Assembly Programming and Interfacing

(Lecture notes for Experiments)

Author: Yin LU

Affiliation: The Computer Science School of

Northwestern Polytechnical University

Date: 2021-4-3

**Catalogue**

[Chapter 1 Knowing the Applications 3](#_Toc29505)

[1. Emu8086, the Intel 8086 emulator 3](#_Toc17235)

[1.1 How to install the Emu8086 program 4](#_Toc8519)

[1.2 How to program with Emu8086 4](#_Toc31009)

[1.3 How to debug with Emu8086 6](#_Toc18406)

[1.4 Peripheral devices provides with Emu8086 7](#_Toc5708)

[2. VMWare and Windows XP virtual machine 8](#_Toc490)

[2.1 How to install the VMWare application 8](#_Toc26062)

[2.2 How to mount the Windows XP virtual machine image 12](#_Toc1114)

[2.3 How to import and export files between vm and its host 14](#_Toc20154)

[3. Define code generation tools for Proteus 18](#_Toc12158)

[3.1 Install MASM32 SDK 18](#_Toc30644)

[3.2 Define code generation tool in Proteus 21](#_Toc8447)

[4. EV-Capture (EV-录屏) 23](#_Toc6924)

[4.1 How to Install EV-Capture 23](#_Toc6155)

[4.2 How to use EV-Capture 24](#_Toc14544)

[5. Prepare your experiment report 25](#_Toc17320)

[Chapter 2 Experiment 1 Brunch and Loop structure 27](#_Toc32340)

[1. Assignments 27](#_Toc5281)

[2. Experiment Preparation 28](#_Toc14074)

[3. Experiment Circuit Scheme 28](#_Toc14242)

[4. Experiment Process 28](#_Toc30787)

[Chapter 3 Experiment 2 Simple IO and Lantern Control 30](#_Toc16774)

[1. Assignments 30](#_Toc2933)

[2. Experiment Preparation 33](#_Toc9969)

[3. Experiment Circuit Scheme 33](#_Toc26965)

[4. Experiment Process 33](#_Toc8596)

[Chapter 4 Experiment 3 Display devices Interfacing 38](#_Toc197)

[1. Assignments 38](#_Toc30939)

[2. Experiment Preparation 40](#_Toc1865)

[3. Experiment Circuit Scheme 40](#_Toc4316)

[4. Experiment Process 40](#_Toc19136)

[Chapter 5 Experiment 4 41](#_Toc31147)

[1. Assignments 41](#_Toc30164)

[2. Experiment Preparation 42](#_Toc19087)

[3. Experiment Circuit Scheme 42](#_Toc1256)

[4. Experiment Process 42](#_Toc12207)

## Chapter 1 Knowing the Applications

## Emu8086, the Intel 8086 emulator

Emu8086 is Intel x86 processor emulator designed for x86 assembly language programming and debugging. It integrates the source code editor, the x86 masm assembler, and the the x86 debugger in a GUI application, also a x86 virtual processor on the back. With the help of the pure software virtual processor, emu8086 can run a program designed for the Intel 8086/8088 processor. And moreover, if the executable program is generated by its own assembler, emu8086 will support single step forward debugging.

The main window of Emu8086 is its source code editor by default (see figure 1.1). When you press the [emulate] button on the tool bar, the emulation window together with the source code view window appear (figure 1.2). The x86 emulator provides machine code view, register view, and the buttons on the bottom toolbar will open other addition view windows including command line screen, memory view, variable view, stack view and flag register view. All these view windows will help the programmer to debug their assembly language program.

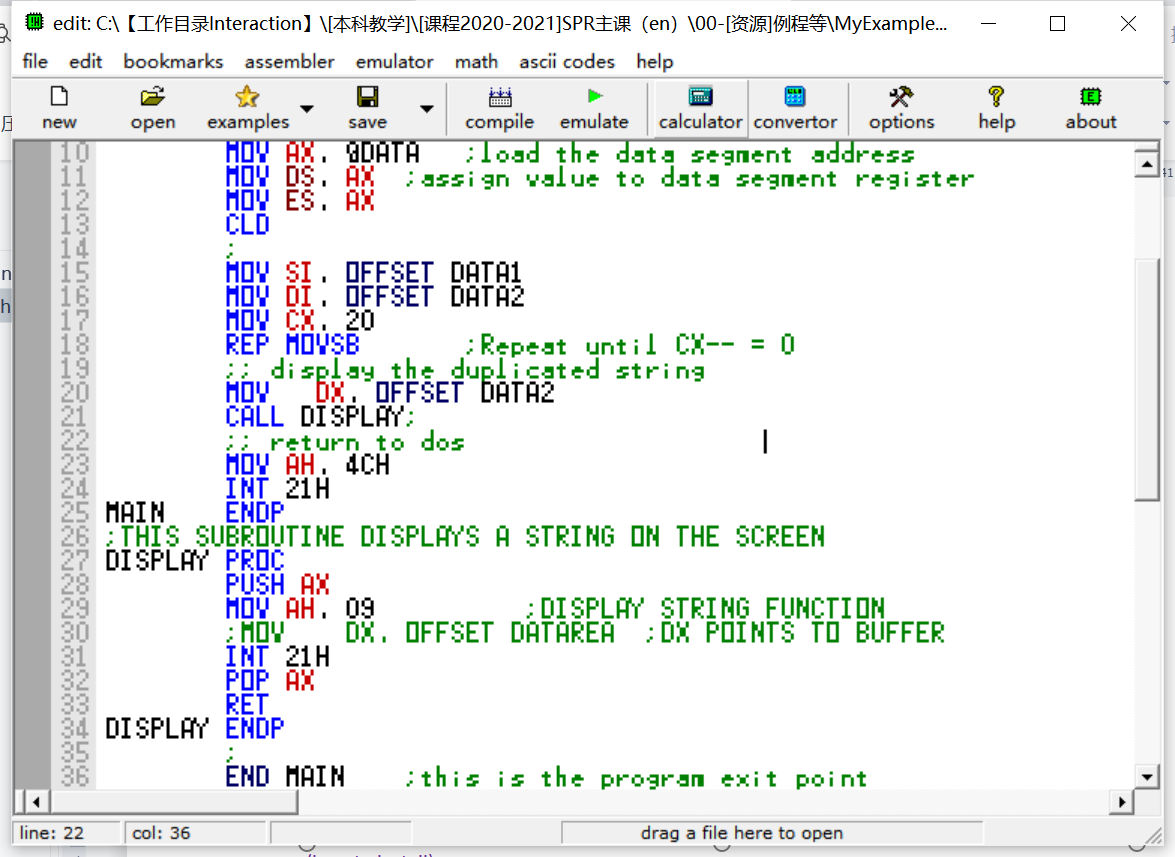


Figure1.1 The main window of Emu8086

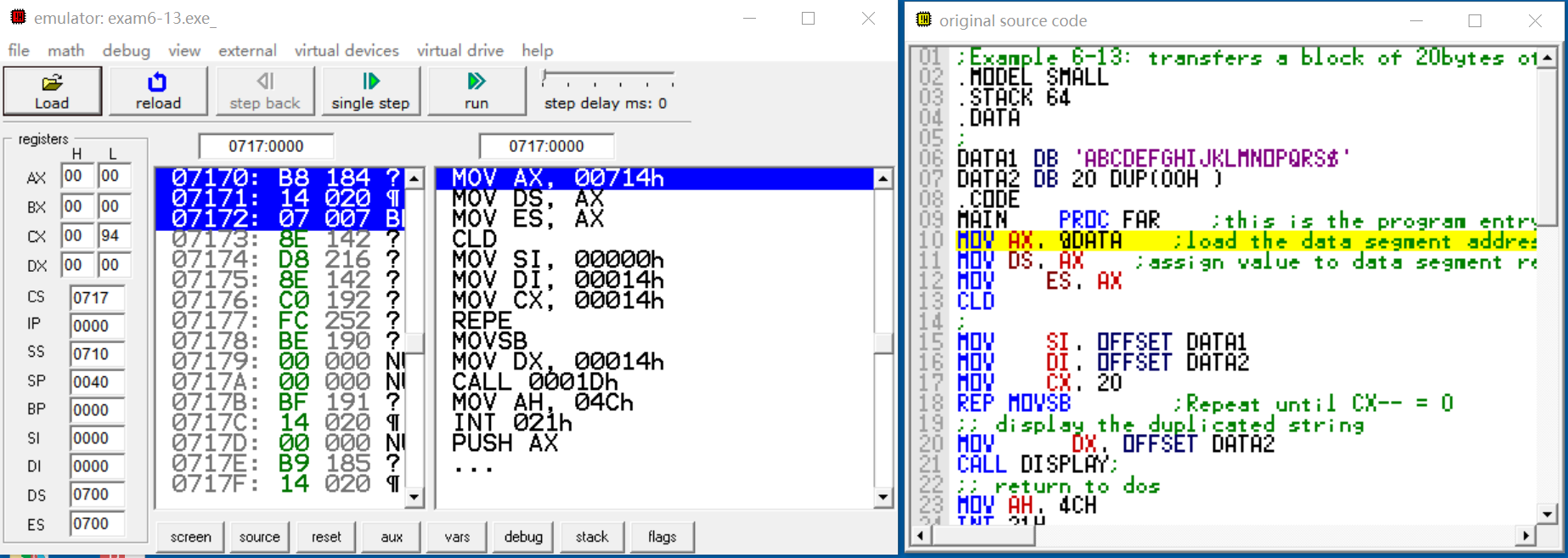


Figure1.2 The emulation and source code view windows

#### 1.1 How to install the Emu8086 program

Most Emu8086 installation program are provided as a self-extraction and auto-start package file, like this . Double click the package file, and start install process.

Click the [next] button to take the default settings, and the Emu8086 program will be installed into your C: disk like this .

After the installation, a short cut group will be created in your starting menu (see Figure1.3). You can click the [emu8086] menu to start the program.

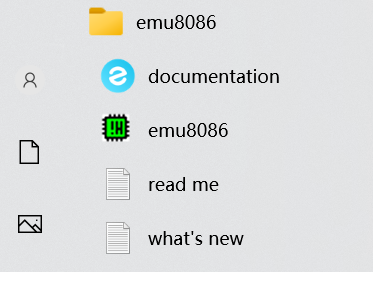


Figure1.3 emu8086 short cut group

#### 1.2 How to program with Emu8086

Start the emu8086 program, then click the [new]  botton to start a project (figure 1.4). Choose [BOOT template \ empty workspace]in the next popup dialog, then press [ok] button to open a new workspace for a boot-able program. The source code editor window appears, and you can type in your program source code (figure 1.5 shows a program to display a down-counting number with the a virtual LED displayer).

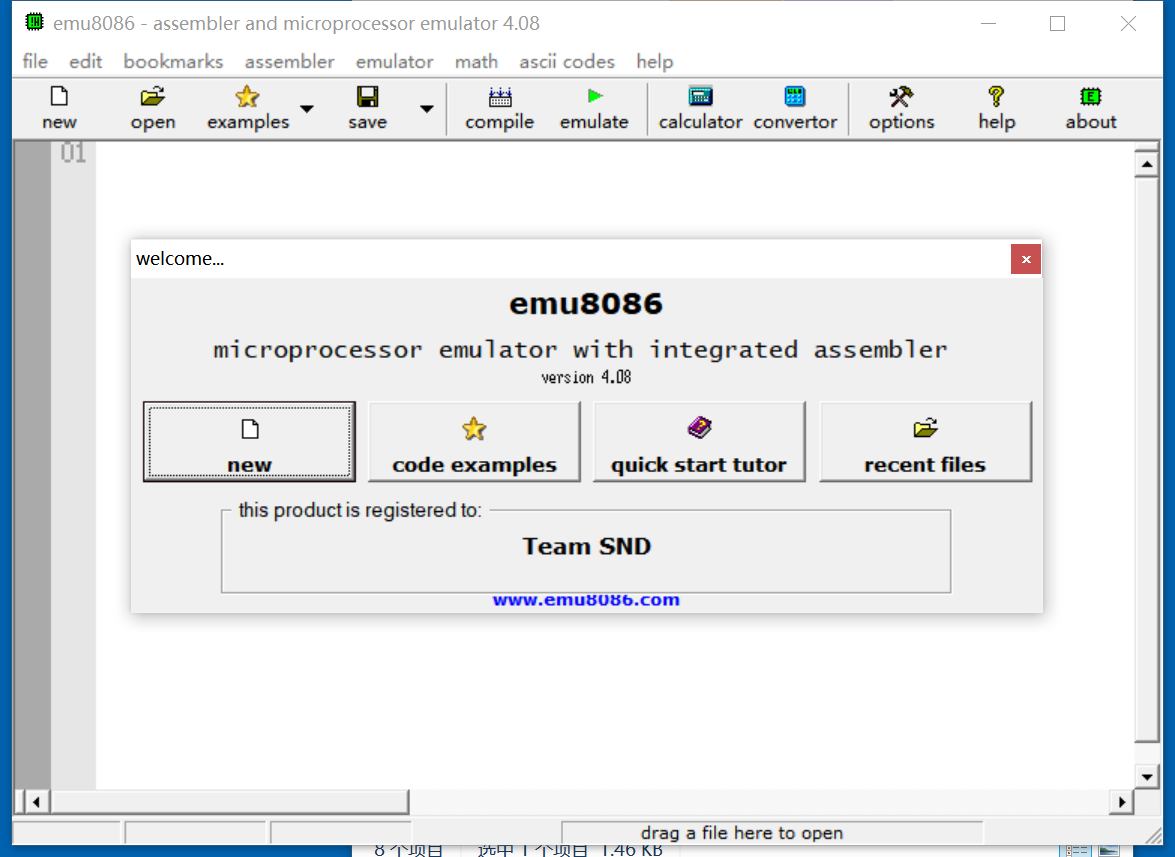


Figure 1.4 The emu8086 starting window

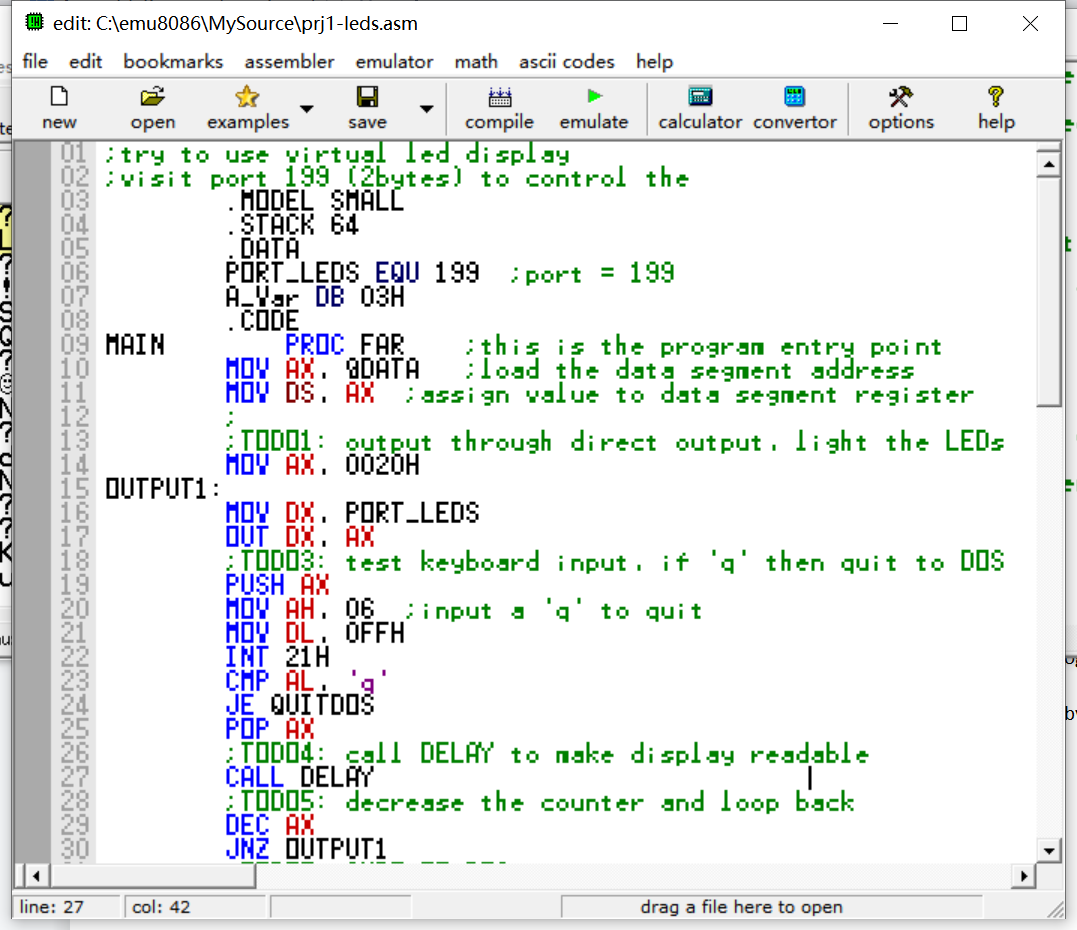


Figure1.5 The source code editor window

Remember to press the [save] button to save your source code in the hard disk, otherwise you will lose it when quit the program.

#### 1.3 How to debug with Emu8086

Click the [emulate]  button on the toolbar, to start the debug windows, including the emulation and source code view window. Then debug window below (figure 1.6 )

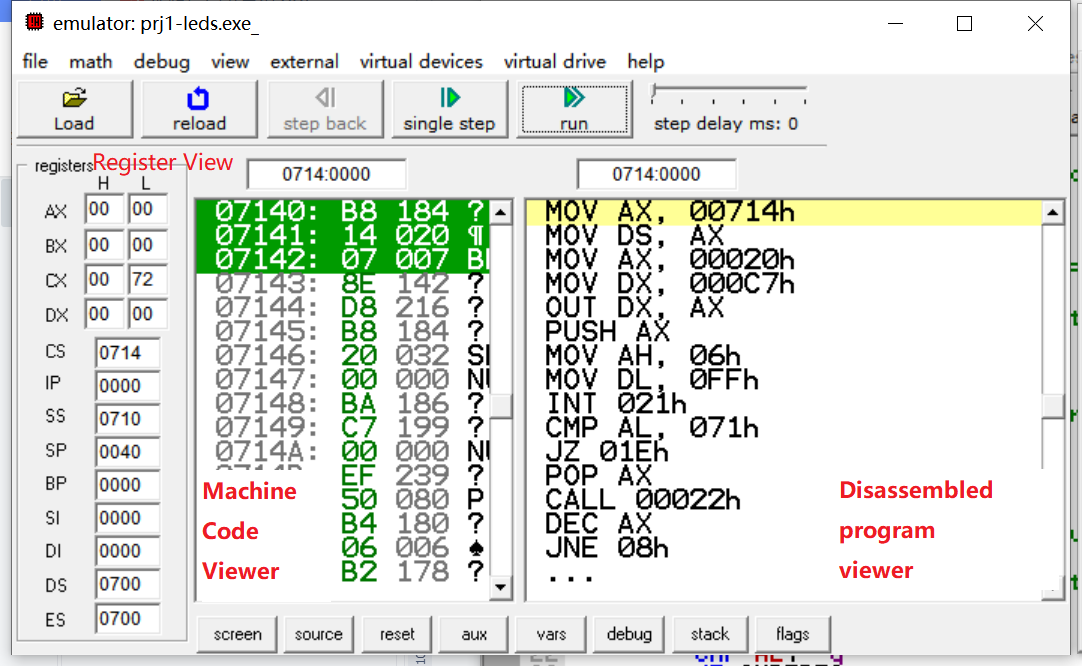
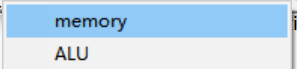


Figure1.6 The debug window

Use the [single step] button to run your program single direction forward. Each time the debugger halts, you can check the effect your program do to registers in the register view.

If you want to check the variable, press the [aux]  button, and click [memory]  menu item in the popup menu. A memory view dialog will appear (figure 1.7). Beware to change the display beginning address to the start of your program, which is in the first line of your program in the disassembled program view. And in this example, it is 00714H (0714:0000 logical address).

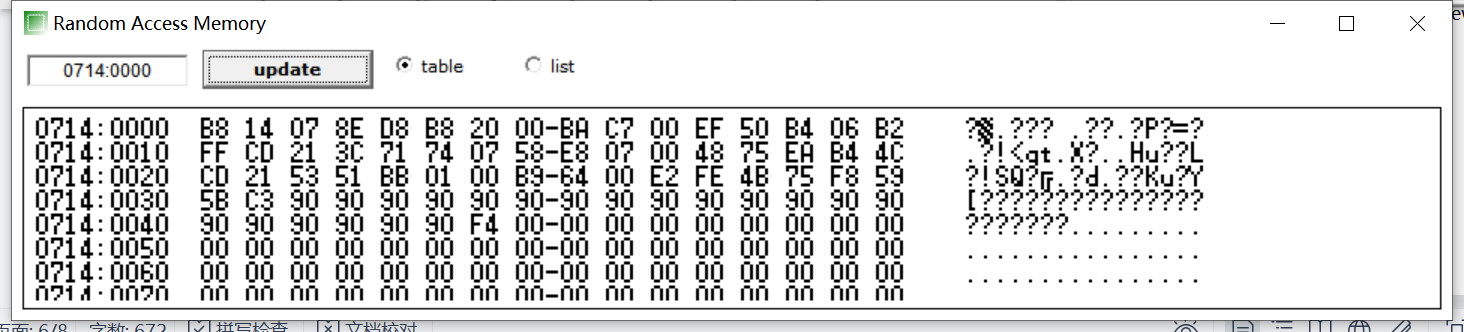


Figure1.7 the memory view dialog

An other way to check the output of your program, is to open the variable view dialog (figure 1.8 (A)) by touching the [vars]  button. And the flag register can be verified by open the flag register view(figure 1.8 (B)) by pressing the [flags]  button.

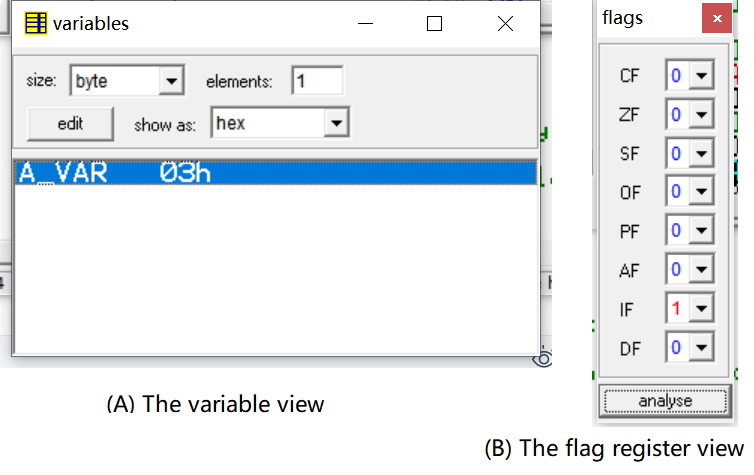
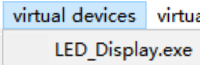
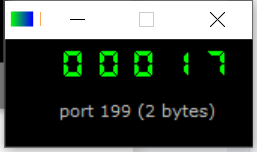


Figure1.8 The variable view and the flag register view

#### 1.4 Peripheral devices provides with Emu8086

The emu8086 also provides some virtual devices that can be manipulated by output command words to interfacing ports. The example in figure 1.5 will use the LED displayer device. You can open the device by using the [virtual devices]  menu in the debug window. The menu item [LED-display.exe] will open the LED displayer window below. .

Examples to show how to use these virtual devices can be found in example sub-directory in the emu8086 installed directory. Or you can open an example by using the [file\examples] menu group, as shown in figure 1.9.

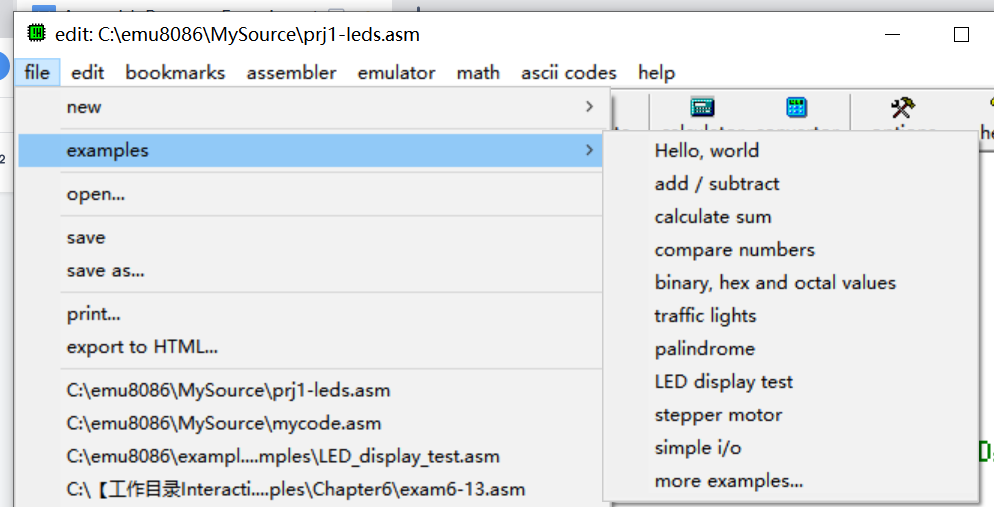


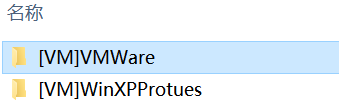
Figure1.9 Examples provided by Emu8086

## VMWare and Windows XP virtual machine

VMWare (Virtual Machine ware) is a virtual machine application software. It can emulate a virtual operation system different from your current one in your desktop computer. For example, a Windows 10 system in a Linux system, or a Windows XP over a Windows 10 system. We use VMWare to create a virtual Windows XP computer to install and run the Emu8086 and Protues software. Although both of them runs on Windows 10 platform, we insist that you to install VMWare and use the virtual machine. For the reason that we will release a all-in-one virtual machine image with all the necessary software installed. Once you mount the image in your VMWare application, your ready to write and debug your experiment program.

#### 2.1 How to install the VMWare application

The installer of VMWare is provided as a self extraction package like this .

Create a sub directory called “[VM]VMWare” in the root directory of one of you hard-disk partition, for example “D:\[VM]VMware” . Then double click the .exe file to open it and start to install.

The Installation guide window appears, see figure1.10. Click [next]  button to continue. In the next window, check the  to accept the licence agreement. Then, .

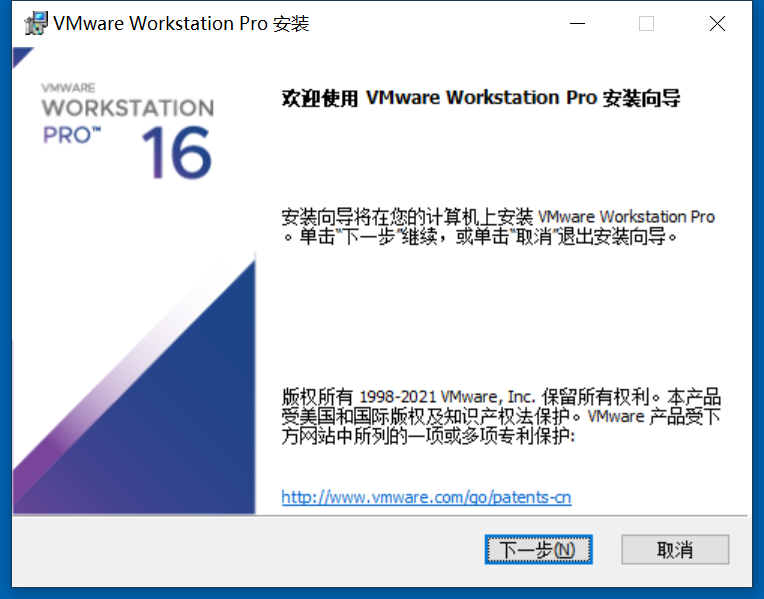


Figure1.10 VMWare installer guide window

In the third window, click  to choose your installation directory. We suggest that you choose the subdirectory you created in section 2.1, for example, see figure 1.11. And then, click [next] .

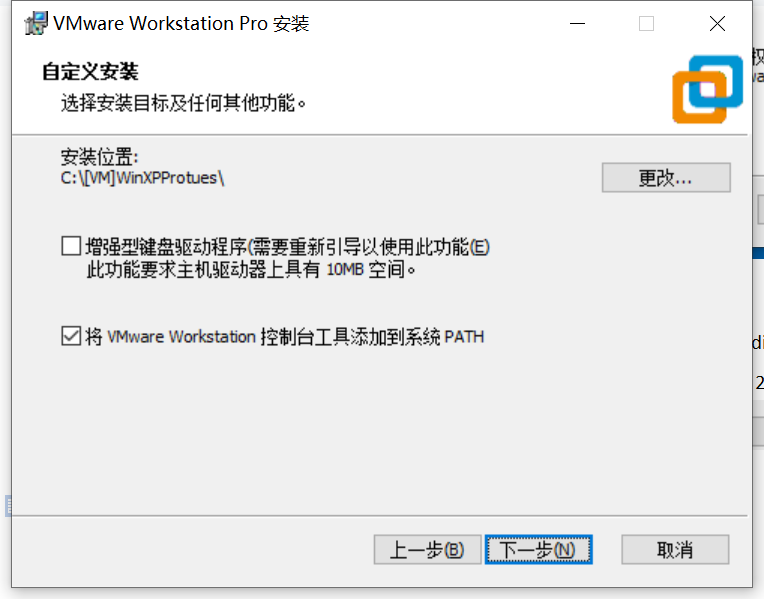


Figure1.11 Change installation directory

In the fourth and fifth window, click . And in the sixth window, click [install]  to start installation process (see figure1.12)

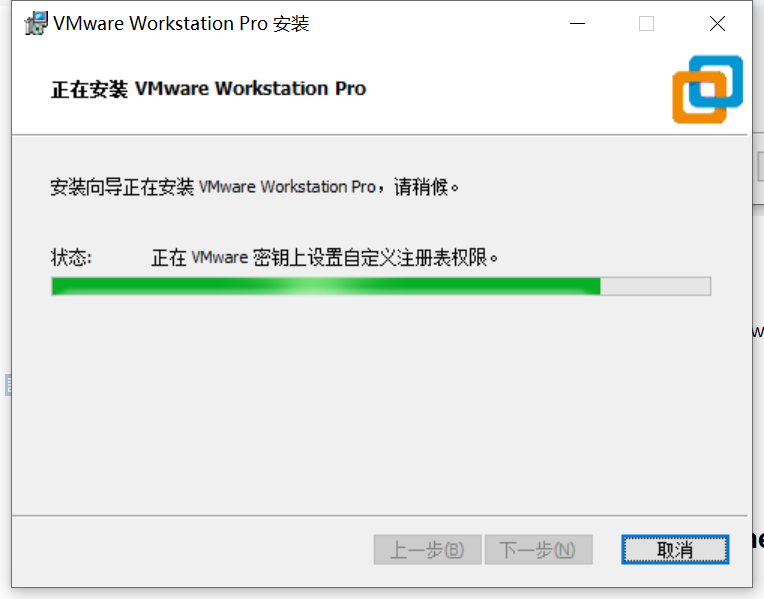


Figure1.12 VMWare installing

When the file copying process finished, the eighth window appears to hint you to provide the authorization key (figure 1.13). Click [Authorization key]  button to open the input window (figure1.14). Input a authorization key, and press the [continue]  button. In the next window, click [over]  button.

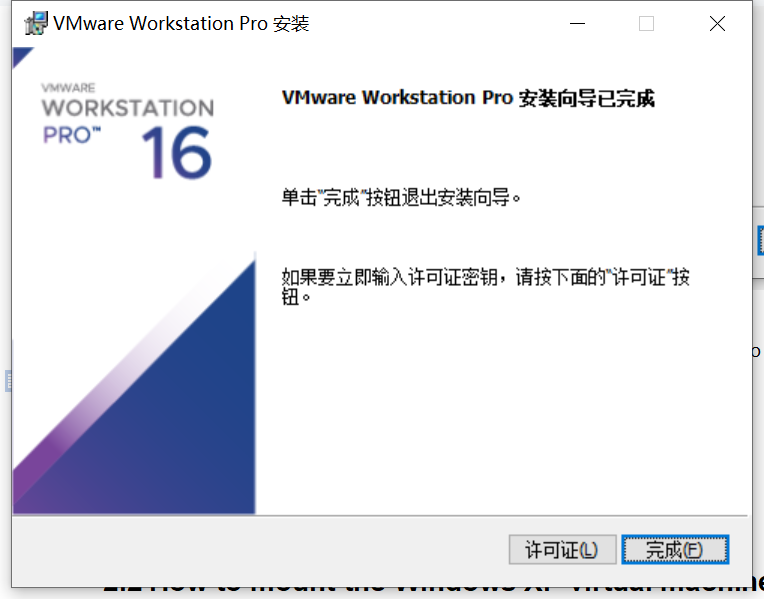


Figure1.13 Installation Hint window



Figure1.14 input authorization key

Now, the VMWare application has already installed. Follow the steps in section 2.2 to mount the all-in-one vm image.

#### 2.2 How to mount the Windows XP virtual machine image

Start the VMWare application from your starting menu. The vmware home page appears, see figure 1.15. Click the [open a virtual machine] button to mount the vm image.

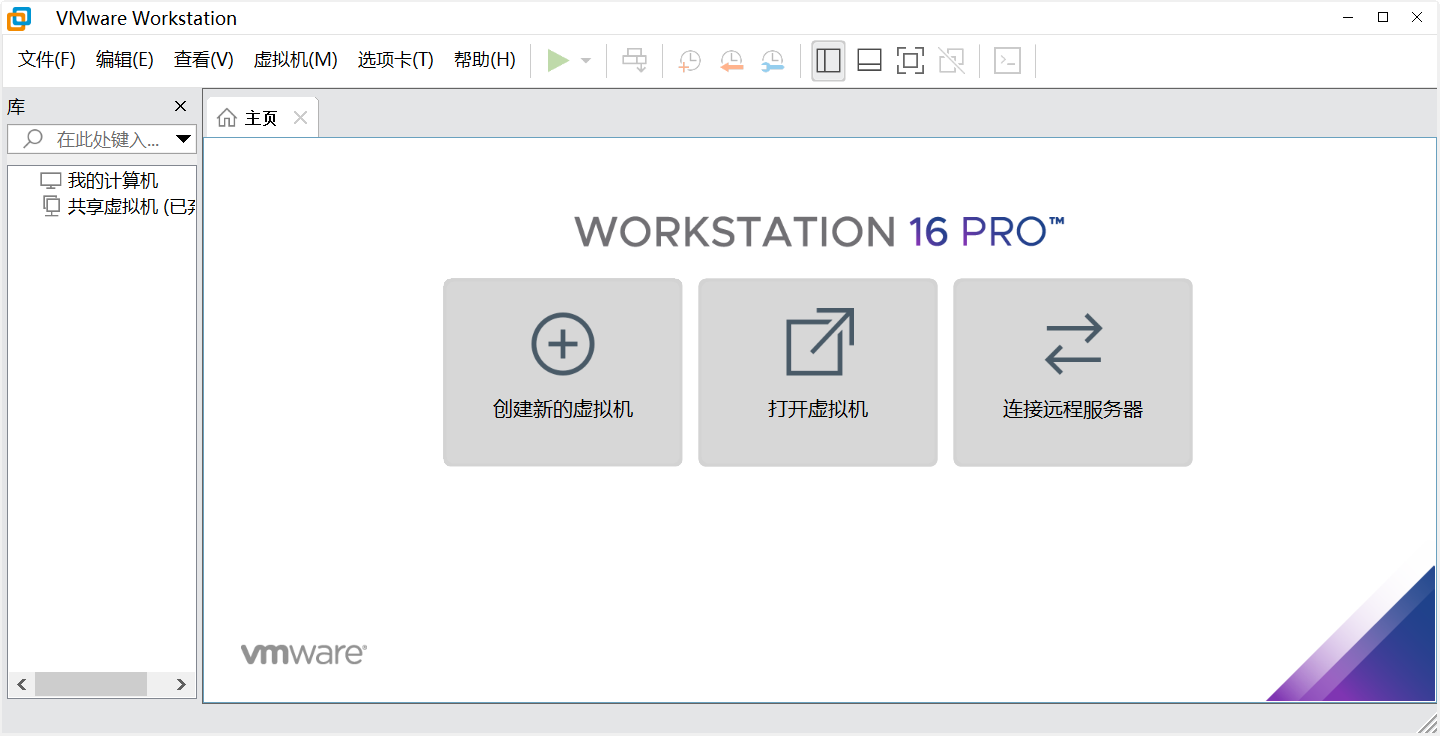


Figure1.15 VMWare home page

Suppose that you have unpacked the image we provide you in a directory called “C:\[VM]WinXPProteus”, and in the windows open file dialog, open this directory, and choose the .vmx file as shown in figure 1.16.

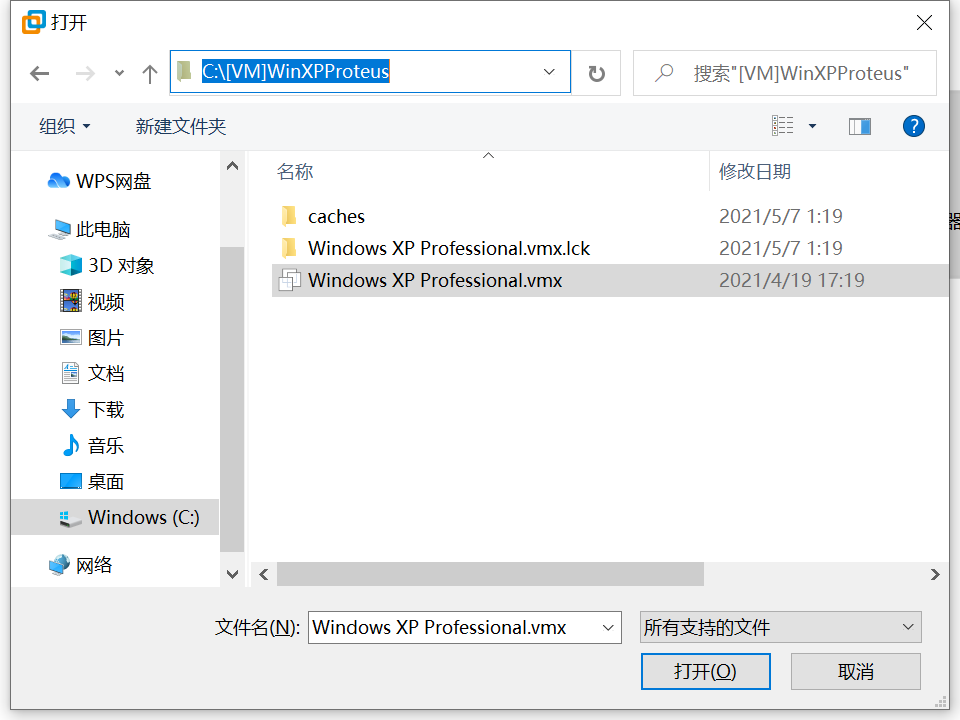


Figure1.16 Choose the vm image file

If everything is right, the virtual machine image will be mounted, figure 1.17. Click [start this virtual machine] menu item to boot up this virtual machine.



Figure1.17 WinXP vm image mounted

Maybe your vmware application will notice you that this vm has been moved or duplicated, just click [I have already duplicated the vm image]  button as directed. The Windows XP virtual machine will boot up. And the Windows XP desktop appears (see figure 1.19).

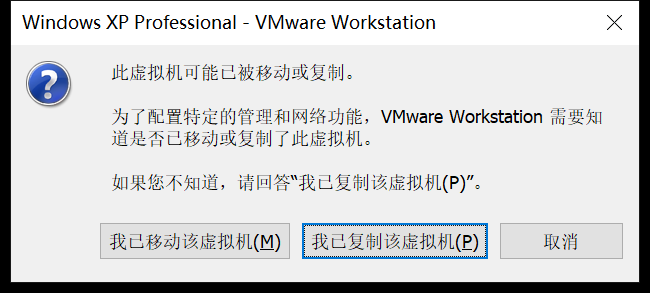


Figure1.18 Notification dialog

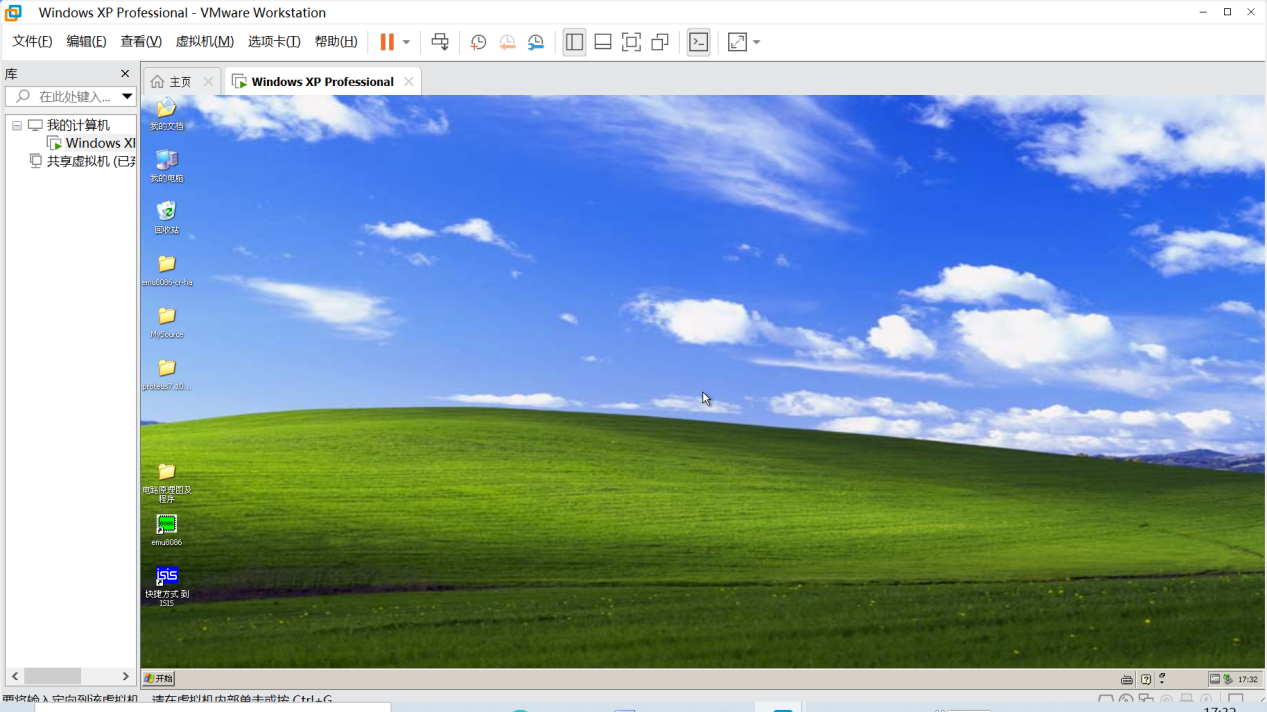


Figure1.19 Window XP virtual machine

#### 2.3 How to import and export files between vm and its host

The virtual machine and the host can not access files between each other. Files have to be import to the vm or export to the host before open. In this section, we will setup a network file sharing directory to do import and export between the vm and the host.

1. Start vmware and boot up the WinXPProteus virtual machine, by choosing [boot up this virtual machine] menu item in the vm manager window;



Figure1.20 boot up the virtual machine

1. Then, in the VMWare application menu choose [virtual machine\setup...] as shown in figure 1.21;



Figure1.21 setup the virtual machine

1. In the vm setup window popped out, choose page [options], then [Shared Folder], and [always enable] in the right side window. See figure 1.22.

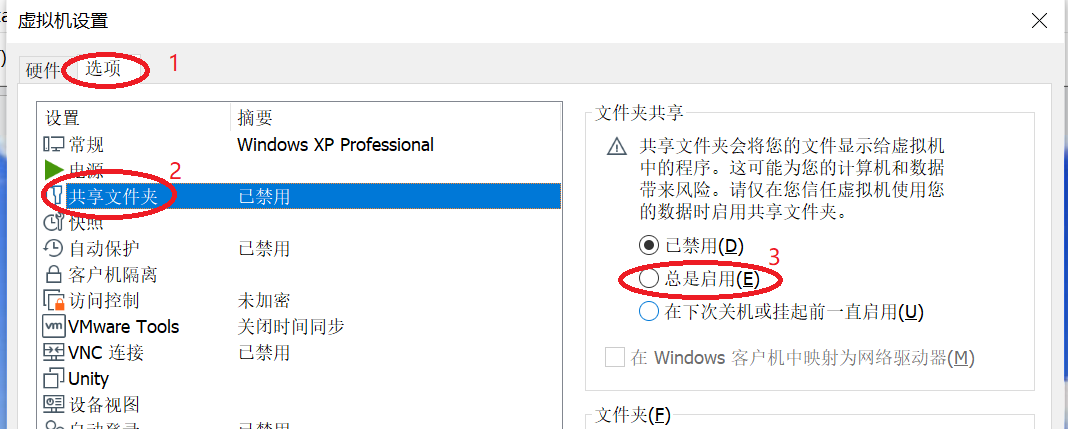


Figure 1.22 vm setup window -- enable file folder sharing

1. Then, in the file list below, insert a record of the directory path, which will be used to preserve the transmitted files. Press the [append...]button below the list to open the guide dialog, as shown in figure 1.23.

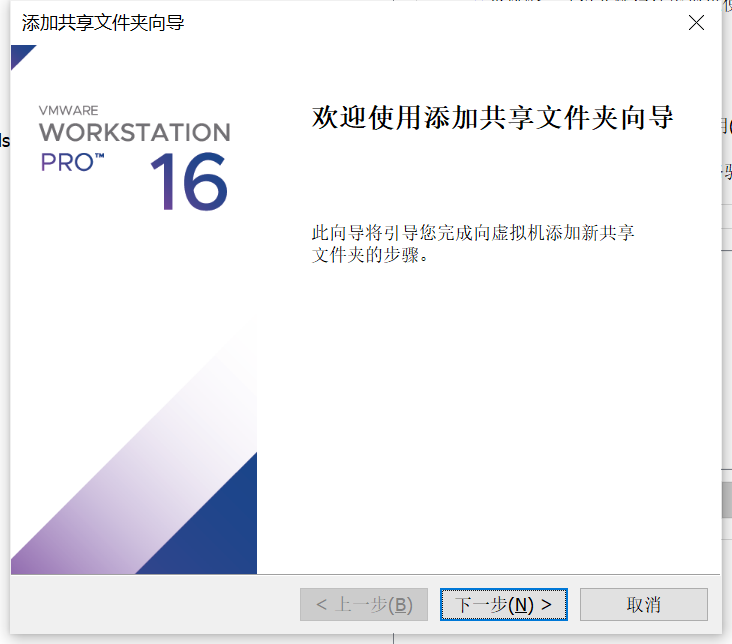


Figure1.23 shared folder append guide dialog

1. Press [next]  to continue. In the following window, press [view...] to input the folder path. And then, give it a name appears in the virtual machine operation system. In the example in figure 1.24, I choose folder “[VM]FileExchange” to be the shard folder. Press [confirm]  to confirm your selection, and return back to the second window. Give the file folder a name to be appeared in the virtual machine operating system, for example “[host]FileExchange”, as shown in figure1.25. Then, press [next] .

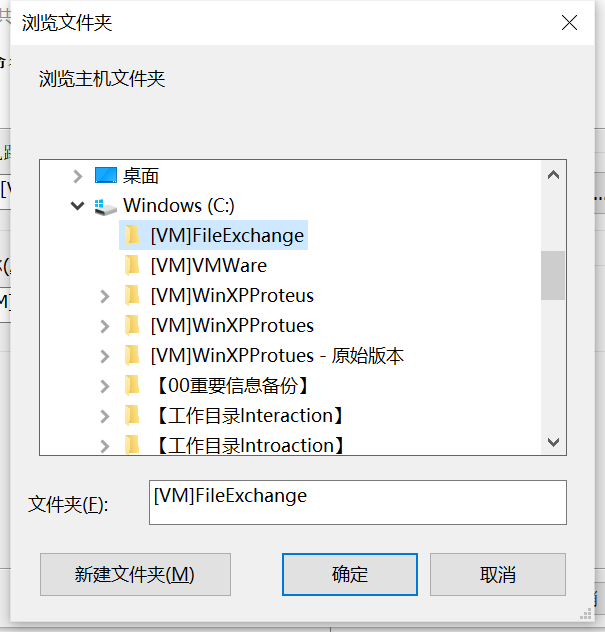


Figure1.24 select a file folder

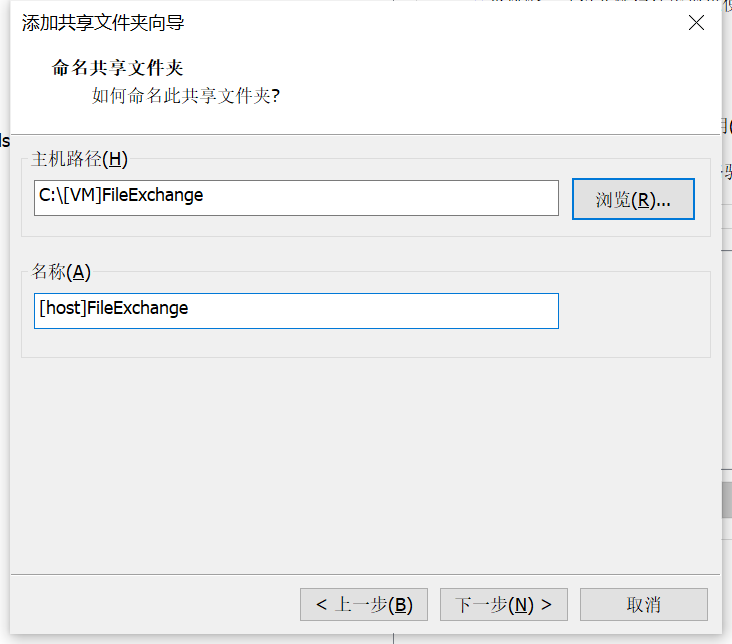


Figure1.25 name the file exchange folder

1. In the third window, make sure that you have checked the selection [start sharing] , then press [complete] . The shared folder append guide dialog will close, and in the vm setup window press [confirm]  to complete setup process.
2. Now, you can open the windows resource manager, as shown in figure 1.26. The example is a windowXP system which is the vm we provided. In file path [network neighborhood\VMWare shared folders\vmware-host\Shared Folders”, you can find the host file folder, which is named as “[host]FileExchange” . File copy into this folder, can be accessed by both host and virtual machine system.

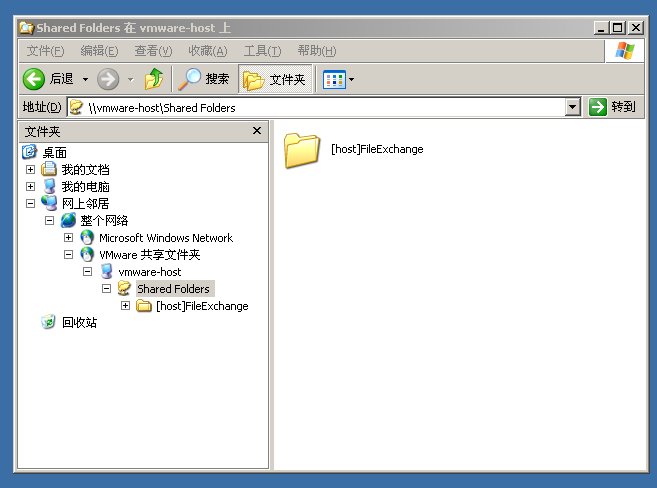


Figure1.26 vm windowsXP resource manager -- find the shared file folder

## Define code generation tools for Proteus

The Proteus does not provide a default assembler for Intel 8086 assembly language. Thus we have too install an compatible compiler and define it as the execution code generator tools. We make use of MASM32 SDK version 11 as our source code assembler in the experiment. In this section, we will discuss how to install MASM32 SDK and setup it as the compiler.

#### Install MASM32 SDK

The MASM32 SDK installer is a zip package . There are two files in it, a installer file, and a DOS batch file, which is used to invoke the ml.exe assembler and link16.exe linker, so that to compile the source code file into executable file. Please unzip the zip file into a folder and run the install.exe installer. The welcome window of MASM32 installer appears(figure1.27). Click [Install] button on the left side to start the installation process.

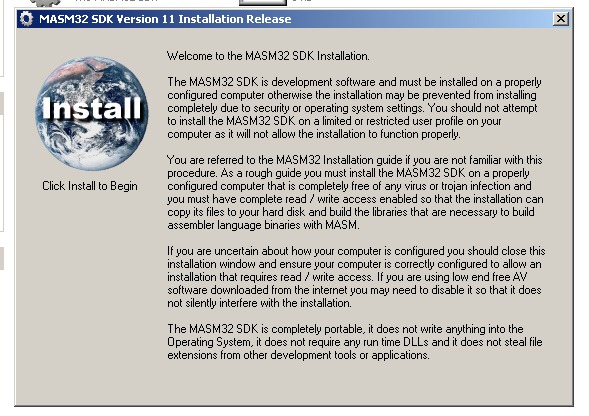


Figure1.27 Welcome window of MASM32 installer

In the popup dialog, choose install path for MASM32 SDK (figure1.28). We suggest that you choose root director in disc C (C:\), as suggested in the dialog. For that the batch file to do compile take C:\MASM32 as its default application path . If you change the installation path in figure1.28 dialog, you should have to change commands in the batch file. Press [OK] button to continue.

The MASM32 SDK installer will hints you that it will run a test of your computer system to determine whether it is suitable to install such a SDK. Press [confirm] to continue. And the following 4 notification dialog appears one by one (figure1.29). Press [Extract] to start extract files into the installation path.

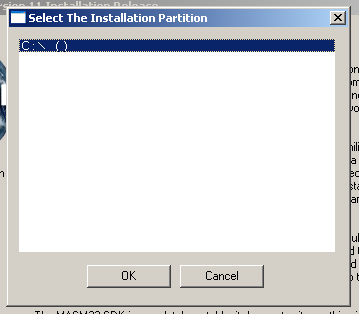


Figure1.28 Choose installation path dialog

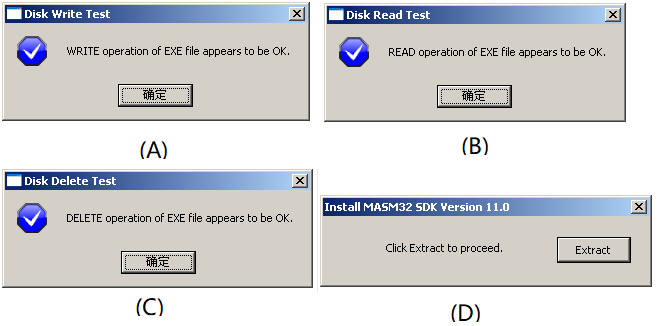


Figure1.29 three system diagnostic process conformation

When all the files are extracted into the installation path, the install will try to build necessary libraries by the compiler extracted (figure1.30), and a command line window will popup and show the processing of compiling (figure1.31). If there are no errors during the compiling process, two dialog will popup (figure1.32).

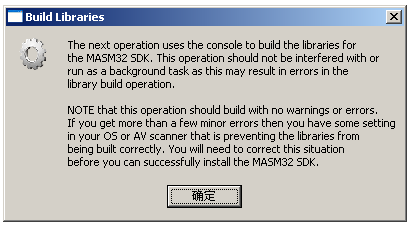


Figure1.30 Notification of library building

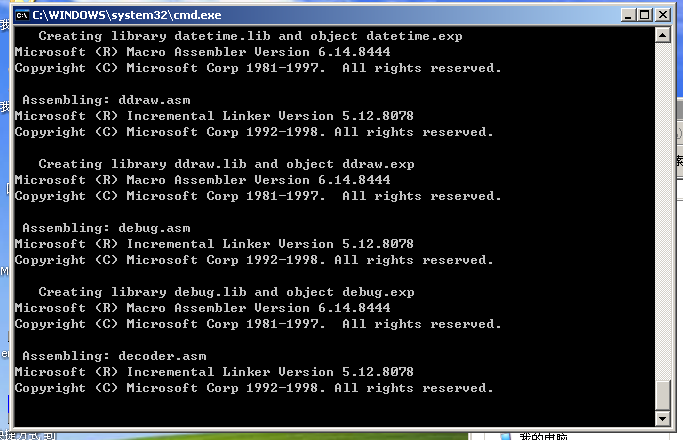
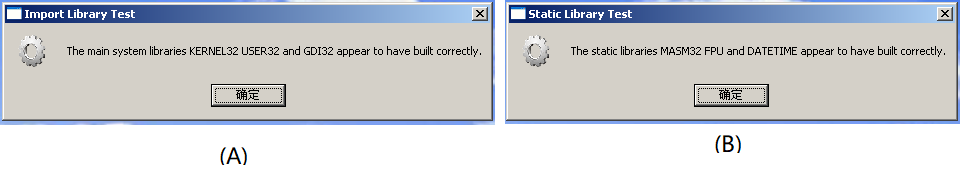


Figure1.31 Library building process



Figur1.32 notification of compiling succeeded

After the notification, a new dialog appears and asks whether to install icon script, click [Yes] button to complete the installation of MASM32 SDK.

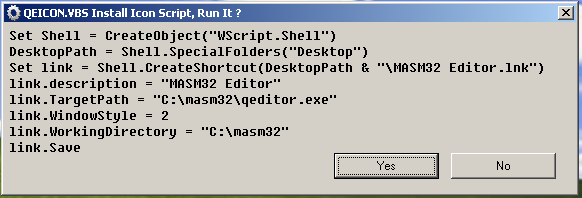


Figure1.33 Install icon scrip

For the last step, copy the masm32.bat batch file into the [installation path]\bin director, as done in figure1.34 below.

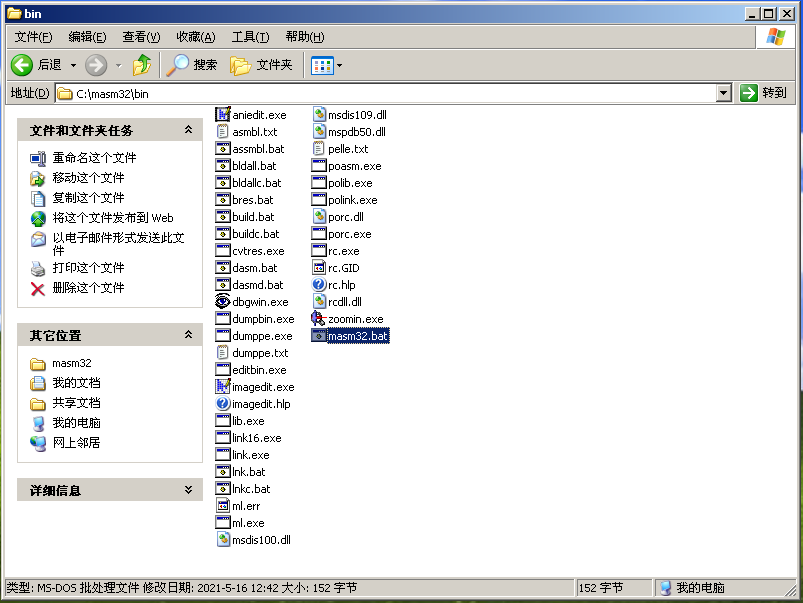
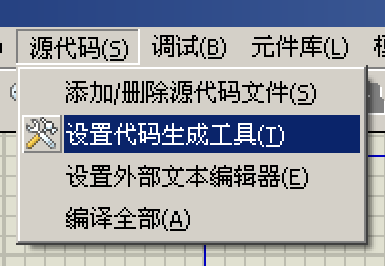


Figure1.34 copy masm32.bat into the bin directory

#### Define code generation tool in Proteus

Start the Windows XP virtual machine and start Proteus application by double click the shortcut icon in the desktop. In the Proteus main window, choose menu item [source code\Define code generation tools (T)]  to open a dialog of “append / remove code generation tool” . Figure1.35 shows the dialog with a default tool, which is called “ASEM51”.

Do not change anything in the default dialog, press the [new]  button at the bottom to append MASM32 SDK as a new tool. A windows file selection dialog appears. Move to the [MASM32 SDK installation directory\bin] directory, into which is the directory we copy MASM32.bat, and select the MASM32.bat batch file, then press [Open] button, as shown in Figure1.36.

Then in text input box in the middle square, type “ASM” and “OBJ”, as shown in figure1.37, then press the [confirm]  button.

By now, the MASM32 SDK can be selected as the compiler in our experiments. Latter, in the section of experiment process, we will learn how to setup source code program for a hardware circuit schema, and how to use MASM32 SDK to compile the program, and then, how to debug the program in source code level.



Figure1.35 append / remove code generation tool dialog

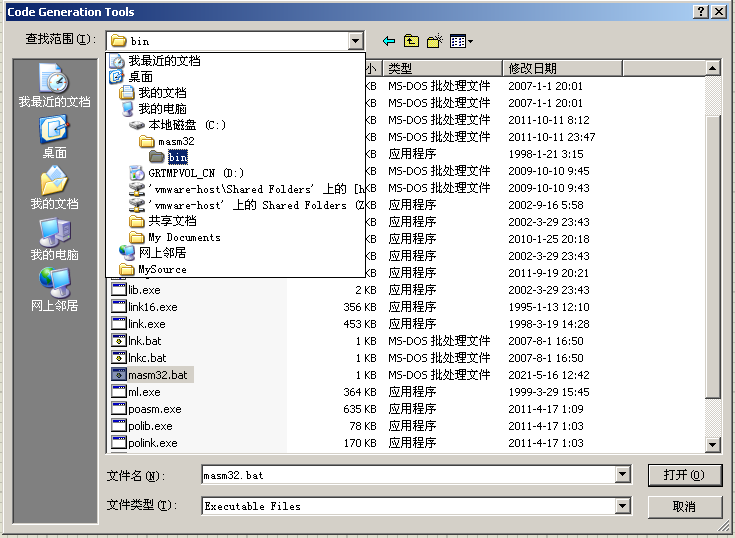


Figure1.36 file selection dialog, choose MASM32.bat

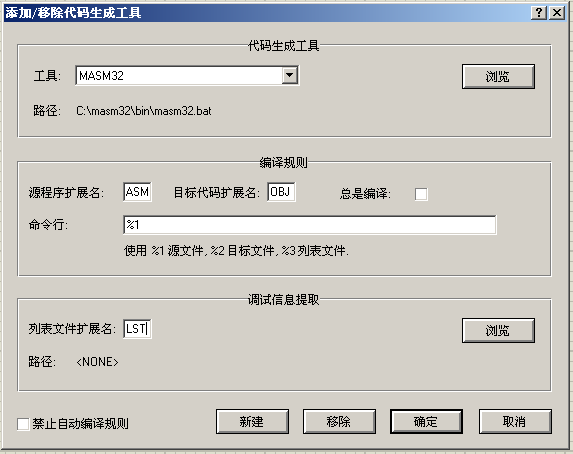


Figure1.37 setup MASM32 SDK as a new code generation tool

## EV-Capture (EV-录屏)

EV-Capture is small utilization application. It can make a video record of all the thing happen on your desktop screen. We will use it to make the video of experiment process. And the process will be handed in as a part of the experiment report.

You can use other application software to make the record if you are quite familiar with it. However, it must be mpeg4 compatible, which means, not a .mov file.

#### How to Install EV-Capture

The EV-Capture installer is provided as a zip file . Unzip it, and you will find the installer . Double click the installer to start installation, figure1.38.

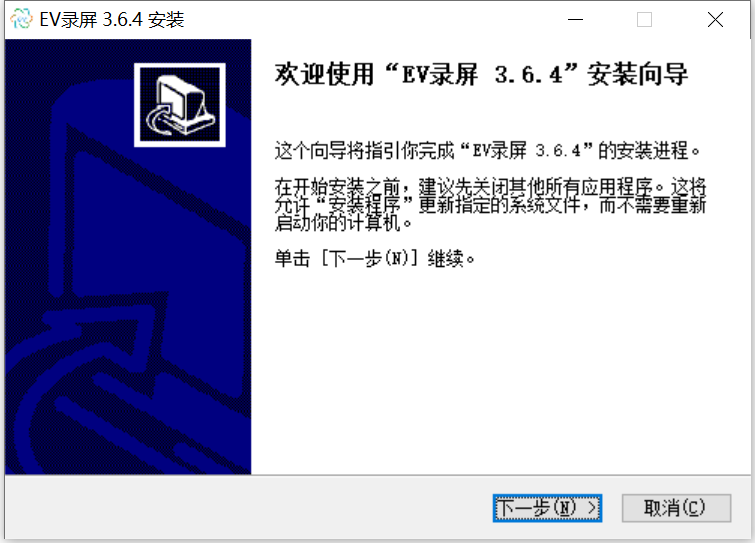


Figure1.38 EV-Capture install guide

Click the [next]  button to continue. In the next window, choose [I accept]. And in the third window (figure1.39), it will let you choose the install target directory. We suggest that take the default settings. So , click [install]  button to start installation.

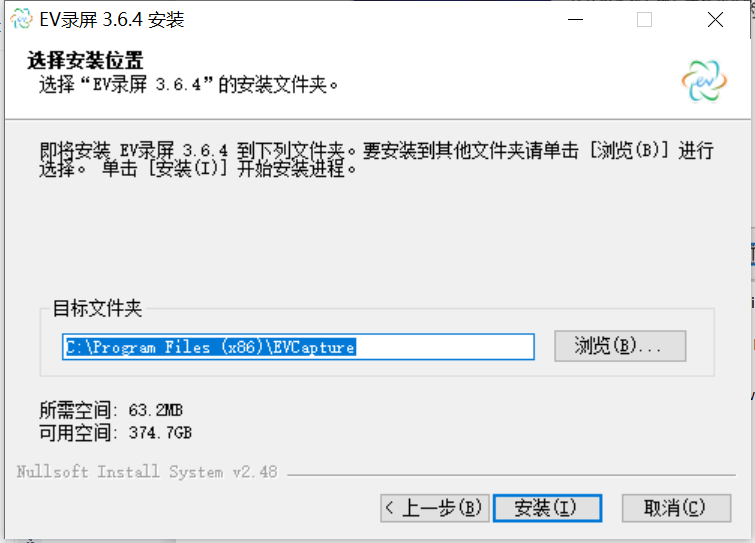


Figure1.39Choose install target directory

#### How to use EV-Capture

Open the EV-Capture application, a update dialog may appears. Choose [cancel] button to cancel update, for we do not need to update.

The main window of EV-Capture looks like figure 1.40.



Figure1.40 The main window of EV-Capture

Now click the [start record]  button in the bottom-left corner. After 3 down counting, the main window minimized, and the record begin. From now on, everything happens on the screen will be captured and recorded. And now you can carry on the experiment.

When you complete the experiment, double click the EV-Capture icon in the hot pan, so that to recall the main window of EV-Capture. You can find how long have you made the video record: . Now, press the [stop]  button to terminate the capturing process. And the main window will shift to the file list window in figure 1.41. The newly made record file is been selected and wait for you to give it a more meaningful name. Press the [return] key on your keyboard after you type in the new file name.



Figure1.41W Record file list window

If you want to hand in the video, use the [open file directory] button. It will open your window resource manager, and shift to the sub-directory where the video files being saved. Choose the file you want, and upload it.

## Prepare your experiment report

Experiment report is the summary of your experiment work and process. For we will conduct our experiment class online, the report will be different from the traditional type. It will be constructed by three files:

1. A .doc file of the experiment assignment problems. You can make a copy of the assignment description from this lecture notes.
2. The program source code file. And at the very beginning of the program file, there should be a comments to indicate who the file belongs to. Like the example below:

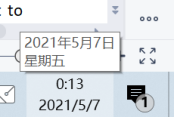
;=========================================================

;Description: this is the program of assignment 1 experiment 1

;Author: Yin LU, 2018xxxxxxxx

;Date: 2021-4-22

;=========================================================

1. A video of the experiment process. In the beginning of the video, there should be a shot to show the date of video recording, that is, the date of experiment completion. If you are making the video with the EV-Capture program, you can make the shot of date by showing system date and time: 

## Chapter 2 Experiment 1 Brunch and Loop structure

In Chapter 2, it is our first experiment. And in this experiment, we will practice to create and debug x86 assembly language program with the Emu8086 application software.

There are all together 3 assignments in this experiment. Please program to solve the problem, and make a video of how you debug the program, and the result of your program execution. In order to see the result, please open the VAR window.

## Assignments

1. Addition of multi-bytes BCD numbers. There are two large BCD numbers in the data segment, DATA1 and DATA2. Each number takes 5 bytes, and it is stored in little endian format (lower digit goes to lower address). Please adds DATA1 and DATA2 together, and save the result in the third variable DATA3. The Template of the program is provided below, and all the variables are predefined in the data segment.

Hint: A similar multi-bytes data addition program can be found in example 3-4 chapter 3. And BCD addition algorithm can be found in section3.4.

;========================================================

;Description: Program of Assignment 1 Experiment1

;Author:[name][student ID]

;Date:[Date]

;========================================================

.MODEL SMALL

.STACK 32

.DATA

DATA1 DB 10H,34H,56H,78H,00H

DATA2 DB 90H,88H,77H,66H,00H

DATA3 DB 00H,00H,00H,00H

.CODE

MAIN PROC FAR

; INITIALIZE DATA SEGMENT

MOV AX, @DATA

MOV DS, AX

;here is the program body

; RETURN TO DOS

MOV AX, 4C00H

INT 21H

MAIN ENDP

END MAIN

1. Sorting of 10 Defined Byte data. There are 10 Defined Byte data in the variable called DATAS, and the count of 10 in the variable DATANUM. Please sort the 10 data in DATAS in the order of from small to large. The Template of the program is provided below, and all the variables are predefined in the data segment.

;========================================================

;Description: Program of Assignment 2 Experiment1

;Author:[name][student ID]

;Date:[Date]

;========================================================

.MODEL SMALL

.STACK 32

.DATA

DATANUM DB 10

DATAS DB 21H,13H,4H,5H,7H, 6H,8H,20H,9H,11H

.CODE

MAIN PROC FAR

; INITIALIZE DATA SEGMENT

MOV AX, @DATA

MOV DS, AX

;here is the program body

; RETURN TO DOS

MOV AX, 4C00H

INT 21H

MAIN ENDP

END MAIN

## Experiment Preparation

All the things below should be prepared for experiment1:

1. The programs for assignment 1 and 2;
2. Emu8086 application. You can install the Emu8086 in your own computer system. Or you can use the Emu8086 in the virtual machine image we provided.

## Experiment Circuit Scheme

We don’t have hardware device involved in experiment 1.

## Experiment Process

1. Starting EMU8086, and use [NEW] to create a new empty workspace:

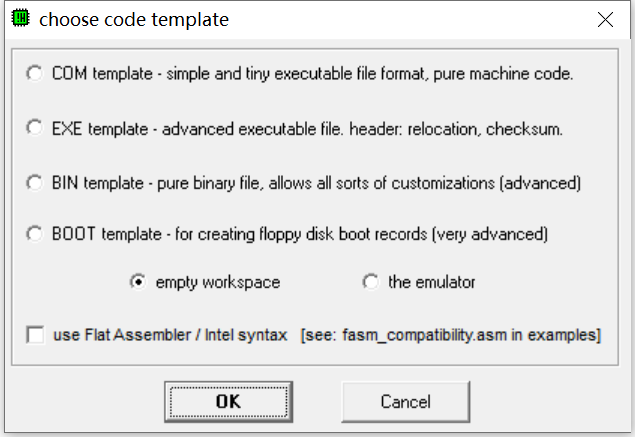
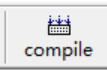


Figure2.1 create a assemble program project

1. Input your program, and save it as a .asm file;
2. Use  button to assemble your program. If there is errors, a hint dialog will appear. Fix these bugs.

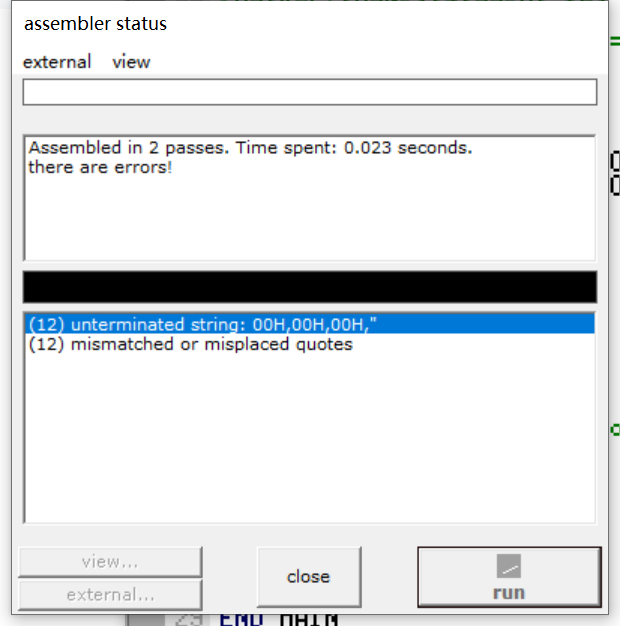
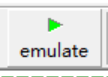


Figure2.2 try to assemble the source code

1. Use the  button to run and debug you program.
2. Use the function dialog of [aux/memory] or [vars] to see the result.

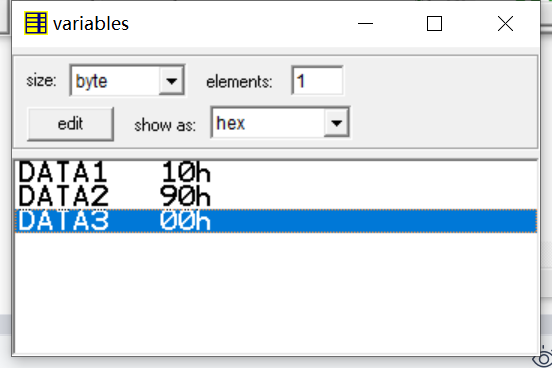


Figure2.3 check the variables

## Chapter 3 Experiment 2 Simple IO and Lantern Control

In Chapter 3 we will practicing how to manipulate peripheral devices, which means, try to use input and output instruction to access device ports.

We will start from a virtual device provided by the emu8086. Then move to the Proteus and try to run some program on the virtual hardware circuit.

## Assignments

1. Out put a data to a typical device port.

The emu8086 provides a virtual led display device, which is emulated by a program called “led\_display.exe”. The virtual device can display decimal number up to 5 digits, as shown in figur 3.1.

By output a word type data to port 199, which is a word sized IO port address, you can change the display to the number you output. Now write a program to display numbers from 0 to 65535 in a loop.

Each time you write a number to the port, remember to call a sub program called “delay”, so that to wait for the display to be stable. The template of the program is provided in the list below.

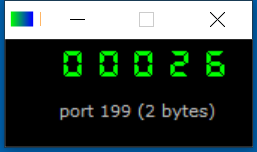


Figure3.1 The virtual led display device

List3.1 program template of assignment 1

;========================================================

;Description: Program of Assignment 1 Experiment1

;Author:[name][student ID]

;Date:[Date]

;========================================================

;This is the program for experiment2 assignment 1

;In this program, we try to display numbers with

; a virtual led display device provided by emu8086.

;The port to setup the display is 199

;========================================================

;start the virtual LED display device with the command below:

#start=led\_display.exe#

.MODEL SMALL

.STACK 64

.DATA

PORT\_LED EQU 199

.CODE

MAIN PROC FAR ;this is the program entry point

MOV AX, @DATA ;load the data segment address

MOV DS, AX ;assign value to data segment register

;TODO1: display 8888 to test the device

MOV AX, 8888

MOV DX, PORT\_LED

OUT DX, AX

CALL DELAY ;call delay sub procedure

;TODO2: start to display numbers

;(put your program to do the display of numbers here)

MOV AH, 4CH ;set up to

INT 21H ;return to DOS

MAIN ENDP

;===========================================================

DELAY PROC NEAR

PUSH BX;

PUSH CX;

MOV BX,0Ah

loop\_OUT: MOV CX, 03h

loop\_inner: LOOP loop\_inner

DEC BX

JNZ loop\_OUT

POP CX;

POP BX;

RET

DELAY ENDP

END MAIN ;this is the program exit point

1. Do output with 8255 IO controller.

Now let us move to real hardware devices, although it is still an emulated one.

In this experiment, 8 LED lights are connected to the data bus the 8086 processor through port A of a piece of 8255, and a piece of 74LS138 is used to do address decoding. As is shown in figure3.2. The Y0 output of 74LS138 is assigned to 8255. And Address BUS A3 to A5, together A6 and A7 are used to be input of address decoder, as is shown in figure3.3. The port address of 8255 can be calculated based on figure3.3.

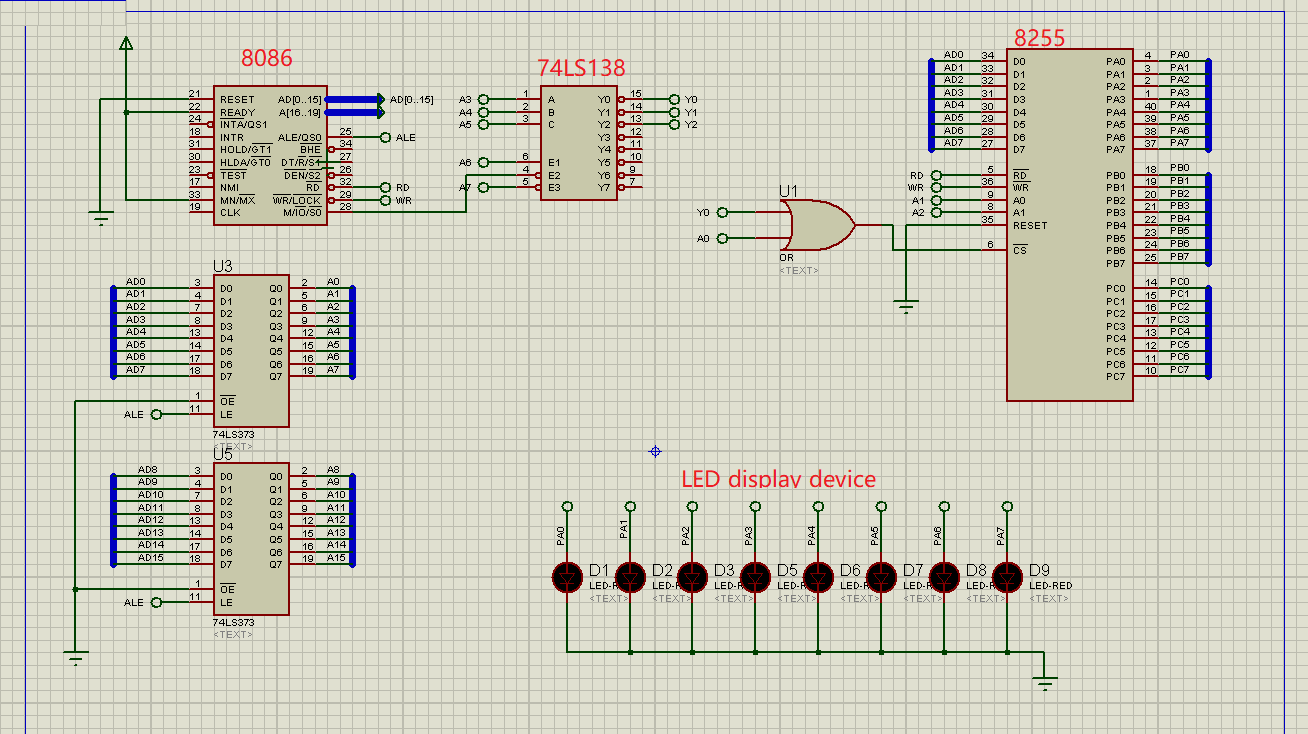


Figure3.2 Circuit schema of assignement2

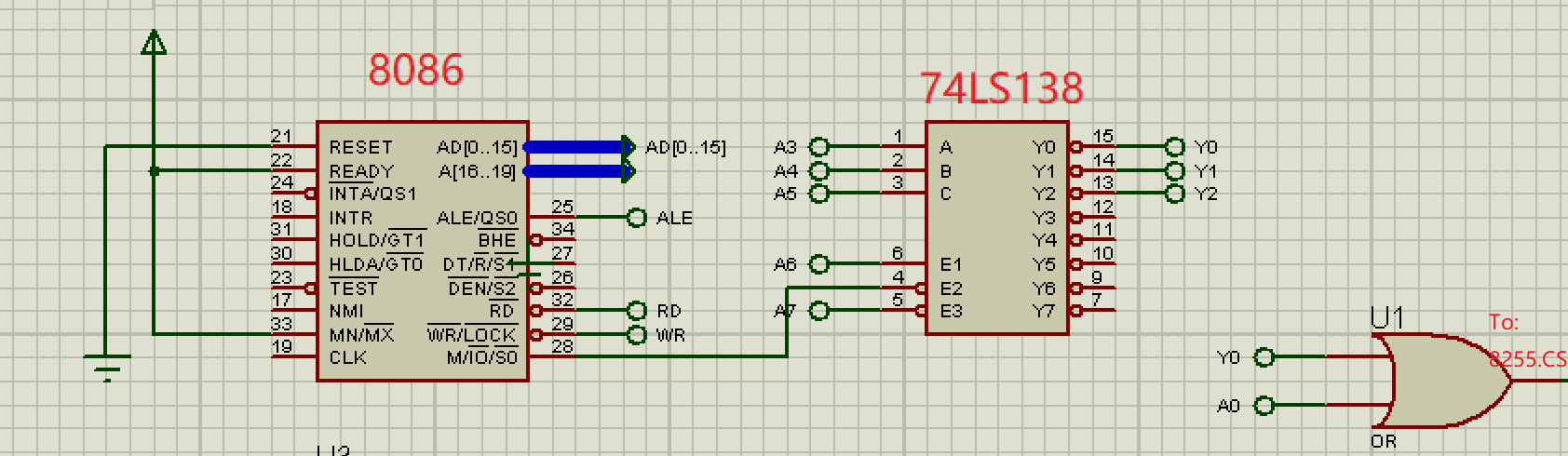


Figure3.3 Address decoding of assignement2

You are required to write a program to light the led one by one. You may start with D9, then light D8 and turn off D9, then move to the next D7, etc. When you reaches D1, please roll back to D2, and so on, till you turn on D9 again. And please do it repeatedly.

You may use logic shift instructions like SHL and SHR to generate the pattern code, and output pattern code through 8255.PortA to light the LEDs. And make use of a flag variable in your program to indicate on which direction should you shift the light.

1. Input and out put with 8255.

Now we try to do input and output together.

In assignment 3, we change the led lights into a nixie tube, and 4 pin switches. The switches are connected with PortA.PA0 to PA3, and are used to input a single digit hex decimal number. This number should be displayed by the nixie tube. The nixie tube is connect with PortB of 8255.

You are required to write a program, input the number setup by the switches, then convert the number into a pattern code. And then, use the pattern code to light the nixie tube and display the input hex decimal number.

You can make use of the XLAT instruction to do the convert from hex into pattern code.

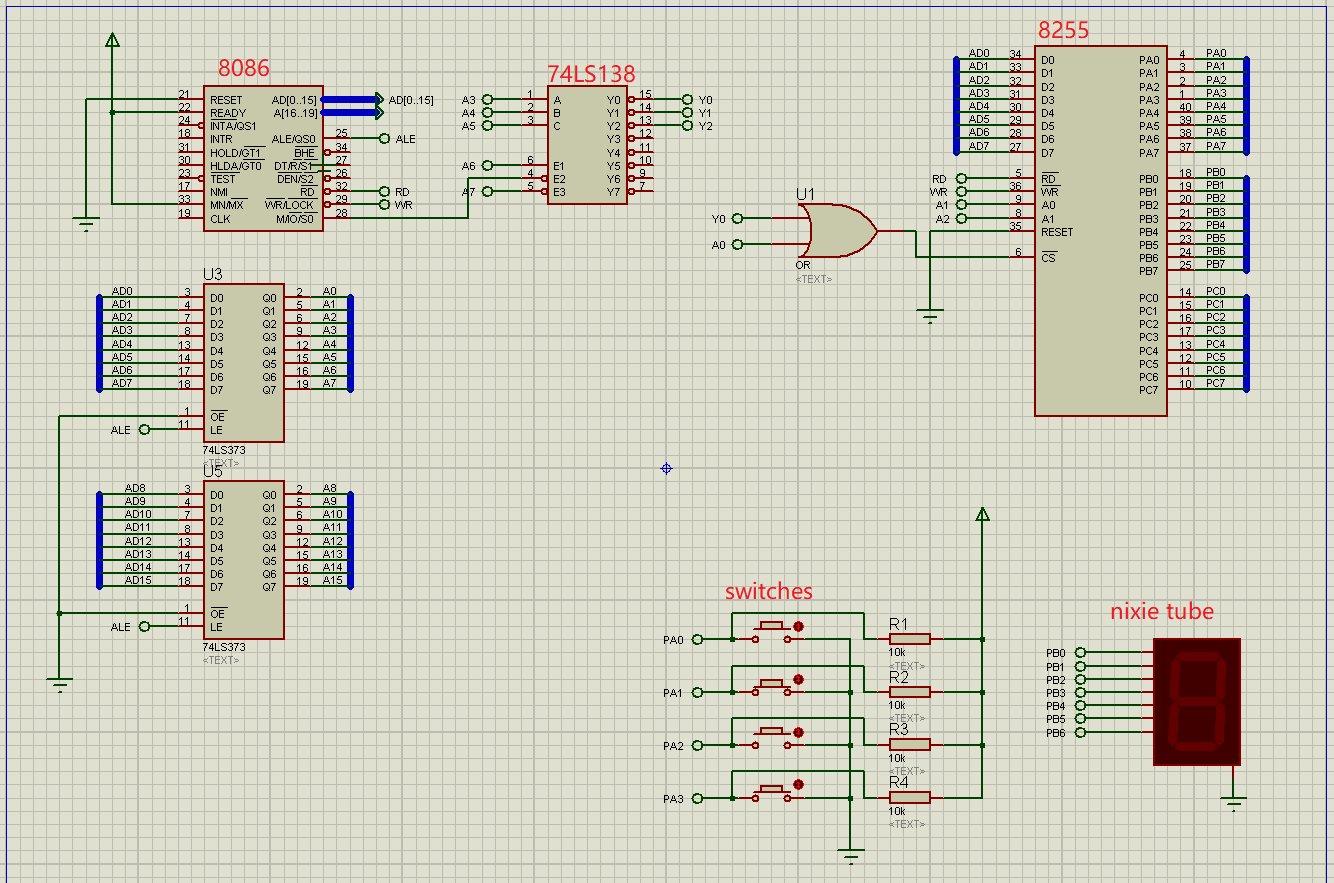


Figure3.4 Circuit schema of assignement3

## Experiment Preparation

All the thing below should be prepared for experiment1:

1. The programs for assignment 1, 2 and 3;
2. The VMWare Virtual Machine WindowsXP with Emu8086 and Proteus installed.

## Experiment Circuit Scheme

See figure 3.2 and figure 3.4.

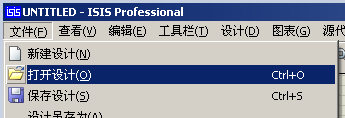
## Experiment Process

Start the the virtual machine Windows XP, and import all the three programs of the assignments.

**Assignment 1: Out put a data to a typical device port.**

1. Start Emu8086, and open the program of assignment1;
2. Emulate  run the program. The virtual led display device will appear automatically;
3. Run  the program, and debug it till you see correct display.

**Assignment2:Do output with 8255 IO controller.**

1. Start Emu8086, and open the program source code file of assignment2;
2. Try to compile the program, so that to remove any syntax error in the source code if exists; If the source code file an be compiled with Emu8086, you are ready to move to Proteus to do hardware involved simulation.
3. Start Proteus application, use menu item [File\Open schematic design] to load the hardware circuit schema file 8255LED.DSN into Proteus. A file choosing dialog will open for you to locate the .DSN file, as shown in Figure3.5.

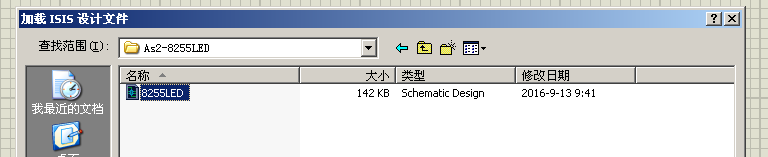
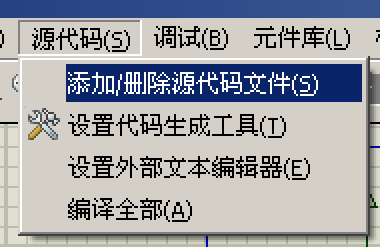


Figure3.5 load schematic design file

1. Copy the program source code file into the directory where hardware schematic design file (8255LED.DSN) resides. Then, in Proteus main window, choose menu item [Source Code\Append/Remove source code file(S)] to open the “Append/Remove source code file dialog” (figure 3.6).

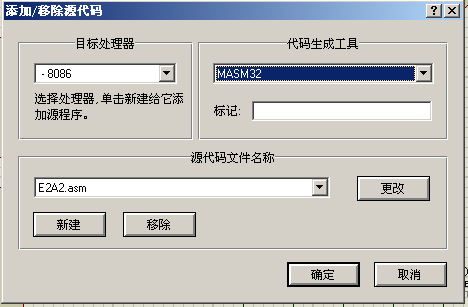


Figure3.6 Append/Remove source code file dialog

1. In the append/Remove source code file dialog (figure 3.6), press button [new]. A windows file choosing dialog appears, as shown in figure 3.7 below. When you choose your source code file and press [Open]  button, the source code file you selected (in this example, it is E2A2.asm) is attached to current hardware schematic design file 8255LED.DSN. Now in the right side [code generation tool] square, use the drop down selection box to assign MASM32 to the source code file, as shown in figure 3.6. Then press [confirm]  button.

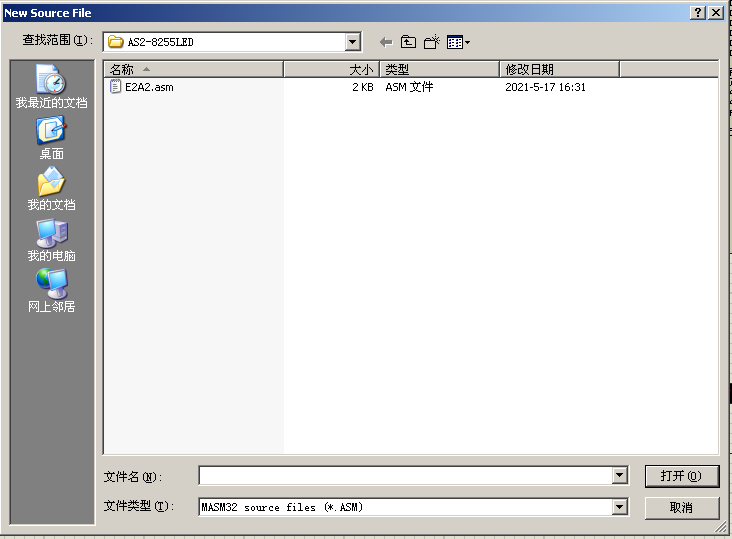
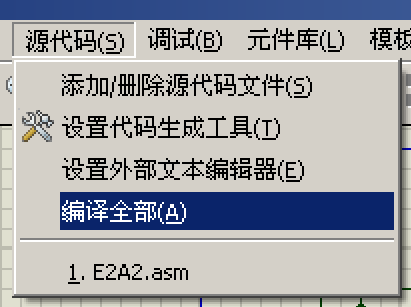


Figure3.7 choose the source code file

1. Now, open the [source code file] menu group again, you can find the source code file appended in step (5) appears as one of the menu item . When you click the source code file name menu item, a source code editor window will popup. However, as we do not need to modify the program, we choose [Compile All(A)]  menu item. The source code program will be assembled by MASM32 SDK, and a “Build Log” window popup. If there is no error in the source code, it will appears as shown in figure 3.8.

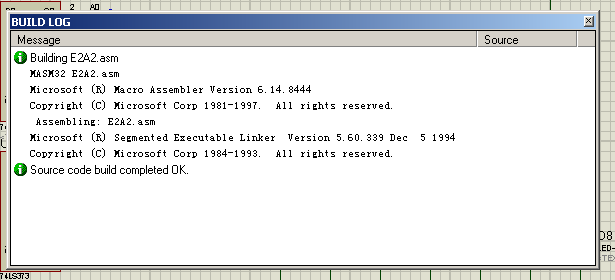
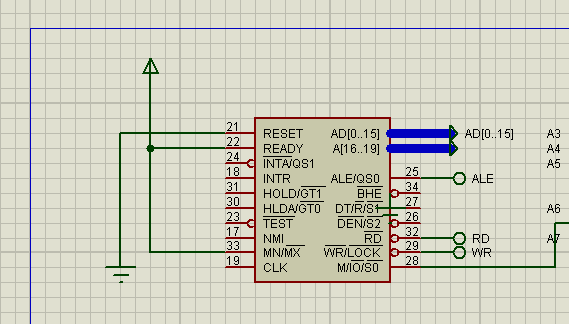


Figure3.8 Build Log window -- build completed OK.

1. Close the Build Log window, and all the other popup windows, and double click the 8086 processor item in the main window to open a property editor window, as shown in figure 3.9. Click the [file open]  button beside the [Program File: input box] to open a file choosing window. Select the executable file you produced in step (6) and press [Open]  button. The executable file E2A2.exe is attached to the 8086 processor item. The results looks like figure3.9. Click [confirm]  button to close properties edit window.

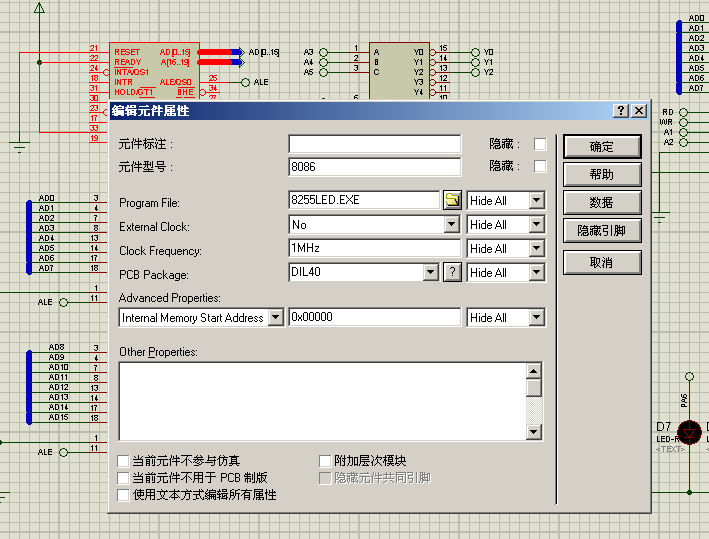
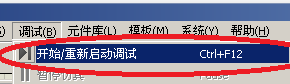


Figure3.9 Properties edit window -- attach executable file to 8086

1. Now, you can use the [run] button in main window bottom left corner to emulate run your program with current hardware schematic design.
2. If the program contain bugs, use the [Debug\start/restart to debug...] menu item  to start single step debugging process of your program. As shown in figure 3.8.

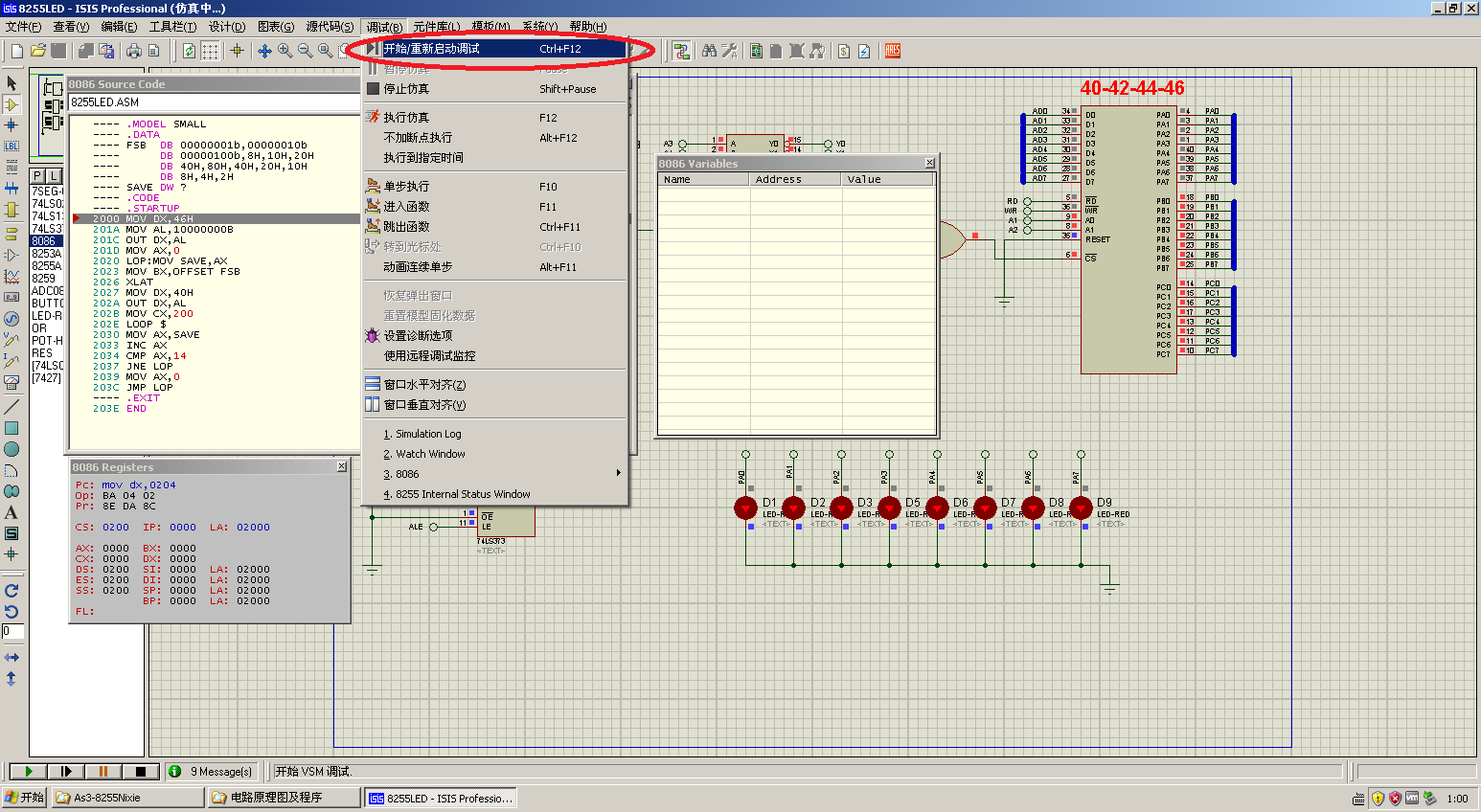


Figure3.10 Debug the program

**Assignment3:Input and out put with 8255.**

Follow the steps described in assignment 2 to run or debug your program for assignment3. Make use of hardware schematic design 8255NUM.DSN to do the simulation.

Click the red spot in schematic to toggle switches ON/OFF state, and change the displayed number, as shown in figure3.11.

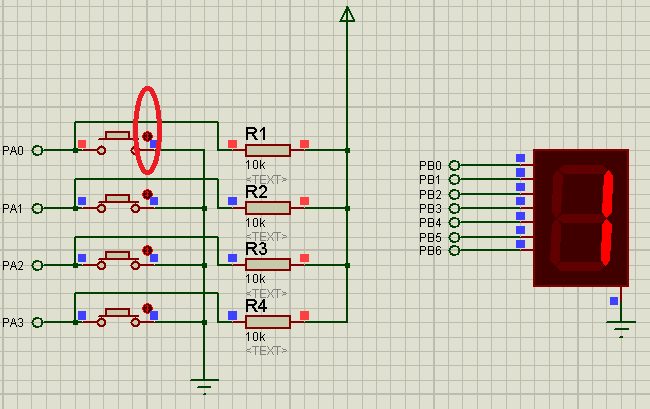


Figure3.11 toggle switches and change display

## Chapter 4 Experiment 3 Display devices Interfacing

In Chapter 4 we will practicing how to make use of 8254 PIT controller. And moreover, move a little bit forward, program and debug a small complicated program to display something with six nixie tubes.

## Assignments

1. Frequency division practicing .

In this experiment, a crystal oscillator of 500kHz is used as the clock source of a piece 8253 PIT controller, as shown in Figure4.1 the circuit schema. And there are 3 led lights are connected to the signal out pins of the 8253. You are required to write a short initialization program to setup the 8253, so that to twinkle one of the led in a frequency of 0.5Hz, and the led should be light for nearly 1s, and be off for another 1s. A template is provide for the program in 8253Div.asm, as is shown in text box below.

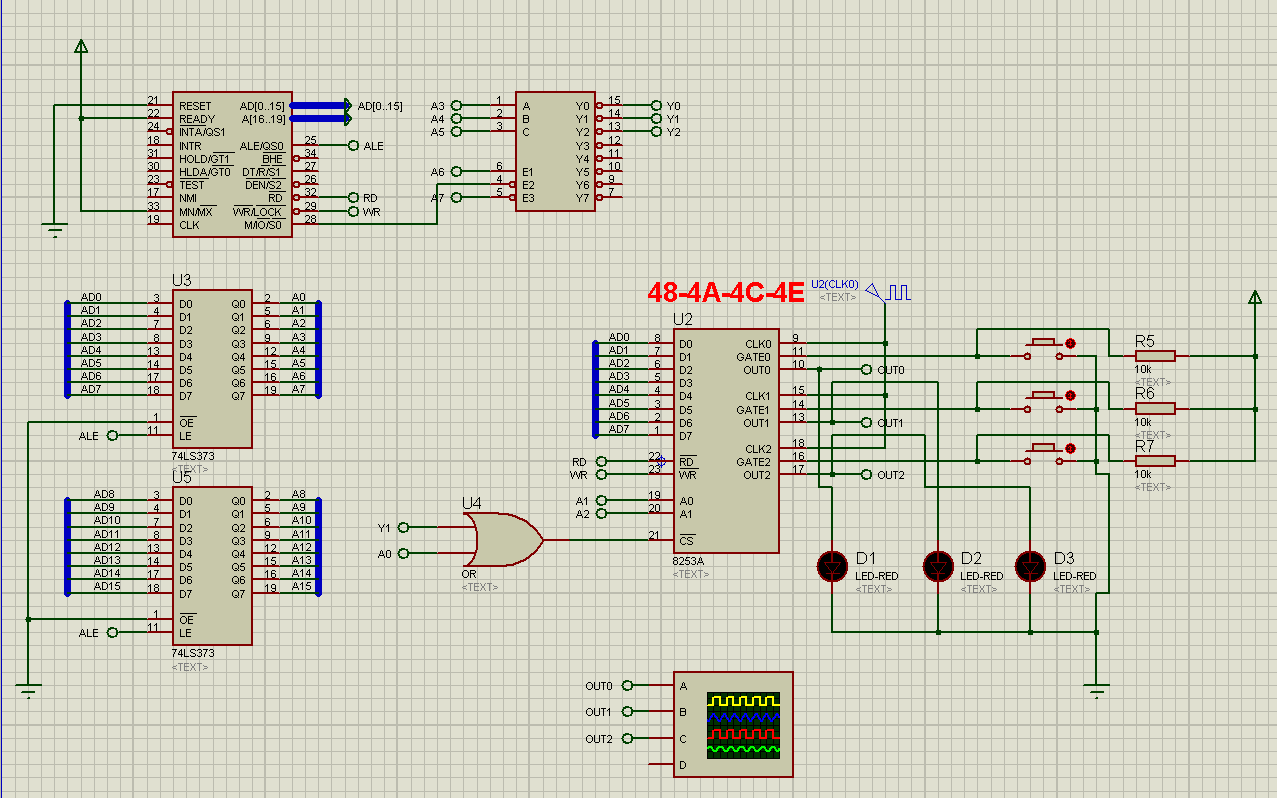
****

Figure4.1 Circuit schema of assignment1

;========================================================

;Description: Program of Assignment 3 Experiment1

;Author:[name][student ID]

;Date:[Date]

;========================================================

;view 8254 output

;LED1 displays out0 of timer0

;LED2 displays out1 of timer1, and LED1 should light for 1s and off for 1s

.MODEL SMALL

.STACK 32

.DATA

IOS4 EQU 048H;

.CODE

MAIN PROC FAR

;TODO1: program timer0, both cmd and initial value

;TODO1: program timer1, both cmd and initial value

MOV AX, 4C00H

INT 21H

MAIN ENDP

END MAIN

1. Digital tube display control.

In this experiment, a digital tube constructed by 6 nixie tubes are provided as a display, as shown in figure4.2. These six nixie tubes share a group of common inputs: pinA, B, C,...P,...U,DP. By providing a high voltage at these pins, the display fields of all the tubes will be turned on. Thus if we want to display only one digit number, we should active only one tube but not all of them before we send pattern\_code to the common input pins. The tube selective pins are on the right side: pin1,2,...5. By providing a GND voltage to one of the selective pins, and high voltage to others, we can active only one nixie tube for display. In the digital tube display in Figure4.2, pin1 will active the tube on the leading left side, and pin6 will active the tube on the most right side. As a result, if you want to display 6 digits, you should active tube1 in the first place, and output pattern\_code of the first digit to the digital tube display, then turn off tube1, and turn on tube2, output the pattern\_code of the second digit. And so on till you display the sixth digit. Then roll back to display the first digit with tube1, and repeat continuously.

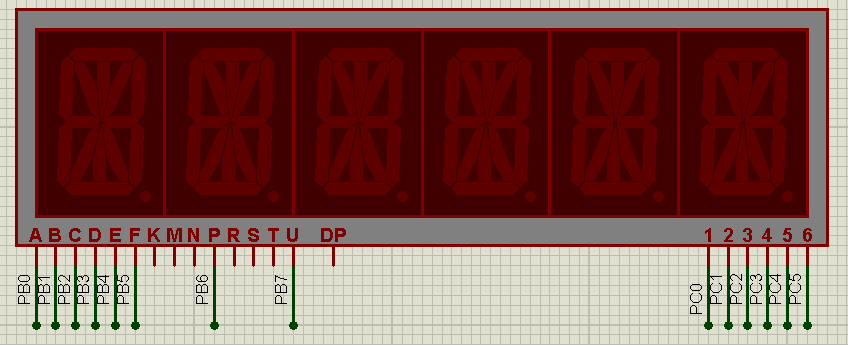


Figure4.2 Digital tube display

In the hardware schematic design of assignment2 (figure4.3), the digital tube display is connected to the ports of a piece of 8255. PortB is used to send pattern\_code, and PortC is used to send tube-active-code.

There is a variable DATA1 defined in the DATA segment. It contains 6 digits and is stored in the DATA segment in unpacked BCD format. Please write a program and display DATA1 with the digital tube display. A template of the program is provided in file 8255Tube.asm. And port numbers and pattern\_codes table are defined in the template code file.

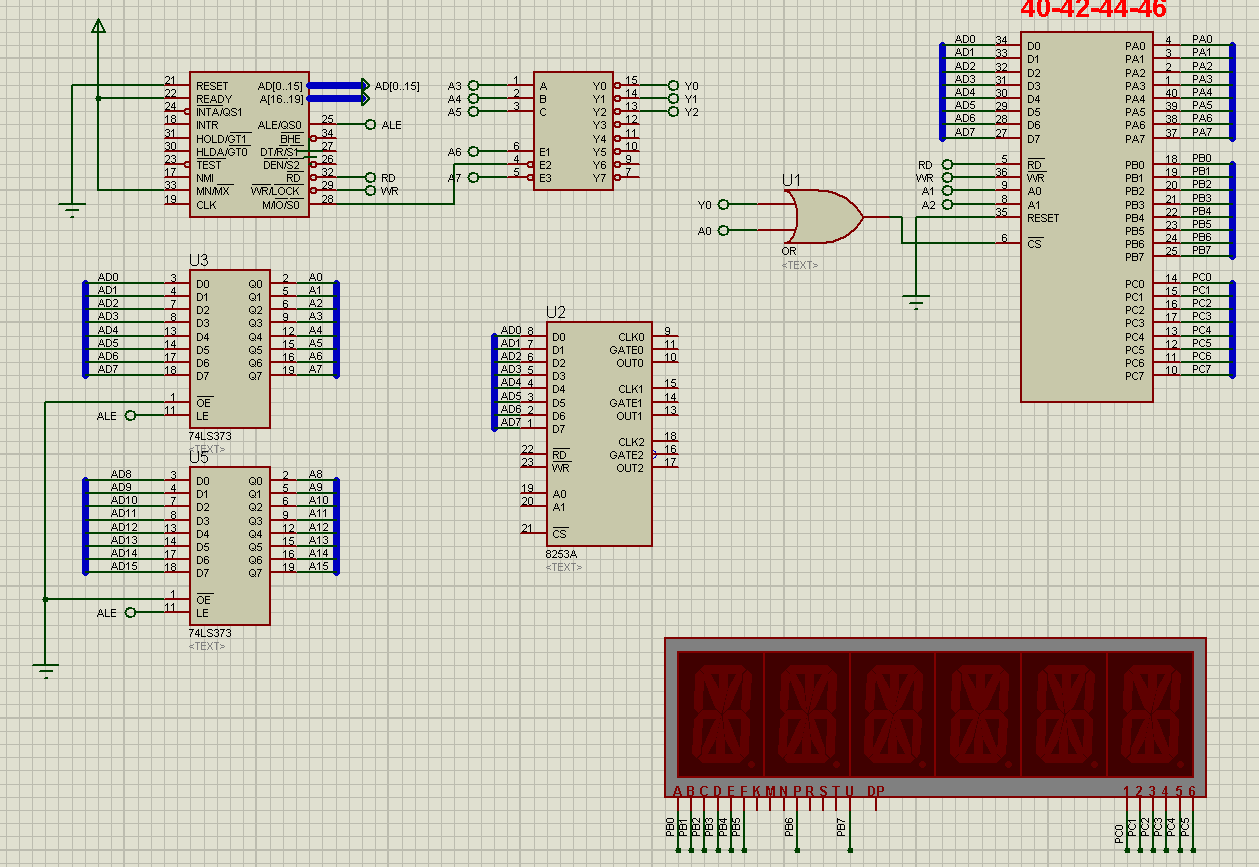


Figure4.2 Circuit schema of assignement2

## Experiment Preparation

All the thing below should be prepared for experiment1:

1. The programs for assignment 1 and 2;
2. The VMWare Virtual Machine Windows XP with Emu8086 and Proteus installed.
3. Import hardware schematic design files into Windows XP virtual machine.

## Experiment Circuit Scheme

See figure 4.1 and figure 4.2.

## Experiment Process

Start the the virtual machine Windows XP, and import all the programs of assignments.

Debug the program till it runs properly.

## Chapter 5 Experiment 4

In Chapter 5 we will practicing how setup a interrupt handling system by using a piece of programmable interrupt controller 8259. We generate a interrupt request by pressing a spin button, and in the interrupt service routing, we will move the light of a lantern.

## Assignments

1. Simple interrupt handling systems .

In this experiment, a piece of 8259 is used to expand the hardware interrupt source of a 8088 processor from one to eight. In its interrupt request IR0, a spin button is connected. When we press the spin button, a high level input will generate a interrupt request. The PIT 8259 is set to be working single piece, normal EOI, level triggered, slave buffered mode, and the interrupt number for IR0 is 40H. As is shown in Figure 5.1 and Figure 5.2.

On the other side, a latch buffer chip 74LS373 is used to interface 8 led lights, and this unit is used as the output device of experiment.

You are required to do:

1. Program the PIC 8259;
2. Write a interrupt service routine (ISR), which is a far typed sub procedure. In the ISR, you are required to change display of the lantern, and move the light upward. If the light has already been the top led, then roll back to the bottom one;
3. Register the ISR to the operating system, and run your program with the hardware schematic design.

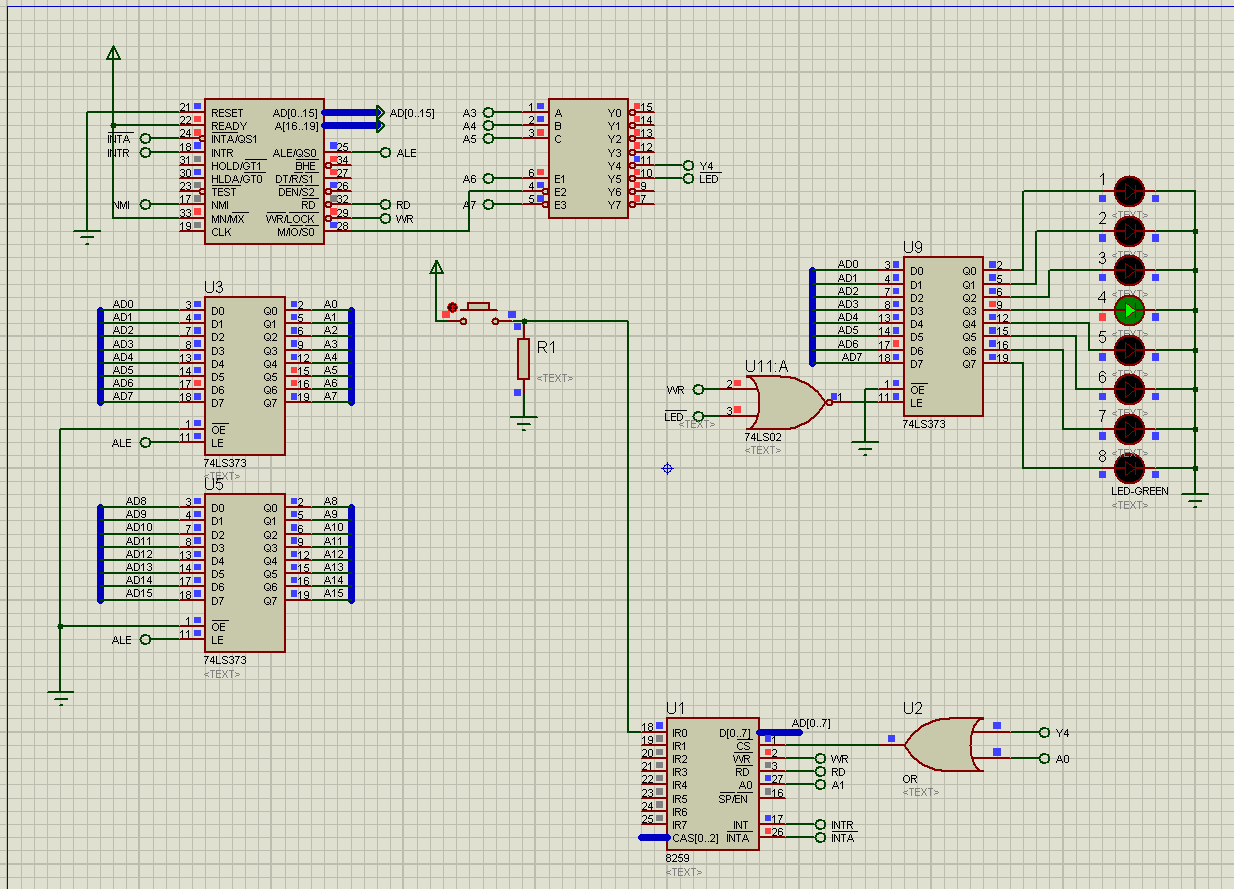


Figure5.1 The hardware schematic design

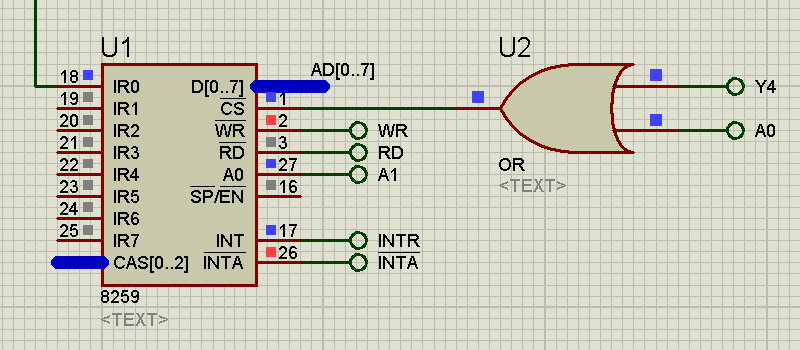


Figure5.2 The PIC 8259

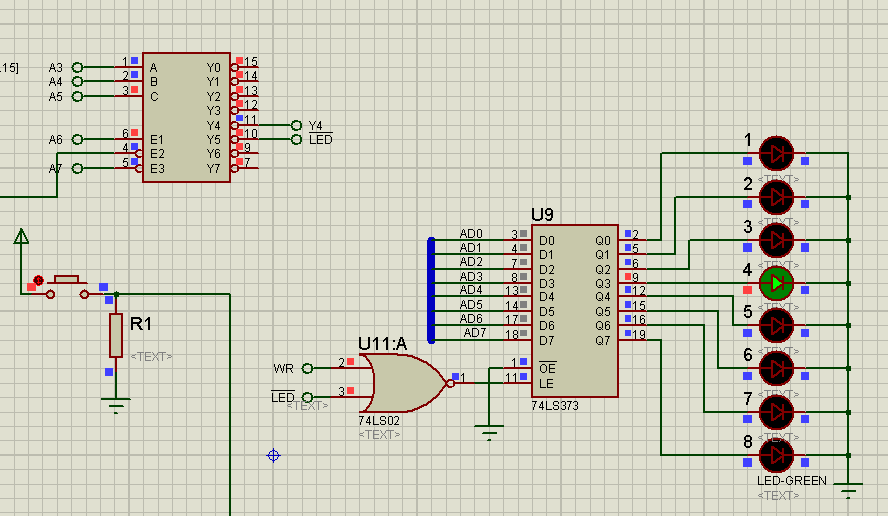


Figure5.3 Address decoder and lantern display unit

## Experiment Preparation

All the thing below should be prepared for experiment1:

1. The programs for assignment 1;
2. The VMWare Virtual Machine Windows XP with Emu8086 and Proteus installed.
3. Import hardware schematic design files into Windows XP virtual machine.

## Experiment Circuit Scheme

See figure 5.1, figure 5.2, and figure 5.3.

## Experiment Process

Start the the virtual machine Windows XP, and import all the programs of assignments.

Debug the program till it runs properly. 

Notice：

There is some bugs with the 8259 component of Proteus software. It will not response to the second round of INTA signal. Thus 8086 will not retrieve correct interrupt number from 8259.

As a result, we must ensure that when the 8086 is handling the interrupt request and try to retrieve interrupt number from 8259, the correct interrupt number appears in the data bus. We satisfied the requirement above by output the interrupt number 40H to a null port in a continuous loop, as is shown in Figure5.4

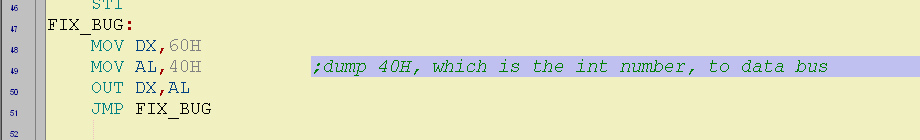


Figure 5.4 output interrupt number to the data bus