'I'
        je vowel
        cmp bl, 'O'
        je vowel
        cmp bl, 'U'
        je vowel
        cmp bl, 'a'
        je vowel
        cmp bl, 'e'
        je vowel
        cmp bl, 'i'
        je vowel
        cmp bl, 'o'
        je vowel
        cmp bl, 'u'
        je vowel
        cmp bl, '0'
        jb not_num
        cmp bl, '9'
        ja not_num
        jmp number
```

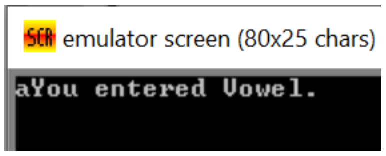


```
not_num:
      cmp bl, 'A'
      jb symbol
      cmp bl, 'Z'
      jbe alphabet
      cmp bl, 'a'
      jb symbol
      cmp bl, 'z'
      jbe alphabet
      jmp symbol
vowel:
      mov ah, 09h
      lea dx, a
      int 21h
      jmp exit
number:
      mov ah, 09h
      lea dx, b
      int 21h
      jmp exit

alphabet:
      mov ah, 09h
      lea dx, c
      int 21h
      jmp exit

symbol:
      mov ah, 09h
      lea dx, d
      int 21h
      jmp exit

exit:
      mov ah, 4Ch
      int 21h
```

Explanation:

1. **Takes one character input** from the user.
2. **Checks** whether it's:
   - a **vowel** (`A, E, I, O, U` or lowercase versions),
   - a **numeric digit** (`0-9`),
   - an **alphabet character** (A–Z or a–z),
   - or a **special symbol** (anything else).
3. **Displays** the appropriate message.

Output:



```
aYou entered Vowel.
```