**Introduction**

Turbo Assembler (TASM) is a [computer](https://en.wikipedia.org/wiki/Computer) [assembler](https://en.wikipedia.org/wiki/Assembly_language#Assembler) (software for program development) developed by [Borland](https://en.wikipedia.org/wiki/Borland) which runs on and produces code for 16- or 32-bit [x86](https://en.wikipedia.org/wiki/X86) [DOS](https://en.wikipedia.org/wiki/DOS) or [Microsoft Windows](https://en.wikipedia.org/wiki/Microsoft_Windows). It can be used with Borland's [high-level language](https://en.wikipedia.org/wiki/High-level_language) [compilers](https://en.wikipedia.org/wiki/Compiler), such as [Turbo Pascal](https://en.wikipedia.org/wiki/Turbo_Pascal), [Turbo Basic](https://en.wikipedia.org/wiki/Turbo_Basic), [Turbo C](https://en.wikipedia.org/wiki/Turbo_C) and [Turbo C++](https://en.wikipedia.org/wiki/Turbo_C%2B%2B). The Turbo Assembler package is bundled with the Turbo [Linker](https://en.wikipedia.org/wiki/Linker_(computing)), and is interoperable with the [Turbo Debugger](https://en.wikipedia.org/wiki/Turbo_Debugger). TASM can assemble [Microsoft Macro Assembler](https://en.wikipedia.org/wiki/Microsoft_Macro_Assembler) (MASM) source using its MASM mode and has an ideal mode with a few enhancements. [Object-Oriented programming](https://en.wikipedia.org/wiki/Object-Oriented_programming) has been supported since version 3.0. The last version of Turbo Assembler is 5.4, with files dated 1996 and patches up to 2010; it is still supplied with [Delphi](https://en.wikipedia.org/wiki/Embarcadero_Delphi) and [C++Builder](https://en.wikipedia.org/wiki/C%2B%2BBuilder).

TASM itself is a 16-bit program; it will run on 16- and 32-bit versions of Windows, and produce code for the same versions. There are ways to run 16-bit programs such as TASM on 64-bit Windows (e.g., on a [virtual machine](https://en.wikipedia.org/wiki/Virtual_machine)), but it will not generate 64-bit Windows code.

**Digital Clock**

A simple assembly code for the 8086 microprocessor to display a digital clock in real time

**Get System Time**

INT 21h / AH=2CH

Read System Date and Load

CH = Hours

CL = Minutes

DH = Seconds

DL =1/100 Seconds

**Get System Date**

INT 21h / AH=2AH

Read System Date by using AH=2AH interrupt

CX = year (1980- 2099)

DH = month

DL = Day

AL = Day of Week (00-Sunday)

e.g.

MOV CX, 0000H        ; Initially load all registers with 00H

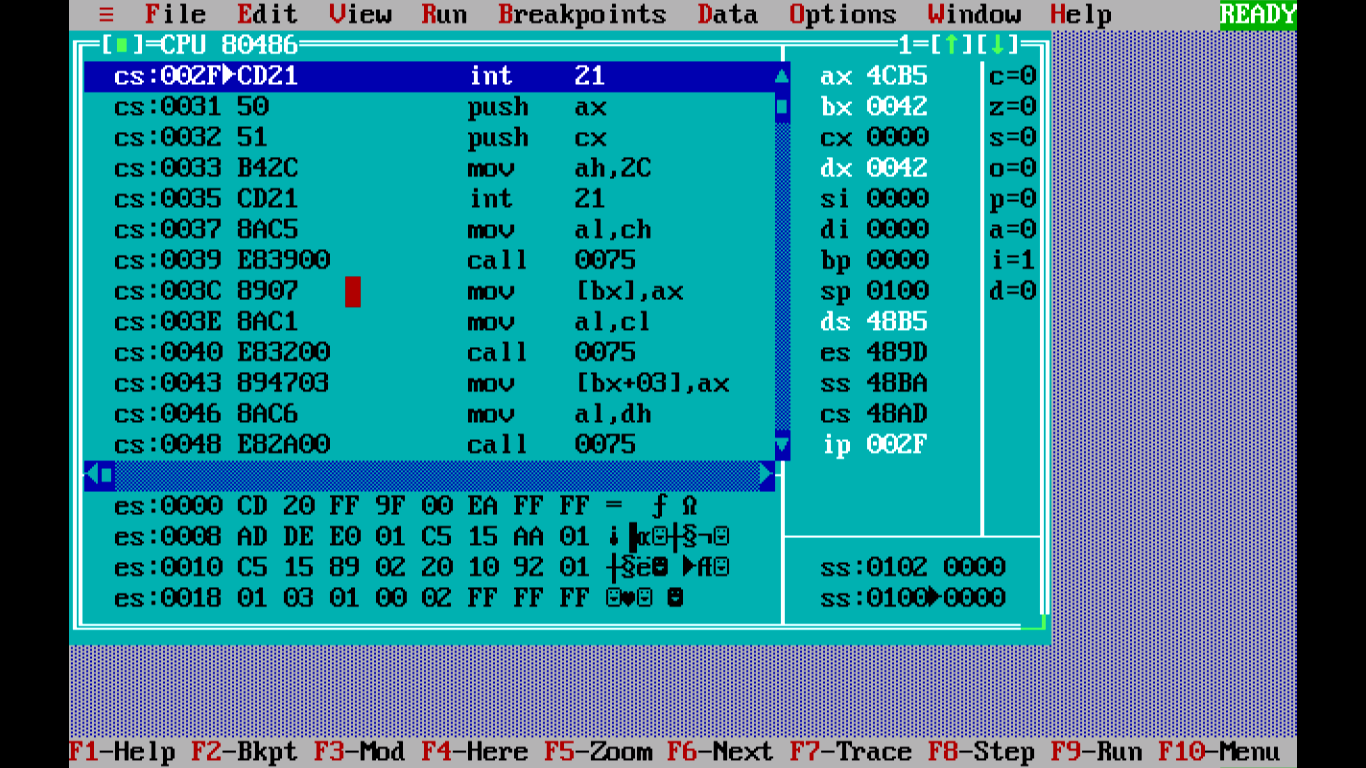
MOV DX,0000H

MOV AL, 00H

MOV AH, 2AH           ; Command to read system date

INT 21H

**Output**

****

