

```

; *****
;
; CLOCK.ASM - print current date & time (Retro Unix 8086 v1 - sample program)
; -----
;
; RETRO UNIX 8086 (Retro Unix == Turkish Rational Unix)
; Operating System Project (v0.1) by ERDOGAN TAN (Beginning: 11/07/2012)
; 1.44 MB Floppy Disk
;
; [ Last Modification: 20/01/2014 ]
;
; Derivation from UNIX Operating System (v1.0 for PDP-11)
; (Original) Source Code by Ken Thompson (1971-1972)
; <Bell Laboratories (17/3/1972)>
; <Preliminary Release of UNIX Implementation Document> (Section E.12)
;
; *****
; CLOCK1.ASM (17/01/2014)
; CLOCK0.ASM (12/12/2013)
;

```

.8086

```

; UNIX v1 system calls

```

```

_rele equ 0
_exit equ 1
_fork equ 2
_read equ 3
_write equ 4
_open equ 5
_close equ 6
_wait equ 7
_creat equ 8
_link equ 9
_unlink equ 10
_exec equ 11
_chdir equ 12
_time equ 13
_mkdir equ 14
_chmod equ 15
_chown equ 16
_break equ 17
_stat equ 18
_seek equ 19
_tell equ 20
_mount equ 21
_umount equ 22
_setuideo equ 23
_getuideo equ 24
_stime equ 25
_quit equ 26
_intr equ 27
_fstat equ 28
_emt equ 29
_mdate equ 30
_stty equ 31
_gtty equ 32
_ilginse equ 33

```

```

sys macro syscallnumber, arg1, arg2, arg3

```

```

; Retro UNIX 8086 v1 system call.

```

```

    ifnb <arg1>
        mov bx, arg1
    endif
    ifnb <arg2>
        mov cx, arg2
    endif
    ifnb <arg3>
        mov dx, arg3
    endif
    mov ax, syscallnumber
    int 20h
endm

```

```

; Retro UNIX 8086 v1 system call format:

```

```

; sys syscall (ax) <arg1 (bx)>, <arg2 (cx)>, <arg3 (dx)>

```

```

code    SEGMENT PUBLIC 'CODE'
        assume cs:code,ds:code,es:code,ss:code
start_code:
        sys      _write, 1, dt_txt, dt_size
        ;jc      short terminate
        sys      _gtty, 0, 1 ; get console tty, cursor position
        ;jc      short terminate
        mov      byte ptr [ttynum], al
        mov      ah, 7
        cmp      al, ah ; cmp al, 7
        jna      short @f
        mov      byte ptr [lf_b], al ; 7 = beep
        jmp      short clk0
@@:
        mov      word ptr [cursor_pos], bx ; cursor position
clk0:
        xor      bx, bx
        xor      cx, cx
        ;push    bx ; 0, hw
        ;push    cx ; 0, lw
        sys      _time
        ; DX:AX = Unix epoch time
        ;pop     cx ; previous time, lw
        ;pop     bx ; previous time, hw
        cmp      ax, cx
        jne      short clk1
        cmp      dx, bx
        je       short clk4
clk1:
        push     dx ; current time, hw
        push     ax ; current time, lw
        call     ctime
        sys      _write, 1, cbuf, 25
        sys      _gtty, 0, 0
        or       bx, bx
        jz       short clk2
        sys      _read, 0, chr, 1
        cmp      byte ptr [chr], 1Bh ; ESC key
        je       short clk3 ; exit
clk2:
        cmp      byte ptr [ttynum], 8
        jnb      short clk_pt
        sys      _write, 1, dt_txt, dt_size
        jmp      short @f
clk_pt:
        mov      dx, word ptr [cursor_pos]
        mov      cx, 0FFFFh ; set cursor position only
        xor      bx, bx ; set for console tty
        sys      _stty
@@:
        pop      cx ; current -> previous time, lw
        pop      bx ; current -> previous time, hw
        jmp      short clk0
clk3:
        pop      ax
        pop      ax
terminate:
        sys      _exit
clk4:
        nop
        nop
        nop
        nop
        jmp      short clk0

include ctime.inc
cursor_pos: dw 0
dt_txt:
        db 0Dh
lf_b:    db 0Ah
        db 'Current Date & Time : '
dt_size equ $ - offset dt_txt
chr:     db 0
ttynum:  db 0
code     ends

        end start_code

```