**3.43**

**A.**

|  |  |
| --- | --- |
| 08 04 86 43 | 返回地址 |
| bf ff fc 94 | 保存的%ebp (%ebp指向这里) |
| 00 00 00 02 | 保存的%edi |
| 00 00 00 03 | 保存的%esi |
| 00 00 00 01 | 保存的%ebx |
|  | buf[7] – buf[4] |
|  | buf[3] – buf[0] |
|  |  |

**B.**

|  |  |
| --- | --- |
| 08 04 86 00 | 返回地址 |
| 33 32 31 30 | 保存的%ebp (%ebp指向这里) |
| 39 38 37 36 | 保存的%edi |
| 35 34 33 32 | 保存的%esi |
| 31 30 39 38 | 保存的%ebx |
| 37 36 35 34 | buf[7] – buf[4] |
| 33 32 31 30 | buf[3] – buf[0] |
|  |  |

**C.**

0x08048600

**D.**

%ebp, %edi, %esi, %ebx.

**E.**

C代码第9行应为: result = malloc(strlen(buf) + 1);

其次应该检查返回值是否为NULL.

**3.44**

**A.**

213

**B.**

26

**3.45**

**A.**

|  |  |  |
| --- | --- | --- |
|  | 不带保护者 | 带保护者 |
| buf | %ebp - 20 | %ebp - 20 |
| v | %ebp - 8 | %ebp - 24 |
| canary | 无 | %ebp - 8 |

**B.**

局部变量v比buf更靠近栈顶, 从而可防止buf溢出破坏v的值.

**3.47**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| src\_t | dest\_t | 指令 | S | D |
| long | long | movq | %rdi | %rax |
| int | long | movslq | %edi | %rax |
| char | long | movsbq | %dil | %rax |
| unsigned int | unsigned long | movl | %edi | %eax |
| unsigned char | unsigned long | movzbq | %dil | %rax |
| long | int | movl | %edi | %eax |
| unsigned long | unsigned | movl | %edi | %eax |

**3.48**

long arithprob(int a, char b, long c, int d);

**3.49**

**A.**

long fun\_c(unsigned long x)

{

long val = 0;

int i;

for(i = 0; i != 8; i++)

{

val += x & 0x0101010101010101L;

x >> 1;

}

val += (val >> 32);

val += (val >> 16);

val += (val >> 8);

return val & 0xFF;

}

**B.**

计算x的二进制串中1的个数.

**3.69**

**A.**

long trace(tree\_ptr tp)

{

long result = 0;

while(tp != NULL)

{

result = tp->val;

tp = tp->left;

}

return result;

}

**B.**

从一个二叉树的当前结点开始, 遍历左子树直到叶子结点, 并返回该叶子结点的值.

**3.70**

**A.**

long traverse(tree\_ptr tp)

{

long v = MAX\_LONG, rv, lv;

if(tp != NULL)

{

lv = traverse(tp->left);

rv = traverse(tp->right);

v = lv < rv ? lv : rv;

v = v > tp->v ? tp->v : v;

}

return v;

}

**B.**

后序遍历二叉树, 找到所有结点中的最小值.