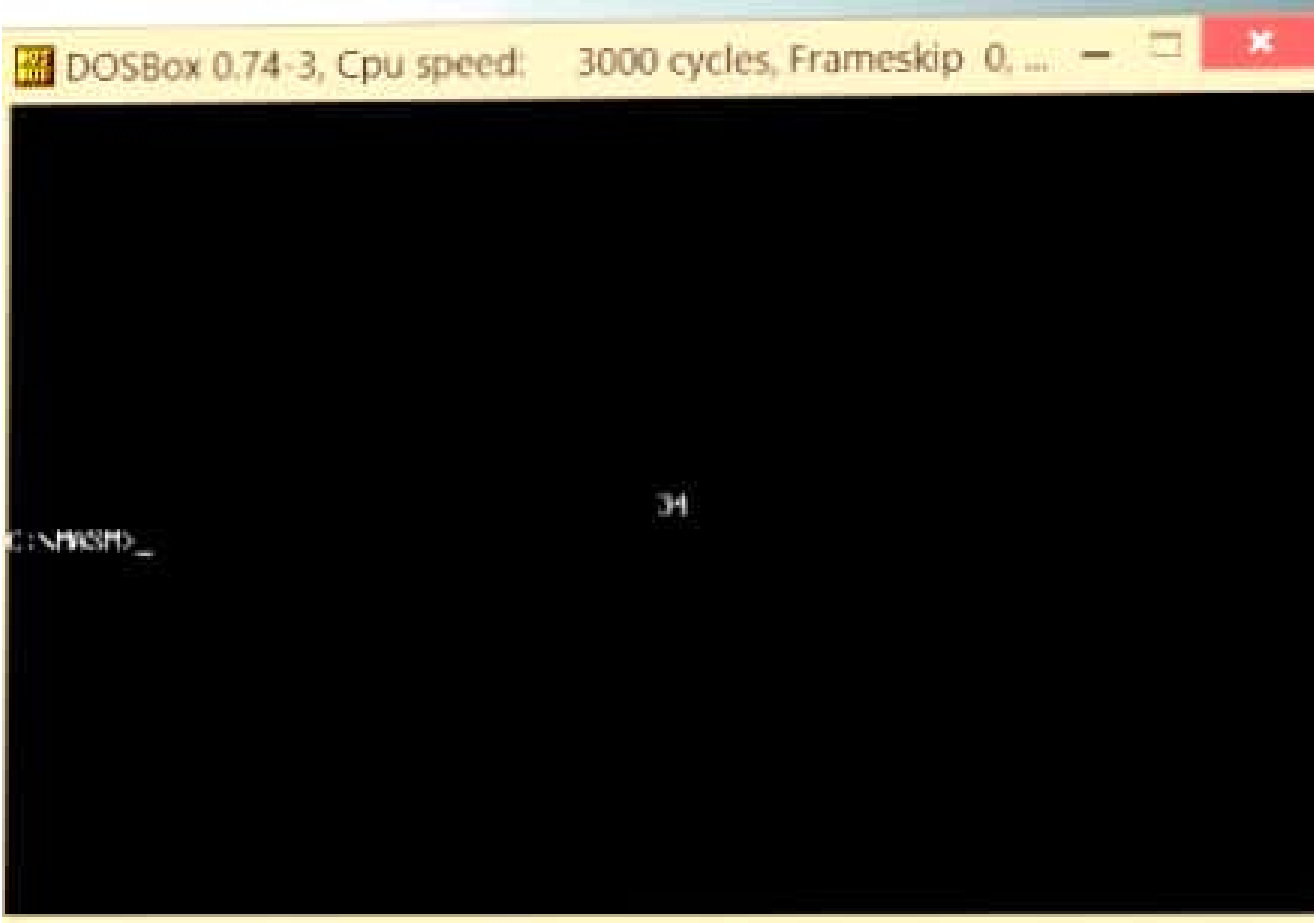
Program to find ASCII equivalent. ASCII.ASM

```
.model small
.data
msg1 db 0dh, 0ah, "Enter alphanumeric character $"
res db 02 dup(0)
.code
mov ax, @data
mov ds, ax
lea dx, msg1
call disp
mov ah, 01h  } → takes input from the keyboard.
int 21h
mov bl, al
mov cl, 4 → count
shr al, cl → before shift: AL = 34, After shift = 03
cmp al, 0ah → 10   If al > 9 or < 9.
jc digit
Add al, 07h
digit: add al, 30h
       mov res, al
       and bl, 0fh
       cmp bl, 0ah
       jc digit1
       add bl, 07h
digit1: add bl, 30h
        mov res+1, bl
        mov ah, 02h
        mov bh, 00h   → set cursor position
        mov dh, 0ch   → Page no.
        mov dl, 28h   → Row val 00 'is top)
        int 10h       → column value

        mov ah, 00h   → TEXT MODE
        mov al, 03h
        int 10h       ⇒ instructions to display string
```

```
mov  res+2 , '$'
lea  dx, res
call disp
mov  ah, 4ch
int  21h

disp  proc near
mov  ah, 09h
int  21h
ret
disp endp
end
```



Checking if a string is a palindrome or not.

```
.model small
display macro msg
        lea dx, msg
        mov ah, 09h
        int 21h
endm
.data
msg1 db 0dh, 0ah, "Enter String : $"
msg2 db 0dh, 0ah, "Reverse string :$"
msg3                "Input string is palindrome.$"
msg4                "Input string is not a palindrome$"
string db 80h DUP (?)
Rstring db 80h DUP(?)
.code
Start: mov ax, @data
       mov ds, ax
       display msg1
       ; take the string from keyboard character by
           character.
       mov SI, offset string
       XOR cl, cl
Again: Mov ah, 01h
       int 21h
       cmp al, 0dh
```

```
        je next
        mov [SI], Al
        inc SI
        inc CL
        jmp again
Next:   mov [SI], Byte ptr '$'
        ; string input over....
        dec SI
        mov ch, cl
        ; Reverse the string and store in Rstring
        mov di, offset rstring
Back:   mov al, [SI]
        mov [DI], al
        dec SI
        inc DI
        dec ch
        jnz BACK
        mov [DI], Byte ptr '$'
        Display msg2
        Display Rstring
        mov SI, offset string
        move DI, offset Rstring
AG:     MOV Al, [SI]
        cmp Al, [DI]
        jne Fail
        inc SI
        inc DI
        dec cx
```

```
        jz   SUCCESS
        jmp  AG
FAIL:   Display msg4
        jmp  final
SUCCESS: Display msg3
FINAL:  mov  ah, 4ch
        int  21h

END
```