Homework5

姓名：田原

学号：2023200406

题目1

Operand	Value
%ebx	0x10
$0x150	0x150
0x170	0x170
(%ebx)	0x10
(%ebx,%eax)	0x11
0x30(%ebx)	0x13
80(%ebx,%eax,2)	0x17

Instruction	Destination	Value
addl %eax,%ebx	%ebx	0x110
subl %eax,(%ebx)	0x100	0x0
leal 0x50(%eax), %edx	%edx	0x60
movzbl %al, %ebx	%edx	0x10
movsbl %bh, %ecx	%ecx	0x1

Instruction	OF	SF	ZF	CF
leal(%eax),%ebx	0	0	0	0
subl %ebx, %eax	0	1	0	1
xorl %eax, %eax	0	0	1	0
test %eax, %ebx	0	0	1	0

题目2

```
int -0xc(%ebp)=3;
int -0x8(%ebp)=2;
int -0x4(%ebp)=1;
int -0x10(%ebp);
while(-0xc(%ebp)<=5)
{
    -0x10(%ebp)=-0x4(%ebp);
    -0x4(%ebp)=-0x8(%ebp);
    -0x8(%ebp)+=-0x10(%ebp);
    -0xc(%ebp)+=1;
}
```

题目3

```
    .file   "findmin.c"
    .text
    .section    .rodata
.LC0:
    .string "minimum element is %d"
    .text
    .globl  main
    .type   main, @function
main:
.LFB0:
    .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
    movq    %fs:40, %rax
    movq    %rax, -8(%rbp)
    xorl    %eax, %eax
    movl    $1, -48(%rbp)
    movl    $10, -44(%rbp)
    movl    $9, -40(%rbp)
    movl    $8, -36(%rbp)
    movl    $7, -32(%rbp)
    movl    $6, -28(%rbp)
    movl    $5, -24(%rbp)
    movl    $4, -20(%rbp)
    movl    $3, -16(%rbp)
    movl    $2, -12(%rbp)
    movl    $-1, -56(%rbp)
    movl    $0, -52(%rbp)
.L4:
    cmpl    $9, -52(%rbp)
    jg  .L2
    movl    -52(%rbp), %eax
    cltq
```

```
    movl    -48(%rbp,%rax,4), %eax
    cmpl    %eax, -56(%rbp)
    jle .L3
    movl    -52(%rbp), %eax
    cltq
    movl    -48(%rbp,%rax,4), %eax
    movl    %eax, -56(%rbp)
.L3:
    addl    $1, -52(%rbp)
    jmp .L4
.L2:
    movl    -56(%rbp), %eax
    movl    %eax, %esi
    leaq    .LC0(%rip), %rdi
    movl    $0, %eax
    call    printf@PLT
    movl    $0, %eax
    movq    -8(%rbp), %rdx
    xorq    %fs:40, %rdx
    je  .L6
    call    __stack_chk_fail@PLT
.L6:
    leave
    .cfi_def_cfa 7, 8
    ret
    .cfi_endproc
.LFE0:
    .size   main, .-main
    .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:
```

```
a.out:     file format elf64-x86-64


Disassembly of section .init:

0000000000001000 <_init>:
```

```
    1000:   f3 0f 1e fa             endbr64
    1004:   48 83 ec 08             sub    $0x8,%rsp
    1008:   48 8b 05 d9 2f 00 00    mov    0x2fd9(%rip),%rax        # 3fe8
<__gmon_start__>
    100f:   48 85 c0                test   %rax,%rax
    1012:   74 02                   je     1016 <_init+0x16>
    1014:   ff d0                   callq  *%rax
    1016:   48 83 c4 08             add    $0x8,%rsp
    101a:   c3                      retq

Disassembly of section .plt:

0000000000001020 <.plt>:
    1020:   ff 35 92 2f 00 00       pushq  0x2f92(%rip)        # 3fb8
<_GLOBAL_OFFSET_TABLE_+0x8>
    1026:   f2 ff 25 93 2f 00 00    bnd jmpq *0x2f93(%rip)        # 3fc0
<_GLOBAL_OFFSET_TABLE_+0x10>
    102d:   0f 1f 00                nopl   (%rax)
    1030:   f3 0f 1e fa             endbr64
    1034:   68 00 00 00 00          pushq  $0x0
    1039:   f2 e9 e1 ff ff ff       bnd jmpq 1020 <.plt>
    103f:   90                      nop
    1040:   f3 0f 1e fa             endbr64
    1044:   68 01 00 00 00          pushq  $0x1
    1049:   f2 e9 d1 ff ff ff       bnd jmpq 1020 <.plt>
    104f:   90                      nop

Disassembly of section .plt.got:

0000000000001050 <__cxa_finalize@plt>:
    1050:   f3 0f 1e fa             endbr64
    1054:   f2 ff 25 9d 2f 00 00    bnd jmpq *0x2f9d(%rip)        # 3ff8
<__cxa_finalize@GLIBC_2.2.5>
    105b:   0f 1f 44 00 00          nopl   0x0(%rax,%rax,1)

Disassembly of section .plt.sec:

0000000000001060 <__stack_chk_fail@plt>:
    1060:   f3 0f 1e fa             endbr64
    1064:   f2 ff 25 5d 2f 00 00    bnd jmpq *0x2f5d(%rip)        # 3fc8
<__stack_chk_fail@GLIBC_2.4>
    106b:   0f 1f 44 00 00          nopl   0x0(%rax,%rax,1)

0000000000001070 <printf@plt>:
    1070:   f3 0f 1e fa             endbr64
    1074:   f2 ff 25 55 2f 00 00    bnd jmpq *0x2f55(%rip)        # 3fd0
<printf@GLIBC_2.2.5>
    107b:   0f 1f 44 00 00          nopl   0x0(%rax,%rax,1)

Disassembly of section .text:

0000000000001080 <_start>:
    1080:   f3 0f 1e fa             endbr64
    1084:   31 ed                   xor    %ebp,%ebp
    1086:   49 89 d1                mov    %rdx,%r9
    1089:   5e                      pop    %rsi
```

```
    108a:   48 89 e2                mov    %rsp,%rdx
    108d:   48 83 e4 f0             and    $0xfffffffffffffff0,%rsp
    1091:   50                      push   %rax
    1092:   54                      push   %rsp
    1093:   4c 8d 05 06 02 00 00    lea    0x206(%rip),%r8        # 12a0
<__libc_csu_fini>
    109a:   48 8d 0d 8f 01 00 00    lea    0x18f(%rip),%rcx        # 1230
<__libc_csu_init>
    10a1:   48 8d 3d c1 00 00 00    lea    0xc1(%rip),%rdi        # 1169 <main>
    10a8:   ff 15 32 2f 00 00       callq  *0x2f32(%rip)        # 3fe0
<__libc_start_main@GLIBC_2.2.5>
    10ae:   f4                      hlt
    10af:   90                      nop

00000000000010b0 <deregister_tm_clones>:
    10b0:   48 8d 3d 59 2f 00 00    lea    0x2f59(%rip),%rdi        # 4010
<__TMC_END__>
    10b7:   48 8d 05 52 2f 00 00    lea    0x2f52(%rip),%rax        # 4010
<__TMC_END__>
    10be:   48 39 f8                cmp    %rdi,%rax
    10c1:   74 15                   je     10d8 <deregister_tm_clones+0x28>
    10c3:   48 8b 05 0e 2f 00 00    mov    0x2f0e(%rip),%rax        # 3fd8
<_ITM_deregisterTMCloneTable>
    10ca:   48 85 c0                test   %rax,%rax
    10cd:   74 09                   je     10d8 <deregister_tm_clones+0x28>
    10cf:   ff e0                   jmpq   *%rax
    10d1:   0f 1f 80 00 00 00 00    nopl   0x0(%rax)
    10d8:   c3                      retq
    10d9:   0f 1f 80 00 00 00 00    nopl   0x0(%rax)

00000000000010e0 <register_tm_clones>:
    10e0:   48 8d 3d 29 2f 00 00    lea    0x2f29(%rip),%rdi        # 4010
<__TMC_END__>
    10e7:   48 8d 35 22 2f 00 00    lea    0x2f22(%rip),%rsi        # 4010
<__TMC_END__>
    10ee:   48 29 fe                sub    %rdi,%rsi
    10f1:   48 89 f0                mov    %rsi,%rax
    10f4:   48 c1 ee 3f             shr    $0x3f,%rsi
    10f8:   48 c1 f8 03             sar    $0x3,%rax
    10fc:   48 01 c6                add    %rax,%rsi
    10ff:   48 d1 fe                sar    %rsi
    1102:   74 14                   je     1118 <register_tm_clones+0x38>
    1104:   48 8b 05 e5 2e 00 00    mov    0x2ee5(%rip),%rax        # 3ff0
<_ITM_registerTMCloneTable>
    110b:   48 85 c0                test   %rax,%rax
    110e:   74 08                   je     1118 <register_tm_clones+0x38>
    1110:   ff e0                   jmpq   *%rax
    1112:   66 0f 1f 44 00 00       nopw   0x0(%rax,%rax,1)
    1118:   c3                      retq
    1119:   0f 1f 80 00 00 00 00    nopl   0x0(%rax)

0000000000001120 <__do_global_dtors_aux>:
    1120:   f3 0f 1e fa             endbr64
    1124:   80 3d e5 2e 00 00 00    cmpb   $0x0,0x2ee5(%rip)        # 4010
<__TMC_END__>
    112b:   75 2b                   jne    1158 <__do_global_dtors_aux+0x38>
```

```
    112d:   55                      push   %rbp
    112e:   48 83 3d c2 2e 00 00    cmpq   $0x0,0x2ec2(%rip)        # 3ff8
<__cxa_finalize@GLIBC_2.2.5>
    1135:   00
    1136:   48 89 e5                mov    %rsp,%rbp
    1139:   74 0c                   je     1147 <__do_global_dtors_aux+0x27>
    113b:   48 8b 3d c6 2e 00 00    mov    0x2ec6(%rip),%rdi        # 4008
<__dso_handle>
    1142:   e8 09 ff ff ff          callq  1050 <__cxa_finalize@plt>
    1147:   e8 64 ff ff ff          callq  10b0 <deregister_tm_clones>
    114c:   c6 05 bd 2e 00 00 01    movb   $0x1,0x2ebd(%rip)        # 4010
<__TMC_END__>
    1153:   5d                      pop    %rbp
    1154:   c3                      retq
    1155:   0f 1f 00                nopl   (%rax)
    1158:   c3                      retq
    1159:   0f 1f 80 00 00 00 00    nopl   0x0(%rax)

0000000000001160 <frame_dummy>:
    1160:   f3 0f 1e fa             endbr64
    1164:   e9 77 ff ff ff          jmpq   10e0 <register_tm_clones>

0000000000001169 <main>:
    1169:   f3 0f 1e fa             endbr64
    116d:   55                      push   %rbp
    116e:   48 89 e5                mov    %rsp,%rbp
    1171:   48 83 ec 40             sub    $0x40,%rsp
    1175:   64 48 8b 04 25 28 00    mov    %fs:0x28,%rax
    117c:   00 00
    117e:   48 89 45 f8             mov    %rax,-0x8(%rbp)
    1182:   31 c0                   xor    %eax,%eax
    1184:   c7 45 d0 01 00 00 00    movl   $0x1,-0x30(%rbp)
    118b:   c7 45 d4 0a 00 00 00    movl   $0xa,-0x2c(%rbp)
    1192:   c7 45 d8 09 00 00 00    movl   $0x9,-0x28(%rbp)
    1199:   c7 45 dc 08 00 00 00    movl   $0x8,-0x24(%rbp)
    11a0:   c7 45 e0 07 00 00 00    movl   $0x7,-0x20(%rbp)
    11a7:   c7 45 e4 06 00 00 00    movl   $0x6,-0x1c(%rbp)
    11ae:   c7 45 e8 05 00 00 00    movl   $0x5,-0x18(%rbp)
    11b5:   c7 45 ec 04 00 00 00    movl   $0x4,-0x14(%rbp)
    11bc:   c7 45 f0 03 00 00 00    movl   $0x3,-0x10(%rbp)
    11c3:   c7 45 f4 02 00 00 00    movl   $0x2,-0xc(%rbp)
    11ca:   c7 45 c8 ff ff ff ff    movl   $0xffffffff,-0x38(%rbp)
    11d1:   c7 45 cc 00 00 00 00    movl   $0x0,-0x34(%rbp)
    11d8:   83 7d cc 09             cmpl   $0x9,-0x34(%rbp)
    11dc:   7f 20                   jg     11fe <main+0x95>
    11de:   8b 45 cc                mov    -0x34(%rbp),%eax
    11e1:   48 98                   cltq
    11e3:   8b 44 85 d0             mov    -0x30(%rbp,%rax,4),%eax
    11e7:   39 45 c8                cmp    %eax,-0x38(%rbp)
    11ea:   7e 0c                   jle    11f8 <main+0x8f>
    11ec:   8b 45 cc                mov    -0x34(%rbp),%eax
    11ef:   48 98                   cltq
    11f1:   8b 44 85 d0             mov    -0x30(%rbp,%rax,4),%eax
    11f5:   89 45 c8                mov    %eax,-0x38(%rbp)
    11f8:   83 45 cc 01             addl   $0x1,-0x34(%rbp)
    11fc:   eb da                   jmp    11d8 <main+0x6f>
```

```
    11fe:   8b 45 c8                mov    -0x38(%rbp),%eax
    1201:   89 c6                   mov    %eax,%esi
    1203:   48 8d 3d fa 0d 00 00    lea    0xdfa(%rip),%rdi        # 2004
<_IO_stdin_used+0x4>
    120a:   b8 00 00 00 00          mov    $0x0,%eax
    120f:   e8 5c fe ff ff          callq  1070 <printf@plt>
    1214:   b8 00 00 00 00          mov    $0x0,%eax
    1219:   48 8b 55 f8             mov    -0x8(%rbp),%rdx
    121d:   64 48 33 14 25 28 00    xor    %fs:0x28,%rdx
    1224:   00 00
    1226:   74 05                   je     122d <main+0xc4>
    1228:   e8 33 fe ff ff          callq  1060 <__stack_chk_fail@plt>
    122d:   c9                      leaveq
    122e:   c3                      retq
    122f:   90                      nop

0000000000001230 <__libc_csu_init>:
    1230:   f3 0f 1e fa             endbr64
    1234:   41 57                   push   %r15
    1236:   4c 8d 3d 73 2b 00 00    lea    0x2b73(%rip),%r15        # 3db0
<__frame_dummy_init_array_entry>
    123d:   41 56                   push   %r14
    123f:   49 89 d6                mov    %rdx,%r14
    1242:   41 55                   push   %r13
    1244:   49 89 f5                mov    %rsi,%r13
    1247:   41 54                   push   %r12
    1249:   41 89 fc                mov    %edi,%r12d
    124c:   55                      push   %rbp
    124d:   48 8d 2d 64 2b 00 00    lea    0x2b64(%rip),%rbp        # 3db8
<__do_global_dtors_aux_fini_array_entry>
    1254:   53                      push   %rbx
    1255:   4c 29 fd                sub    %r15,%rbp
    1258:   48 83 ec 08             sub    $0x8,%rsp
    125c:   e8 9f fd ff ff          callq  1000 <_init>
    1261:   48 c1 fd 03             sar    $0x3,%rbp
    1265:   74 1f                   je     1286 <__libc_csu_init+0x56>
    1267:   31 db                   xor    %ebx,%ebx
    1269:   0f 1f 80 00 00 00 00    nopl   0x0(%rax)
    1270:   4c 89 f2                mov    %r14,%rdx
    1273:   4c 89 ee                mov    %r13,%rsi
    1276:   44 89 e7                mov    %r12d,%edi
    1279:   41 ff 14 df             callq  *(%r15,%rbx,8)
    127d:   48 83 c3 01             add    $0x1,%rbx
    1281:   48 39 dd                cmp    %rbx,%rbp
    1284:   75 ea                   jne    1270 <__libc_csu_init+0x40>
    1286:   48 83 c4 08             add    $0x8,%rsp
    128a:   5b                      pop    %rbx
    128b:   5d                      pop    %rbp
    128c:   41 5c                   pop    %r12
    128e:   41 5d                   pop    %r13
    1290:   41 5e                   pop    %r14
    1292:   41 5f                   pop    %r15
    1294:   c3                      retq
    1295:   66 66 2e 0f 1f 84 00    data16 nopw %cs:0x0(%rax,%rax,1)
    129c:   00 00 00 00
```

```
00000000000012a0 <__libc_csu_fini>:
    12a0:   f3 0f 1e fa             endbr64
    12a4:   c3                      retq

Disassembly of section .fini:

00000000000012a8 <_fini>:
    12a8:   f3 0f 1e fa             endbr64
    12ac:   48 83 ec 08             sub    $0x8,%rsp
    12b0:   48 83 c4 08             add    $0x8,%rsp
    12b4:   c3                      retq
```

1.反汇编代码使用16进制表示
2.反汇编代码没有L1等跳转入口

题目4

%eax	**0x10000000**
%ecx	22
$0x10000004	0x10000004
0x10000012	None
0xFFFFFF8	None
(%eax, %ecx, 8)	44

题目5

```
int dw_loop(int x, int y, int n) {
     do{
    x+=n;
    y*=n;
    n-=1;
      }while (n==0 && y<n);
      return x;
}
```

题目6

```
mov %rdi, %rax
imul %rsi, %rax
mov %rdi, %rdx
add %rsi, %rdx
imul %rsi, %rdx
cmp %rsi,%rdi
cmovge %rdx, %rax
```

乘法的时间复杂度较高，如果使用条件传送则两个乘法都需要计算，时间成本增加

题目7

```
    movl    $24, -4(%rbp)
    movl    $0, -8(%rbp)
    cmpl    $30, -4(%rbp)
    jg  .L2
    cmpl    $29, -4(%rbp)
    jge .L3
    cmpl    $28, -4(%rbp)
    jg  .L2
    cmpl    $27, -4(%rbp)
    jge .L4
    cmpl    $24, -4(%rbp)
    je  .L5
    cmpl    $26, -4(%rbp)
    je  .L6
    jmp .L2
.L5:
    movl    -4(%rbp), %eax
    addl    %eax, %eax
    movl    %eax, -8(%rbp)
    jmp .L7
.L4:
    movl    -4(%rbp), %eax
    addl    $10, %eax
    movl    %eax, -8(%rbp)
    jmp .L7
.L6:
    movl    -4(%rbp), %eax
    addl    %eax, %eax
    movl    %eax, -8(%rbp)
.L3:
    addl    $5, -8(%rbp)
    jmp .L7
.L2:
    movl    $3, -8(%rbp)
    nop
.L7:
    movl    $0, %eax
    popq    %rbp
```

题目8

fuc(short c, char d, int *p, int x)

函数参数从由向左入栈，且栈是向下扩展的，所以x的地址最大，其余参数根据地址依次传递

```
Movsbl  12(%ebp), %edx  #说明12(%ebp)处为1字节的参数
Movl    16(%ebp), %eax  #对应函数的取出指针p的操作，所以16(%ebp)为p
Movl    %edx, (%eax)    #(%ebx)对应*p，
Movswl  8(%ebp), %eax   #说明8(%ebp)处为2字节参数
Movl    20(%ebp), %edx  #%edx为被减数，所以20(%ebp)为x
Subl    %eax, %edx      #8(%ebp)为c
Movl    %edx, %eax
```

题目9

按照pushl %ebp写（栈帧操作）

(1) Instruction 1: %ebp = 0x7FFFFFF4 %esp = 0x7FFFFFC0

(2) Instruction 2: %ebp = 0x7FFFFFC0 %esp = 0x7FFFFFC0

(3) Instruction 3: %ebp = 0x7FFFFFF4 %esp = 0x7FFFFFC4

题目10

```c
int main()
{
    int a,b;
    scanf("%d %d",&a,&b);
    int c = a^b;
    printf("%d %d %d\n",c,a,b);
    return 0;
}
```

24行调用函数printf

%edi: .LC1的地址

%esi: a ^ b

%edx: a的值

%ecx: b的值

%eax: 0

%rsp: 0x8000408

栈状态：

0x8000420 <- 初始%rsp

0x800041C

0x8000418

0x8000414 | a的值 | <- 0x8000408 + 12

0x8000410 | b的值 | <- 0x8000408 + 8

0x800040C

0x8000408 <- 当前%rsp (0x8000408)