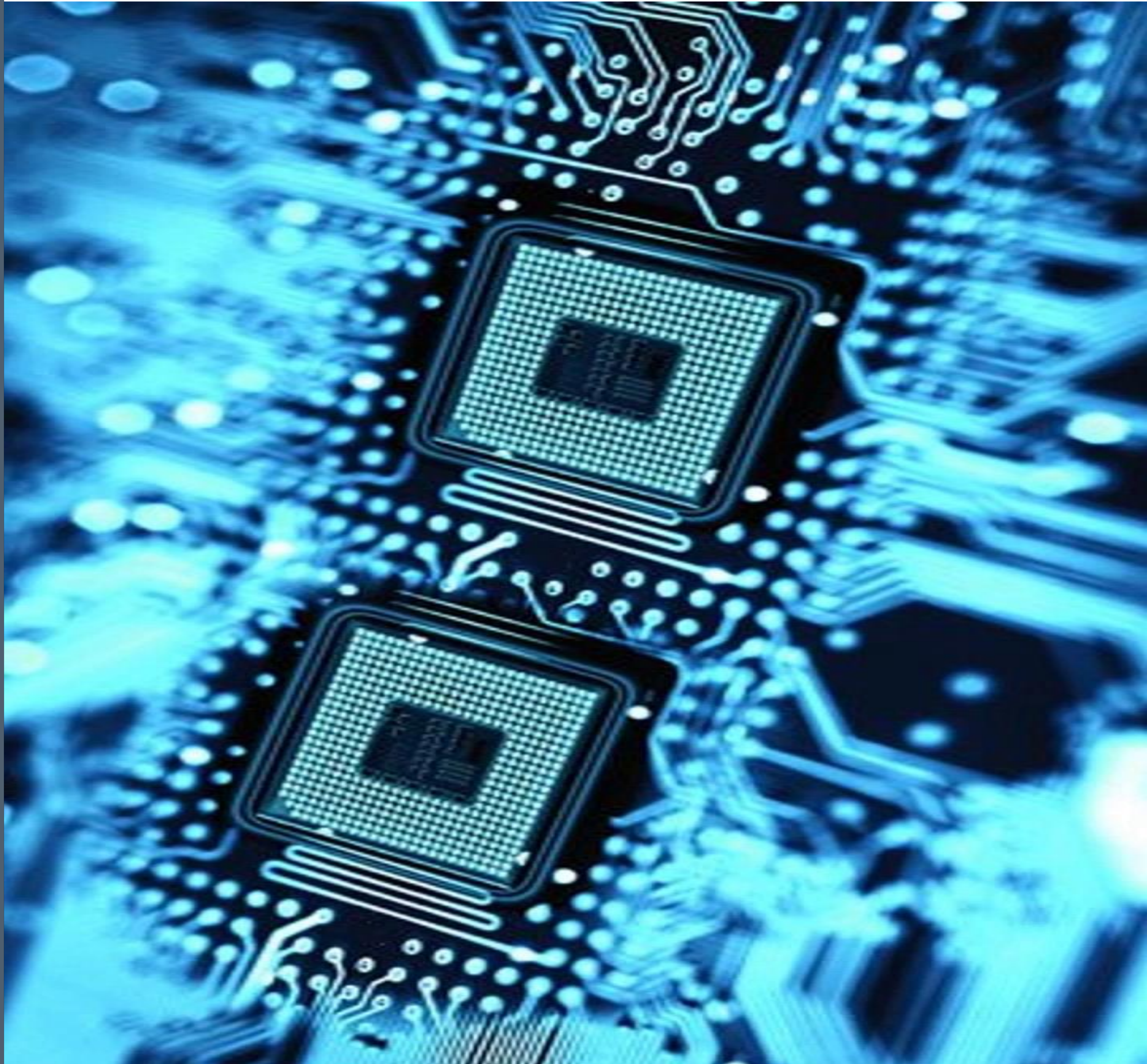
Computer organization & architecture

Course by: Dr. Ahmed Sadek

Lab By: Mahmoud Badry

MDA 8086 Kit

Project



IN and OUT instruction

- **Output** from **AL** to the specified **address:**

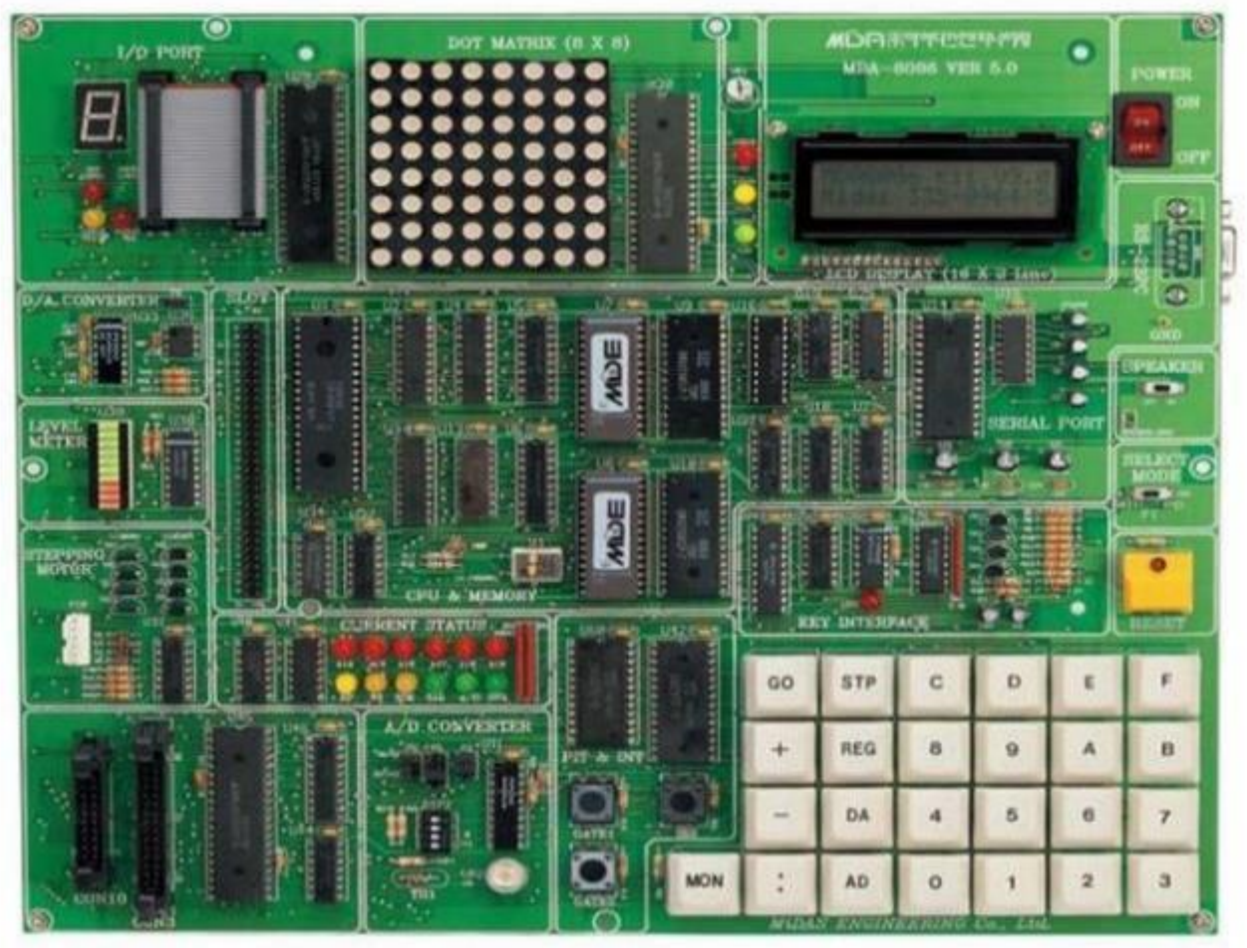
```
Out address , al
```

- **Input** from the specified **address** into **AL**.

```
In al , address
```

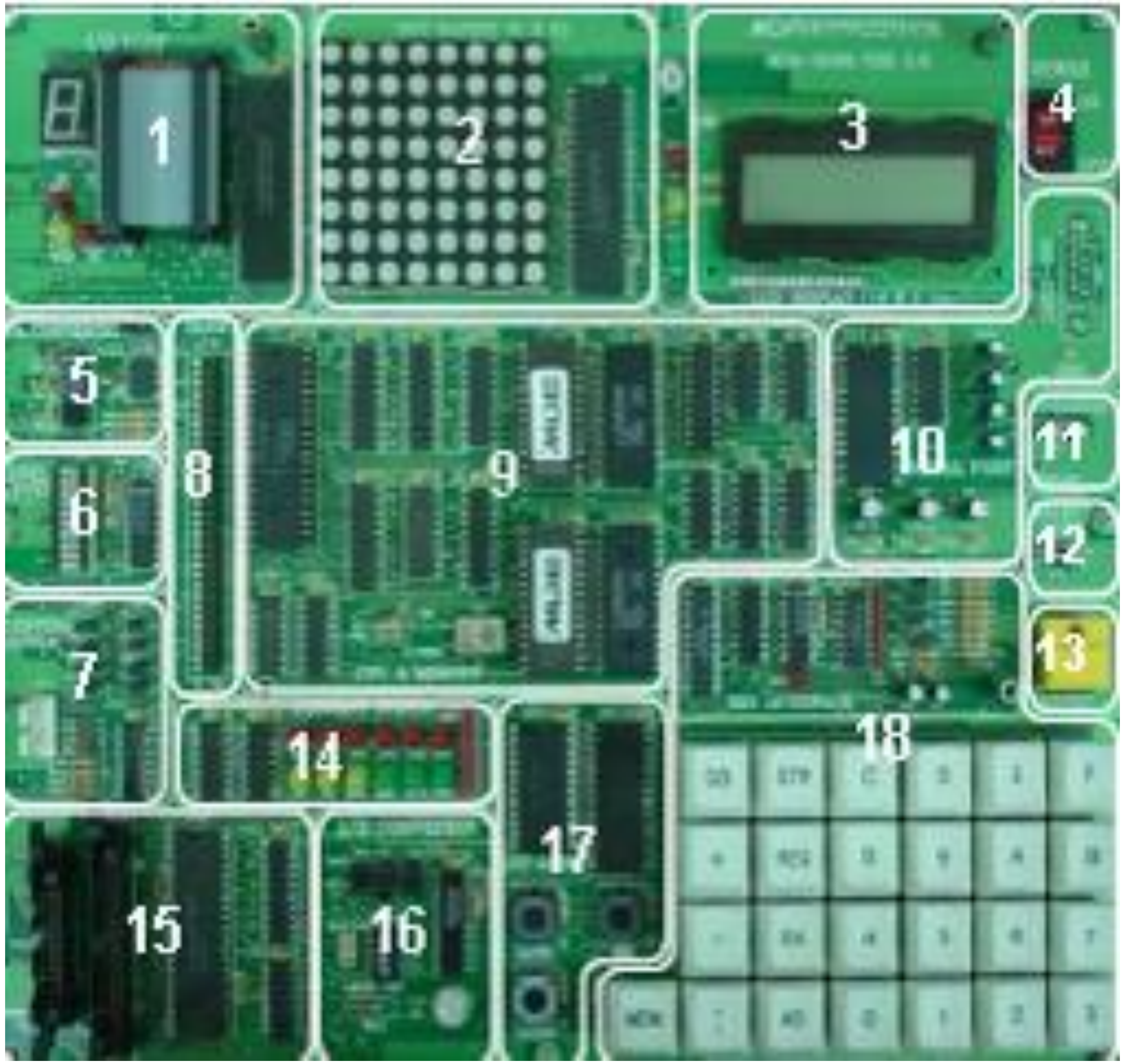
MDA 8086 Kit basics

Introduction



- **MDA-8086** is **kit** having **microprocessor** and **various** other **components** for the detailed understanding of **8086 microprocessor**.
  - **MDA** is a company name ”**Midas Engineering**“.
  - It consists of an **LCD** screen, **16** data **keys** and **10 function** keys.
  - It has 2 **RAM** (**2x32Kb**) and 2 **ROM** (**2x32Kb**) included.
  - It can be operated in two **modes**: **KIT** MODE , **PC** MODE
  - It also includes a **7 segment** , **dot matrix**, D/A , A/D , Level meter and stepping motor projects.

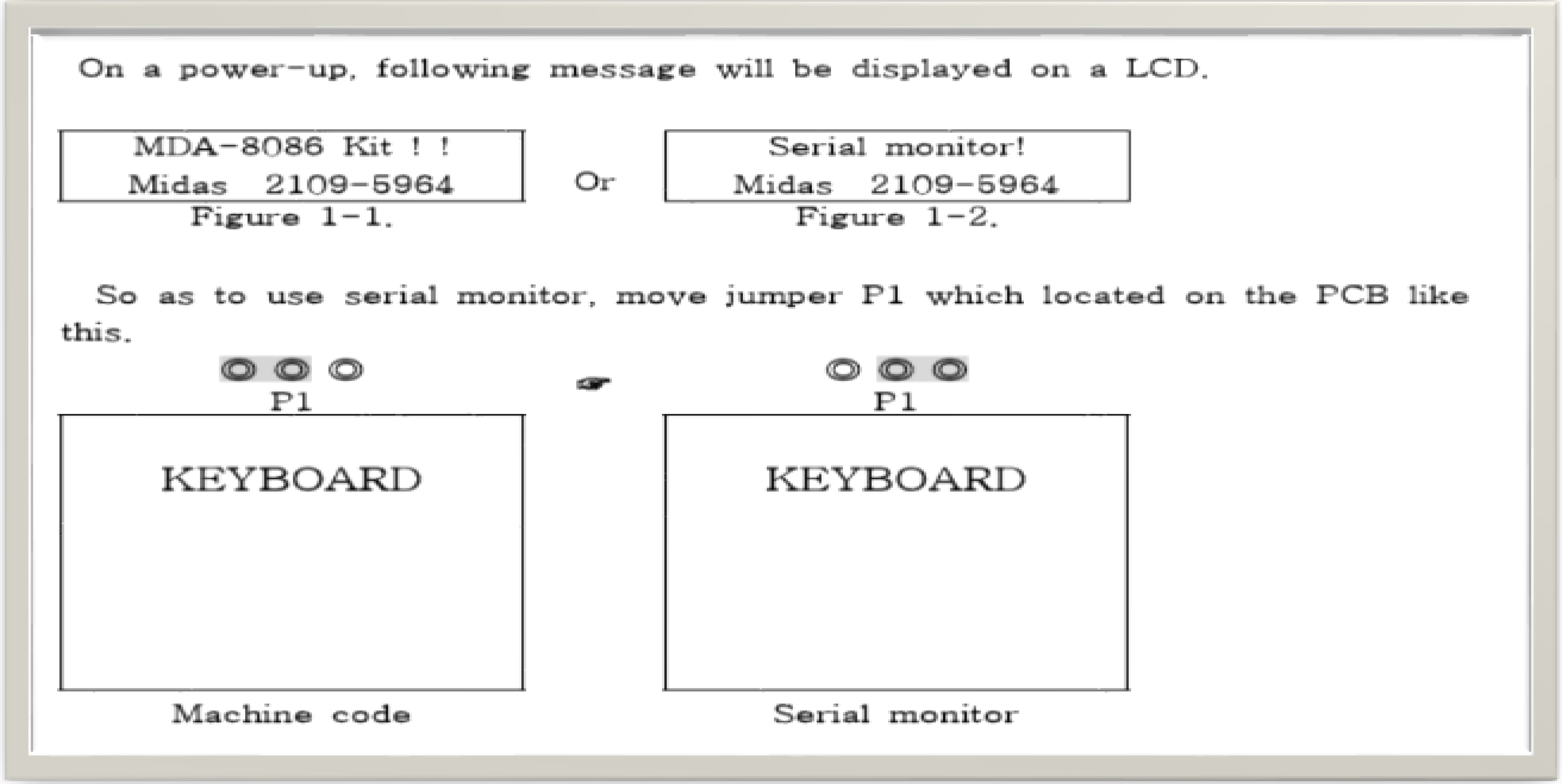
Kit Components



1:I/O Port
2:Dot Matrix Display
3:LCD Display,
4:Power switch,
5:DAC,
6:Level meter.
7:Stepping motor interface,
8:Extension slot,
9:CPU & memory,
10:Serial port RS-232C connector
11:Speaker switch,
12:Mode selection switch,
13:Reset switch,
14:Status display,
15:Application ports,
16:ADC,
17:Timer interrupt controller,
18:Key interface

Basic operation

- Whenever **RES** is pressed, the **display** becomes **FIGURE** 1 and user can **operate keyboard** only in this situation.
- We will work in **serial monitor** mode so keep **P1** at serial monitor



On a power-up, following message will be displayed on a LCD.

MDA-8086 Kit ! ! Midas 2109-5964	Or	Serial monitor! Midas 2109-5964
Figure 1-1.		Figure 1-2.

So as to use serial monitor, move jumper P1 which located on the PCB like this.

P1
KEYBOARD
P1
KEYBOARD

Machine code

Serial monitor

Serial monitor

- **Serial monitor** is the basic **monitor** program to do data **communicate** between **MDA-8086** and **computer**.
- How to **use serial monitor**? move jumper **P1** which located on the **PCB** .
- **8086** cannot take a **simple** written **text** file. It takes **machine code** (**hex** files).
- So we need an **assembler** to convert the assembly file to **machine** or **hex** file
- For **develop** the program more **efficiently**, make source file using **editor program** of computer then **assembling** this **file** and make **HEX(Intel file format**), down-load to **MDA-8086** using with **serial monitor**.

Preparing hex code for the kit

Convert assembly file to hex file

- **First** you write the **code** in assembly language ”**.asm**”
- Then use **assembler** like **MASM** (Macro Assembler) to convert it to **machine code** object file(.obj)
- Then use **program loader LOD186** that convert it to **hex file absolute** file(**.abs**) that the kit use.
- Create a **directory** named as **8086kit** in the hard drive **D**. And now **accommodate** the following **program** files under this **directory**. These **programs** may easily be collected from the **CD** that has been **supplied** with the **MDA-8086** trainer.

File Name	Purpose
MASM.EXE	To create *.LST and *.OBJ files form *.ASM file
LOD186.EXE	To create ‘Absolute (*.ABS)’ file from *.obj file
A.BAT	To create ‘Absolute (*.ABS)’ file from *.ASM file

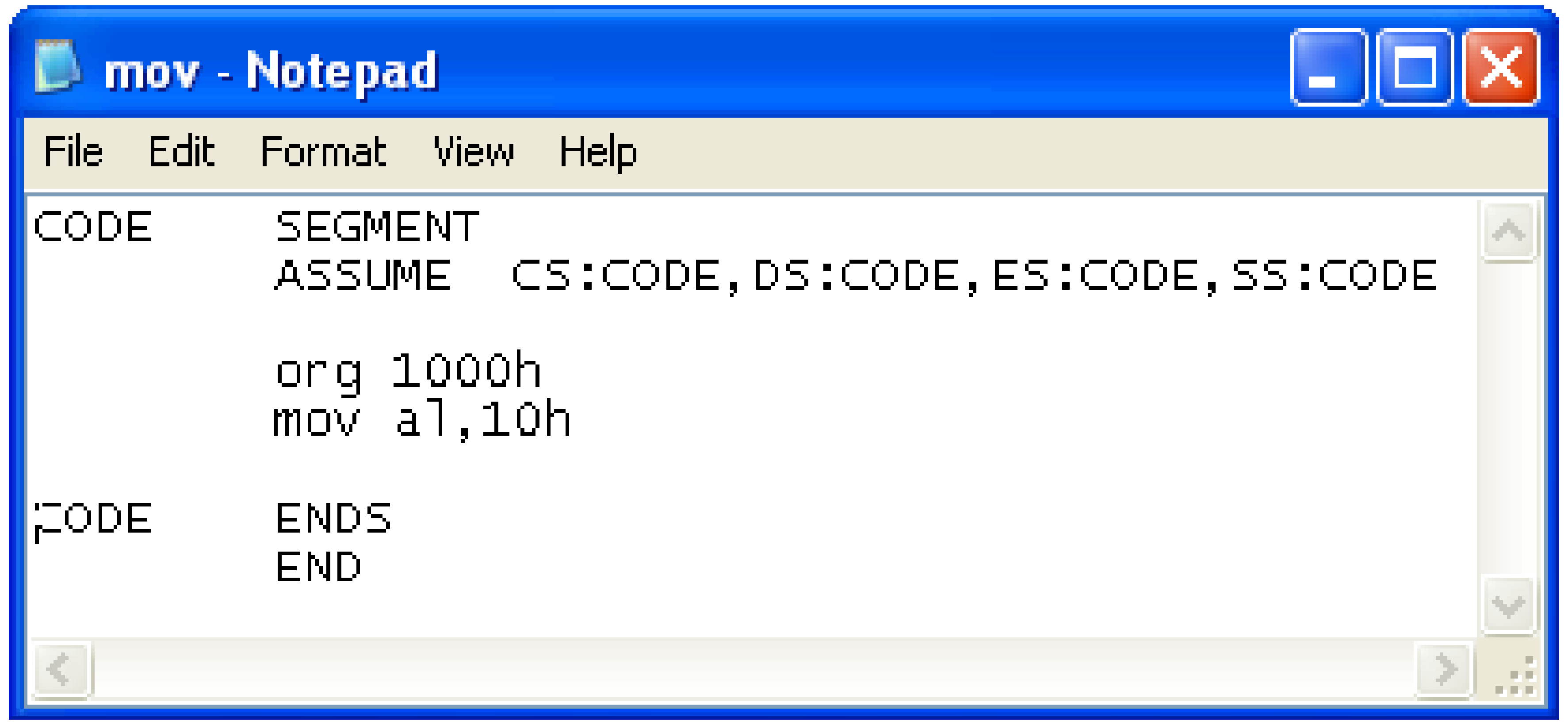
Step 1: Writing your program in you favorite text editor

1. **Write** your **code** in a **notepad** file with necessary **instructions**.
2. **Save** the **file** as **mov.asm** (for example) in the folder created before **8086 kit** in **D drive**.
3. **Type** the **following lines** (called Assembler Statements) at the **top** of your **program**.

```
CODE  SEGMENT

ASSUME    CS:CODE,DS:CODE,ES:CODE,SS:CODE

ORG 1000h
```

4. Type the **following assembler** statements at the **end** of your **assembly program**.

```
CODE       ENDS

END
```



```
CODE     SEGMENT
         ASSUME   CS:CODE,DS:CODE,ES:CODE,SS:CODE

         org 1000h
         mov al,10h

CODE     ENDS
         END
```

Step 2: Setup MASM ASSEMBLER

1. Go to **command prompt** window. Go to **folder 8086 kit.** (You must place MASM,LOD186,A (kit files in your work folder)
2. **Set up MASM ASSEMBLER**

   D:\8086 kit>MASM mov.asm

   Microsoft (R) Macro Assembler Version 5.10

   Copyright (C) Microsoft Corp 1981, 1988. All right reserved.

   Object filename [C:mov.OBJ]: (Press enter)

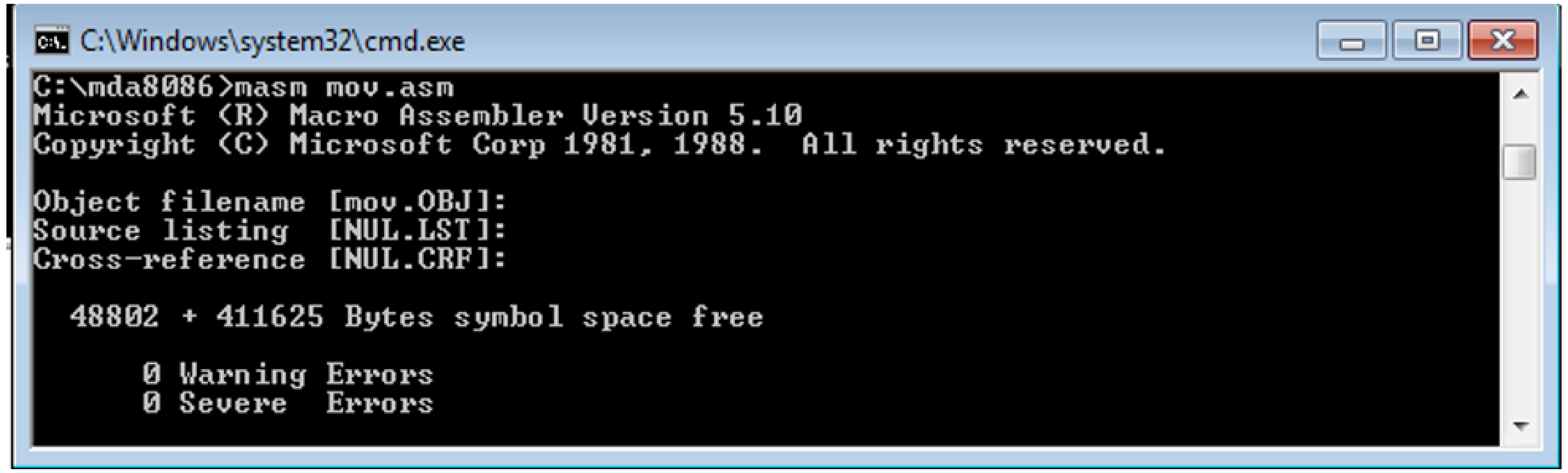
   Source listing [NUL.LST]: (Press enter)

   Cross reference [NUL.CRF]: (Press enter)

   47838 + 452253 Bytes symbol space free

   0 Warning Errors

   0 Severe Errors



```
C:\mda8086>masm mov.asm
Microsoft (R) Macro Assembler Version 5.10
Copyright (C) Microsoft Corp 1981, 1988.  All rights reserved.

Object filename [mov.OBJ]:
Source listing  [NUL.LST]:
Cross-reference [NUL.CRF]:

  48802 + 411625 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors
```

Step 3: Make HEX(ABS) file

1. **Begin** with the **same folder** we were in before.
2. **Run Lod186**

   D:\8086 kit>LOD186 mov

   Paragon LOD186 Loader - Version 4.0h

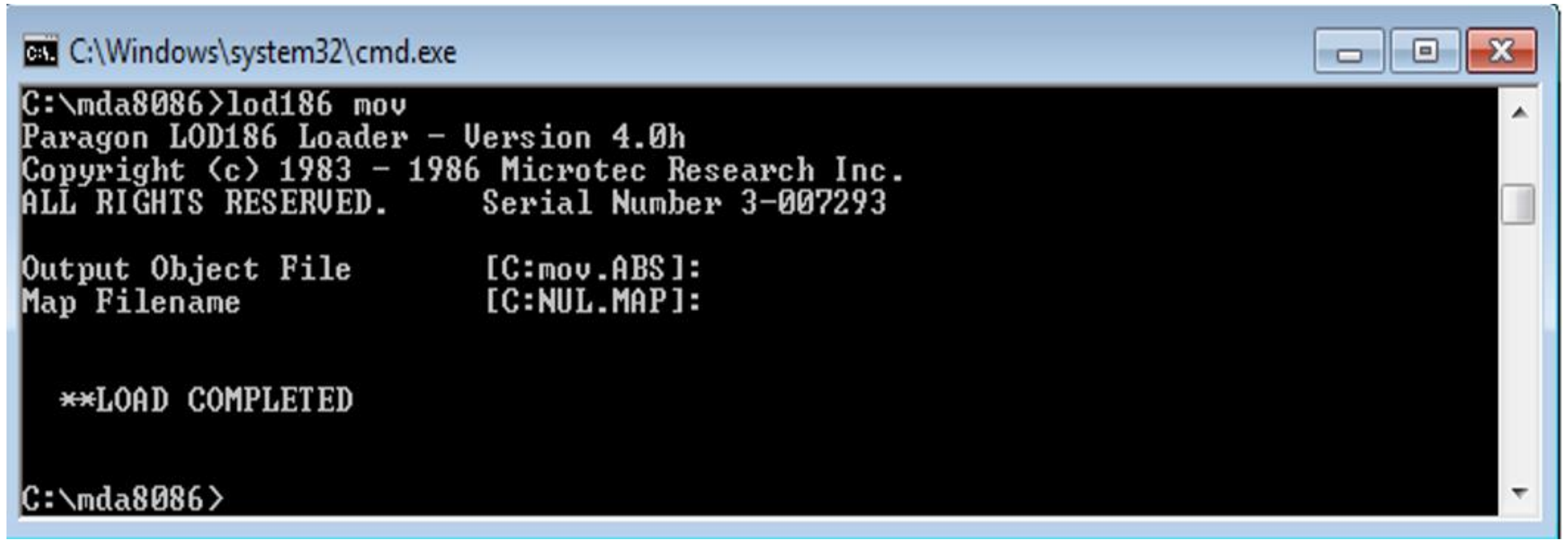
   Copyright (C) 1983 - 1986 Microtec Research Inc.

   ALL RIGHT RESERVED.

   Output Object File [D:mov.ABS]:

   Map Filename [D:NUL.MAP]:

   **LOAD COMPLETE

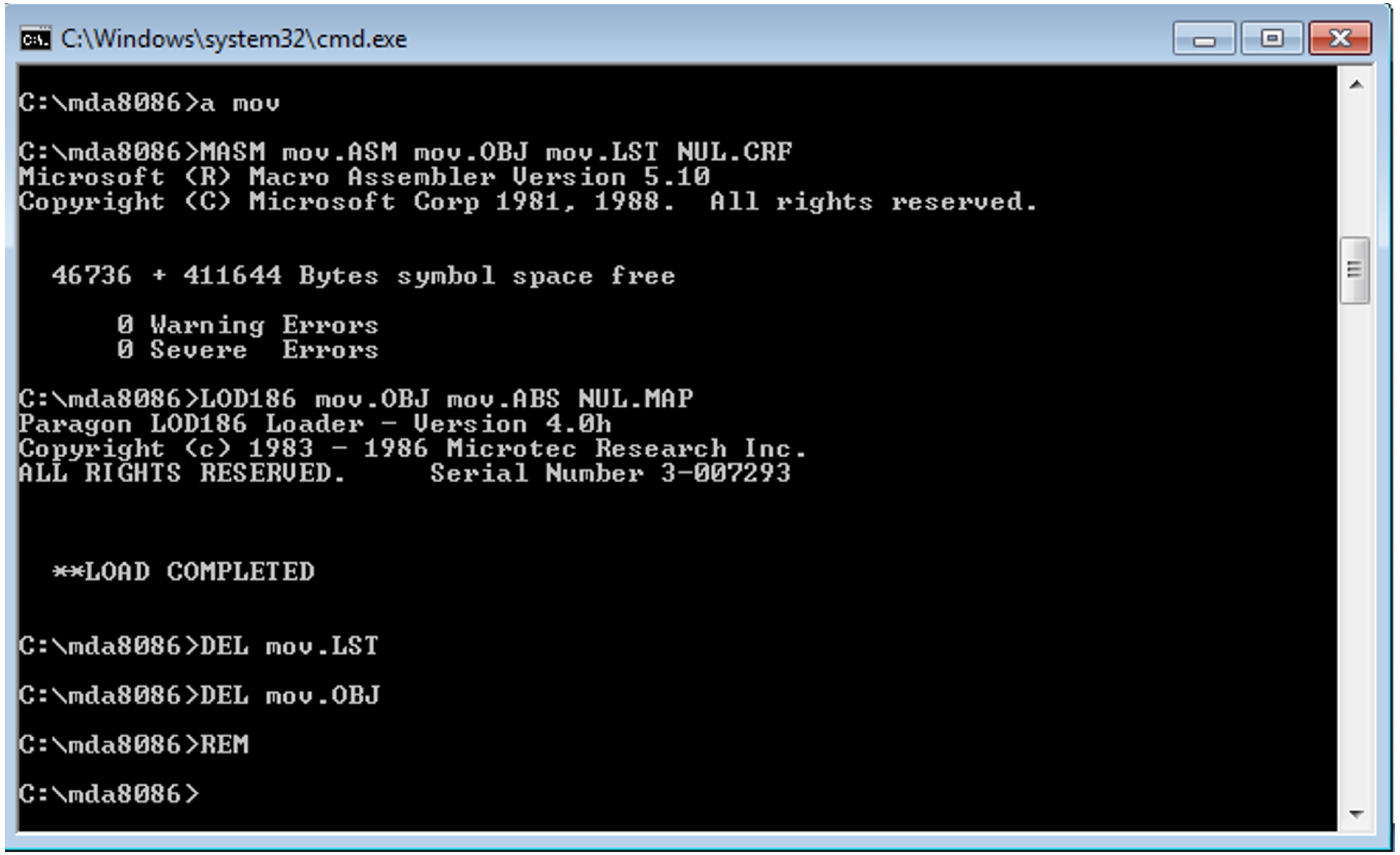


```
C:\mda8086>lod186 mov
Paragon LOD186 Loader - Version 4.0h
Copyright (c) 1983 - 1986 Microtec Research Inc.
ALL RIGHTS RESERVED.     Serial Number 3-007293

Output Object File       [C:mov.ABS]:
Map Filename             [C:NUL.MAP]:


  **LOAD COMPLETED


C:\mda8086>
```

Output ABS directly using a.bat

1. Put a.bat, MASM.EXE, LOD186.EXE, mov.asm in the same folder.
2. Run a.bat

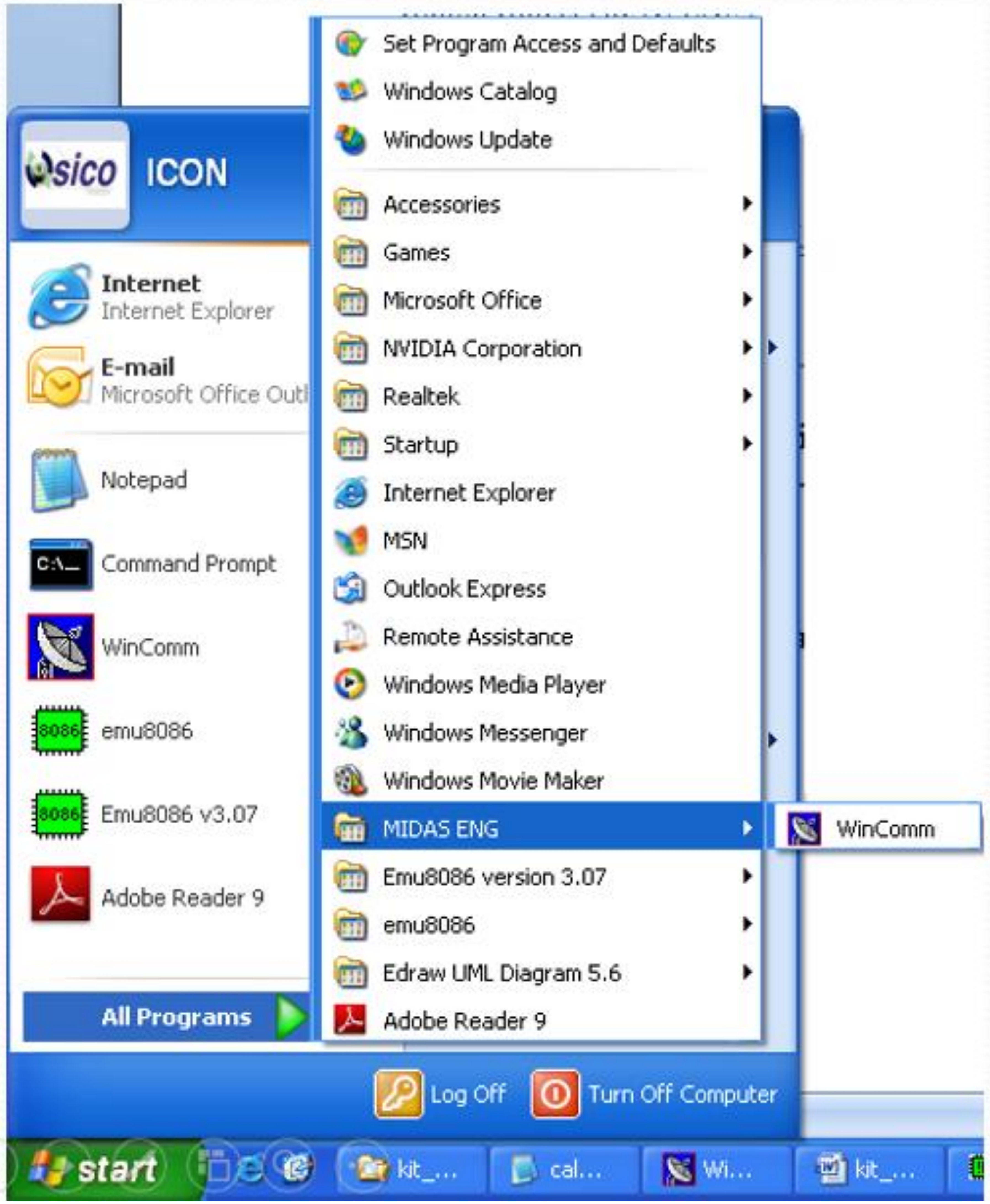
   D:\8086 kit>a mov

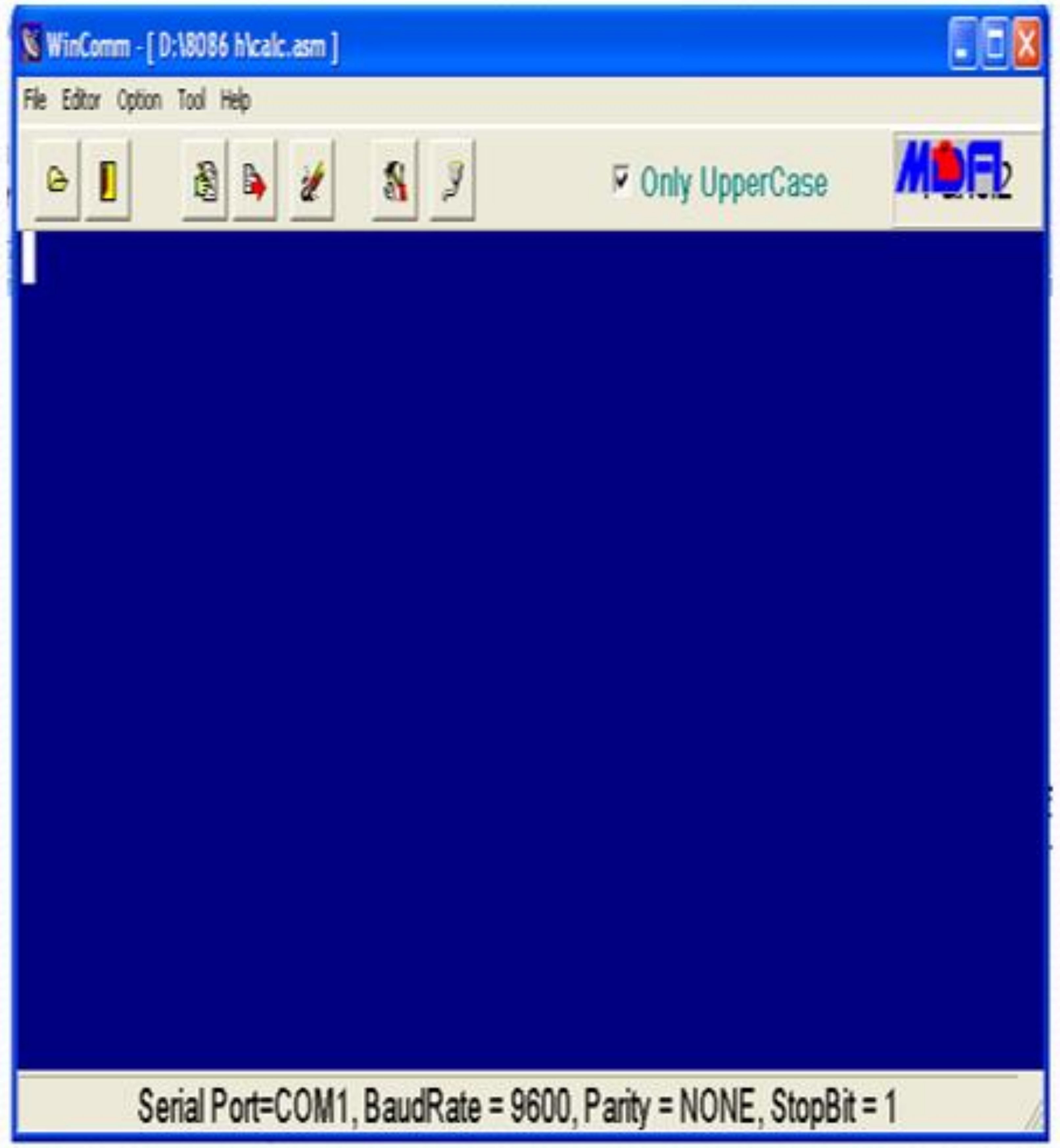


```
C:\mda8086>a mov

C:\mda8086>MASM mov.ASM mov.OBJ mov.LST NUL.CRF
Microsoft (R) Macro Assembler Version 5.10
Copyright (C) Microsoft Corp 1981, 1988.  All rights reserved.


  46736 + 411644 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

C:\mda8086>LOD186 mov.OBJ mov.ABS NUL.MAP
Paragon LOD186 Loader - Version 4.0h
Copyright (c) 1983 - 1986 Microtec Research Inc.
ALL RIGHTS RESERVED.     Serial Number 3-007293



  **LOAD COMPLETED


C:\mda8086>DEL mov.LST

C:\mda8086>DEL mov.OBJ

C:\mda8086>REM

C:\mda8086>
```

Download the hex file in MDA 8086

Download the hex file in MDA 8086

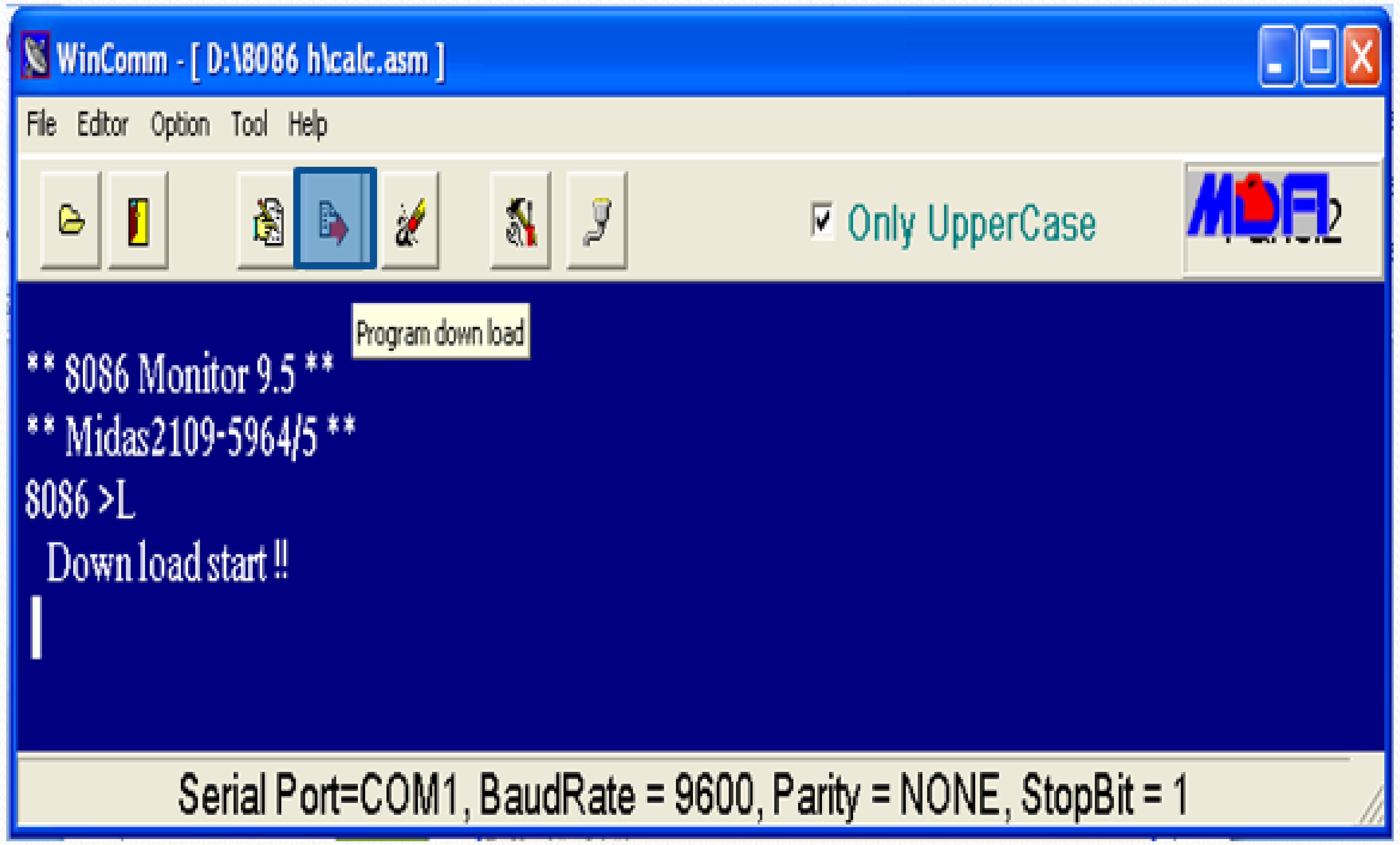




Serial Port=COM1, BaudRate = 9600, Parity = NONE, StopBit = 1

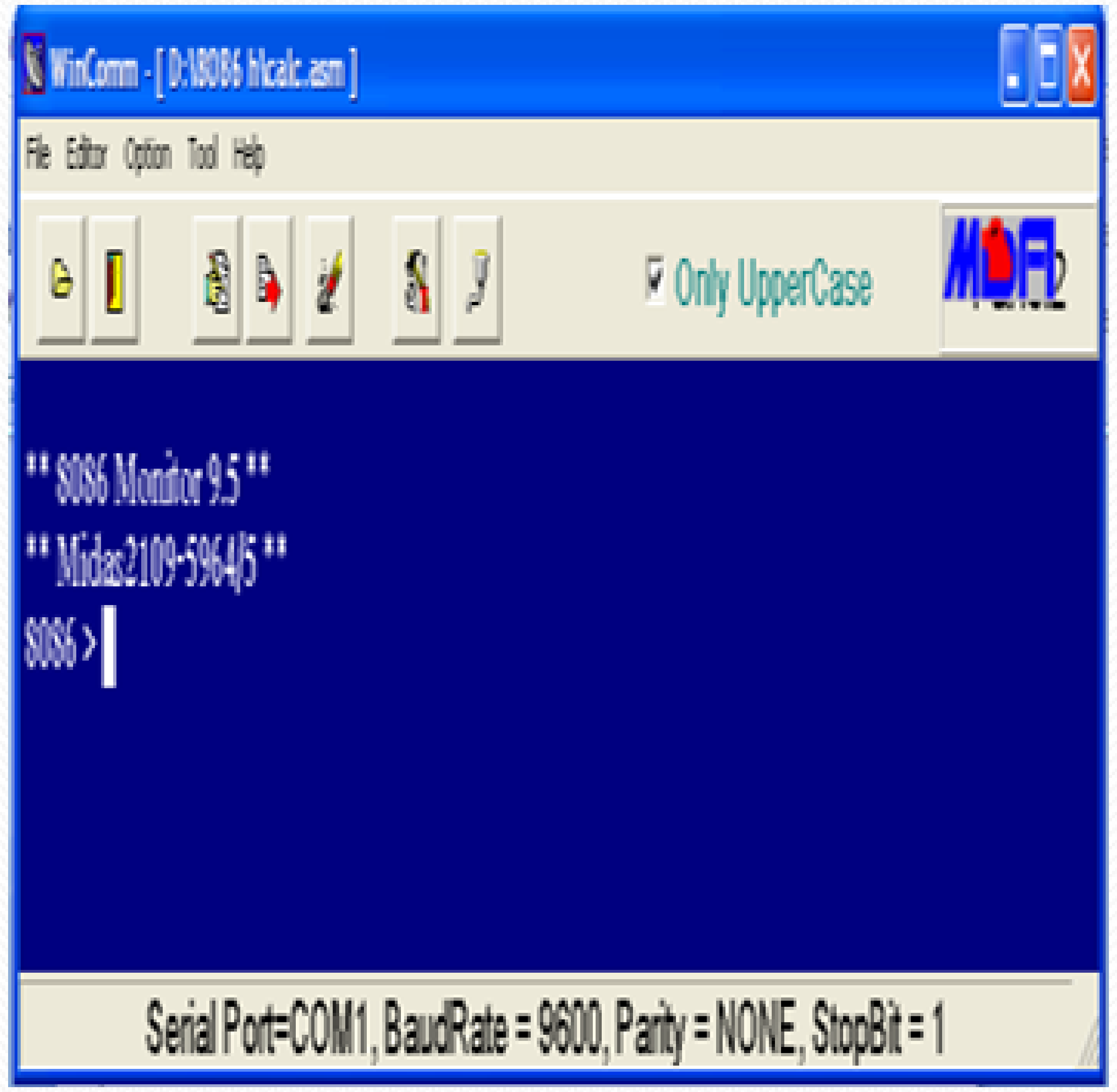
Download the hex file in MDA 8086

- **Power down** and **Power up** the MDA-8086 trainer. Press the **RESET** key of the trainer to get the prompt **message** on the **screen** of the your **PC**.
- Type **L**
- **Press** program **download button**


WinComm - [D:\8086 h\calc.asm]
File Editor Option Tool Help
☑ Only UpperCase
MDA
Program down load

```
** 8086 Monitor 9.5 **
** Midas2109-5964/5 **
8086 >L
 Down load start !!
```

Serial Port=COM1, BaudRate = 9600, Parity = NONE, StopBit = 1

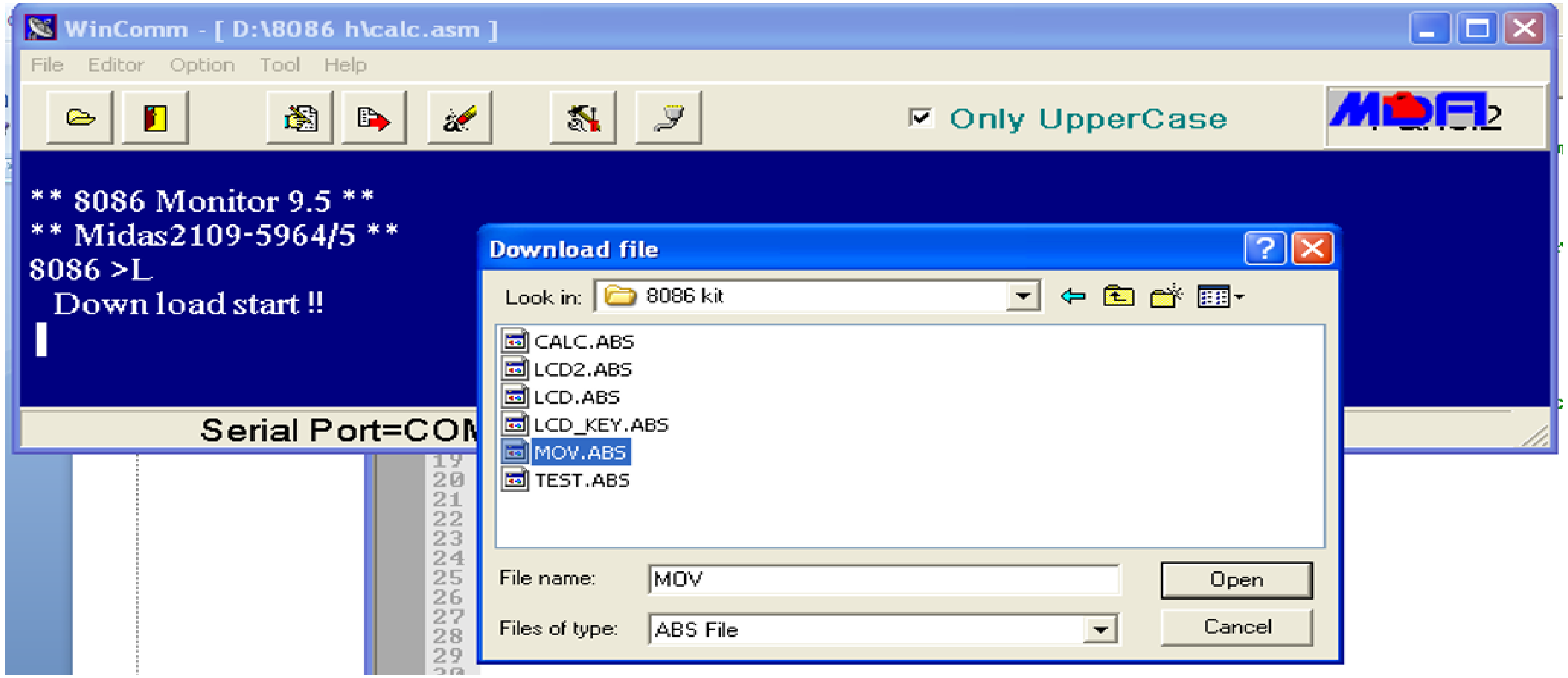

WinComm - [D:\8086 h\calc.asm]
File Editor Option Tool Help
☑ Only UpperCase
MDA

```
** 8086 Monitor 9.5 **
** Midas2109-5964/5 **
8086 >
```

Serial Port=COM1, BaudRate = 9600, Parity = NONE, StopBit = 1

Download the hex file in MDA 8086

- From the **menu** appear **choose**
  - **file** type: **ABS** file
  - **file name**: choose the file that you create with **extension.ABS** → **mov.abs**
  - Using **T** command for **single** step, or **G** for **run** the program



WinComm - [D:\8086 h\calc.asm]

☑ Only UpperCase

```
** 8086 Monitor 9.5 **
** Midas2109-5964/5 **
8086 >L
  Down load start !!
```

Serial Port=COM

Download file

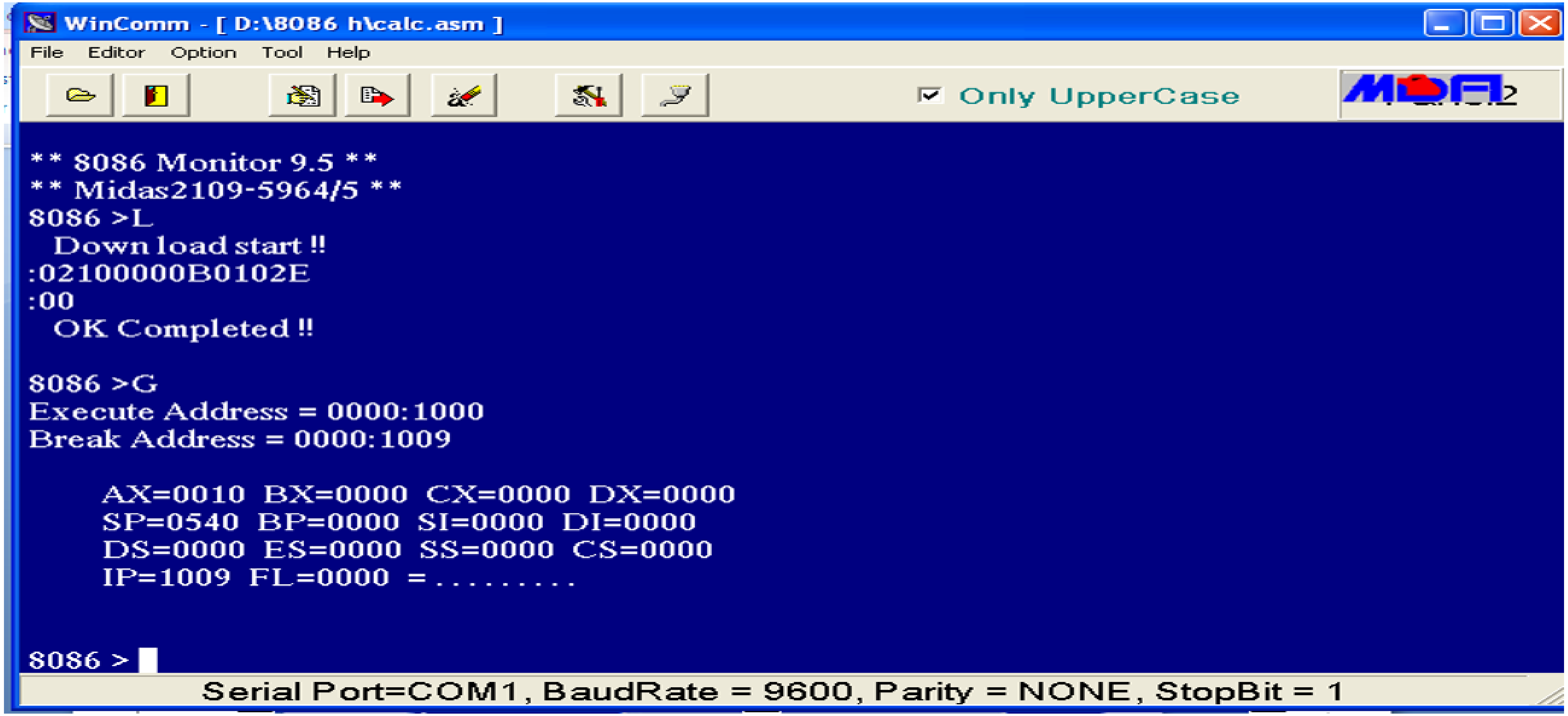
Look in: 8086 kit

CALC.ABS
LCD2.ABS
LCD.ABS
LCD_KEY.ABS
MOV.ABS
TEST.ABS

File name: MOV

Files of type: ABS File

Download the hex file in MDA 8086



WinComm - [D:\8086 h\calc.asm]
File Editor Option Tool Help
☑ Only UpperCase

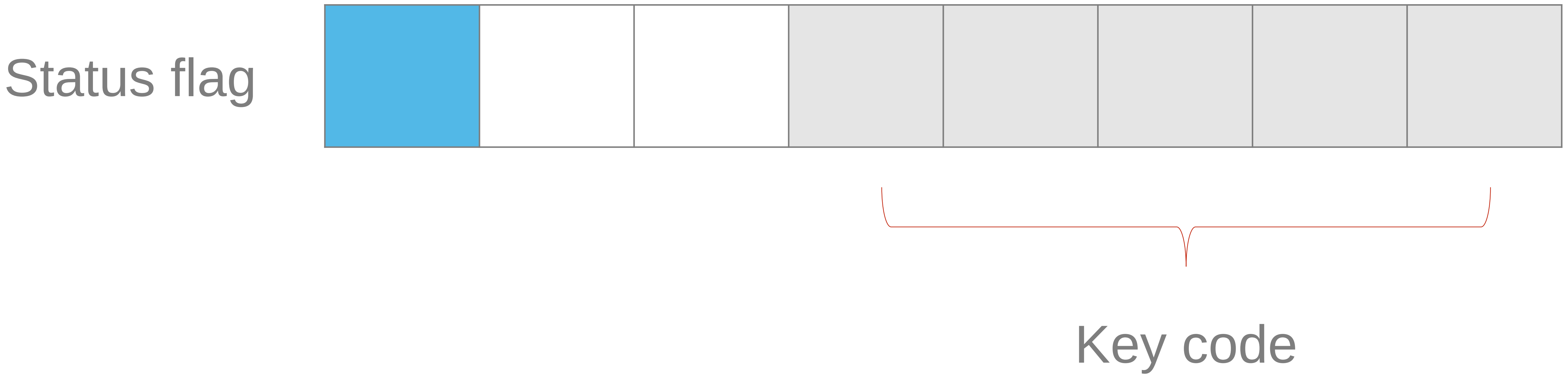
```
** 8086 Monitor 9.5 **
** Midas2109-5964/5 **
8086 >L
  Down load start !!
:02100000B0102E
:00
  OK Completed !!

8086 >G
Execute Address = 0000:1000
Break Address = 0000:1009

      AX=0010  BX=0000  CX=0000  DX=0000
      SP=0540  BP=0000  SI=0000  DI=0000
      DS=0000  ES=0000  SS=0000  CS=0000
      IP=1009  FL=0000  = .........


8086 >
```

Serial Port=COM1, BaudRate = 9600, Parity = NONE, StopBit = 1

Code examples

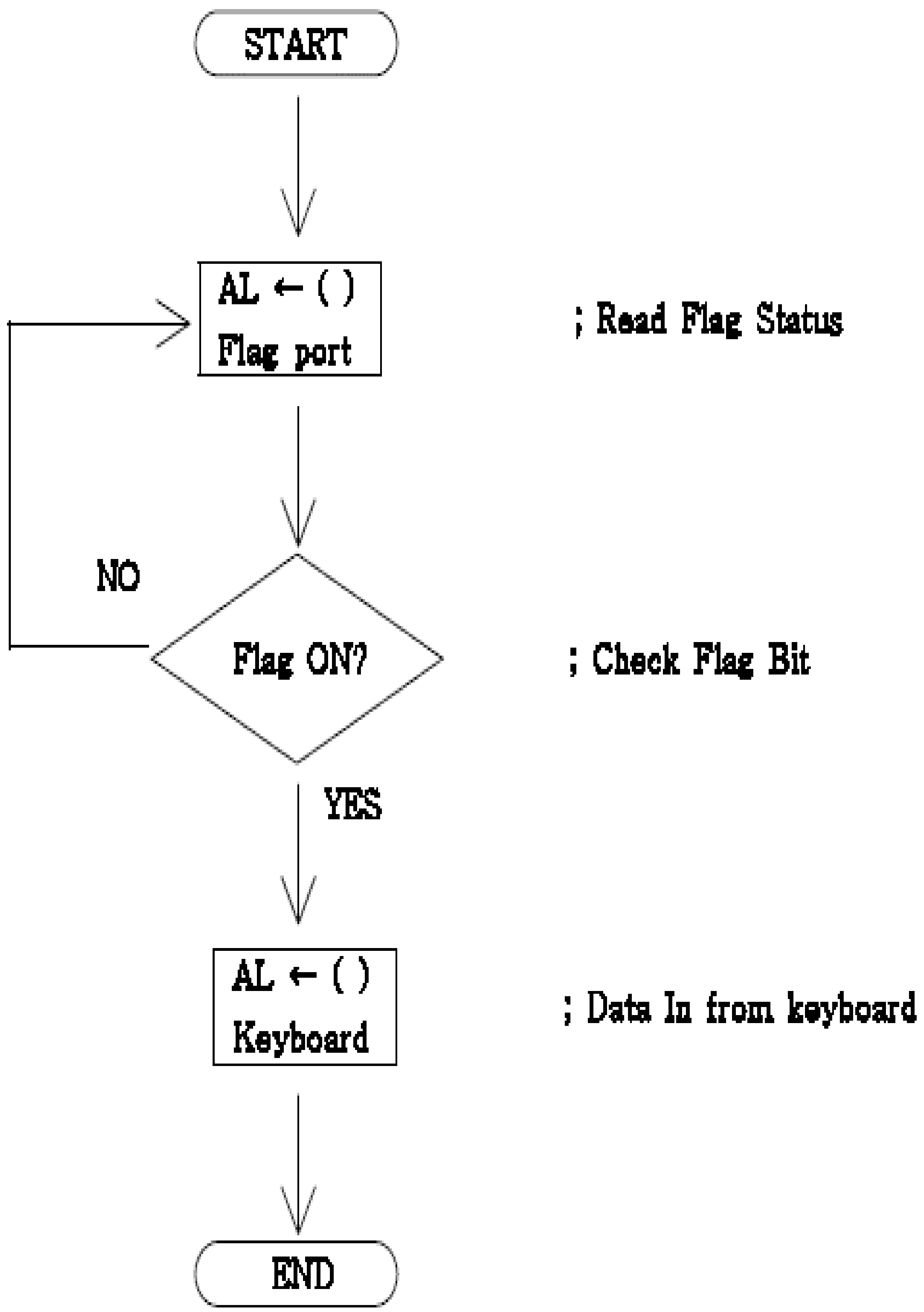
Keyboard interface

- The **keyboard scanner** generates a **code** for each **key**. There are **24 codes** for **24 keys**.
- The **keyboard register** is **8-bits**. The **least significant 5** bits contain the **codes** of the keys.
- The **most significant bit** is the **status flag**. When the status flag = '0', then the **register holds** a **valid** code for a **key**.
- **After** you **read** the **code** from the keyboard, you have to **write** it **back** to the **keyboard register** so that the keyboard can **scan another** key.



Keyboard interface

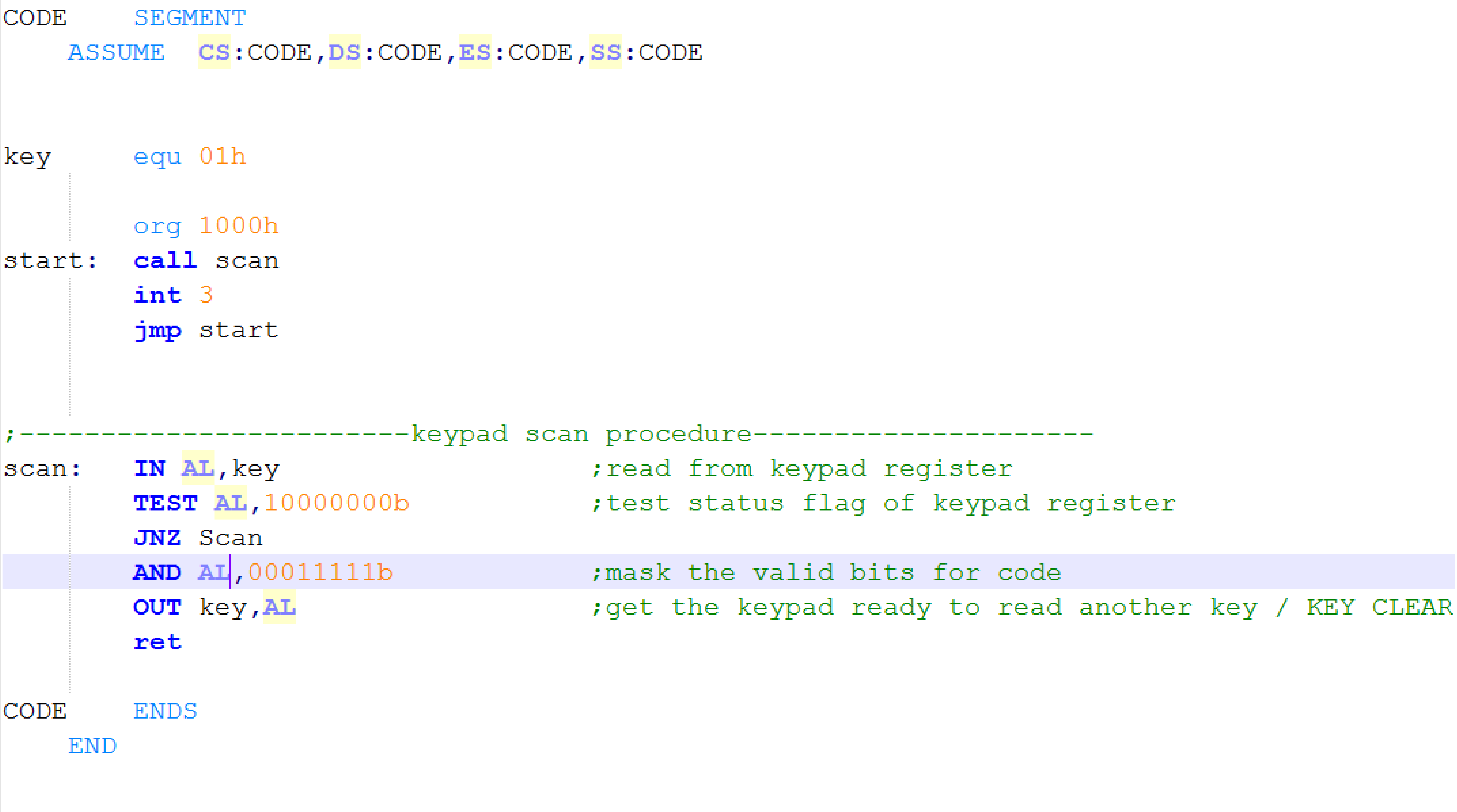
※ Key Input Flowchart



- Lets run “Keyboard.asm”

key	0	1	2	3	4	5	6	7
code	00h	01h	02h	03h	04h	05h	06h	07h
key	8	9	A	B	C	D	E	F
code	08h	09h	0Ah	0Bh	0Ch	0Dh	0Eh	0Fh
key	:	STP	GO	-	+	REG	DA	AD
code	10h	11h	12h	13h	14h	15h	16h	17h

Program to scan keyboard input



```
CODE    SEGMENT
    ASSUME  CS:CODE,DS:CODE,ES:CODE,SS:CODE


key     equ 01h

        org 1000h
start:  call scan
        int 3
        jmp start


;------------------------keypad scan procedure---------------------
scan:   IN AL,key                           ;read from keypad register
        TEST AL,10000000b                   ;test status flag of keypad register
        JNZ Scan
        AND AL,00011111b                    ;mask the valid bits for code
        OUT key,AL                          ;get the keypad ready to read another key / KEY CLEAR
        ret

CODE    ENDS
    END
```

LCD initialization procedure

1. Check if the module is **ready** or **not** (check status register) (call **busy**)
2. Set the **function**: (**00110000** = **30h**) for 8-bits mode, **one line** & **5x7** dots.

```
mov al , 30h
out instr , al
```

3. Check if the module is **ready** or **not** (Call **busy**)
4. Turn the **display** and **cursor** ON, and **set cursor** to **blink** (**00001111=0Fh**)

```
mov al , 0Fh
out instr , al
```

5. Check if the module is **ready** or **not** (Call **busy**)

6. Set **entry mode**: (**00000110 = 06h**) cursor is to be moved to **right**

```
mov al , 06h
out instr , al
```

7. Check if the module is **ready** or **not** (Call **busy**)

8. Return **cursor** to **home**: (**00000010 = 02h**)

```
mov al , 02h
out instr , al
```

9. Check if the module is **ready** or **not** (Call **busy**)

10. **Clear** the **display**: (**00000001 = 01h**)

```
mov al , 01h
out instr , al
```

Now Lets run **display.asm**

NOP instruction

- **NOP** (**no operation**) instruction **does nothing**, but it may be **used** inside a **timing loop**.
- One purpose for **NOP**, it to **introduce time delays**. For example you want to program a **microprocessor which** has to output to some **LEDs** with a 1s delay. This **delay** can be implemented with **NOP** (and branches).
- **Instruction** Format:

```
NOP
```

- **Example:**

```
TIMER: MOV CX,0FFFFH
  TIMER1: NOP
          NOP
          NOP
  LOOP TIMER1
RET
```

THANKS