**3.1 请简要说明C语言中未初始化局部变量的初值是随机值的原因。**

C语言的未初始化局部变量放在内存中的栈区，例如申请一个字节的char变量，则实质上执行的是：

DEC ESP;

而赋值时，才执行：

MOV [ESP+1], *src*;

因此，当只申请局部变量而未初始化（赋值）时，初值是内存中的随机值。

**3.2 请列出使得寄存器EAX内容为0的多种方法（每种方法，最多采用2条指令）。**

**（MOV** **EAX, 0 ;方法一）**

AND EAX, 0 ;方法二

SUB EAX, EAX ;方法三

MOV EBX, 0 ;方法四

MUL EAX. EBX ;

LEA EAX, [0] ;方法五

SAL EAX, 32 ;方法六

SHL EAX, 32 ;方法七

SHR EAX, 32 ;方法八

XOR EAX, EAX ;方法九

**3.3 请列出使得寄存器AL内容为1的多种方法（每种方法，最多采用2条指令）。**

AND AL, 1 ;方法一

MOV AL, 1 ;方法二

LEA AL, [1] ;方法三

MOV AL, O ;方法四

OR AL, 1;

MOV AL, O ;方法五

XOR AL, 1;

**3.4 实现同一功能，可能有多种方法。在选择方法时，要考虑哪些因素？**

一、时间复杂度：尽可能选择运行效率更高的方法，例如：更高效的算法，更合理的数据结构，汇编代码中，更多地利用高速的通用寄存器，减少对存储器的访问；

二、方法实现的难度，应选择更成熟更完善的方法，编写起来简洁明了的代码，易阅读易维护，不易出bug；

**3.5 请画出3.1节的例7中调用函数cf37期间堆栈的变化示意图。**

EBP

|  |
| --- |
|  |
| 返回地址 |
| EBP |

a）建立堆栈框架

push ebp

mov ebp, esp

ESP

|  |
| --- |
|  |
| 返回地址 |
| EBP |
| i = 1 |
| sum = 0 |

b）变量sum和i入栈

sub esp, 8

mov DWORD PTR [ebp-8], 0

mov DWORD PTR [ebp-4], 1

EBP

ESP

|  |
| --- |
|  |
| 返回地址 |
| EBP |
| i++ |
| sum += i |

c）循环

LN2cf37: i++

LN3cf37: sum += I

EBP

ESP

|  |
| --- |
|  |
| 返回地址 |
| EBP |
| i |
| sum |

d）撤销堆栈框架

Mov esp, ebp

Pop ebp

ESP

EBP

**3.6 请举一个例子，说明堆栈的四种用途。**

堆栈的用途：

一、保存函数的返回地址；

二、用于向函数传递函数；

三、安排函数的局部变量；

四、保护寄存器内容，保护程序现场状态。

\_asm{

PUSH EAX; //二、用于向函数传递函数

CALL DEMO; //一、保存函数的返回地址

}

\_asm{

DEMO:

PUSH EBP;

MOV EBP, ESP;

MOV EAX, DWORD PTR[ESP+8]; //二、用于向函数传递函数

MOV EBX, 1;

PUSH EBX; //三、安排函数的局部变量

POP EBX;

POP EBP; //四、保护寄存器内容，保护程序现场状态

}