Compile-Principle HW11

Question 1

a 识别该流图的循环

循环1：B_2-B_5-B_2

循环2：B_2-B_3-B_5-B_2

循环3：B_3-B_4-B_3

b 复写语句

(5),(6),(7)中对a的引用可以被用常量替代

(3),(4),(8),(9)中对a的引用可以被常量替代

c 每个循环的全局公共子表达式

loop1 a+b c-a

loop2 a+b c-a

loop3

d 归纳变量

loop2：b

loop3：d，e

e 循环不变计算

没有循环不变计算

Question 3

b

Block	e_gen	e_kill	IN	OUT
ENTRY			VOID	
B1	(1)(2)	(8)(10)(11)	VOID	(1)(2)
B2	(3)(4)	(5)(6)	(1)(2)(3)(4)(5)(8)(9)	(1)(2)(3)(4)(8)(9)
B3	(5)	(4)(6)	(1)(2)(3)(4)(5)(8)(9)	(1)(2)(3)(5)(8)(9)
B4	(6)(7)	(5)(4)(9)	(1)(2)(3)(5)(8)(9)	(1)(2)(3)(6)(7)(8)
B5	(8)(9)	(2)(11)(7)	(1)(2)(3)(4)(5)(7)(8)	(1)(3)(4)(5)(8)(9)
B6	(10)(11)	(1)(2)(8)	(1)(3)(4)(5)(8)(9)	(3)(4)(5)(9)(10)(11)
EXIT				VOID

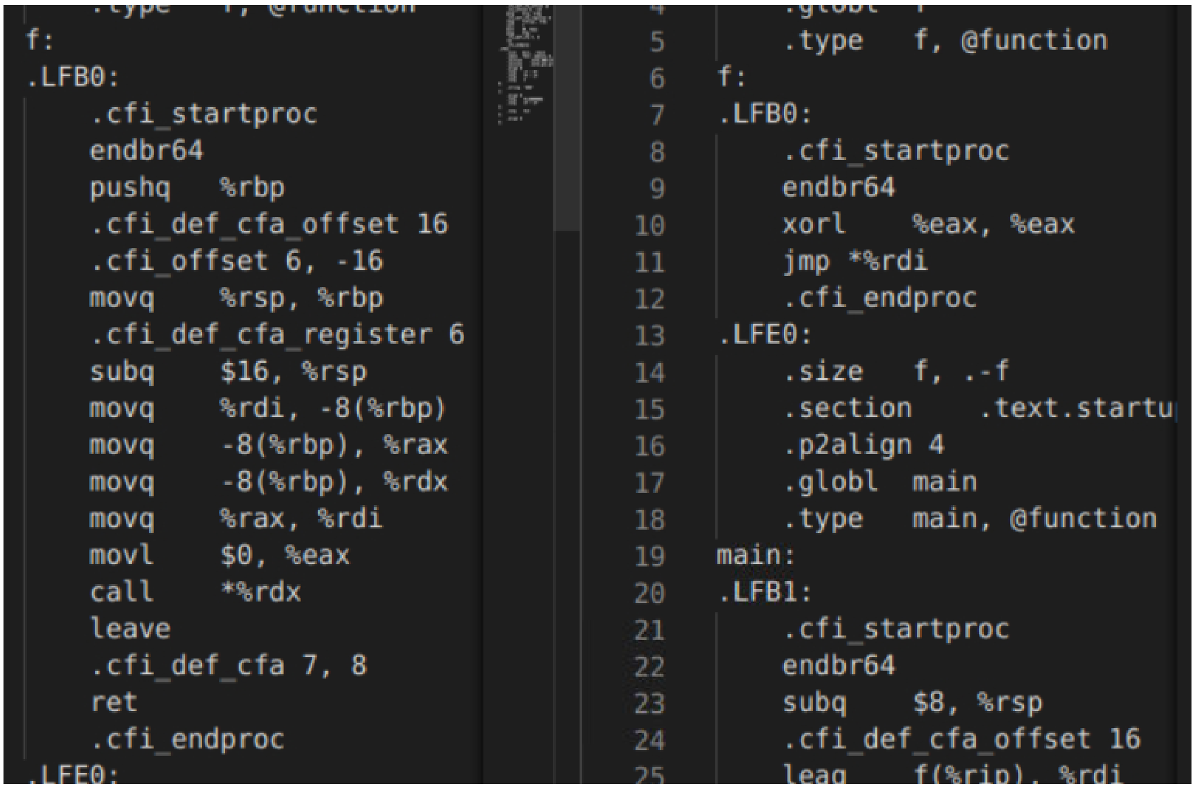
C

BLOCK	DEF	USE	IN	OUT
ENTRY			void	
B1	a,b		void	a,b
B2	c,d	b,c	a,b,d,e	a,c,d,e
B3	d	b,d	a,c,d,e	a,c,d,e
B4	d,e	b,e	a,c,d,e	a,c,d,e
B5	b,e	b,c	a,c,d,e	a,b,d,e
B6	a,b	b,d.a	a,b,d,e	a,b,e
EXIT				

Question 3

不开优化出现了segmentation fault，开优化死循环

汇编层次主要的不同



```
	.type	f, @function
f:
.LFB0:
	.cfi_startproc
	endbr64
	pushq	%rbp
	.cfi_def_cfa_offset 16
	.cfi_offset 6, -16
	movq	%rsp, %rbp
	.cfi_def_cfa_register 6
	subq	$16, %rsp
	movq	%rdi, -8(%rbp)
	movq	-8(%rbp), %rax
	movq	-8(%rbp), %rdx
	movq	%rax, %rdi
	movl	$0, %eax
	call	*%rdx
	leave
	.cfi_def_cfa 7, 8
	ret
	.cfi_endproc
.LFE0:
```

```
	.globl	f
	.type	f, @function
f:
.LFB0:
	.cfi_startproc
	endbr64
	xorl	%eax, %eax
	jmp *%rdi
	.cfi_endproc
.LFE0:
	.size	f, .-f
	.section	.text.startu
	.p2align 4
	.globl	main
	.type	main, @function
main:
.LFB1:
	.cfi_startproc
	endbr64
	subq	$8, %rsp
	.cfi_def_cfa_offset 16
	leaq	f(%rip), %rdi
```

左边是没优化的，右边加了优化，关注LFB0这一段

可见，没有优化的代码对于并没有对上下文进行检查，而是采用了函数调用的方式。

优化过后的代码采取的是直接跳转执行，跳转的目标地址存在寄存器中，跳转和函数调用的区别在于，跳转不会在栈空间中分配额外的空间，而函数调用需要在栈空间中分配额外的空间，这样就会导致栈溢出这种错误