CS、IP的指令：

CPU从何处执行指令是由CS、IP中的内容决定的，程序员可以通过改变CS、IP中的内容来控制CPU执行目标指令。

若想同时修改CS、IP的内容，可用形如"jmp段地址：偏移地址"的指令完成，如

jmp 2AE3:3，执行后：CS=2AE3H，IP=0003H，CPU将从2AE33H处读取指令。

jmp 3:0B16，执行后：CS=0003H，IP=0B16H，CPU将从00B46H处读取指令。

"jmp 段地址：偏移地址"指令的功能为：用指令中给出的段地址修改CS，偏移地址修改IP。

汇编语言中CS，DS，SS还有一个ES定义如下：

CS（Code Segment）：代码段寄存器；

DS（Data Segment）：数据段寄存器；

SS（Stack Segment）：堆栈段寄存器；

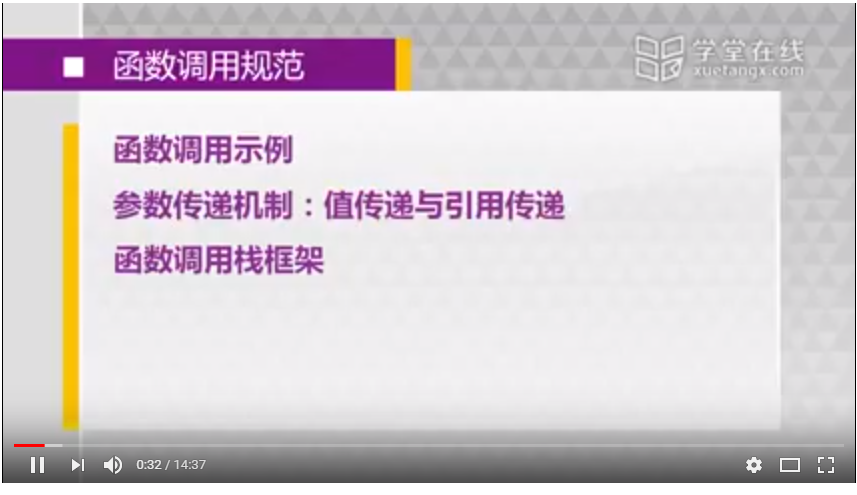
ES（Extra Segment）：附加段寄存器。搜索

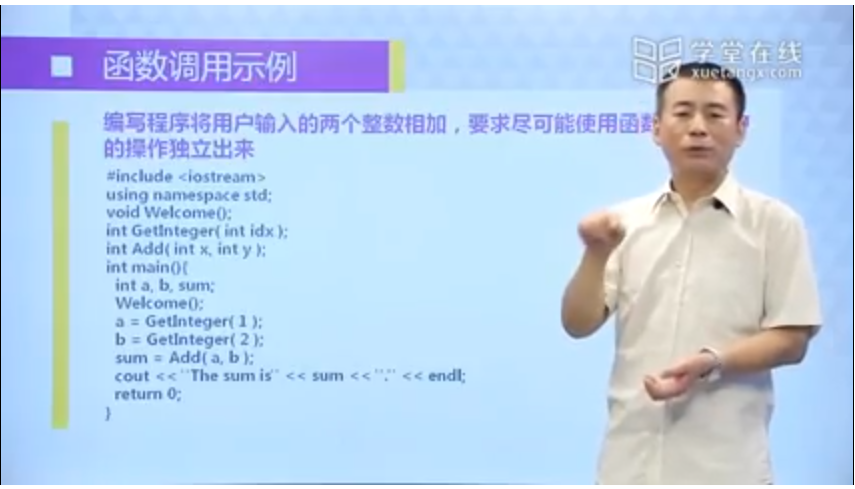
当一个程序要执行时，就要决定程序代码、数据和堆栈各要用到内存的哪些位置，通过设定段寄存器 CS，DS，SS 来指向这些起始位置。通常是将DS固定，而根据需要修改CS。

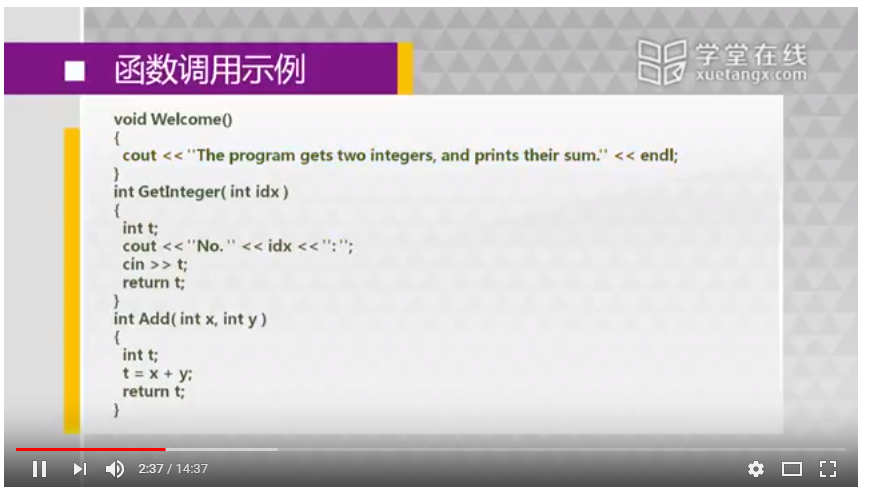
8086CPU中，有两个寄存器，段寄存器SS和SP指向栈顶元素。Push指令和pop指令执行时，CPU从SS和SP中得到栈顶的地址。

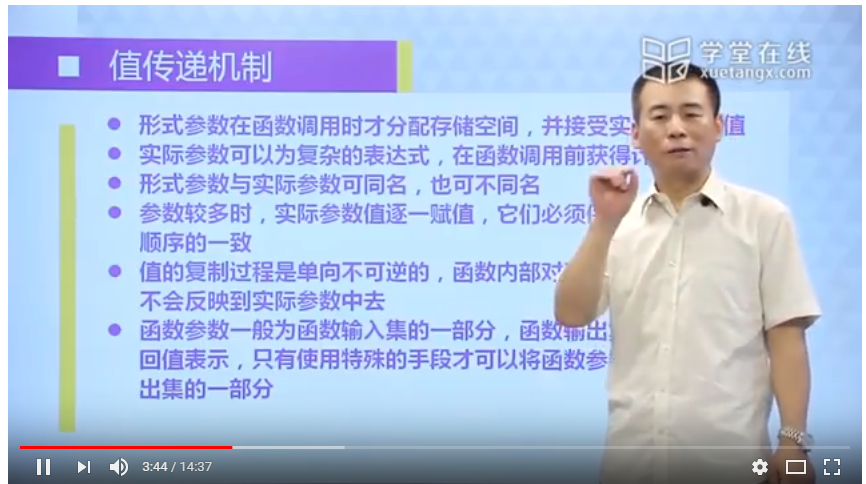
（1）ESP：栈指针寄存器(extended stack pointer)，其内存放着一个指针，该指针永远指向系统栈最上面一个栈帧的栈顶。

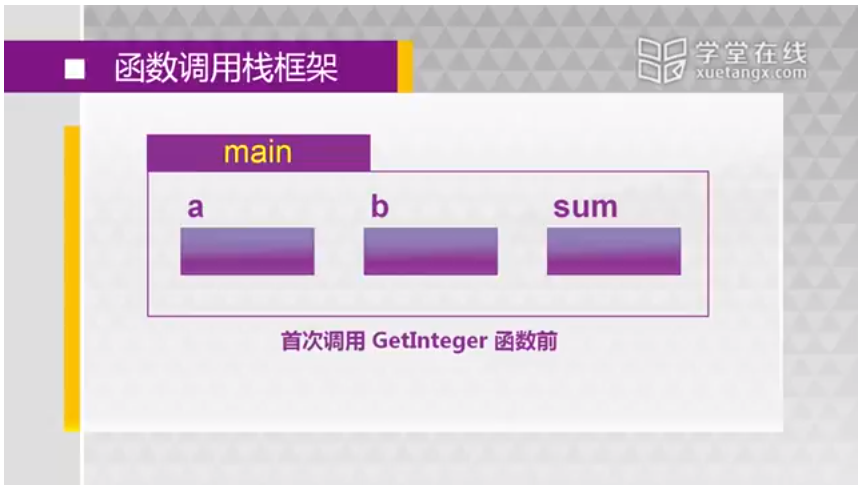
（2）EBP：基址指针寄存器(extended base pointer)，其内存放着一个指针，该指针永远指向系统栈最上面一个栈帧的底部。

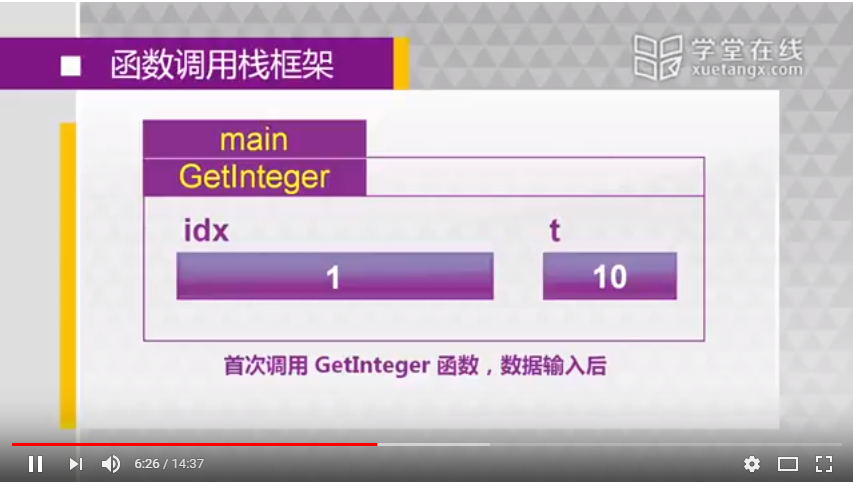




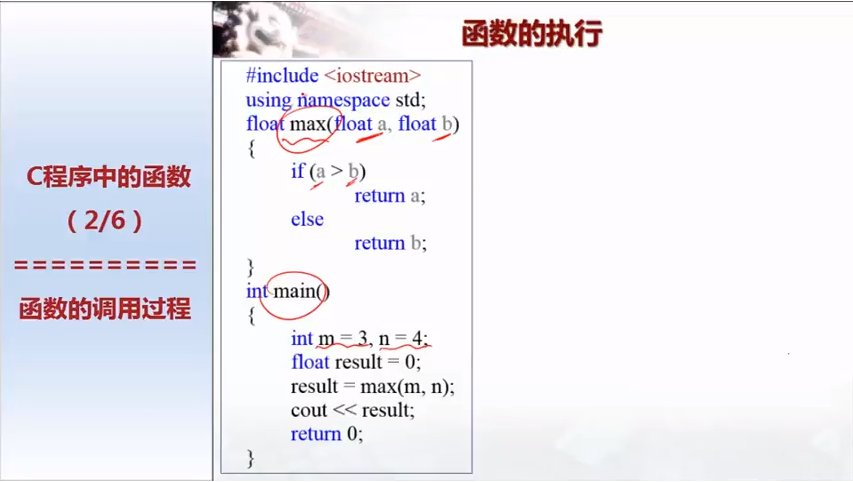


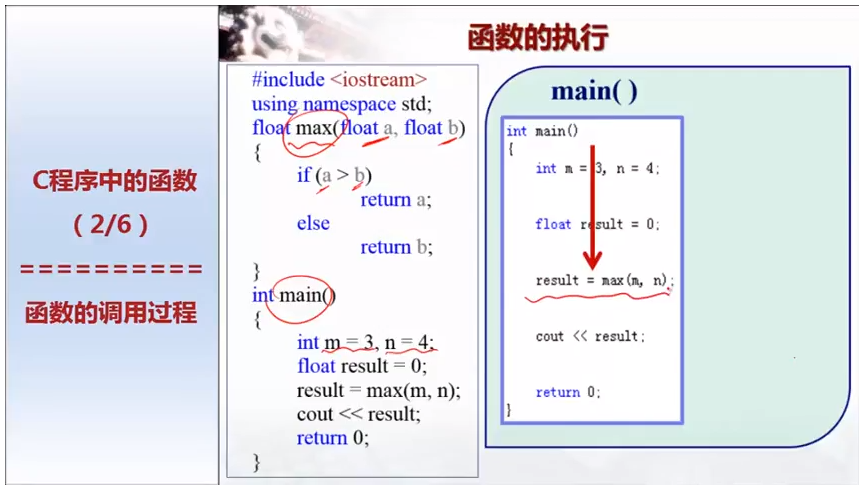




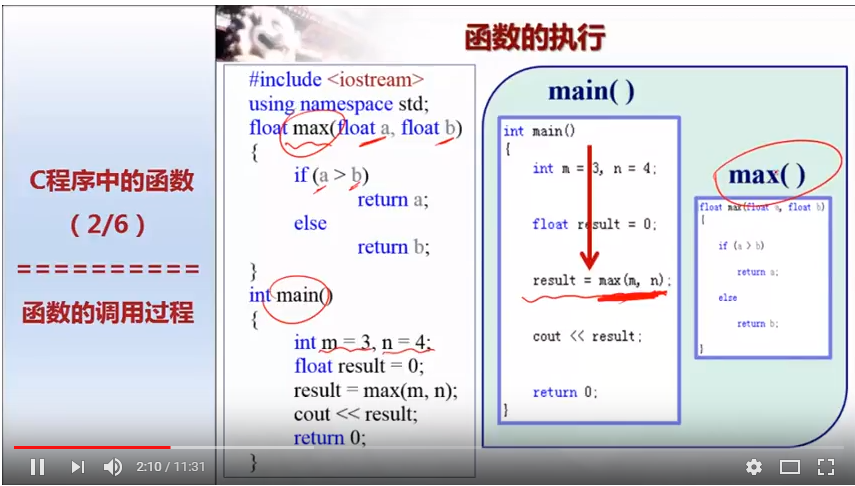


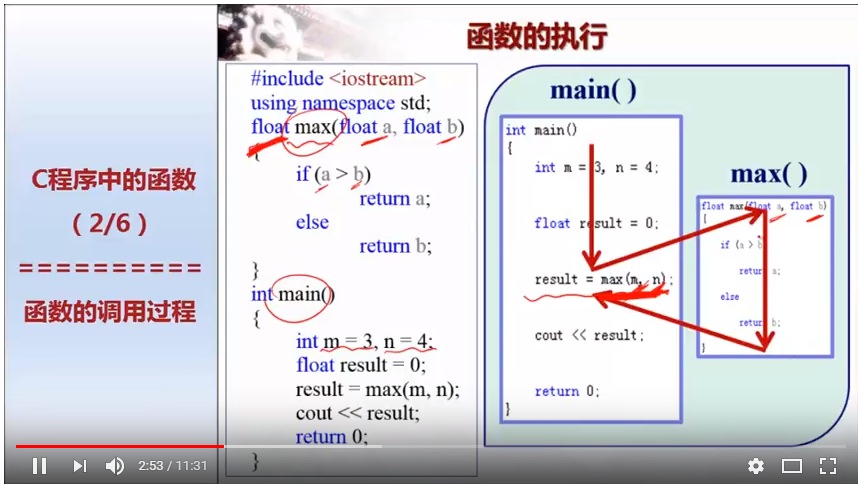


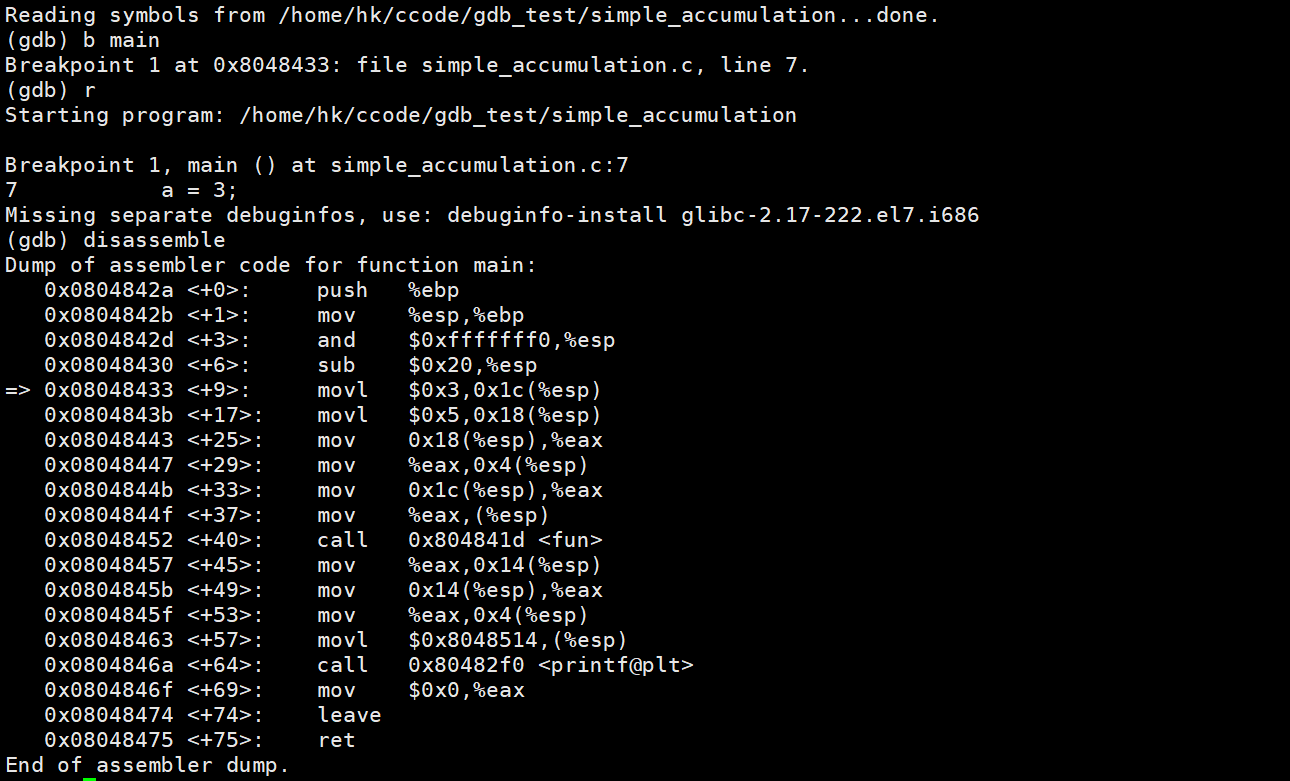


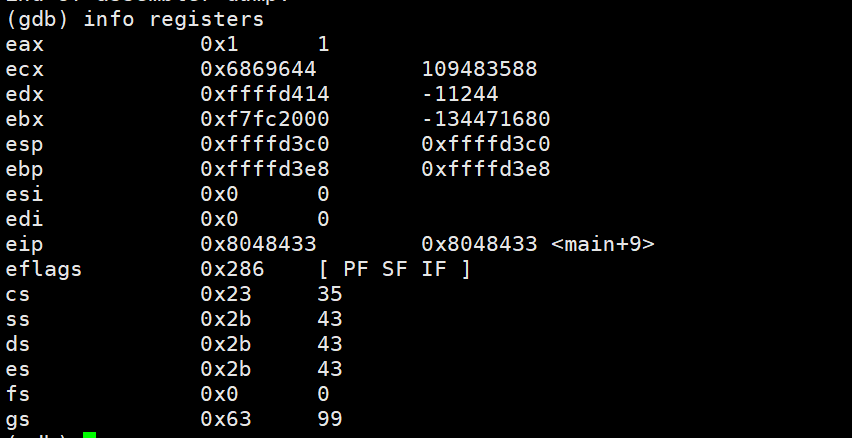


开辟max（）函数内存空间









%ebp是帧指针，%esp是栈指针。你提的这两句就是建栈的语句。

释放当前子程序在堆栈中的局部变量,使BP和SP恢复成最近一次的ENTER指令被执行前的值。

相当于：

MOV SP,BP

POP BP

ENTER指令相当于

PUSH BP

MOV BP,SP

我的理解：

调用一个函数时，先将堆栈原先的基址（EBP）入栈，以保存之前任务的信息。然后将栈顶指针的值赋给EBP，将之前的栈顶作为新的基址（栈底），然后再这个基址上开辟相应的空间用作被调用函数的堆栈。函数返回后，从EBP中可取出之前的ESP值，使栈顶恢复函数调用前的位置；再从恢复后的栈顶可弹出之前的EBP值，因为这个值在函数调用前一步被压入堆栈。这样，EBP和ESP就都恢复了调用前的位置，堆栈恢复函数调用前的状态。

mov ebp,esp实际上是把ebp进栈后的栈顶地址给了ebp。

mov esp,ebp之后，esp将指向ebp