# 2023春《计算机系统》小班讨论-4

## 讨论课实施方法：

每次讨论课布置三道选题，每个班分为六个小组，同一道题有两个小组选择（小组间协商或随机分配）并进行准备。在讨论课时，每道题随机选择一组进行汇报，另外一组或其他组进行质疑与提问。

讨论课的成绩由本组表现及个人表现组成。

## 讨论课选题

选题一

如下程序pwd.c用于判断输入的密码是否正确

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define PASSWORD "1234567"

int verify(char \*password)

{

int auth;

char buffer[8];

auth = strcmp(password, PASSWORD);

strcpy(buffer, password);

return auth;

}

int main(void){

int flag = 0;

char pass[1024];

while(1){

printf("enter the password:\t");

scanf("%s", pass);

flag = verify(pass);

if(flag)

printf("password incorrect!\n");

else{

printf("congratulation!\n");

break;

}

}

}

在关闭缓冲区溢出保护后进行编译：（Ubuntu 12.04 32位,gcc版本4.6.3）

gcc -fno-stack-protector -z execstack -o pwd.out pwc.c

在命令行输入

./pwd.out运行，

发现输入：11111111、12344444等字符串时会提示密码不正确，而在输入qqqqqqqq、12355555时会提示密码正确。

请：

1. 分析这些现象产生的原因，并总结规律；
2. 在缺省编译状态下，例如gcc pwd.c -o pwd.out后，运行程序不会产生这样的结果，试从汇编代码这一级别进行解释；
3. 测试其他的c语言编译器（例如windows下的visual studio，gcc的低版本2.7.3等）是否有缓冲区溢出保护功能。

选题二

小学生小军在学习递归的概念后，很轻松的写出了求菲波拉契数列的c语言代码：

#include "stdio.h"

int f(int n)

{

if (n<=0) return 0;

if (n==1 || n==2) return 1;

return f(n-1)+f(n-2);

}

int main()

{

int i=f(5);

printf("%d\n",i);

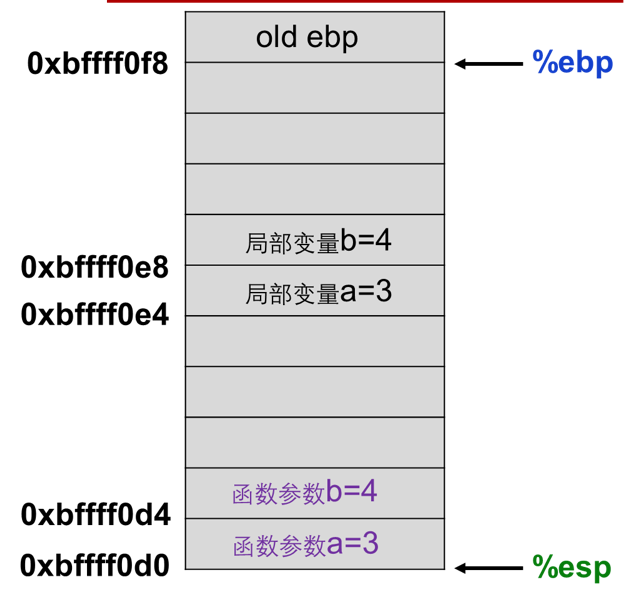
return 0;

}

在得知某些计算机专业的大学生要很费劲才能写出这个程序后，他很骄傲自满，为了让小军明白计算机专业知识的博大精深，我们决定向小军演示整个函数执行过程中栈帧变化情况。

要求：

从进入main函数开始到结束，每一次函数调用都至少有一个栈帧的示意图，并标注栈顶、栈底，局部变量，参数，旧ebp，返回地址等内容，例如：



（具体数值以在本机上的运行情况为准）

选题三

小明在《计算机系统》的期末考试复习过程中，预感到老师会出如下的题目，但小明不会做，请告诉小明答案及详细的解题过程。

请参照汇编代码将如下的c语言程序补充完整：

#include "stdio.h"

#include "stdlib.h"

int main()

{

int a[]={3,-5,6,7,2,-8,10,2,4};

struct link

{

int i;

struct link \* next;

struct link \* pre;

}head,\*p1,\*p2;

head.i=a[0];

head.next=NULL;

head.pre=NULL;

int j=0;

p1=\_\_\_\_\_\_\_\_\_\_;

for(j=1;j<=8;j++)

{

p2=(struct link\*)malloc(sizeof(head));

p2->i=\_\_\_\_\_\_;

p1->next=p2;

p2->next=NULL;

p2->pre=p1;

p1=p2;

}

p1=&head;

while(p1->next)

{

if(\_\_\_\_\_\_\_\_\_\_)

\_\_\_\_\_\_\_\_\_\_\_\_;

else

p1->i-=p1->next->i;

printf("%d\n",p1->i);

p1=p1->next;

}

return 0;

}

编译后的汇编代码如下：

main:

pushl %ebp

movl %esp, %ebp

andl $-16, %esp

subl $80, %esp

movl $3, 20(%esp)

movl $-5, 24(%esp)

movl $6, 28(%esp)

movl $7, 32(%esp)

movl $2, 36(%esp)

movl $-8, 40(%esp)

movl $10, 44(%esp)

movl $2, 48(%esp)

movl $4, 52(%esp)

movl 20(%esp), %eax

movl %eax, 56(%esp)

movl $0, 60(%esp)

movl $0, 64(%esp)

movl $0, 72(%esp)

leal 56(%esp), %eax

movl %eax, 68(%esp)

movl $1, 72(%esp)

jmp .L2

.L3:

movl $12, (%esp)

call malloc

movl %eax, 76(%esp)

movl 72(%esp), %eax

movl 20(%esp,%eax,4), %eax

movl %eax, %edx

imull 72(%esp), %edx

movl 76(%esp), %eax

movl %edx, (%eax)

movl 68(%esp), %eax

movl 76(%esp), %edx

movl %edx, 4(%eax)

movl 76(%esp), %eax

movl $0, 4(%eax)

movl 76(%esp), %eax

movl 68(%esp), %edx

movl %edx, 8(%eax)

movl 76(%esp), %eax

movl %eax, 68(%esp)

addl $1, 72(%esp)

.L2:

cmpl $8, 72(%esp)

jle .L3

leal 56(%esp), %eax

movl %eax, 68(%esp)

jmp .L4

.L7:

movl 68(%esp), %eax

movl (%eax), %edx

movl 68(%esp), %eax

movl 4(%eax), %eax

movl (%eax), %eax

leal (%edx,%eax), %ecx

movl 68(%esp), %eax

movl (%eax), %edx

movl 68(%esp), %eax

movl 4(%eax), %eax

movl (%eax), %eax

imull %edx, %eax

movl %eax, %edx

shrl $31, %edx

addl %edx, %eax

sarl %eax

cmpl %eax, %ecx

jle .L5

movl 68(%esp), %eax

movl (%eax), %edx

movl 68(%esp), %eax

movl 4(%eax), %eax

movl (%eax), %eax

addl %eax, %edx

movl 68(%esp), %eax

movl %edx, (%eax)

jmp .L6

.L5:

movl 68(%esp), %eax

movl (%eax), %edx

movl 68(%esp), %eax

movl 4(%eax), %eax

movl (%eax), %eax

subl %eax, %edx

movl 68(%esp), %eax

movl %edx, (%eax)

.L6:

movl 68(%esp), %eax

movl (%eax), %edx

movl $.LC0, %eax

movl %edx, 4(%esp)

movl %eax, (%esp)

call printf

movl 68(%esp), %eax

movl 4(%eax), %eax

movl %eax, 68(%esp)

.L4:

movl 68(%esp), %eax

movl 4(%eax), %eax

testl %eax, %eax

jne .L7

movl $0, %eax

leave

ret