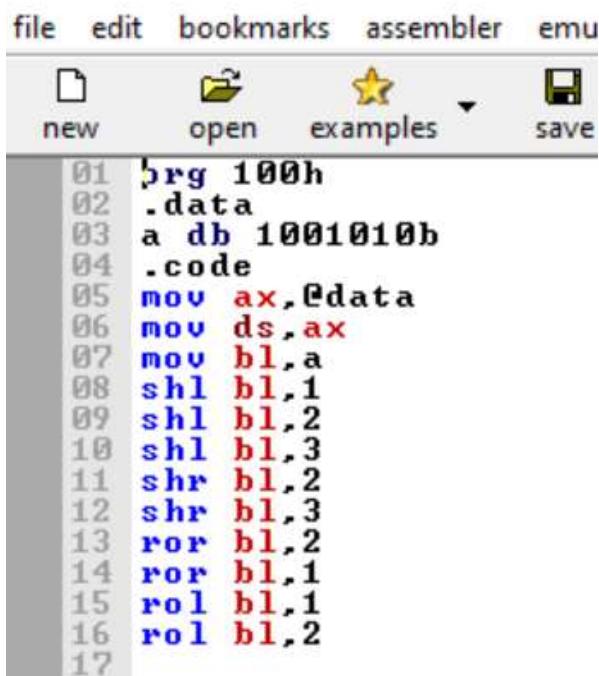


## Task 1:

edit: C:\emu8086\MySource\t1-l11.asm

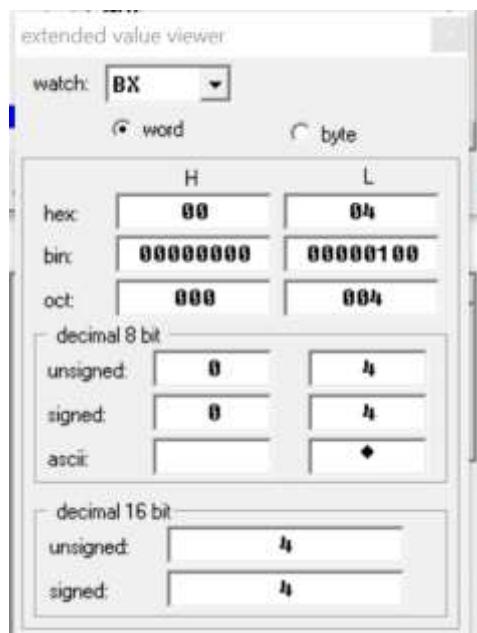


```
file edit bookmarks assembler emu
new open examples save
01 b8 100h
02 .data
03 a db 1001010b
04 .code
05 mov ax,@data
06 mov ds,ax
07 mov bl,a
08 shl bl,1
09 shl bl,2
10 shr bl,3
11 shr bl,2
12 shr bl,3
13 ror bl,2
14 ror bl,1
15 rol bl,1
16 rol bl,2
17
```

## Explanation:

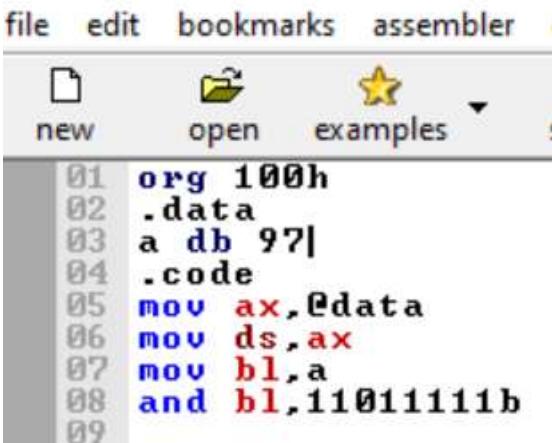
This program demonstrates the difference between **logical shifts** (SHL, SHR) and **rotations** (ROL, ROR).

## Output:



## Task 2:

edit: C:\emu8086\MySource\t2-l11

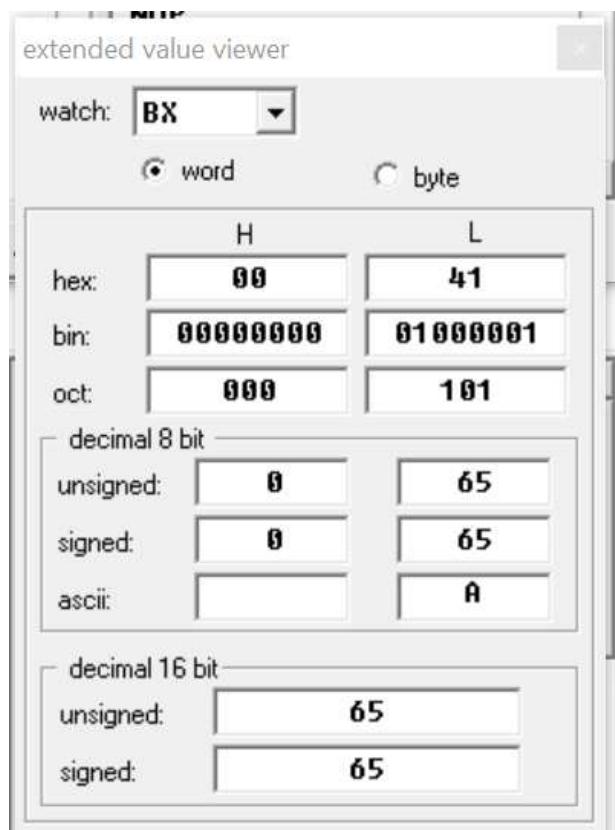


```
01 org 100h
02 .data
03 a db 97
04 .code
05 mov ax, @data
06 mov ds, ax
07 mov bl, a
08 and bl, 11011111b
09
```

### Explanation:

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

### Output:



### Task 3:

edit: C:\emu8086\MySource\t3-l11.asm

The screenshot shows the emu8086 assembler interface with the following assembly code in the editor window:

```
01 org 100h
02 .data
03 a db 80
04 .code
05 mov ax,@data
06 mov ds,ax
07 mov bl,a
08 or bl, 00100000b
```

### Explanation:

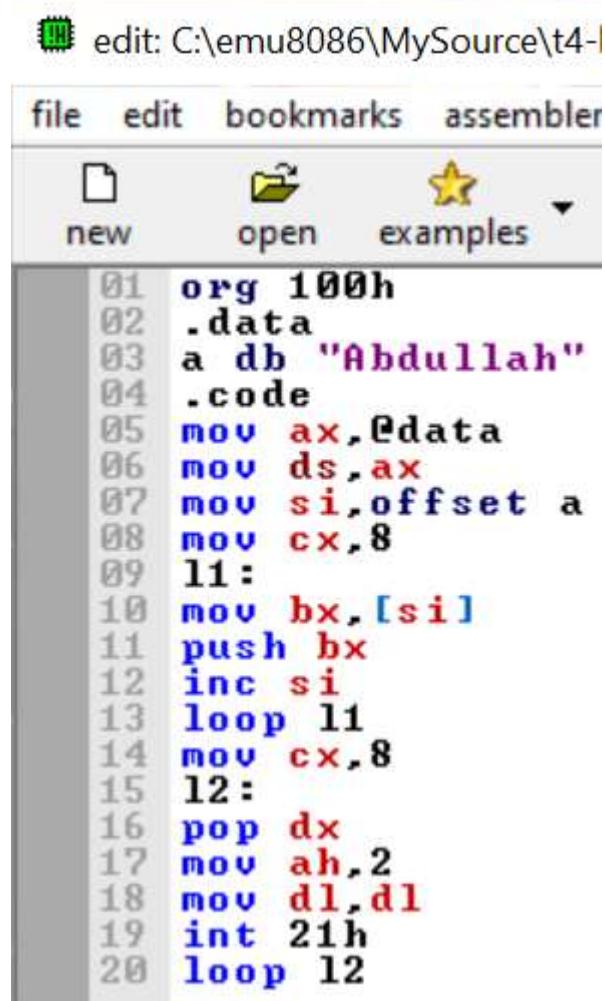
This program takes an uppercase letter 'P' and converts it to lowercase 'p' by **setting bit 5** with the OR instruction.

### Output:

The screenshot shows the extended value viewer window with the register **BX** selected. The window displays the following values for register BX:

	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:



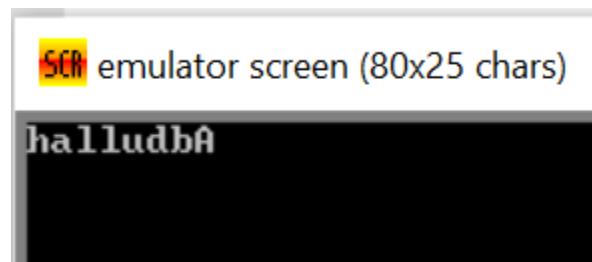
The screenshot shows the emu8086 assembly editor interface. The menu bar includes 'file', 'edit', 'bookmarks', and 'assembler'. The toolbar has icons for 'new', 'open', and 'examples'. The assembly code in the main window is as follows:

```
01 org 100h
02 .data
03 a db "Abdullah"
04 .code
05 mov ax, @data
06 mov ds, ax
07 mov si, offset a
08 mov cx, 8
09 l1:
10 mov bx, [si]
11 push bx
12 inc si
13 loop l1
14 mov cx, 8
15 l2:
16 pop dx
17 mov ah, 2
18 mov dl, dl
19 int 21h
20 loop l2
```

#### Explanation:

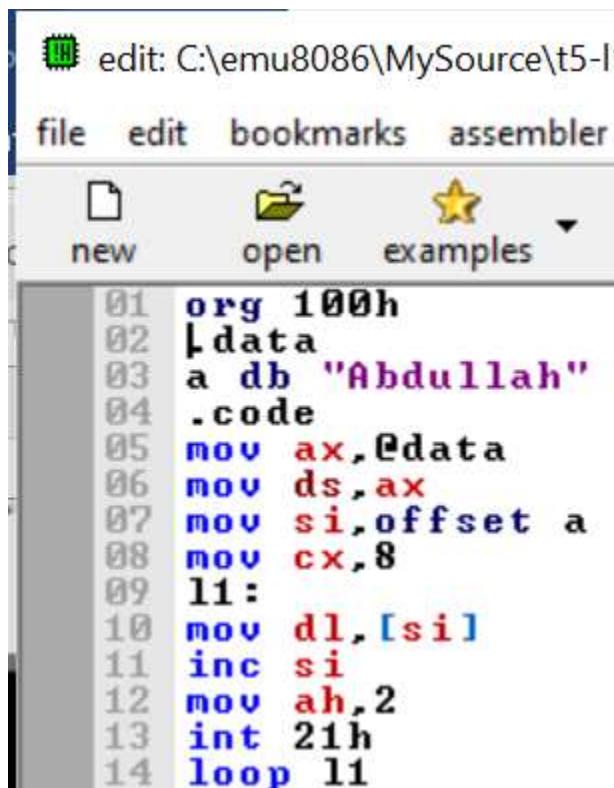
It's a simple **string reversal and display** program using the **stack** and **DOS interrupt 21h (function 2)** for character output.

#### Output:



The screenshot shows the emulator screen with the title 'SCR emulator screen (80x25 chars)'. The screen displays the reversed string 'halludbA'.

### Task 5:



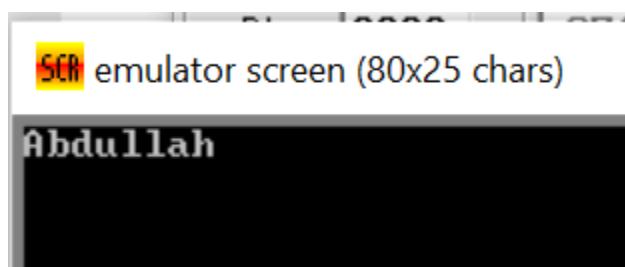
The screenshot shows the emu8086 assembly editor interface. The menu bar includes 'file', 'edit', 'bookmarks', and 'assembler'. The toolbar has icons for 'new', 'open', and 'examples'. The assembly code in the main window is as follows:

```
01 org 100h
02 Ldata
03 a db "Abdullah"
04 .code
05 mov ax, @data
06 mov ds, ax
07 mov si, offset a
08 mov cx, 8
09 l1:
10 mov dl, [si]
11 inc si
12 mov ah, 2
13 int 21h
14 loop l1
```

### Explanation:

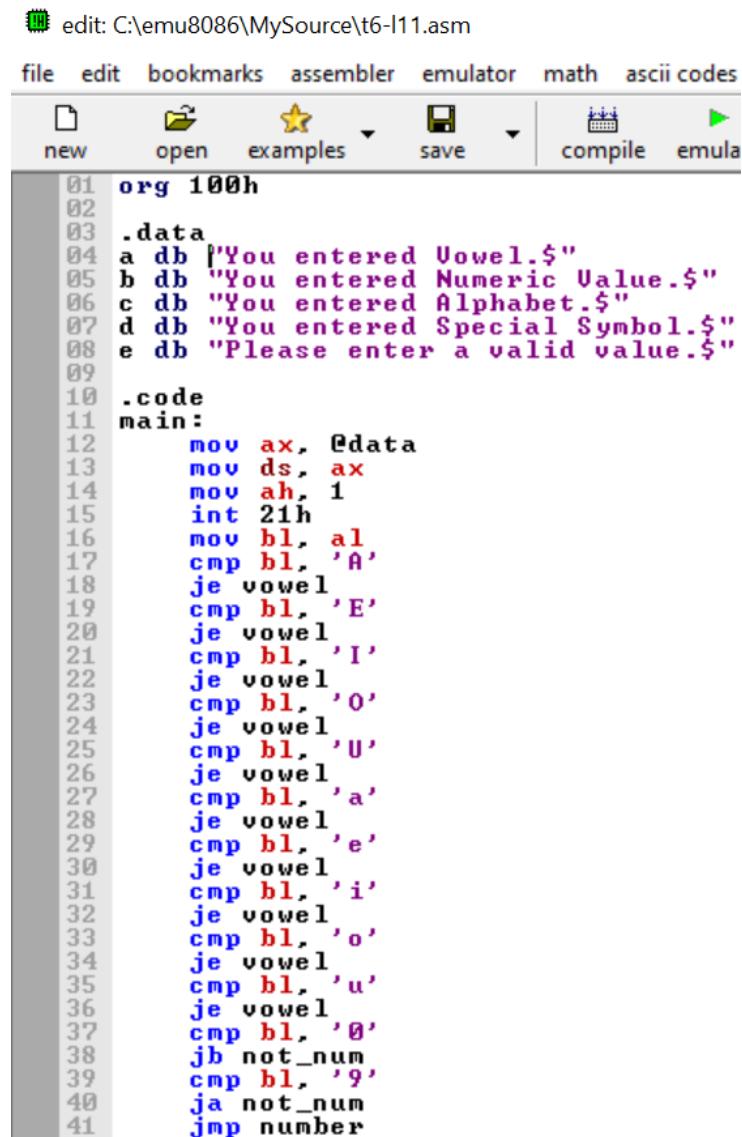
- Points to the start of the string.
- Loops four times (for each character).
- Loads each character into **DL**.
- Calls **INT 21h / AH=2** to display it.

### Output:



The screenshot shows the emulator screen with the title 'SCR emulator screen (80x25 chars)'. The screen displays the string 'Abdullah' in white text on a black background.

## Task 6:



The screenshot shows the emu8086 assembly editor interface. The menu bar includes file, edit, bookmarks, assembler, emulator, math, and ascii codes. The toolbar has icons for new, open, examples, save, compile, and emulate. The assembly code is displayed in the main window, starting with .org 100h and defining data for various messages. The .code section contains a main routine that reads a character from the user and compares it against all five vowels ('A', 'E', 'I', 'O', 'U'). If none match, it checks if the character is a numeric digit ('0'-'9'). If neither condition is met, it prints a message indicating a valid value was not entered.

```
01 org 100h
02
03 .data
04 a db "You entered Vowel.$"
05 b db "You entered Numeric Value.$"
06 c db "You entered Alphabet.$"
07 d db "You entered Special Symbol.$"
08 e db "Please enter a valid value.$"
09
10 .code
11 main:
12     mov ax, @data
13     mov ds, ax
14     mov ah, 1
15     int 21h
16     mov bl, al
17     cmp bl, 'A'
18     je vowel
19     cmp bl, 'E'
20     je vowel
21     cmp bl, 'I'
22     je vowel
23     cmp bl, 'O'
24     je vowel
25     cmp bl, 'U'
26     je vowel
27     cmp bl, 'a'
28     je vowel
29     cmp bl, 'e'
30     je vowel
31     cmp bl, 'i'
32     je vowel
33     cmp bl, 'o'
34     je vowel
35     cmp bl, 'u'
36     je vowel
37     cmp bl, '0'
38     jb not_num
39     cmp bl, '9'
40     ja not_num
41     jmp number
```

```

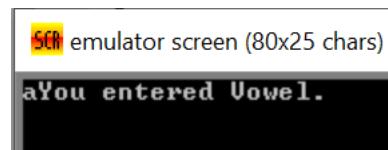
42 not_num:
43     cmp bl, 'A'
44     jb symbol
45     cmp bl, 'Z'
46     jbe alphabet
47     cmp bl, 'a'
48     jb symbol
49     cmp bl, 'z'
50     jbe alphabet
51     jmp symbol
52 vowel:
53     mov ah, 09h
54     lea dx, a
55     int 21h
56     jmp exit
57 number:
58     mov ah, 09h
59     lea dx, b
60     int 21h
61     jmp exit
62 alphabet:
63     mov ah, 09h
64     lea dx, c
65     int 21h
66     jmp exit
67 symbol:
68     mov ah, 09h
69     lea dx, d
70     int 21h
71     jmp exit
72
73 exit:
74     mov ah, 4Ch
75     int 21h
76
77

```

### Explanation:

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

### Output:



The screenshot shows a DOS terminal window with the title "emu8086 emulator screen (80x25 chars)". The window displays the text "You entered Vowel." in white on a black background.