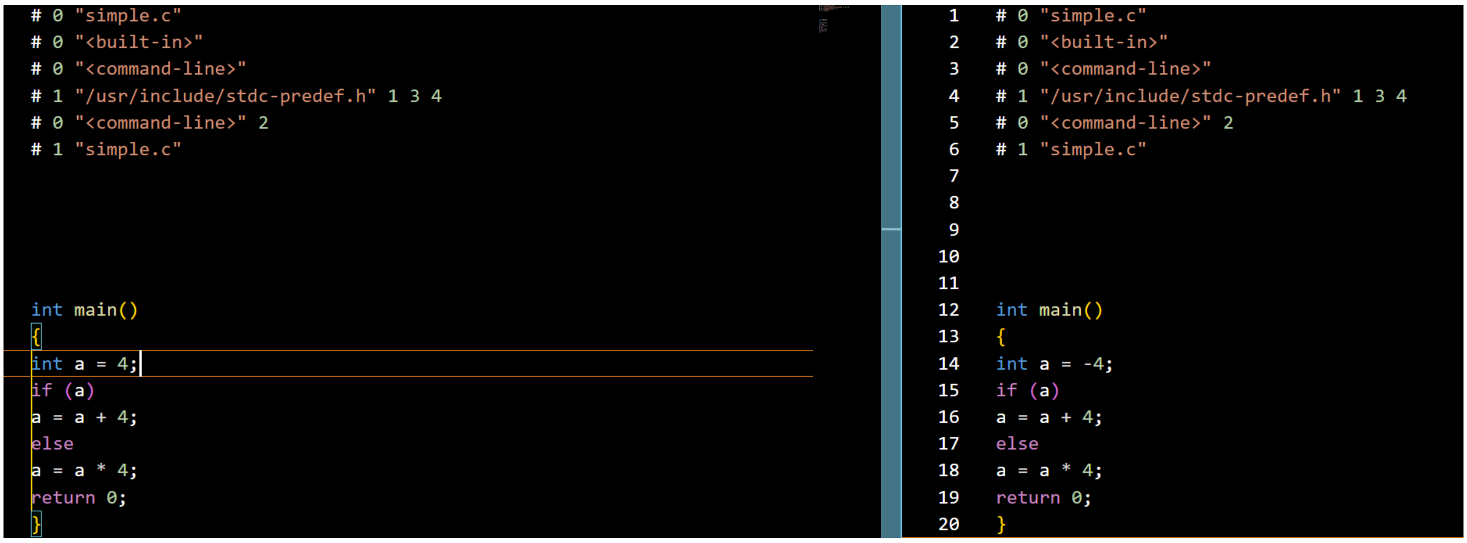
answer

问题1-1 如果在命令行下执行 gcc -DNEG -E sample.c -o sample.i 生成的 sample.i 与之前的有何区别?

区别在于对a的初识赋值从4 变为了 -4

原因是 gcc 编译使用了 -D 选项,表示编译前预定义了 宏NEG ,而源文件中的宏M 定义会受 宏NEG 定义与否影响



```
# 0 "simple.c"
# 0 "<built-in>"
# 0 "<command-line>"
# 1 "/usr/include/stdc-predef.h" 1 3 4
# 0 "<command-line>" 2
# 1 "simple.c"





int main()
{
int a = 4;
if (a)
a = a + 4;
else
a = a * 4;
return 0;
}
```

```
# 0 "simple.c"
# 0 "<built-in>"
# 0 "<command-line>"
# 1 "/usr/include/stdc-predef.h" 1 3 4
# 0 "<command-line>" 2
# 1 "simple.c"





int main()
{
int a = -4;
if (a)
a = a + 4;
else
a = a * 4;
return 0;
}
```

问题1-2 请对比 sample-32.s 和 sample.s， 找出它们的区别，并上网检索给出产生这些区别的原因.

区别1 ,指令助记符不同

原因: 编译采用的环境位数不同,在 32位环境下助记符后缀为l 例如 pushl,movl,subl,cmpl, 而64位环境下助记符后缀为q

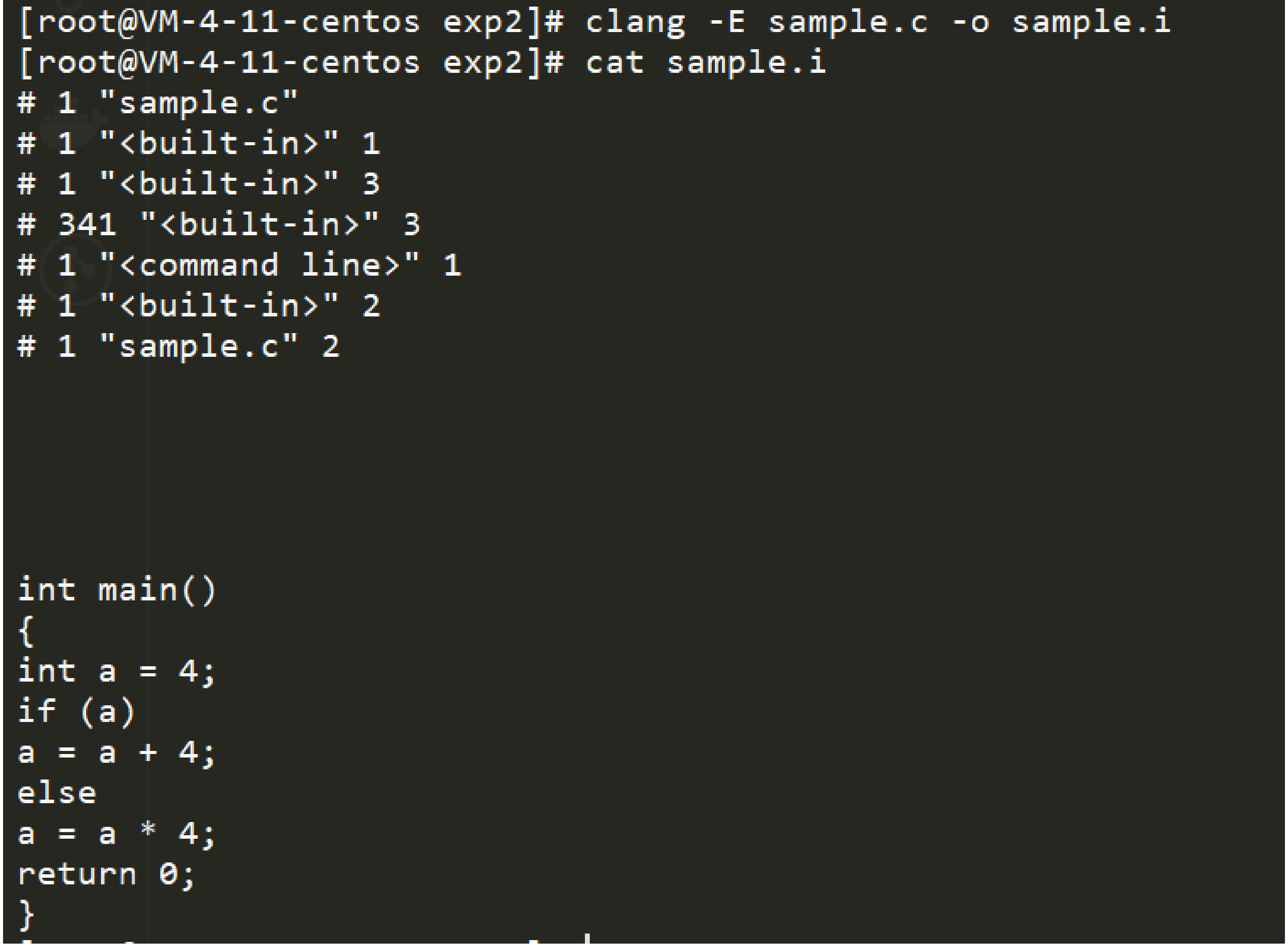
区别2 寄存器描述符不同

在32 位环境下寄存器描述为 eax,ebx,ecx,edx 64位环境下则是rax,rbx,rcx,rdx

问题1-3 你可以用 clang 替换 gcc， 重复上面的各步，比较使用 clang 和 gcc 分别输出的结果有何异同。

预处理头部注释有区别

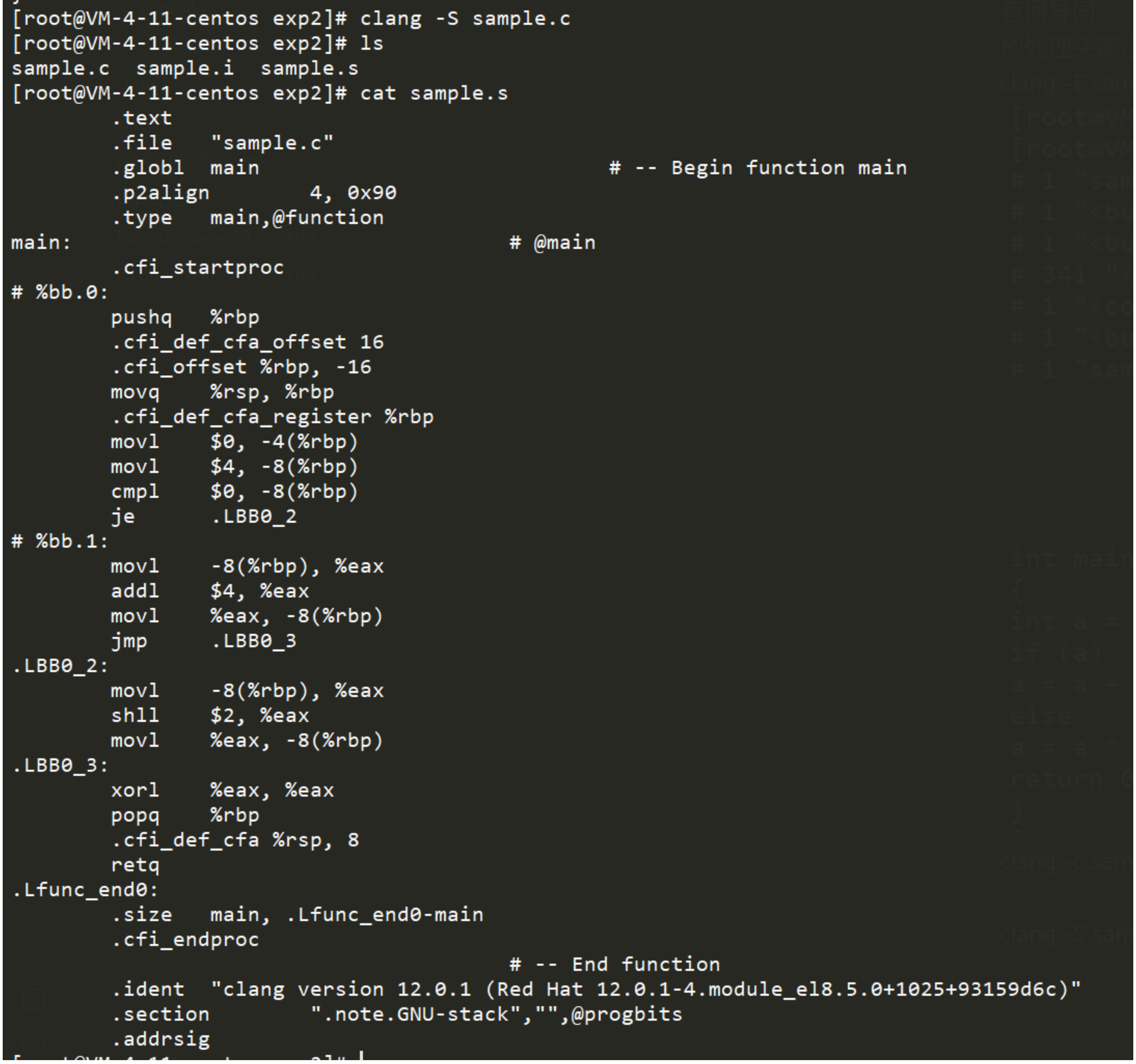
clang -E sample.c



```
[root@VM-4-11-centos exp2]# clang -E sample.c -o sample.i
[root@VM-4-11-centos exp2]# cat sample.i
# 1 "sample.c"
# 1 "<built-in>" 1
# 1 "<built-in>" 3
# 341 "<built-in>" 3
# 1 "<command line>" 1
# 1 "<built-in>" 2
# 1 "sample.c" 2






int main()
{
int a = 4;
if (a)
a = a + 4;
else
a = a * 4;
return 0;
}
```

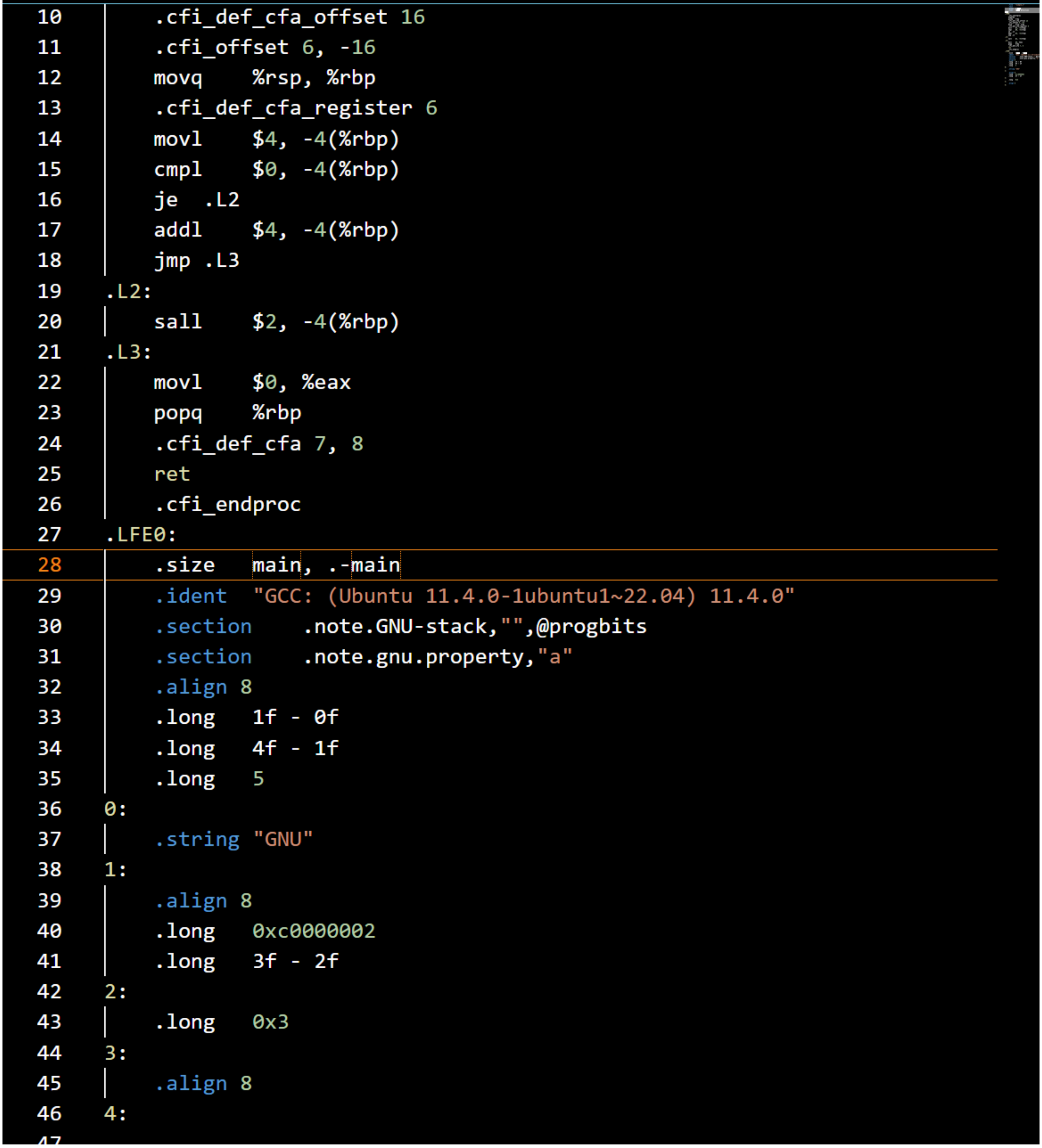
编译文件:区别是.ident 后面的身份信息不同

clang -S sample.c



```
[root@VM-4-11-centos exp2]# clang -S sample.c
[root@VM-4-11-centos exp2]# ls
sample.c  sample.i  sample.s
[root@VM-4-11-centos exp2]# cat sample.s
        .text
        .file   "sample.c"
        .globl  main                            # -- Begin function main
        .p2align        4, 0x90
        .type   main,@function
main:                                   # @main
        .cfi_startproc
# %bb.0:
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset %rbp, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register %rbp
        movl    $0, -4(%rbp)
        movl    $4, -8(%rbp)
        cmpl    $0, -8(%rbp)
        je      .LBB0_2
# %bb.1:
        movl    -8(%rbp), %eax
        addl    $4, %eax
        movl    %eax, -8(%rbp)
        jmp     .LBB0_3
.LBB0_2:
        movl    -8(%rbp), %eax
        shll    $2, %eax
        movl    %eax, -8(%rbp)
.LBB0_3:
        xorl    %eax, %eax
        popq    %rbp
        .cfi_def_cfa %rsp, 8
        retq
.Lfunc_end0:
        .size   main, .Lfunc_end0-main
        .cfi_endproc
                                        # -- End function
        .ident  "clang version 12.0.1 (Red Hat 12.0.1-4.module_el8.5.0+1025+93159d6c)"
        .section        ".note.GNU-stack","",@progbits
        .addrsig
```

gcc -S sample.c

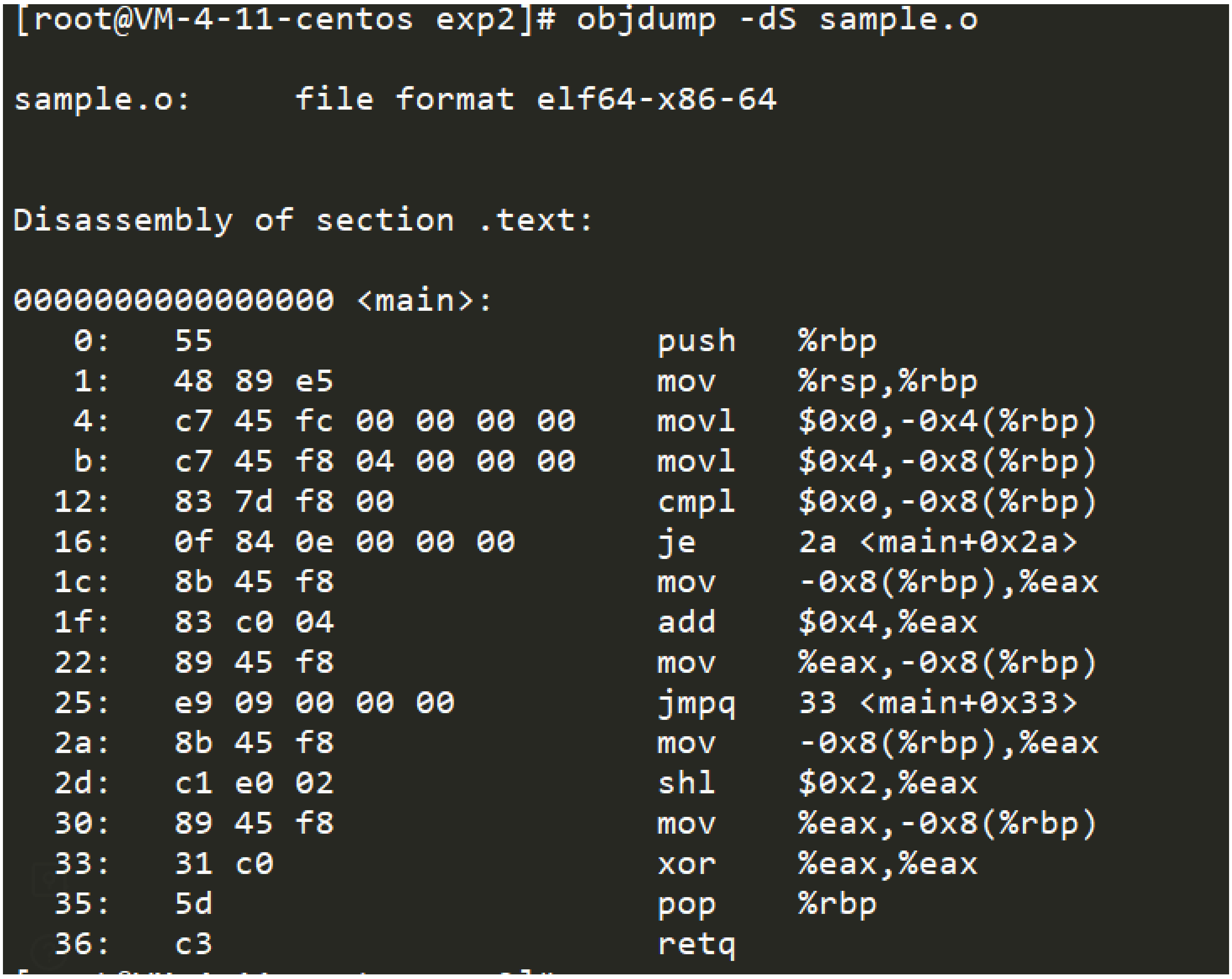


```
	.cfi_def_cfa_offset 16
	.cfi_offset 6, -16
	movq	%rsp, %rbp
	.cfi_def_cfa_register 6
	movl	$4, -4(%rbp)
	cmpl	$0, -4(%rbp)
	je	.L2
	addl	$4, -4(%rbp)
	jmp .L3
.L2:
	sall	$2, -4(%rbp)
.L3:
	movl	$0, %eax
	popq	%rbp
	.cfi_def_cfa 7, 8
	ret
	.cfi_endproc
.LFE0:
	.size	main, .-main
	.ident	"GCC: (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0"
	.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:
```

clang -c sample.c

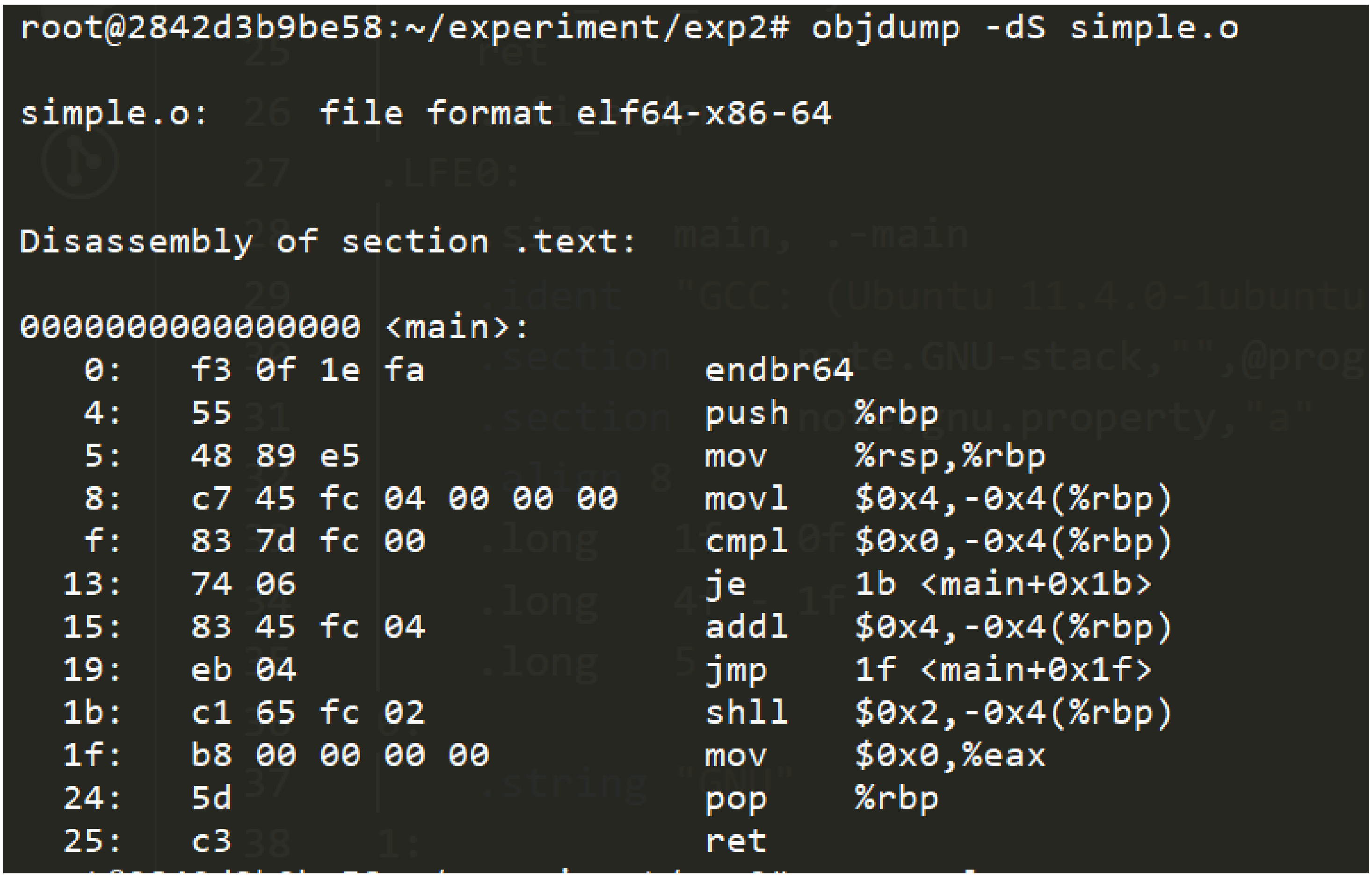
依然是汇编之后的机器码形式

objdump -dS sample.o (clang 汇编)



```
[root@VM-4-11-centos exp2]# objdump -dS sample.o

sample.o:     file format elf64-x86-64


Disassembly of section .text:

0000000000000000 <main>:
   0:	55                   	push   %rbp
   1:	48 89 e5             	mov    %rsp,%rbp
   4:	c7 45 fc 00 00 00 00 	movl   $0x0,-0x4(%rbp)
   b:	c7 45 f8 04 00 00 00 	movl   $0x4,-0x8(%rbp)
  12:	83 7d f8 00          	cmpl   $0x0,-0x8(%rbp)
  16:	0f 84 0e 00 00 00    	je     2a <main+0x2a>
  1c:	8b 45 f8             	mov    -0x8(%rbp),%eax
  1f:	83 c0 04             	add    $0x4,%eax
  22:	89 45 f8             	mov    %eax,-0x8(%rbp)
  25:	e9 09 00 00 00       	jmpq   33 <main+0x33>
  2a:	8b 45 f8             	mov    -0x8(%rbp),%eax
  2d:	c1 e0 02             	shl    $0x2,%eax
  30:	89 45 f8             	mov    %eax,-0x8(%rbp)
  33:	31 c0                	xor    %eax,%eax
  35:	5d                   	pop    %rbp
  36:	c3                   	retq
```

objdump -dS sample.o (gcc 汇编)



```
root@2842d3b9be58:~/experiment/exp2# objdump -dS simple.o

simple.o:     file format elf64-x86-64


Disassembly of section .text:

0000000000000000 <main>:
   0:   f3 0f 1e fa             endbr64
   4:   55                      push   %rbp
   5:   48 89 e5                mov    %rsp,%rbp
   8:   c7 45 fc 04 00 00 00    movl   $0x4,-0x4(%rbp)
   f:   83 7d fc 00             cmpl   $0x0,-0x4(%rbp)
  13:   74 06                   je     1b <main+0x1b>
  15:   83 45 fc 04             addl   $0x4,-0x4(%rbp)
  19:   eb 04                   jmp    1f <main+0x1f>
  1b:   c1 65 fc 02             shll   $0x2,-0x4(%rbp)
  1f:   b8 00 00 00 00          mov    $0x0,%eax
  24:   5d                      pop    %rbp
  25:   c3                      ret
```