1.编写ur1.c,生成ur1.exe.

main()

{

\_AX=1;

\_BX=1;

\_CX=1;

}

用debug 调试,却发现找不到main函数里的代码。

我思考main 函数在代码段中，但是代码段也可能有好几个，debug开始里

的CS存放的是我们的main函数代码段吗？要找到main 函数必须找到main函数的段地址和偏移地址

我就按照课本编写了一个打印main的偏移地址的函数。

main()

{

\_AX=1;

\_BX=1;

\_CX=1;

printf("%x\n",main);

}

果然可以打印出来，我的main函数的偏移地址为01fah,然后我继续用debug调试，

用试试心态看main函数的段地址是否在当前的CS中 然后用 g cs:0f1a 就找到main函数的程序，发现main函数的段地址和准备阶段的代码所在的

段地址是一样的。

我又编写一个函数证明main函数的段地址是和准备阶段的代码所在的段地址是一样的

main()

{

int a;

\_AX=1;

\_BX=1;

\_CX=1;

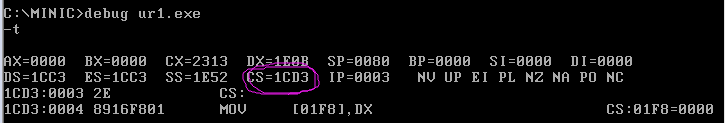
a=\_CS;

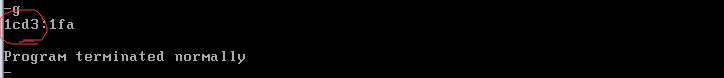
printf("%x:",a);

printf("%x\n",main);

}

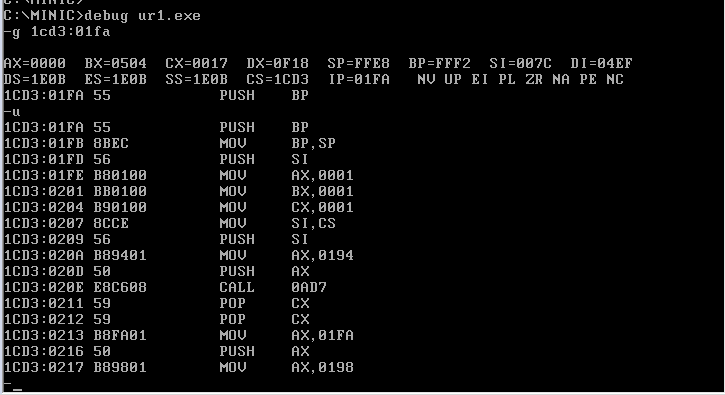
在debug 调试如下





得到我们main函数的偏移地址和段地址

重新debug 通过 g 1cd3:1fa 找到main函数：



但是 我直接运行ur1.c 结果打印让我大吃一惊！

QQ截图20160514225909

怎么会出现不一样的结果呢？

我想了好几种方法，但最终还是失败，没有找到原因。

对于printf("%x\n",main);为什么能打印出偏移地址？ 其实我们知道函数名是代表个地址。

通过main函数的ret指令，我们可以设想我们以前的子程序，子程序名就是标号，所以main 函数其实也是个子程序。我们的代码从主函数（特殊的子程序）开始，这个由编译环境初始化所决定的（说明我们可以改变初始化程序，定义我们自己的主函数），然后主函数调用其它子函数。

编写如下代码，验证我们的设想

Void f(void);

Main()

{

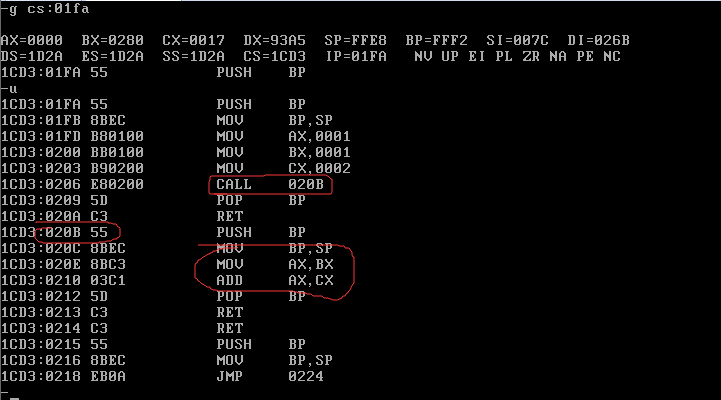
\_AX=1;\_BX=1;\_CX=2;

f();

}

Void f(void)

{ \_AX=\_BX+\_CX;}



通过debug发现main 的调用

QQ截图20160514233737

所以验证了我们设想。

机器码为 E8 DD00

Cpu 执行call 指令时，进行如下操作

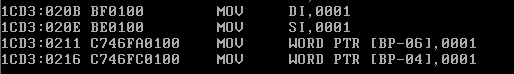
1. (sp)=(sp)-2

((ss)\*16+(sp))=(IP)

1. (IP)=(IP)+16位位移

16位移=标号处的地址-call 指令后的第一个字节的地址即：01fa-(011a+3)=00ddh 与机器码比较我们可以得出 call指令是段内转移，是依据位移转移。

最后我发现 a=1,b=1,c=1,d=1,e=1(int 型)相应的汇编代码为



QQ截图20160516201240

a=1,b=1,c=1(char 型)相应的汇编代码为

QQ截图20160516203101