1，

（1）

0 000(a) 0000(b)

c=1.b\*2^(a-bias)

当a不变时，b增大，数值变大

当a发生变化时，排在后面的数比前面的数位数多，数值更大

排越后，b, a越大，数值越大

（2）

当a不变时，步长不变；当a增大时，步长会逐渐增大

2，

#include <stdio.h>

#define X 16

#define Y (1) 23

int array1[X][Y];

int array2[X];

int test()

{

int sum= (2) ; 6 //-12(%ebp)

int i=0,j=0,k=0; movl $1, -20(%ebp) movl $2, -16(%ebp) movl $0, -8(%ebp) i和j和k

for(i=1,j=2;i<X && j<Y;i++) X=16 Y=23

{

k= (3) ; j>15? 0: j-15; //array1+(i\*23+j)\*4==array1[i][j]

array2[i]+= (4) ; (2\*(0-array2[k])+array1[i][j])

j++;

}

return (5) ; array2[0]+sum+array1[0][0] ????!!!!!

}

int main() {

return 0;

}

3,

#include <stdio.h>

union u

{

int i;

int \*pi;

char \*pc;

}ua;

int a[3]={0,1,2};

int test1(int \*x,int y)

{

int result =0; movl $0, -4(%ebp)

if( (1) ) (\*x)>(y<<2+0xa)

result=2;

return result;

}

int test2(int x,int y)

{

int result=0; movl $0, -4(%ebp)

if( (2) ) ua.i<x\*y

result= (3) ; test1(ua.pi, test1(ua.pi, ua.i+10))

return result;

}

int main() {

int i=0; movl $0, -8(%ebp)

ua.pi=&i;

int b=ua.i;

i=test2(20,b+10);

return i;

}

4,（1）

esp = 0xbffff1c8

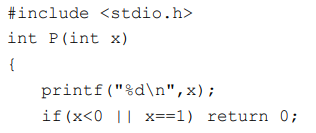
p(5) ebp

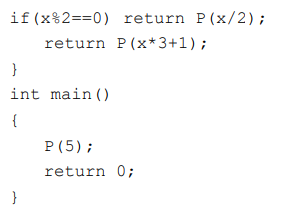
esp = 0xbffff1e8

Main ebp = 0xbffff1f8

|  |
| --- |
| 返回地址 |
| Old ebp |
|  |
|  |
|  |
| 0x5 |
| 0x804845c |
| p(5) old ebp |
|  |
|  |
|  |
|  |
|  |
| 0x10(16) |
|  |

一次调用减0x20（8个格子(int大小)）





（1）

Esp=0xbffff1e8-0x60-0x4=0xbffff184

Ebp=0xbffff184+0x1c=0xbffff1a0

(%esp)= 0x804842f

(%esp+4)=0x4

（2）

Int p( int x ) {

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

If(x<0 || x==1) return 0;

While(x!=1) {

If(x%2==0) x/=2;

Else 3\*x+1;

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

}

Return 1;

}

该函数没有使用递归的做法，在运行过程中不会有太多的的函数调用，也就不会消耗太多的栈空间，不会出现此类问题。

5，

就随便加？

6,

（1）

ELF头中字段e\_entry给出执行程序时第一条指令的地址，而在可重定位文件中，此字段为0

 多一个程序头表（Program Headers table）是一个结构体数组：偏移量，虚存地址，物理地址，段大小，地址对齐等信息（readelf -l）

 多一个.init节，用于定义\_init函数，该函数用来进行可执行目标文件开始执行时的初始化工作

 可执行目标文件中少两.rel节（无需重定位）

（2）

X在两个文件中都有定义且都是弱符号，编译器会随机选一个，那么最后输出不确定。

（3）

R\_386\_PC32：由目标函数和节的运行时地址和引用偏移计算出函数运行时地址与PC的偏移

R\_386\_32：将符号引用指向运行时符号地址

动态链接重定位(printf)：生成可执行文件时只是链接上符号和重定位信息，等到加载时由加载器通过plt和got完成全部的链接。

简述重定位过程

（4）shell调用fork创建一个新子进程，子进程中调用exec类的系统调用在当前进程的下上下文中加载并运行一个程序（p）。

（5）SIGHLD，子进程的父进程；对于一种信号最多只可以有一个待处理信号，多出来的信号会被丢弃。

7，

0x9fd28 c10

1001 1111 11/01 0010 1000 c10

A1）27f4a 不命中

A2）

A3）0045d 27f 0

A4）失败 0x0045d27f

0x0a32f cd0

0000 1010 00/11 0010 1111 cd0

A1）028cb 不命中

A2）

A3）0045d 028 000c4297+32f=000c45c6

A4）失败 0x000c45c6

0x0d418 2c0

0000 1101 01/00 0001 1000 2c0

A1）03506 命中

A2）0x98f8a 2c0

A3）

A4）

假设主存2^32=4G，页大小为4k，一个条目4b

一页装1k(2^10)个项，共有2^20个虚拟页，需要2^10个物理页来装下，这2^10个物理页的基址又在装入一个物理页，形成2级页表