

Homework 5

Problem 1

Consider the following declaration:

```
short    S[5];
short    *T[4];
int      **U[6];
double   V[8];
double   *W[7];
```

Fill in the table below:

Array	Size of Element	Size of Array	Start Address	Address of i-th Element
S	2	10	x_S	$x_S + 2i$
T	4	16	x_T	$x_T + 4i$
U	4	24	x_U	$x_U + 4i$
V	8	64	x_V	$x_V + 8i$
W	4	28	x_W	$x_W + 4i$

Problem 2

Given a C function

```
1  int proc(void)
2  {
3      int x, y;
4      scanf("%x %x", &y, &x);
5      return x-y;
6  }
```

GCC generates following assembly codes:

```
1  proc:
2      pushl   %ebp
3      movl    %esp, %ebp
4      subl    $24, %esp
5      addl    $-4, %esp
6      leal    -4(%ebp), %eax
7      pushl   %eax
8      leal    -8(%ebp), %eax
9      pushl   %eax
10     pushl   $.LC0          ;Pointer to string "%x %x"
```

```

11    call    scanf
12    movl    -8(%ebp), %eax
13    movl    -4(%ebp), %edx
14    subl    %eax, %edx
15    movl    %edx, %eax
16    leave
17    ret

```

Assume stack pointer register and frame pointer register value are set as follows at the beginning of `proc()` :

`%esp` `0x800040`

`%ebp` `0x800060`

And `scanf` reads `0x46` and `0x53` from `stdin`. String “`%x %x`” is stored at `0x300070` in memory.

Questions:

- In line 3, what value is assigned to `%ebp`
`0x80003C`
- Where are local variables `x` and `y` stored (Indicate their addresses)
`x: 0x800038`
`y: 0x800034`
- What is the value of `%esp` after line 10
`0x800014`
- Draw a diagram to illustrate the stack frame after `scanf` returns

<code>0x80003C</code>	<code>0x800060</code>	← <code>%ebp</code>
<code>0x800038</code>	<code>0x53</code>	<code>x</code>
<code>0x800034</code>	<code>0x46</code>	<code>y</code>
<code>0x800030</code>		
<code>0x80002C</code>		
<code>0x800028</code>		
<code>0x800024</code>		
<code>0x800020</code>		
<code>0x80001C</code>	<code>0x800038</code>	<code>&x</code>
<code>0x800018</code>	<code>0x800034</code>	<code>&y</code>
<code>0x800014</code>	<code>0x300070</code>	← <code>%esp</code>

- Indicate the unused stack frame of `proc()`. (p.s. these areas are allocated to improve the performance of high-speed cache)

`0x800020 ~ 0x800033` are unused.

Problem 3

Consider the following codes:

```

1  int mat1[M][N];

```

```

2  int mat2[N][M];
3
4  int sum_element(int i, int j)
5  {
6      return mat1[i][j] + mat2[j][i];
7  }

```

GCC generates the following assembly codes:

```

1      movl    8(%ebp), %ecx
2      movl    12(%ebp), %eax
3      leal    0(,%eax,4), %ebx
4      leal    0(,%ecx,8), %edx
5      subl    %ecx, %edx
6      addl    %ebx, %eax
7      sall    $2, %eax
8      movl    mat2(%eax,%ecx,4), %eax
9      addl    mat1(%ebx,%edx,4), %eax

```

Use reverse engineering techniques to determine the values of M and N according to assembly codes above.

M=5

N=7

Problem 4

Suppose you are responsible for maintaining a C program and encounter the following codes:

```

1  typedef    struct {
2      int left;
3      a_struct a[CNT];
4      int right;
5  } b_struct;
6
7  void test(int i, b_struct *bp)
8  {
9      int n = bp->left + bp->right;
10     a_struct *ap = &bp->a[i];
11     ap->x[ap->idx] = n;
12 }

```

Unfortunately, you are not permitted to access “.h” file which defines constant CNT and struct a_struct. But you can access “.o” file and use *objdump* to disassemble the object file. Disassembled codes are displayed as follows:

```

1 00000000    <test>:
2  0:      55                      pushl    %ebp
3  1:      89 e5                      movl     %esp, %ebp

```

```

4  3:      53                pushl    %ebx
5  4:      8b 45 08          movl     0x8(%ebp), %eax
6  7:      8b 4d 0c          movl     0xc(%ebp), %ecx
7  a:      8d 04 80          leal     (%eax, %eax, 4), %eax
8  d:      8d 44 81 04       leal     0x4(%ecx, %eax, 4), %eax
9  11:     8b 10             movl     (%eax), %edx
10 13:     c1 e2 02          shll     $0x2, %edx
11 16:     8b 99 b8 00 00 00 movl     0xb8(%ecx), %ebx
12 1c:     03 19             addl     (%ecx), %ebx
13 1e:     89 5c 02 04       movl     %ebx, 0x4(%edx, %eax, 1)
14 22:     5b               popl     %ebx
15 23:     89 ec             movl     %ebp, %esp
16 25:     5d               popl     %ebp
17 26:     c3               ret

```

Use reverse engineering techniques to get following values:

A. Value of CNT;

CNT = 9

B. Declaration of struct a_struct, assuming that a_struct contains only idx and x;

```

struct a_struct {
    int idx;
    int x[4];
}

```

结构体中成员的声明顺序不能颠倒。