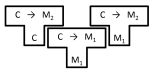
演習問題解答

第1章　序　　論

1  より，  コンパイラが得られる．このコンパイラを用いて目的とする  コンパイラを生成すればよい．

2 省略

3 省略

4 省略

5 $\{v_2, v_3, v_6\}$

第2章　コンパイラの概要

1 省略

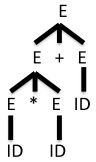
2 省略

3 $\{\mathtt{a}^n\mathtt{b}^n \mid n \geq 1\}$

4 (1)　$(\{S\}, \{\mathtt{a}, \mathtt{b}\}, \{S \to \mathtt{a}S\mathtt{b}, S \to \mathtt{a}S, S \to \mathtt{ab}\}, S)$

(2)　$(\{S\}, \{0, 1\}, \{S \to 0S1S, S \to 1S0S, S \to \varepsilon\}, S)$

5 右図



6

```
Load id1
Add id2
Add id3
Sub id4
Store t3
```

第3章 字 句 解 析

1 省略

2 省略

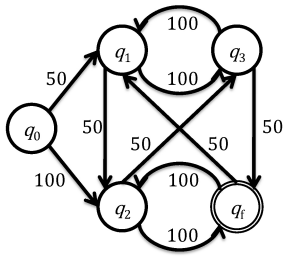
3 省略

4 (a) 正規集合である

(b) 正規集合である

5 省略

6 右図



7 省略

8 省略

9 省略

第4章 構 文 解 析

1 (a) `ab*c+`

(b) `ab*cde+*-`

2 $(\{A, B\}, \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}, \{A \to \mathtt{c}B|\mathtt{d}B|\mathtt{c}|\mathtt{d}, B \to \mathtt{a}B|\mathtt{b}B|\mathtt{a}|\mathtt{b}\}, A)$

3 省略

4 $\mathrm{First}_1(S') = \mathrm{First}_1(\text{論理式}) = \mathrm{First}_1(\text{AND 式}) = \{\mathtt{EQT}\}$

$\mathrm{Follow}_1(S') = \{\#(\text{端記号})\}$

$\mathrm{Follow}_1(\text{論理式}) = \{\#, \mathtt{OR}\}$

$\mathrm{Follow}_1(\text{AND 式}) = \{\#, \mathtt{OR}, \mathtt{AND}\}$

5 省略

6 省略

第 5 章　中間コード生成

1 (a) 97965 98752 98752 3 979865 97964 1

(b) (i)

$S \rightarrow$ `do`
{`newLabel`(&S.`label`);
`generate`(S.`label` :); }
$S^{(1)}$`while(`E`);`
{`generate`(`if`, E.`var`, `goto`, S.`label`); }

(ii)　省略

(iii)

$S \rightarrow$ `if(`E`)`
{`newLabel`(&S.`label1`);
`generate`(`if not`, E.`var`, `goto`, S.`label1`); }
$S^{(1)}$
{`newLabel`(&S.`label2`);
`generate`(`goto`, S.`label2`);
`generate`(S.`label1` :); }
$S^{(2)}$
{`generate`(S.`label2` :); }

(iv)〜(ix)　省略

(c)

l_1 :

$a_1 = id_1$
$a_2 = a_1$
$a_3 = id_2$
$a_4 = a_2 < a_3$
$a_5 = a_4$
`if not` a_5 `goto` l_2
$a_6 = id_1$
$a_7 = a_6 - id_3$
$a_8 = a_7$
$a_9 = a_8$
$id_1 = a_9$
`goto` l_3

l_2 :

$a_{10} = id_2$
$a_{11} = a_{10} - id_4$
$a_{12} = a_{11}$
$a_{13} = a_{12}$
$id_2 = a_{13}$

l_3 :

$a_{14} = id_5$
$a_{15} = a_{14}$
$a_{16} = id_1$
$a_{17} = a_{16} - id_6$
$a_{18} = a_{15} < a_{17}$
$a_{19} = a_{18}$
$a_{20} = id_7$
$a_{21} = a_{20}$
$a_{22} = id_1$
$a_{23} = a_{21} < a_{22}$
$a_{24} = a_{19}$ `and` a_{23}
`if` a_{24} `goto` l_1

2

式 → 式 (1)?
{newTemp(& 式 .var);
newLabel(& 式 .label1);
generate(if not, 式 (1).var, goto, 式 .label1); }
式 (2) :
{newLabel(& 式 .label2);
generate(式 .var, =, 式 (2).var);
generate(goto, 式 .label2);
generate(式 .label1 :); }
式 (3)
{generate(式 .var, =, 式 (3).var);
generate(式 .label2 :); }

記号表

表 Ans.1 2 行目の終りまで走査した時点の記号表

	名前	種類	値	レベル	番地	型	引数個数	引数ポインタ
1	x	var		0	1	int		
2	f	func	$a_{\mathtt{f}}$	0		int	1	3
3	y	param		1	3	int		nil

表 Ans.2 6 行目の終りまで走査した時点の記号表

	名前	種類	値	レベル	番地	型	引数個数	引数ポインタ
1	x	var		0	1	int		
2	f	func	$a_{\mathtt{f}}$	0		int	1	3
3	y	param		1	3	int		nil
	main	func	$a_{\mathtt{m}}$	0		int	0	nil
4	x	var		1	3	int		
5	y	var		1	4	int		
6	x	var		2	5	int		

SC コード

```
        IBS 1
af :    ENT 3
        LDC 3
        STG 1
        LGD 1
        LLD 3
        BOP +
        STL 0
        RET 0
am :    ENT 4
        LDC 6
        STL 3
        ISP 1
        LDC 7
        STL 5
        LDC af
        MST
        LLD 5
        CUP 1
a0 :    STL 4
        DSP 1
        LLD 3
        LLD 4
        BOP +
        OUT
        HLT
```

4 省略

5 SC コード命令

```
INP             sp++; scanf("%d", &stack[sp]);
```

再帰降下パーザ

```
void F(void) {
     if(nextsym.type!=SCANF) error( );
     else{get(&nextsym);
          if(nextsym.type!='(') error( );
          else{get(&nextsym);
               if(nextsym.type!=INPUT) error( );
               else{get(&nextsym);
                    if(nextsym.type!=',') error( );
                    else{get(&nextsym);
                         E( );
                         generate(INP);
                         generate(STO);
                         if(nextsym.type==')') get(&nextsym);
                         else error( );
                        }
                    }
               }
           }
}
```

6 省略

第6章 コード最適化

1 フローグラフ

ただし,

S_1 : l_1 : $a_5 = id_1 < id_2$
　　if not a_5 goto l_2
S_2 : 　　$id_1 = id_1 - id_3$
　　goto l_3
S_3 : l_2 : $id_2 = id_2 - id_4$
S_4 : l_3 : $a_{17} = id_1 - id_6$
　　$a_{18} = id_5 < a_{17}$
　　$a_{23} = id_7 < id_1$
　　$a_{24} = a_{18}$ and a_{24}
　　if a_{24} goto l_1

2 (a) 関数 sort は末端再帰の除去により，つぎのように変更できる．

```
void sort(int i, int n) {
     int j;
     while(i<n) {
         for(j=i+1; j<n; j++)
             if(a[i]>a[j]) {int t=a[i]; a[i]=a[j]; a[j]=t; }
         i=i+1; }
```

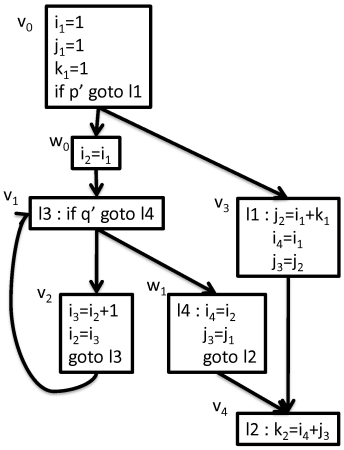
この最適化により，sort は再帰関数ではなくなったので，インライン展開が可能となる．また変数 k は 1000 で初期化された後，値が不変であるので定数伝播による最適化が可能である．これらの最適化によって得られたプログラムをつぎに示す．

```
int a[1000];

int main(void) {int i;
    for(i=0; i<1000; i++) scanf("%d", &a[i]);
    {int i=0; int n=1000;

        {int j;
         while(i<n) {
             for(j=i+1; j<n; j++)
                if(a[i]>a[j]) {int t=a[i]; a[i]=a[j]; a[j]=t; }
             i=i+1; }
         }
    }
     for(i=0; i<1000; i++) printf("%d\n", a[i]);
}
```

さらに，n の値は不変であるので，定数伝播による最適化が可能である．また，while 文中の式 i+1 を共通部分式と認識して，最適化することもできる．

(b) 省略

3 右図



4 省略

第 7 章　目的コード生成

1 (a) , (b)　省略

(c)

$$
\begin{array}{ll}
\texttt{Load} & r_1, B \\
/ & r_1, C \\
\texttt{Load} & r_2, A \\
- & r_2, r_1 \\
\texttt{Load} & r_1, E \\
- & r_1, F \\
\texttt{Load} & r_3, D \\
/ & r_3, r_1 \\
- & r_2, r_3
\end{array}
$$

2 レジスタ割当て

機械コード

$$
\begin{aligned}
+ \quad & r_1, r_2 \\
* \quad & r_2, r_1 \\
*, \quad & r_1, r_2 \\
+, \quad & r_2, r_1 \\
-, \quad & r_3, 1
\end{aligned}
$$

3 レジスタ干渉グラフとレジスタ割当て

a と b の値はそれぞれレジスタ r_1 と r_2 に格納されていると仮定すると，機械コードはつぎのように与えられる．

$$
\begin{aligned}
* \quad & r_1, r_2 \\
+ \quad & r_2, r_1 \\
* \quad & r_1, r_2
\end{aligned}
$$