Many language rules are checked by the compiler, and it is possible to bypass the rules using assembly language after compilation. Consider the following C program:

```
//c.c
1
2
    #include<stdio.h>
3
   int x=3;
4
6
   int main(void)
7
    int x=5;
8
     printf("x = %d\n", x);
9
10
    }
11
```

1. Compile the program and generate its assembly code.

在终端输入命令

```
1 gcc -S c.c
```

获得汇编代码c.s

```
.file "c.c"
 2
     .text
 3
     .globl _x
     .data
     .align 4
 5
   _x:
 6
     .long 3
 7
     .def ___main; .scl 2; .type 32; .endef
8
     .section .rdata, "dr"
9
10
   LC0:
     .ascii "x = %d\12\0"
11
     .text
12
     .globl _main
13
14
     .def _main; .scl 2; .type 32; .endef
15
   main:
16
   LFB11:
17
     .cfi_startproc
    pushl %ebp
18
19
     .cfi_def_cfa_offset 8
20
     .cfi_offset 5, -8
    movl %esp, %ebp
21
22
     .cfi def cfa register 5
    andl $-16, %esp
```

```
24
     subl $32, %esp
25
     call ___main
     movl $5, 28(%esp)
26
27
    movl 28(%esp), %eax
28
    movl %eax, 4(%esp)
29
    movl $LC0, (%esp)
30
    call _printf
31
    movl $0, %eax
32
     leave
     .cfi restore 5
33
34
     .cfi_def_cfa 4, 4
35
    ret
     .cfi endproc
36
37 LFE11:
38
     .ident "GCC: (MinGW.org GCC-8.2.0-3) 8.2.0"
     .def _printf; .scl 2; .type 32; .endef
39
40
```

2. Understand the assembly code and modify it to let the program print the global variable \mathbf{x} instead of the local variable \mathbf{x} .

·globl _x 中_x代表一个地址,.globl表明_x会被链接器使用,需要在目标文件的符号表中标记为一个全局符号。

.cfi startproc 和 .cfi endproc 分别位于汇编函数的开头和结尾,标志函数的起止。

pushl %ebp 和 movl %esp, %ebp 两条指令是将栈顶指向ebp的地址。对寄存器esp做一系列运算之后,定位到x的栈地址,通过 movl \$5, 28(%esp) 将5写入x。栈中存放的是局部变量,代表C语言中的int x=5;语句。 movl 28(%esp), %eax 将x的值放入基地址寄存器 eax。 movl %eax, 4(%esp) 将eax中的值放入4(%esp)。LCO字段存放需要打印的信息。通过 movl \$LCO, (%esp) 和 call printf

调用打印出x=5。

因为_x代表全局变量,通过 mov1 _x, %eax 将全局变量放到eax中,打印出来的就是全局变量x。修改汇编指令如下:

```
1
    .file "c.c"
    .text
2
3
    .globl _x
4
    .data
5
    .align 4
   _x:
6
    .long 3
7
    .def main; .scl 2; .type 32; .endef
8
9
    .section .rdata,"dr"
10 LC0:
11
    .ascii "x = %d\12\0"
12
     .text
```

```
.globl _main
13
      .def _main; .scl 2; .type 32; .endef
14
15
    main:
16 LFB11:
17
     .cfi_startproc
    pushl %ebp
18
19
     .cfi def cfa offset 8
     .cfi offset 5, -8
20
21
    movl %esp, %ebp
    .cfi def cfa register 5
22
    andl $-16, %esp
23
    subl $16, %esp
24
    call ___main
25
    movl $5, 28(%esp)
26
27
    movl _x, %eax
28
    movl %eax, 4(%esp)
    movl $LC0, (%esp)
29
    call _printf
30
31
    movl $0, %eax
    leave
33
    .cfi_restore 5
34
    .cfi_def_cfa 4, 4
35
     ret
36
     .cfi_endproc
37 LFE11:
     .ident "GCC: (MinGW.org GCC-8.2.0-3) 8.2.0"
38
      .def _printf; .scl 2; .type 32; .endef
39
```

检验:

```
C:\Users\zjy\cd C:\Users\zjy\Desktop
C:\Users\zjy\Desktop>gcc -c c.s -o c.o
C:\Users\zjy\Desktop>gcc c.o -o c
C:\Users\zjy\Desktop>c.exe
x = 3
```