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):

```
#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
```

- (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。

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

(a) 程序的输出结果为

36313032 2016

(b) 补全后的汇编代码如下

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

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
         f,@function
   .type
f:
   pushl
           %ebp
           %esp, %ebp
   movl
   movl
           $100, 8(%ebp)
           $16, 12(%ebp)
   movl
           $65, ___
   movb
           , %eax
   movl
   pushl
   pushl
   pushl
   pushl
   call
   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:
            %ebp
    pushl
            %esp, %ebp
    movl
            %edi
    pushl
    pushl
            %esi
    movl
            $100, 8(%ebp)
            ____, 12(%ebp)
    movl
    movb
            $65, _
            $8, %esp
    subl
            _____, %eax
    movl
            $24, %esp
    subl
            %esp, %edi
%eax, %esi
    movl
    movl
    cld
    movl
                ___, %eax
            %eax, %ecx
    movl
    rep
    movsl
    call
            $32, %esp
    addl
            -8(%ebp), %esp
    leal
            %esi
    popl
            %edi
    popl
    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
            f,@function
    .type
f:
            %ebp
    pushl
            %esp, %ebp
    movl
            $100, 8(%ebp)
    movl
            $16, 12(%ebp)
    movl
            $65, 17(%ebp)
    movb
            20(%ebp), %eax
    movl
            8(%ebp)
    pushl
    pushl
            12(%ebp)
            16(%ebp)
    pushl
            17(%ebp)
    pushl
    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
            f,@function
    .type
f:
            %ebp
   pushl
           %esp, %ebp
   movl
   pushl
            %edi
            %esi
   pushl
            $100, 8(%ebp)
   movl
            $24, 12(%ebp)
   movl
            $65, 17(%ebp)
   movb
   subl
            $8, %esp
            $6, %eax
   movl
    subl
            $24, %esp
            %esp, %edi
   movl
            %eax, %esi
   movl
   cld
            $34, %eax
   movl
            %eax, %ecx
   movl
   rep
   movsl
   call
            f
   addl
            $32, %esp
           -8(%ebp), %esp
   leal
   popl
            %esi
            %edi
   popl
   leave
   ret
// 当 N=11 时,补全后的汇编代码片段
```

- (b) addl \$16, %esp指令将栈项指针恢复到参数压栈之前的位置, leal -8(%ebp), %esp加载栈项指针地址为%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。

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</pre>
```

```
.globl g
    .type
           g,@function
g:
   pushl
           %ebp
   movl
           %esp, %ebp
   movl
            ④ , %eax
            ⑤ , %eax
   Tvom
            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
            $72, %esp
    subl
            $-16, %esp
    andl
            $0, %eax
   movl
    subl
            %eax, %esp
    leal
            -56(%ebp), %eax
            \%eax, -64(\%ebp)
    movl
    movl
            $0, -60(\%ebp)
.L2:
            (1)
    jle
            .L5
            .L3
    jmp
.L5:
    movl
            -64(\%ebp), \%edx
            -60(%ebp), %eax
    movl
    movl
            %eax, (%edx)
            $12, %esp
    subl
    leal
            -64(\%ebp), \%eax
    pushl
            %eax
    call
            -60(%ebp), %eax
    leal
    incl
            (%eax)
            (3)
.L3:
            $0, %eax
    movl
    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?

(a) 补全后的汇编代码分别如下

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

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

针对如下 C 程序及其汇编代码 (片段):

```
#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
           $24, %esp
   subl
           $-16, %esp
   andl
   movl
           $0, %eax
           %eax, %esp
   subl
   movl
           $0,
           $0, ___
   movl
           $1, _
   movl
           $2, -12(%ebp)
   movl
   movl
           $3,
           $0, %eax
   movl
   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:
           %ebp
   pushl
           %esp, %ebp
   movl
           $24, %esp
   subl
           $-16, %esp
   andl
   movl
           $0, %eax
         %eax, %esp
   subl
   movl
         $0, %ebp
         $0, \overline{-4(\%ebp)}
   movl
         $1, -8(%ebp)
   movl
         $2, -12(\%ebp)
   movl
           $3, -16(%ebp)
   movl
           $0, %eax
   movl
   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。

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。

```
#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
```

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

解

- (a) 将从源地址\$.LCO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

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

```
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)

	1				
返回值					
返回地址					
调用者 bp					
f(Yevel, arg())					
dummy()					
返回值	返回值	返回值	返回值	返回值	返回值
level = 17	level = 14	level = 11	level = 8	level = 5	level = 2
arg = dummy	arg = local	arg = local	arg = arg	arg = arg	arg = arg
POINT TO dummy	POINT TO local				
返回地址	返回地址	返回地址	返回地址	返回地址	返回地址
调用者 bp	调用者 bp	调用者 bp	個用者 bp	调用者 bp	调用者 bp
local()	Yocal()	Yocal()	//ocal()	//ocal()	Yocal()
返回值	// 返回值	返車値	返回值	返回值 /	返回值
level = 16	level = 13	level = 10	level = 7	level = 4	level = 1
arg = local	arg = local	arg = local	arg = arg	arg = arg	arg = arg
POINT TO Jocal	POINT TO Local	POINT TO Local	POINT TO Jocal	POINT TO Local	POINT TO local
返回地址	返回地址	返回地址	返回地址	返回地址	返回地址
调用者 bp //	调用者 bp	调用者 bp	调用者bp	调用者 bp	调用者 bp ✓
local()	local()	local()	local()	local()	local()
/ 返回值 /	/ /返回值 /	/ 返車值	/ 返回值	返回值	返回值 = 11
level = /15 /	level = /12 /	level = 9 /	level = 6	level = 3	返回地址
arg = local	arg = local	arg = arg /	arg = arg	arg = arg	调用者 bp
POINT TO local	POINT TO local	POINT TO local	POINT TO local	POINT TO local	
返回地址	返回地址	返回地址	返回地址	返回地址	
调用者 bp	调用者 bp	调用者 bp	调用者 bp	调用者 bp	
local()	local()	local()	local()	local()	

图 1: 栈上包含 19 个活动记录表时的图示