**Time：7月5日**

**昨日作业**

1W5的阶乘：

12345678

X 5256785

xxxx yyyy -> x 为进位 y为保存位

15W的阶乘：

1\*2\*3\*4\*5\*6\*7\*8\*9\*10

1\*3\*5\*7\*9 2\*4\*6\*8\*10

5！ \* 2^5

1\*2\*3\*4\*5 (去除5 计算有多少个5)

**上课内容（函数的机制、递归）**

1. **什么是\r\n**

\r :列清0行不变。

\n :行清0行不变。

1. **栈的解析**

（1）知识点：

栈顶是活动的、栈底是牢固的

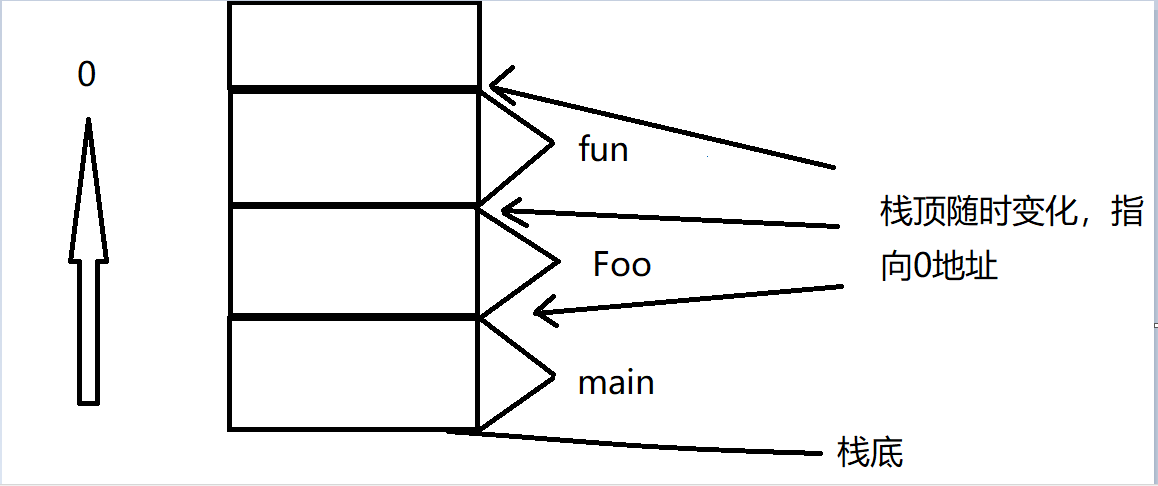
后进先出 🡪 栈结构

（2）画栈结构图的内容：

A)栈顶在哪里

B)栈的伸展方向

C)加一个0地址



(3)函数调用的过程：

1)按调用约定传递参数。

2）调用约定：调用方（caller）需要和被调方（callee）作出以下约定。

A:参数的传递方向

B:参数的传输媒介(用栈传还是用别的传)

C:函数返回值的位置

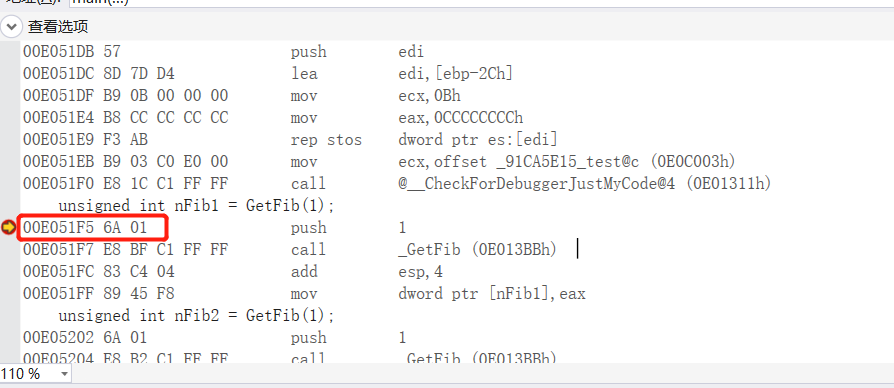
D:释放参数空间的负责方，有且仅有一方去释放参数空间。

!调用约定的细节（4种）：

\_\_cdecl：参数使用栈空间传递，从右往左，函数返回值在寄存器，由调用方负责释放空间。(可跨平台)

\_\_stdcall: 参数使用栈空间传递，从右往左，函数返回值在寄存器，由被调用方负责释放空间。(可跨平台)

\_\_fastcall: 左数前两个参数使用寄存器传递，其他参数使用栈空间传递，从右往左，函数返回值在寄存器，由被调用方负责释放空间。(微软独有）



3)保持返回地址（保持到栈中）

4)保存调用方的栈信息（保存栈底）

5)更新当前栈到到栈顶（把当前栈顶当作为被调方的栈底）

栈顶做加法能访问到参数空间，栈底做减法就访问到栈内其他数据

为局部变量申请空间（抬高栈顶）

6)保存寄存器环境（把即将使用的寄存器原值保存在栈里）

把寄存器里面的值保存

7\*)如果编译选项有/ZI /zi(启用调试信息的)则将局部变量初始化为0xcccccccc

1. 执行函数体

--------------------开始返回--------------------------->

9）恢复寄存器环境

10）释放局部变量的空间

11)恢复调用方的栈信息（栈底）。

12A）：\_cdecl，取出当前栈顶作为返回的流程地址，返回到调用方清理参数空间。

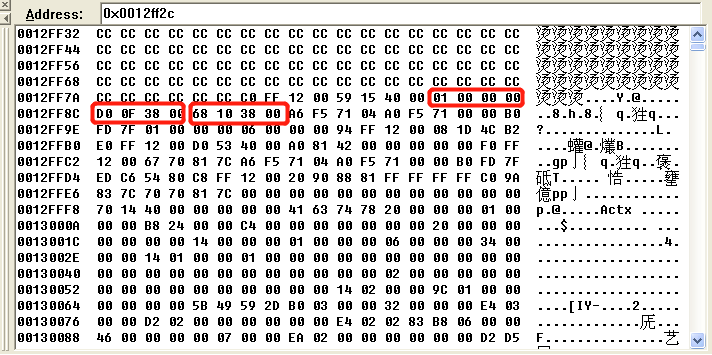
B）：其他约定，取出当前栈顶作为返回的流程地址，同时由被调方清理参数空间后才返回。

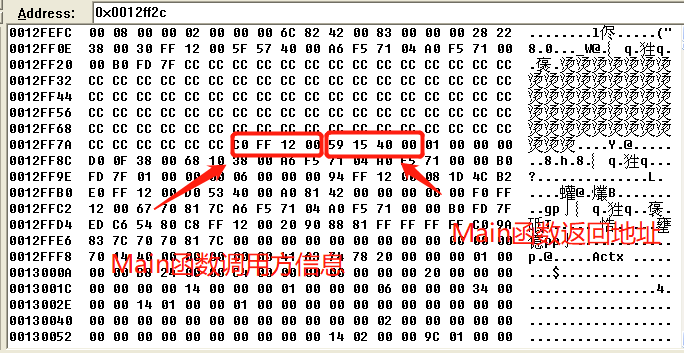
//栈帧

Stack frame

**图片流程：**

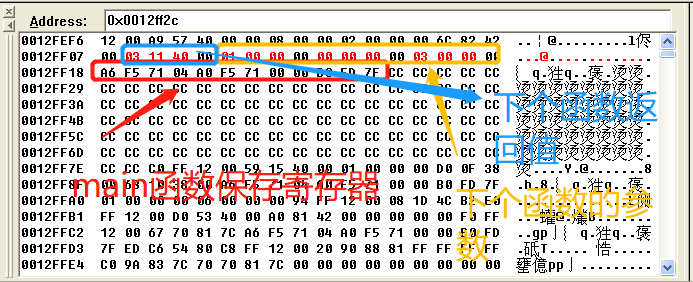
1、Main函数的三个参数（命令行个数-整形、命令行列表-字符型指针数值、环境变量列表-字符型指针数值）

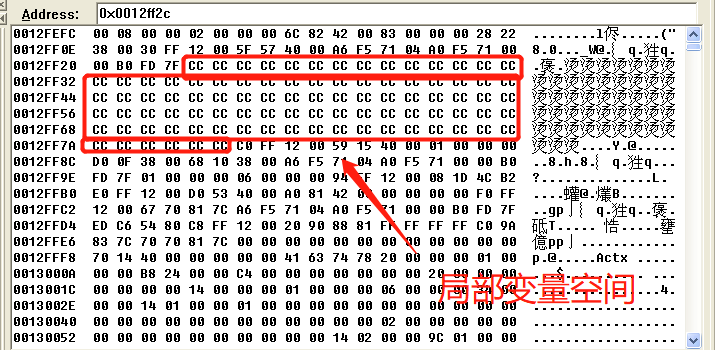


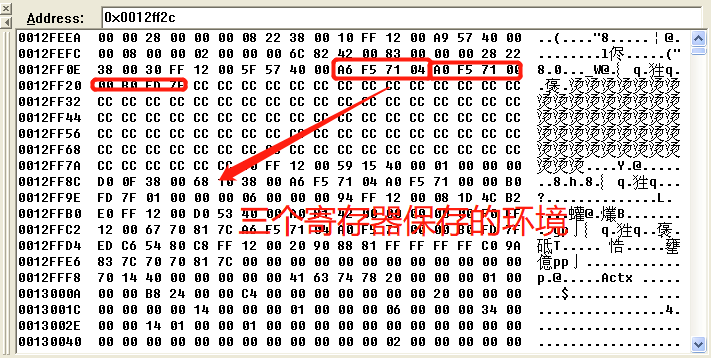


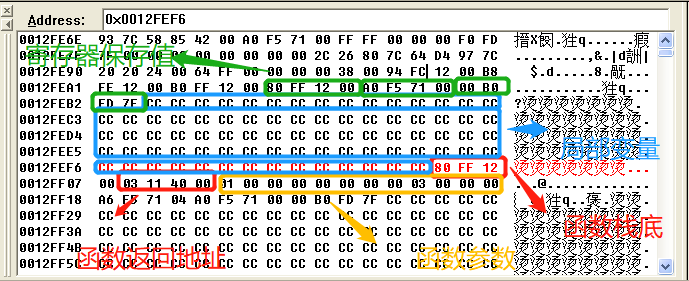
2、返回地址：为call后面的哪个地址

0012FFC0：同时作为Main函数的栈底，如果想要访问main函数栈空间中的变量，则用0012FFC0做减法进行访问，访问参数的话是这个位置做加法去访问。

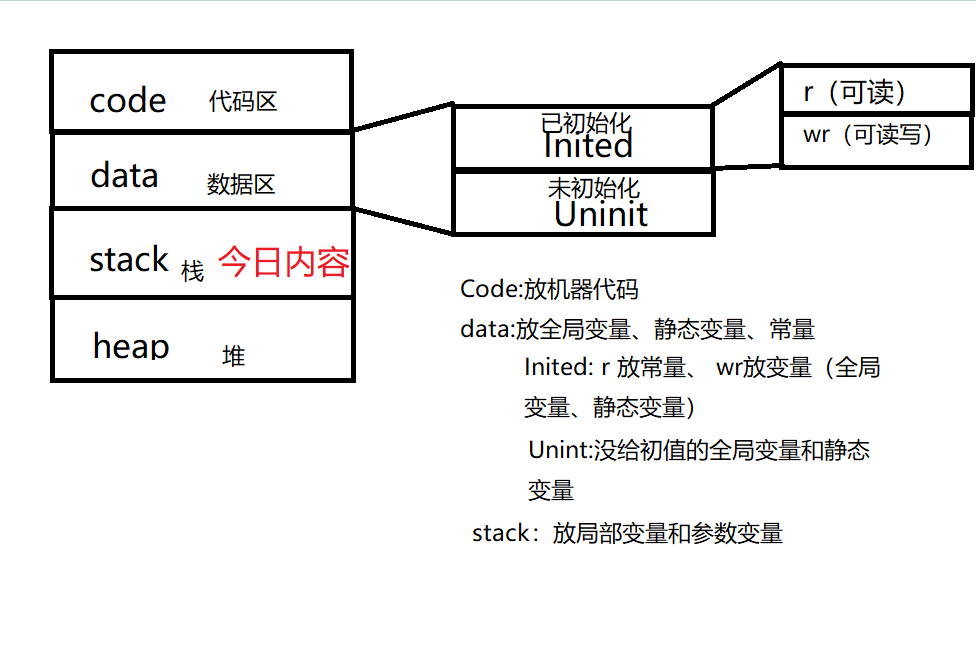








**3、程序要了解的4大分区**



**课堂小知识：**

1、如何修改调用约定：

int \_\_cdecl test();

2、优化与调试不可以同时输出

栈结构 图文并茂

两侧 栈结构 分析

Main() 函数 看出了哪些信息 10条