第　1　章
程序语言简介及汇编语言程序结构

个人信息

姓名：倪一涛

邮箱：838419947@QQ.com

课程群：707631629

课程资料信息：

- 教材：蒋晓捷 32汇编语言程序设计，清华大学出版社
- 参考书： Kip R. Irvine .Intel汇编语言程序设计（第6版）（影印版）

课程工具：

1.汇编器：masm6.15

2.调试器：ollydbg

3.Dev-C++

(https://sourceforge.net/projects/orwelldevcpp/);

内容

1. 课程目标
2. 学习汇编语言的理由
3. 学习汇编语言的方法
4. 程序分析实例
5. 汇编语言程序格式

1. 课程目标

- (自然)语言是用于表达思想的工具，旨在有效地传播思想或与他人交流思想。
- 汇编语言是用于表达解决问题算法的一种工具，旨在让计算机能有效地完成解决问题所需的计算，以及有效地与其他程序员交流思想。
- 目标：能将解决某类问题的算法,正确地用汇编语言表示。

2.为什么要学习汇编语言？

- 写出运行速度更快的代码
- 写出更安全的代码
- 有助于深入理解计算机系统
- 具备以二进制代码为师的能力
- 舒缓学习高级语言不适感

3. 汇编语言学习方法

- 多看代码

  从哪里获取汇编代码？

- 多写代码

  先写简单的、再写复杂的

- 注意细节！

3.1汇编语言要素

自然语言要素：

- 词汇：包含完整语义最小单位
- 句子：根据语法由词汇构成句子
- 文章：根据逻辑关系由句子组成文章

汇编语言要素

- 词汇：操作码、操作数
- 句子：指令, 其格式如下：

  操作码

  操作码　操作数

  操作码　目的操作数，源操作数

操作码：体现指令的语义，即指令所要完成的操作；

操作数：指令操作的对象。包括常数、内存地址、或寄存器。

➢文章:程序(控制结构：顺序、分支、循环)

如何表示？

顺序结构

```
指令1
...
指令n
```

分支结构

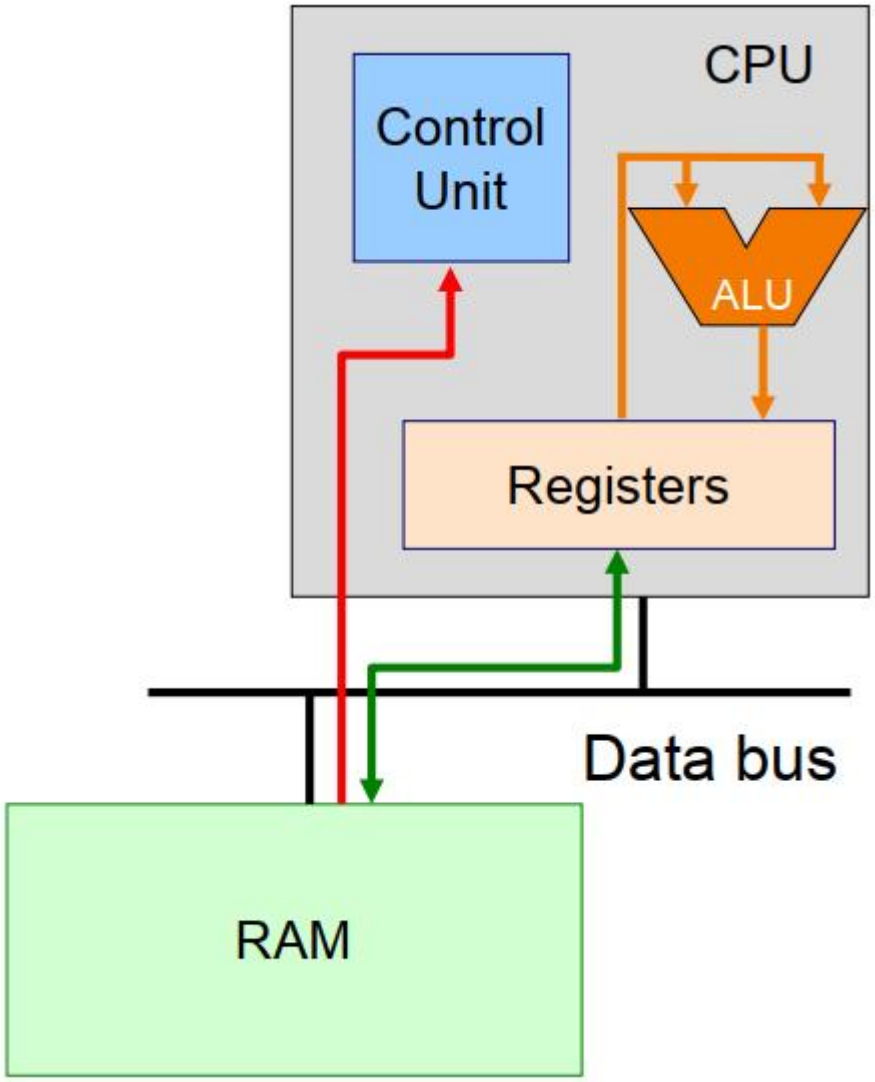
```
if (条件1 ) {
    程序片段1
}else if (条件2){
    程序片段2
}else ...
    ...
else if  (条件k){
    程序片段k
}
```

循环结构

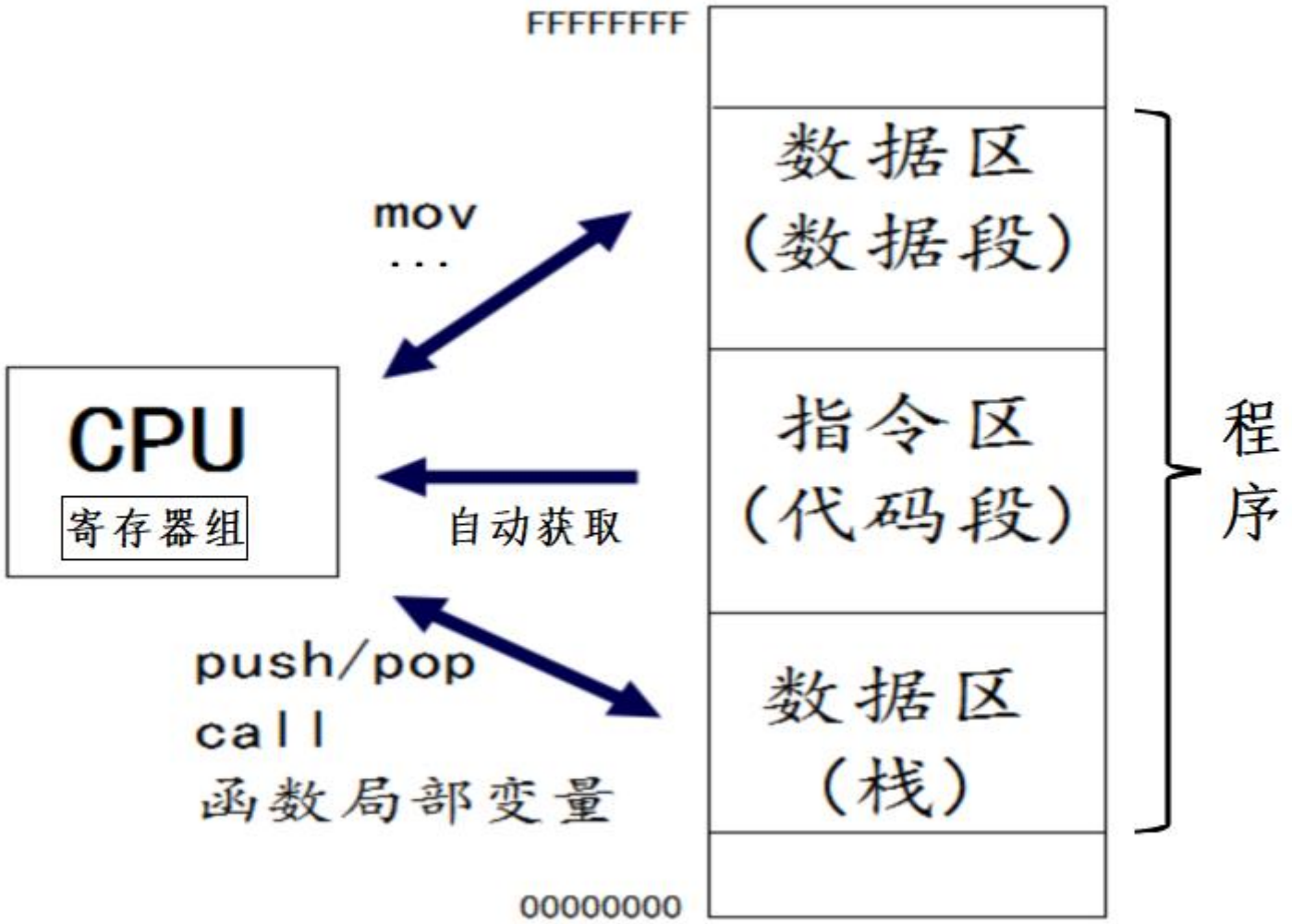
```
while ( 条件 ) {
    程序片段
}

for (i=0; i<n; i++){
    程序片段
}
```

3.3 Von Neumann 计算机模型

- 程序存储在RAM中
- 控制单元控制CPU自动地通过总线从RAM取指令
- 控制单元解释指令、选定与驱动数据流经ALU中对应的逻辑电路。
- 指令执行时，在寄存器与内存之间、或寄存器与寄存器之间传送数据。



程序员视角下的计算机模型



汇编语言位置

自然语言: 显示A乘以B加C的值.

C++: cout << (A * B + C);

汇编语言:

```
mov eax,A
mul  B
add  eax,C
call WriteInt
```

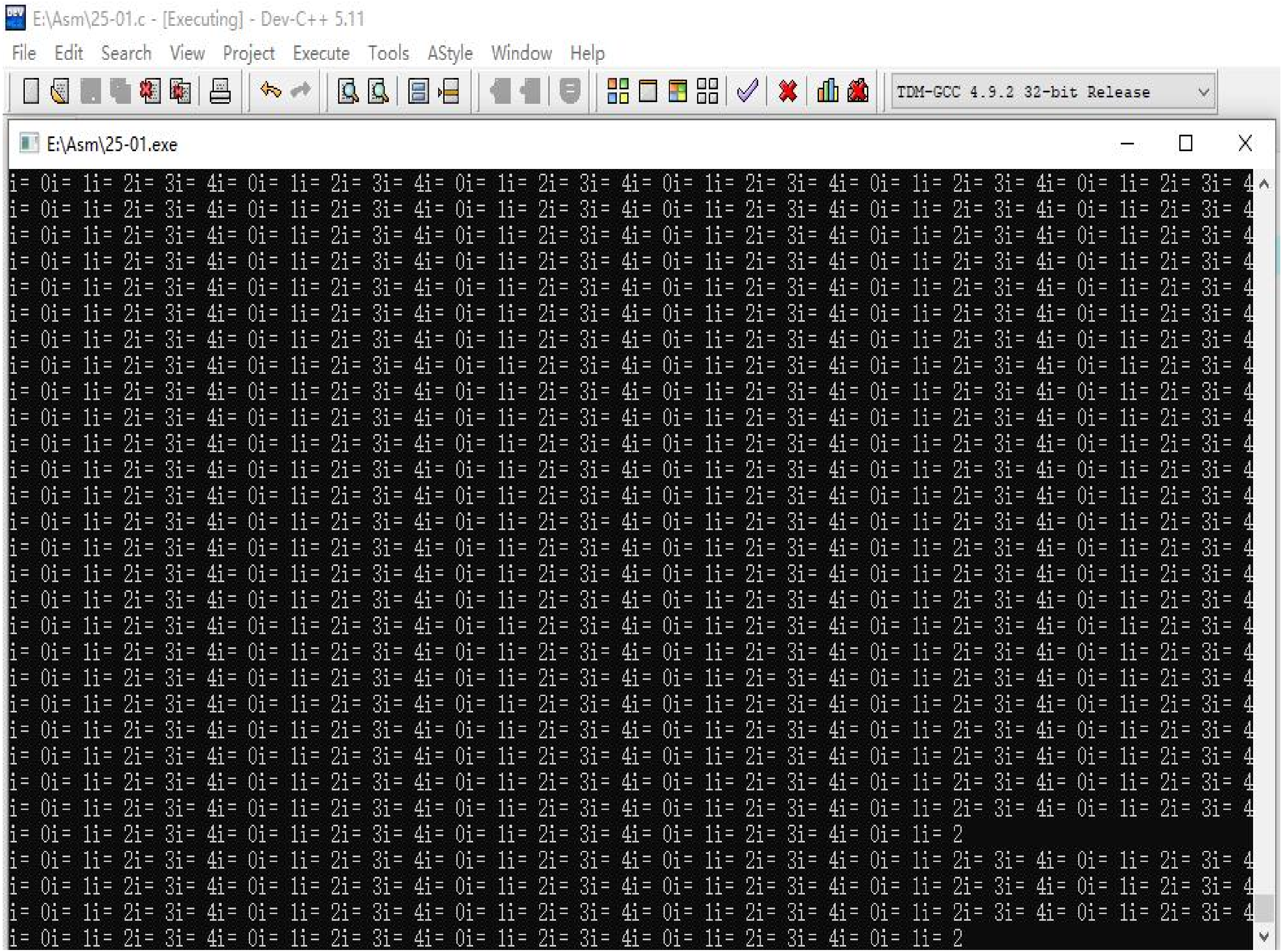
Intel 处理器语言:

A1 00000000
F7 25 00000004
03 05 00000008
E8 00500000

4. 程序分析示例

```
#include <stdio.h>
int main(int argc, char* argv[ ]) {
        int i;
        int a[5];
        for(i=0;i<=5;i++){
                a[i]=0;
                printf("i= %d",i);
        }
        return 0;
}
```

试Dev-C++编译并运行该程序，看运行结果如何？

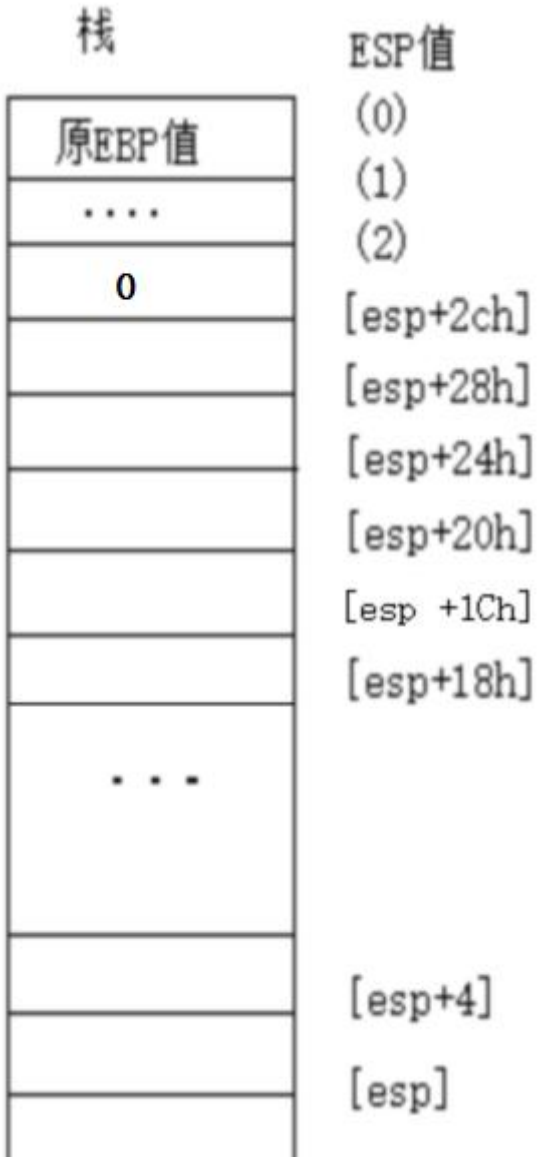


E:\Asm\25-01.c - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 32-bit Release
E:\Asm\25-01.exe
i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4
i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4
i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4
i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4
i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4
i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4
i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4
i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4
i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4
i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4
i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4
i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4
i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4
i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4
i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4
i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4
i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4
i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4
i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4
i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4
i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4
i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4
i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4
i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4
i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4
i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2
i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4
i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4
i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4
i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2i= 3i= 4i= 0i= 1i= 2

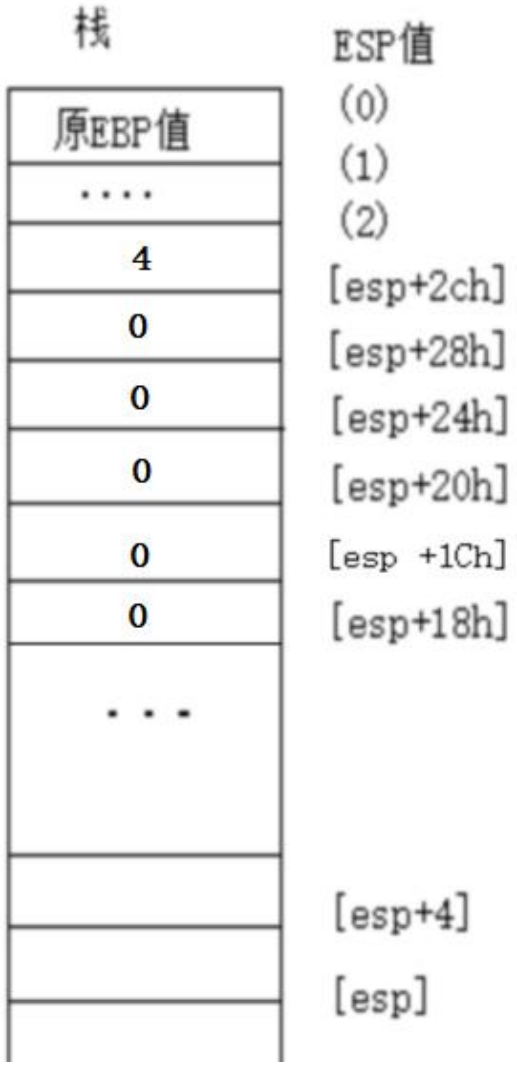
怎么会这样呢？

看看其汇编程序！(IDA Pro 6.8反汇编结果)

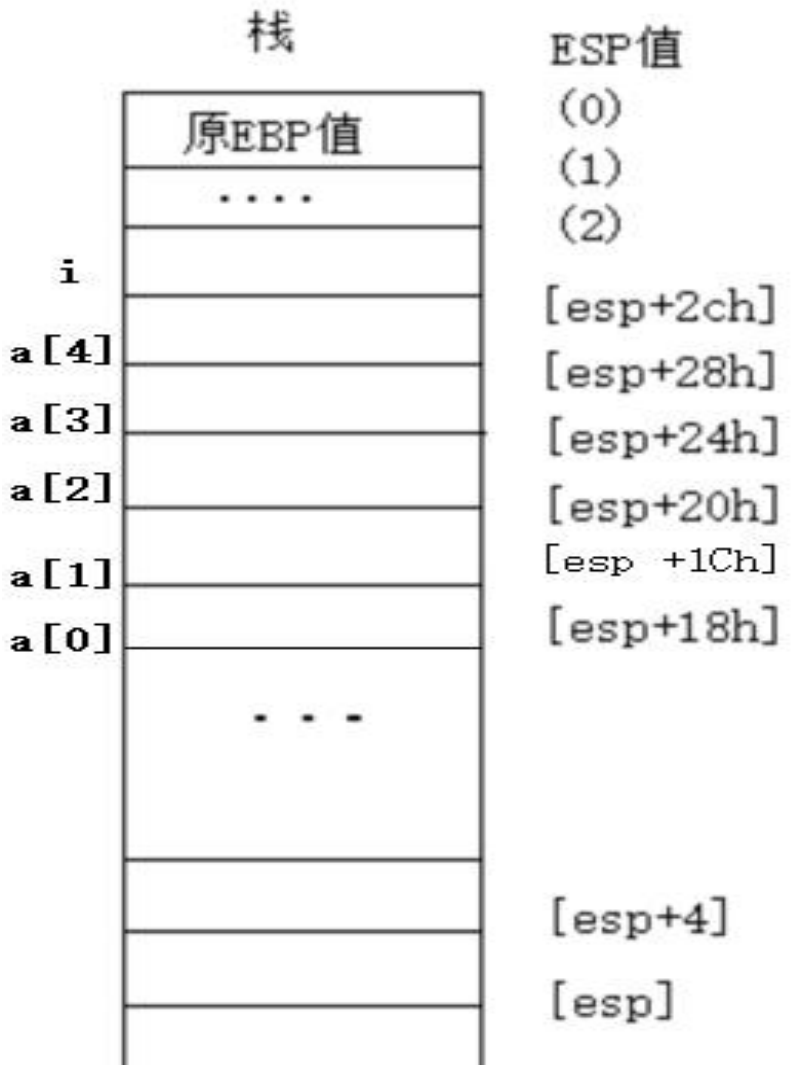
```
_main   proc near
        push    ebp
        mov     ebp, esp
        and     esp, 0FFFFFFF0h
        sub      esp, 30h
        mov     dword ptr [esp+2Ch], 0
        jmp     short loc_40153D
```



```
loc_401518:
      mov     eax, [esp+2Ch]
      mov     dword  ptr [esp+eax*4+18h], 0
      mov     eax, [esp+2Ch]
      mov     [esp+4], eax
      mov     dword ptr [esp],  offset  aID
      call      _printf
      add     dword ptr [esp+2Ch], 1
loc_40153D:
      cmp     dword ptr [esp+2Ch], 5
      jle       short   loc_401518
```



```
        mov     eax, 0
        leave
        retn
_main   endp
```



分析结果：C程序段中变量i与数组a在内存中排列位置如右上所示，且循环结构中超越数组边界访问。从而导致死循环。

思考题：程序分析

```
#include <stdio.h>
#include <string.h>
int main(int argc, char* argv[ ]) {
        char  key[6]=”12345”;
        char  pass[6];
        scanf(“%s”,pass);
        if (strncmp(key, pass, 5)==0){
                printf("Sucess\n”);
        }else
                printf("Failure\n”);
        return 0;
}
```

问题：除了输入12345之外，还有其它值使得上面程序输出“Sucess”吗？

```
#include <stdio.h>
#include <string.h>
int main(int argc, char* argv[ ]) {
        char  pass[6];
        char  key[6]=”12345”;
        scanf(“%s”,pass);
        if (strncmp(key, pass, 5)==0){
                printf("Sucess\n”);
        }else
                printf("Failure\n”);
        return 0;
}
```

问题：该程序与上面程序一样吗？

思考题1的反汇编结果

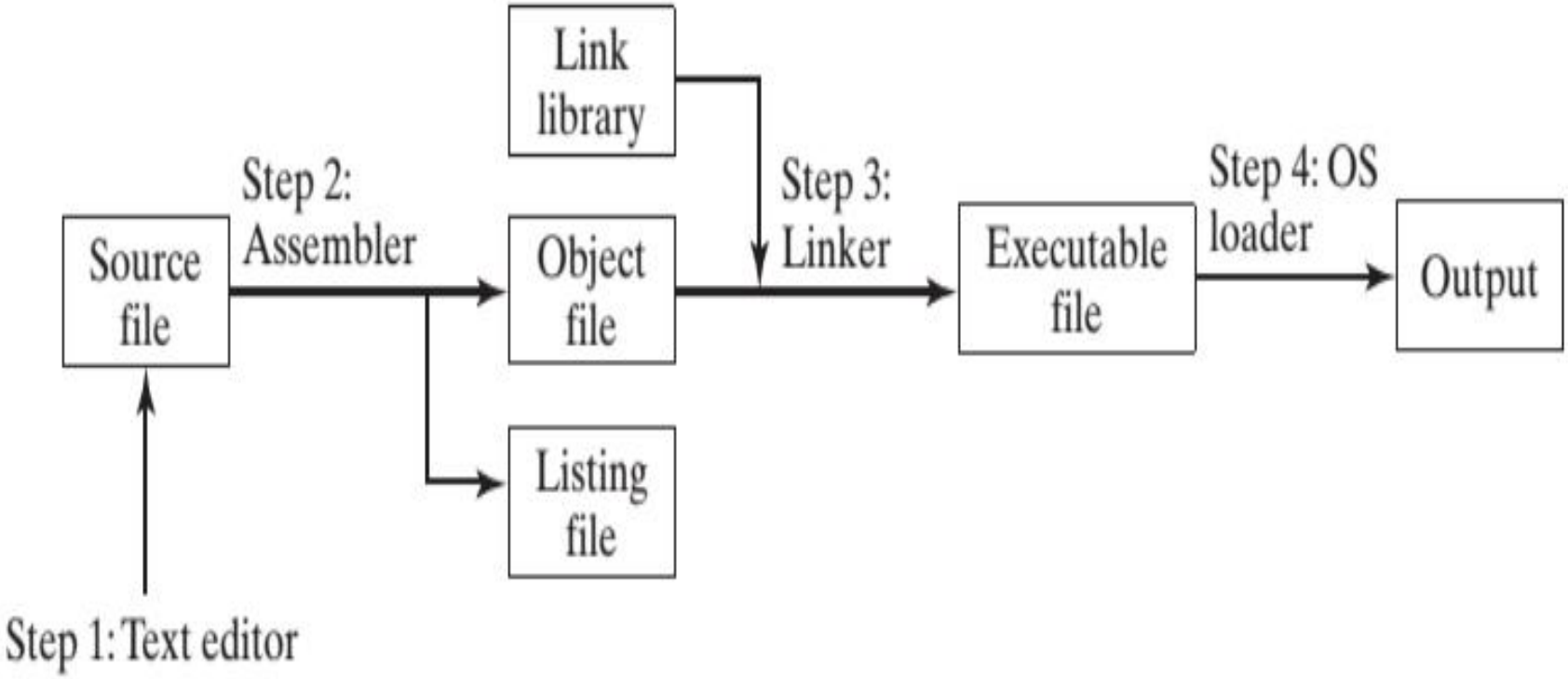
```
_main      proc near
        push    ebp
        mov     ebp, esp
        and     esp, 0FFFFFFF0h
        sub     esp, 20h
        mov     dword ptr [esp+1Ah], 34333231h
        mov     word ptr [esp+1Eh], 35h
        lea     eax, [esp+14h]
        mov     [esp+4], eax
        mov     dword ptr [esp], offset aS  ; "%s"
        call    _scanf
```

```
mov     dword ptr [esp+8], 5
lea     eax, [esp+14h]
mov     [esp+4], eax
lea     eax, [esp+1Ah]
mov     [esp], eax
call    _strncmp
test    eax, eax
jnz     short loc_40155F
mov     dword ptr [esp],  offset aSucess ; "Sucess"
call    _puts
jmp     short loc_40156B
```

```
loc_40155F:
        mov     dword ptr [esp], offset aFailure ; "Failure"
        call    _puts
loc_40156B:
        mov     eax, 0
        leave
        retn
 _main      endp
```

阅读上述汇编代码，pass 与key对应的内存地址是什么？可以得出什么结论？

5.汇编语言程序格式


Link library
Step 2: Assembler
Step 3: Linker
Step 4: OS loader
Source file
Object file
Executable file
Output
Listing file
Step 1: Text editor

程序格式

```
TITLE Program Template (Template.asm)

INCLUDE Irvine32.inc
.data
```

定义全局变量

```
.code

main PROC

    exit
main ENDP

F1 PROC

   ret
F1 ENDP
.....
END main
```

代码区

例1：计算A*B+D，并将结果显示在屏幕

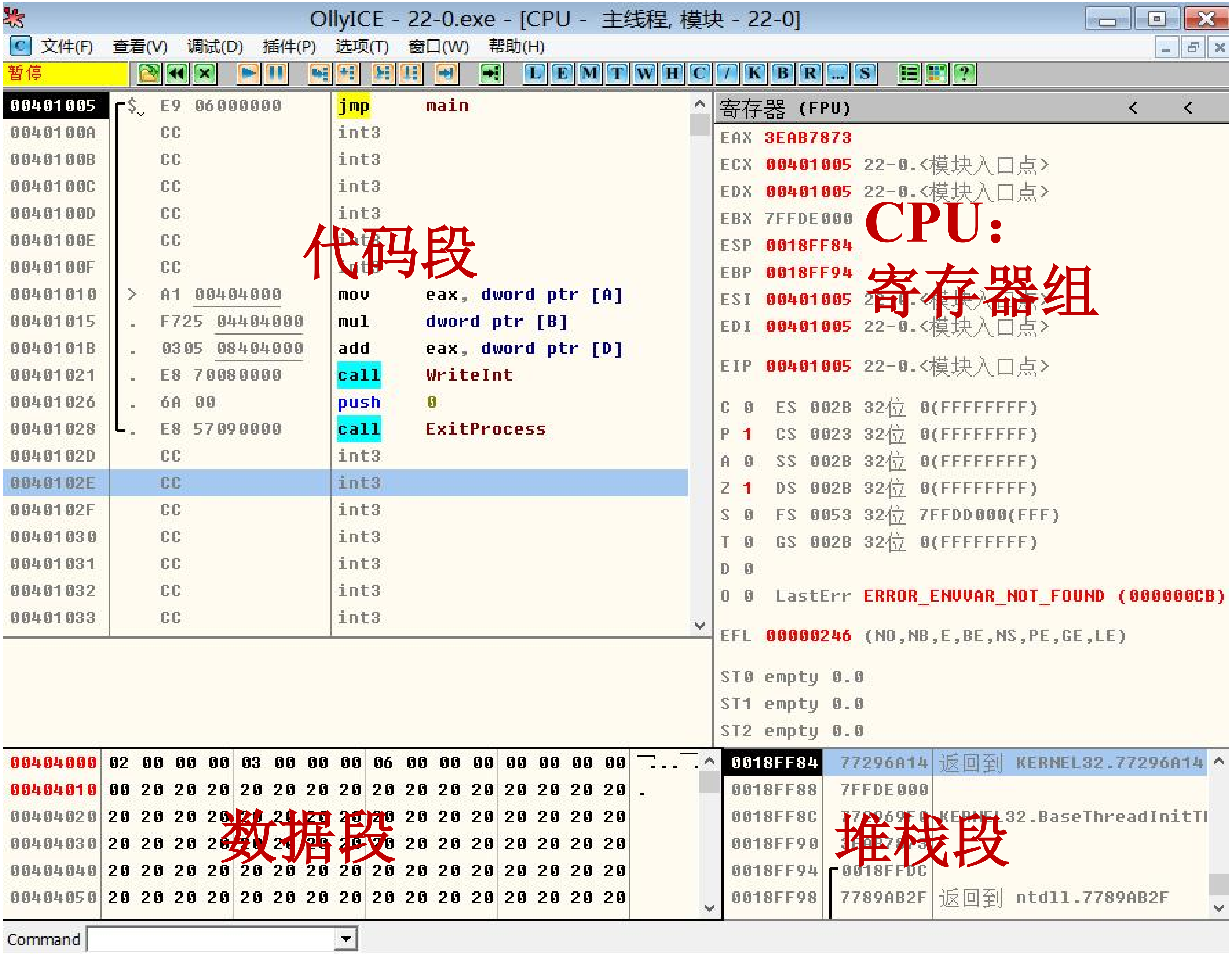
```
 1 INCLUDE Irvine32.inc
 2 .data
 3             A dword 2
 4             B dword 3
 5             D dword 6
 6 .code
 7 main PROC
 8             mov  eax, A
 9             mul  B
10             add  eax, D
11             call writeint
12             exit
13 main ENDP
14 END main
```

```
G:\fzuasm>dir *.asm
 驱动器 G 中的卷是 Experiment
 卷的序列号是 4EF8-443C

 G:\fzuasm 的目录

2021/04/22  10:15             2,894 2021_A3.asm
2022/02/20  11:34               514 22-0.asm
2021/04/12  22:57             1,457 4-2.asm
2021/04/14  00:01             1,209 414.asm
```

```
G:\fzuasm>make32.bat 22-0
```

```
Microsoft (R) Macro Assembler Version 6.15.8803
Copyright (C) Microsoft Corp 1981-2000.  All rights reserved.

 Assembling: 22-0.asm
Microsoft (R) Incremental Linker Version 6.00.8447
Copyright (C) Microsoft Corp 1992-1998. All rights reserved.

 驱动器 G 中的卷是 Experiment
 卷的序列号是 4EF8-443C

 G:\fzuasm 的目录

2022/02/20  11:34               514 22-0.asm
2022/02/20  10:53               159 22-0.asm.bak
2022/02/20  17:59            28,707 22-0.exe
```

```
G:\fzuasm>22-0
+12
G:\fzuasm>
```


OllyICE - 22-0.exe - [CPU - 主线程, 模块 - 22-0]
文件(F) 查看(V) 调试(D) 插件(P) 选项(T) 窗口(W) 帮助(H)
暂停
代码段
CPU：
寄存器组
数据段
堆栈段
寄存器 (FPU)
EAX 3EAB7873
ECX 00401005 22-0.<模块入口点>
EDX 00401005 22-0.<模块入口点>
EBX 7FFDE000
ESP 0018FF84
EBP 0018FF94
ESI 00401005
EDI 00401005 22-0.<模块入口点>
EIP 00401005 22-0.<模块入口点>
00401005 E9 06000000 jmp main
00401010 A1 00404000 mov eax, dword ptr [A]
00401015 F725 04404000 mul dword ptr [B]
0040101B 0305 08404000 add eax, dword ptr [D]
00401021 E8 70080000 call WriteInt
00401026 6A 00 push 0
00401028 E8 57090000 call ExitProcess
LastErr ERROR_ENVVAR_NOT_FOUND (000000CB)
EFL 00000246 (NO,NB,E,BE,NS,PE,GE,LE)
0018FF84 77296A14 返回到 KERNEL32.77296A14
0018FF98 7789AB2F 返回到 ntdll.7789AB2F
Command

可执行程序反汇编结果

内存地址　指令机器码　　汇编指令

```
00401010 | A1 00404000   | mov  eax, dword ptr [00404000]
00401015 | F725 04404000 | mul  dword ptr [00404004]
0040101B | 0305 08404000 | add  eax, dword ptr [00404008]
00401021 | E8 70080000   | call 00401896
00401026 | 6A 00         | push 0
00401028 | E8 57090000   | call 00401984
```

课堂练习

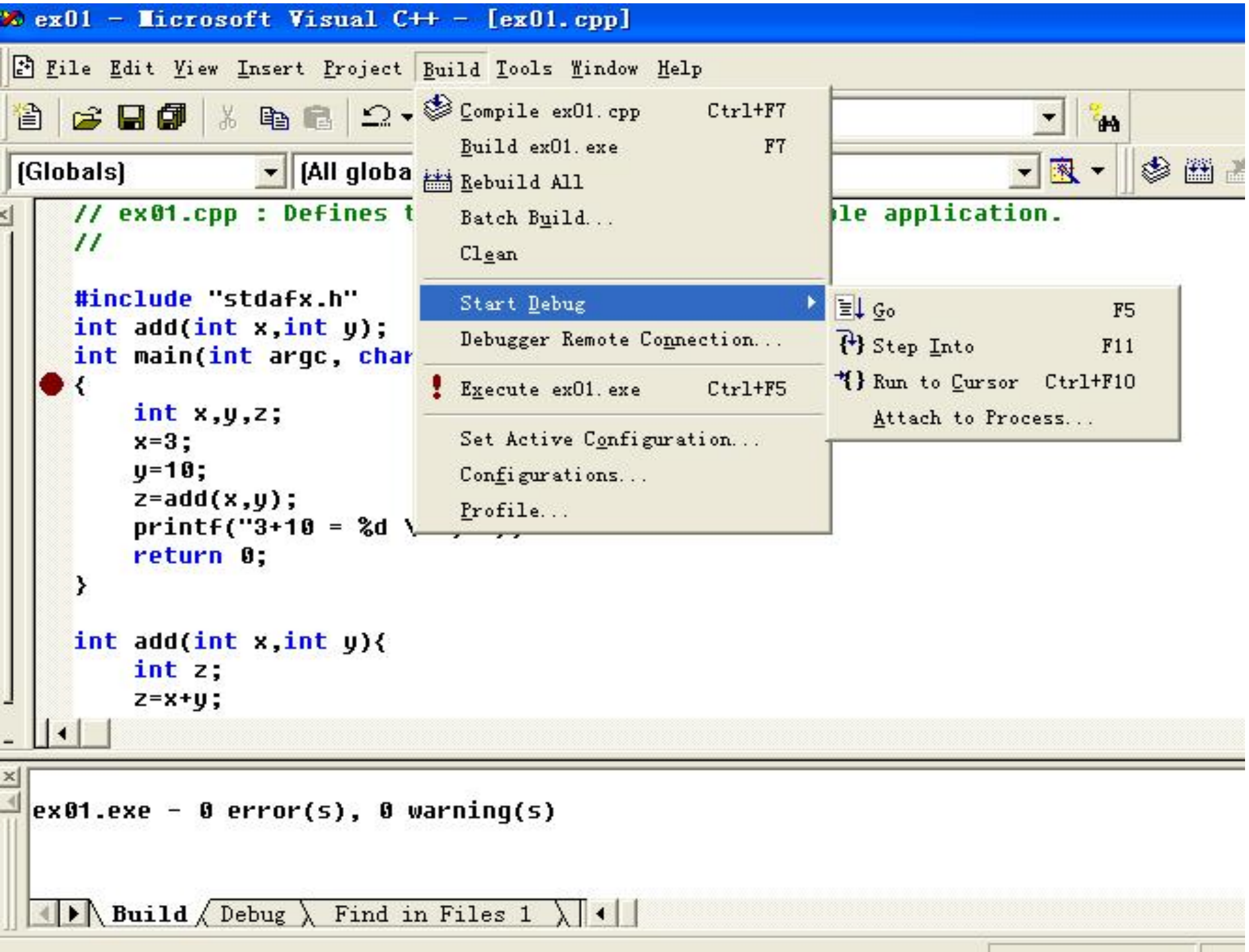
1. 试用汇编语言编写程序，使之在屏幕显示“Hello World!”;

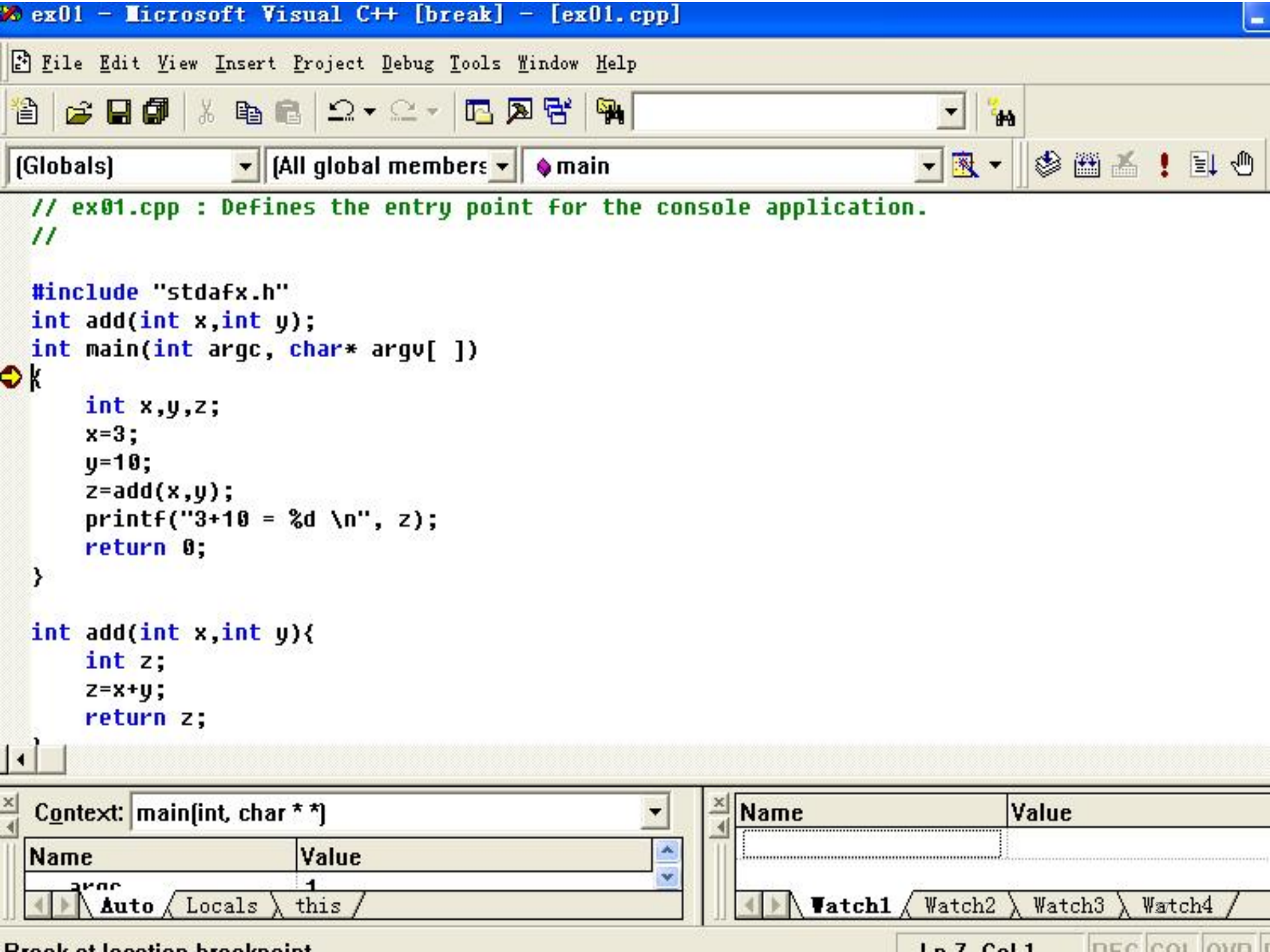
2. 试用汇编语言编写程序实现，使之在屏幕显示内存中3个整数中的最大值;

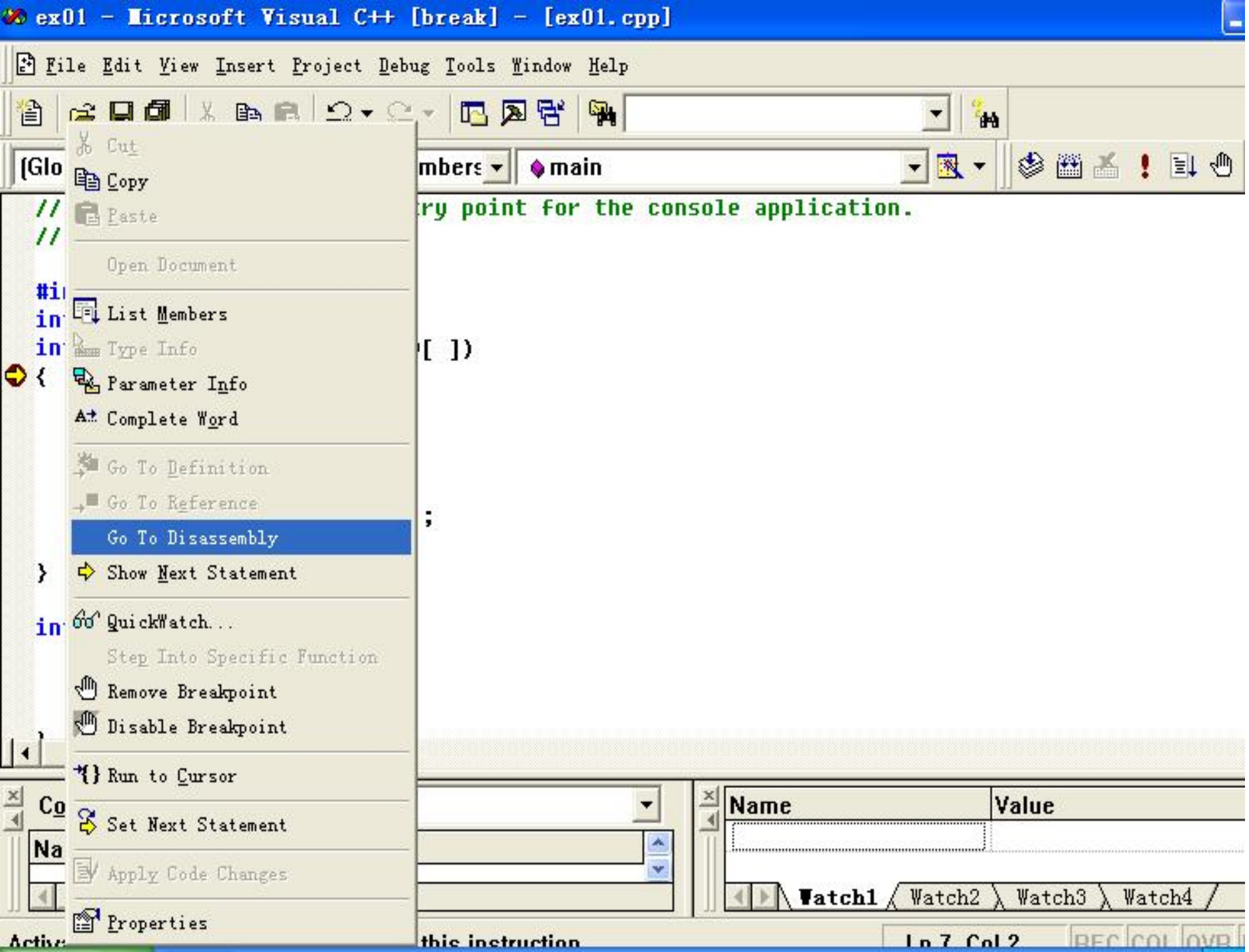
汇编程序设计杂谈一

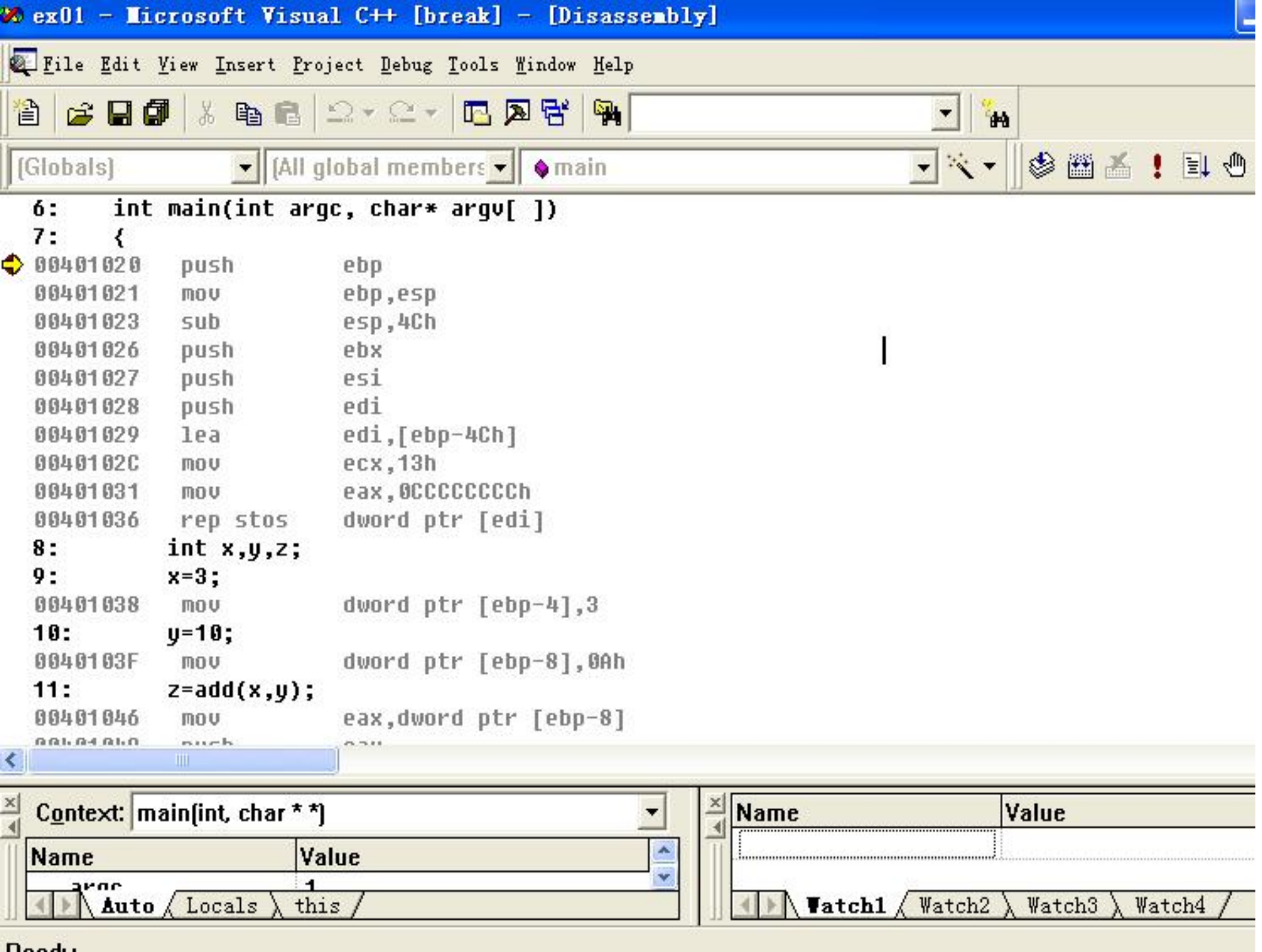
在VC6中查看C语言程序对于的汇编代码

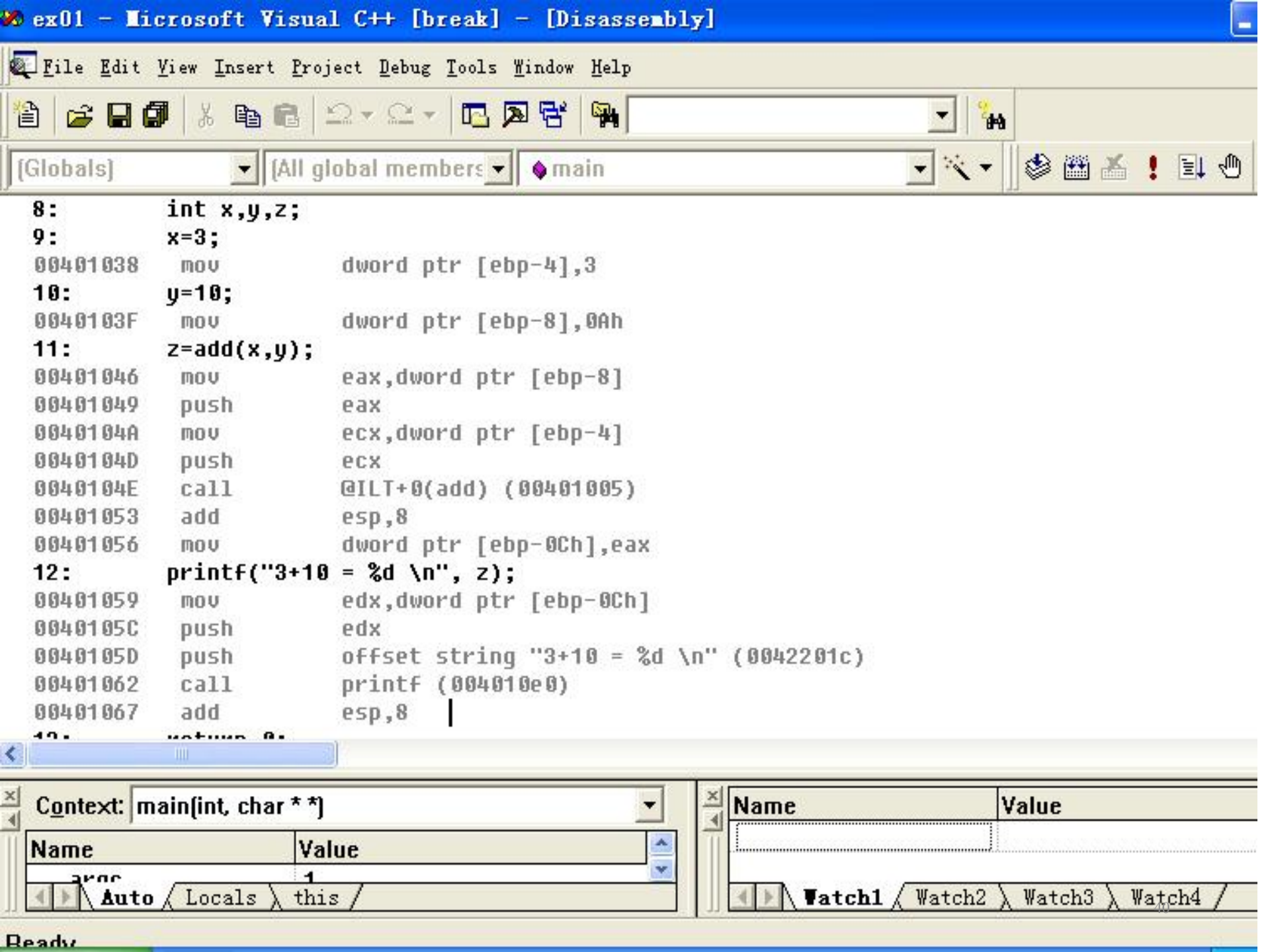
```
int add(int x,int y);
int main(int argc, char* argv[ ])
{
  int x,y,z;
  x=3;
  y=10;
  z=add(x,y);
  printf("3+10 = %d \n", z);
  return 0;
}

int add(int x,int y){
  int z;
  z=x+y;
  return z;
}
```


ex01 - Microsoft Visual C++ - [ex01.cpp]
File Edit View Insert Project Build Tools Window Help
Compile ex01.cpp Ctrl+F7
Build ex01.exe F7
Rebuild All
Batch Build...
Clean
Start Debug
Debugger Remote Connection...
Execute ex01.exe Ctrl+F5
Set Active Configuration...
Configurations...
Profile...
Go F5
Step Into F11
Run to Cursor Ctrl+F10
Attach to Process...
(Globals)
// ex01.cpp : Defines t
le application.
//
#include "stdafx.h"
int add(int x,int y);
int main(int argc, char
{
int x,y,z;
x=3;
y=10;
z=add(x,y);
printf("3+10 = %d \
return 0;
}
int add(int x,int y){
int z;
z=x+y;
ex01.exe - 0 error(s), 0 warning(s)
Build
Debug
Find in Files 1


ex01 - Microsoft Visual C++ [break] - [ex01.cpp]
File Edit View Insert Project Debug Tools Window Help
(Globals)
(All global members
main
// ex01.cpp : Defines the entry point for the console application.
//
#include "stdafx.h"
int add(int x,int y);
int main(int argc, char* argv[ ])
{
int x,y,z;
x=3;
y=10;
z=add(x,y);
printf("3+10 = %d \n", z);
return 0;
}
int add(int x,int y){
int z;
z=x+y;
return z;
Context: main(int, char * *)
Name
Value
Auto
Locals
this
Name
Value
Watch1
Watch2
Watch3
Watch4


ex01 - Microsoft Visual C++ [break] - [ex01.cpp]
File Edit View Insert Project Debug Tools Window Help
Cut
Copy
Paste
Open Document
List Members
Type Info
Parameter Info
Complete Word
Go To Definition
Go To Reference
Go To Disassembly
Show Next Statement
QuickWatch...
Step Into Specific Function
Remove Breakpoint
Disable Breakpoint
Run to Cursor
Set Next Statement
Apply Code Changes
Properties
main
ry point for the console application.
Name
Value
Watch1
Watch2
Watch3
Watch4


ex01 - Microsoft Visual C++ [break] - [Disassembly]
File Edit View Insert Project Debug Tools Window Help
(Globals)
(All global members
main
6: int main(int argc, char* argv[ ])
7: {
00401020 push ebp
00401021 mov ebp,esp
00401023 sub esp,4Ch
00401026 push ebx
00401027 push esi
00401028 push edi
00401029 lea edi,[ebp-4Ch]
0040102C mov ecx,13h
00401031 mov eax,0CCCCCCCCh
00401036 rep stos dword ptr [edi]
8: int x,y,z;
9: x=3;
00401038 mov dword ptr [ebp-4],3
10: y=10;
0040103F mov dword ptr [ebp-8],0Ah
11: z=add(x,y);
00401046 mov eax,dword ptr [ebp-8]
Context: main(int, char * *)
Name
Value
Auto
Locals
this
Name
Value
Watch1
Watch2
Watch3
Watch4


ex01 - Microsoft Visual C++ [break] - [Disassembly]
File Edit View Insert Project Debug Tools Window Help
(Globals)
(All global members
main
8: int x,y,z;
9: x=3;
00401038 mov dword ptr [ebp-4],3
10: y=10;
0040103F mov dword ptr [ebp-8],0Ah
11: z=add(x,y);
00401046 mov eax,dword ptr [ebp-8]
00401049 push eax
0040104A mov ecx,dword ptr [ebp-4]
0040104D push ecx
0040104E call @ILT+0(add) (00401005)
00401053 add esp,8
00401056 mov dword ptr [ebp-0Ch],eax
12: printf("3+10 = %d \n", z);
00401059 mov edx,dword ptr [ebp-0Ch]
0040105C push edx
0040105D push offset string "3+10 = %d \n" (0042201c)
00401062 call printf (004010e0)
00401067 add esp,8
Context: main(int, char * *)
Name
Value
argc
1
Auto
Locals
this
Name
Value
Watch1
Watch2
Watch3
Watch4
Ready

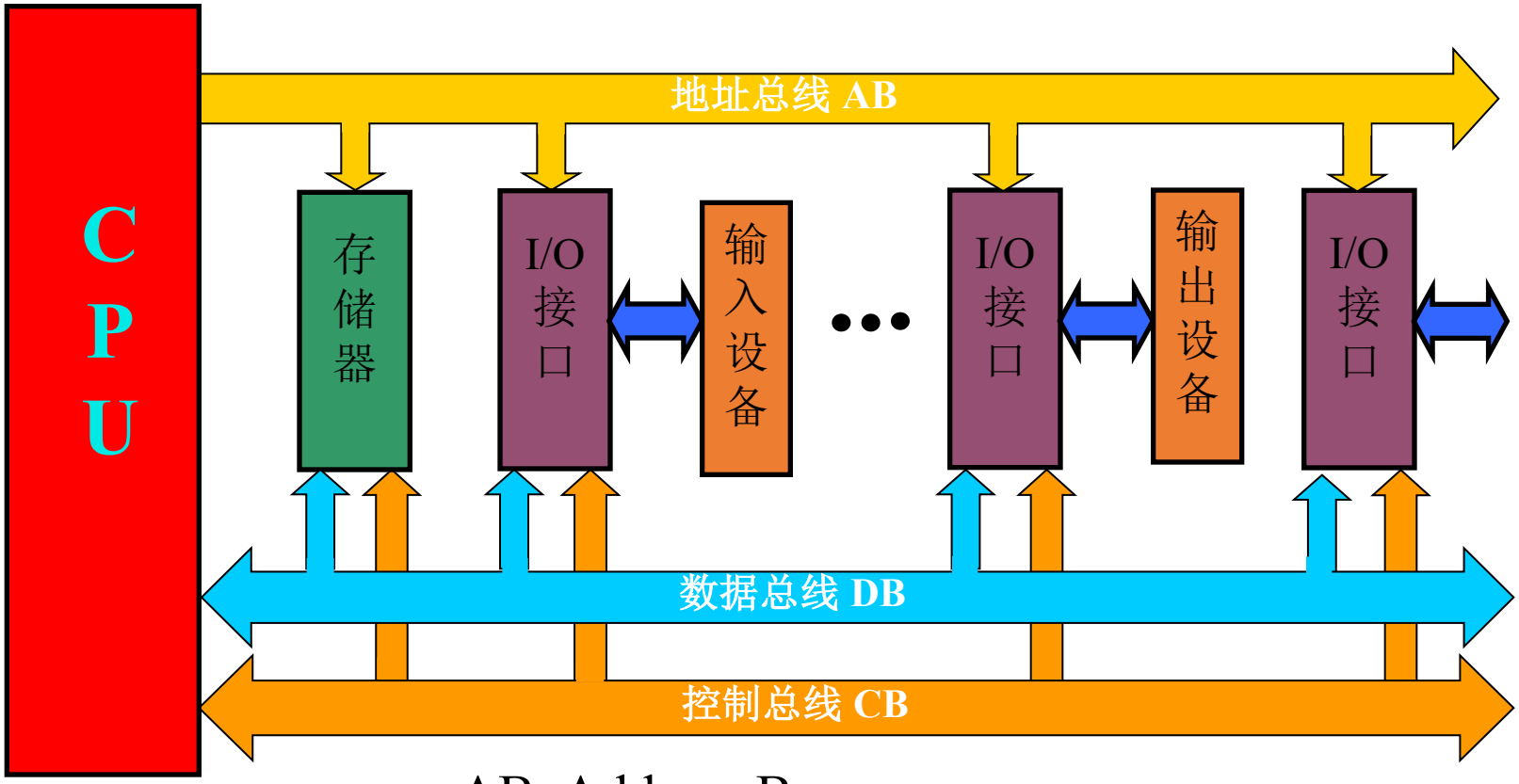
本页之后内容不做要求

二、微型计算机的基本结构

1. 硬件系统

- 微处理器（CPU）
- 存储器
- 输入/输出接口

微型计算机的概念结构



AB: Address Bus
DB: Data Bus
CB: Control Bus

微处理器CPU

- 计算机的核心，包括：
  - 运算器ALU
  - 控制器CU
  - 寄存器组Registers
- 实现运算功能和控制功能

存储器

- 存放**程序**和**数据**的记忆装置
- 用途：存放程序和要操作的各类信息（数据、文字、图像、...）
- 内存：ROM、RAM

  特点：随机存取，速度快，容量小
- 外存：磁盘、光盘、半导体盘、...

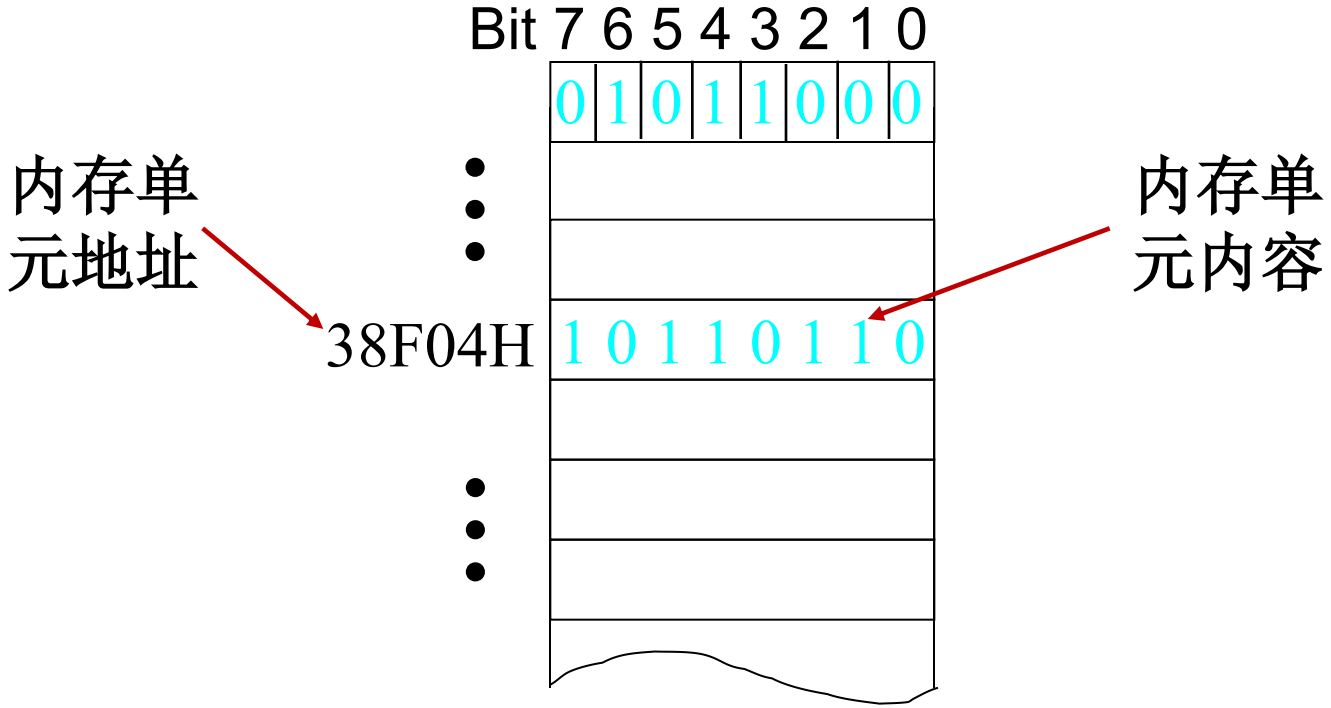
  特点：顺序存取/块存取，速度慢，容量大

有关内存储器的几个概念

- 内存单元的地址和内容
- 内存容量
- 内存的操作

内存单元的地址和内容

- 每个内存单元包含8 bit，
- 对每个内存单元进行编号，**内存单元的编号就称为内存单元的地址。**



内存容量

- 即内存单元的个数，以字节为单位。

内存空间与内存容量的区别

内存容量：实际配置的内存大小。

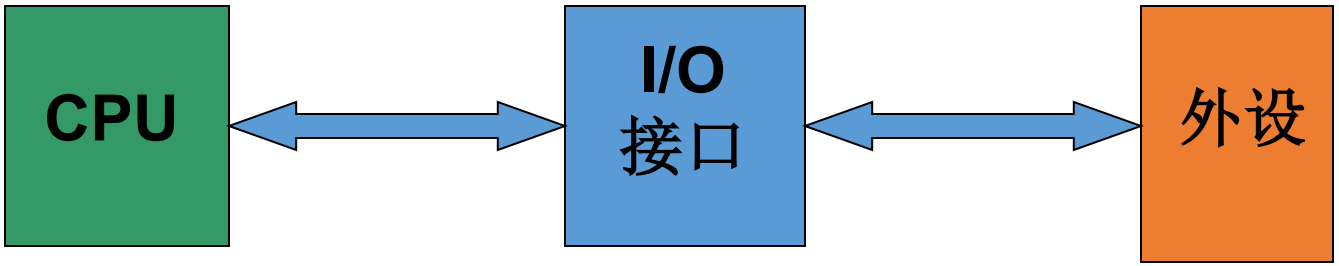
内存空间：又称为存储空间、寻址范围，是指微机的寻址能力，与CPU的地址总线宽度有关。

内存操作

- 读：将内存单元的内容取入CPU，原单元内容不改变；
- 写：CPU将信息放入内存单元，单元中原内容被覆盖；
- 刷新：对CPU透明，仅动态存储器有此操作
- 内存的读写的步骤为：
  - CPU把要读写的内存单元的地址放到AB上
  - 若是写操作， CPU紧接着把要写入的数据放到DB上
  - CPU发出读写命令
  - 数据被写入指定的单元或从指定的单元读出到DB
  - 若是读操作， CPU紧接着从DB上取回数据

输入/输出接口

- 简写为I/O接口，是CPU与外部设备间的桥梁



总线BUS

- 连接多个功能部件的一组公共信号线
  - 地址总线AB：用来传送CPU输出的地址信号，确定被访问的存储单元、I/O端口。地址线的根数决定了CPU的寻址范围。

    CPU的寻址范围 = 2^n，　n-地址线根数
  - 数据总线DB：在CPU与存储器、I/O接口之间数据传送的公共通路。数据总线的条数决定CPU一次最多可以传送的数据宽度。
  - 控制总线CB：用来传送各种控制信号

三、计算机的工作过程*

存储程序计算机—又称为冯•诺依曼型计算机

- 以运算器为核心、以存储程序原理为基础
- 将计算过程描述为由许多条指令按一定顺序组成的程序，即程序是由多条有逻辑关系的指令组成，指令的长度不等（一般为1～4字节）
- 数据和程序均以二进制代码的形式不加区别地存放在存储器中，存放位置由地址指定，地址码也是二进制形式
- 由控制器控制整个程序和数据的存取以及程序的执行

存储程序计算机的工作原理*

- 控制器按预先存放在计算机存储器中的程序的流程自动地连续取出指令并执行之。