2001 年编译原理试题

1.（10 分）处于/* 和 */之间的串构成注解，注解中间没有*/。画出接受这种注解的 DFA 的状态转换图。

2.（10 分）为语言

$L = \{a^m b^n \mid 0 \le m \le 2n\}$（即 a 的个数不超过 b 的个数的两倍）

写一个 LR（1）文法，不准超过 6 个产生式。（若超过 6 个产生式，不给分。若所写文法不是 LR（1）文法，最多给 5 分。）

3.（10 分）构造下面文法的 LL（1）分析表。

$D \rightarrow TL$
$T \rightarrow int \mid real$
$L \rightarrow id\ R$
$R \rightarrow , id\ R \mid \varepsilon$

4.（15 分）就下面文法

$S \rightarrow (L) \mid a \qquad L \rightarrow L , S \mid S$

- 给出一个语法制导定义，它输出配对括号的个数。
- 给出一个翻译方案，它输出每个 a 的嵌套深度。

如句子(a, (a, a))，第一小题的输出是 2，第二小题的输出是 1 2 2。

5.（10 分）Pascal 语言 for 语句的含义见教材第 222 页习题 7.13。请为该语句设计一种**合理**的中间代码结构。你可以按第 215 页图 7.17 的方式或者第 219 页图 7.19 的方式写出你的设计，不需要写产生中间代码的语法制导定义。

6.（5 分）一个 C 语言程序如下：

```
func(i1,i2,i3)
long i1,i2,i3;
{
    long j1,j2,j3;
    printf("Addresses of i1,i2,i3 = %o,%o,%o\n",&i1,&i2,&i3);

    printf("Addresses of j1,j2,j3 = %o,%o,%o\n",&j1,&j2,&j3);

}
```

```
main()
{
    long i1,i2,i3;
    func(i1,i2,i3);
}
```

该程序在某种机器的 Linux 上的运行结果如下：

```
Addresses of i1,i2,i3 = 27777775460,27777775464,27777775470

Addresses of j1,j2,j3 = 27777775444,27777775440,27777775434
```

从上面的结果可以看出，func 函数的 3 个形式参数的地址依次升高，而 3 个局部变量的地址依次降低。试说明为什么会有这个区别。

7.（15 分）一个 C 语言程序及其在某种机器 linux 操作系统上的编译结果如下。根据所生成的汇编程序来解释程序中四个变量的作用域、生存期和置初值方式等方面的区别。

```
static long aa = 10;
short bb = 20;

func()
{
    static long cc = 30;
    short dd = 40;
}
```

```
    .file  "static.c"
    .version  "01.01"
gcc2_compiled.:
.data
    .align 4
    .type   aa,@object
    .size   aa,4
aa:
    .long 10
.globl bb
    .align 2
    .type   bb,@object
    .size   bb,2
```

```
bb:
    .value 20
    .align 4
    .type    cc.2,@object
    .size    cc.2,4
cc.2:
    .long 30
.text
    .align 4
.globl func
    .type    func,@function
func:
    pushl %ebp
    movl %esp,%ebp
    subl $4,%esp
    movw $40,-2(%ebp)
.L1:
    leave
    ret
.Lfe1:
    .size    func,.Lfe1-func
    .ident "GCC: (GNU) egcs-2.91.66 19990314/Linux (egcs-1.1.2 release)"
```

8.（10 分）C 语言是一种类型语言，但它不是强类型语言，因为编译时的类型检查不能保证所接受的程序没有运行时的类型错误。例如，编译时的类型检查一般不能保证运行时没有数组越界。请你再举一个这样的例子说明 C 语言不是强类型语言。

9.（10 分）如果在 A 机器上我们有 C 语言编译器 CC_A，也有它的源码 S_A（用 C 语言写成）。如何利用它通过尽量少的工作来得到 B 机器的 C 语言编译器 CC_B。

10.（5 分）表达式$(\lambda x.(\lambda yz.(x + y) + z)\ 3)\ 4\ 5$ 和$(\lambda x.(\lambda yz.(x + y) + z)\ 3\ 5)\ 4$ 有同样的结果。在抽象机 FAM 上，哪一个表达式对应的目标代码的执行效率高？为什么？

2003 年编译原理试题

1.（20 分）写出字母表Σ = {a, b}上语言 L = {*w* | *w* 中 a 的个数是偶数}的正规式，并画出接受该语言的最简 DFA。

2.（15 分）考虑下面的表达式文法，它包括数组访问、加和赋值：

E → E[E] | E + E | E = E | (E) | **id**

该文法是二义的。请写一个接受同样语言的 LR(1)文法，其优先级从高到低依次是数组访问、加和赋值，并且加运算是左结合，赋值是右结合。

3.（10 分）下面是产生字母表Σ = { 0, 1, 2}上数字串的一个文法：

S → D S D | 2

D → 0 | 1

写一个语法制导定义，它打印一个句子是否为回文数（一个数字串，从左向右读和从右向左读都一样时，称它为回文数）。

4.（10 分）教材上 7.2.1 节的翻译方案

P →　　{*offset* := 0}
　D

D → *D* ; *D*

D → **id** : *T*　　{ *enter*(**id**.*name*, *T.type*, *offset*); *offset* := *offset* + *T.width* }

T → **integer**　　{*T.type* := *integer*;　*T.width* := 4 }

T→ **real**　　{*T.type* := *real*;　*T.width* := 8 }

使用了变量 *offset*。请重写该翻译方案，它完成同样的事情，但只使用文法符号的属性，而不使用变量。

5.（5 分）一个 C 语言程序如下：

```
void fun(struct {int x; double r;} val) {      }

main()
{
    struct {int x; double r;} val;

    fun(val);
}
```

该程序在 X86/Linux 机器上的用 cc 命令编译时，报告的错误信息如下：

1: warning: structure defined inside parms

1: warning: anonymous struct declared inside parameter list

1: warning: its scope is only this definition or declaration,

1: warning: which is probably not what you want.
7: incompatible type for argument 1 of ‘fun’

请问，报告最后一行的错误的原因是什么？如何修改程序，使得编译时不再出现这个错误信息。

6.（10 分）一个 C 语言程序如下：

```
typedef struct _a{
        short   i;
        short   j;
        short   k;
}a;

typedef struct _b{
        long    i;
        short   k;
}b;

main()
{
        printf("Size of short, long, a and b = %d,%d,%d,%d\n",

                                sizeof(short), sizeof(long),sizeof(a),sizeof(b));
}
```

该程序在 X86/Linux 机器上的运行结果如下：

Size of short, long, a and b = 2, 4, 6, 8

已知 short 类型和 long 类型分别对齐到 2 的倍数和 4 的倍数。试问，为什么类型 b 的 size 会等于 8？

7.（15 分）一个 C 语言程序如下：

```
int fact(i)
int i;
{
    if(i==0)
        return 1;
    else
        return i*fact(i-1);
}

main()
{
```

```
    printf("%d\n", fact(5));
    printf("%d\n", fact(5,10,15));
    printf("%d\n", fact(5.0));
    printf("%d\n", fact());
}
```

该程序在 X86/Linux 机器上的运行结果如下：

```
120
120
1
Segmentation fault (core dumped)
```

请解释下面问题：

- 第二个 fact 调用：结果为什么没有受参数过多的影响？
- 第三个 fact 调用：为什么用浮点数 5.0 作为参数时结果变成 1?
- 第四个 fact 调用：为什么没有提供参数时会出现 Segmentation fault?

8.（5 分）C 语言的赋值操作并非仅对简单类型而言，例如若有类型声明 long a[100], b[100];，则赋值 a=b 是允许的。同样，若 a 和 b 是同一类型的两个结构，则赋值 a=b 也是允许的。

用教材上第七章所给出的三地址语句，我们能否为这种赋值产生中间代码？若你持肯定态度，请你给出对应这种赋值的中间代码序列；否则请你为这种赋值设计一种三地址语句。你所选用或设计的三地址语句要便于目标代码的生成。

9.（5 分）一个 C 程序的三个文件的内容如下：

```
head.h:
short int a = 10;

file1.c:
#include "head.h"

main()
{
}

file2.c:
#include "head.h"
```

在 X86/Linux 机器上的编译命令如下：

```
cc file1.c file2.c
```

编译结果报错的主要信息如下：

```
multiple definition of ‘a’
```

试分析为什么会报这样的错误。

10.（5 分）按照教材上介绍的方法，把下面 C++语言的函数翻译成 C 的函数。

```
void zoom (GraphicalObj &obj, double zoom_factor, Point &center) {
    obj.translate (–center.x, –center.y);      // 将中心点移至原点(0, 0)
        obj.scale (zoom_factor);                    // 缩放
}
```

2004 年编译原理试题

1.（20 分）写出字母表Σ = {a, b}上语言 L = {*w* | *w* 的最后两个字母是 *aa* 或 *bb*}的正规式，并画出接受该语言的最简 DFA。

2.（15 分）说明下面的文法不是 SLR(1)文法，并重写一个等价的 SLR(1)文法。

S → M a | b M c | d c | b d a

M → d

3.（10 分）为下面的语言写一个无二义的文法：

ML 语言中用分号分隔语句的语句块，例如：

((s ; s) ; (s ; s ; s) ; s) ; (s ; s)

4.（20 分）考虑一个类 Pascal 的语言，其中所有的变量都是整型（不需要显式声明），并且仅包含赋值语句、读语句、写语句，条件语句和循环语句。下面的产生式定义了该语言的语法（其中 **lit** 表示整型常量；OP 的产生式没有给出，因为它和下面讨论的问题无关）。

定义 Stmt 的两个属性：*MayDef* 表示它可能定值的变量集合，*MayUse* 表示它可能引用的变量集合。

（1）写一个语法制导定义或翻译方案，它计算 Stmt 的 *MayDef* 和 *MayUse* 属性。

（2）基于 *MayDef* 和 *MayUse* 属性，说明 $Stmt_1;Stmt_2$ 和 $Stmt_2;Stmt_1$ 在什么情况下有同样的语义。

Program → Stmt

Stmt → **id** := Exp

Stmt → read (**id**)

Stmt → write (Exp)

Stmt → Stmt ; Stmt
Stmt → **if** (Exp) **then begin** Stmt **end else begin** Stmt **end**

Stmt → **while** (Exp) **do begin** Stmt **end**
Exp → **id**
Exp → **lit**
Exp → Exp OP Exp

5.（10 分）下面是一个 C 语言程序：

```
main()
{
    long i;
    long a[0][4];
    long j;

    i = 4; j = 8;
    printf("%d, %d\n", sizeof(a), a[0][0]);
}
```

虽然出现 long a[0][4]这样的声明，在 X86/Linux 机器上该程序还是能通过编译并生成目标代码。请回答下面两个问题：

（1）sizeof(a)的值是多少，请说明理由。

（2）a[0][0]的值是多少，请说明理由。

6.（15 分）考虑下面的三地址语句序列：

```
        b := 1
        b := 2
        if w <= x goto L2
        e := b
        goto L2
L1:     goto L3
L2:     c := 3
        b := 4
        c := 6
L3:     if y <= z goto L4
        goto L5
L4:     g := g + 1
        h := 8
        goto L1
L5:     h := 9
```

（1）在该代码中用水平的横线将代码分成基本块，并给每个基本块一个序号。

（2）画出该代码的控制流图，每个基本块就用（1）的序号表示。

（3）若有循环的话，列出构成每个循环的结点。

7.（5 分）如果

（1）用编译命令 cc test.c 会报告有未定义的符号；

（2）用编译命令 cc test.c –lusr.a 会得到可执行程序（–lusr.a 表示连接库 libusr.a）。

那么，用编译命令 cc test.c –lusr.a –lusr.a 是否会报告有多重定义的符号？请说明理由。

8.（5 分）C++中的对象声明语句应如何翻译成 C 语句？如书上图 11.11 程序中的

Point _center;

应翻译成什么？

中 国 科 学 技 术 大 学

2004－2005 学年第二学期考试试卷

考试科目：编译原理和技术　　得分：_____

1.（15 分）

(a) 字母表Σ = { (,) }上的语言{ ( ), (( )( )), ((( ))), ( )( )( )( )( )}是不是正规语言？为什么？

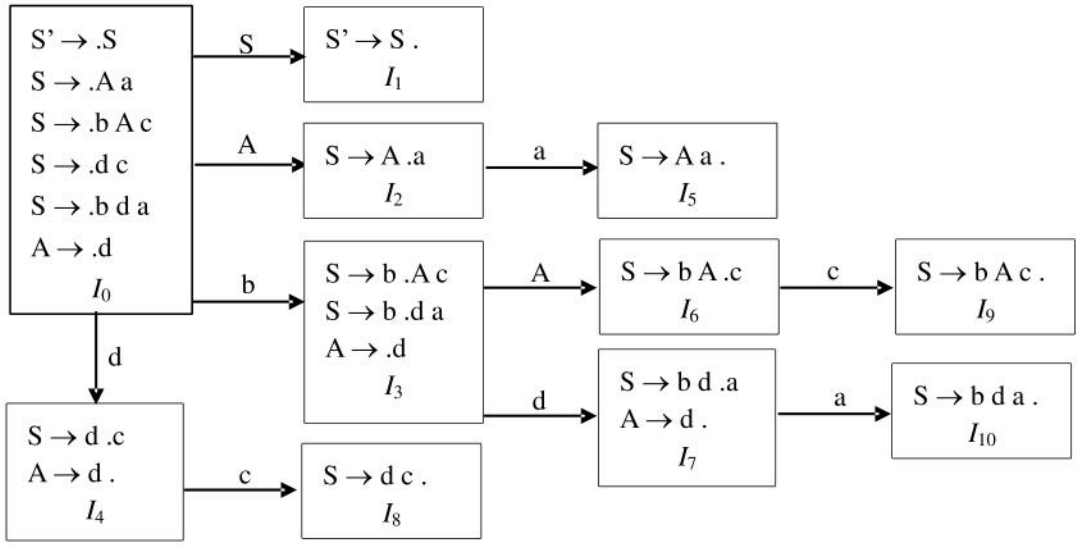
(b) 正规式 $(0 \mid 1)^*$ 和$((\varepsilon \mid 0) 1^*)^*$是否等价，说明理由。

2.（15 分）接受文法

$S \rightarrow A\ a \mid b\ A\ c \mid d\ c \mid b\ d\ a$

$A \rightarrow d$

活前缀的 DFA 见下图。请根据这个 DFA 来构造该文法的 SLR(1)分析表，并说明该文法为什么不是 SLR(1)文法。



3.（10 分）现有字母表Σ={ a }，写一个和正规式 a^*等价的上下文无关文法，要求所写的文法既不是 LR 文法，也不是二义文法。

4.（20 分）

(a) 下面的文法定义语言 $L = \{ a^n b^n c^m \mid m, n \geq 1\}$。写一个语法制导定义，其语义规则的作用是：对不属于语言 L 的子集 $L_1 = \{ a^n b^n c^n \mid n \geq 1\}$的句子，打印出错信息。

$S \rightarrow D\ C \qquad D \rightarrow a\ D\ b \mid a\ b \qquad C \rightarrow C\ c \mid c$

(b) 语句的文法如下：

$S \rightarrow$ **id** := E | **if** E **then** S | **while** E **do** S | **begin** S ; S **end** | **break**

写一个翻译方案，其语义动作的作用是：若发现 **break** 不是出现在循环语句中，及时报告错误。

5.（5 分）C 程序设计的教材上说，可以用两种形式表示字符串：其一是用字符数组存放一个字符串，另一种是用字符指针指向一个字符串。教材上同时介绍了这两种形式的很多共同点和不同点，但是有一种可能的区别没有介绍。下面是一个包含这两种形式的 C 程序：

```
char c1[]="good!";
char *c2="good!";

main()
{
    c1[0]= 'G';
    printf("c1=%s\n", c1);
    c2[0]= 'G';
    printf("c2=%s\n", c2);
```

```
}
```

该程序在 X86/Linux 机器上运行时的信息如下：

```
c1=Good!
Segmentation fault (core dumped)
```

请问，出现 Segmentation fault 的原因是什么？

6.（15 分）下面是一个 C 语言程序：

```
long f1( i )
long i;
{
    return(i*10);
}

long f2(long i)
{
    return(i*10);
}

main()
{
    printf("f1 = %d, f2 = %d\n", f1(10.0), f2(10.0) );
}
```

其中函数 f1 和 f2 仅形式参数的描述方式不一样。该程序在 X86/Linux 机器上的运行结果如下：

```
f1 = 0, f2 = 100
```

请解释为什么用同样的实在参数调用这两个函数的结果不一样。

7.（10 分）下面是一个 C 语言程序和在 X86/Linux 机器上编译（未使用优化）该程序得到的汇编代码（为便于理解，略去了和讨论本问题无关的部分，并改动了一个地方）。

(a) 为什么会出现一条指令前有多个标号的情况，如.L2 和.L4，还有.L5、.L3 和.L1？从控制流语句的中间代码结构加以解释。

(b) 每个函数都有这样的标号.L1，它的作用是什么，为什么本函数没有引用该标号的地方？

```
main()
{
        long i,j;

        if ( j )
            i++;
        else
            while ( i ) j++;
}
```

```
main:
        pushl %ebp              ——将老的基地址指针压栈
        movl %esp,%ebp          ——将当前栈顶指针作为基地址指针
        subl $8,%esp            ——为局部变量分配空间
        cmpl $0,-8(%ebp)
        je .L2
        incl -4(%ebp)
        jmp .L3
.L2:
.L4:
        cmpl $0,-4(%ebp)
        jne .L6
        jmp .L5
.L6:
        incl -8(%ebp)
        jmp .L4
.L5:
.L3:
.L1:
        leave                   ——和下一条指令一起完成恢复老的基地址指针，将栈顶
        ret                     ——指针恢复到调用前参数压栈后的位置，并返回调用者
```

8.（5 分）cc 是 UNIX 系统上 C 语言编译命令，–l 是连接库函数的选择项。两个程序员分别编写了函数库 libuser1.a 和 libuser2.a。当用命令

```
cc  test.c  –luser1.a  –luser2.a
```

编译时，报告有重复定义的符号。（备注：库名中的 lib 在命令中省略。该命令和命令 cc test.c libuser1.a libuser2.a 的效果是一致的）。而改用命令

```
cc  test.c  –luser2.a  –luser1.a
```

时，能得到可执行程序。试分析原因。

9.（5 分）根据教材上所介绍的方法，C++中的对象声明语句应如何翻译成 C 语句？如教材图 11.11（旧教材的图 10.11）程序中的

```
Point  _center;
```

应怎样翻译？

中 国 科 学 技 术 大 学

2005－2006 学年第二学期考试试卷

考试科目：编译原理和技术　　得分：_____

学生所在系：_____ 姓名：_____ 学号：_____

1、（15 分）

(a) 用正规式表示字母表{a, b}上，a 不会相邻的所有串。

(b) 画出一个最简的确定有限自动机，它接受所有大于 101 的二进制整数。

2、（10 分）构造下面文法的 LL(1)分析表。

S → a B S | b A S | ε

A → b A A | a

B → a B B | b

3、（10 分）下面的文法是二义文法

S → E

E → **while** E **do** E | **id** := E | E + E | **id** | (E)

请你为该语言重写一个规范的 LR(1)文法，它为该语言中的各种运算体现通常的优先级和结合规则。不需要证明你的文法是规范 LR(1)的。

4、（10 分）为下面文法写一个语法制导的定义，它完成一个句子的 **while-do** 最大嵌套层次的计算并输出这个计算结果。

S → E

E → **while** E **do** E | **id** := E | E + E | **id** | (E)

5、（15 分）考虑一个类似 Pascal 的语言，其中所有的变量都是整型（不需要显式声明），并且仅包含赋值语句、读语句、写语句、条件语句和循环语句。下面的产生式定义了该语言的语法（其中 **lit** 表示整型常量；OP 的产生式没有给出，因为它和下面讨论的问题无关）。

定义 Stmt 的两个属性：*Def* 表示在 Stmt 中一定会定值且在该定值前没有引用的变量集合，*MayUse* 表示在 Stmt 中有引用且在该引用前可能没有定值的变量集合。

(a) 写一个语法制导定义或翻译方案，它计算 Stmt 的 *Def* 和 *MayUse* 属性。

(b) 基于上面的计算，程序可能未赋初值的变量集合从哪儿可以得到？

可能未赋初值的变量是这样定义的：若存在从程序开始点到达变量 a 某引用点的一条路径，在这条路径上没有对变量 a 赋值，则变量 a 属于程序可能未赋初值的变量集合。

Program → Stmt

Stmt → **id** := Exp

Stmt → read (**id**)

Stmt → write (Exp)

Stmt → Stmt ; Stmt

Stmt → **if** (Exp) **then begin** Stmt **end else begin** Stmt **end**

Stmt → **while** (Exp) **do begin** Stmt **end**

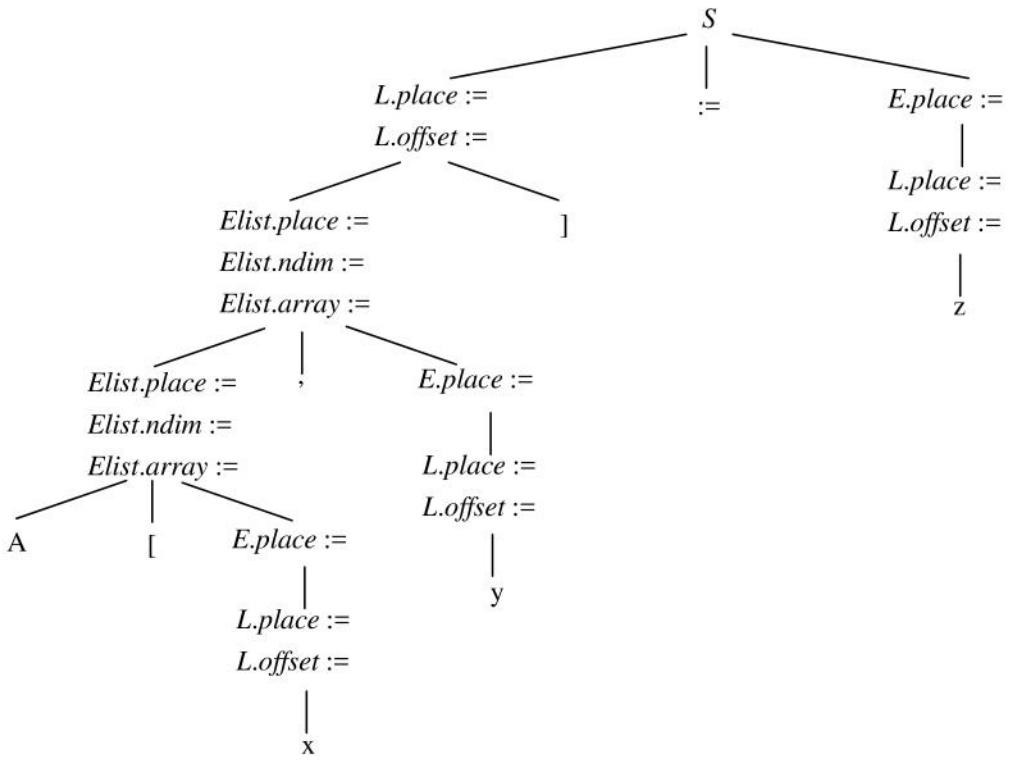
Exp → **id**

Exp → **lit**

Exp → Exp OP Exp

6、（15 分）赋值语句 A[x, y]:= z（其中 A 是 10 × 5 的数组）的注释分析树如下图。请根据

教材上 7.3.4 节的翻译方案，把图中的属性值都补上（像图 7.9 那样），并且把每步归约产生的中间代码写在相应产生式的旁边。



7、（15 分）通常，函数调用的返回值是简单类型时，用寄存器传递函数值。当返回值是结构类型时需要采用别的方式。下面是一个 C 语言文件和它在 x86/Linux 上经某版本 GCC 编译器编译生成的汇编代码。（备注，该汇编码略经修改，以便于阅读。该修改没有影响结果。）

(a) 请你分析这些代码，总结出函数返回值是结构类型时，返回值的传递方式。

(b) 若 m 函数的语句 s = f(10)改成 s.i = f(10).i + f(20).i + f(30).i，你认为 m 函数的局部存储分配应该怎样修改，以适用该语句的计算。

源文件 return.c 的内容如下：

```
typedef struct{long i;}S;

    S f(k)
    long k;
    {
            S s;
            s.i =k;
            return s;
    }

    m()
    {
```

```
        S s;
        s.i = 20;
        s = f(10);
}
```

汇编文件 return.s 的内容如下：

```
    .file  "return.c"
    .text
.globl f
    .typef, @function
f:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $4, %esp
    movl    8(%ebp), %eax
    movl    12(%ebp), %edx
    movl    %edx, -4(%ebp)
    movl    -4(%ebp), %edx
    movl    %edx, (%eax)
    leave
    ret
    .size f, .-f
.globl m
    .typem, @function
m:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $4, %esp
    movl    $20, -4(%ebp)
    leal    -4(%ebp), %eax
    pushl   $10
    pushl   %eax
    call  f
    addl    $8, %esp
    leave
    ret
    .size m, .-m
    .section    .note.GNU-stack,"",@progbits
    .ident      "GCC: (GNU) 3.3.5 (Debian 1:3.3.5-13)"
```

8、(5 分)把下面左边的文件 file1.c 提交给编译器，编译器没有报告任何错误。而把文件 file2.c 提交给编译器，错误报告如下：

file2.c: 2: error: conflicting types for ‘func’

file2.c: 1: error: previous declaration of ‘func’

试分析原因。(在这两个文件中，第 1 行都是函数 func 的原型，第 2 行都是函数 func 的定

义，函数体为空。）

file1.c

```
int func(double);
int func(f) float f; {}
```

file2.c

```
int func(double);
int func(float f) {}
```

9、（5 分）教材上第 342 页倒数第 7 行说“将 C++语言中一个类的所有非静态属性构成一个 C 语言的结构类型，取类的名字作为结构类型的名字”。在这一章都学过后，你认为这句话需要修改吗？

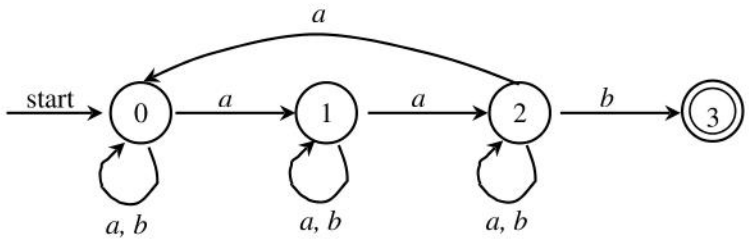
中 国 科 学 技 术 大 学

2006－2007 学年第二学期考试试卷（A）

考试科目：编译原理和技术　　得分：_____

学生所在系：_____ 姓名：_____ 学号：_____

1、（10 分）用子集构造法给出由下面的 NFA 得到的 DFA 的转换表



2、（20 分）（1）通过构造识别活前缀的 DFA 和构造分析表，来证明文法 E → E + **id** | **id** 是 SLR(1)文法。

（2）下面左右两个文法都和（1）的文法等价

E → E + M **id** | **id**　　E → M E + **id** | **id**

M → ε　　M → ε

请指出其中有几个文法不是 LR(1)文法，并给出它们不是 LR(1)文法的理由。

3、（10 分）下面是 C 语言两个函数 f 和 g 的概略（它们不再有其它的局部变量）：

```
int f (int x) { int i; …return i +1; … }
int g (int y) {int j; … f (j +1); … }
```

请按照教材上例 6.5 中图 6.13 的形式，画出函数 g 调用 f，f 的函数体正在执行时，活动记录栈的内容及相关信息，并按图 6.12 左侧箭头方式画出控制链。假定函数返回值是通过寄存器传递的。

4、（10 分）C 语言函数 f 的定义如下：

```
int f (int x, int *py, int **ppz)
{
    **ppz +=1; *py +=2; x +=3; return x + *py + **ppz;
}
```

变量 a 是指向 b 的指针，变量 b 是指向 c 的指针，c 是整型变量并且当前值是 4。那么执行 f (c, b, a)的返回值是多少？

5、（10 分）Java 语言的实现通常把对象和数组都分配在堆上，把指向它们的指针分配在栈上，依靠运行时的垃圾收集器来回收堆上那些从栈不可达的空间（垃圾）。这种方式提高了语言的安全性，但是增加了运行开销。编译时能否采用一些技术，以降低垃圾收集所占运行开销？概述你的方案。

6、（5 分）在面向对象语言中，编译器给每个对象分配空间的第 1 个域存放虚方法表的指针。是否可以把虚方法表指针作为最后 1 个域而不是第 1 个域？请简要说明理由。

7、（15 分）考虑一个简单语言，其中所有的变量都是整型（不需要显式声明），并且仅包含

赋值语句、读语句和写语句。下面的产生式定义了该语言的语法（其中 **lit** 表示整型常量；OP 的产生式没有给出，因为它和下面讨论的问题无关）。

Program → StmtList
StmtList → Stmt StmtList | Stmt
Stmt → **id** := Exp; | read (**id**); | write (Exp);
Exp → **id** | **lit** | Exp OP Exp

我们把不影响 write 语句输出值的赋值（包括通过 read 语句来赋值）称为无用赋值，写一个语法指导定义，它确定一个程序中出现过赋予无用值的变量集合（不需要知道无用赋值的位置）和没有置初值的变量集合（不影响 write 语句输出值的未置初值变量不在考虑之中）。

非终结符 StmtList 和 Stmt 用下面 3 个属性（你根据需要来定义其它文法符号的属性）：

（1）uses_in：在本语句表或语句入口点的引用变量集合，它们的值影响在该程序点后的输出。

（2）uses_out：在本语句表或语句出口点的引用变量集合，它们的值影响在该程序点后的输出。

（3）useless：本语句表或语句中出现的无用赋值变量集合。

8、（10 分）一个 C 文件 array.c 仅有下面两行代码：

```
char a[ ][4] = { “123”, “456”};
char *p[ ]   = { “123”, “456”};
```

从编译生成的下列汇编代码可以看出对数组 a 和指针 p 的存储分配是不同的。试依据这里的存储分配，为置了初值后的数组 a 和指针 p 写出类型表达式。

```
        .file   “array.c”
.globl a
        .data
        .type a, @object
        .size a, 8
a:
        .string “123”
        .string “456”
        .section    .rodata
.LC0:
        .string “123”
.LC1:
        .string “456”
.globl p
        .data
        .align 4
        .type p, @object
        .size p, 8
p:
        .long .LC0
        .long .LC1
```

```
        .section        .note.GNU-stack, "", @progbits
        .ident      "GCC: (GNU) 3.3.5 (Debian 1:3.3.5-13)"
```

9、（10 分）两个 C 文件 long.c 和 short.c 的内容分别是

```
long i = 32768*2;
```

和

```
extern short i;
main() { printf("%d\n", i); }
```

在 X86/Linux 机器上，用 cc long.c short.c 命令编译这两个文件，能否得到可执行目标程序？若能得到目标程序，运行时是否报错？若不报错，则运行结果输出的值是否为 65536？若不等于 65536，原因是什么？

2007－2008 学年第二学期考试试卷（A）

考试科目：编译原理和技术　　得分：_____

1、（10 分）用正规式表示字母表{a, b}上 a 不会相邻的所有句子的集合，并给出接受该语言的最简 DFA。

2、（15 分）（1）为下面文法构造规范 LR(1)分析表，画出像教材上图 3.19 这样的状态转换图就可以了。

$S \to V = E \mid E$

$V \to * E \mid \mathbf{id}$

$E \to V$

（2）上述状态转换图有同心项目集吗？若有，合并同心项目集后是否会出现动作冲突？

3、（10 分）为字母表{0, 1}上的回文数集合写一个 LR 文法；若你认为该语言不存在 LR 文法，则说明理由。（注：一个数字串，从左向右读和从右向左读都一样时，称它为回文数。）

4、（10 分）下面的翻译方案计算 0 和 1 的串的值（解释为二进制的正整数）。

$B \to B_1\ 0$ { $B.val = B_1.val \times 2$ }

$B \to B_1\ 1$ { $B.val = B_1.val \times 2 + 1$ }

$B \to 1$ { $B.val = 1$ }

重写该翻译方案，使得它基础文法没有左递归，并且为整个输入串计算的 *B.val* 和原来的一样。

5、（10 分）若布尔表达式有异或 **xor**（exclusive-or）运算（异或运算的结果为真，当且仅当它的一个运算对象为真），请按照教材上表 7.4 的风格给出异或运算的语义规则。（备注：该题目设计得不好。）

6、（5 分）在面向对象语言中，编译器给每个类建立虚方法表，如教材上图 11.3 和图 11.4 那样。请简要说明，为什么编译器给每个类仅建立虚方法表，而不是建立所有方法的方法表。

7、（15 分）C 语言和 Java 语言的数组声明和数组元素引用的语法形式同教材上 7.3.3 节和 7.3.4 节讨论的不一样，例如 float A[10][20]和 A[i+1][j-1]，并且每一维的下界都是 0。若适应这种情况的赋值语句的文法如下：

$S \to L := E$

$E \to E + E \mid (E) \mid L$

$L \to L\ [E\ ] \mid \mathbf{id}$

（1）重新设计教材上 7.3.3 节数组元素的地址计算公式，以方便编译器产生数组元素地址计算的中间代码。不要忘记每一维的下界都是 0。

（2）重新设计数组元素地址计算的翻译方案。只需写出产生式 $L \to L\ [E\ ] \mid \mathbf{id}$ 的翻译方案，但要能和 7.3.4 节中产生式 $S \to L := E$ 和 $E \to E + E \mid (E) \mid L$ 的翻译方案衔接。若翻译方

案中引入新的函数调用，要解释这些函数的含义。

8、（15 分）一个 C 语言的文件如下：

```
func(long i, long j, long k) {
    k = (i + j) – (i – j – f(k));
}
```

经 GCC 3.4.6 编译器编译得到的汇编代码分两列在下面给出：

```
        .file   “call.c”                |       movl    16(%ebp), %eax
        .text                           |       movl    %eax, (%esp)
.globl func                             |       call    f
        .type       func, @function     |       subl    %eax, %ebx
func:                                   |       movl    %ebx, %eax
        pushl   %ebp                    |       subl    %eax, %esi
        movl    $esp, %ebp              |       movl    %esi, %eax
        pushl   %esi                    |       movl    %eax, 16(%ebp)
        pushl   %ebx                    |       popl    %ebx
        movl    12(%ebp), %eax          |       popl    %esi
        movl    8(%ebp), %esi           |       popl    %ebp
        addl    %eax, %esi              |       ret
        movl    12(%ebp), %edx          |       .size   func, .-func
        movl    8(%ebp), %eax           |       .section    .note.GNU-stack,””,@progbits

        movl    %eax, %ebx              |       .ident  “GCC: (GNU)  3.4.6”
        subl    %edx, %ebx
```

请根据上述汇编代码，并参考教材上例 6.5 和第 6 章到第 9 章习题中的汇编代码进行总结：为适应函数调用引起的跨函数执行，该编译器在寄存器的值的保护方面有些什么约定？

9、（10 分）下面左边的函数被 GCC: (GNU) 3.3.5 (Debian 1:3.3.5-13)优化成右边的代码。若你认为该优化结果不对或该优化不合理，则阐述你的理由。若你认为该优化结果是对的，请说明实施了哪些优化，并解释循环优化结果的合理性。

```
f(a,b,c,d,x,y,z)                    |       f:
int a,b,c,d,x,y,z;                  |           pushl   %ebp
{                                   |           movl    %esp, %ebp
    while(a<b) {                    |           movl    8(%ebp), %edx
        if (c<d)                    |           movl    12(%ebp), %eax
            x = x+z;                |           cmpl    %eax, %edx
        else                        |           jge  .L9
            x = y-z;                |       .L7:
    }                               |           jl    .L7
}                                   |       .L9:
                                    |           popl%ebp
                                    |           ret
```

版权申明