

“编译原理”考试试卷（A卷）参考答案

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	15	15	10	5	15	15	5	10	100	
得分											

分 数	
评卷人	

一、简答（10分）

1. 计算机执行用高级语言编写的程序有哪两种方式？它们之间的主要区别是什么？（5分）

答：解释和编译。（2分）

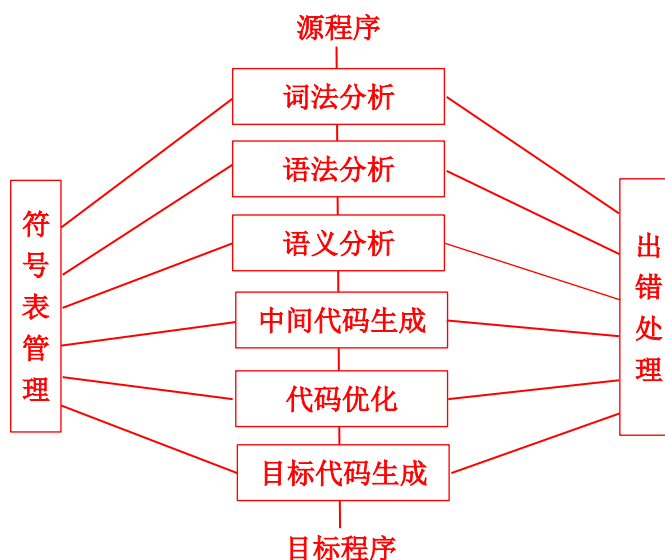
两种方式的根本区别在于：是否生成目标代码。（2分）

编译程序把源程序翻译成等价的目标程序，然后再执行目标程序，此后目标程序的运行不再依赖于编译程序。（0.5分）

解释程序则边翻译（解释）边执行，不产生目标代码，即按源程序中语句动态执行顺序逐句地进行分析解释，并立即予以执行，输出运行结果。源程序的每一次运行，都离不开解释程序。（0.5分）

2. 编译程序完成从源程序到目标程序的翻译工作，是分阶段进行的，每个阶段的任务由其对应的模块完成，各模块接力完成翻译工作，最终生成目标程序。在整个翻译工作中，还有一些特殊的模块伴随翻译的全过程。请画出一个典型的编译程序的结构框图。（5分）

答：



（5分）

分 数	
评卷人	

二、文法与语言（15 分）

1. 设有语言 $L(G) = \{a \mid a \in \{0, 1\}^+, \text{ 并且 } a \text{ 中的每个 } 1 \text{ 后面至少有 } 2 \text{ 个相继的 } 0 \text{ 直接跟随}\}$, 请构造生成 $L(G)$ 的正规文法。(7 分)

答: $G[Z]: Z \rightarrow 0Z \mid 1A \mid 0$

$A \rightarrow 0B$

$B \rightarrow 0Z \mid 0$ (7 分, 答案不唯一, 其它写法, 正确的同样得分。)

解析: a 中的每个 1 后面至少中跟随两个 0, 而对 0 后的 1 的个数没有要求, 所以在构造文法规则时, 注意每得到一个 1, 就让其后的符号负责产生两个 0。Z 生成满足要求的句子: 一旦 Z 利用 $Z \rightarrow 1A$ 推导一个 1, 则将由 A 负责产生两个 0 ($A \rightarrow 0B$ 产生一个 0, $B \rightarrow 0Z \mid 0$ 产生另一个 0) 结束或再次进入递归 (进入新一轮由 Z 开始的推导)。

2. 设有文法 $G[S]$:

$S \rightarrow (T) \mid a \mid \varepsilon$

$T \rightarrow T, S \mid S$

请给出句子 $(a, (a, a))$ 的规范推导过程, 并指出这个规范推导的逆过程 (归范规约) 每一步的句柄。(8 分)

答: 规范推导 归范规约的句柄

$S \Rightarrow (T)$	(T)
$\Rightarrow (T, S)$	T, S
$\Rightarrow (T, (T))$	(T)
$\Rightarrow (T, (T, S))$	T, S
$\Rightarrow (T, (T, a))$	a
$\Rightarrow (T, (S, a))$	S
$\Rightarrow (T, (a, a))$	最左的 a
$\Rightarrow (S, (a, a))$	S
$\Rightarrow (a, (a, a))$	最左的 a
(4 分)	(4 分)

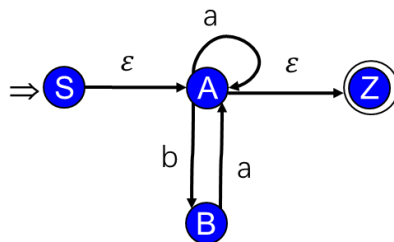
分 数	
评卷人	

三、词法分析（15 分）

设字母表 $\Sigma = \{a, b\}$ 上的正规表达式 $R = (a|ba)^*$ 。

- (1) 构造 NFA M' , 使得 $L(M') = L(R)$; (5 分)
- (2) 将 NFA M' 确定化、最小化, 得到 DFA M , 使得 $L(M) = L(M')$; (5 分)
- (3) 求右线性文法 G , 使得 $L(G) = L(M)$ 。(5 分)

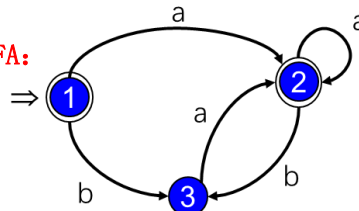
答: (1) NFA M' : (5 分, 如直接给出了 DFA, 下一步需说明已为 DFA, 且已最小化)



(2) 用子集法构造 DFA, 得子集的状态转移矩阵:

序号	子集	a	b
1	S, A, Z	A, Z	B
2	A, Z	A, Z	B
3	B	A, Z	

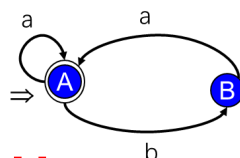
由状态转移矩阵, 并给状态子集重命名, 得 DFA:



DFA 最小化: 接受状态 $\{1, 2\}$ 为一组, 非接受态 $\{3\}$ 为另一组。状态 1, 2 输入 a 都到达 2, 输入 b 都到达 3, 这样, $\{1, 2\}$ 不可再分割。原状态集分割成以下两个子集:

$\{1, 2\}, \{3\}$

合并 $\{1, 2\}$ 并重命名为 A, 将 $\{3\}$ 重命名为 B, 得最小化的 DFA M : (5 分)



(3) 由 DFA M 得右线性文法 $G[A]$:

$A \rightarrow aA \mid bB \mid \epsilon$

$B \rightarrow aA$ (5 分, 状态取名不同, 将得到不同的语义等价产生式)

分 数	
评卷人	

四、自顶向下的语法分析（10 分）

设有文法 $G[S]$:

$S \rightarrow aBc \mid bAB$

$A \rightarrow aAb \mid b$

$B \rightarrow b \mid \varepsilon$

- (1) 证明 $G[S]$ 是 LL(1) 文法 (3 分);
- (2) 构造文法 $G[S]$ 的 LL(1) 分析表 (表项填写产生式右部, 可省略 “ \rightarrow ” 符号); (3 分)
- (3) 根据 LL(1) 分析器, 分析符号串 baabbb 是否该文法的句子, 列表写出分析过程, 列表内容包括: 步聚, 符号栈内容, 待分析串内容, 该步骤的执行的内容或结论。(4 分)

答: (1) 求左部相同的产生式的 select 集:

$\text{Select}(S \rightarrow aBc) = \{a\}; \text{Select}(S \rightarrow bAB) = \{b\}$ 。 $\{a\} \cap \{b\} = \emptyset$ 。

$\text{Select}(A \rightarrow aAb) = \{a\}; \text{Select}(A \rightarrow b) = \{b\}$ 。 $\{a\} \cap \{b\} = \emptyset$ 。

$\text{Select}(B \rightarrow b) = \{b\}; \text{Select}(B \rightarrow \varepsilon) = \text{Follow}(B) = \{c, \#\}$ 。 $\{b\} \cap \{c, \#\} = \emptyset$ 。

故文法 $G[S]$ 是 LL(1) 文法。(3 分)

- (2) 根据以上各产生式的 Select 集构造 LL(1) 分析表如下: (3 分)

	a	b	c	#
S	$S \rightarrow aBc$	$S \rightarrow bAB$		
A	$A \rightarrow aAb$	$A \rightarrow b$		
B		$B \rightarrow b$	$B \rightarrow \varepsilon$	$B \rightarrow \varepsilon$

- (3) 句子 baabbb 的分析过程见下表: (4 分)

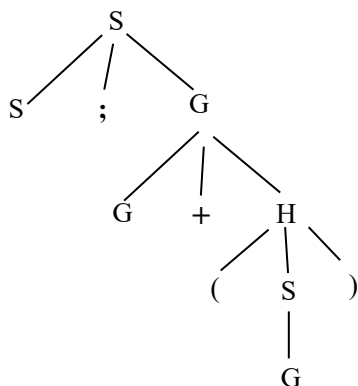
步骤	符号栈	待分析串	匹配的的产生式, 动作及结论
1	#S	baabbb#	$S \rightarrow bAB$
2	#BA b	baabbb#	终结符匹配出栈
3	#BA	aabbb#	$A \rightarrow aAb$
4	#BbA a	aabbb#	终结符匹配出栈
5	#BbA	abbb#	$A \rightarrow aAb$
6	#BbbA a	abbb#	终结符匹配出栈
7	#BbbA	bbb#	$A \rightarrow b$
8	#Bbbb	bbb#	终结符匹配出栈
9	#Bbb	bb#	终结符匹配出栈
10	#Bb	b#	终结符匹配出栈
11	#B	#	$B \rightarrow \varepsilon$
12	#	#	匹配成功, 接受! 句子是符合该文法的句子

分 数	
评卷人	

五、算符优先文法（5分）。

设有文法： $G[S]: S \rightarrow S;G \mid G \quad G \rightarrow G+H \mid H \quad H \rightarrow (S) \mid a$

和句型 $\#S;G+(G)\#$ 的语法树：



- (1) 试给出该句型的句柄；（1分）
- (2) 试给出该句型的最左素短语；（2分）
- (3) 试比较算符优先分析法与规范规约分析法的效率。（2分）

答：(1) G （1分）

(2) (G) （2分）

(3) 算符优先分析的效率高于规范规约分析的效率，原因是避免了非终结符的单符号规约。（2分）

解析：句型 $S;G+(G)$ 的短语有：‘ G ’，‘ (G) ’，‘ $G+(G)$ ’，‘ $S;G+(G)$ ’，只有 G 是唯一的直接短语，当然也是最左的，故为句柄。但其不含终结符，不是素短语。 (G) 才是，且是唯一的素短语（因为其它短语都包含 (G) ）。

分 数	
评卷人	

六、LR 分析（15分）。

设允许使用指针的赋值语句文法简化为：

$S \rightarrow A=E \quad S \rightarrow E \quad A \rightarrow *E \quad A \rightarrow i \quad E \rightarrow A$

- (1) 画出该文法的 LR(1) 项目集族和转换函数 (DFA)； （7分）
- (2) 画出该文法的 LR(1) 分析表 （7分）。
- (3) 分析 LR(0)、SLR(1)、LR(1)、LALR(1) 的表达能力关系。（1分）

答：(1) 先拓展文法：

(0) $S' \rightarrow S$

(1) $S \rightarrow A = E$

(2) $S \rightarrow E$

(3) $A \rightarrow * E$

(4) $A \rightarrow i$

(5) $E \rightarrow A$ （1分）

题外话：Follow(E)的值

$\because S \rightarrow A=E, S \rightarrow E, A \rightarrow *E$

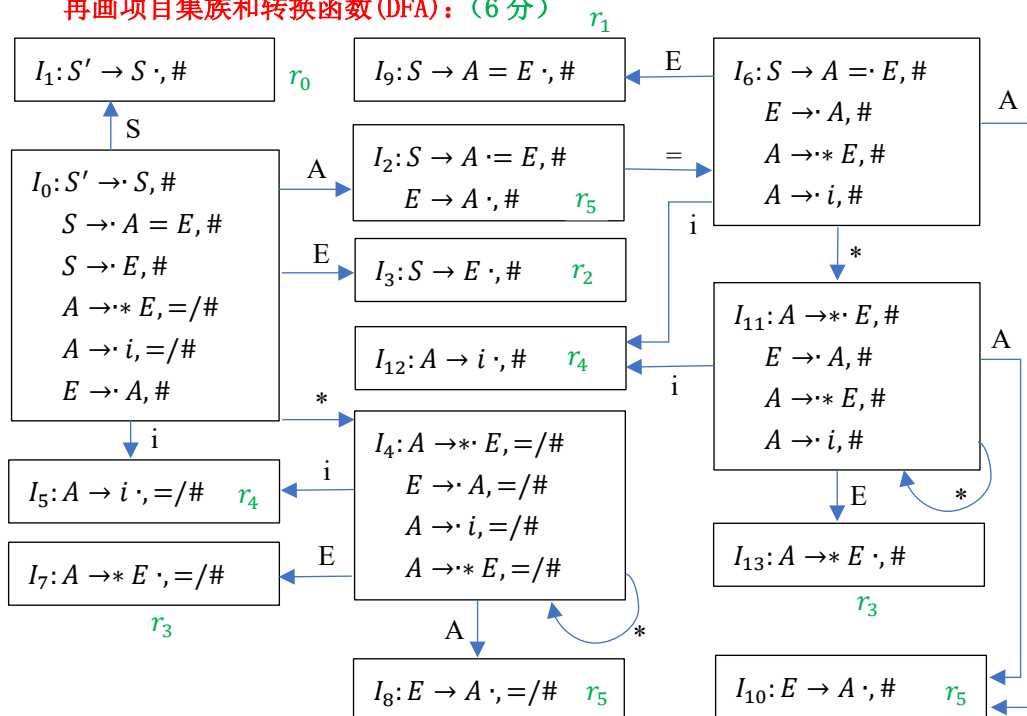
$\therefore \text{Follow}(E) \cup = \text{Follow}(S) \cup \text{Follow}(A)$

$\text{Follow}(S) = \text{Follow}(S') = \{\#\}$

$\text{Follow}(A) = \{=, \#\}$

$\therefore \text{Follow}(E) = \{=, \#\}$

再画项目集族和转换函数(DFA)：(6分)



题外话: 上图中状态 I_2 有移进/归约冲突, 移进符号集为 $\{=\}$, 而归约产生式左部 $\text{Follow}(E) = \{=, \#\}$, 二者有交集, 故本文法既不是 LR(0) 文法, 也不是 SLR(1) 文法。上图显示其为 LR(1) 文法, 也是 LALR(1) 文法 (同心集 I_4 与 I_{11} , I_5 与 I_{12} , I_7 与 I_{13} , I_8 与 I_{10} 合并后显无冲突)。

(2) LR(1) 分析表 (7分)

	ACTION				GOTO		
	=	*	i	#	S	A	E
0		S4	S5		1	2	3
1				acc			
2	S6			r5			
3				r2			
4		S4	S5			8	7
5	r4			r4			
6		S11	S12			10	9
7	r3			r3			
8	r5			r5			
9				r1			
10				r5			
11		S11	S12			10	13
12				r4			
13				r3			

(3) LR(1) 的表达能强于 LALR(1), LALR(1) 强于 SLR(1), SLR(1) 强于 LR(0)。(1分)

分 数	
评卷人	

七、语法制导的翻译模式及中间代码生成（15 分）。

下面是 C 语言文法的部分产生式及相应的语义动作集合：

```

S → id = E { S.code := E.code || gen(id.place ‘:=’ E.place) }
E → num {E.place := newtemp;
          E.code := gen (E.place ‘:=’ num.val) }
E → id {E.place := id.place;
        E.code := “ ” }
E → E1 + E2 { E.place := newtemp; E.code := E1.code || E2.code
                || gen (E.place ‘:=’ E1.place ‘+’ E2.place) }
E → { E1.true := E.true; E1.false := newlabel }E1 ||
    { E2.true := E.true; E2.false := E.false }E2
    {E.code := E1.code || gen (E1.false ‘:’ ) || E2.code }
E → { E1.true := newlabel; E1.false := E.false }E1 &&
    {E2.true := E.true; E2.false := E.false }E2
    {E.code := E1.code || gen (E1.true ‘:’ ) || E2.code }
E → id1 rop id2
    { E.code := gen ( ‘if’ id1.place rop.op id2.place ‘goto’ E.true )
    || gen ( ‘goto’ E.false) }
E → ( { E1.true := E.true; E1.false := E.false } E1 )
    { E.code := E1.code }

```

语义属性说明：

id.place : id 对应的符号表的存储位置；
E.place : 用来存放 E 的值的存储位置；
E.code : 对 E 求值的三地址代码序列；
S.code : 对应于 S 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，程序转移的目标位置。

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) 上述语义规则和动作采用的是什么类型的翻译模式；（1分）

(2) 在前述文法中增加对应for循环语句的产生式

SFOR → for (S₁;E;S₂) S₃

试给出该产生式相应的语义动作集合。（提示：增加S.next属性表示 S 之后要执行的首条三地址代码的标号，以及其它必要属性和语义动作）（4分）

(3) 根据题设给出的文法（结合（2））画出语句 S₀：

for (a = 0; a < b || (c<d && e>f) ; a = a + 1) b = 10 #

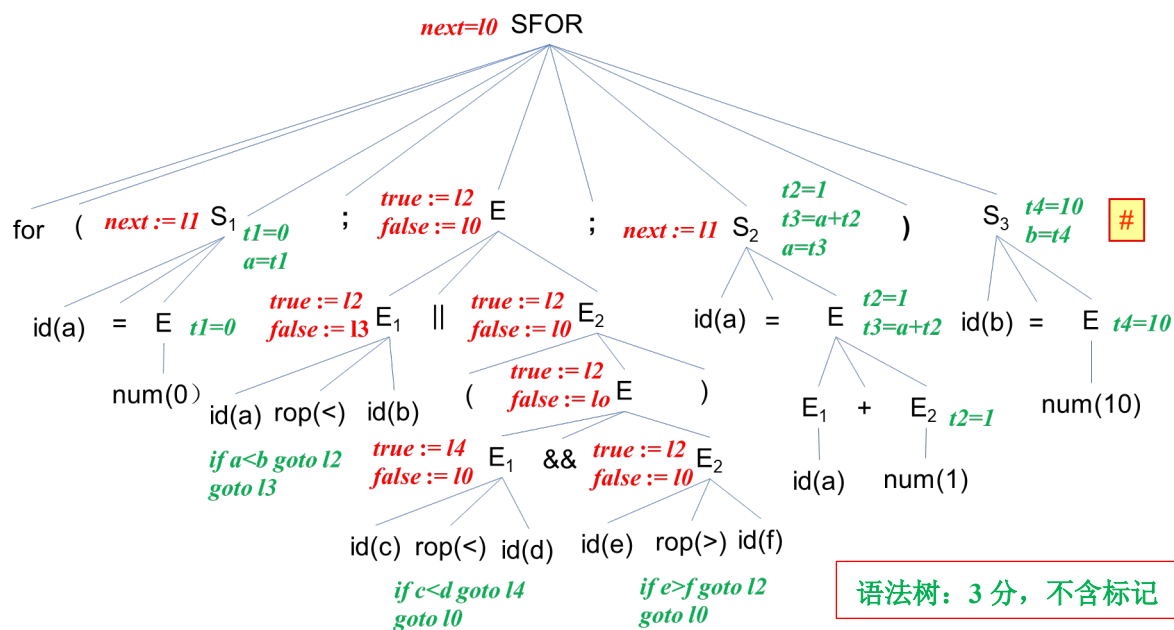
的语法分析树,#为句末符（表示语句到此结束）；（3 分）

(4) 根据题设给出的文法和语义规则（结合（2）），标注(3)的语法树各结点的继承属性值，并将语句 S₀ 翻译成三地址代码序列。（7 分）

答：(1) L 翻译模式；(1 分)

(2) SFOR \rightarrow for ({S1.next:=newlabel} S1;
 {E.true:=newlabel;E.false:=SFOR.next} E;
 {S2.next:=S1.next} S2) S3;
 {SFOR.code:=S1.code||gen(S1.next':')||E.code||gen(E.true':')
 ||S3.code||S2.code||gen('goto ' S1.next) } (4 分，翻译模式有别解)

(3) 语法树 (假设根结点的继承属性 SFOR.next=label10, 图中简写成 10, 下同)



图中红色继承属性，需要标出。绿色为综合属性，不必标出，此处标出是为了生成中间代码。

(4) 继承属性已标注在语法树上。生成的中间代码如下 (label10~label14 简写成 10~14)：

```

t1=0 }
a=t1 } =S1.code

11:
  if a<b goto l2
  goto l3
13:
  if c<d goto l4
  goto l0
14:
  if e>f goto l2
  goto l0
12:
  t4=10 }
  b=t4 } =S3.code

  t2=1 }
  t3=a+t2 } =S2.code
  a=t3 }
  goto 11
10:
  
```

继承属性标记：2 分

中间代码：5 分

label10~label14 简写成 10~14

temp1~temp4 简写成 t1~t4

代码右部的块标记仅为方便阅读，
不是答案的一部分。

如果设计的翻译模式不同，可能
会有不同继承属性，属性标注可以不同。

分 数	
评卷人	


八、运行时存储组织（5 分）。

现有 c++语言的程序片段：

```
void f(int i1, int& i2)
{
    int i3, i5;
    int i4[2];
    i3 = 26;
    i4[0] = 1;
    i4[1] = 9;
    i5 = i3 + i4[0] * i4[1];
    .../*程序运行点1*/
}

int main()
{
    int x, y;
    x = 3;
    y = 4;
    f(x, y);
    return 0;
}
```

假设某编译器给出的栈帧结构如下：

形式参数	高地址	 栈生长方向
返回地址		
相关寄存器		
局部变量区		
临时变量区	低地址	

假设该编译器没有开启任何优化选项，并且对于函数调用时，参数是从右向左依次入栈。程序经过该编译器编译成目标代码后在某 32 位平台上运行，当运行到“程序运行点 1”时，请填写函数 f 的栈帧内容(每个空行代表 4 个字节)。

1	变量 y 的地址 (1 分)
2	3 (变量 x 的值) (1 分)
	返回地址
	相关寄存器
3	26 (i3 的值)
4	35 (i5 的值) (1 分)
5	9 (i4[1] 的值)
6	1 (i4[0] 的值) (1 分)
7	9(临时变量 t1= i4[0] * i4[1])
8	35 (临时变量 t2= i3+t1) (1 分)

分 数	
评卷人	

九、代码优化（10 分）。

以下为中间代码片段，前面 L1~L12 是标号

```

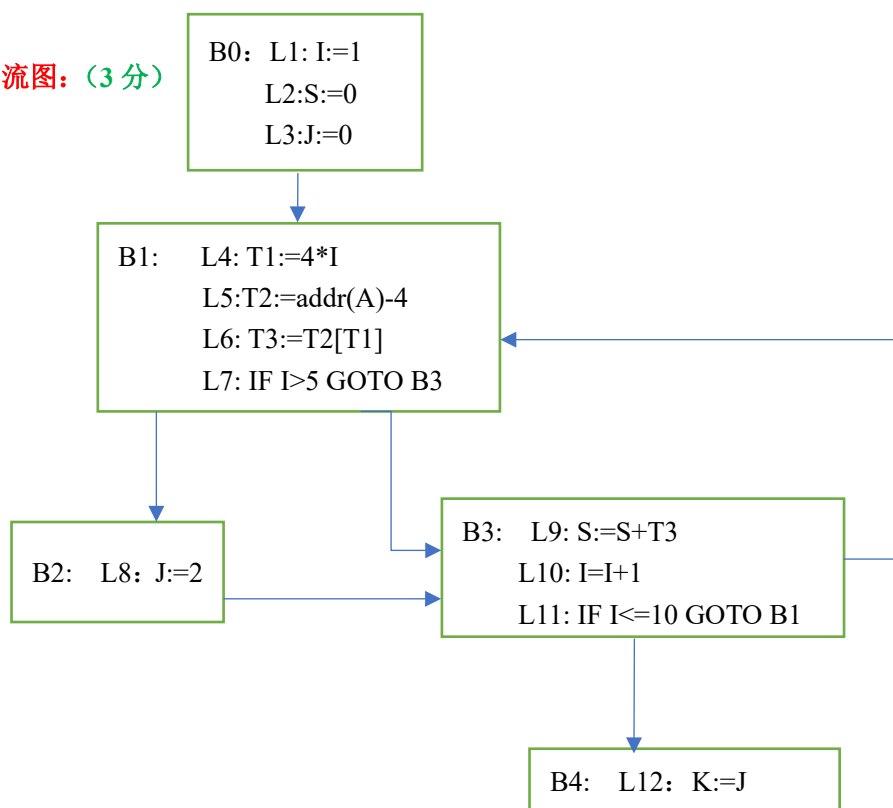
L1: I:=1
L2: S:=0
L3: J:=0
L4: T1:=4*I
L5: T2:=addr(A)-4
L6: T3:=T2[T1]
L7: IF I>5 GOTO L9
L8: J:=2
L9: S:=S+T3
L10: I=I+1
L11: IF I<=10 GOTO L4
L12: K:=J

```

- (1) 请将三地址码序列划分为基本块并给出流图(3 分)；
- (2) 找出流图中的循环，找出循环不变运算，并优化之(3 分)；
- (3) 找出循环中的归纳变量，并在可能的地方删除它(3 分)；
- (4) 浅谈目标代码优化和芯片发展之间有何关系(1 分)；

答：

(1) 基本块与流图：（3 分）



(2) 循环 {B1, B2, B3}

循环不变式: L8:J:=2, 不能外提; (不是循环出口的支配结点)

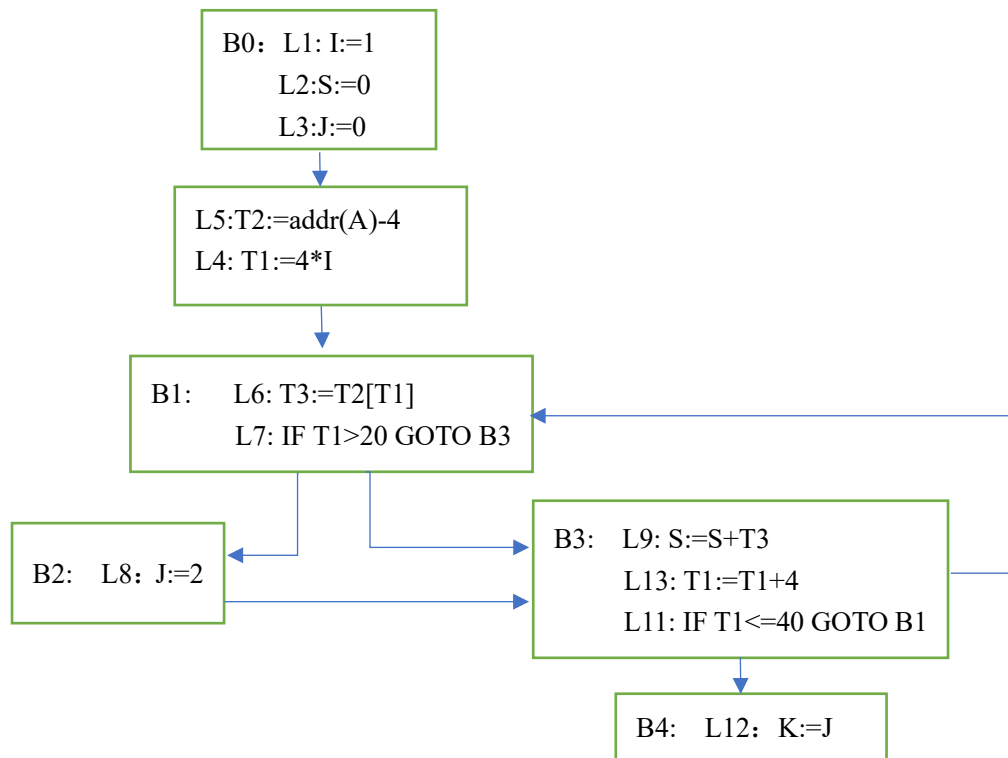
L5:T2:=addr(A)-4 可以外提 (3分)

(3) 归纳变量:

基本归纳量 I;

归纳量 T1, 可以消除 I。

优化后的结果是: (3分)



(4) 目标代码优化分为独立于机器的优化和依赖于机器的优化, 依赖于机器的优化与芯片高度相关, 如指令选择, 寄存器分配, 指令调度等。多个高性能处理器集成在一块芯片上已经是常态, 需要考虑并行优化和数据局部优化。(1分)