# UNIVERSITY OF ASIA PACIFIC

## Department of Computer Science & Engineering

## LAB ASSIGNMENT-01

**Course Title** : Microprocessors and Assembly Language Lab

**Course Code** : CSE 312

**Date of Submission:** 14/09/2022

| **Submitted by:** | | **Submitted To:** |
|---|---|---|
| **Name** | : Sheikh Nafez Sadnan | Faria Zarin Subah |
| **Reg. No.** | : 20101106 | Lecturer |
| **Roll No.** | : 106 | Department of CSE |
| **Section** | : B (2) | University of Asia Pacific |

# Problem Statement 1:

Perform the following operations using emu8086:

a) Declare the following variables assigning the stated values:

**Table1:**

| Variable_name | Variable_size | Value_assigned |
|---|---|---|
| NUMB | DB | 01111000B |
| NUMD | DB | 120D |
| NUMO | DB | 170O |
| NUMH | DB | 78H |
| CHAR | DB | '3' |
| CHARASC | DB | 3 |
| STR | DB | "HELLO WORLD" |
| UNKNOWN | DB | ? |

b) Take any character as input and assign that character as the value of the UNKNOWN variable.

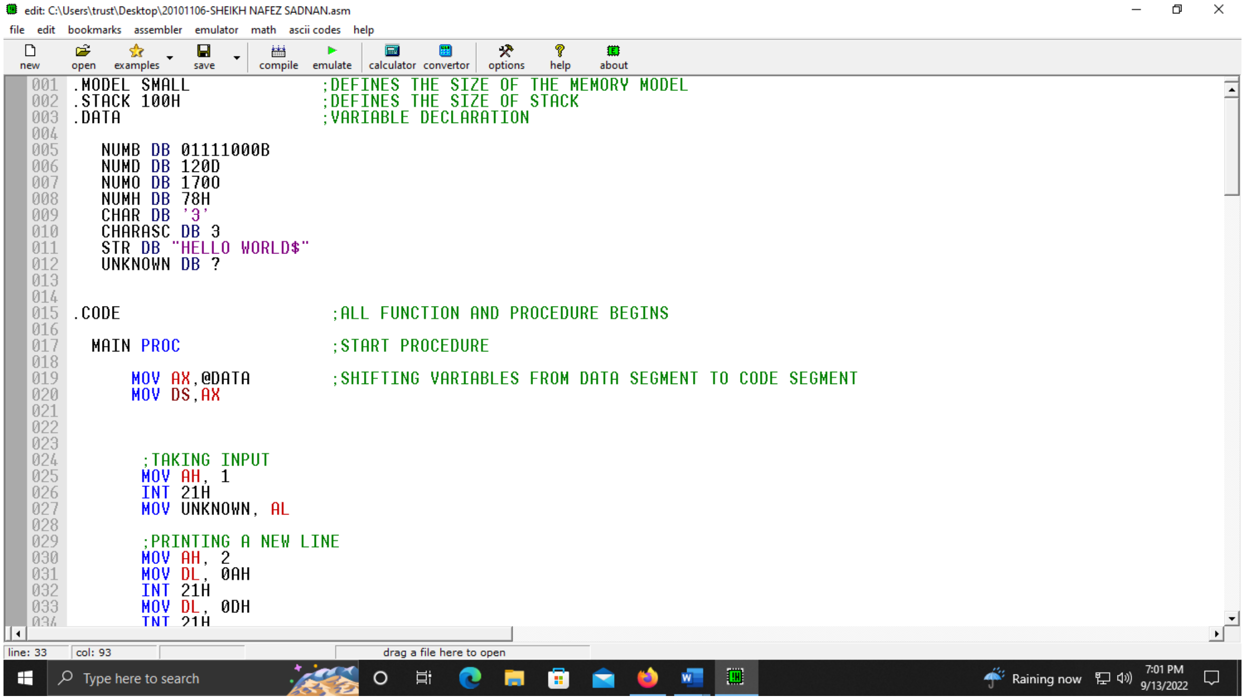
c) Print the values of all 8 variables with a newline between each print statement. Provide a screenshot of your output screen.

d) Provide a screenshot showing the status of the variables after emulating your code.

e) State the reason why each output shown on the output screen of emu8086 is different from that of **Table1** ? Provide justification for each of the 8 variables.

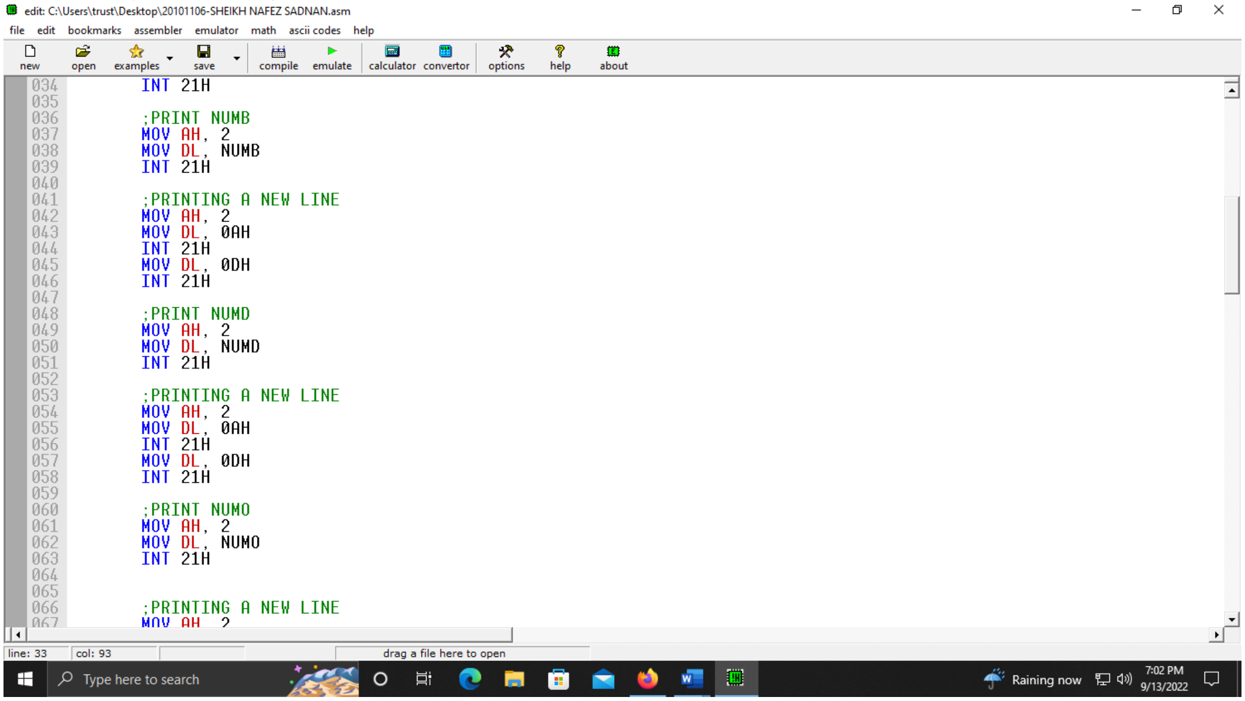
# Screenshot of the assembly code:

## Line (1-34)



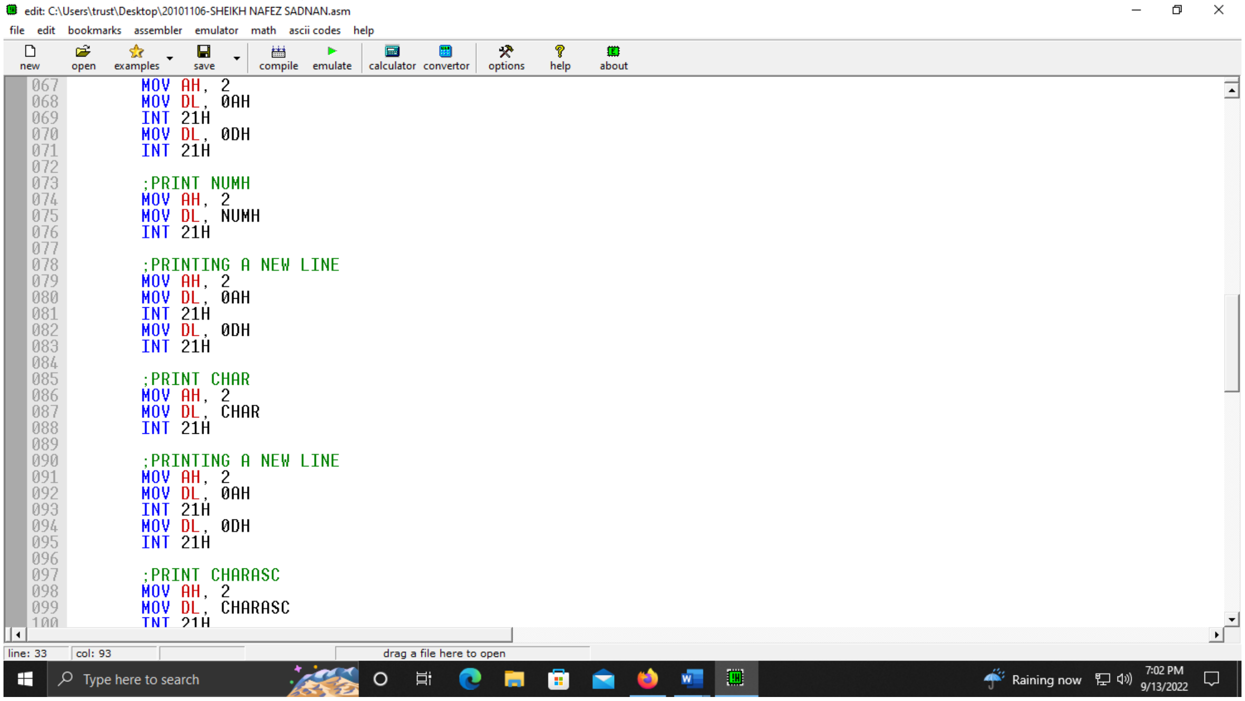
```
001 .MODEL SMALL                 ;DEFINES THE SIZE OF THE MEMORY MODEL
002 .STACK 100H                  ;DEFINES THE SIZE OF STACK
003 .DATA                        ;VARIABLE DECLARATION
004
005    NUMB DB 01111000B
006    NUMD DB 120D
007    NUMO DB 170O
008    NUMH DB 78H
009    CHAR DB '3'
010    CHARASC DB 3
011    STR DB "HELLO WORLD$"
012    UNKNOWN DB ?
013
014
015 .CODE                         ;ALL FUNCTION AND PROCEDURE BEGINS
016
017   MAIN PROC                   ;START PROCEDURE
018
019       MOV AX,@DATA            ;SHIFTING VARIABLES FROM DATA SEGMENT TO CODE SEGMENT
020       MOV DS,AX
021
022
023
024        ;TAKING INPUT
025        MOV AH, 1
026        INT 21H
027        MOV UNKNOWN, AL
028
029        ;PRINTING A NEW LINE
030        MOV AH, 2
031        MOV DL, 0AH
032        INT 21H
033        MOV DL, 0DH
034        INT 21H
```

## Line (34-67)



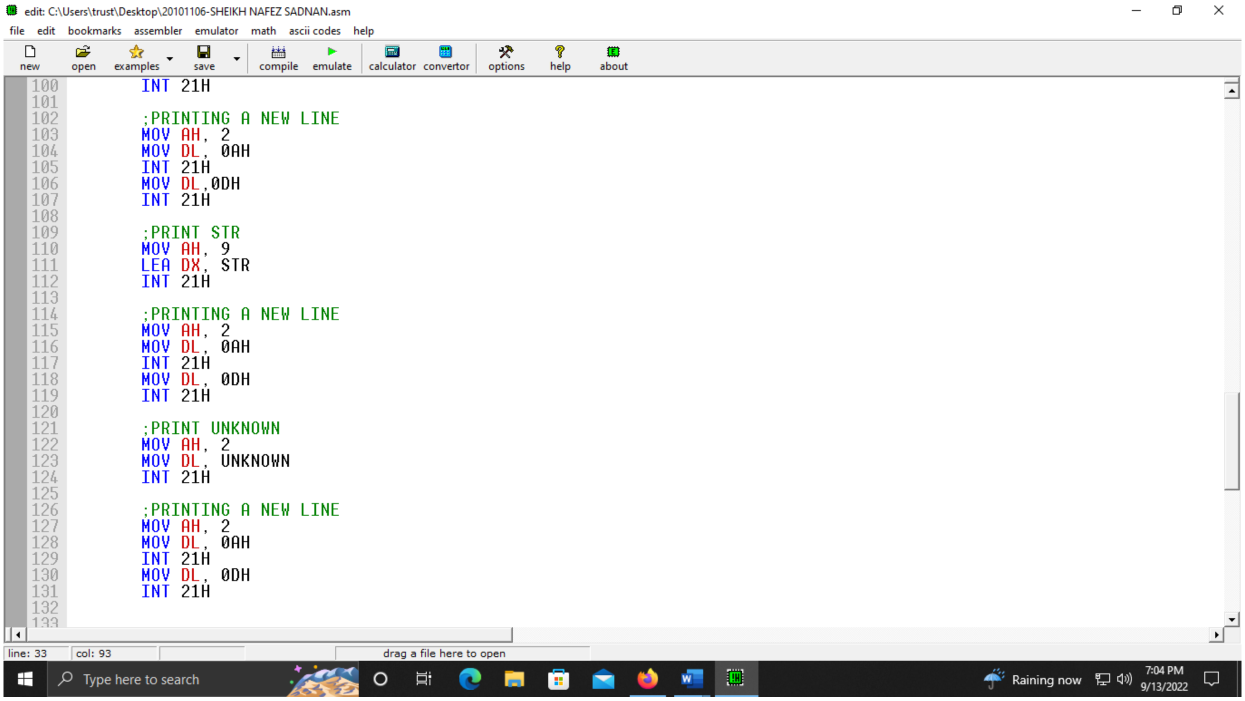
```
034        INT 21H
035
036        ;PRINT NUMB
037        MOV AH, 2
038        MOV DL, NUMB
039        INT 21H
040
041        ;PRINTING A NEW LINE
042        MOV AH, 2
043        MOV DL, 0AH
044        INT 21H
045        MOV DL, 0DH
046        INT 21H
047
048        ;PRINT NUMD
049        MOV AH, 2
050        MOV DL, NUMD
051        INT 21H
052
053        ;PRINTING A NEW LINE
054        MOV AH, 2
055        MOV DL, 0AH
056        INT 21H
057        MOV DL, 0DH
058        INT 21H
059
060        ;PRINT NUMO
061        MOV AH, 2
062        MOV DL, NUMO
063        INT 21H
064
065
066        ;PRINTING A NEW LINE
067        MOV AH, 2
```

# Line (67-100)



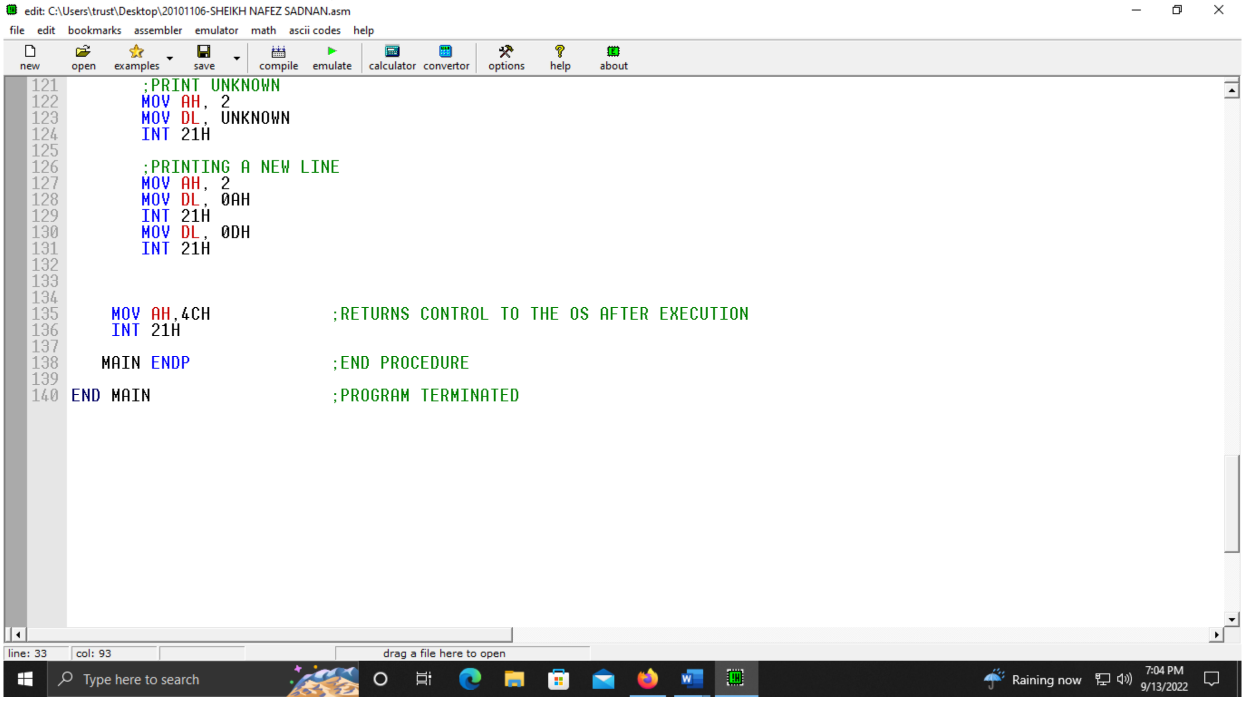
```
        MOV AH, 2
        MOV DL, 0AH
        INT 21H
        MOV DL, 0DH
        INT 21H

        ;PRINT NUMH
        MOV AH, 2
        MOV DL, NUMH
        INT 21H

        ;PRINTING A NEW LINE
        MOV AH, 2
        MOV DL, 0AH
        INT 21H
        MOV DL, 0DH
        INT 21H

        ;PRINT CHAR
        MOV AH, 2
        MOV DL, CHAR
        INT 21H

        ;PRINTING A NEW LINE
        MOV AH, 2
        MOV DL, 0AH
        INT 21H
        MOV DL, 0DH
        INT 21H

        ;PRINT CHARASC
        MOV AH, 2
        MOV DL, CHARASC
        INT 21H
```

# Line (100-133)



```
        INT 21H

        ;PRINTING A NEW LINE
        MOV AH, 2
        MOV DL, 0AH
        INT 21H
        MOV DL,0DH
        INT 21H

        ;PRINT STR
        MOV AH, 9
        LEA DX, STR
        INT 21H

        ;PRINTING A NEW LINE
        MOV AH, 2
        MOV DL, 0AH
        INT 21H
        MOV DL, 0DH
        INT 21H

        ;PRINT UNKNOWN
        MOV AH, 2
        MOV DL, UNKNOWN
        INT 21H

        ;PRINTING A NEW LINE
        MOV AH, 2
        MOV DL, 0AH
        INT 21H
        MOV DL, 0DH
        INT 21H
```

# Line (121-140)

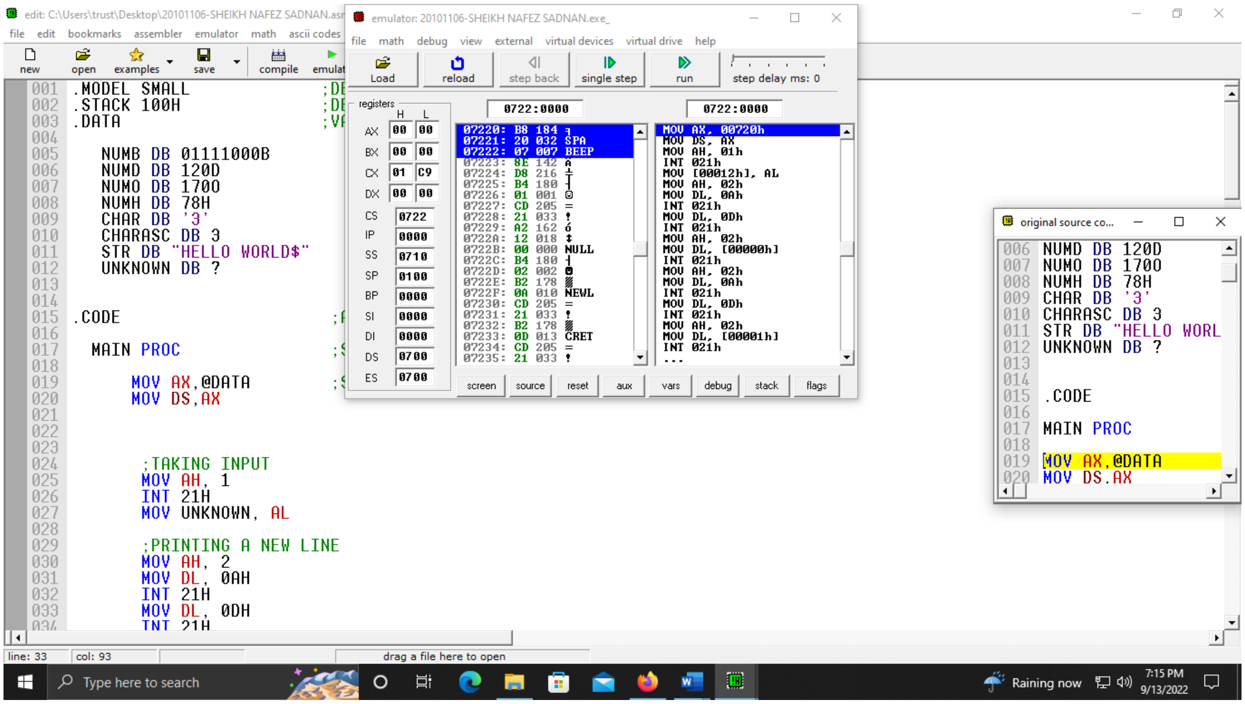


edit: C:\Users\trust\Desktop\20101106-SHEIKH NAFEZ SADNAN.asm

```
121        ;PRINT UNKNOWN
122        MOV AH, 2
123        MOV DL, UNKNOWN
124        INT 21H
125
126        ;PRINTING A NEW LINE
127        MOV AH, 2
128        MOV DL, 0AH
129        INT 21H
130        MOV DL, 0DH
131        INT 21H
132
133
134
135     MOV AH,4CH                ;RETURNS CONTROL TO THE OS AFTER EXECUTION
136     INT 21H
137
138    MAIN ENDP                  ;END PROCEDURE
139
140 END MAIN                      ;PROGRAM TERMINATED
```

line: 33 col: 93

# Assembly Code Screen Shot (While Emulating):



edit: C:\Users\trust\Desktop\20101106-SHEIKH NAFEZ SADNAN.asm

```
.MODEL SMALL                                          ;DE
.STACK 100H                                           ;DE
.DATA                                                 ;VA

   NUMB DB 01111000B
   NUMD DB 120D
   NUMO DB 170O
   NUMH DB 78H
   CHAR DB '3'
   CHARASC DB 3
   STR DB "HELLO WORLD$"
   UNKNOWN DB ?


.CODE                                                 ;

  MAIN PROC                                           ;

      MOV AX,@DATA                                    ;
      MOV DS,AX



       ;TAKING INPUT
       MOV AH, 1
       INT 21H
       MOV UNKNOWN, AL

       ;PRINTING A NEW LINE
       MOV AH, 2
       MOV DL, 0AH
       INT 21H
       MOV DL, 0DH
       INT 21H
```

emulator: 20101106-SHEIKH NAFEZ SADNAN.exe_

step delay ms: 0

| registers | H | L |
|---|---|---|
| AX | 00 | 00 |
| BX | 00 | 00 |
| CX | 01 | C9 |
| DX | 00 | 00 |
| CS | 0722 | |
| IP | 0000 | |
| SS | 0710 | |
| SP | 0100 | |
| BP | 0000 | |
| SI | 0000 | |
| DI | 0000 | |
| DS | 0700 | |
| ES | 0700 | |

```
0722:0000
07220: B8 184 ╕
07221: 20 032 SPA
07222: 07 007 BEEP
07223: 8E 142 Ä
07224: D8 216 ╪
07225: B4 180 ┤
07226: 01 001 ☺
07227: CD 205 ═
07228: 21 033 !
07229: A2 162 ó
0722A: 12 018 ↕
0722B: 00 000 NULL
0722C: B4 180 ┤
0722D: 02 002 ☻
0722E: B2 178 ▓
0722F: 0A 010 NEWL
07230: CD 205 ═
07231: 21 033 !
07232: B2 178 ▓
07233: 0D 013 CRET
07234: CD 205 ═
07235: 21 033 !
```

```
0722:0000
MOV AX, 00720h
MOV DS, AX
MOV AH, 01h
INT 021h
MOV [00012h], AL
MOV AH, 02h
MOV DL, 0Ah
INT 021h
MOV DL, 0Dh
INT 021h
MOV AH, 02h
MOV DL, [00000h]
INT 021h
MOV AH, 02h
MOV DL, 0Ah
INT 021h
MOV DL, 0Dh
INT 021h
MOV AH, 02h
MOV DL, [00001h]
INT 021h
...
```

original source co...

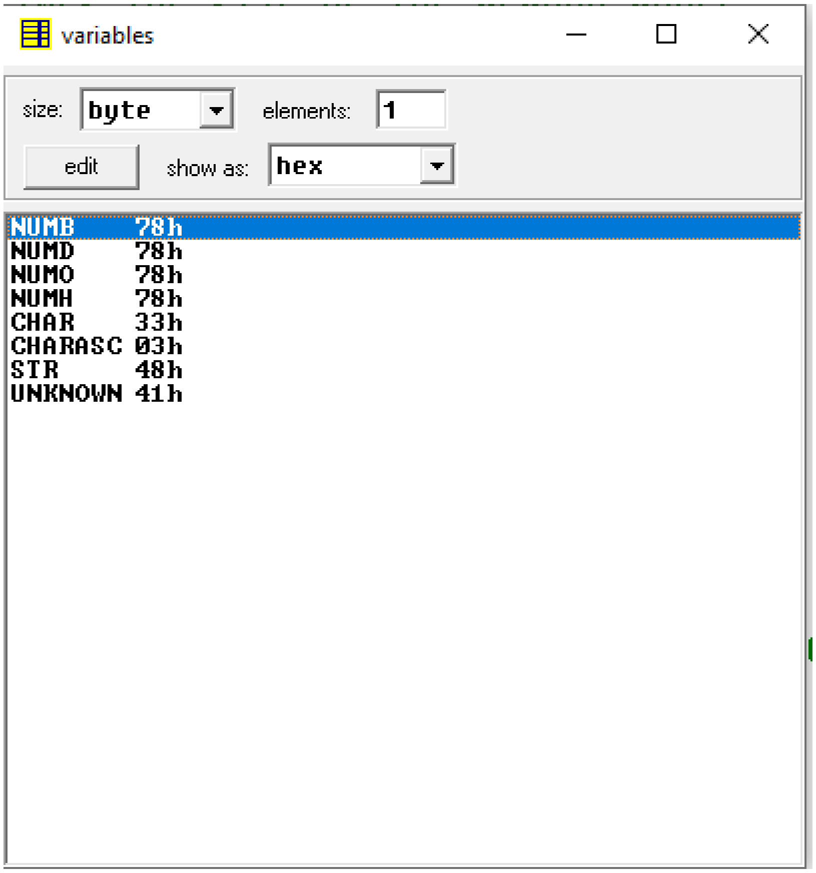
```
NUMD DB 120D
NUMO DB 170O
NUMH DB 78H
CHAR DB '3'
CHARASC DB 3
STR DB "HELLO WORL
UNKNOWN DB ?


.CODE

MAIN PROC

MOV AX,@DATA
MOV DS,AX
```

line: 33 col: 93

## Status of all 8 variables after assigning values:

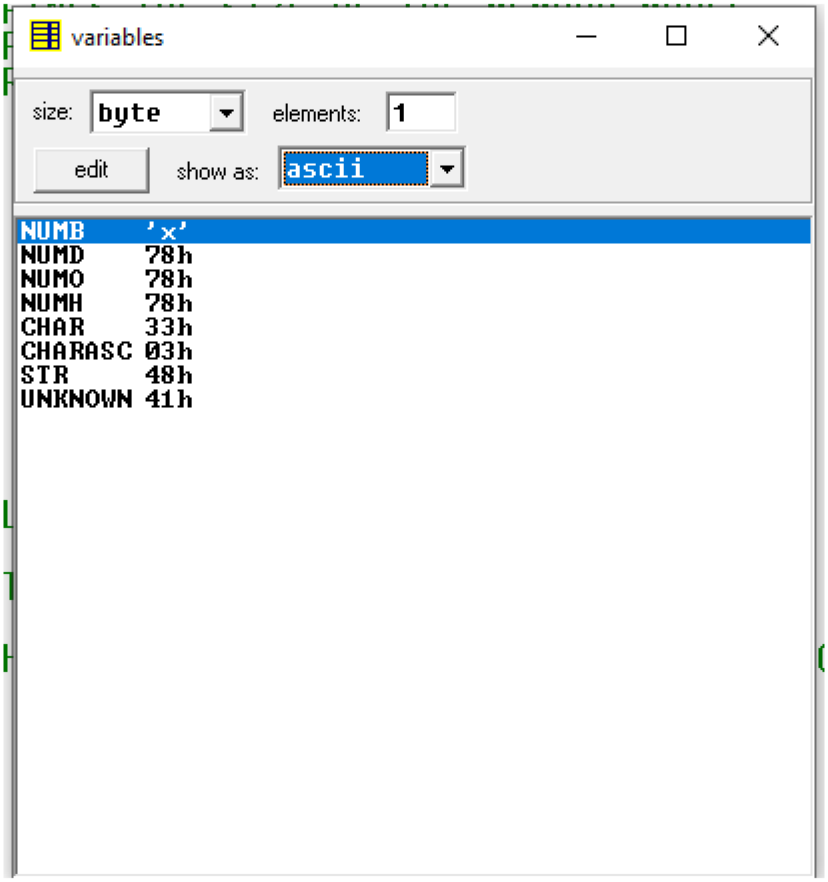
**NUMB DB 01111000B:**
After, assigning  NUMB DB 01111000B,
AH got 02 (Calling the output subroutine)
DL got 78H (Equivalent to 01111000B)

Output is x which is different from Table-1
Because, in ASCII table **120:x** (01111000B/78H equivalent to 120 in decimal.)



size: byte
elements: 1
show as: hex

```
NUMB    78h
NUMD    78h
NUMO    78h
NUMH    78h
CHAR    33h
CHARASC 03h
STR     48h
UNKNOWN 41h
```

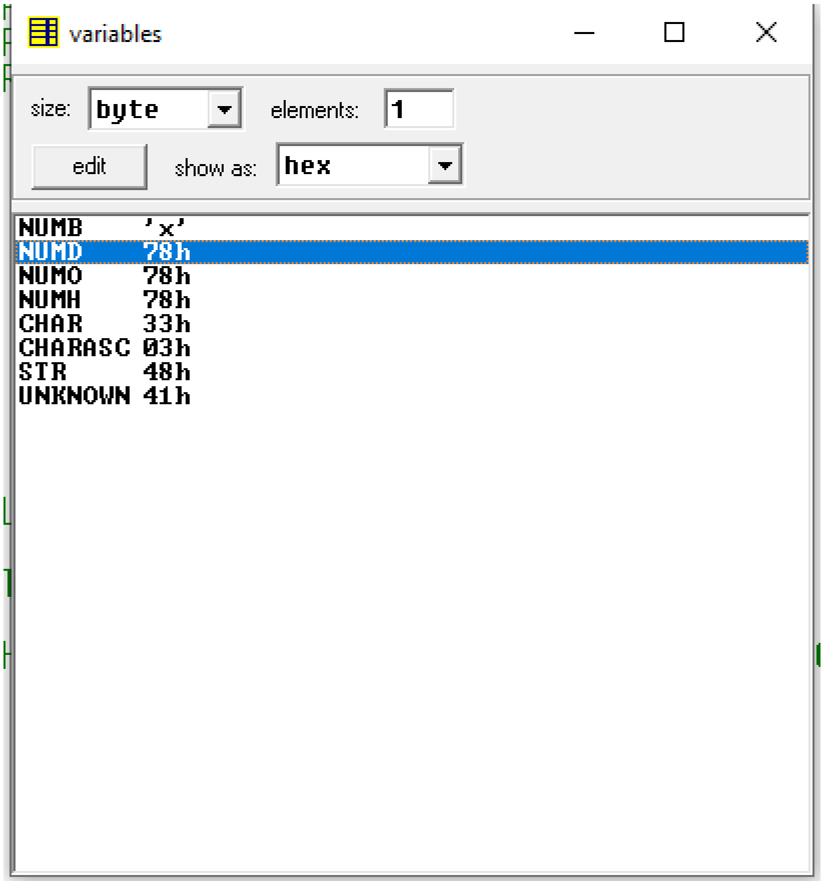


```
NUMB    'x'
NUMD    78h
NUMO    78h
NUMH    78h
CHAR    33h
CHARASC 03h
STR     48h
UNKNOWN 41h
```

**NUMD DB 120D:**

After, assigning NUMD DB 120D,
AH got 02 (Calling the output subroutine)
DL got 78H (Equivalent to 120D)

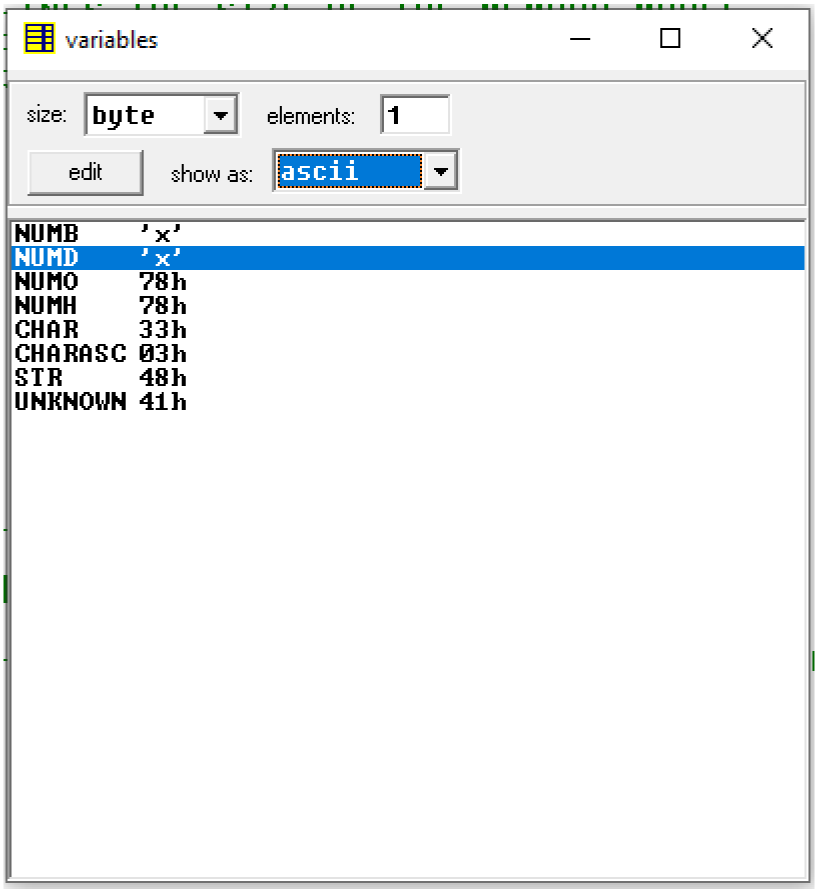
Output is x which is different from Table-1
Because, in ASCII table **120:x** (78H equivalent to 120 in decimal.)



size: byte elements: 1

show as: hex

```
NUMB    'x'
NUMD    78h
NUMO    78h
NUMH    78h
CHAR    33h
CHARASC 03h
STR     48h
UNKNOWN 41h
```



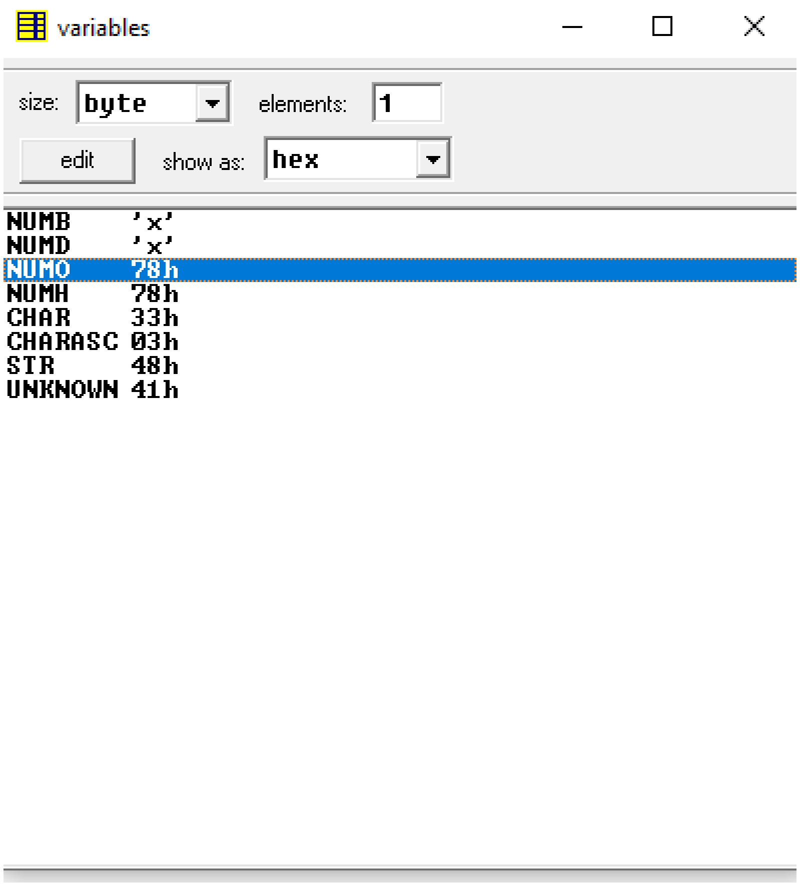
size: byte elements: 1

show as: ascii

```
NUMB    'x'
NUMD    'x'
NUMO    78h
NUMH    78h
CHAR    33h
CHARASC 03h
STR     48h
UNKNOWN 41h
```

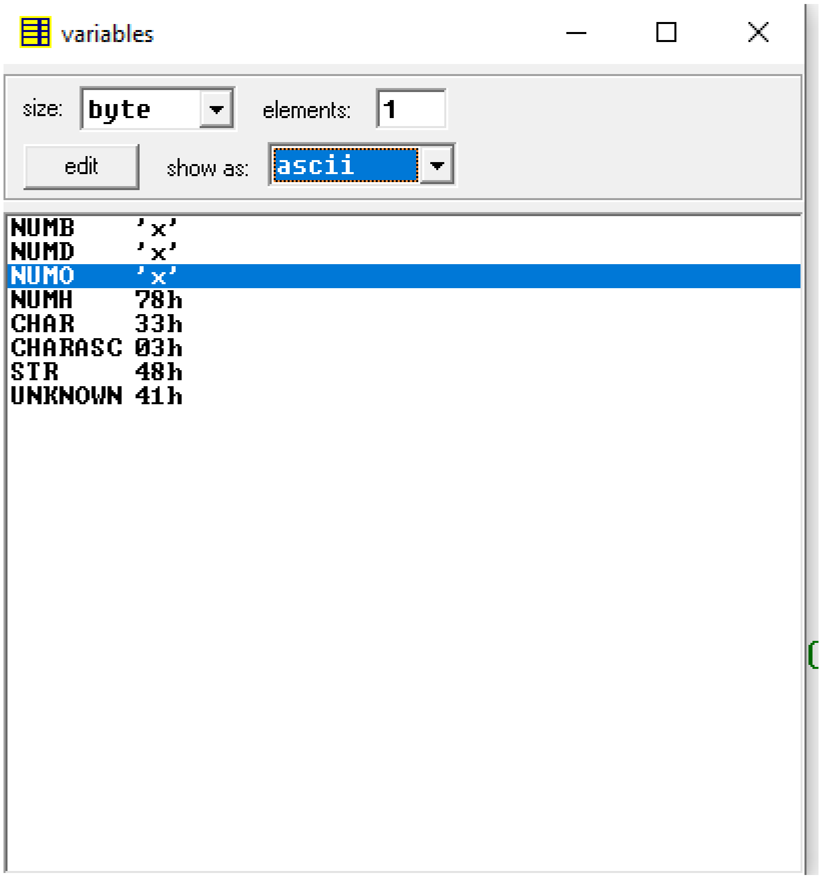
**NUMO DB 170O:**
After, assigning  NUMO DB 170O,
AH got 02 (Calling the output subroutine)
DL got 78H (Equivalent to 170O)

Output is x which is different from Table-1
Because, in ASCII table **120:x** (170O/78H equivalent to 120 in decimal.)



size: byte elements: 1
show as: hex

```
NUMB    'x'
NUMD    'x'
NUMO    78h
NUMH    78h
CHAR    33h
CHARASC 03h
STR     48h
UNKNOWN 41h
```



size: byte elements: 1

show as: ascii

```
NUMB    'x'
NUMD    'x'
NUMO    'x'
NUMH    78h
CHAR    33h
CHARASC 03h
STR     48h
UNKNOWN 41h
```

**NUMH DB 78H:**

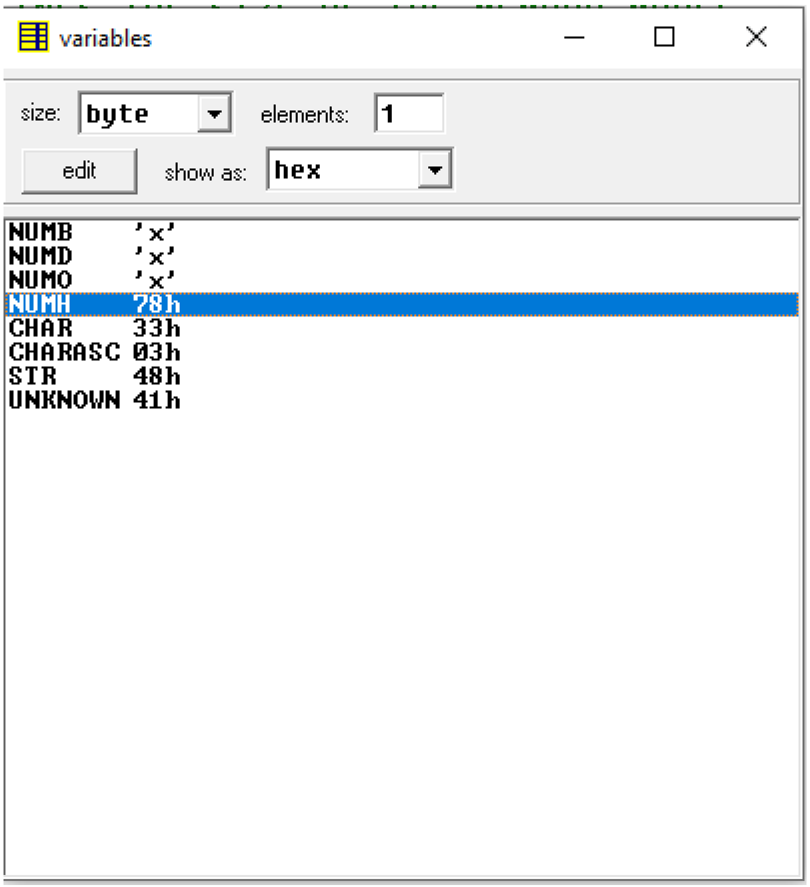
After, assigning  NUMH DB 78H,

AH got 02 (Calling the output subroutine)

DL got 78H

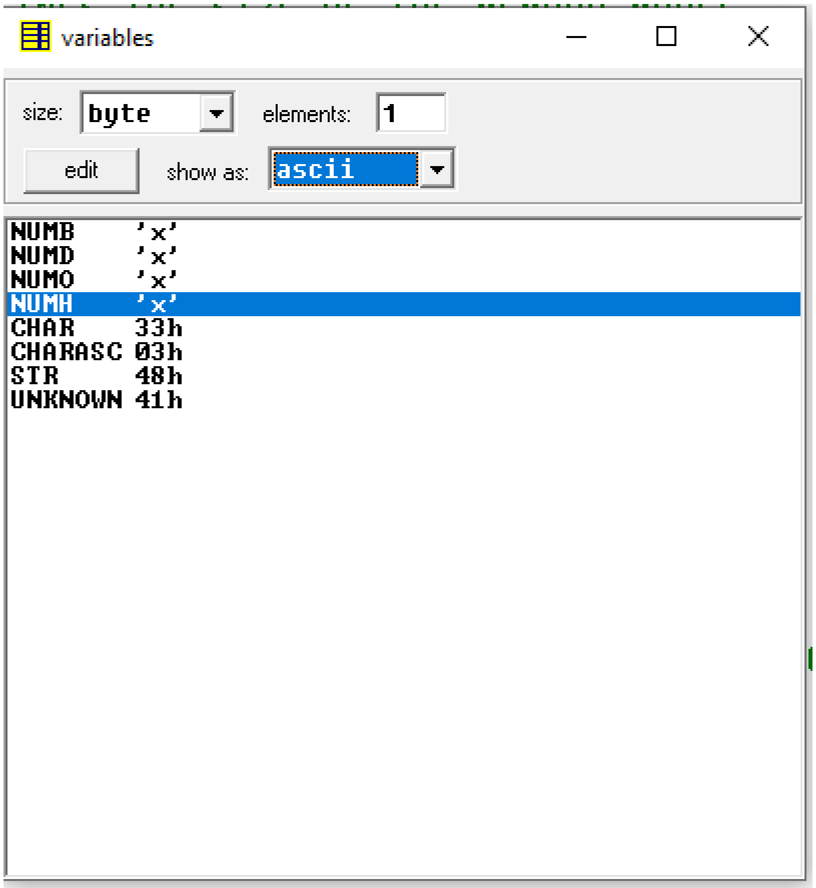
Output is x which is different from Table-1

Because, in ASCII table **120:x** (78H equivalent to 120 in decimal.)



size: byte elements: 1
show as: hex

```
NUMB    'x'
NUMD    'x'
NUMO    'x'
NUMH    78h
CHAR    33h
CHARASC 03h
STR     48h
UNKNOWN 41h
```

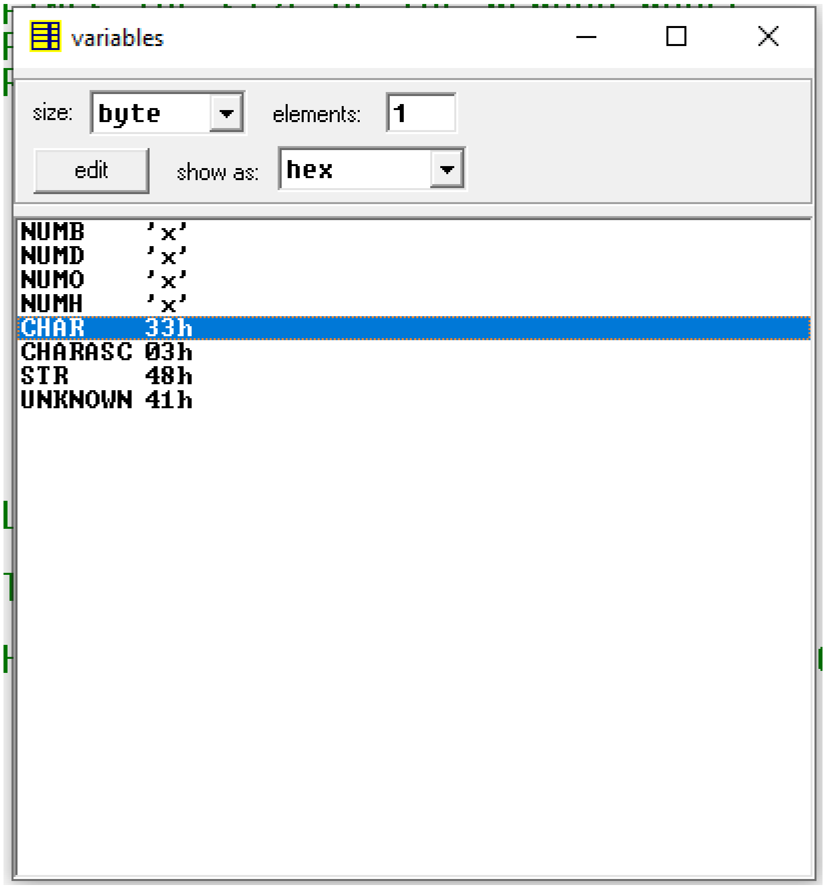


size: byte elements: 1
show as: ascii

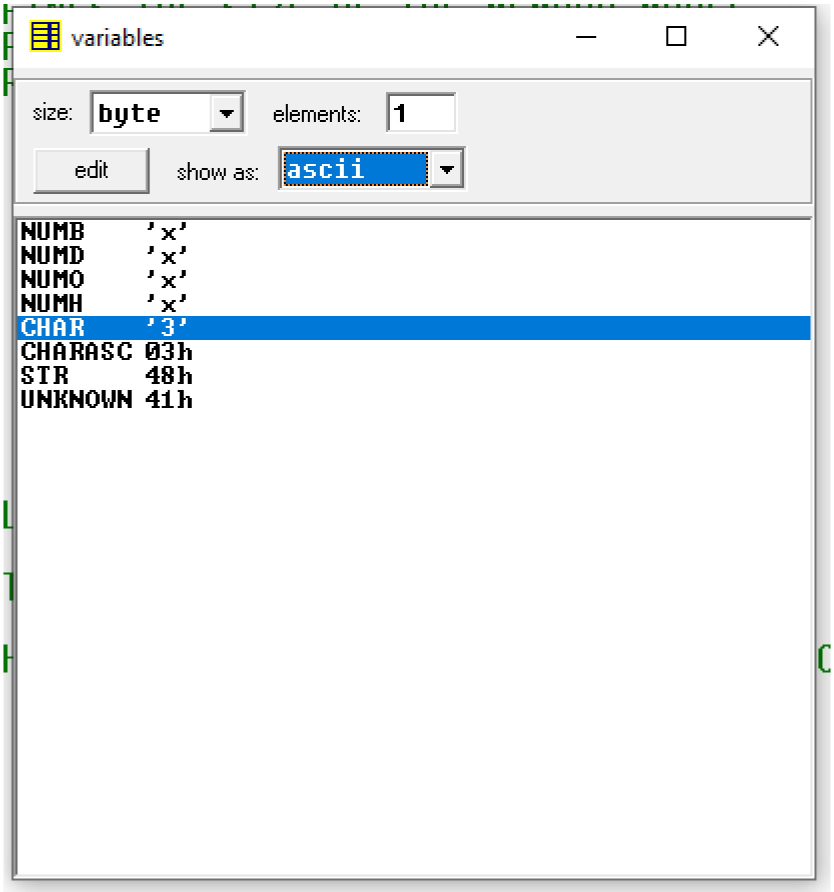
```
NUMB    'x'
NUMD    'x'
NUMO    'x'
NUMH    'x'
CHAR    33h
CHARASC 03h
STR     48h
UNKNOWN 41h
```

**CHAR DB '3':**
After, assigning  CHAR DB ‘3’,
AH got 02 (Calling the output subroutine)
DL got 33H

Output is 3 which is different from Table-1
Because, in ASCII table **051:3** (33H equivalent to 051 in decimal.)



```
NUMB    'x'
NUMD    'x'
NUMO    'x'
NUMH    'x'
CHAR    33h
CHARASC 03h
STR     48h
UNKNOWN 41h
```



size: byte elements: 1

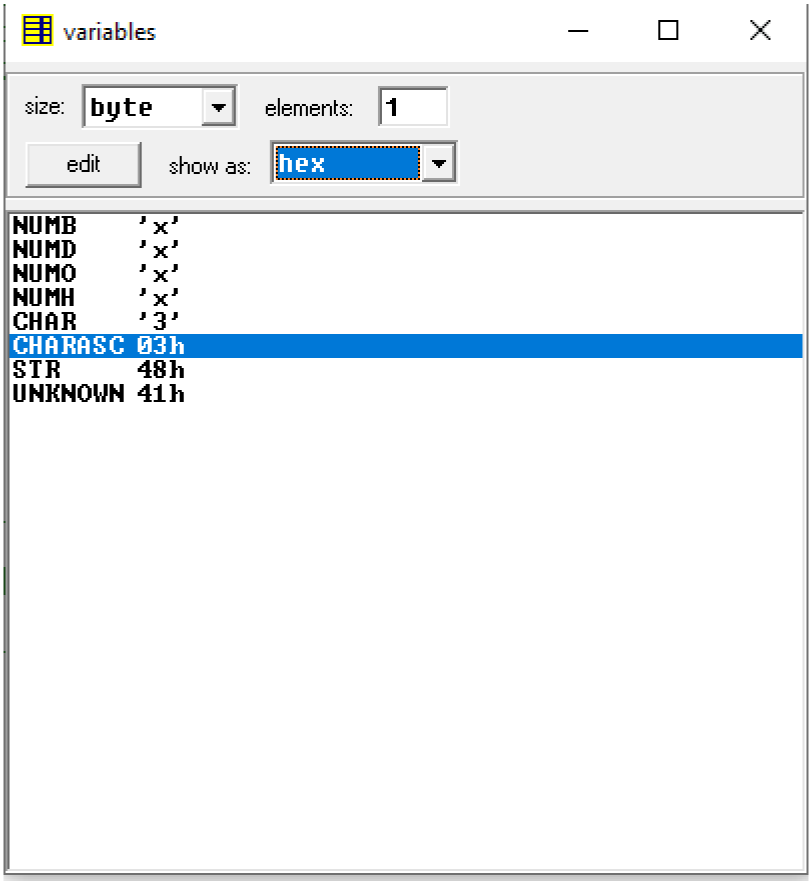
show as: ascii

```
NUMB    'x'
NUMD    'x'
NUMO    'x'
NUMH    'x'
CHAR    '3'
CHARASC 03h
STR     48h
UNKNOWN 41h
```

**CHARASC DB 3:**

After, assigning  CHARASC DB 3,
AH got 02 (Calling the output subroutine)
DL got 03H

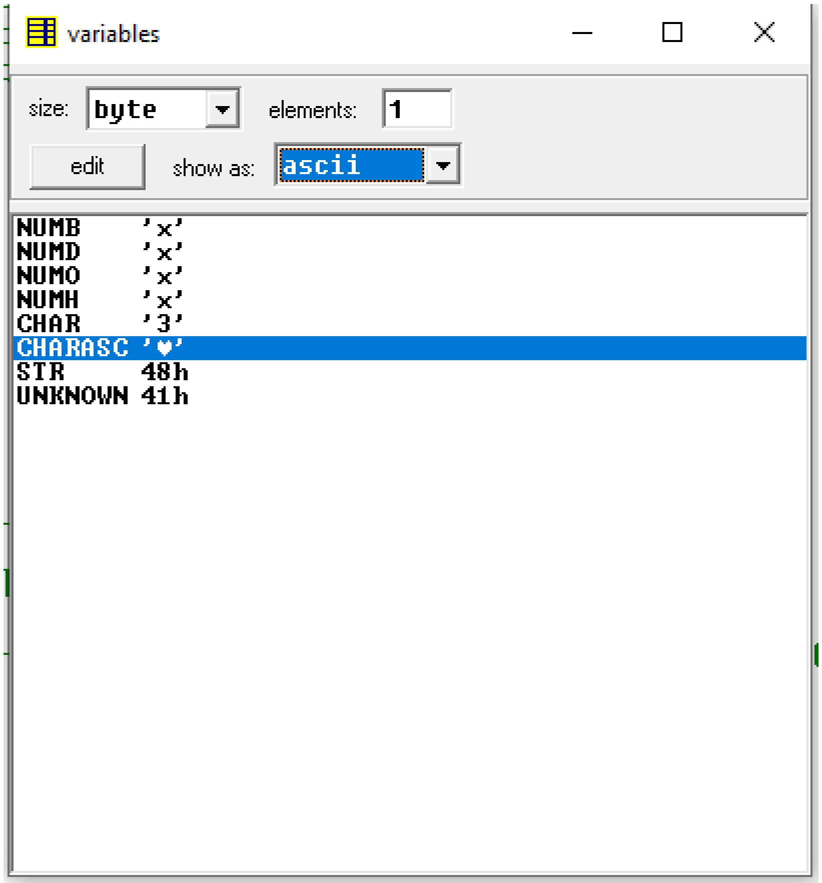
Output is a heart shape emoji which is different from Table-1
Because, in ASCII table **003:** ♥ (03H equivalent to 003 in decimal.)



size: byte elements: 1

show as: hex

```
NUMB    'x'
NUMD    'x'
NUMO    'x'
NUMH    'x'
CHAR    '3'
CHARASC 03h
STR     48h
UNKNOWN 41h
```



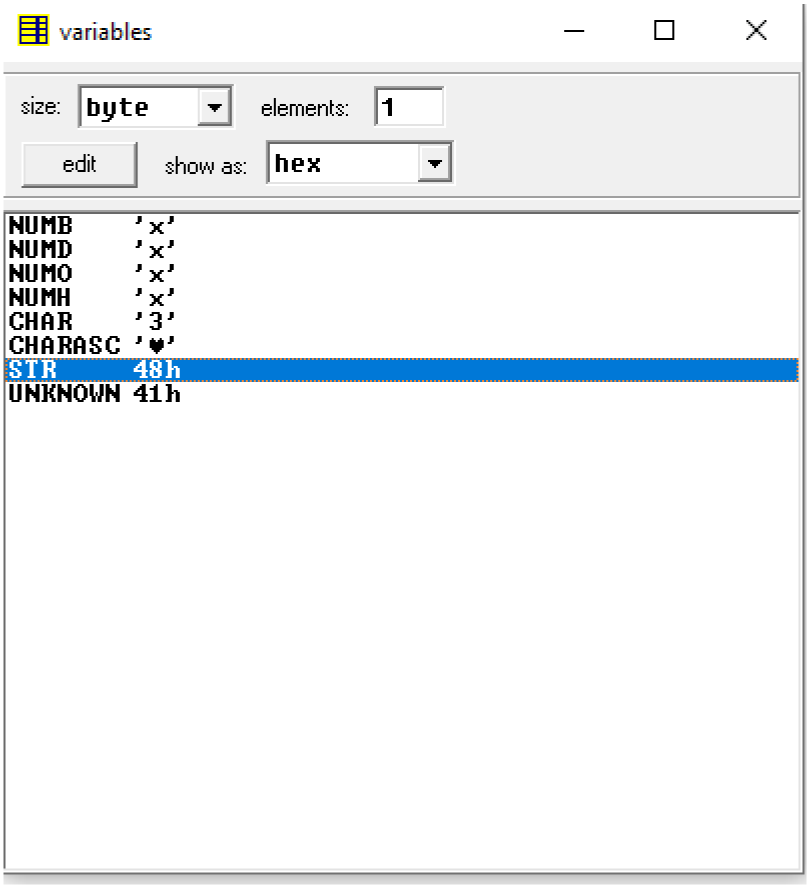
size: byte elements: 1

show as: ascii

```
NUMB    'x'
NUMD    'x'
NUMO    'x'
NUMH    'x'
CHAR    '3'
CHARASC '♥'
STR     48h
UNKNOWN 41h
```

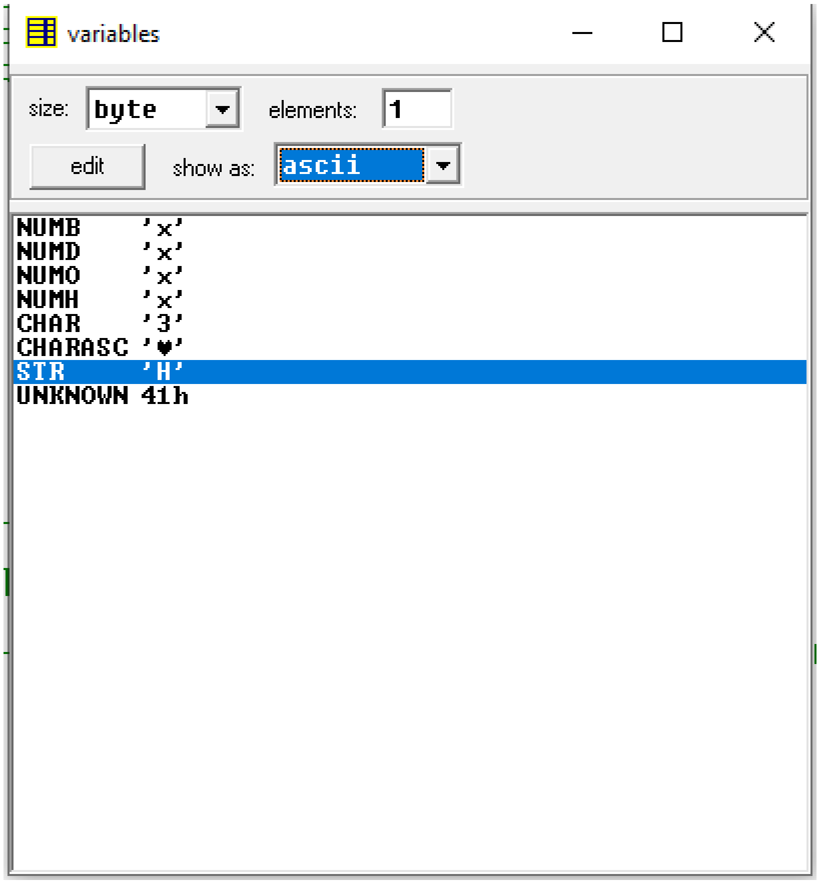
**STR DB "HELLO WORLD":**
Input needs a dollar ($) sign after the string.
Input- STR DB "HELLO WORLD$"
After, assigning STR DB “HELLO WORLD$”,
AH got 09 (Display string function)

Output is HELLO WORLD which is different from Table-1
Because, LEA DX, STR receives the string and INT 21H displays it.



size: byte elements: 1
show as: hex

```
NUMB    'x'
NUMD    'x'
NUMO    'x'
NUMH    'x'
CHAR    '3'
CHARASC '♥'
STR     48h
UNKNOWN 41h
```



size: byte elements: 1

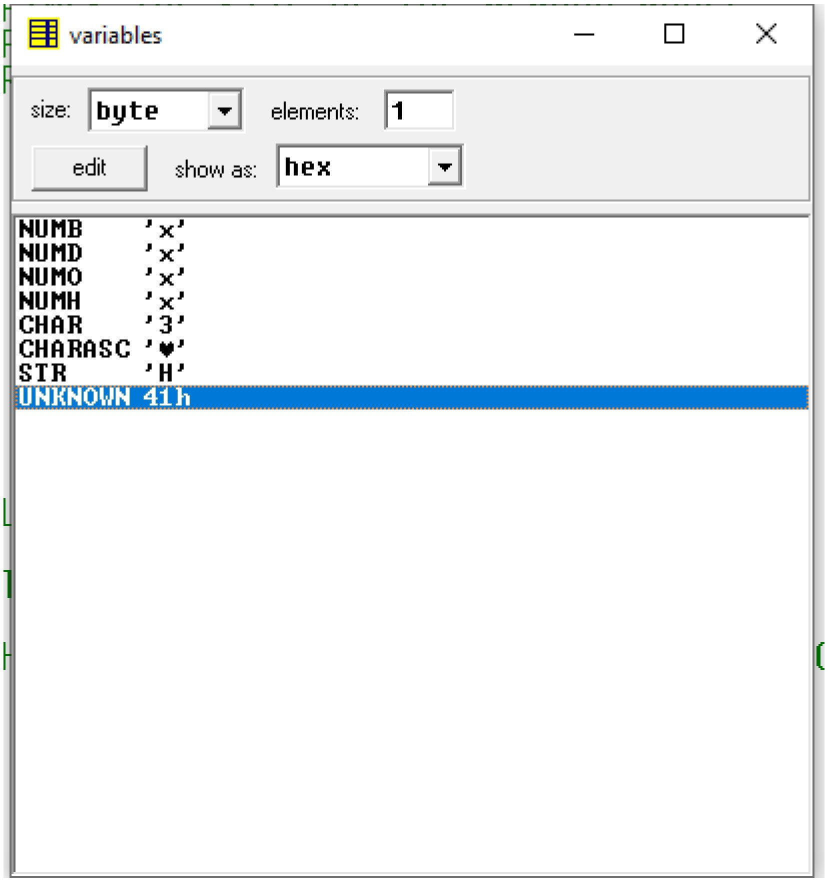
show as: ascii

```
NUMB     'x'
NUMD     'x'
NUMO     'x'
NUMH     'x'
CHAR     '3'
CHARASC '♥'
STR      'H'
UNKNOWN 41h
```

**UNKNOWN DB ? :**

After, assigning  UNKNOWN DB ?,
AH got 02 (Calling the output subroutine)
DL got 41H

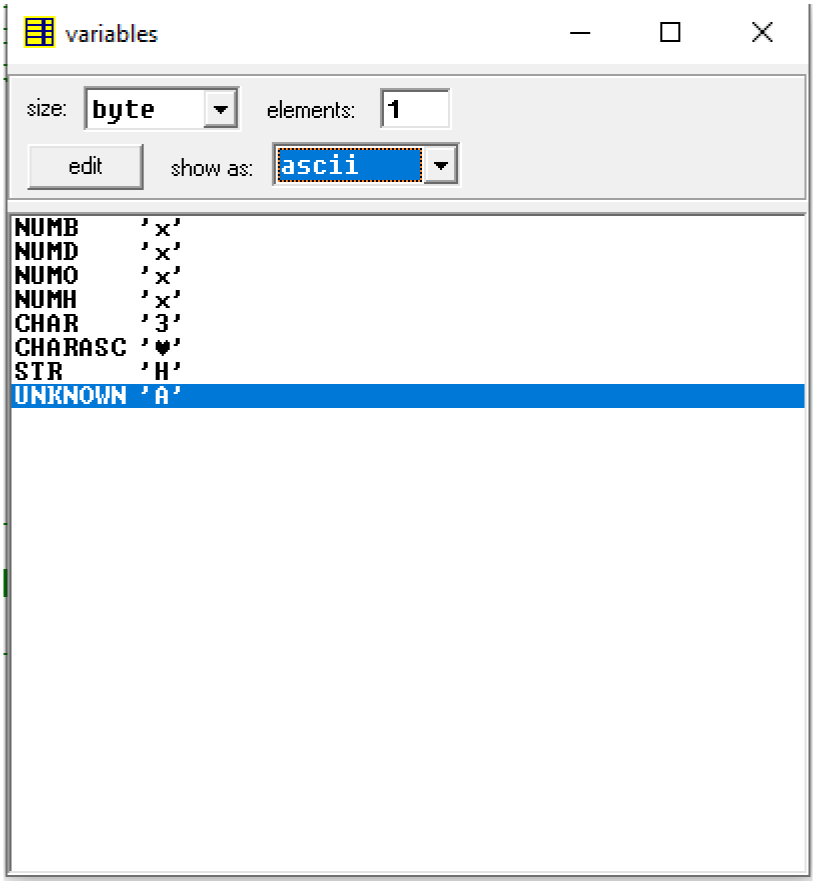
Output is A which was given as input, which is different from Table-1
Because, in ASCII table **065:A** (41H equivalent to 065 in decimal.)



size: byte   elements: 1

show as: hex

```
NUMB    'x'
NUMD    'x'
NUMO    'x'
NUMH    'x'
CHAR    '3'
CHARASC '♥'
STR     'H'
UNKNOWN 41h
```



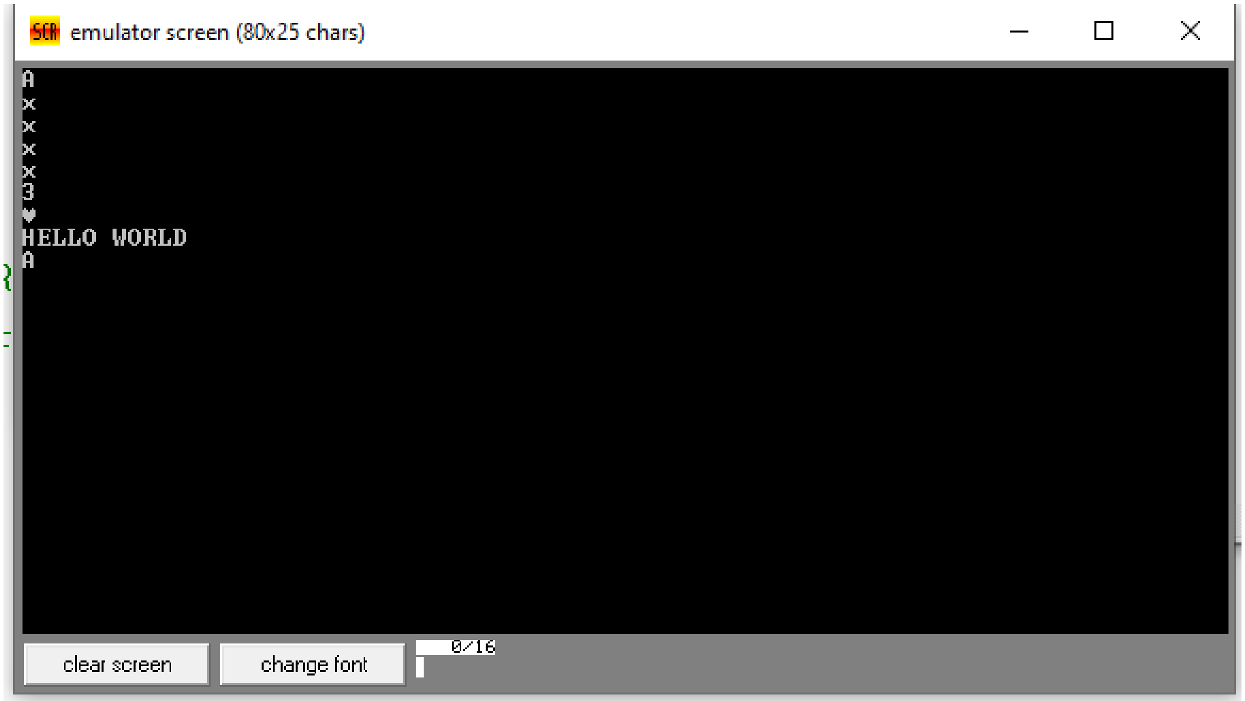
size: byte   elements: 1

show as: ascii

```
NUMB    'x'
NUMD    'x'
NUMO    'x'
NUMH    'x'
CHAR    '3'
CHARASC '♥'
STR     'H'
UNKNOWN 'A'
```

## A screenshot of the output screen:

After all values are assigned, output displays on screen:



```
A
x
x
x
x
3
♥
HELLO WORLD
A
```

Here,

A (Given Input)

x (Output of 01111000B)

x (Output of 120D)

x (Output of 170O)

x (Output of 78H)

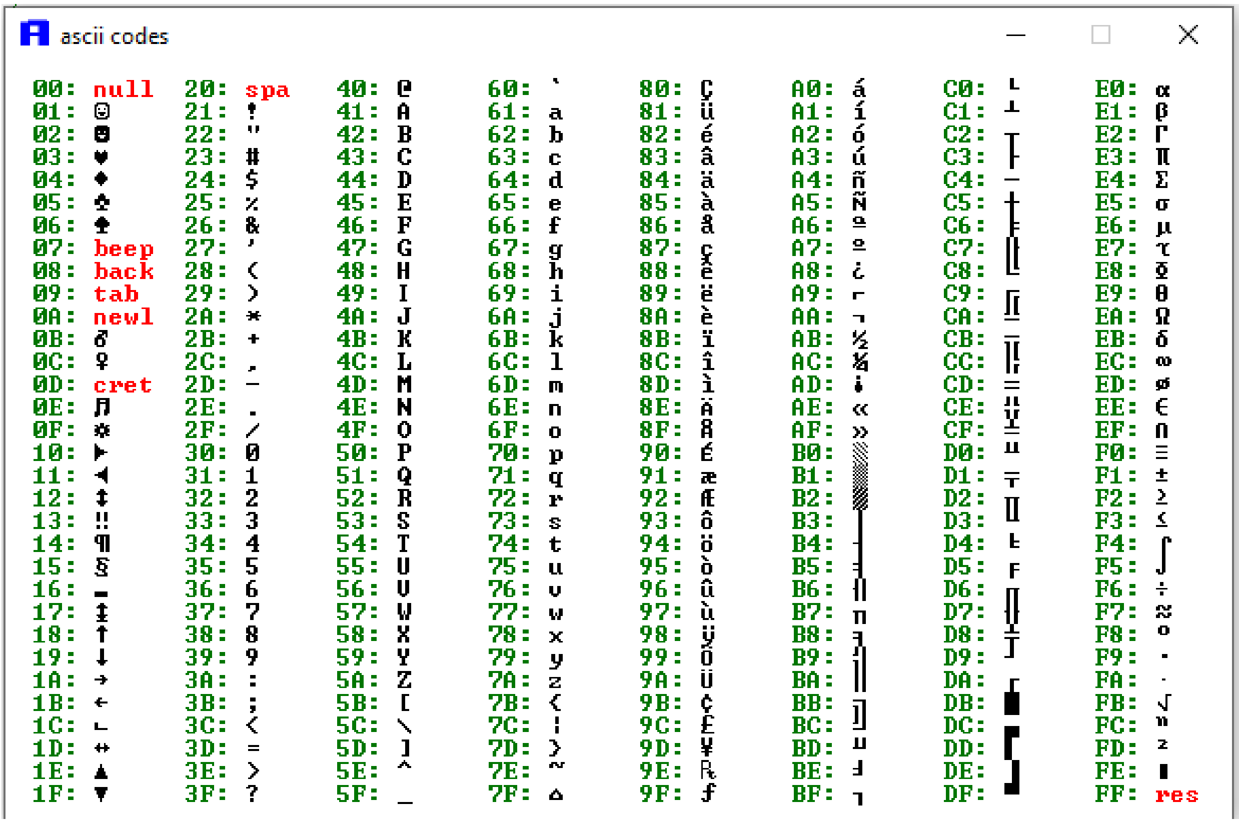
3 (Output of '3')

♥ (Output of 3)

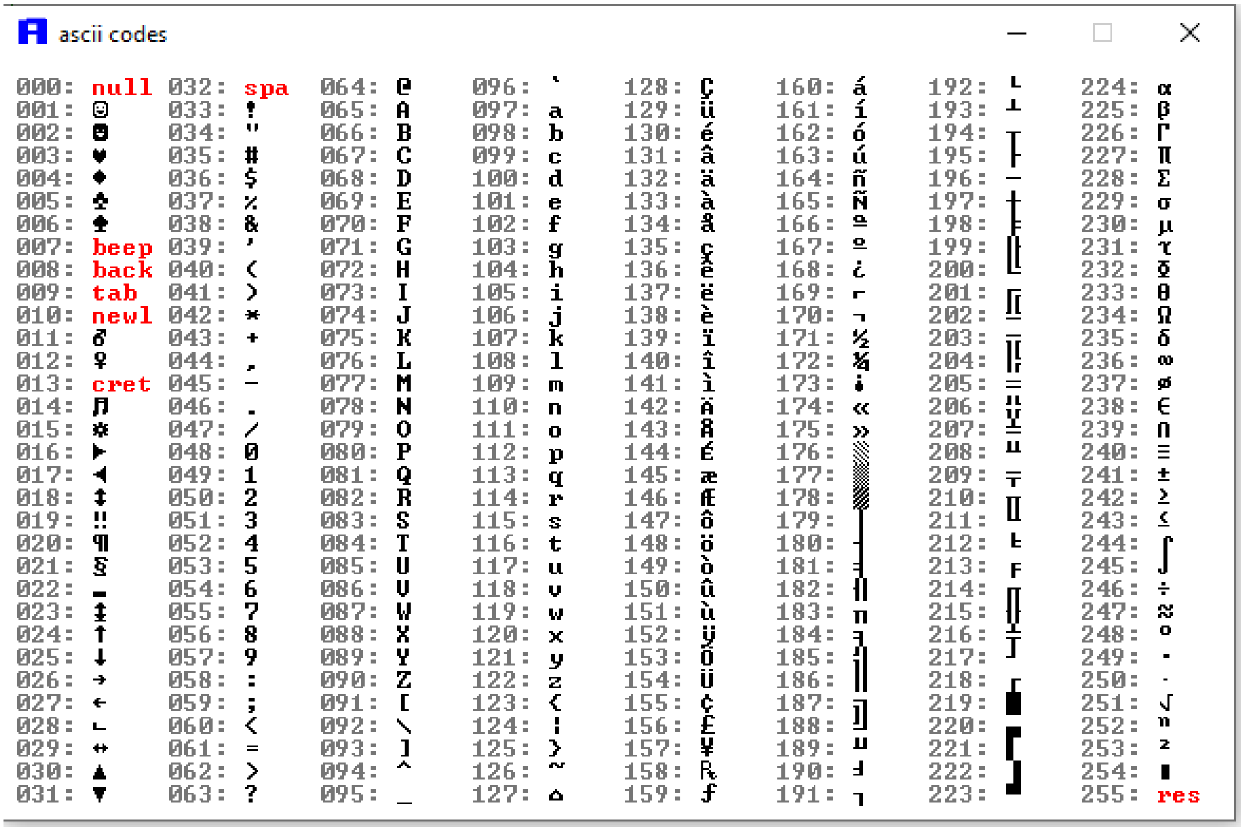
HELLO WORLD (Output of ‘HELLO WORLD’)

A (Output of given Input)

**Reference:** The ASCII code table-



```
00: null  20: spa  40: @  60: `  80: Ç  A0: á  C0: └  E0: α
01: ☺     21: !    41: A  61: a  81: ü  A1: í  C1: ┴  E1: ß
02: ☻     22: "    42: B  62: b  82: é  A2: ó  C2: ┬  E2: Γ
03: ♥     23: #    43: C  63: c  83: â  A3: ú  C3: ├  E3: π
04: ♦     24: $    44: D  64: d  84: ä  A4: ñ  C4: ─  E4: Σ
05: ♣     25: %    45: E  65: e  85: à  A5: Ñ  C5: ┼  E5: σ
06: ♠     26: &    46: F  66: f  86: å  A6: ª  C6: ╞  E6: µ
07: beep  27: '    47: G  67: g  87: ç  A7: º  C7: ╟  E7: τ
08: back  28: (    48: H  68: h  88: ê  A8: ¿  C8: ╚  E8: Φ
09: tab   29: )    49: I  69: i  89: ë  A9: ⌐  C9: ╔  E9: Θ
0A: newl  2A: *    4A: J  6A: j  8A: è  AA: ¬  CA: ╩  EA: Ω
0B: ♂     2B: +    4B: K  6B: k  8B: ï  AB: ½  CB: ╦  EB: δ
0C: ♀     2C: ,    4C: L  6C: l  8C: î  AC: ¼  CC: ╠  EC: ∞
0D: cret  2D: -    4D: M  6D: m  8D: ì  AD: ¡  CD: ═  ED: φ
0E: ♫     2E: .    4E: N  6E: n  8E: Ä  AE: «  CE: ╬  EE: ε
0F: ☼     2F: /    4F: O  6F: o  8F: Å  AF: »  CF: ╧  EF: ∩
10: ►     30: 0    50: P  70: p  90: É  B0: ░  D0: ╨  F0: ≡
11: ◄     31: 1    51: Q  71: q  91: æ  B1: ▒  D1: ╤  F1: ±
12: ↕     32: 2    52: R  72: r  92: Æ  B2: ▓  D2: ╥  F2: ≥
13: ‼     33: 3    53: S  73: s  93: ô  B3: │  D3: ╙  F3: ≤
14: ¶     34: 4    54: T  74: t  94: ö  B4: ┤  D4: ╘  F4: ⌠
15: §     35: 5    55: U  75: u  95: ò  B5: ╡  D5: ╒  F5: ⌡
16: ▬     36: 6    56: V  76: v  96: û  B6: ╢  D6: ╓  F6: ÷
17: ↨     37: 7    57: W  77: w  97: ù  B7: ╖  D7: ╫  F7: ≈
18: ↑     38: 8    58: X  78: x  98: ÿ  B8: ╕  D8: ╪  F8: °
19: ↓     39: 9    59: Y  79: y  99: Ö  B9: ╣  D9: ┘  F9: ∙
1A: →     3A: :    5A: Z  7A: z  9A: Ü  BA: ║  DA: ┌  FA: ·
1B: ←     3B: ;    5B: [  7B: {  9B: ¢  BB: ╗  DB: █  FB: √
1C: ∟     3C: <    5C: \  7C: |  9C: £  BC: ╝  DC: ▄  FC: ⁿ
1D: ↔     3D: =    5D: ]  7D: }  9D: ¥  BD: ╜  DD: ▌  FD: ²
1E: ▲     3E: >    5E: ^  7E: ~  9E: ₧  BE: ╛  DE: ▐  FE: ■
1F: ▼     3F: ?    5F: _  7F: ⌂  9F: ƒ  BF: ┐  DF: ▀  FF: res
```



```
000: null 032: spa  064: @  096: `  128: Ç  160: á  192: └  224: α
001: ☺    033: !    065: A  097: a  129: ü  161: í  193: ┴  225: ß
002: ☻    034: "    066: B  098: b  130: é  162: ó  194: ┬  226: Γ
003: ♥    035: #    067: C  099: c  131: â  163: ú  195: ├  227: π
004: ♦    036: $    068: D  100: d  132: ä  164: ñ  196: ─  228: Σ
005: ♣    037: %    069: E  101: e  133: à  165: Ñ  197: ┼  229: σ
006: ♠    038: &    070: F  102: f  134: å  166: ª  198: ╞  230: µ
007: beep 039: '    071: G  103: g  135: ç  167: º  199: ╟  231: τ
008: back 040: (    072: H  104: h  136: ê  168: ¿  200: ╚  232: Φ
009: tab  041: )    073: I  105: i  137: ë  169: ⌐  201: ╔  233: Θ
010: newl 042: *    074: J  106: j  138: è  170: ¬  202: ╩  234: Ω
011: ♂    043: +    075: K  107: k  139: ï  171: ½  203: ╦  235: δ
012: ♀    044: ,    076: L  108: l  140: î  172: ¼  204: ╠  236: ∞
013: cret 045: -    077: M  109: m  141: ì  173: ¡  205: ═  237: φ
014: ♫    046: .    078: N  110: n  142: Ä  174: «  206: ╬  238: ε
015: ☼    047: /    079: O  111: o  143: Å  175: »  207: ╧  239: ∩
016: ►    048: 0    080: P  112: p  144: É  176: ░  208: ╨  240: ≡
017: ◄    049: 1    081: Q  113: q  145: æ  177: ▒  209: ╤  241: ±
018: ↕    050: 2    082: R  114: r  146: Æ  178: ▓  210: ╥  242: ≥
019: ‼    051: 3    083: S  115: s  147: ô  179: │  211: ╙  243: ≤
020: ¶    052: 4    084: T  116: t  148: ö  180: ┤  212: ╘  244: ⌠
021: §    053: 5    085: U  117: u  149: ò  181: ╡  213: ╒  245: ⌡
022: ▬    054: 6    086: V  118: v  150: û  182: ╢  214: ╓  246: ÷
023: ↨    055: 7    087: W  119: w  151: ù  183: ╖  215: ╫  247: ≈
024: ↑    056: 8    088: X  120: x  152: ÿ  184: ╕  216: ╪  248: °
025: ↓    057: 9    089: Y  121: y  153: Ö  185: ╣  217: ┘  249: ∙
026: →    058: :    090: Z  122: z  154: Ü  186: ║  218: ┌  250: ·
027: ←    059: ;    091: [  123: {  155: ¢  187: ╗  219: █  251: √
028: ∟    060: <    092: \  124: |  156: £  188: ╝  220: ▄  252: ⁿ
029: ↔    061: =    093: ]  125: }  157: ¥  189: ╜  221: ▌  253: ²
030: ▲    062: >    094: ^  126: ~  158: ₧  190: ╛  222: ▐  254: ■
031: ▼    063: ?    095: _  127: ⌂  159: ƒ  191: ┐  223: ▀  255: res
```