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）：

(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

```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
```

```
    .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
```

Exercise 2

针对如下 C 程序及其汇编代码（片段）：

```
#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
```

(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。

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）：

(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
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