SDCC
8051 Assembly Language Programming Guide

Date: August 31, 2010

Document Revision: 1.01

BiPOM Electronics

16301 Blue Ridge Road, Missouri City, Texas 77489
Telephone: 1-713-283-9970. Fax: 1-281-416-2806
E-mail: info@bipom.com
Web: www.bipom.com

Overview

SDCC 8051 Development System has a built in 8051 Assembler that allows developing programs in assembly language or mixed C and assembly for project development and educational purpose.

Assembly language can be embedded in C programs as inline assembly. Alternatively, assembly language code may have its own source file. SDCC 8051 Development System allows having multiple assembly source files or mixing assembly and C source files in the same project.

Our 8051 Simulator and 8051 Debugger tools also support assembly language development. For example, it is possible to single step through assembly code, set breakpoints and watch register values. This simplifies assembly program development and is also a valuable learning tool for users learning about microcontrollers.

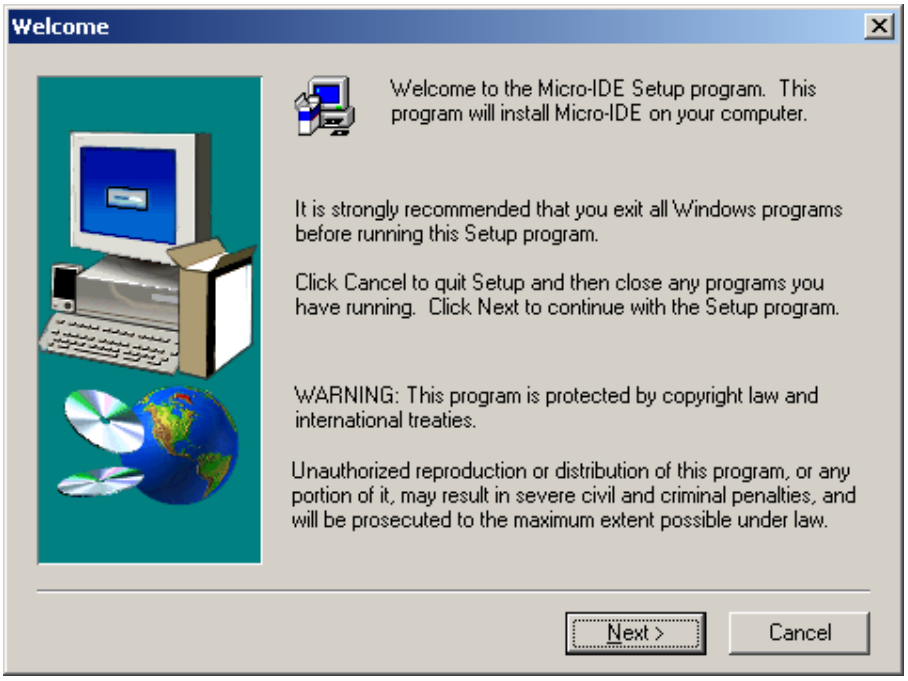
Software Setup

Download SDCC 8051 Development System from:

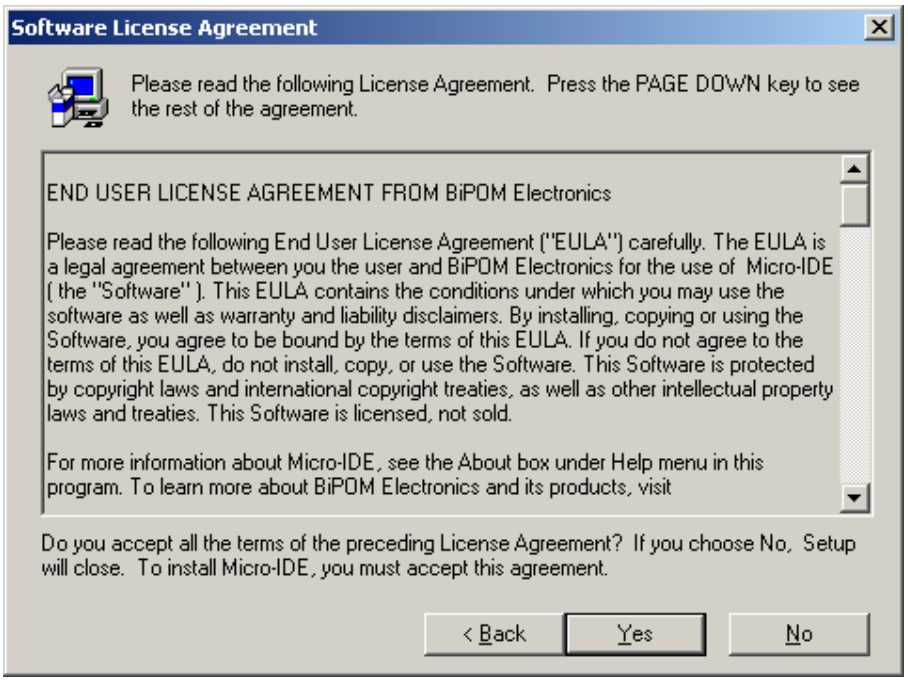
http://www.bipom.com/sdcc_down.php

Start the installation by downloading and running sdcc_devsys.exe

A Welcome screen will appear:

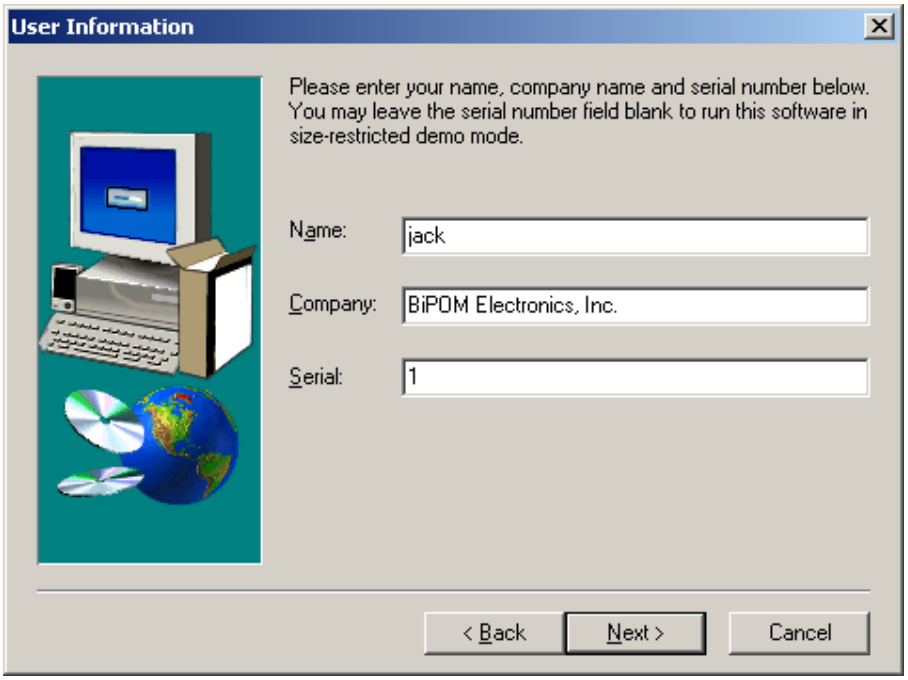


Click on Next. End User Agreement will appear:



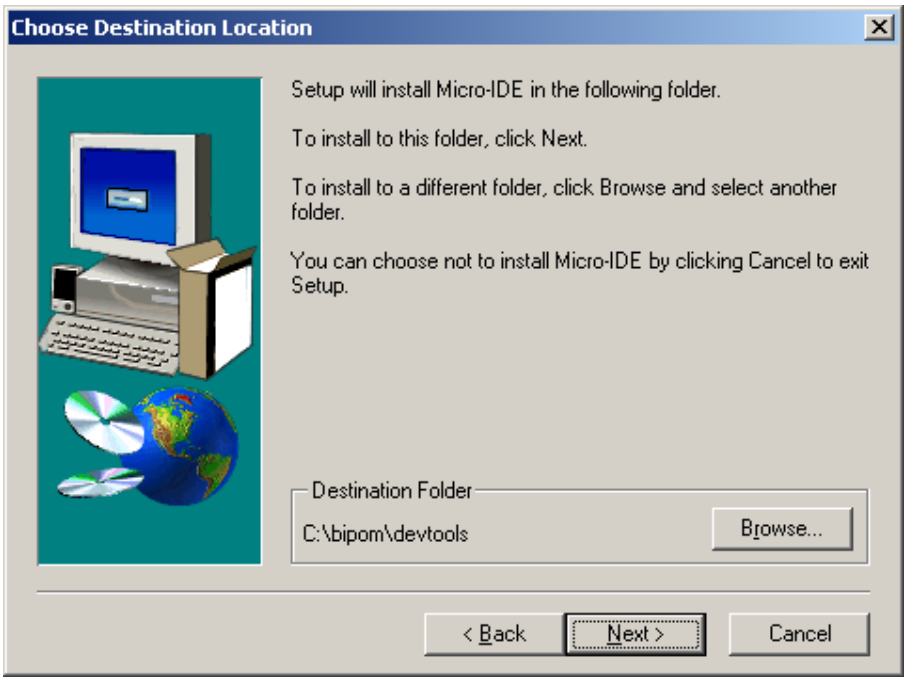
Please read the agreement and click Yes if you wish to continue with installation.

Enter your name, company (if applicable) and serial number:

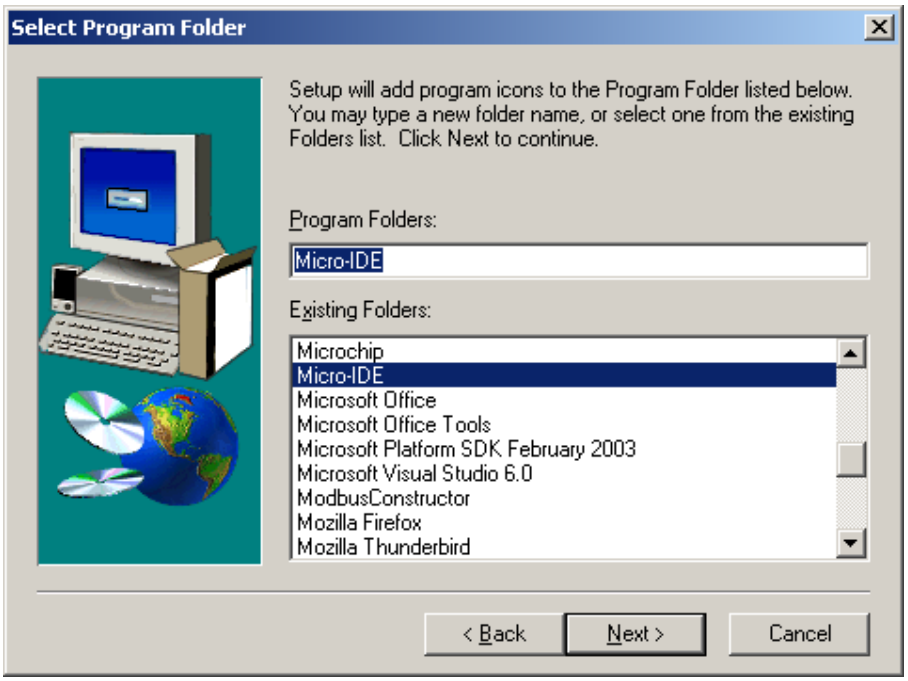


Enter a serial number of 1. Click Next.

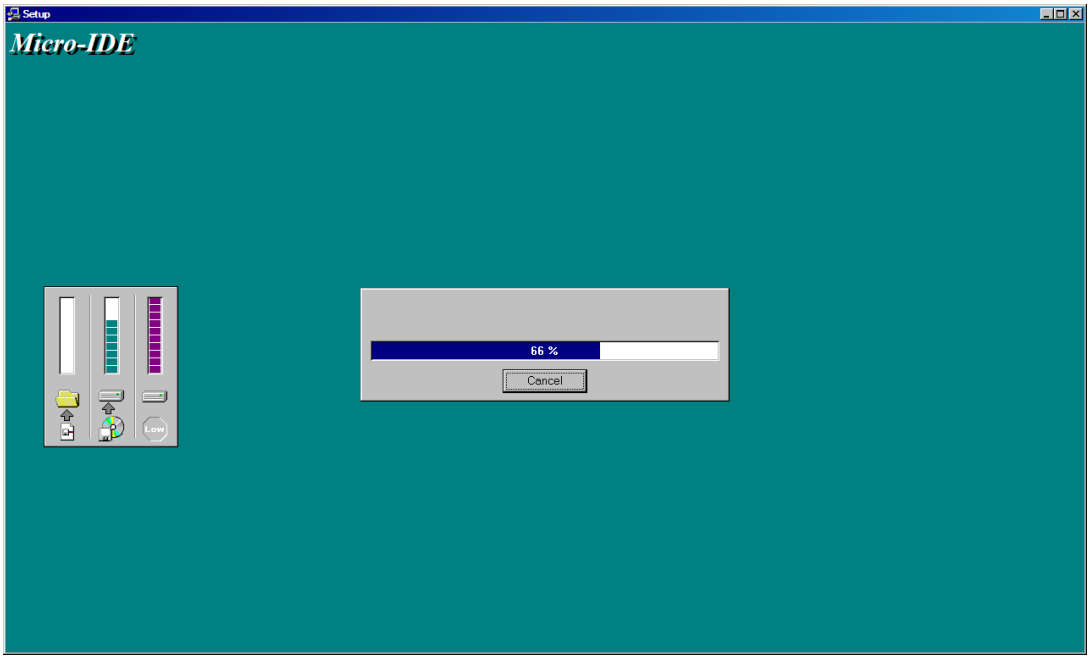
Select the disk location where the software will be installed. Using the default location of c:\bipom\devtools is recommended:



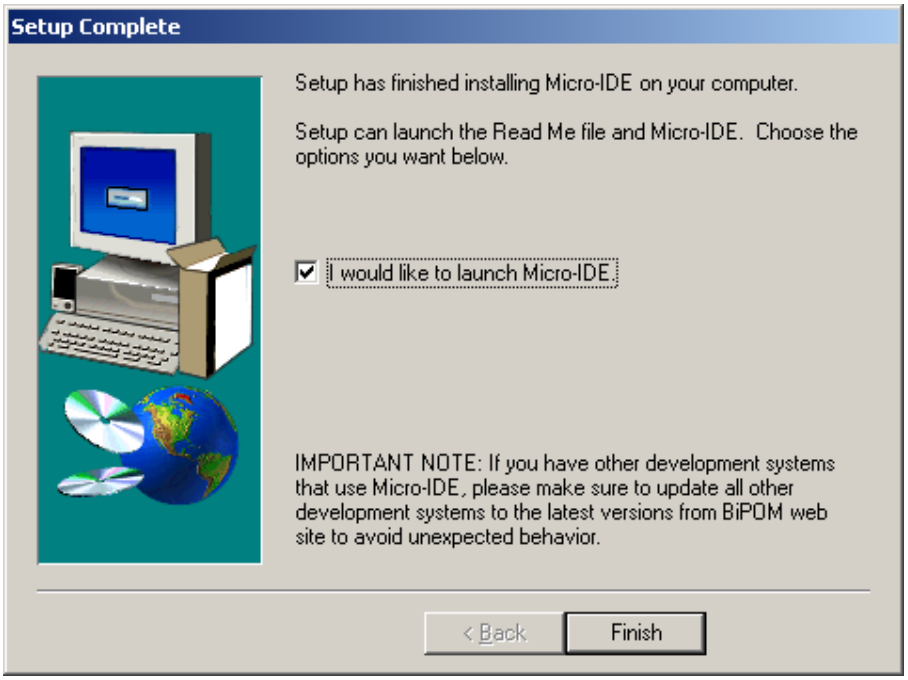
Click Next. Select the Program Folder where the icons for Micro-IDE will be installed. Default selection is **Micro-IDE** folder.



Click Next. Micro-IDE will be installed and you will see the progress:



When the installation is complete, you will be given an option to start Micro-IDE now:



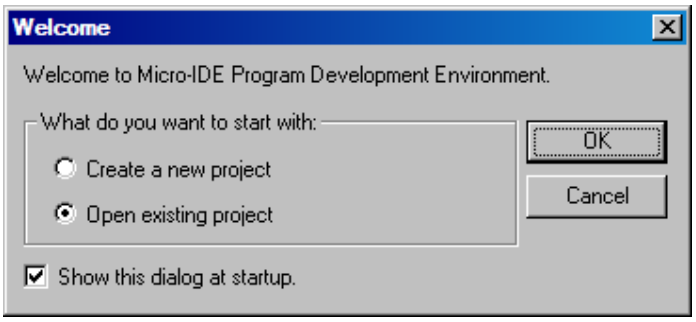
Click Finish and Micro-IDE will start.

Downloading Example Programs

After installing the software, you can build assembly language programs and download programs to the board. Follow the steps below:

1. Make sure that your target board is powered and connected to the PC. This manual assumes that you are using a BiPOM microcontroller board such as MINI-MAX/51-C2, MINI-MAX/51-D, MINI-MAX/51-E or MINI-MAX/51-F. You can use other 8051 boards, too.

2. Run Micro-IDE from Windows Start menu. When Micro-IDE is started, the Project selection window appears:

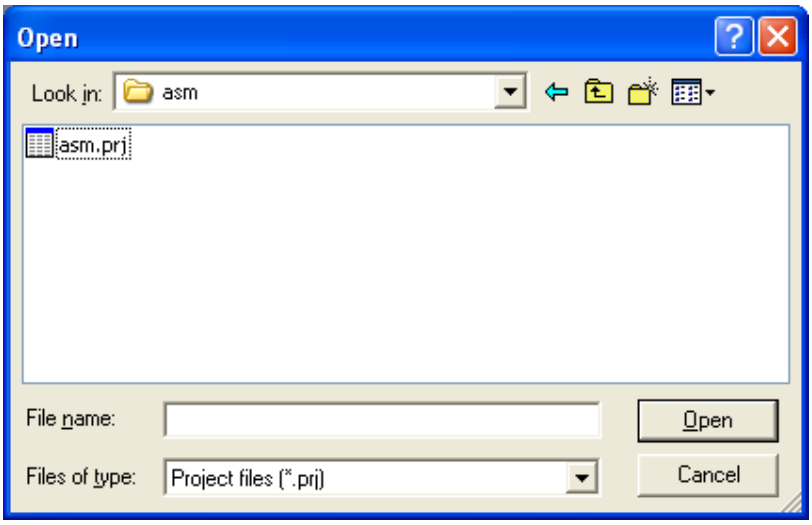


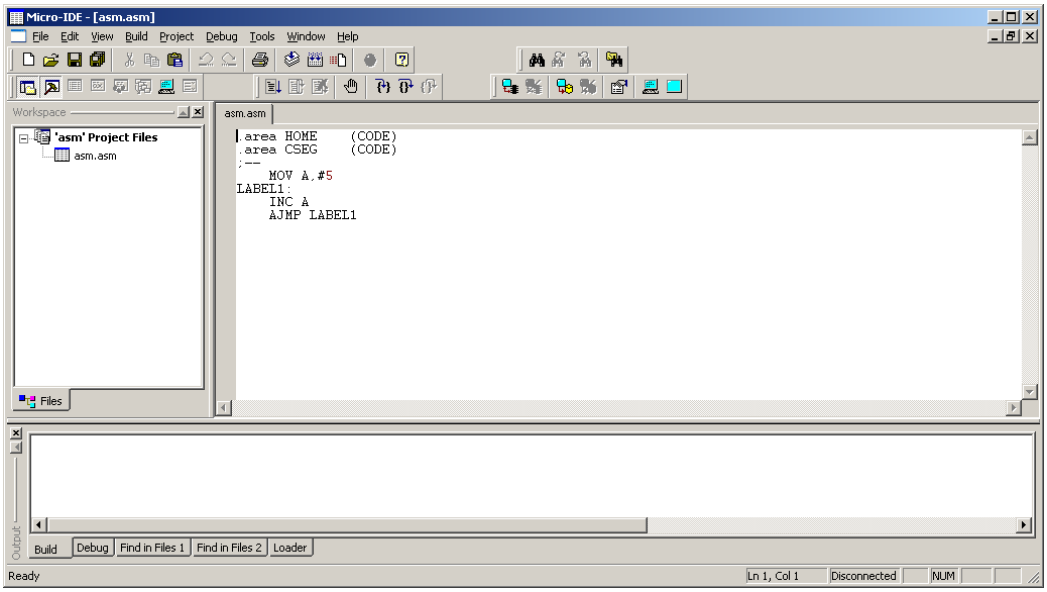
Click OK to select an existing example project.

Micro-IDE is distributed with several example programs that illustrate how to program the 8051 micro-controller. Example projects are located under the Examples folder under the folder where you installed Micro-IDE. Some of the examples are written in C and some of them are written in assembly language.

3. Select Project, Open Project and open the example project **asm.prj** from:

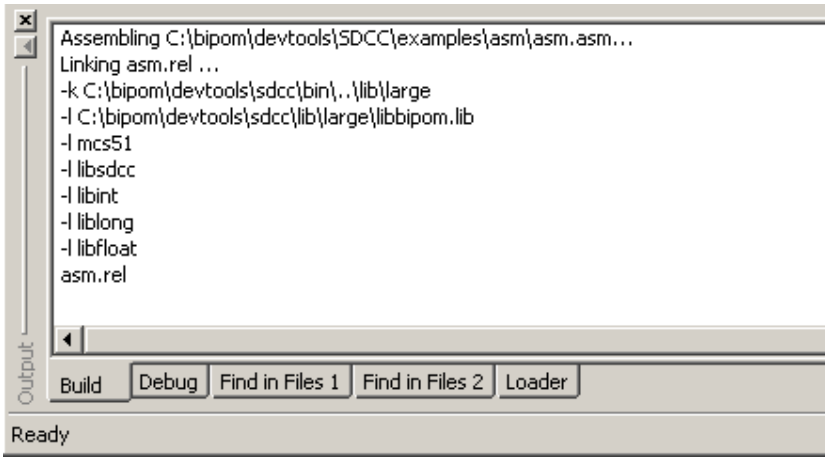
C:\bipom\devtools\SDCC\Examples\8051\tiny\asm





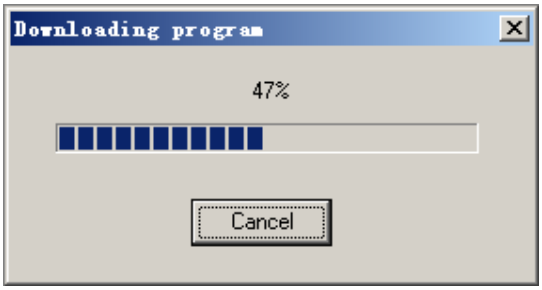
4. Click the Build button on the main toolbar. This will build the asm project:

If the project builds successfully, you should see the output messages from the assembler on the Output Window:

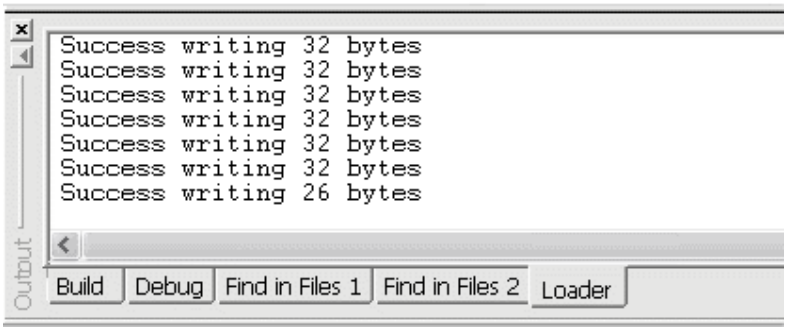


5. Download the executable (asm.ihx) file to the board by selecting Download under Build menu:

If the MINI-MAX/51-C2 board is powered and connected properly to the PC, a progress dialog will appear:



The progress dialog will disappear following a successful download. Details of the download are
shown on the Output Window:



When the download is finished, the progress indicator disappears. This means that the board has received the program successfully. After the program has been successful downloaded, it can be started using the Mode button on the main Toolbar:

Mode button puts the board into **Run** or **Program** mode. In Run mode, the microcontroller is executing the program in its memory. In Program mode, the microcontroller is in Reset state so no programs are running. In Program mode, microcontroller's flash memory can be changed and a new program can be downloaded.

The Mode button is Red in Program mode and Green in Run mode. Following a download, the Mode button will be Red. Click the Mode button to change the mode to Run mode. The program **asm.hex** that you just downloaded starts executing.

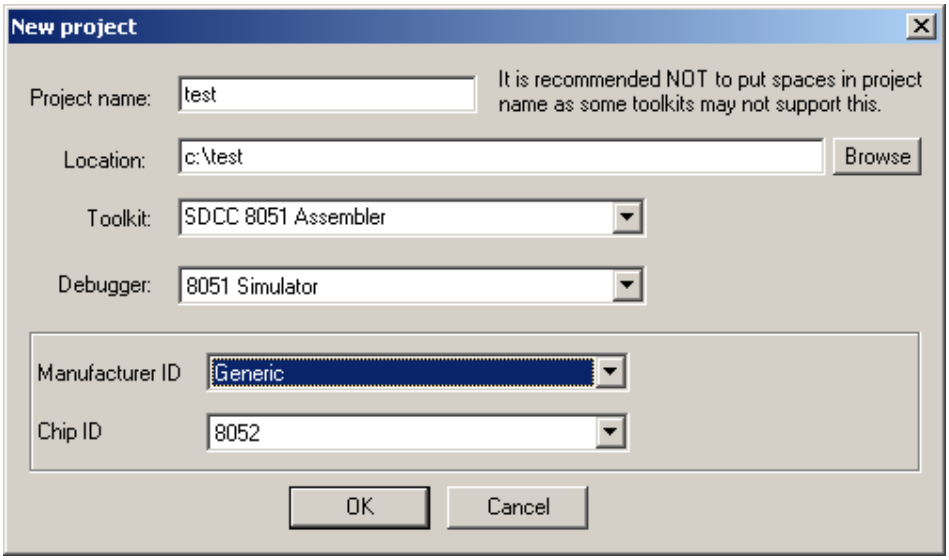
You should now see the characters that you type on the Terminal window being echoed back to you.

Congratulations!!! You have built and executed your first assembly language program on the MINI-MAX/51.

Click the Mode button once again so it turns Red. The board is in Program mode

Writing Your Own Programs

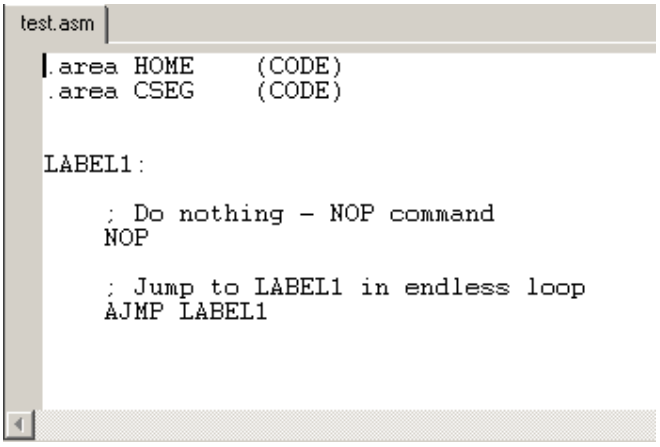
To create your own project, select Project menu and select New Project. This will display the New Project dialog:



Enter the name of the new project and its location (this examples uses **test** as the project name and **c:\test** as the project location). Select the Toolkit as SDCC 8051 Assembler. Click OK. Say Yes to create new directory. You will be prompted to select a project type:

Select “SDCC Empty ASM Project” and click OK.

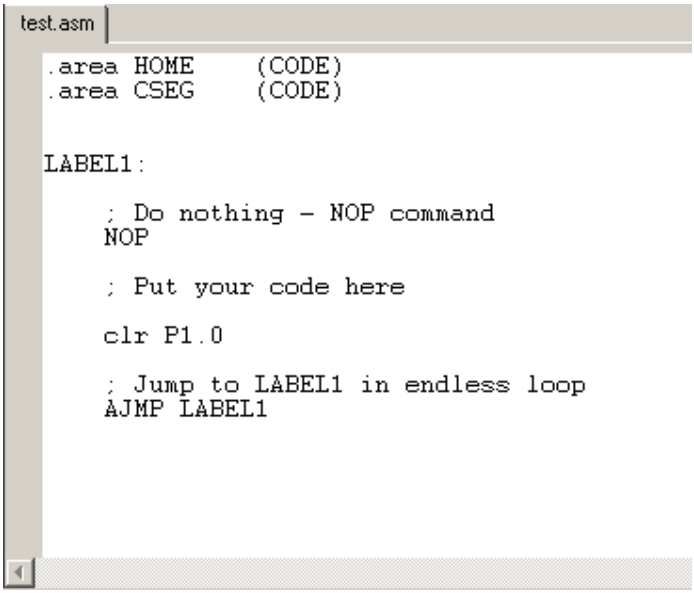
The new project with the name of **test** under **c:\test** will be created. A blank assembly language file (test.asm) will be automatically created:



Type the following sentence into the section where it says “Put your code here”:

clr P1.0

The assembly program should now look like this:



This program changes port pin P1.0 to logic low level. P1.0 is connected to the Green LED on the Training Board (TB-1) so this program will turn on the Green LED if TB-1 is installed. Build the program by clicking the Build button. If the program builds successfully, you will see the following messages on the Output Window:

Assembling c:\test\test.asm…
Linking test.rel …
-k c:\bipom\devtools\sdcc\bin\..\lib\large
-l c:\bipom\devtools\sdcc\lib\large\libbipom.lib
-l mcs51
-l libsdcc
-l libint
-l liblong
-l libfloat
test.rel

Download the program to the board by clicking the Download button on the main toolbar.
Run the program by clicking the Mode button on the main toolbar. You will see the Green LED on TB-1 turning on if TB-1 is connected.

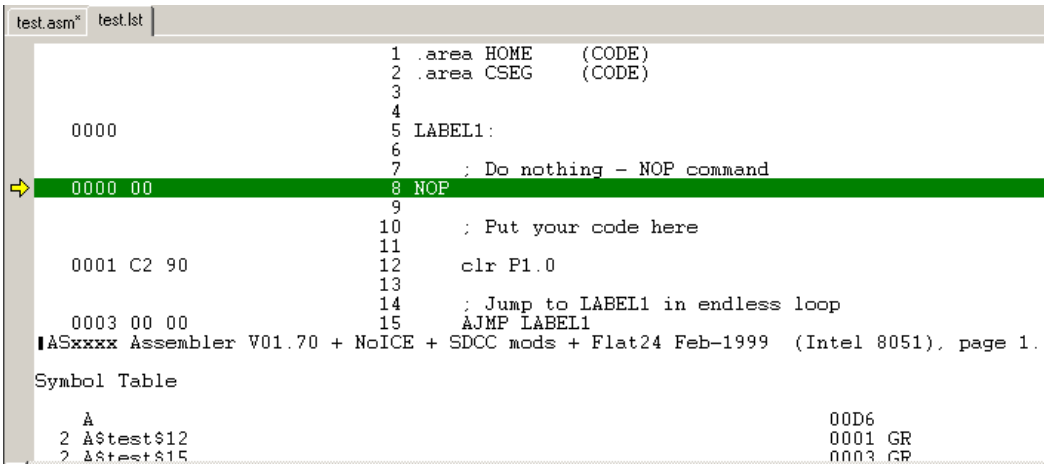
Using the 8051 Simulator

BiPOM's 8051 Simulator fully supports the 8051 Assembly language programs written in SDCC. In the simulator, the 8051 instructions are simulated on the PC as if they are executing on the target board.

You can simulate the micro-controller and single-step through your program without actually having the board connected to your PC. The demo Simulator that comes with Micro-IDE is limited to 1K of 8051 code. You can purchase the SDCC 8051 Simulator option from BiPOM Electronics to remove this restriction and obtain a full-featured Simulator license.

To start the simulation, press F11 or click the Step Into button on the Debug toolbar:

This will start a simulation session and will bring up the Listing file (test.lst) for this project. Listing file is an Assembly language representation of your program with additional information. The execution will stop on the first code line that will be highlighted:



To view the simulated 8051 registers, activate the Register window (if it is not already visible) by clicking the Register window button on the Window toolbar. To view the simulated 8051 memory, click the Memory window button on the Window Toolbar:

Registers

Name	Value	Extended
ACC	00	
B	00	
SP	07	00
DPTR	00	58
PC	02	
PSW	00	
IE	00	
IP	00	
R0	00	00
R1	00	00
R2	00	
R3	00	
R4	00	
R5	00	
R6	00	
R7	00	
Port0	FF	
Port1	FE	
Port2	FF	
Port3	FF	
PCON	00	
SBUF	00	
SCON	00	
TCON	00	
TMOD	00	
T0	00	

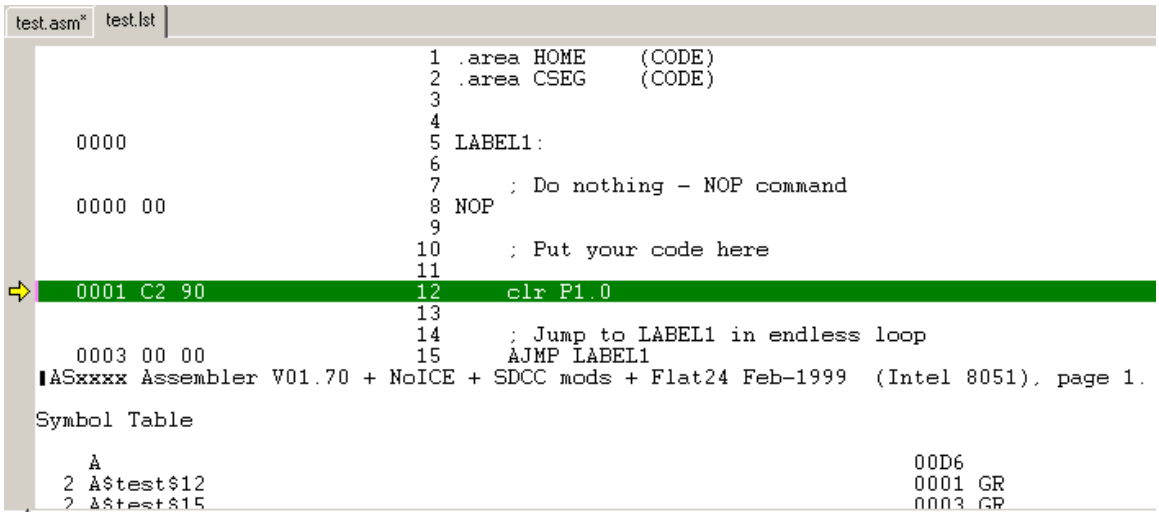
Memory

Program Memory | Address 0x00000000

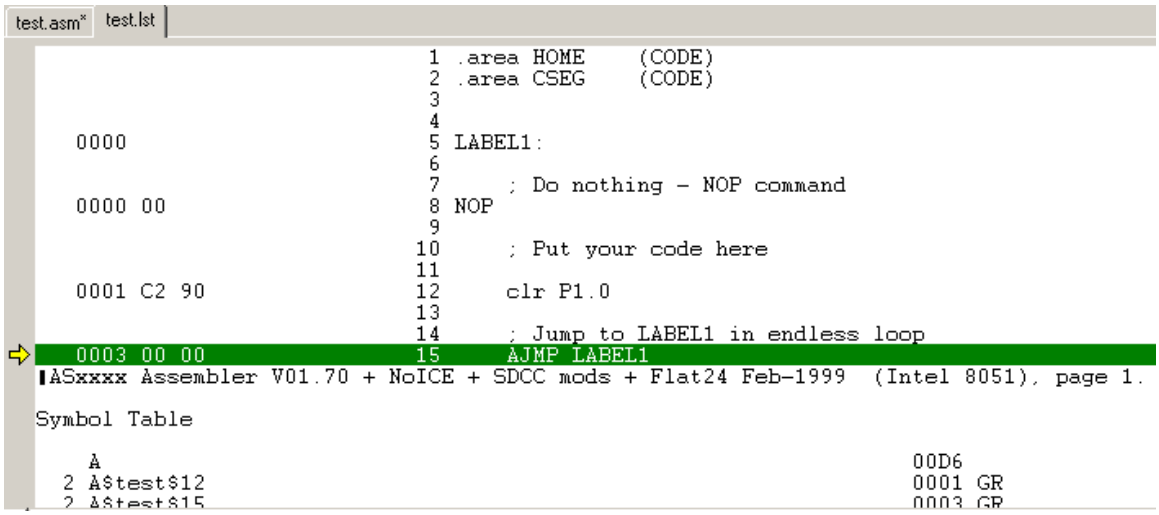
```
00000000 C2 90 01 02 00 00 00 00 ........
00000008 00 00 00 00 00 00 00 00 ........
00000010 00 00 00 00 00 00 00 00 ........
00000018 00 00 00 00 00 00 00 00 ........
00000020 00 00 00 00 00 00 00 00 ........
00000028 00 00 00 00 00 00 00 00 ........
00000030 00 00 00 00 00 00 00 00 ........
00000038 00 00 00 00 00 00 00 00 ........
00000040 00 00 00 00 00 00 00 00 ........
00000048 00 00 00 00 00 00 00 00 ........
00000050 00 00 00 00 00 00 00 00 ........
00000058 00 00 00 00 00 00 00 00 ........
00000060 00 00 00 00 00 00 00 00 ........
00000068 00 00 00 00 00 00 00 00 ........
00000070 00 00 00 00 00 00 00 00 ........
00000078 00 00 00 00 00 00 00 00 ........
00000080 00 00 00 00 00 00 00 00 ........
00000088 00 00 00 00 00 00 00 00 ........
00000090 00 00 00 00 00 00 00 00 ........
00000098 00 00 00 00 00 00 00 00 ........
000000A0 00 00 00 00 00 00 00 00 ........
000000A8 00 00 00 00 00 00 00 00 ........
000000B0 00 00 00 00 00 00 00 00 ........
000000B8 00 00 00 00 00 00 00 00 ........
```

All the standard 8051 registers are simulated and viewed on the Register window. To see the remainder of the registers, scroll down the Register window. Note that at program startup, ports are all 0xFF, which means that all the Port pins are high logic level.

Press F11 one more time. You will notice that the current execution line (green bar) has advanced to the next line:



Press F11 one more time. You will notice that the current execution line (green bar) has advanced to the next line:

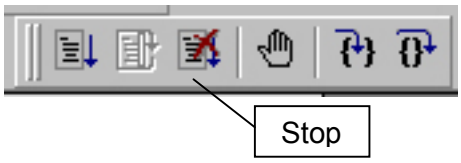


This means that the current line

clr P1.0

has already executed. In the Register window, Port1 will change from 0xFF to 0xFE indicating that bit 0 (P1.0) has been reset.

If you continue to press F11, the program execution will loop to the beginning and start over because **AJMP LABEL1** statement (jump to LABEL1). To stop debugging, select Debug menu and select Stop Debugging. Or click the Stop Debugging button:



When debugging is stopped, all debugging related windows such as Register window, Memory window and Variable window are automatically closed.

You can obtain more detailed help on how to use Micro-IDE by selecting Help Topics and Help Pages on the Web under the Help menu.

Appendix A: 8051 Assembly Language Instructions

Transfer (Move) Instructions

Syntax:

MOV destination, source destination = source

There are 6 basic types:

Instruction	Description
MOV A, byte	Move byte to accumulator
MOV byte, A	Move accumulator to byte
MOV Rn, byte	Move byte to register of current bank
MOV direct, byte	Move byte to internal RAM
MOV @Rn, byte	Move byte to internal RAM with address contained in Rn
MOV DPTR, data16	Move 16-bit data into data pointer

Stack Instructions

Instruction	Description
PUSH byte	Increment stack pointer, move byte on stack
POP byte	Move from stack to byte, decrement stack pointer

Exchange Instructions

Instruction	Description
XCH A, byte	Exchange accumulator and byte
XCHD A, byte	Exchange low nibbles of accumulator and byte

Arithmetic Instructions

Instruction	Description
ADD A, byte	Add accumulator to byte, put result in accumulator
ADDC A, byte	Add with carry
SUBB A, byte	Subtract with borrow
INC A	Increment accumulator
INC byte	Increment byte in memory
INC DPTR	Increment data pointer
DEC A	Decrement accumulator
DEC byte	Decrement byte in memory
MUL AB	Multiply accumulator by b register
DIV AB	Divide accumulator by b register
DAA	Decimally adjust the accumulator

Logic Instructions

Instruction	Description
ANL A, byte	AND accumulator with byte, put result in accumulator
ANL byte, A	AND byte with accumulator, put result in byte
ANL byte, #constant	AND byte with constant, put result in byte
ANL C, bit	AND carry with bit, put result in carry
ORL A, byte	OR accumulator with byte, put result in accumulator
ORL byte, A	OR byte with accumulator, put result in byte
ORL byte, #constant	OR byte with constant, put result in byte
ORL C, bit	OR carry with bit, put result in carry
XRL A, byte	XOR accumulator with byte, put result in accumulator
XRL byte, A	XOR byte with accumulator, put result in byte
XRL byte, #constant	XOR byte with constant, put result in byte

Bit Manipulation Instructions

Instruction	Description
CLR A	Clear all bits of accumulator
CLR byte	Clear all bits of byte
CLR Rn	Clear all bits of register in current bank
CLR @Ri	Clear all bits of byte pointed to by register
CLR C	Clear carry bit
CLR bit	Clear a bit-addressable RAM location or SFR
RL A	Rotate left accumulator
RLC A	Rotate left accumulator through carry
RR A	Rotate right accumulator
RRC A	Rotate right accumulator through carry
SWAP A	Swap the nibbles of accumulator
CPL C	Complement carry bit
CPL bit	Complement a bit-addressable RAM location or SFR
CPL A	2’s complement A
SETB C	Set carry bit
SETB bit	Set a bit-addressable RAM location or SFR

Jump Instructions

Instruction	Description
SJMP <relative addr>	Short jump up to 127 bytes forward or 128 bytes back
LJMP <address 16>	Long jump to a 16-bit address
AJMP <address 11>	Absolute jump to within 2K of program memory
JMP @A + DPTR	Long indexed jump

Conditional Jump Instructions

Instruction	Description
JZ <relative addr>	Jump if accumulator is zero
JNZ <relative addr>	Jump if accumulator is non-zero
JC <relative addr>	Jump if carry is set
JNC <relative addr>	Jump if carry is cleared
JB <bit>, <rel addr>	Jump if bit is set
JNB <bit>,<rel addr>	Jump if bit is cleared
JBC <bit>, <rel addr>	Jump if bit is set, clear bit
CJNE A, direct, <rel addr>	Compare accumulator and memory, jump if not equal
CJNE A, #data <rel addr>	Compare accumulator and data, jump if not equal
CJNE Rn, #data <rel addr>	Compare Rn and data, jump if not equal
CJNE @Rn, #data <rel addr>	Compare data and the byte pointed to by Rn, jump if not equal
DJNZ Rn, <rel addr>	Decrement Rn and jump if not zero
DJNZ direct, <rel addr>	Decrement memory and jump if not zero

Subroutine Instructions

Instruction	Description
ACALL <address 11>	Absolute call to within 2K of program memory
LCALL <address 16>	Long call to a 16-bit address
RET	Return from subroutine
RETI	Return from interrupt