“ 编译原理 ”考试试卷（A 卷）

考试方式 闭卷 考试日期 单击或点击此处输入日期。 考试时间 _____

专业班级 _____ 学 号 _____ 姓 名 _____

题号	一	二	三	四	五	六	七	八	九	总分	核对人
分值	10	10	15	15	15	15	5	5	10	100	
得分											

分 数	
评卷人	

一、简答(10 分)

1. 请阐述什么是编译器，以及编译器在我国基础软件发展中的重要作用。（5 分）

参考答案：编译器是一种将源代码转化为等价的目标代码的程序。由词法分析器，语法分析器，语义分析器，优化器和代码生成器等组件组成。

编译器在我国基础软件发展中起着重要作用：、

提升软件开发效率：将高级语言代码转化为机器码或汇编代码，使得开发人员可以使用更高级、更易于理解和编写的代码来进行软件开发，提高开发效率和代码质量。

支持多样化的应用领域：编译器可以支持多种编程语言，为不同应用领域提供了更多的选择和灵活性。

促进技术创新和产业发展：编译器的研发和应用推动了编程语言、编译技术和软件工程等领域的创新和发展，以及人工智能、云计算、大数据等相关产业的发展。

加强软件安全和可靠性：编译器可以在编译过程中进行代码优化和静态分析，发现并修复潜在的安全漏洞和错误。它还可以生成更可靠、高效的目标代码，提高软件的性能和稳定性。

2. 请解释什么是编译器的前端和后端，以及这样设计有什么好处。(5 分)

参考解答：依赖于源语言而与目标机无关的部分称为前端，通常包括语法分析、语法分析、语义分析和中间代码生成及部分优化，以及出错处理和符号表管理。后端 指的是依赖于目标机器而一般不依赖于源语言，只与中间代码有关的部分，包括优化器和目标代码生成器。编译器的前端和后端分离设计提供了模块化、可移植性、灵活性和性能优化等优势，使得编译器更高效、可维护和可扩展。

解答内容不得超过装订线

分 数	
评卷人	

二、文法与语言(10 分)

1. 给定文法 G[S]:

S → Aab

A → Aab | B

B → a

请指出该文法类型（乔姆斯基分类），并给出该文法所描述的语言。(4 分)

参考答案：2 型文法（上下文无关文法）。L(G) = aab(ab)*

2. 给定文法 G[S]:

S → A

A → A+A | B++

B → y

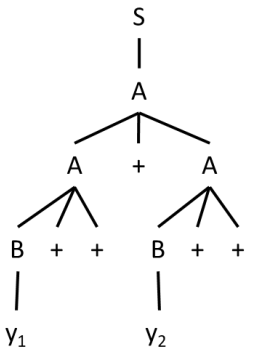
(1) 证明 y+++y++是文法的句子。(1 分)

(2) 画出句子的语法推导树。(2 分)

(3) 指出这个句子中的短语，直接(简单)短语和句柄。(3 分)

参考解答：(1) S ⇒A ⇒A+A ⇒B+++A ⇒y+++A ⇒y+++B++ ⇒y+++y++。故是文法的句子。

(2) 语法推导树：



注：下标可以没有

短语：y+++y++, y_1++, y_2++, y_1, y_2

直接短语：y_1, y_2

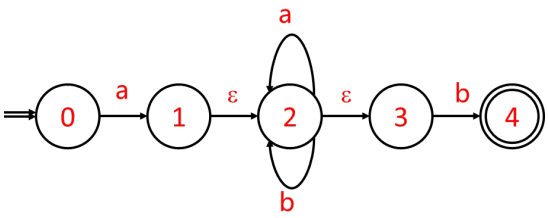
句柄:y_1(最左一个)

分 数	
评卷人	

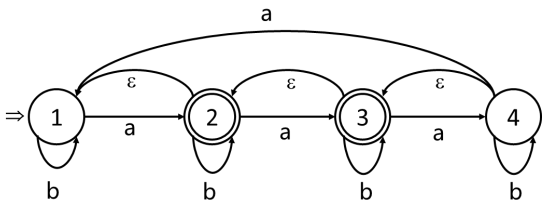
三、词法分析（15 分）

1．给定正规式 a(a|b)*b，构造其相应的 NFA（5 分）

参考答案：（直接给出 DFA 也可以）

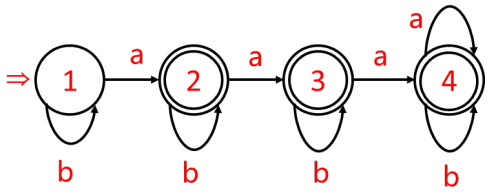


2．请将如下 NFA 转换成 DFA（6 分）

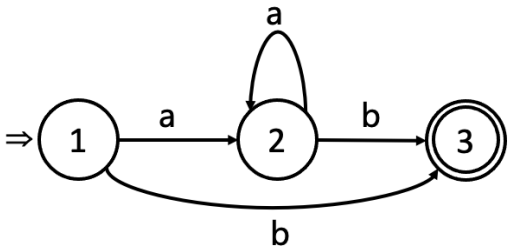


参考答案：

	a	b
{1}	{1, 2}	{1}
{1, 2}	{1, 2, 3}	{1, 2}
{1, 2, 3}	{1, 2, 3, 4}	{1, 2, 3}
{1, 2, 3, 4}	{1, 2, 3, 4}	{1, 2, 3, 4}

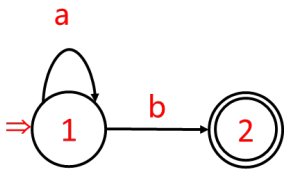


3. 请用分割法将如下 DFA 最小化（4 分）



参考答案:先分割成两组，A:{1，2}， B:{3}

	a	b
1	2	3
2	2	3
3		



可见，状态 1 和 2 等价，可合并，得最小 DFA。

解答内容不得超过装订线

分 数	
评卷人	

四、自顶向下的语法分析（15 分）

考虑文法 G（{∨， ∧， ¬，(,)，i }，{S,X,Y,E},{S}，P)，其中产生式集合 P 由下列产生式构成：

S→S∨X|X

X→X∧Y|Y

Y→¬E|E

E→(S)|i

(1) 判断文法 G[S]是不是 LL(1)文法，并说明理由；如果不是 LL(1)文法，写出与该文法等价的 LL(1)文法 G1[S]。

(2) 构造 G1[S]的 LL(1)分析表。

参考解答：

(1) 因为文法含有左递归，故不是 LL(1)文法。消除左递归后得等价的文法 G1[S]：

S→X S’　　{ ¬, (, i }

S’→∨X S’　　{ ∨ }

S’→ε　　{), # }

X→YX’　　{ ¬, (, i }

X’→∧Y X’　　{ ∧ }

X’→ε　　{ ∨,), # }

Y→¬E　　{ ¬ }

Y→E　　{ (, i }

E→(S)　　{ (}

E→i　　{ i }

(2) 上述各产生式右部给出了该产生式的 select 集，显见左部相同的产生式的 select 集交叉为空集，故消除左递归后的方法 G1[S]是 LL(1)文法。其 LL(1)分析表如下：

	∨	∧	¬	(	)	i	#
S			S→X S’	S→X S’		S→X S’	
S’	S’→∨XS’				S’→ε		S’→ε
X			X→YX’	X→YX’		X→YX’	
X’	X’→ε	X’→∧YX’			X’→ε		X’→ε
Y			Y→¬E	Y→E		Y→E	
E				E→(S)		E→i	

分 数	
评卷人	

五、LR 分析(15 分)

说明下面的文法 G[E]：

E→Fa|bFc|Gc|bGa

F→d

G→d

是 LR(1)的，但不是 LALR(1)的。

参考解答：

(1)拓展文法并对产生式编号：

r0：E'→E

r1：E→Fa

r2：E→bFc

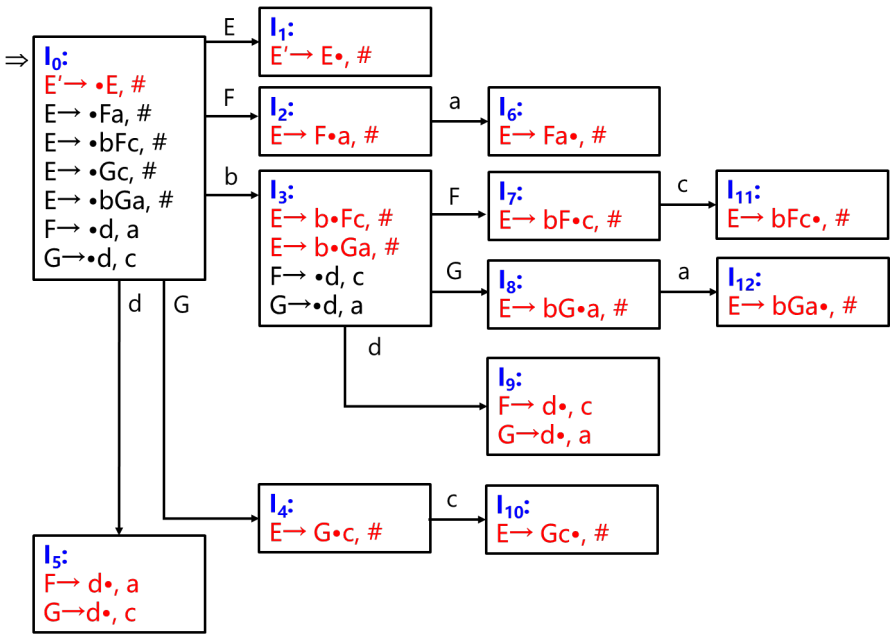
r3：E→Gc

r4：E→bGa

r5：F→d

r6：G→d

(2)构造拓广文法的 LR(1)项目集族和转换函数(DFA)：



(3) 显见，只有 I4,I8 项目集中各有两个规约项目，但向前搜索符号集没有交集，不存在归约-归约冲突，其它项目集都不存在移进-归约或归约=归约冲突。所以该文法是 LR(1)的。

(4) 上述 LR(1)项目集族中，存在同心项目集 I4 和 I8,合并后变成：

F→d•,a|c

G→d•,a|c

两个规约项目的向前搜索符号相同，显然存在归约-归约冲突。故不是 LALR(1)文法。

解答内容不得超过装订线

分 数	
评卷人	

六、语法制导的翻译模式及中间代码生成（15 分）。

下面是某语言文法的部分产生式及相应的 L-翻译模式片断：

S → id '=' A ';' { S.code := A.code || gen(id .place '=' A.place) }

S → 'while' '(' { E.true := newlabel; E.false := S.next } E ')' { S_1.next := newlabel} S_1

{ S.code := gen(S_1.next ':')|| E.code || gen(E.true ':')|| S_1.code || gen('goto' S_1.next) }

S →'if' '(' { E.true :=newlabel; E.false :=newlabel} E ')'

{ S_1.next := S.next } S_1 'else' { S_2.next := S.next } S_2

{ S.code := E.code|| gen(E.true ':') || S1.code|| gen('goto' S.next)

|| gen(E.false ':') || S2.code}

A → id {A.place := id.place; A.code := "" }

A → A_1 '-' A_2

{ A.place := newtemp; A.code := A_1.code || A_2.code || gen (A.place '=' A_1.place '-' A_2.place) }

E → id_1 rop id_2

{ E.code := gen ('if' id_1.place rop.op id_2.place 'goto' E.true) || gen ('goto' E. false) }

注：S代表语句,id代表标识符,A代表算术表达式,E代表布尔表达式, rop代表关系比较运算符(如'!= ', '>'等)。

语义属性说明:

id.place : 对应id的存储位置;

A.place: 用来存放A的值的存储位置

A.code|E.code : 对A|E求值的三地址代码序列;

E.true和E.false分别表示布尔表达式为真和假时，程序要跳转到的位置，即标号。

S.code : 对应于 S 的三地址代码序列 ;

S.next：表示 S 之后要执行的首条 TAC 语句的标号。

语义函数/过程说明：

gen() : 生成一条三地址代码;

newtemp : 在符号表中新建一个从未使用过的名字,并返回该名字的存储位置;

||: 是三地址代码序列之间的链接运算;

newlabel 返回一个新的语句标号。

(1) 根据题设给出的文法画出语句 **S:**

while (a != b) if (a>b) a = a - b; else b = b - a;

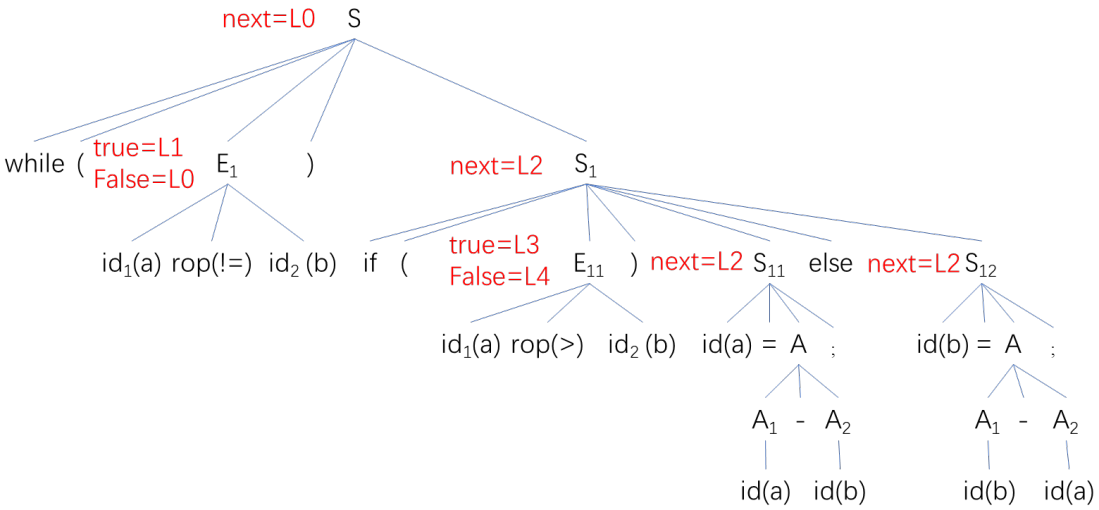
的语法分析树;（5分）

(2) 根据题设给出的翻译模式, 计算语法树各结点的继承属性值，设语句 **S** 的 **next** 属性值为 **'L0'**, newlabel 返回的第一个标号为 **L1, newtemp** 返回的第一个临时变量名为 **t1**。语法树中相同的符号请用下标以示区别，如：**S** 的产生式右部的 **S** 分别用 S_1，S_2 等区分，S_1 的产生式右部的 **S** 用 S_{11}，S_{12} 等区别。如果语法树有足够空间，你也可以将继承属性及其值标注在语法树上（**4** 分）

(3) 根据题设给出的翻译模式，将语句 **S** 翻译成三地址代码序列。（**6** 分）

解答内容不得超过装订线

参考解答：(1)语法分析树：（两个布尔表达式 E_1, E_{11}:1 分，S_{11}，S_{12}，S_1，S：各 1 分）



（2） S 有继承属性 next，E 有继承属性 true,false(各继承属性已标注在语法树上)：

$S.next = L0$;

$E_1.true = L1$, $E_1.false = L0$;

$S_1.next = L2$;

$E_{11}.true = L3$, $E_{11}.false = L4$;

$S_{11}.next = L2$, $S_{12}.next = L2$. （8 个继承属性，每个 0.5 分）

(3) 三地址码：

三地址码序列		while 语句	if 语句
L2:		$S_1.next$	
	if a != b goto L1 goto L0	$E_1.code$	
L1:		$E_1.true$	
	if a > b goto L3 goto L4	$S_1.code$	$E_{11}.code$
L3:			$E_{11}.true$
	t1 = a - b a = t1		$S_{11}.code$
	goto L2		goto $S_{11}.next$
L4:			$E_{11}.true$
	t2 = b - a b = t2		$S_{12}.code$
	goto L2	goto $S_1.next$	
L0:			

蓝色标注部分不是必须的。$E_1.code$, $E_{11}.code$, $S_{11}.code$, $S_{11}.code$:4 分，标号位置，跳转结构正确：2 分。

分 数	
评卷人	

七、运行时存储组织（5 分）。

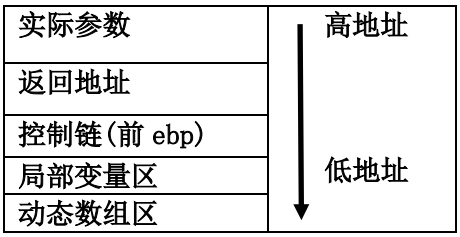
有 c 语言的程序如下：

```
#include <stdio.h>
int bonus = 52;
int f(int n){
    int a[n]; //定义变长数组
    int s = 0;
    if (n != 0){
        for (int i = 0; i < n; i++) {
            a[i] = i;
            s = s + i;
        }
    }
    /* 断点1 */
    return s;
}

int  main() {
    int sum;
    unsigned  int  size;
    scanf("%d", &size);
    sum = f(size);

    return sum + bonus;
}
```

该程序在 X86_64 机器上经某编译器(未开启任何优化选项)编译成 32 位目标代码后运行，并在运行时输入 4 作为 size 的值。设栈帧结构如下图所示(栈从高地址向低地址增长)：



int 型变量占 4 个字节，目标代码 32 位，故指针或地址亦为 4 字节， 字符占 1 个字节。

下表记录了一次实际运行时存储分配的部分情况，请在下表空白的地方填写当程序运行至“断点 1”时，静态数据区和函数 f 的栈帧的主要内容（符号/含义及值），填写时请参考已填写部分的格式,阴影部分不需要填写。

地址	内容	备注
		main 的栈帧
0xffffd248	控制链(前 ebp)	
0xffffd240	sum(未赋值)	
0xffffd23c	size=4	
0xffffd230	n=4	f 的栈帧
0xffffd22c	返回地址=0x804922d	
0xffffd228	控制链=0xffffd248	
0xffffd21c	s=6	
0xffffd218	i=4	
0xffffd214	a 的首地址=0xffffd200	
0xffffd20c	a[3]=3	
0xffffd208	a[2]=2	
0xffffd204	a[1]=1	
0xffffd200	a[0]=0	
		堆区
0x804c01c	bonus=52	静态数据区
		代码区

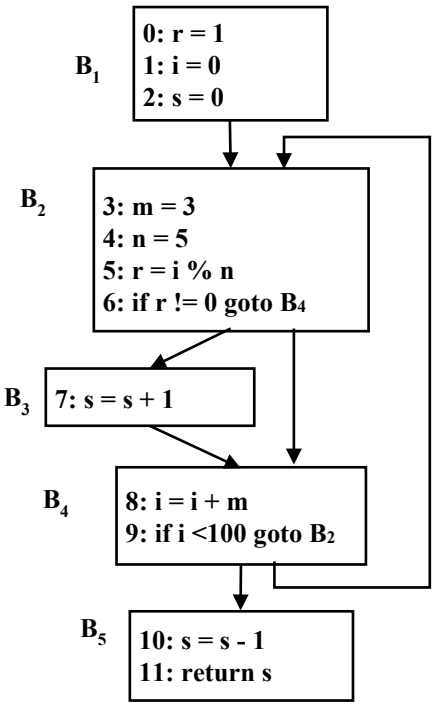
(注：填写内容包括变量 bonus，函数 f 的参数，数组 a 和变量 i 的位置和对应存储单元的十进制值.)

数组 2 分，其余每空 1 分。

分 数	
评卷人	

八、数据流图（5 分）。

有如下基本块与流图(B_1为入口基本块,B_5为出口基本块)：



(1) 请根据到达-定值数据流方程，迭代求解每个基本块入口和出口的定值点集合。将 Gen[B]，Kill[B]，以及 in[B]和 out[B]迭代结束的值直接填写在表中,假设 IN[B_1]= Ø。(3 分)

(2) 给出该流图中,变量 i 在引用点 8 的 UD 链。(1 分)

(3) 给出该流图中,变量 m 在定值点 3 的 DU 链。(1 分)

基本块 B	Gen[B]	Kill[B]	In[B]	Out[B]
B_1	{1, 2}	{6, 7, 9}	Ø	{1, 2}
B_2	{3, 4}	Ø	{1, 2, 3, 4, 6, 7}	{1, 2, 3, 4, 6, 7}
B_3	{6}	{2, 9}	{1, 2, 3, 4, 6, 7}	{1, 3, 4, 6, 7}
B_4	{7}	{1}	{1, 2, 3, 4, 6, 7}	{2, 3, 4, 6, 7}
B_5	{9}	{2, 6}	{2, 3, 4, 6, 7}	{3, 4, 7, 9}

参考解答：

(1) 见上表。(Gen/Kill 每项 0.1 分，In/Out 每项 0.2 分，计 3 分)

(2) {7} (1 分)

(3) {7} (1 分)

解答内容不得超过装订线

分 数	
评卷人	

九、中间代码优化（10 分）。

右图为三地址代码片断(代码片断结束时只有 r 是活跃的)：

```
1: i = 1
2: r = 0
3: if i < 10 goto 5
4: goto 18
5: p = 1
6: t1 = a + 4
7: t2 = 4 * i
8: t3 = t1[t2]
9: t4 = b + 4
10: t5 = 4 * i
11: t6 = t4[t5]
12: t7 = t3 * t6
13: t8 = r + t7
14: r = t8
15: t9 = i + p
16: i = t9
17: goto 3
18: return r
```

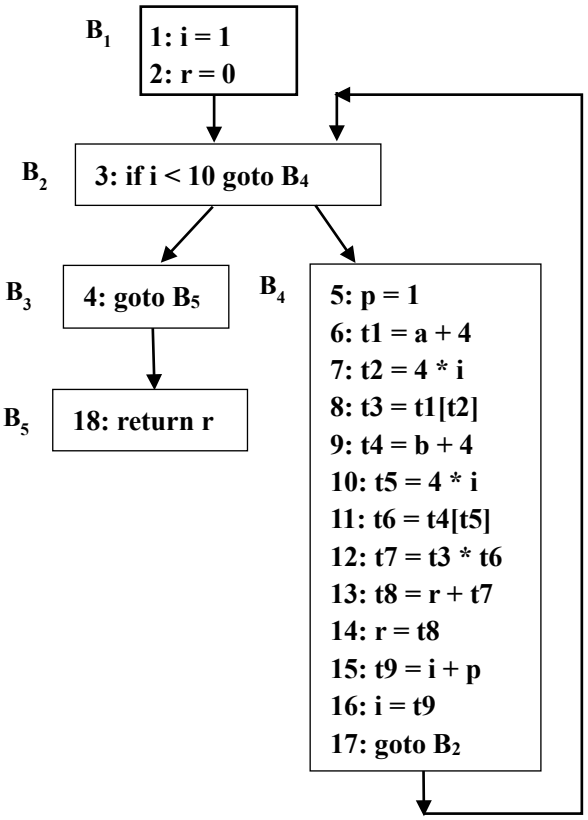
(1) 请将该三地址代码片断划分为基本块，并画出其流图。(3 分)

(2) 对(1)得出的基本块进行常量传播、删除公共子表达式、复写传播、删除死代码等优化；简述优化过程，给出优化后的代码序列。(3 分)

(3) 找出流图中的循环，对循环进行不变计算代码外提、归纳变量强度削弱、删除基本归纳变量等优化，简述优化过程，画出优化后流图。(4 分)；

参考解答：

(1) 基本块与流图(图 8C)：(3 分)



基本块与流图

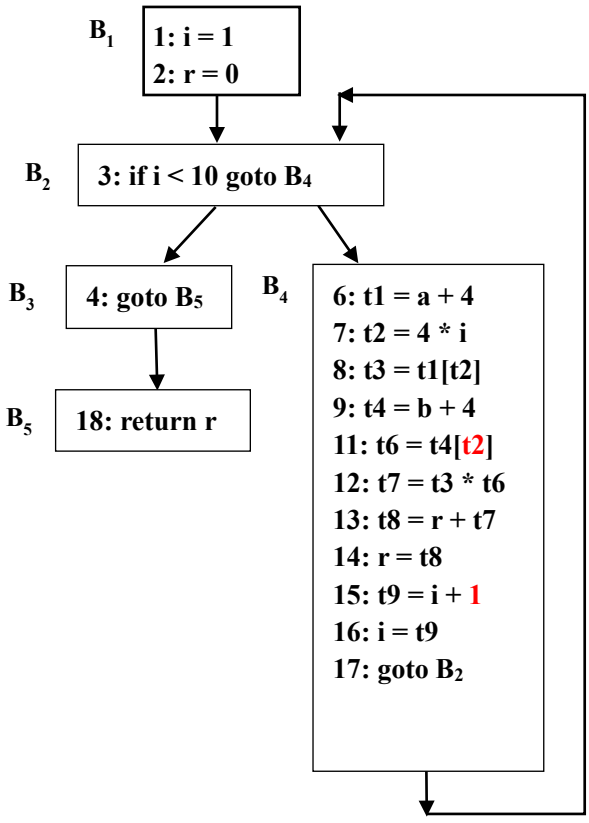


图 8D　B_4基本块优化后

(2) 变量 p 在引用点 15 的 UD 链={5}，即只有一个定值点 5 到达该引用点，且 p 的定值为常量 1，可执行常量传播，将语句 15 对 p 的引用，改为引用常量 1；语句 5 对 p 的定值不再使用，成为死代码。语句 10 中的表达式 4*i 是公共子表达式，可删除重复计算，并用 t2 代替：t5 = t2；对该复写语句进行传播：凡引用 t5 的地方改引用 t2。这导致 t5 在语句 10 的定值不再使用，删除死代码语句 5 和语句 10。优化后的流图如图 8D 所示。

常量传播：1 分；公共子表达式删除/复写传播：1 分； 死代码删除：1 分。

(3) $\{B_2, B_4\}$构成循环。语句 6 和 9 是循环不变计算，可外提。i 是基本归纳变量，t2 是与 i 同族的归纳变量，对 t2 的计算进行强度削弱，将 t2=4*i 改成 t2=t2+4，并在前置结点对 t2 初始化：t2=4*i。循环的终止条件 i<10 用 t2<40 代替后，可删除基本归纳变量 i。至此，t2=4*i 中对变量 i 的引用，只有语句 1 的定值可以到达，且定值是常量 1，可执行常量传播，并触发常量折叠和死代码删除： 4*1 被 4 取代，语句 1 对 i 的定值因其 DU 链为空而成为死代码，可删除。循环优化后的流图如图 8E 所示。最后一次常量传播、常量折叠和死代码删除是可选的，因题目未明确要求。

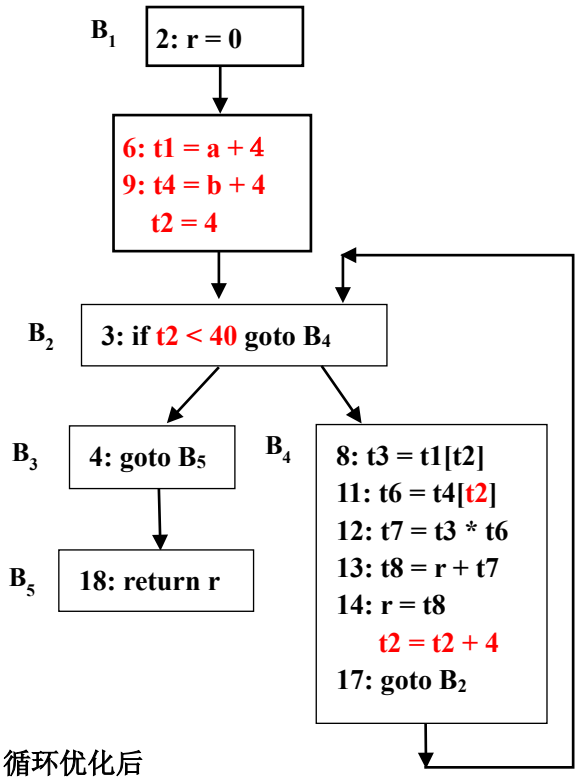


图 8E 循环优化后

循环识别：1 分；循环不变计算 6/9 的外提：1 分；t2 的强度消弱和初始化:1 分；循环终止条件的替代和归纳变量 i 的删除:1 分