Homework 6

PB17000297 罗晏宸

October 23 2019

Exercise 1

针对如下 C 程序及其在 i386 Linux 下的汇编代码（片段）：

```c
#include<stdio.h>
union var{
    char c[5];
    int i;
};
int main(){
    union var data;
    char *c;
    data.c[0] = '2';
    data.c[1] = '0';
    data.c[2] = '1';
    data.c[3] = '6';
    data.c[4] = '\0';
    c = (char*)&data;
    printf("%x %s\n", data.i, c);
    return 0;
}
// 第一题 C 程序
```

```
    .section .rodata
.LC0:
    .string "%x %s\n"
    .text
.globl main
    .type   main, @function
main:
    _____________________
    movl    %esp, %ebp
    subl    $40, %esp
    andl    $-16, %esp
    movl    $0, %eax
    subl    %eax, %esp
    movb    $50, -24(%ebp)
    _____________________
    _____________________
    _____________________
    _____________________
    _____________________
    movl    %eax, -28(%ebp)
    _____________________
    _____________________
    _____________________
    pushl   $.LC0
    call    printf
    addl    $16, %esp
    _____________________
    leave
    ret
// 第一题汇编程序
```

(a) 上述 C 程序的输出是什么？

(b) 补全 10 处划线部分的汇编代码。

解

(a) 程序的输出结果为

```
36313032 2016
```

(b) 补全后的汇编代码如下

```
.section .rodata
.LC0:
    .string "%x %s\n"
    .text
.globl main
    .type   main, @function
main:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $40, %esp
    andl    $-16, %esp
    movl    $0, %eax
    subl    %eax, %esp
    movb    $50, -24(%ebp)
    movb    $48, -23(%ebp)
    movb    $49, -22(%ebp)
    movb    $54, -21(%ebp)
    movb    $0, -20(%ebp)
    leal    -24(%ebp), %eax
    movl    %eax, -28(%ebp)
    movl    -24(%ebp), %eax
    movl    -28(%ebp), %edx
    movl    %edx, %esp
    pushl   $.LC0
    call    printf
    addl    $16, %esp
    popl    %esp
    leave
    ret
// 补全后的第一题汇编程序
```

Exercise 2

针对如下 C 程序及其汇编代码（片段）：

(a) 补全划线处的汇编代码；

(b) 从运行时环境看，`addl $16, %esp`和`leal -8(%ebp), %esp`这两条汇编指令的作用是什么？

(c) 结合上述两种汇编代码，简述编译器在按值传递结构变量时的处理方式。

```
#define N 2
// #define N 11
typedef struct POINT {
    int x, y ;
    char z[ N ];
    struct POINT *next;
} DOT;
void f(DOT p)
{
    p.x = 100;
    p.y = sizeof(p);
    p.z[1] = 'A';
    f(*(p.next));
}
// 第二题 C 程序
```

```
    .file "test1.c"
    .text
.globl f
    .type   f,@function
f:
    pushl   %ebp
    movl    %esp, %ebp
    movl    $100, 8(%ebp)
    movl    $16, 12(%ebp)
    movb    $65, _________
    movl    ________, %eax
    pushl   _______________
    pushl   _______________
    pushl   _______________
    pushl   _______________
    call    f
    addl    $16, %esp
    leave
    ret
// 当 N=2 时，生成的汇编代码片段。
```

```
    .file "test1.c"
    .text
.globl f
    .type   f,@function
f:
    pushl   %ebp
    movl    %esp, %ebp
    pushl   %edi
    pushl   %esi
    movl    $100, 8(%ebp)
    movl    ___, 12(%ebp)
    movb    $65, _______
    subl    $8, %esp
    movl    ______, %eax
    subl    $24, %esp
    movl    %esp, %edi
    movl    %eax, %esi
    cld
    movl    ______, %eax
    movl    %eax, %ecx
    rep
    movsl
    call    f
    addl    $32, %esp
    leal    -8(%ebp), %esp
    popl    %esi
    popl    %edi
    leave
    ret
// rep movsl 为数据传送指令，即，由源
    地址 esi 开始的 ecx 个字的数据传送
    到由 edi 指示的目的地址。
// 当 N=11 时，生成的汇编代码片段
```

解

(a) 补全后的汇编代码分别如下

```
    .file "test1.c"
    .text
.globl f
    .type   f,@function
f:
    pushl   %ebp
    movl    %esp, %ebp
    movl    $100, 8(%ebp)
    movl    $16, 12(%ebp)
    movb    $65, 17(%ebp)
    movl    20(%ebp), %eax
    pushl   8(%ebp)
    pushl   12(%ebp)
    pushl   16(%ebp)
    pushl   17(%ebp)
    call    f
    addl    $16, %esp
    leave
    ret
// 当 N=2 时，补全后的汇编代码片段。
```

```
    .file "test1.c"
    .text
.globl f
    .type   f,@function
f:
    pushl   %ebp
    movl    %esp, %ebp
    pushl   %edi
    pushl   %esi
    movl    $100, 8(%ebp)
    movl    $24, 12(%ebp)
    movb    $65, 17(%ebp)
    subl    $8, %esp
    movl    $6, %eax
    subl    $24, %esp
    movl    %esp, %edi
    movl    %eax, %esi
    cld
    movl    $34, %eax
    movl    %eax, %ecx
    rep
    movsl
    call    f
    addl    $32, %esp
    leal    -8(%ebp), %esp
    popl    %esi
    popl    %edi
    leave
    ret
// 当 N=11 时，补全后的汇编代码片段
```

(b) `addl $16, %esp`指令将栈顶指针恢复到参数压栈之前的位置，`leal -8(%ebp), %esp`加载栈顶指针地址为`%ebp`的值减去 8

(c) 编译器在按值传递结构变量时，会以处理局部变量的方式，为结构体成员变量分配空间。在递归调用中，完成实参的值压栈并调用递归函数后，将恢复栈顶指针到参数压栈前的位置。

Exercise 3

针对如下 C 程序及其汇编代码（片段）：

```
void g(int**);
int main()
{
    int line[10], i;
    int *p = line;
    for (i = 0; i < 10; i++)
    {
        *p = i;
        g(&p);
    }
    return 0;
}
void g(int**p)
{
    (**p)++;
    (*p)++;
}
// 第三题 C 程序
```

```
.globl g
    .type   g,@function
g:
    pushl   %ebp
    movl    %esp, %ebp
    movl     ④ , %eax
    movl     ⑤ , %eax
             ⑥
    movl     ⑦ , %eax
             ⑧
    leave
    ret
// 第三题函数 g 的汇编代码片段
```

```
    .file "p.c"
    .text
.globl main
    .type   main,@function
main:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $72, %esp
    andl    $-16, %esp
    movl    $0, %eax
    subl    %eax, %esp
    leal    -56(%ebp), %eax
    movl    %eax, -64(%ebp)
    movl    $0, -60(%ebp)
.L2:
            ①
    jle     .L5
    jmp     .L3
.L5:
    movl    -64(%ebp), %edx
    movl    -60(%ebp), %eax
    movl    %eax, (%edx)
    subl    $12, %esp
    leal    -64(%ebp), %eax
    pushl   %eax
    call    g
            ②
    leal    -60(%ebp), %eax
    incl    (%eax)
            ③
.L3:
    movl    $0, %eax
    leave
    ret
// 第三题函数 main 的汇编代码片段
```

(a) 补全下划线处的空白汇编代码;

(b) main 函数中 for 循环结束时，数组 line 各元素值是多少？

解

(a) 补全后的汇编代码分别如下

```
.globl g
    .type   g,@function
g:
    pushl   %ebp
    movl    %esp, %ebp
    movl   ④8(%ebp), %eax
    movl   ⑤(%eax), %eax
  ⑥movl    -1(%eax), (%eax)
    movl   ⑦8(%ebp), %eax
  ⑧movl    (%eax), %eax
    leave
    ret
// 补全后第三题函数 g 的汇编代码片段
```

```
    .file "p.c"
    .text
.globl main
    .type   main,@function
main:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $72, %esp
    andl    $-16, %esp
    movl    $0, %eax
    subl    %eax, %esp
    leal    -56(%ebp), %eax
    movl    %eax, -64(%ebp)
    movl    $0, -60(%ebp)
.L2:
  ①cmpl     $9, -60(%ebp)
    jle     .L5
    jmp     .L3
.L5:
    movl    -64(%ebp), %edx
    movl    -60(%ebp), %eax
    movl    %eax, (%edx)
    subl    $12, %esp
    leal    -64(%ebp), %eax
    pushl   %eax
    call    g
  ②movl     %eax, (%ebp)
    leal    -60(%ebp), %eax
    incl    (%eax)
  ③jmpl     .L2
.L3:
    movl    $0, %eax
    leave
    ret
// 补全后第三题函数 main 的汇编代码片段
```

(b) main 函数中 for 循环结束时，数组 line 各元素值如下

```
line[0] = 1
line[1] = 2
line[2] = 3
line[3] = 4
line[4] = 5
line[5] = 6
line[6] = 7
line[7] = 8
line[8] = 9
line[9] = 10
```

Exercise 4

针对如下 C 程序及其汇编代码（片段）：

```c
#include <stdio.h>
int main()
{
    int a = 0, b = 0;
    { int a = 1; }
    { int b = 2;
        { int a = 3; }
    }
    return 0;
}// 第四题 C 程序
```

```
main:
    pushl   %ebp
    _____________________
    subl    $24, %esp
    andl    $-16, %esp
    movl    $0, %eax
    subl    %eax, %esp
    movl    $0, _________
    movl    $0, _________
    movl    $1, _________
    movl    $2, -12(%ebp)
    movl    $3, _________
    movl    $0, %eax
    leave
    ret
// 第四题汇编代码片段
```

(a) 补全下划线处的空白汇编代码；

(b) 描述所用编译器对 C 分程序所声明变量的存储分配策略；

解

(a) 补全后的汇编代码如下

```
main:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $24, %esp
    andl    $-16, %esp
    movl    $0, %eax
    subl    %eax, %esp
    movl    $0, %ebp
    movl    $0, -4(%ebp)
    movl    $1, -8(%ebp)
    movl    $2, -12(%ebp)
    movl    $3, -16(%ebp)
    movl    $0, %eax
    leave
    ret
// 补全后第四题汇编代码片段
```

(b) 对于过程中出现的作用域不同的局部变量，编译器按出现次序为其在堆栈中分配空间。

Exercise 5

仔细阅读所给 C 程序及其汇编代码片段。

(a) 指出波浪线处的汇编代码的作用；

(b) 补全下划线处的空白汇编代码。

```c
#include <stdio.h>
int main()
{
    int a[6] = {0, 1, 2, 3, 4, 5};
    int i = 6, j = 7;
    int *p = (int*)(&a + 1);
    printf("%d\n",*(p - 1));
    return 0;
} // 第五题 C 程序
```

```
.LC0:
    .long   0
    .long   1
    .long   2
    .long   3
    .long   4
    .long   5
.LC1:
    .string "%d\n"
    .text
.globl main
    .type   main, @function
main:
    pushl   %ebp
    movl    %esp, %ebp
    pushl   %edi
    pushl   %esi
    subl    $48, %esp
    andl    $-16, %esp
    movl    $0, %eax
    subl    %eax, %esp
    leal    -40(%ebp), %edi
    movl    $.LC0, %esi
    cld
    movl    $6, %eax
    movl    %eax, %ecx
    rep
    movsl
    movl    $6, -44(%ebp)
    movl    $7, -48(%ebp)
    leal    -40(%ebp), %eax
    addl    _______________
    movl    %eax, -52(%ebp)
    subl    $8, %esp
    movl    -52(%ebp), %eax
    subl    $_________, %eax
    pushl   _______________
    pushl   $.LC1
    call    printf
    addl    $_________ , %esp
    movl    $0, %eax
    leal    __________, %esp
    popl    %esi
    popl    %edi
    leave
    ret
// 第五题汇编代码片段
```

解

(a) 将从源地址`$.LC0`开始的 6 个字的数据，即`0, 1, 2, 3, 4, 5`传送到地址为`%ebp - 40`即栈顶的存储单元中。

(b) 补全后的汇编代码如下

```
.LC0:
    .long   0
    .long   1
    .long   2
    .long   3
    .long   4
    .long   5
.LC1:
    .string "%d\n"
    .text
.globl main
    .type   main, @function
main:
    pushl   %ebp
    movl    %esp, %ebp
    pushl   %edi
    pushl   %esi
    subl    $48, %esp
    andl    $-16, %esp
    movl    $0, %eax
    subl    %eax, %esp
    leal    -40(%ebp), %edi
    movl    $.LC0, %esi
    cld
    movl    $6, %eax
    movl    %eax, %ecx
    rep
    movsl
    movl    $6, -44(%ebp)
    movl    $7, -48(%ebp)
    leal    -40(%ebp), %eax
    addl    $24, %eax
    movl    %eax, -52(%ebp)
    subl    $8, %esp
    movl    -52(%ebp), %eax
    subl    $4, %eax
    pushl   (%eax)
    pushl   $.LC1
    call    printf
    addl    $8 , %esp
    movl    $0, %eax
    leal    -12(%ebp), %esp
    popl    %esi
    popl    %edi
    leave
    ret
// 补全后的第五题汇编代码片段
```

Exercise 6

假设以下假想的程序采用静态嵌套作用域规则：

```
program staticLink
    procedure f(level, arg())
    // 函数 f 有两个参数，整型变量 level，无参函数 arg

        procedure local() // 嵌套在 f 中的函数
        begin // 无参函数 local，返回一个整型值。
            return level;
        end

    begin // f 函数体
        if (level > 10) return f(level - 1, local);
        else if (level > 1) return f(level - 1, arg); else return arg();
    end

    procedure dummy()
    begin
        /* 空的函数体 */
    end

begin // staticLink 函数体
    print(f(17, dummy));
end
```

(a) 给出该程序运行结果；

(b) 给出函数调用f(17, dummy)执行时运行栈上包含活动记录最多时的相关图示。假设按照逆序方式传递参数；函数作为参数传递时，需要两个单元，一为函数入口地址（可用函数名表示），二为其访问链。

解

(a) 程序的输出结果为

```
11
```

(b)

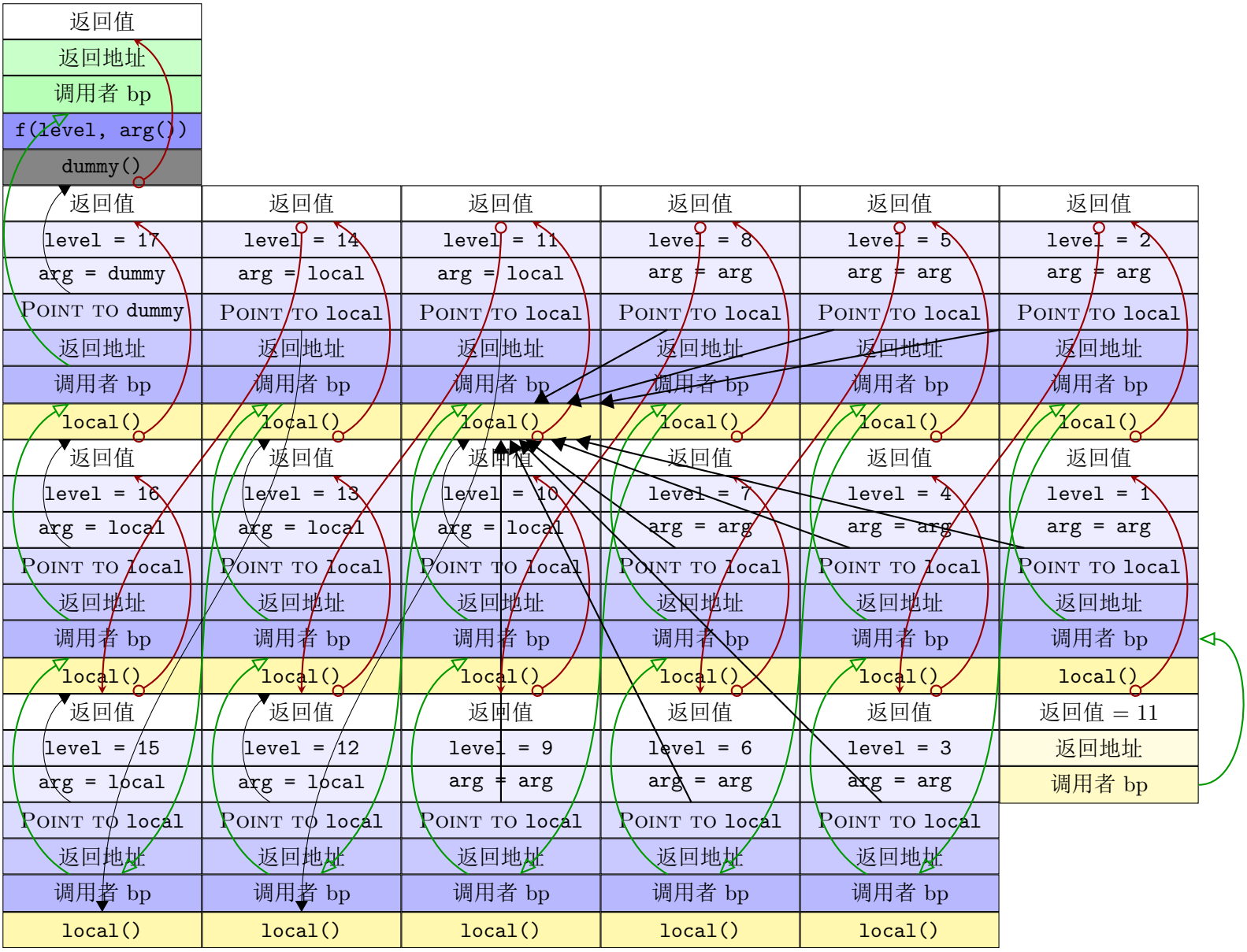

图 1: 栈上包含 19 个活动记录表时的图示