# 编译原理 - 作业(4)：语义分析、中间代码

**截止时间:** 2024/6/20（周四）课前，14:19:59
**提交方式:** https://easyhpc.net/course/164

**1.** 对于如下文法 G:

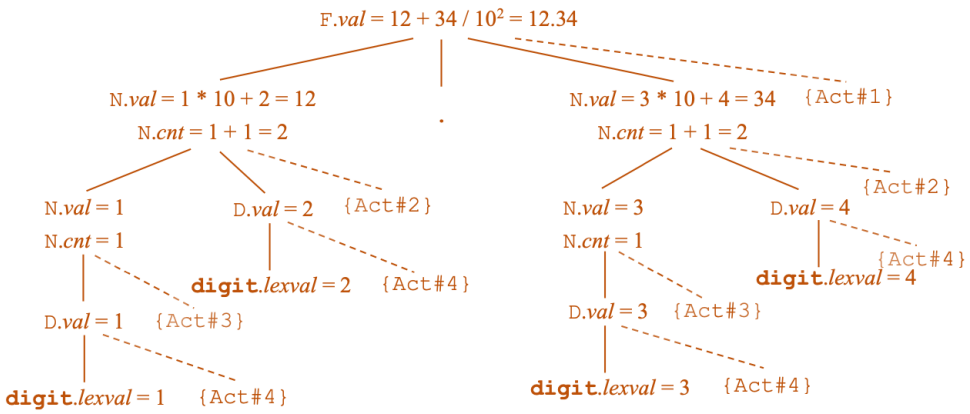
F → N . N
N → N D | D
D → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

（1） 请用自然语言描述上述文法所定义的语言；
该文法定义了表示实数的串，一个实数必须由整数和小数两部分组成，整数部分允许以 0 开头。

（2） 给出该文法的一个翻译模式（Translation Scheme），其语义为计算一个十进制输入串的实数值（例如，对于输入串 **1 2 3 . 4 5 6**，其实数值的计算结果为 123.456）；
正确答案有多种。其中一种较为简单、直观的翻译模式如下:

$F \rightarrow N_1 \ . \ N_2 \quad \{ \ F.val = N_1.val + N_2.val \ / \ 10^{N_2.cnt} \ \}$
$N \rightarrow N_1 \ D \quad \{ \ N.val = N_1.val * 10 + D.val;$
$\quad N.cnt = N_1.cnt + 1 \ \}$
$\quad | \ D \quad \{ \ N.val = D.val;$
$\quad N.cnt = 1 \ \}$
$D \rightarrow \mathbf{digit} \quad \{ \ D.val = \mathbf{digit}.lexval \ \}$

（3） 根据（2）中的翻译模式，画出输入串 1 2 . 3 4 的带注释的、带动作的分析树（Annotated Parse Tree with Actions）。
带注释、带动作的分析树如下(其中，Act#n 表示上述翻译模式中的第 n 个语义动作):

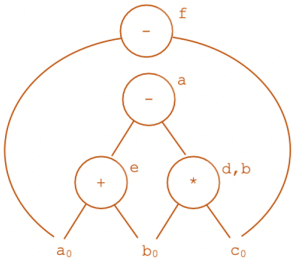



**2.** 给定如下代码的基本块（Basic Block）：

```
d = b * c
e = a + b
f = a -c
b = b * c
a = e - d
```

（1） 构造该基本块的有向无环图(Directed Acyclic Graph，简称 DAG)；

所构造的 DAG 如下：




（2） 分别有如下假设：

- 假设#1：仅变量 a 在基本块的出口(exit)是活跃的(live)；
- 假设#2：变量 f 和 a 在基本块的出口均是活跃的。

试分上述 2 种不同的假设情况，分别基于你构造出来的 DAG 对基本块进行优化。

假设#1 的情况，优化结果为

```
d = b * c
e = a + b
a = e - d
```

假设#2 的情况，优化结果为

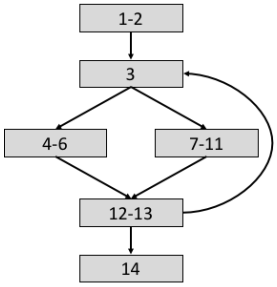
```
d = b * c
e = a + b
f = a - c
a = e - b
```

**3.** 给定如下中间代码片段:

```
1:    x = 0
2:    y = 0
3:  L0: if n / 2 goto L1
4:    x = x + n
5:    y = y + 1
6:    goto L2
7:  L1: y = y + n
8:    c = 4 / 2
9:    t1 = x * c
10:   t2 = c - 1
11:   x = x + t2
12: L2: n = n -1
13:   if n > 0 goto L0
14:   return x
```

（1） 为上述代码片断划分基本块（basic block），并画出该代码片断的控制流图 (control flow graph，简称 CFG)。你可以直接画出 CFG，在 CFG 的每一结点中用 n-m 表示该基本块由第 n 至 m 条指令组成;

【参考答案】




（2） 对第 7-11 条指令片段，列出两种代码优化方法;

【参考答案】

t1 = x * c是dead code，可以删除

x = x + t2中t2可以通过constant folding和propagation得到，可以替换为x = x + 1

（3） 假定所给代码片段（1-14 行）来自于函数 int Func(int n)，其中 n 是参数,x、y 是局部变量。那么在最终生成的目标代码中，Func 被调用时如何访问到 n、x 和 y? 提示：函数调用的栈空间由$sp（stack pointer，栈指针）和$fp（frame pointer，帧指针）维护。

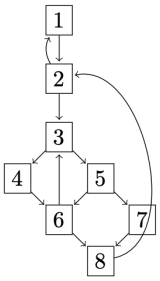
【参考答案】

n: $fp + 8

x: $fp – 4

y: $fp - 8

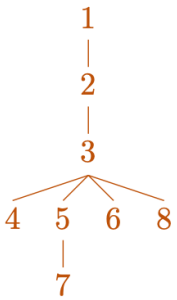
**4.** 给定如下控制流图(control flow graph)：[注：需自行学习，详见龙书 9.6 Loops in Flow Graphs 或链接]:




（1） 节点 8 的直接支配（immediate dominator）？

Node 3

（2） 画出该 CFG 的支配树（dominator tree）；



（3） 列举出该 CFG 中的所有自然循环（natural loops），给出循环的头节点和其他节点。

Loop 2 → 1: header node 1, other nodes 2, 3, 4, 5, 6, 7, 8

Loop 6 → 3: header node 3, other nodes 4, 5, 6

Loop 8 → 2: header node 2, other nodes 3, 4, 5, 6, 7, 8