《编译原理》复习题

一、简答

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？

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。

二、文法与语言

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

2. 设有文法 G[S]:

S→(T)|a| ε

T→T,S|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。

三、词法分析

设字母表Σ＝{a,b}上的正规表达式 R=(a|ba)*。

(1) 构造 NFA M′,使得 L(M′)=L(R)；

(2) 将 NFA M′确定化、最小化，得到 DFA M， 使得 L(M)=L(M′)；

(3) 求右线性文法 G，使得 L(G)=L(M)。

四、自顶向下的语法分析

设有文法 G[S]:

S→aBc | bAB

A→aAb | b

B→b | ε

（1） 证明 G[S]是 LL(1)文法；

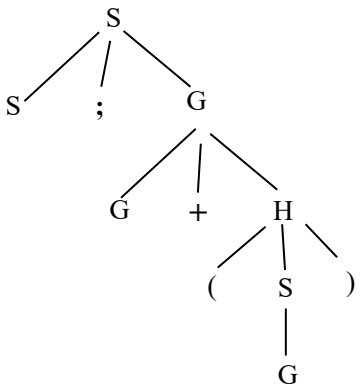
（2） 构造文法 G[S]的 LL(1)分析表（表项填写产生式右部，可省略“→”符号）；

（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。

五、算符优先文法。

设有文法：G[S]：S→S;G|G　　G→G+H|H　　H→(S)|a

和句型#S;G+(G)#的语法树：



(1) 试给出该句型的句柄；

(2) 试给出该句型的最左素短语；

(3) 试比较算符优先分析法与规范规约分析法的效率。

六、LR 分析。

设允许使用指针的赋值语句文法简化为：

S→A=E　S→E　A→*E　A→i　E→A

（1）画出该文法的 LR(1)项目集族和转换函数(DFA)；

（2）画出该文法的 LR(1)分析表。

（3）分析 LR（0)、SLR（1)、 LR（1)、LALR（1）的表达能力关系。

七、语法制导的翻译模式及中间代码生成。

下面是某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 → E_1 + E_2 { E.place := newtemp; E.code := E_1.code || E_2.code
　　|| gen (E.place ‘:=’ E_1.place ‘+’ E_2.place) }

E →{ E_1.true := E.true; E_1.false := newlabel }E_1 ||
　{ E_2.true := E.true; E_2.false := E.false }E_2
　{E.code := E_1 .code || gen (E_1.false ‘:’) || E_2 .code }

E →{ E_1.true := newlabel; E_1.false := E.false }E_1 &&

{E_2.true := E.true; E_2.false := E.false }E_2

{E.code := E_1 .code || gen (E_1.true ':') || E_2 .code }

E → $\underline{id_1}$ $\underline{rop}$ $\underline{id_2}$

{ E.code := gen ('if' $\underline{id_1}$.place $\underline{rop}$.op $\underline{id_2}$.place 'goto' E.true)

|| gen ('goto' E. false) }

E → ({ E_1.true := E.true; E_1.false := E.false } E_1)

{ E.code := E_1.code }

语义属性说明：

id.place : $\underline{id}$ 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)在前述文法中增加对应for循环语句的产生式

SFOR → for (S_1;E;S_2) S_3

试给出该产生式相应的语义动作集合。（提示：为语句S.next属性表示 语句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）

(2)根据题设给出的文法 (结合 (1))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) 根据题设给出的文法和语义规则(结合 (1))，标注(2)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。

八、运行时存储组织。

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：

实际参数	高地址	栈
返回地址		生
相关寄存器		长
局部变量区		方
临时变量区	低地址	向

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)。

返回地址
相关寄存器

九、代码优化。

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；

(2) 找出流图中的循环，找出循环不变运算，并优化之；

(3) 找出循环中的归纳变量，并在可能的地方删除它；

(4) 浅谈目标代码优化和芯片发展之间有何关系；