《汇编语言》作业 7

1 结构体作为参数传递

1.1 c 语言程序

```c
#include<stdio.h>

struct test {
    int a;
    float b;
    double c;
    short d;
    long e;
    float f;
    int g;
    double h;
    char i;
    long long j;
};

void process_struct(struct test cs) {
    cs.a = 100;
    cs.b = 3.14f;
    // 修改结构体成员
}

int main() {
    struct test s = {
        1, 2.0f, 3.0, 4, 5, 6.0f, 7, 8.0, '9', 10
    };
    process_struct(s);
    return 0;
}
```

1.2 汇编程序

利用命令

```
gcc -S struct.c
```

得到如下汇编程序

```
    .file "struct.c"
    .text
    .globl process_struct
    .type process_struct, @function
process_struct:
```

```
.LFB0:
    .cfi_startproc
    endbr64
    pushq %rbp
    .cfi_def_cfa_offset 16
    .cfi_offset 6, -16
    movq  %rsp, %rbp
    .cfi_def_cfa_register 6
    movl  $100, 16(%rbp)       #修改传入的参数a
    movss .LC0(%rip), %xmm0
    movss %xmm0, 20(%rbp)        #修改传入的参数b
    nop
    popq  %rbp
    .cfi_def_cfa 7, 8
    ret
    .cfi_endproc
.LFE0:
    .size process_struct, .-process_struct
    .globl main
    .type main, @function
main:
.LFB1:
    .cfi_startproc
    endbr64
    pushq %rbp
    .cfi_def_cfa_offset 16
    .cfi_offset 6, -16
    movq  %rsp, %rbp
    .cfi_def_cfa_register 6
    subq  $64, %rsp          #为结构体变量来分配空间
    # 初始化结构体赋值，逐个成员来初始化
    movl  $1, -64(%rbp)      #为成员a赋值
    movss .LC1(%rip), %xmm0
    movss %xmm0, -60(%rbp)  #为成员b赋值
    movsd .LC2(%rip), %xmm0
    movsd %xmm0, -56(%rbp)  #为成员c赋值，因为c为double类型，占8个字节
    movw  $4, -48(%rbp)      #为成员d赋值
    movq  $5, -40(%rbp)      #为成员e赋值
    movss .LC3(%rip), %xmm0
    movss %xmm0, -32(%rbp)  #为成员f赋值
    movl  $7, -28(%rbp)      #为成员e赋值
    movsd .LC4(%rip), %xmm0  #为成员h赋值
    movsd %xmm0, -24(%rbp)
    movb  $57, -16(%rbp)     #为成员i赋值，9的ASCII码是57
    movq  $10, -8(%rbp)      #为成员j赋值
    pushq -8(%rbp)         #将结构体的各个参数压栈，
    pushq -16(%rbp)          #8个字节8个字节来压栈
    pushq -24(%rbp)
    pushq -32(%rbp)
    pushq -40(%rbp)
```

```
    pushq -48(%rbp)
    pushq -56(%rbp)
    pushq -64(%rbp)
    call  process_struct
    addq  $64, %rsp         #清理栈空间
    movl  $0, %eax
    leave
    .cfi_def_cfa 7, 8
    ret
    .cfi_endproc
.LFE1:
    .size main, .-main
    .section .rodata
    .align 4
.LC0:
    .long 1078523331
    .align 4
.LC1:
    .long 1073741824
    .align 8
.LC2:
    .long 0
    .long 1074266112
    .align 4
.LC3:
    .long 1086324736
    .align 8
.LC4:
    .long 0
    .long 1075838976
    .ident "GCC: (Ubuntu 9.4.0-1ubuntu1~20.04.2) 9.4.0"
    .section .note.GNU-stack,"",@progbits
    .section .note.gnu.property,"a"
    .align 8
    .long  1f - 0f
    .long  4f - 1f
    .long  5
0:
    .string  "GNU"
1:
    .align 8
    .long  0xc0000002
    .long  3f - 2f
2:
    .long  0x3
3:
    .align 8
4:
```

具体的语句分析在程序中的注释中有体现, 在汇编语言中, 如果遇到较小的结构体作为参数, 可以直接用寄存器来传递, 而在本实验中的大结构体 (包括 10 个成员), 这时寄存器的空间不够, 于是利用栈进行参数传递, 如程序中所示, 在 main 函数中利用栈来对结构体变量分配空间, 并且利用每个成员在栈中的位置对它们进行赋值, 利用栈将参数传输到 process_struct 函数中, 在这个函数中, 利用栈顶的索引寻址来找到操作的成员, 这就是利用栈来传递结构体参数

2 结构体作为返回值

2.1 c 语言程序

```c
#include<stdio.h>

struct test {
    int a;
    float b;
    double c;
    short d;
    long e;
    float f;
    int g;
    double h;
    char i;
    long long j;
};

struct test create_struct() {
    struct test cs = {
        100, 3.14f, 2.718, 4, 5, 6.28f, 7, 1.414, 'X', 1000
    };
    return cs;
}

int main() {
    struct test s = create_struct();
    return 0;
}
```

2.2 汇编程序

利用命令

```
gcc -S struct2.c
```

得到如下汇编程序

```
    .file "struct2.c"
    .text
    .globl create_struct
```

```
    .type create_struct, @function
create_struct:
.LFB0:
    .cfi_startproc
    endbr64
    pushq %rbp
    .cfi_def_cfa_offset 16
    .cfi_offset 6, -16
    movq  %rsp, %rbp
    .cfi_def_cfa_register 6
    pushq %rbx
    .cfi_offset 3, -24
    movq  %rdi, -80(%rbp)        #获取指针
    movl  $100, -72(%rbp)        #对成员a
    movss .LC0(%rip), %xmm0
    movss %xmm0, -68(%rbp)     #对成员b
    movsd .LC1(%rip), %xmm0
    movsd %xmm0, -64(%rbp)     #对成员c
    movw  $4, -56(%rbp)        #对成员d
    movq  $5, -48(%rbp)        #对成员e
    movss .LC2(%rip), %xmm0
    movss %xmm0, -40(%rbp)     #对成员f
    movl  $7, -36(%rbp)        #对成员g
    movsd .LC3(%rip), %xmm0
    movsd %xmm0, -32(%rbp)     #对成员h
    movb  $88, -24(%rbp)       #对成员i，X的ascii码是88
    movq  $1000, -16(%rbp)     #对成员j（longlong占16个字节）
    # 这部分是关键，将以上结构体的成员通过指针写入，
    # 将这些成员的值全部利用这个指针rax来实现返回
    movq  -80(%rbp), %rax
    movq  -72(%rbp), %rcx
    movq  -64(%rbp), %rbx
    movq  %rcx, (%rax)
    movq  %rbx, 8(%rax)
    movq  -56(%rbp), %rcx
    movq  -48(%rbp), %rbx
    movq  %rcx, 16(%rax)
    movq  %rbx, 24(%rax)
    movq  -40(%rbp), %rcx
    movq  -32(%rbp), %rbx
    movq  %rcx, 32(%rax)
    movq  %rbx, 40(%rax)
    movq  -24(%rbp), %rcx
    movq  -16(%rbp), %rbx
    movq  %rcx, 48(%rax)
    movq  %rbx, 56(%rax)
    movq  -80(%rbp), %rax
    popq  %rbx
    popq  %rbp
    .cfi_def_cfa 7, 8
```

```
    ret
    .cfi_endproc
.LFE0:
    .size create_struct, .-create_struct
    .globl main
    .type main, @function
main:
.LFB1:
    .cfi_startproc
    endbr64
    pushq %rbp
    .cfi_def_cfa_offset 16
    .cfi_offset 6, -16
    movq  %rsp, %rbp
    .cfi_def_cfa_register 6
    subq  $80, %rsp
    movq  %fs:40, %rax
    movq  %rax, -8(%rbp)
    xorl  %eax, %eax
    leaq  -80(%rbp), %rax
    movq  %rax, %rdi
    movl  $0, %eax
    call  create_struct
    movl  $0, %eax
    movq  -8(%rbp), %rdx
    xorq  %fs:40, %rdx
    je  .L5
    call  __stack_chk_fail@PLT
.L5:
    leave
    .cfi_def_cfa 7, 8
    ret
    .cfi_endproc
.LFE1:
    .size main, .-main
    .section .rodata
    .align 4
.LC0:
    .long 1078523331
    .align 8
.LC1:
    .long 3367254360
    .long 1074118262
    .align 4
.LC2:
    .long 1086911939
    .align 8
.LC3:
    .long 1992864825
    .long 1073127358
```

```
    .ident "GCC: (Ubuntu 9.4.0-1ubuntu1~20.04.2) 9.4.0"
    .section .note.GNU-stack,"",@progbits
    .section .note.gnu.property,"a"
    .align 8
    .long  1f - 0f
    .long  4f - 1f
    .long  5
0:
    .string  "GNU"
1:
    .align 8
    .long  0xc0000002
    .long  3f - 2f
2:
    .long  0x3
3:
    .align 8
4:
```

具体的语句分析在程序中的注释中有体现, 在汇编语言中, 如果遇到较小的结构体作为返回值, 可能用寄存器来传递, 而在本实验中的情况, 与结构体作为参数传递的情形相类似, 这时寄存器的空间不够, 经过查阅资料得知程序采用一种隐藏指针的办法来将大结构体作为返回值返回, 具体的操作是调用的函数分配返回结构体的内存空间, 然后将空间地址作为第一个参数传递给函数, 在函数中, 通过这个指针来对结构体的成员进行赋值, 经过写入后, 其他的部分可以用这个指针来对这个结构体进行访问