一.选择

1.（13期中）32 位 x86 计算机、Windows 操作系统下定义的一个 structure S 包含三个部分： double a, int b, char c, 请问 S 在内存空间中最多和最少分别能占据多少个字节(32 位 Windows 系统 按 1 、4 、8 的原则对齐 char、int 、double) ？答：( )

A. 16, 13

B. 16, 16

C. 24, 13

D. 24, 16

2.（14期中）有如下定义的结构， 在 x86-64 下，下述结论中错误的是？ struct {

char c;

union {

char vc;

double value;

int vi;

} u;

int i;

} sa;

1. sizeof(sa) == 24
2. (&sa.i - &sa.u.vi) == 8

C. (&sa.u.vc - &sa.c) == 8

D. 优化成员变量的顺序，可以做到“sizeof(sa) == 16”

3.（14期中）关于如何避免缓冲区溢出带来的程序风险，下述错误的做法为？

A. 编程时定义大的缓冲区数组

B. 编程时避免使用 gets，而采用 fgets

C. 程序运行时随机化栈的偏移地址

D. 在硬件级别引入不可执行代码段的机制

4.（15期中） 假设存储器按“大端法”存储数据对象，已知如下的 C 语言数据结构：union { char c[2]; int i; }; 当 c 的值为 0x01, 0x23 时， i 的值为：

A. 0x0123

B. 0x2301

C. 0x01230000 D.不确定

5.（17期中）有 A 的定义：int A[3][2] = {{1,2}，{3,3}，{2,1}}； 那么 A[2]的值为：

A. &A+16 B. A+16 C. \*A+4 D. \*A+2

6.（17期中） 在 x86-64 架构下，有如下变量：

|  |
| --- |
| union {char c[8], int i;} x; |

在 x.i=0x41424344 时， x.c[2]的值为多少(提示： ‘A’=0x41)：

A. ‘A’ B. ‘B’ C. ‘C’ D. ‘D’

7.（18期中）以下代码输出结果是

union {

double d;

struct {

int i;

char c[4];

} s;

} u;

u.d = 1;

printf("%d\n", u.s.c[2]);

A) 0 B) -16 C) 240 D) 191

8.（18期中）下列关于 C 语言中的结构体(struct)以及联合(union)的说法中，正确的是：

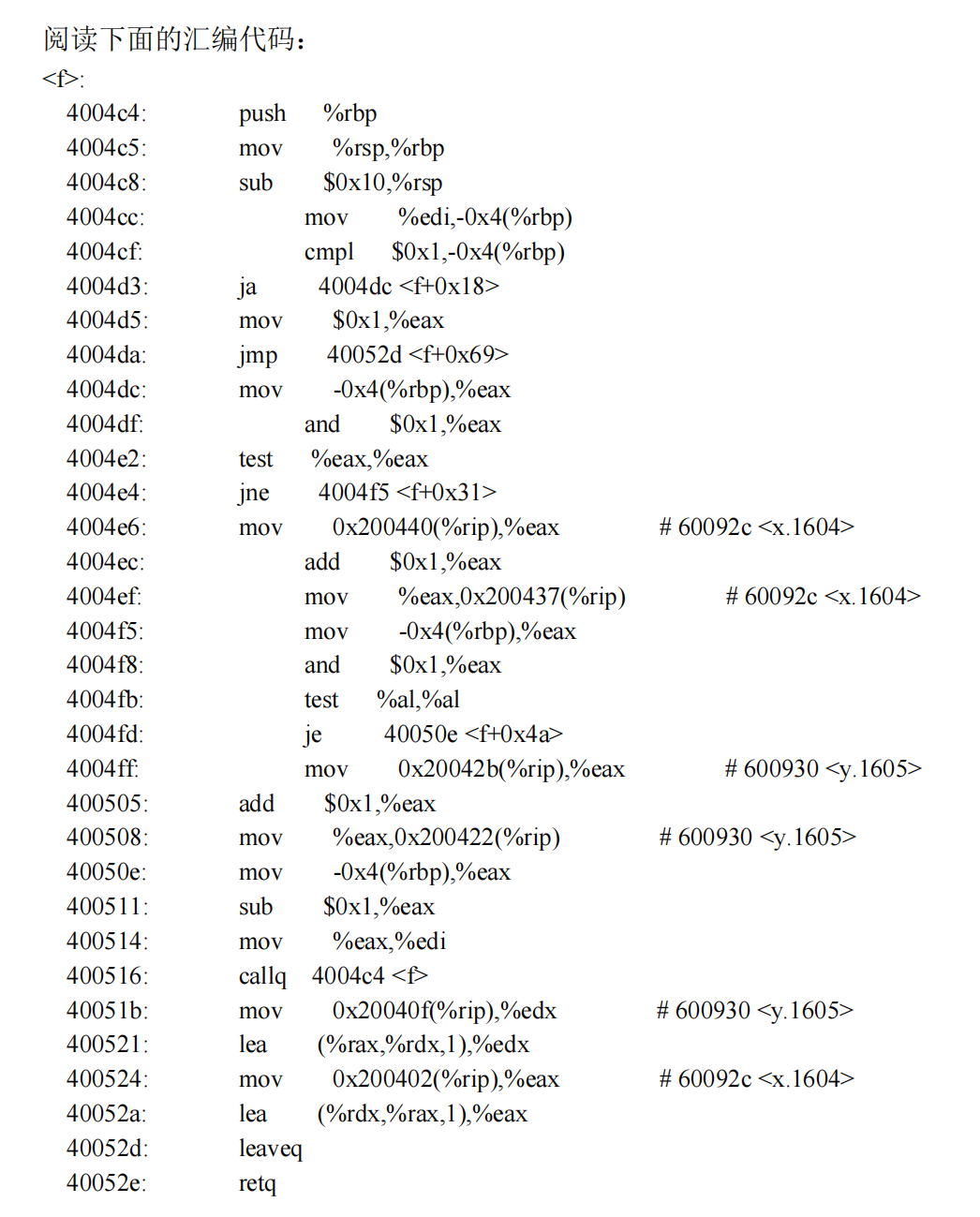
A) 对于任意 struct，将其成员按照其实际占用内存大小从小到大的顺序进行排列 不一定会使之内存占用最小

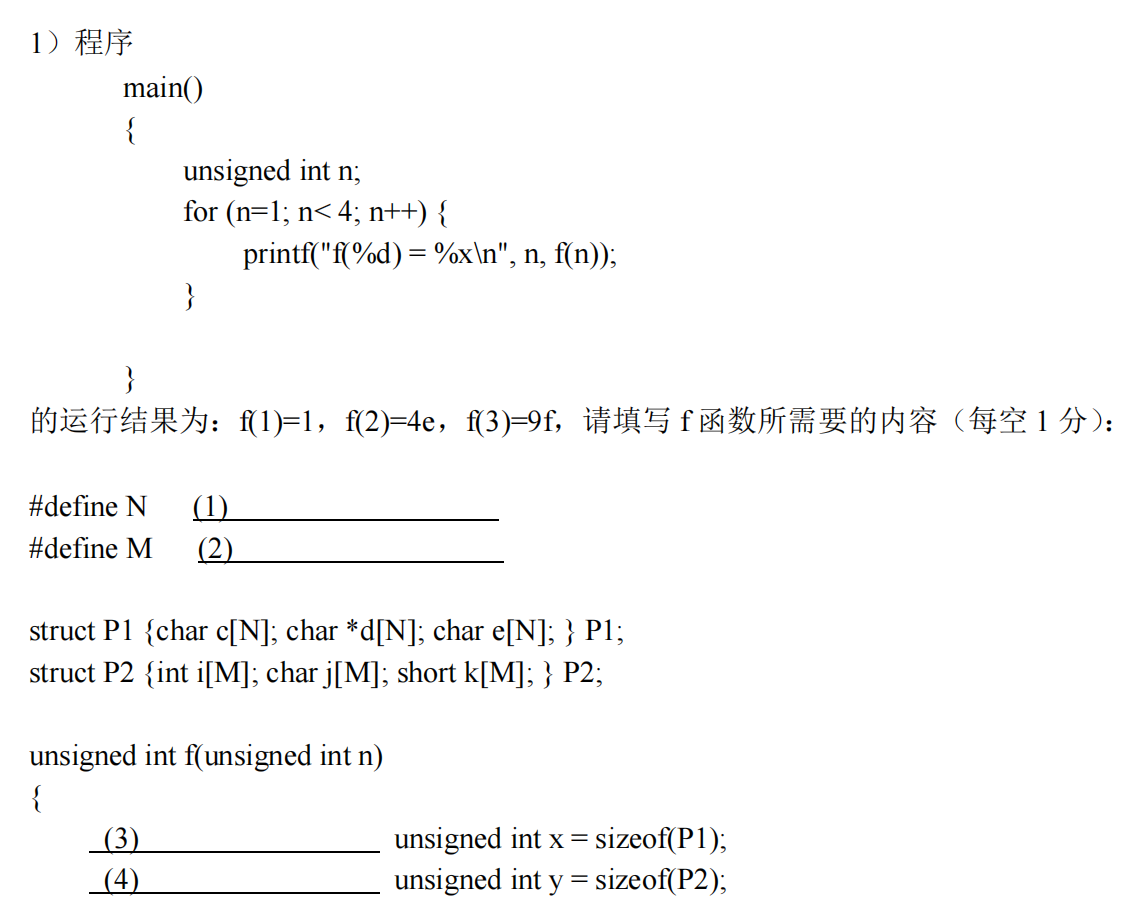
B) 对于任意 struct，将其成员按照其实际占用内存大小从小到大的顺序进行排列 一定不会使之内存占用最大

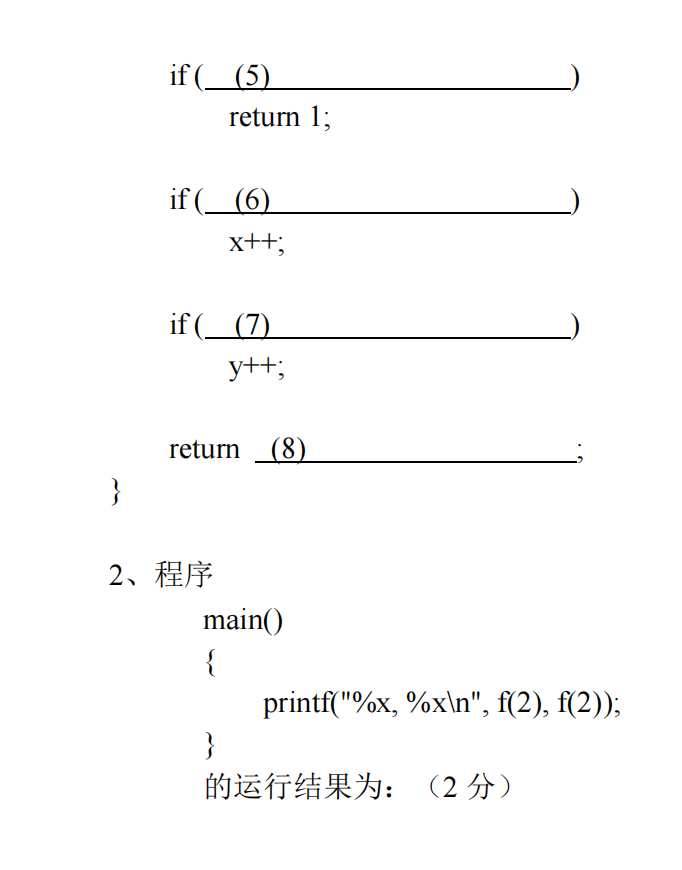
C) 对于任意 union，将其成员按照其实际占用内存大小从小到大的顺序进行排列 不一定会使之内存占用最小

D) 对于任意 union，将其成员按照其实际占用内存大小从小到大的顺序进行排列 一定不会使之内存占用最大

1. 填空与解答
2. （13期中）







2.（15期中）

（1）.考虑下面的union的声明，回答后面的问题。

union ELE {

struct {

int x;

int \*p;

}e1;

struct {

union ELE \* next;

int y;

}e2;

};

(注：32位机器)

这个union总共大小为多少\_\_\_\_字节(2分)

（2）.假设编译器为process的主体产生了如下了代码，请补充完整下面的过程 :

(只有一个不需要任何强制类型转换且不违反任何类型限制的答案)

movl movl movl movl subl movl

8(%ebp),%eax

(%eax),%ecx

4(%ecx),%edx

(%edx),%edx

4(%eax),%edx

%edx,(%ecx)

void process (union ELE \* up)

{

up->\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_=\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ - \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_; }

（3）.请查看下文完成如下功能的汇编代码，定位错误语句并进行更正 : (6 分)

给出 n(在%ebp+8 位置,n>=1),up(在%ebp+12 位置,ELE\* 类型),假设 以\*up 为头元素(设\*up 为第 0 个),由声明中的 next 连接形成了一个链表,请将第 n 个元素(假设链表足够长)的 x 的值放入%eax 中

X86 代码

xorl %ecx,%ecx

movl 8(%edx),%ebp

movl 12(%ebp),%eax

LOOP:

movl (%eax),%eax

add $1,%ecx

test %ecx,%edx

jne LOOP

movl (%eax),%eax

4.阅读下列代码，回答后面的问题

typedef struct {

short x[A][B];

int y;

}str1;

typedef struct {

char array[B];

int t;

short s[B];

int u;

}str2;

void setVal (str1 \*p,str2 \*q) {

int v1=q->t;

int v2=q->u;

p->y=v1+v2;

}

(short 以 2 字节计算)

GCC 为 setVal 的主体产生下面的代码 :

movl movl addl movl movl

请直接写出A和B的值

12(%ebp),%eax

28(%eax),%edx

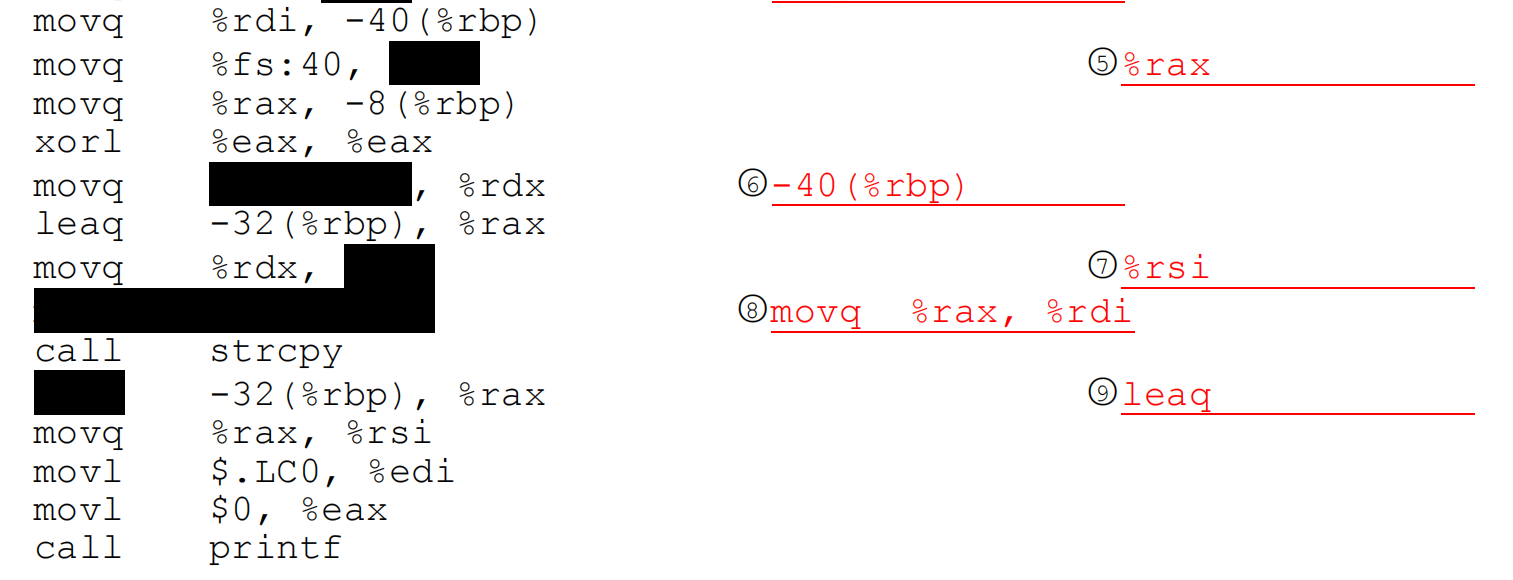
8(%eax),%edx

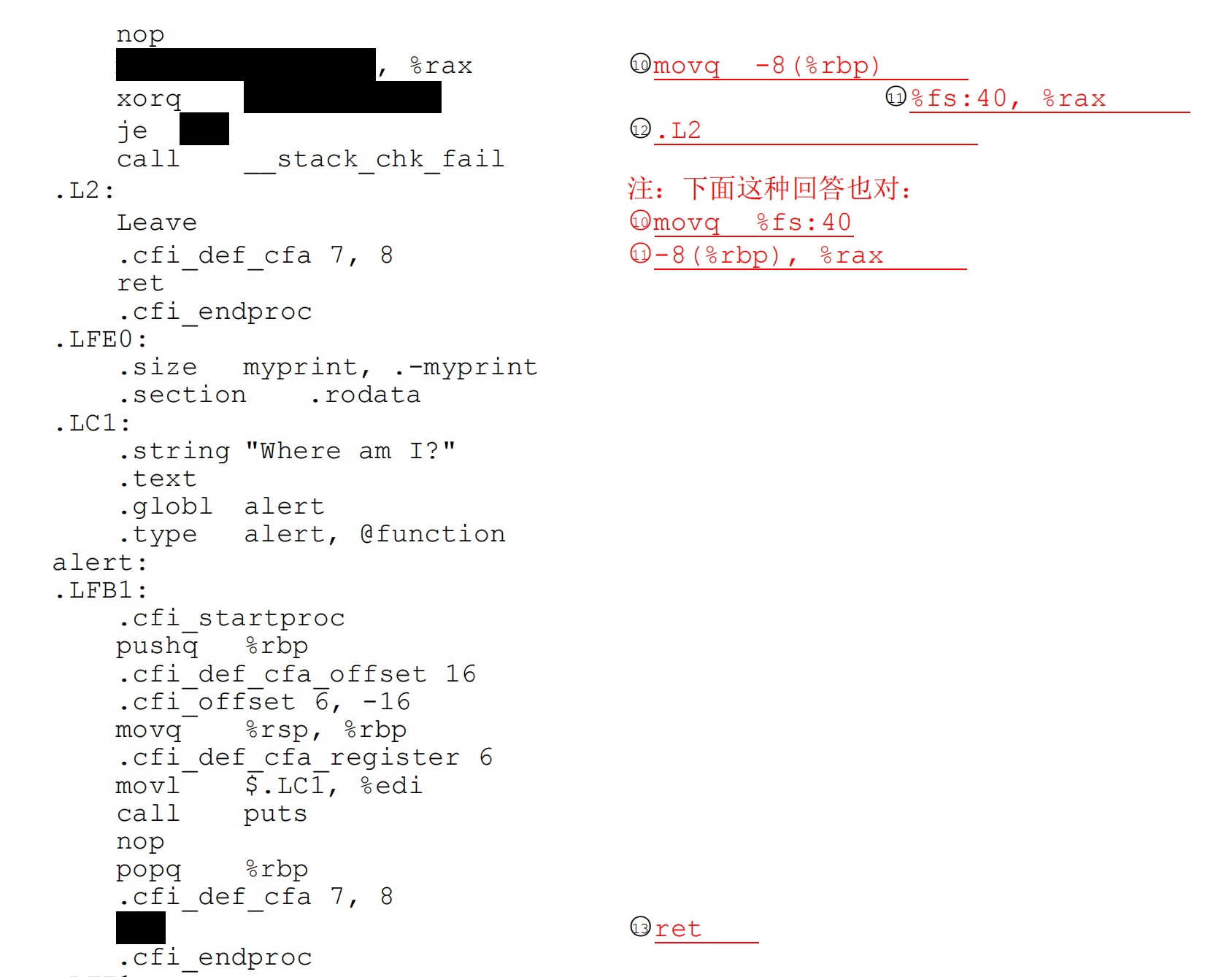
8(%ebp),%eax

%edx,44(%eax)

3.（17期中）









1. （18期中）

数据结构定义如下： (测试用机： 64 位 Linux 机器)

typedef struct s1 {

char cc[N];

int ii[N];

int \*ip;

} S1;

S1 t1[N];

（1）计算该数据结构的空间：

当 N=3 时， sizeof(S1) = (1) ，sizeof(t1) = (2) 当 N=4 时， sizeof(S1) = (3) ，sizeof(t1) = (4) 当 N=5 时， sizeof(S1) = (5) ，sizeof(t1) = (6)

（2）当 N=4 时，该数据结构初始化代码如下：

void init(int n)

{

int i;

for (i=0; i<n; i++) {

t1[i].ip = &(t1[i].ii[i]);

}

}

根据上述代码，填写下面汇编中缺失的内容：

init:

movl $0, %ecx

jmp

.L3:

.L2:

.L2

movslq leaq leaq addq salq movq addl

cmpl jl rep ret

%ecx, %rax

( (1) ,%rax,8), %rsi

0(,%rsi,4), %rdx

$t1+4, %rdx

(2) , %rax

(3) , (4)

$1, %ecx

(5) , %ecx

.L3

（3）当 N=3 时，函数 fun 的汇编代码如下：

fun:

movslq movslq leaq leaq leaq addq movl addq leaq movq movl

%esi, %rax

%edi, %rdi

(%rdi,%rdi), %rdx

(%rdx,%rdi), %r8

(%r8,%r8), %rcx

%rcx, %rax

%esi, t1+4(,%rax,4)

%rdx, %rdi

0(,%rdi,8), %rax

t1+16(%rax), %rax

%esi, (%rax)

ret

根据上述代码，填写函数 fun 的 C 语言代码：

void fun(int x, int y)

{

(1) = (2) ;

(3) = (4) ;}

参考答案：

一．选择

1. D 考虑对齐,windows double 按 8 字节对齐，最长 c, a, b，最短 a,b,c

2.**B**

由于对齐的需求， 在 **x86-64** 下，要保证 **sa.u.value** (**double** 类型变量)的 地址必须 **8**字节对齐，而在 **ia32** 的 **Linux**系统中，可 **4**字节对齐。故 **sizeof(sa)** 在**x86-64**下要占用三个 **8**字节的空间共 **24**字节。**sa.u.vc**与 **sa.c**之间和**sa.i** 与 **sa.u.vi** 之间的地址值之差均为 **8**，但即使不知道 **(&sa.i** **-** **&sa.u.vi)**表 示之间可以放置多少个整数(值为 **2** )， 也可通过优化成员变量顺序后有 **sizeof(sa)** **==** **16**，用排除法得出正确答案。

3. **A**

**B**、**C**、**D** 均为讲义中所提及的解决缓冲区溢出风险的方法。 **A** 策略则无法从根本上 解决缓冲区溢出问题，只要输入足够长数据就仍然可以实现缓冲区溢出攻击。

4.选择 D. 因为 i 的两个高地址字节的值未知

5.C

解析： 参见书 P177 页表格， 机器在计算指针与常数的运算时，会将常数乘以指针 指向的元素大小。&A 常数扩大的倍数为 sizeof(A[3][2]) = 3\*2\*4; A 常数 扩大的倍数为 sizeof(A[0])=2\*4; \*A 常数扩大的倍数为 sizeof(int) = 4。 正确的答案应为 A+2 或\*A+4，故应选择 C

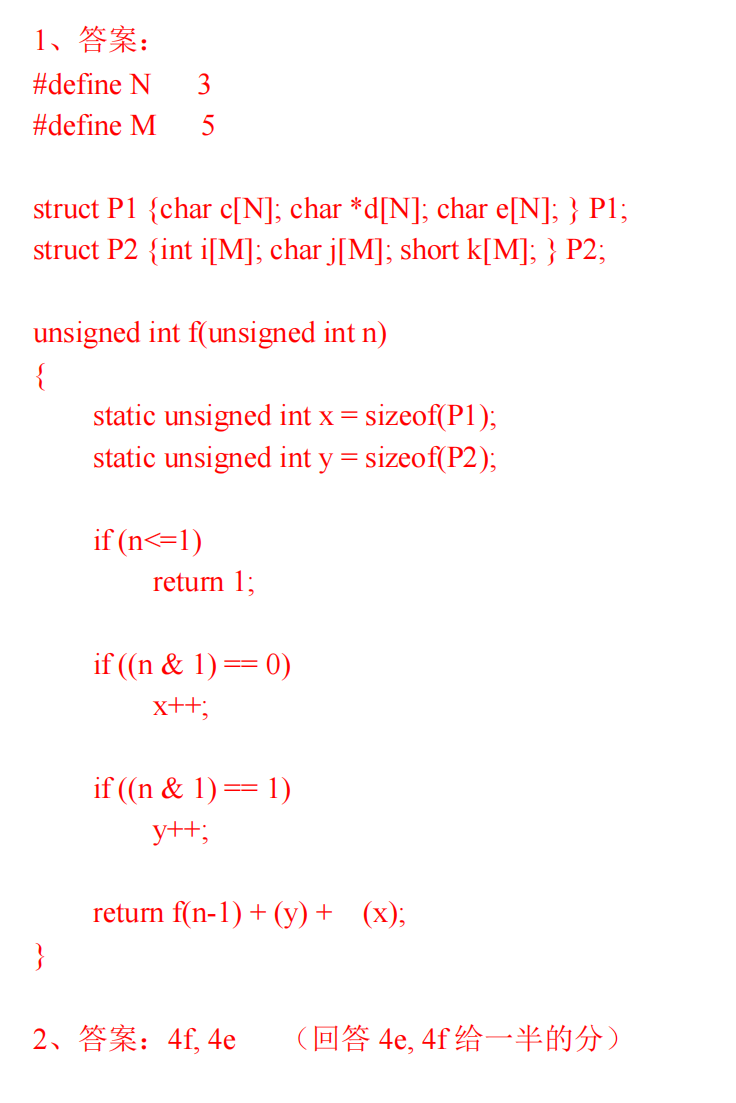
6.B

7.ABC 均正确。题目未说明大小端， 若为大端法， 选择 A。若为小端法，u.s.c[2]为(11110000)2， 由于 char 在不同的编译器上会被实现成 signed char 或 unsigned char，因此扩 展为 int 时-16 和 240 均有可能， B，C 均正确。

8. A。联合以及只含一个基本数据类型成员的结构体的内存占用与其成员排列 方式无关，即任意排列方式都可使得内存占用最小(最大)，故 b、c、d 错误。对 于 struct，应当将成员按照其存储单元所占内存大小从小到大(或从大到小)的顺 序进行排列才能使内存占用最小，故 a 正确。

二．填空与解答

1.



2.

（1）. 8

（2）. up->next->x=\*(up->next->p)- (up->y)

（3）.

X86代码

xorl %ecx,%ecx

movl movl

LOOP:

movl (%eax),%eax

add $1,%ecx

test %ecx,%edx

jne LOOP

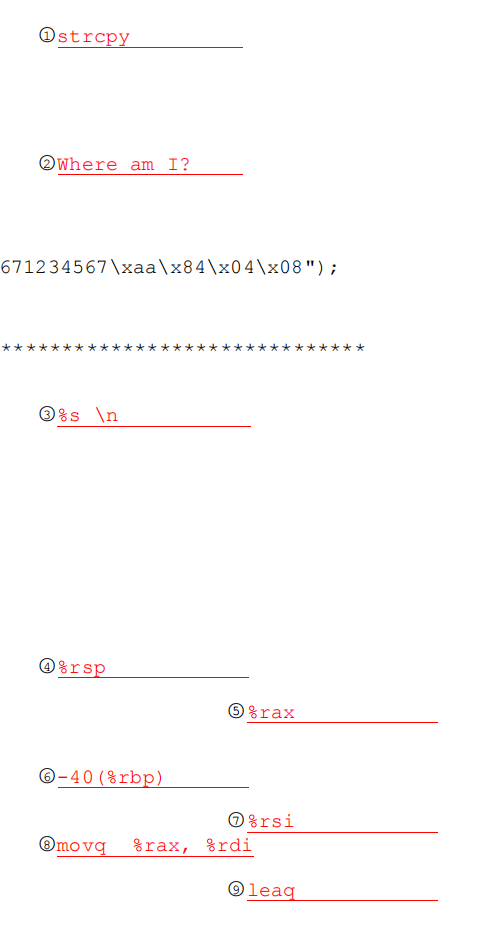
movl (%eax),%eax

（4）.A=3 B=7

8(%edx),%ebp 12(%ebp),%eax

||应为： movl 8(%ebp),%ed

3.



4.

1）(1) 24 (2) 72

(3) 32 (4) 128

(5) 40 (6) 200

2）(1) %rax (2) $5 (3) %rdx (4) t1+24(%rax) (5) %edi

3）

(1) t1[x].ii[y]

(2) y

(3) \*(t1[x].ip)

(4) y-\*\*