# C语言基础01

## 一、使用命令行编译C文件

- 在需要编译文件的文件夹下面打开cmd终端
- 编译c文件：gcc hello.c -o hello1 生成hello1.exe文件
- 运行exe文件：直接输入hello1即可

```
D:\ceshi>
D:\ceshi>gcc hello.c -o hello1

D:\ceshi>hello1
Hello World
D:\ceshi>
```

## 二、C程序编译步骤

- 预处理：宏定义展开，头文件展开，条件编译等，同时将代码中的注释删除，这里并不会检查语法
- 编译：检查语法，将预处理之后的文件编译生成汇编文件
- 汇编：将汇编文件生成目标文件（二进制文件）
- 链接：C语言写的程序是需要依赖各种库的，所以编译之后还需要把库连接到最终的可执行程序中去

### 2.1 分布编译

- 预处理： gcc -E hello.c -o hello.i
- 编译：gcc -S hello.i -o hello.s
- 汇编：gcc -c hello.s -o hello.o
- 链接: gcc hello.o -o hello

预处理：C文件中带#的语句就是预处理指令，预处理指令在预处理的时候就被处理了，#include<stdio.h>包含文件stdio.h(预处理时将stdio.h文件拷贝到预处理文件中),**其实就是头文件展开**，**删除注释**，**宏定义直接被替换掉**，**预处理时不会检查语法错误**，

hello.i文件

```
# 3 "hello.c"
int main()
{
 printf("Hello World");
 return 0;
}
```

编译：将预处理文件编译成汇编文件，**会检查语法错误。**

汇编：将汇编文件编译成二进制文件。

链接：设置运行环境，堆栈等，链接其他库。

## 三、helloworld程序的解释

- #include<stdio.h> 包含stdio.h文件 stdio.h类似于菜单

- 两个斜杠是注释
- 符号与()结合代表这是一个函数
- main()函数也叫做主函数 整个程序中只有一个Main函数，程序从main函数开始执行
- int 代表main函数结束之后返回值类型
- return 结束这个函数，然后返回值，返回值的类型和函数定义时返回值类型一致
- {}里面的是函数体，所有需要执行的代码必须写在{}中

# 四、system库函数

作用：在程序中启动另一个程序　参数：待启动程序的路径名

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
    system("mspaint");// 启动画图板
    printf("Hello World");// 打印终端
    return 0;
}
```

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
    // system启动程序，如果这个程序系统可以找到，不用添加路径
    // 如果环境变量找不到，需要添加路径
    // windows路径以\\ 或者 /
    system("D:\\ceshi\\hello.exe");
    printf("\nHello World");// 打印终端
    return 0;
}
```

# 五、CPU内部结构与寄存器（了解）

## 5.1 64位和32位系统的区别

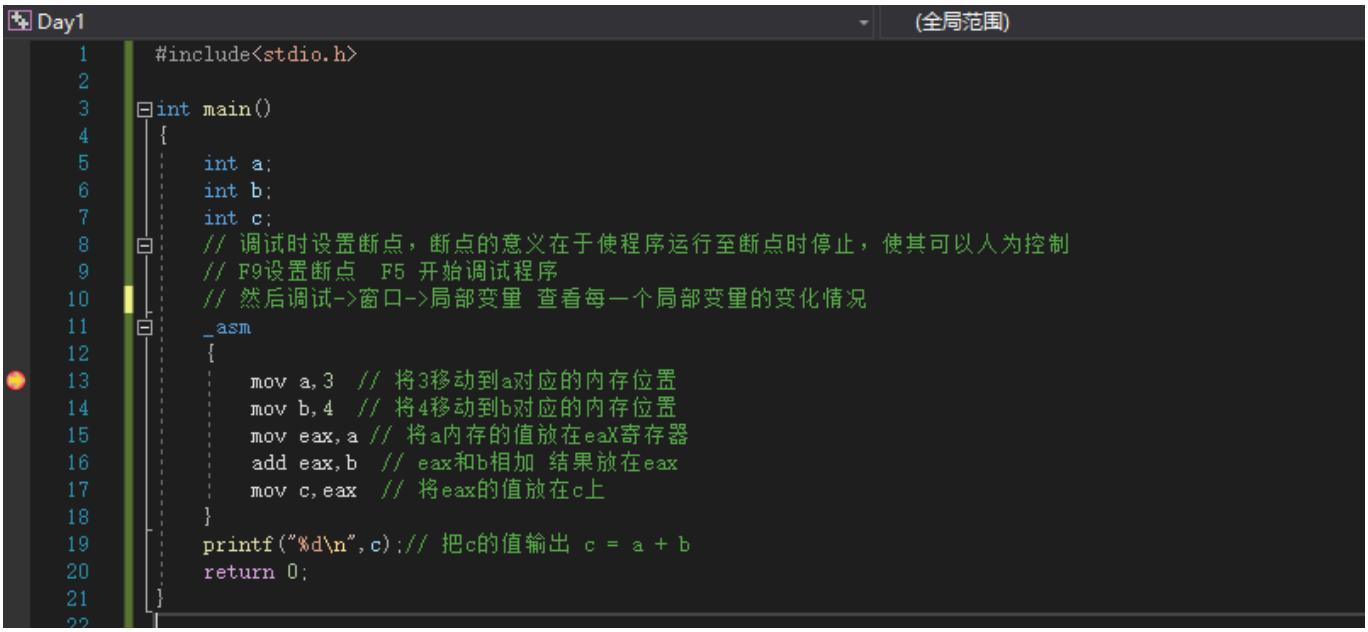
- 寄存器是CPU内部最基本的存储单元
- CPU对外是通过总线来和外部设备进行交互的，总线的宽度是八位，同时cpu的寄存器也是八位，那么这个CPU就叫八位CPU
- 如果总线是32位，寄存器也是32位，那么这个CPU就叫32位CPU
- 有一种CPU内部的寄存器是32位，但是总线是16位，准32位cpu
- 所有的64位CPU可以兼容32位指令，32位要兼容16位的指令，所以在64位的CPU上是可以识别32位的指令
- 在64位的CPU架构上运行了64位软件操作系统，那么这个系统64位
- 在64位的CPU架构上，运行了32位的软件操作系统，那么这个系统是32位
- 64位的软件不可以运行在32位CPU上

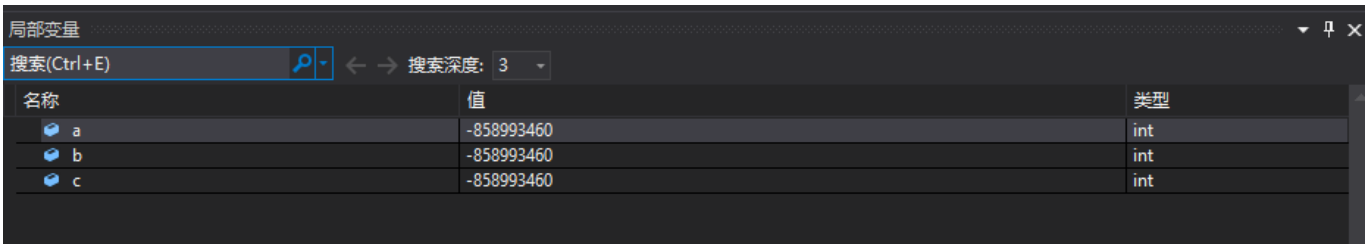
# 六、VS中C语言嵌套汇编代码

```
#include<stdio.h>

int main()
{
    int a;
    int b;
    int c;

    _asm
    {
        mov a,3  // 将3移动到a对应的内存位置
        mov b,4  // 将4移动到b对应的内存位置
        mov eax,a // 将a内存的值放在eaX寄存器
        add eax,b  // eax和b相加 结果放在eax
        mov c,eax  // 将eax的值放在c上
    }
    printf("%d\n",c);// 把c的值输出 c = a + b
    return 0;
}
```

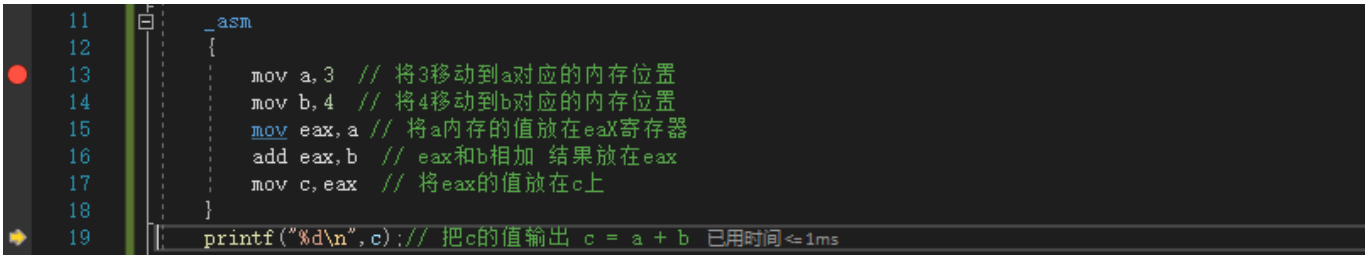
# 七、vs2019调试代码

- 在需要设置断点处的代码设置断点 按下F9
- 按下F5进行调试
- 按下F11进行逐语句执行






逐语句执行，可以看到变量的值随着逐语句执行而变化。




当vs出现4996警告编号，只需要在文件的最前面加上一句话：

```
#define _CRT_SECURE_NO_WARNINGS // 这个宏定义最好要放到.c文件的第一行
#pragma warning(disable:4996) // 或者直接过滤
```

每次运行时，将之前的终端关闭。