## Emu8086使用教程

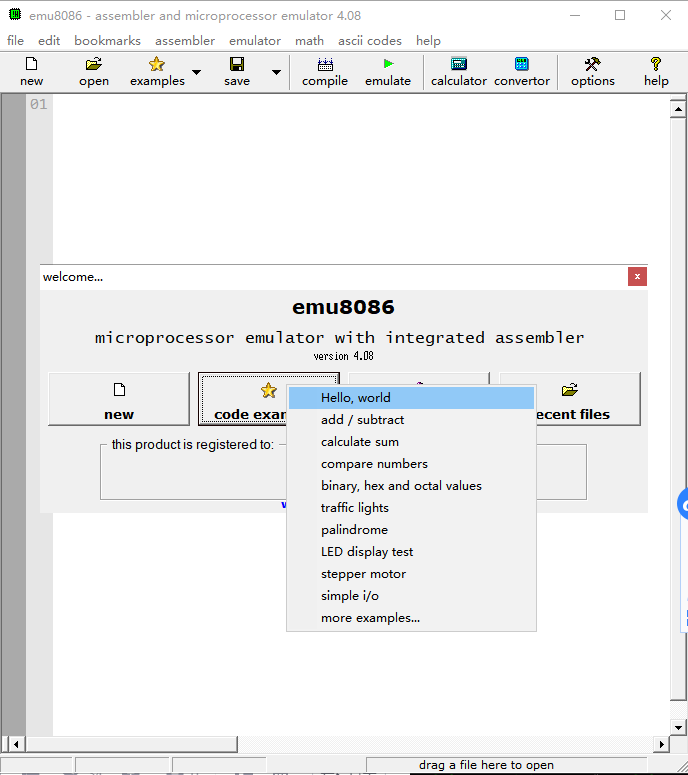
1. 安装

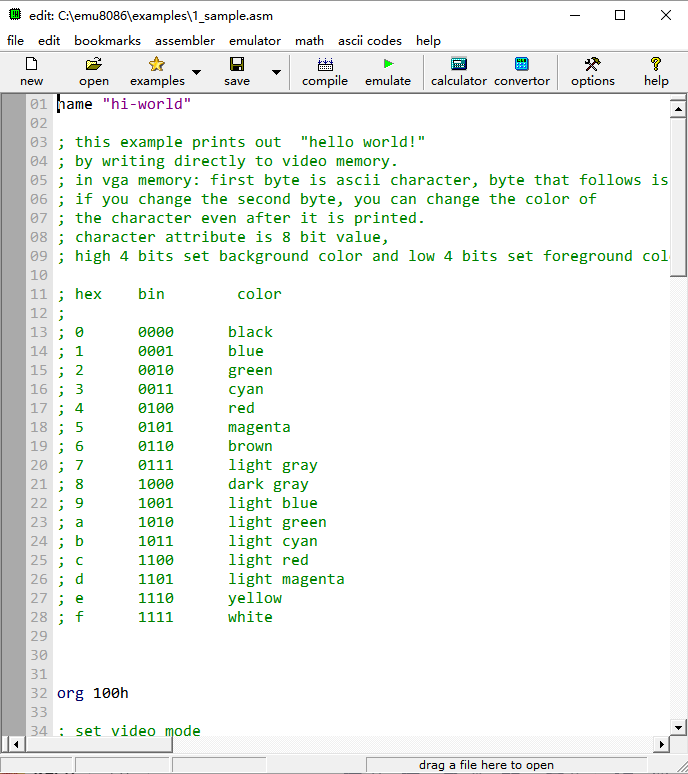
在将安装包解压后，双击Setup.exe就会进入安装指引，按着提示点击next就好（当然如果需要改安装路径，记得更改哦）。

1. 示例代码使用

EMU8086提供了许多示例代码，可供学习。

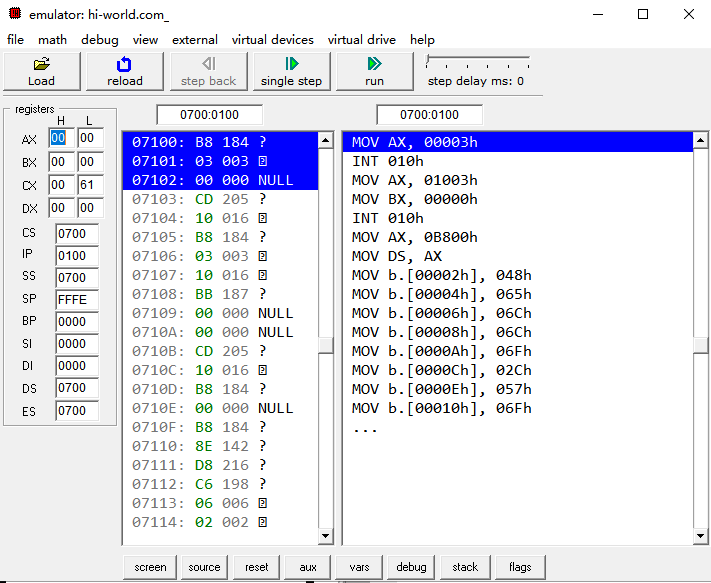
1. 打开安装好的EMU8086后会出现下面的界面，此时我们点击code examples选项，在弹出的小窗中，点选Hello，world，进入示例工程。





1. 在源程序编辑窗体中，可以看到打开的源程序。绿色的部分是注释，汇编语言中注释用分号引导。程序用一个循环把所有字符直接写入显存来高亮显示文本“Hello，World！”。
2. 要在模拟器中运行此示例，需要单击emulate按钮(或按快捷键F5)。如果不存在语法错误，源程序会被汇编、链接并生成可执行文件，然后保存可执行文件到c:\emu8086\ MyBuild。模拟器会自动将其加载到内存中，接着会出现以下两个弹窗：模拟器窗体、源程序窗体。

下面是模拟器窗体，已经加载了可执行程序，定位在程序要执行的第一条指令，等待执行。

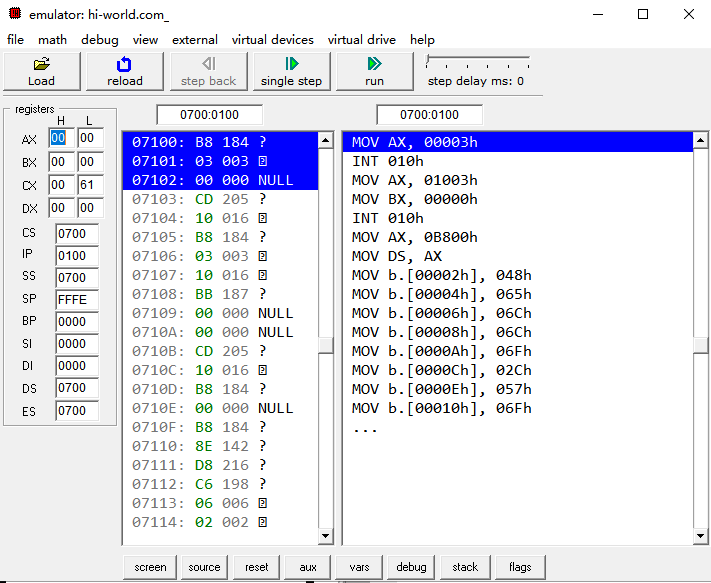


反汇编

内存

寄存器

可通过上面一排按钮控制程序的执行过程。



Load：加载另一个可执行程序。

reload：重新加载当前可执行程序，并定位在第一条指令处等待执行。

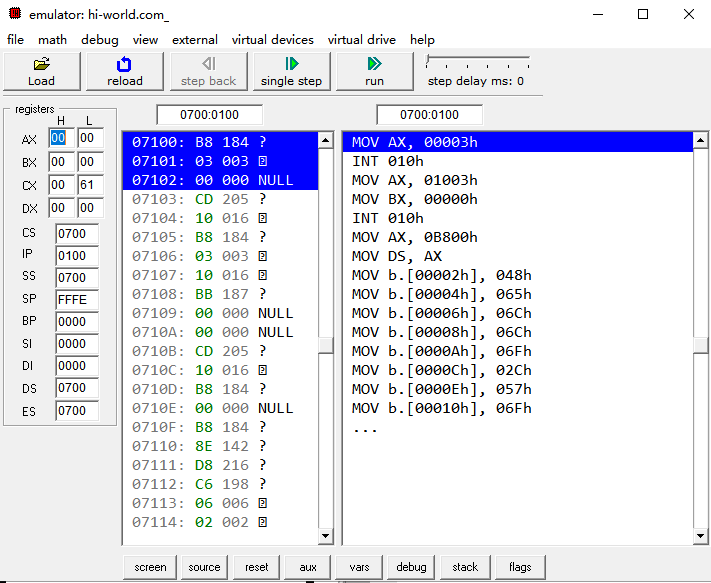
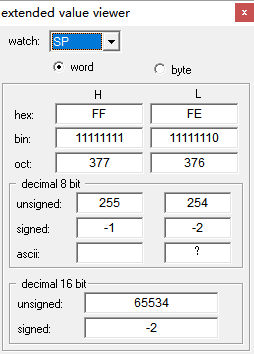
step back：单步退回上一条指令

single step：单步执行下一条指令

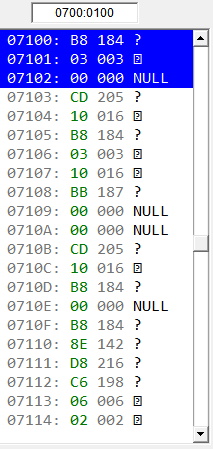
run：连续执行

step delay ms：指令执行后延迟时间（毫秒），可降低模拟执行速度

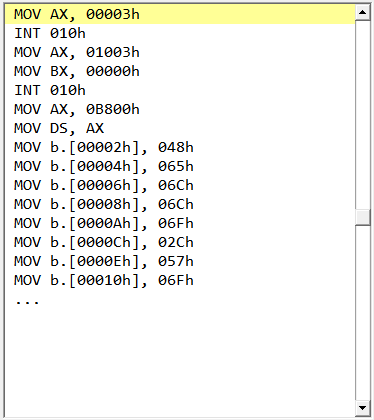
模拟器左边显示所有寄存器的当前值。双击当前值可以显示该寄存器的详细信息。

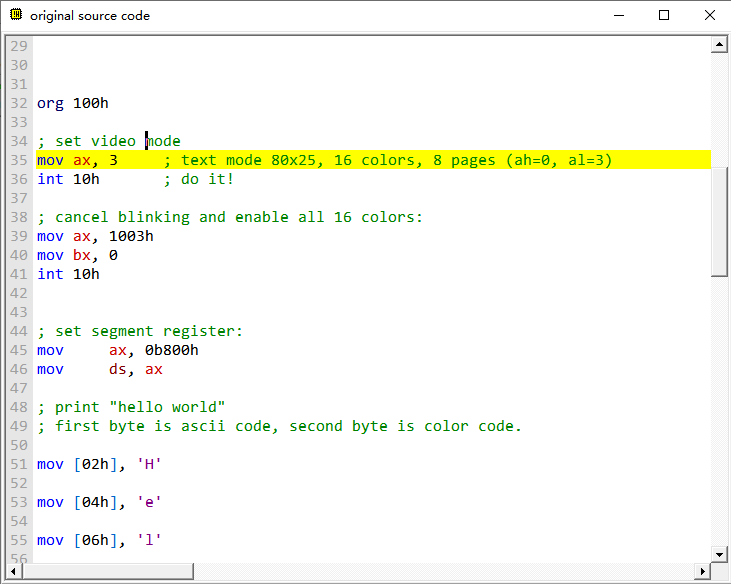
模拟器中间显示加载可执行程序后各个存储单元的值。上面给出了当前显示区域的逻辑地址，下面每行显示一个存储单元，左边是十六进制内存物理地址，冒号后是该单元的值（分别是其十六进制、十进制、字符形式）。蓝色多行部分是CS:IP指向的待执行指令。



模拟器右边显示从当前内存单元开始的内存区域内容的反汇编得到的指令序列。

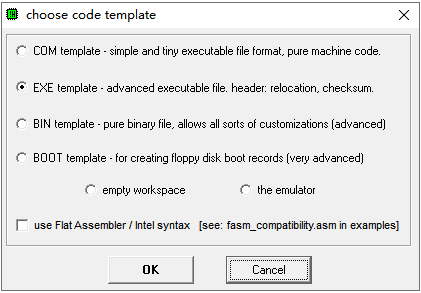


下面是源程序窗体，其可执行程序已模拟加载到内存。



源程序窗体和模拟器窗体是联动的。如果在模拟器窗体中点击内存单元或反汇编指令，在源程序窗体中该指令会被选中，反之在源程序窗体中点击每条语句，在模拟器窗体中该指令的内存单元和反汇编指令也被选中。

1. 程序调试。可以单击single step 按钮(或按快捷键F8)，一次一个指令地执行代码，观察寄存器和模拟器屏幕的变化。您也可以单击step back 按钮(或按快捷键F6)，看看撤销这些更改时会发生什么。
2. 可以单击run按钮，快速执行完示例程序，并将结果显示在模拟器屏幕上。
3. 新建一个源程序。
4. 单击new按钮新建源程序。



选择代码模板。

1. COM模板：COM文件以COM为扩展名，程序只能设置一个段，且不建立堆栈段，程序的长度必须少于64K字节，程序必须预留100H空间，程序默认从偏移地址100H处开始执行。
2. EXE模板：常用EXE文件允许建立若干不同名的代码段、数据段、堆栈段或附加段，程序的长度仅受当前内存可用空间的限制，程序的入口随应用而定，由主模块的END语句指出程序入口的起始标号，并至少有一个具有STACK属性的堆栈段。
3. 如果程序中只是一个指令序列，不需要生成可执行程序（EXE或COM文件），可以选择 empty workspace。
4. 编辑源程序、模拟器中执行调试。
5. **EMU8086和微软宏汇编MASM的语法区别**

EMU8086兼容MASM，但存在个别不同：

1. EMU8086中字节类型变量不能用“？”实现变量的未初始化（MASM中可以）。

X db 1, 2, ?

Y db 5 dup (?)

以上语句在EMU8086中皆语法错误，但其他类型变量的定义无此限制，如：

x dw ?, 1, 2

y dd 3 dup(?)

上面两条语句在EMU8086并会不报错。

1. 段定义语法。EMU8086中伪指令ends前可以省略段名。如：

MASM中定义一个名称为data的段：

data segment

;段的内容

data ends

EMU8086中定义一个名称为data的段：

data segment

;段的内容

ends

1. 过程定义中伪指令endp前可省略过程名，如：

code1 segment

main proc far

push ds

mov ax,0

push ax

call sub1

mov ax,1

ret

endp

sub1 proc near

mov ax,2

ret

endp

ends

end main

1. EMU8086不支持过程的段间直接调用。
2. **过程的段内直接调用（过程名表示被调用过程的入口地址），示例如下：**

code segment

main proc far

push ds

mov ax,0

push ax

call sub1 ;调用过程sub1，段内直接调用方式

ret

endp

sub1 proc near ;定义过程sub1

ret

endp

ends

end main

1. **过程的段内间接调用（采用过程名以外的其他寻址方式表示被调用过程的入口地址），**

示例1：call指令中过程入口地址在寄存器中，采用寄存器寻址方式给出

code segment

main proc far

push ds

mov ax,0

push ax

mov bx, offset sub1 ;bx=过程sub1的入口地址

call bx ;调用过程，过程入口地址的寻址方式为寄存器寻址方式

ret

endp

sub1 proc near

ret

endp

ends

end main

示例2：call指令中过程入口地址在内存中，采用寄存器间接寻址方式（其他寻址方式与此类似）

data segment

addr dw offset sub1 ;变量addr的值是过程sub1的入口地址

ends

code segment

main proc far

push ds

mov ax,0

push ax

mov ax,data

mov ds,ax

mov bx,offset addr ;bx指向变量addr

call [bx] ;过程入口地址在内存中，采用寄存器间接寻址方式给出

ret

endp

sub1 proc near

ret

endp

ends

end main

1. **过程的段间直接调用（过程名表示被调用过程的入口地址），在MASM中程序无误，但EMU8086不支持过程远调用，会提示错误，即便不提示错误，此call指令的汇编结果依然是错误的。示例如下：**

code1 segment

main proc far

push ds

mov ax,0

push ax

call far ptr sub1 ;调用其他段code2中定义的过程sub1，段间直接调用方式

ret

endp

ends

code2 segment

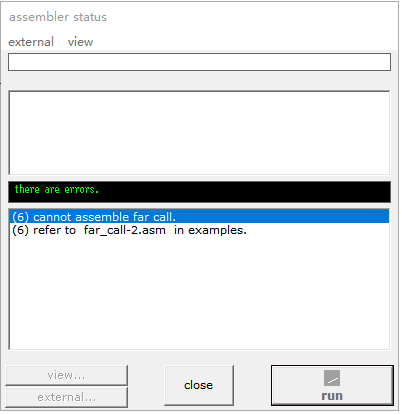
sub1 proc far ;定义过程sub1

ret

endp

ends

end main



1. **过程的段间间接调用（采用过程名以外的其他寻址方式表示被调用过程的入口地址），**

示例：call指令中过程入口地址在内存中，采用寄存器间接寻址方式（其他寻址方式与此类似）

data segment

addr dd ？ ;addr的值是远调用过程的入口地址（偏移地址、段地址）

ends

code1 segment

main proc far

push ds

mov ax,0

push ax

mov ax,data

mov ds,ax

mov word ptr addr,offset sub1 ;过程sub1的偏移地址放变量addr的低字中

mov word ptr addr+2,seg sub1 ;过程sub1的段地址放变量addr的高字中

call dword ptr addr ;远调用过程sub1，段间直接调用方式，注意dword ptr

ret

endp

ends

code2 segment

sub1 proc far ;定义过程sub1

ret

endp

ends

end main