ICS'12 Homework 4

1. Translate the following assembly into C codes. Note that you can name local variables represented by 28(%esp), 24(%esp)… freely as you like.

```
    movl    $1, 28(%esp)
    movl    $1, 24(%esp)
    movl    $10, 16(%esp)
    movl    $3, 20(%esp)
    jmp L2
L3:
    movl    28(%esp), %eax
    movl    %eax, 12(%esp)
    movl    24(%esp), %eax
    movl    %eax, 28(%esp)
    movl    12(%esp), %eax
    addl    %eax, 24(%esp)
    incl    20(%esp)
L2:
    cmpl    $10, 20(%esp)
    jle   L3
    (Following code omitted…)
```

参考答案（不唯一）：

计算第 10 个斐波那契数。

```
int a = 1;
int b = 1;
int n = 10;
int i;
for (i = 3; i <= 10; ++i) {
    int temp = a;
    a = b;
    b += temp;
}
```

2. Translate the following switch statements into assembly **using jump table**.

```
int x = <some value>;
int result = 0;
switch (x) {
    case 64:
```

```
        result = x + x;
        break;
    case 66:
        result = x + 10;
        break;
    case 68:
        result = x * 2;
    // Notice: there is no break here!
    case 69:
        result = result + 1;
        break;
    default:
        result = 3;
        break;
}
```

答案（不唯一）：

Jump Table 部分：标签名称不唯一，但必须与后面的代码一一对应。

```
    .section .rdata,"dr"
    .align 4
L9:
    .long   L3
    .long   L2
    .long   L4
    .long   L2
    .long   L5
    .long   L6
```

代码部分：[x]与[result]可以被替换成 x(%esp)形式或者某些可用的寄存器，只要能够代表这两个变量即可，形式不重要。

```
    .text
L3:
    movl    [x], %eax
    addl    %eax, %eax
    movl    %eax, [result]
    jmp L11
L4:
    movl    [x], %eax
    addl    $10, %eax
    movl    %eax, [result]
    jmp L11
L5:
    movl    [x], %eax
    sall    %eax
    movl    %eax, [result]
L6:
    incl    [result]
    jmp L11
L2:
    movl    $3, [result]
    nop
    jmp L11
L11:
    <the following codes omitted...>
```

3. Describe why and how stack frames are chained. You could present a graph for illustration.

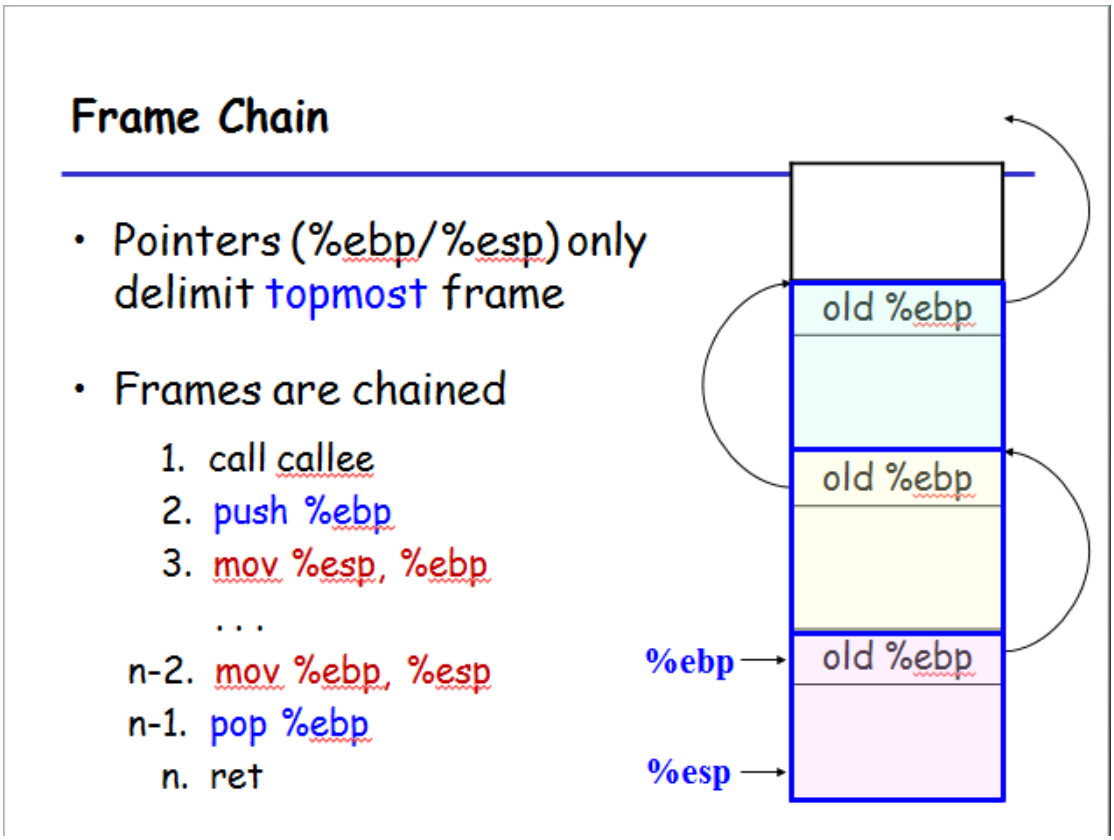
答案（不唯一，大致意思对即可）：

为什么需要对帧（frame）进行链接？

因为帧的连接确保了函数调用结束的顺序与其被调用的顺序恰好相反（正如一个链“环环相扣”的特征）。esp 与 ebp 寄存器是链接的关键，它们配合调用栈，能够实现调用开始时保存调用者状态，调用结

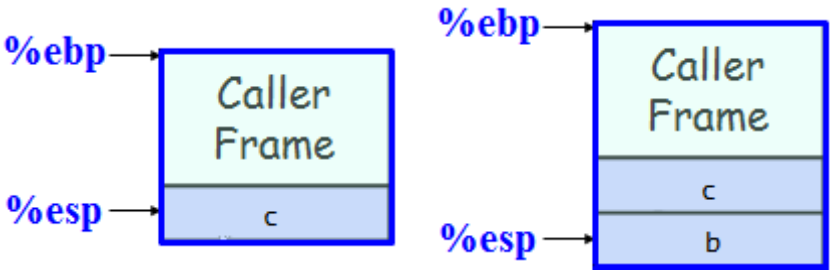
束时恢复调用者状态，以使其继续正常执行。

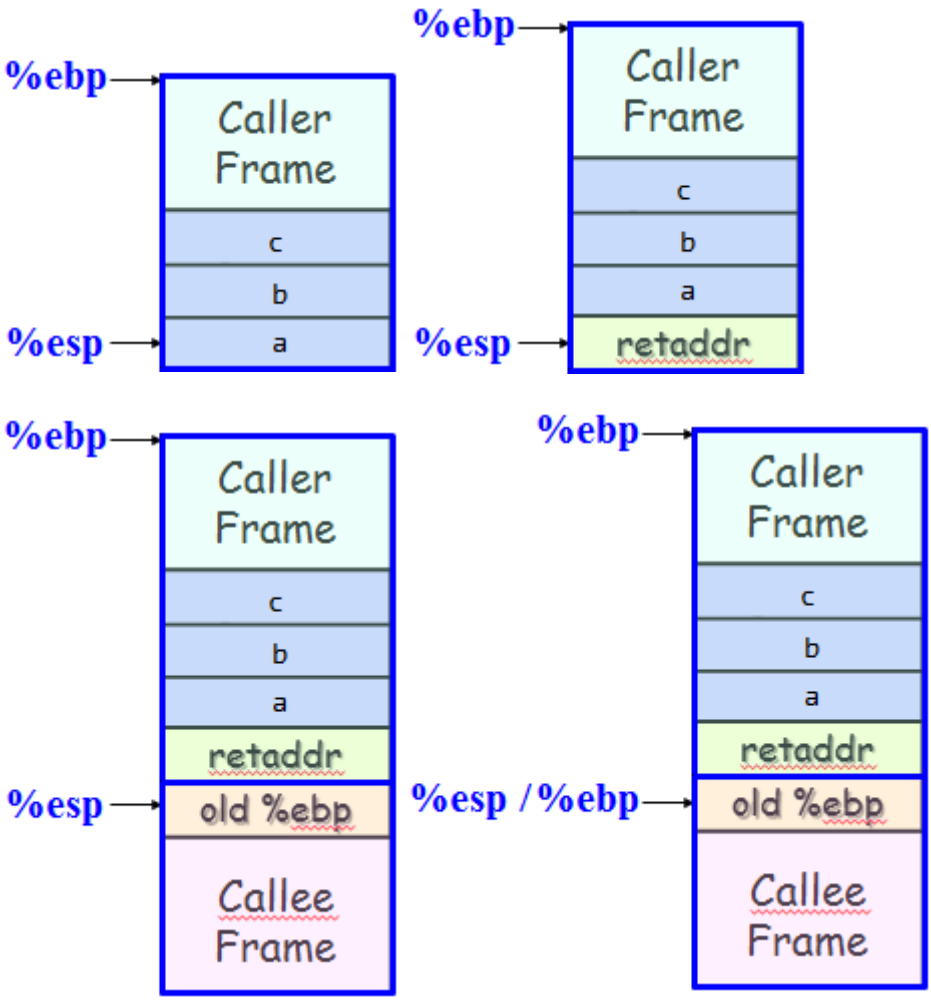
如何实现帧的链接？



4. Draw a stack and indicate where %ebp and %esp points **for each step** when the following function is called. Here is an example where the first step takes place.

int foo(int a, int b, int c);





Programming Exercises

Here are another two C programming exercises which you should also finish this week. And you should submit your answers to SVN server *svn://ipads.se.sjtu.edu.cn/ics-se11/ics<student no.>/tut1*.

The filename for the two problems should be ex4-1.c and ex4-2.c respectively. **Do not write them on your exercise book!**

1. Implement a function *reverse(s)* that gives a reverse string of the original string. You should give your own test example in *main()*. **Do not**

modify the original string.

```
/*
 * example: reverse("abc") should return "cba".
 */
char *reverse(const char *s);
```

答案（不唯一）：

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

char *reverse(const char *s)
{
    size_t len = strlen(s);
    char *result = malloc(len + 1);
    result += len;
    *result = '\0';
    while (*s)
        *--result = *s++;
    return result;
}

int main()
{
    printf("%s\n", reverse(""));
    printf("%s\n", reverse("x"));
    printf("%s\n", reverse("abcdefg"));
    return 0;
}
```

2. Comple the following C codes that generates a 10x10 matrix (a_{ij}),

where $a_{ij} = i+j$.

Hint:

(1) Use *malloc()* or *calloc()* for space allocation.

(2) ‘*int a[M][N]; p = a*’ or ‘*int a[M]; p[0]=a*’ are both illegal C codes.

(3) You should not receive any compiler warnings like ‘incompatible pointer assignment’.

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
   int **p = NULL;

   // Your codes here.

   // Now p should point to the matrix and the following code will be working.
   int i, j;
   for (i = 0; i < 10; ++i) {
      for (j = 0; j < 10; ++j)
        printf("%d\t", p[i][j]);
      printf("\n");
   }
   return 0;
}
```

答案：（应基本一致）

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int **p = NULL;
    p = malloc(10 * sizeof(int *));
    int i, j;
    for (i = 0; i < 10; ++i) {
        p[i] = malloc(10 * sizeof(int));
        for (j = 0; j < 10; ++j)
            p[i][j] = i + j;
    }

    for (i = 0; i < 10; ++i) {
        for (j = 0; j < 10; ++j)
    printf("%d\t", p[i][j]);
        printf("\n");
    }
    return 0;
}
```

输出结果：

```
0	1	2	3	4	5	6	7	8	9
1	2	3	4	5	6	7	8	9	10
2	3	4	5	6	7	8	9	10	11
3	4	5	6	7	8	9	10	11	12
4	5	6	7	8	9	10	11	12	13
5	6	7	8	9	10	11	12	13	14
6	7	8	9	10	11	12	13	14	15
7	8	9	10	11	12	13	14	15	16
8	9	10	11	12	13	14	15	16	17
9	10	11	12	13	14	15	16	17	18
```