1．假设某个 C 语言函数 func 的原型声明如下： void func(int \*xptr, int \*yptr, int \*zptr); 函数 func 的过程体对应的机器级代码用 AT&T 汇编形式表示如下： 1 movl 8(%ebp), %eax 2 movl 12(%ebp), %ebx 3 movl 16(%ebp), %ecx 4 movl (%ebx), %edx 5 movl (%ecx), %esi 6 movl (%eax), %edi 7 movl %edi, (%ebx) 8 movl %edx, (%ecx) 9 movl %esi, (%eax) 请回答下列问题或完成下列任务：

（1）过程体开始时三个入口参数对应实参所存放的存储单元地址是什么？（提示：当前栈 帧底部由帧指针寄存器 EBP 指示）

（2）根据上述机器级代码写出函数 func 的 C 语言代码。

(1) xptr、yptr、zptr对应实参所存放的存储单元地址分别为：R[ebp]+8、R[ebp]+12、R[ebp]+16

（2）

void func(int \*xptr, int \*yptr, int \*zptr)

{

int tempx = \*xptr;

int tempy = \*yptr;

int tempz = \*zptr;

\*yptr = tempx;

\*zptr = tempy;

\*xptr = tempz;

}

2．已知 IA-32 是小端方式处理器，根据给出的 IA-32 机器代码的反汇编结果（部分信息用 x 表示）回答问题：

（1）已知 je 指令的操作码为 01110100，je 指令的转移目标地址是什么？call 指令中的转移 目标地址 0x80483b1 是如何反汇编出来的？

804838c: 74 08 je xxxxxxx

804838e: e8 le 00 00 00 call 80483b1

（2）已知 jb 指令的操作码为 01110010，jb 指令的转移目标地址是什么？movl 指令中的目 的地址如何反汇编出来的？

8048390: 72 f6 jb xxxxxxx

8048392: c6 05 00 a8 04 08 01 movl $0x1, 0x804a800

8048399: 00 00 00

（3）已知 jle 指令的操作码为 01111110, mov 指令的地址是什么？

xxxxxxx: 7e 16 jle 80492e0

xxxxxxx: 89 d0 mov %edx, %eax

（4）已知 jmp 指令的转移目标地址采用相对寻址方式，jmp 指令操作码为 11101001，其转 移目标地址是什么？

8048296: e9 00 ff ff ff jmp xxxxxxx

804829b: 29 c2 sub %eax, %edx

（1）je指令的转移目标地址为804838c+8=8048394

Call指令的吓一跳指令为804838e+5=8048393 根据机器指令的小端排列知偏移量为le则call指令的调用地址围殴8048393+1e=80483B1

（2）jb的转移目标地址为：8048390+f6=8048486

movl的目的地址是根据机器指令反汇编出来的，机器指令为c6 05 00 a8 04 08 01，根据小端存储，其中前两个字节是操作码，00 a8 04 08为目的地址即804a800

（3）jle指令地址应为80492e0-16= 80492CA，mov指令为下一条指令则其地址为jle指令地址偏移2个字节即80492CA+2=80492CC

（4）根据机器指令值偏移量为ff ff ff 00即-256，16进制真值为-100，则jmp转移目标地址为8048296-100=8048196

3.已知函数 f1 的 C 语言代码框架及其过程体对应的汇编代码如下所示，根据对应的汇编 代码填写 C 代码中缺失的部分，并说明函数 f1 的功能。

C 语言代码：

1 int f1(unsigned x)

2 {

3 int y = 0;

4 while( \_\_\_\_\_\_\_\_\_\_\_\_ ) {

5 \_\_\_\_\_\_\_\_\_\_\_\_ ;

6 }

7 return \_\_\_\_\_\_\_\_\_\_\_\_ ;

8 } 汇编代码：

1 movl 8(%ebp), %edx

2 movl $0, %eax

3 testl %edx, %edx

4 je .L1

5 .L2:

6 xorl %edx, %eax

7 shrl $1, %edx

8 jne .L2

9 .L1:

10 andl $1, %eax

C语言代码如下

int f1(unsigned x)

{

int y = 0;

while(x != 0) {

y ^= x;

x >>= 1;

}

return y&0x1;

}

f1的功能是返回(x(x>>1)(x>>2)^…)&0x1，因此f1用于检测x的奇偶性，有奇数个1则返回1，否则返回0。

4．已知函数 funct 的 C 语言代码如下：

1 #include

2 int funct(viod) {

3 int x, y;

4 scanf(“%d %d”, &x, &y);

5 return x-y;

6 }

函数 funct 对应的汇编代码如下：

1 funct:

2 pushl %ebp

3 movl %esp, %ebp

4 subl $40, %esp

5 leal -8(%ebp), %eax

6 movl %eax, 8(%esp)

7 leal -4(%ebp), %eax

8 movl %eax, 4(%esp)

9 movl $.LC0, (%esp) ; 将指向字符串“%d %d”的指针入栈

10 call scanf ; 假定 scanf 执行后 x = 15，y = 20

11 movl -4(%ebp), %eax

12 subl -8(%ebp), %eax

13 leave 1

4 ret

假设函数 funct 开始执行时，R[esp]=0xbc000020，R[ebp]=0xbc000030，指向字符串“%d %d” 的指针为 0x804c000。 回答下列问题或完成下列任务：

（1）执行第 3、10 和 13 行的指令后，寄存器 EBP 中的内容分别是什么？

（2）执行第 3、10 和 13 行的指令后，寄存器 ESP 中的内容分别是什么？

（3）局部变量 x 和 y 所在存储单元的地址分别是什么？

（4）画出执行第 10 行指令后 funct 的栈帧，指出栈帧中的内容及其地址。

（1）执行第3、10、13行的指令后，寄存器EBP中的内容分别是0xbc00001c、0xbc00001c、0xbc000030

（2）执行第3、10、13行的指令后，寄存器ESP中的内容分别是0xbc00001c、0xbbfffff4、0xbc000020

（3）由指令5-8可知x和y的地址分别为R[ebp]-4=0xbc000018、R[ebp]-8=0xbc000014

5．假设函数 sumij 的 C 代码如下，其中 M 和 N 是用#define 声明的常数。

1 int a[M][N], b[N][M];

2

3 int sumij(int i, int j) {

4 return a[i][j]+b[j][i];

5 }

已知函数 sumij 的过程体对应的汇编代码如下：

1 movl 8(%ebp), %ecx

2 movl 12(%ebp), %edx

3 leal (, %ecx, 8), %eax

4 subl %ecx, %eax

5 addl %edx, %eax

6 leal (%edx, %edx, 4), %edx

7 addl %ecx, %edx

8 movl a(, %eax, 4), %eax

9 addl b(, %edx, 4), %eax

根据上述汇编代码，确定 M 和 N 的值。

1 movl 8(%ebp), %ecx //R[ecx] = i

2 movl 12(%ebp), %edx //R[edx] = j

3 leal (,%ecx, 8), %eax //R[eax] = 8\*i

4 subl %ecx, %eax //R[eax] = 8i - i = 7i

5 addl %edx, %eax //R[eax] = 7\*i + j

6 leal (%edx, %edx, 4), %edx //R[edx] = 4j + j = 5j

7 addl %ecx, %edx //R[edx] = 5\*j + i

8 movl a(, %eax, 4), %eax //R[eax] = M[a+74i + 4\*j

9 addl b(,%edx, 4), %eax //R[eax] = M[a + 74i +4j] + M[b + 54j + 4i

由此可知，M=5，N=7

6．假设函数trans\_matrix 的C代码如下，其中M是用#define声明的常数。

1void trans\_matrix(int a[M][M]){

2int i, j, t;

3for (i= 0; i<M; i++)

4for(j =0; j<M; j++){

5t=a[i][j];

6a[i][j]=a[j][i];

7a[j][i]=t;

8}

9 }

已知采用优化编译(选项-O2)后函数trans\_matrix的内循环对应的汇编代码如下:

.L2:

2movl (%ebx), %eax

3movl (%esi,%ecx,4), %edx

4movl %eax, (%esi, %ecx,4)

5addl $1,%ecx

6movl %edx, (%ebx)

7addl $76, %ebx

8cmpl %edi, %ecx

9jl .L2

根据上述汇编代码，回答下列问题或完成下列任务:

(1) M的值是多少?常数M和变量j分别存放在哪个寄存器中?

(2）写出上述优化汇编代码对应的函数trans matrix 的C代码。

1. M的值是76。M存放在edi中，j存放挂在ebx中
2. 优化后的trans\_matrix的C代码如下：

Void trans\_matrix(int a[M]){

int i=0,j=0,t;

while(i<M){

t=a[i][j];

a[i][j]=a[j][i];

a[j][i]=t;

i++;

j+=76;

}

}

7．假设联合类型 utype 的定义如下：

typedef union {

struct {

int x;

short y;

short z; } s1

struct {

short a[2];

int b;

char \*p; } s2

} utype;

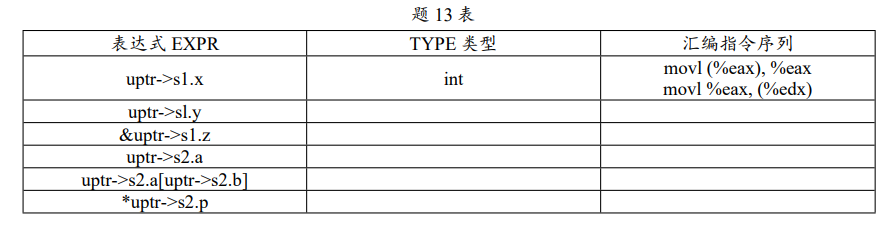
若存在具有如下形式的一组函数：

void getvalue(utype \*uptr, TYPE \*dst) {

\*dst = EXPR;

}

该组函数用于计算不同表达式 EXPR 的值，返回值的数据类型根据表达式的类型确定。假 设函数 getvalue 的入口参数 uptr 和 dst 分别被装入寄存器 EAX 和 EDX 中，仿照例子填写下 表，说明在不同的表达式下的 TYPE 类型以及表达式对应的汇编指令序列（要求尽量只用 EAX 和 EDX，不够用时再使用 ECX）



|  |  |  |
| --- | --- | --- |
| 表达式EXPR | TYPE类型 | 汇编指令序列 |
| uptr->s1.x | int | movl(%eax),%eax  movl%eax,(%edx) |
| uptr->s1.y | short | movw(%eax),%eax  movw %eax,(%edx) |
| &uptr->s1.z | short | lea 4(%eax),%eax  movw (%eax),%eax  movw %eax,(%edx) |
| uptr->s2.a | short\* | movw (%eax), %eax movw %eax, (%edx) |
| uptr->s2.a[uptr->s2.b] | short | movzwl 4(%eax), %eax movw (%eax,%eax,2), %eax movw %eax, (%edx) |
| \*uptr->s2.p | char\* | movl 8(%eax), %eax movl (%eax), %eax movl %eax, (%edx) |

8．函数 lproc 的过程体对应的汇编代码如下：

1 movl 8(%ebp), %edx

2 movl 12(%ebp), %ecx

3 movl $255, %esi

4 movl $-2147483648, %edi

5 .L3:

6 movl %edi, %eax

7 andl %edx, %eax

8 xor1 %eax, %esi

9 movl %ecx, %ebx

10 shrl %bl, %edi

11 testl %edi, %edi

12 jne .L3

13 movl %esi, %eax

上述代码根据以下 lproc 函数的 C 代码编译生成：

1 int

2 lproc(int x, int k) {

3 int val = \_\_\_\_\_\_\_\_\_\_\_\_ ;

4 int i;

5 for (i = \_\_\_\_\_\_\_\_\_\_\_\_ ; i \_\_\_\_\_\_\_\_\_\_\_\_ ; i = \_\_\_\_\_\_\_\_\_\_\_\_ ) {

6 val ^= \_\_\_\_\_\_\_\_\_\_\_\_ ;

7 }

8 return val;

9 }

回答下列问题或完成下列任务：

（1）给每条汇编指令添加注释。

（2）参数 x 和 k 分别存放在哪个寄存器中？局部变量 val 和 i 分别存放在哪个寄存器中？

（3）局部变量 val 和 i 的初始值分别是什么？

（4）循环终止条件是什么？循环控制变量 i 是如何被修改的？

（5）填写 C 代码中缺失的部分。

(1)

1 movl 8(%ebp), %edx //将位于EBP+8的内存位置的值加载到EDX寄存器中（参数x）

2 movl 12(%ebp), %ecx //将位于EBP+12的内存位置的值加载到ECX寄存器中（参数k

3 movl $255, %esi //将立即数255加载到ESI寄存器中

4 movl $-2147483648, %edi //将立即数-2147483648加载到EDI寄存器中

5 .L3: //标号，用于后续跳转

6 movl %edi, %eax //将EDI寄存器的值复制到EAX寄存器中

7 andl %edx, %eax //将EDX寄存器的值与EAX寄存器的值进行按位与运算，并将结果存储在EAX寄存器中

8 xor1 %eax, %esi //将EAX寄存器的值与ESI寄存器的值进行按位异或运算，并将结果存储在ESI寄存器中

9 movl %ecx, %ebx //将ECX寄存器的值复制到EBX寄存器中

10 shrl %bl, %edi //使用BL寄存器的低8位作为右移的位数，将EDI寄存器的值右移，并将结果存储在EDI寄存器中

11 testl %edi, %edi //将EDI寄存器的值与自身进行按位与运算，并更新标志寄存器

12 jne .L3 //如果标志寄存器中的不等于零标志位为真，则跳转到.L3标号处继续执行

13 movl %esi, %eax //将ESI寄存器的值复制到EAX寄存器中

(2)

参数x存放在EDX寄存器中

参数k存放在ECX寄存器中

局部变量val存放在ESI寄存器中

局部变量i存放在EDI寄存器中

(3)

Val初始值为255,i为-2147483648

(4)

循环终止条件为i为0，在第10条指令中，将EBX寄存器（存储i的值）的低8位作为位移量对EDI寄存器进行右移操作，更新了循环控制变量i的值。

(5)C代码如下：

int

lproc(int x, int k) {

int val = 255;

int i;

for (i = k; i != 0; i = i >> (val & 0xFF)) {

val ^= x;

}

return val;

}

9．假设嵌套的联合数据类型 node 声明如下：

1 union node {

2 struct {

3 int \*ptr;

4 int datal;

5 } n1;

6 struct {

7 int data2;

8 union node \*next;

9 } n2;

10 };

有一个进行链表处理的过程 chain\_proc 的部分 C 代码如下：

1 void chain\_proc(union node \*uptr) {

2 uptr-> \_\_\_\_\_\_\_\_\_\_\_\_ = \*(uptr-> \_\_\_\_\_\_\_\_\_\_\_\_ ) - uptr-> \_\_\_\_\_\_\_\_\_\_\_\_ ;

3 }

过程 chain\_proc 的过程体对应的汇编代码如下：

1 movl 8(%ebp), %edx

2 movl 4(%edx), %ecx

3 movl (%ecx), %eax

4 movl (%eax), %eax

5 subl (%edx), %eax

6 movl %eax, 4(%ecx)

回答下列问题或完成下列任务：

（1）node 类型中结构成员 n1.ptr、n1.data1、n2.data2、n2.next 的偏移量分别是多少？

（2）node 类型总大小占多少字节？

（3）根据汇编代码写出 chain\_proc 的 C 代码中缺失的表达式。

(1)

n1.ptr 的偏移量是 0。

n1.data1 的偏移量是 4字节。

n2.data2 的偏移量是 8字节。

n2.next 的偏移量是 12字节

(2)

根据给定的结构体定义，n1 和 n2 是两个结构体成员，占用的空间相互独立。因此，node 类型的总大小等于两个结构体中占用空间较大的那个。n1和n2占用空间都为8字节，所以node的总大小占8个字节。

(3)

void chain\_proc(union node \*uptr) {

uptr->n2.next->n1.data1 = \*(uptr->n2.next->n1.ptr) - \*(uptr->n2.next->n1.ptr);

}