コンパイラ資料（制御構造）

概要

- 論理演算
- 制御構造

制御文(control)

制御文の文法

問題4
http://cis.k.hosei.ac.jp/~asasaki/lect/compiler/2009A/problem/problem4.htm

```
<STATEMENT>::= <SUBSTITUTION> '=' <EXPRESSION> ';'
          |  '{' <STATEMENTLIST> '}'
          |  <IFPREFIX> <STATEMENT> <IFPOSTFIX>
          |  'do' <STATEMENT> 'while' '(' <LOGICALEXPRESSION>  ')'
```

論理式（本日の課題では、比較演算のみ）

```
<LOGICALEXPRESSION>::= <LOGICALTERM>
                     |  <LOGICALEXPRESSION> '||' <LOGICALTERM>
                     |  <LOGICALEXPRESSION> '^^' <LOGICALTERM>
<LOGICALTERM>::= <LOGICALUNARY>
               |  <LOGICALTERM> '&&' <LOGICALUNARY>
<LOGICALUNARY>::= <LOGICALFACTOR>
                |  '!' <LOGICALUNARY>
<LOGICALFACTOR>::= <EXPRESSION> '==' <EXPRESSION>
                 |  <EXPRESSION> '!=' <EXPRESSION>
                 |  <EXPRESSION> '>=' <EXPRESSION>
                 |  <EXPRESSION> '>' <EXPRESSION>
                 |  <EXPRESSION> '<=' <EXPRESSION>
                 |  <EXPRESSION> '<' <EXPRESSION>
                 |  '[' <LOGICALEXPRESSION> ']'
```

演算命令(再掲)

SB 0 0　　引き算命令:
　　sp-- ; S[sp] ←S[sp]-S[sp+1]; pc++
NEG 0 0　　符号反転命令:
　　S[sp] ← -S[sp]; pc++;
LE　 0 0　 関係演算命令<=
　　sp-- ; if (S[sp] <= S[sp+1) then S[sp]←1
　　else S[sp] ← 0; pc++

算術演算:
AD, SB,ML,DV, NEG … +, -, *, /, 反転
関係演算:
EQ, NE, LT,LE,GT,GE … ==, !=, <, <=, >, >=

制御命令

```
J  0  N     ジャンプ命令:
                pc ←N;
FJ  0 N     条件ジャンプ命令:
                if (S[sp] == 0)
                  pc ← N;
                else
                   pc++;
                sp--;
```

```
J 0 20
```

20番地に飛ぶ。
(次の命令は
P[20])

```
LDC 0 1
LDC 0 2
EQ 0 0
FJ 0 20
```

1==2が偽(0)であれば20番地に
飛ぶ。そうでなければ次の命令へ。
（Fall　through
(この場合1==2はa偽であるから、
必ず20番地に飛ぶ)

練習問題(1)

- FJ命令の動きに注意して、VMの動作をシミュレートしてみよ。（紙と鉛筆を使って手で確認せよ。その後hsmで確かめると良い。）
- 最初の命令が0: LDC 0 2の場合どうなるかも考えよ。

```
0: LDC 0 1
1: LDC 0 2
2: EQ 0 0
3: FJ 0 6
4: LDC 0 84
5: WRC 0 0
6: HLT 0 0
```

```
0: LDC 0 1
1: LDC 0 2
2: EQ 0 0
3: FJ 0 7
4: LDC 0 84
5: WRC 0 0
6: J 0 9
7: LDC 0 70
8: WRC 0 0
9: HLT 0 0
```

論理式の例(&&の例）

```
 0:      PUSH    0   2
 1:      LDC     0   1
 2:      STV     0   0
 3:      LDC     0   2
 4:      STV     0   1
 5:      LDV     0   0
 6:      LDV     0   1
 7:      EQ      0   0
 8:      LDV     0   0
 9:      LDV     0   1
10:      GT      0   0
11:      ML      0   0
12:      LDC     0   1
13:      EQ      0   0
14:      FJ      0   17
15:      LDC     0   84
16:      WRC     0   0
17:      LDC     0   61
18:      WRC     0   0
19:      POP     0   2
20:      HLT     0   0
```

i==jのコード

i>jのコード

&&のコード(12,13は省略可能)

```
int main(){
  int i,j;
  i=1;
  j=2;
  if (i==j && i>j )
    putchar('T');
  putchar('=');
}
```

論理式

x&&y演算の真偽値表

AND	True	False
True	*True*	*False*
False	*False*	*False*

ANDという命令は、hsmにはないので他の演算命令、関係演算命令を組み合わせて、&&演算を実現することを考える。

1. 乗算を施す(ML 0 0)

×	1	0
1	*1*	*0*
0	*0*	*0*

→x,yの両者が真（1）の場合のみ1となる。

2. 1の結果が1と等しいかどうかを判定する。
(LDC 0 1 ; EQ 0 0)

	1	0
1	*True*	*False*
0	*False*	*False*

3. x, yは0または1であるから、上記2は省略可能。

論理式

xIIy演算の真偽値表

OR	True	False
True	*True*	*True*
False	*True*	*False*

1. 和算を施す(AD 0 0)

×	1	0
1	*2*	*1*
0	*1*	*0*

→x,yの両者が偽(0)の場合のみ0となる。

2. 1の結果が0と等しくなければTrueとする。
(LDC 0 0 ; NEQ 0 0)

	1	0
1	*True*	*True*
0	*True*	*False*

(x, yは0または1であるが、x=y=1の場合
x+y=2であるが、x ||y = 1とする必要があるので、2は省略はできない。)

制御文の例(if)

```
int main(){
  int i,j;
  i=1;
  j=2;
  if (i==j)
    putchar('T');
  putchar('=');
}
```

制御文の例(if)

```
 0:    PUSH   0  2
 1:    LDC    0  1
 2:    STV    0  0
 3:    LDC    0  2
 4:    STV    0  1
 5:    LDV    0  0
 6:    LDV    0  1
 7:    EQ     0  0
 8:    FJ     0  11
 9:    LDC    0  84
10:    WRC    0  0
11:    LDC    0  61
12:    WRC    0  0
13:    POP    0  2
14:    HLT    0  0
```

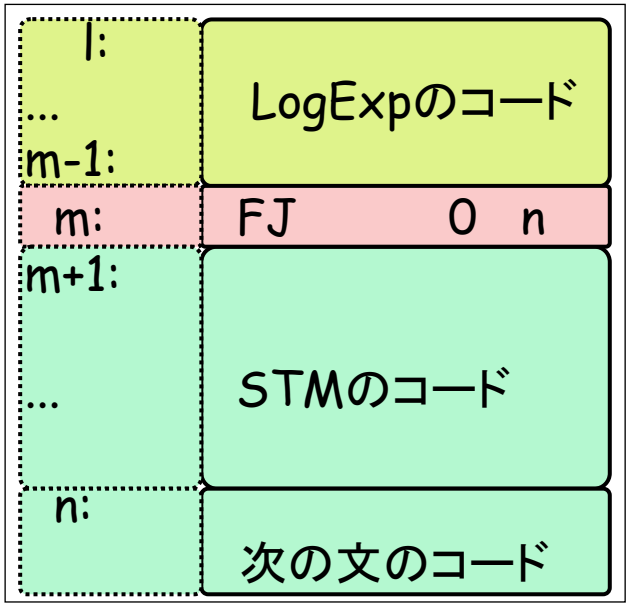

```
int main(){
  int i,j;
  i=1;
  j=2;
  if (i==j)
    putchar('T');
  putchar('=');
}
```

If文(elseなし)

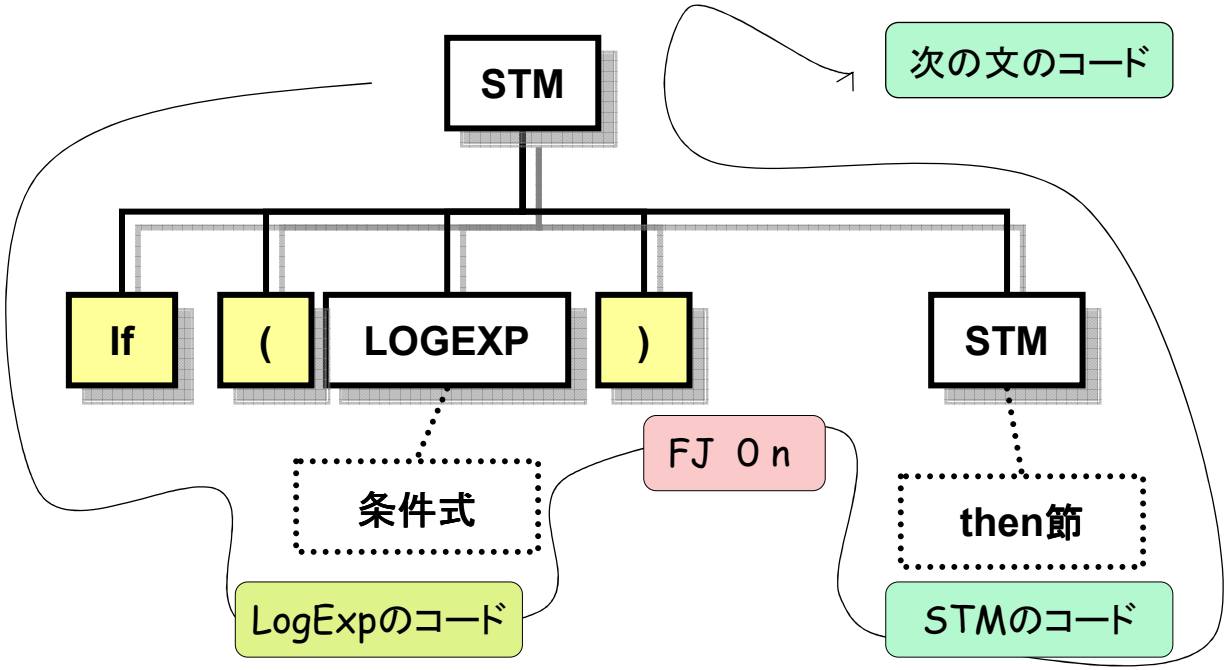
```
 0:    PUSH   0  2
 1:    LDC    0  1
 2:    STV    0  0
 3:    LDC    0  2
 4:    STV    0  1
 5:    LDV    0  0
 6:    LDV    0  1
 7:    EQ     0  0
 8:    FJ     0  11
 9:    LDC    0  84
10:    WRC    0  0
11:    LDC    0  61
12:    WRC    0  0
13:    POP    0  2
14:    HLT    0  0
```

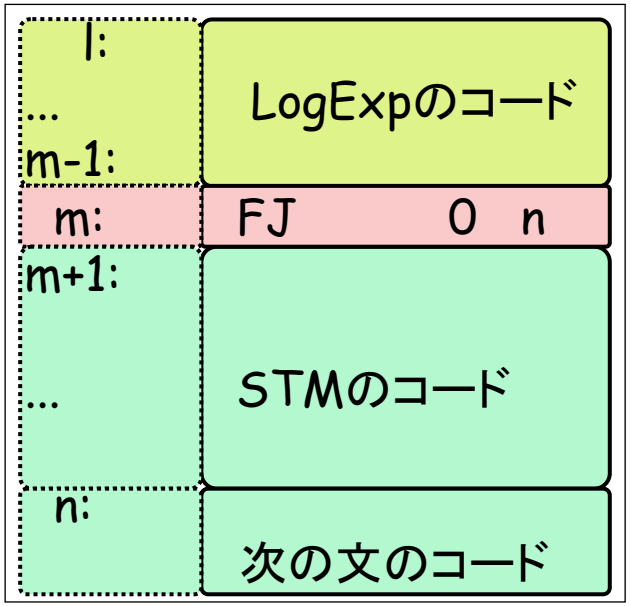
条件式

then節



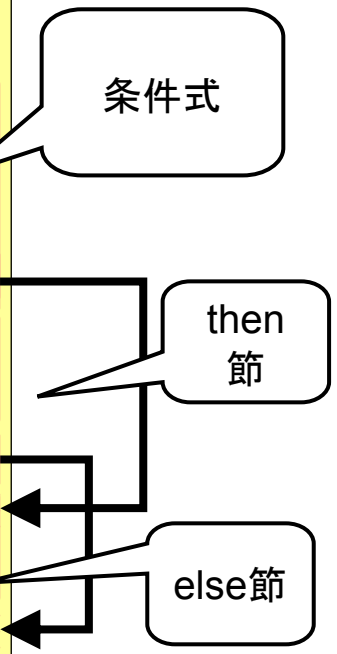
If文


STM
次の文のコード
If
(
LOGEXP
)
STM
FJ 0 n
条件式
then節
LogExpのコード
STMのコード


l:
...
m-1:
LogExpのコード
m:
FJ 0 n
m+1:
...
STMのコード
n:
次の文のコード

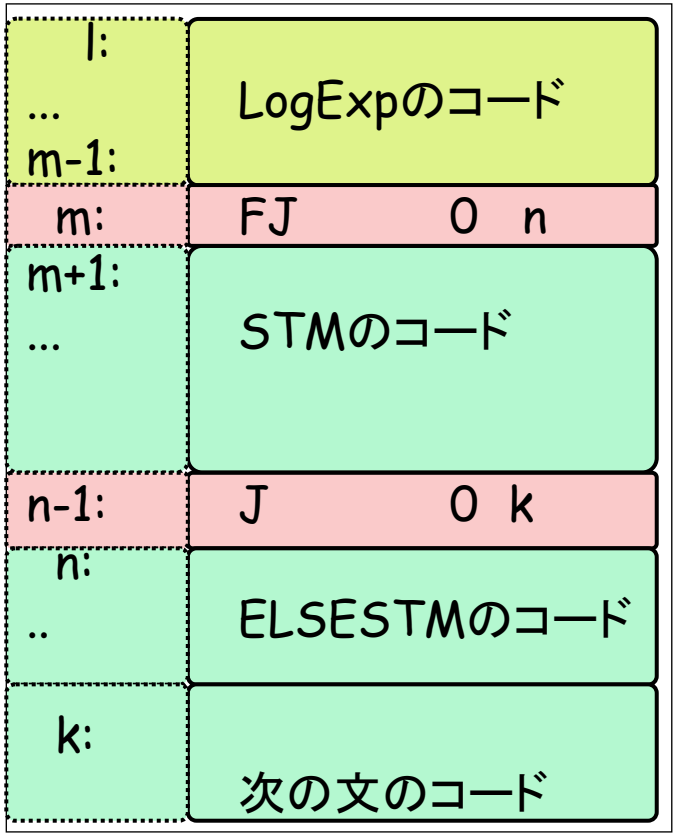
制御文の例(if else)

```
 0:   PUSH  0  2
 1:   LDC   0  1
 2:   STV   0  0
 3:   LDC   0  2
 4:   STV   0  1
 5:   LDV   0  0
 6:   LDV   0  1
 7:   EQ    0  0
 8:   FJ    0  12
 9:   LDC   0  84
10:   WRC   0  0
11:   J     0  14
12:   LDC   0  70
13:   WRC   0  0
14:   LDC   0  61
15:   WRC   0  0
16:   POP   0  2
17:   HLT   0  0
```



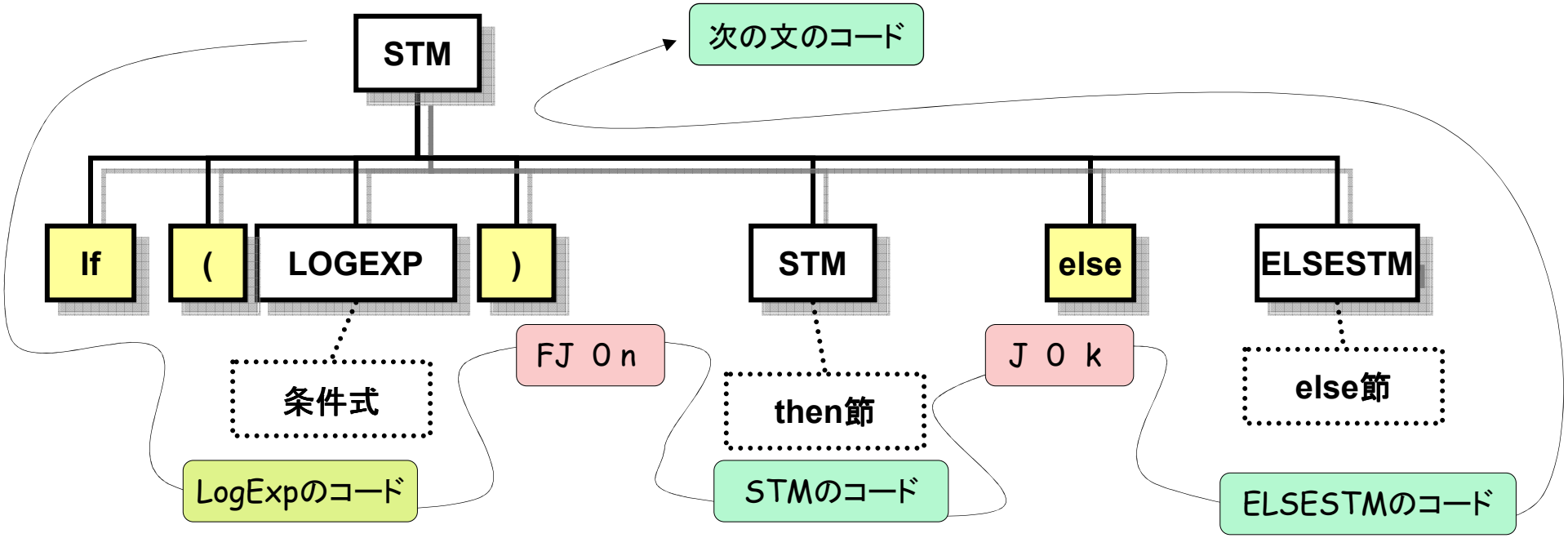

```
int main(){
  int i,j;
  i=1;
  j=2;
  if (i==j)
    putchar('T');
  else
    putchar('F');
  putchar('=');
}
```

If-else



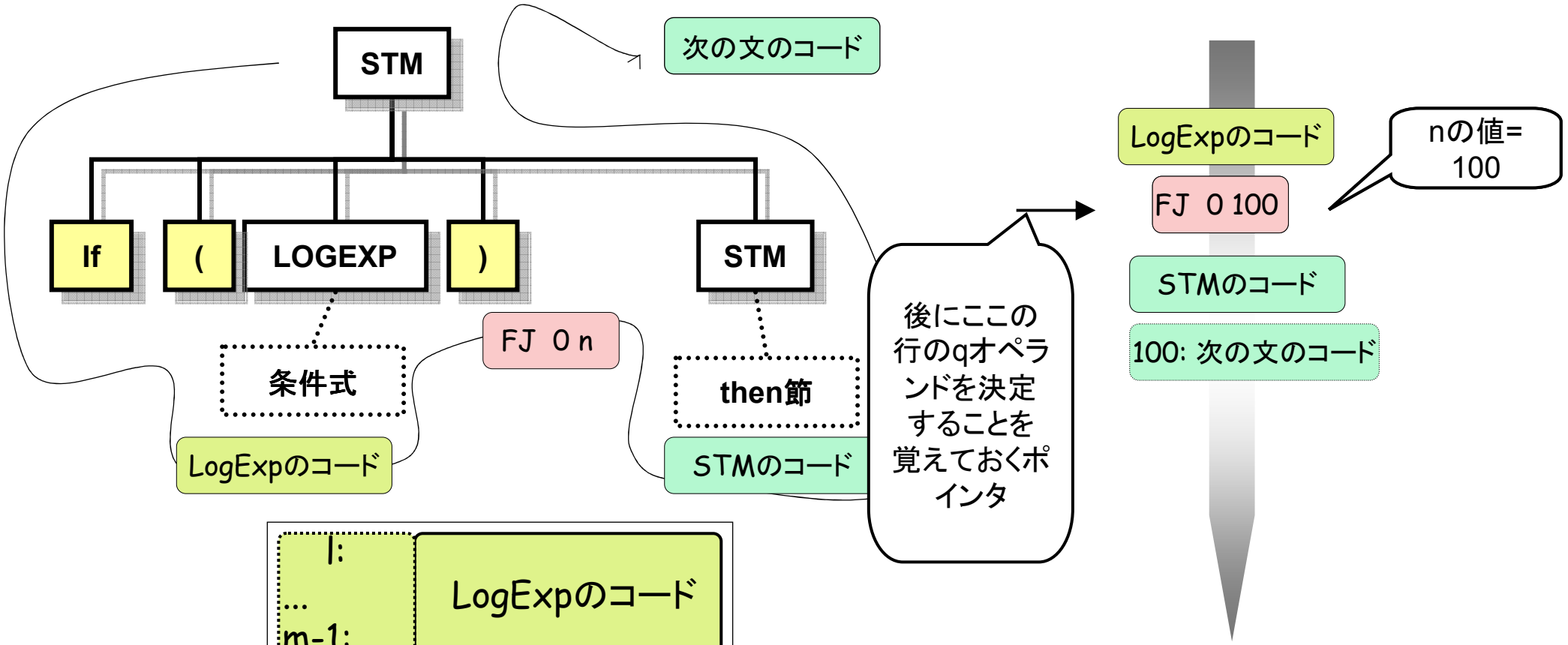

```
int main(){
  int i,j;
  i=1;
  j=2;
  if (i==j)
    putchar('T');
  else
    putchar('F');
  putchar('=');
}
```

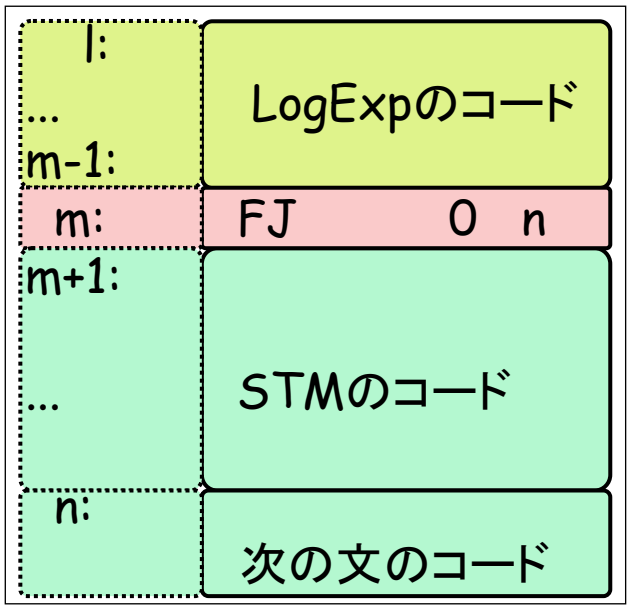
If-else



l: ... m-1:	LogExpのコード
m:	FJ 0 n
m+1: ...	STMのコード
n-1:	J 0 k
n: ..	ELSESTMのコード
k:	次の文のコード

バックパッチ(If文)

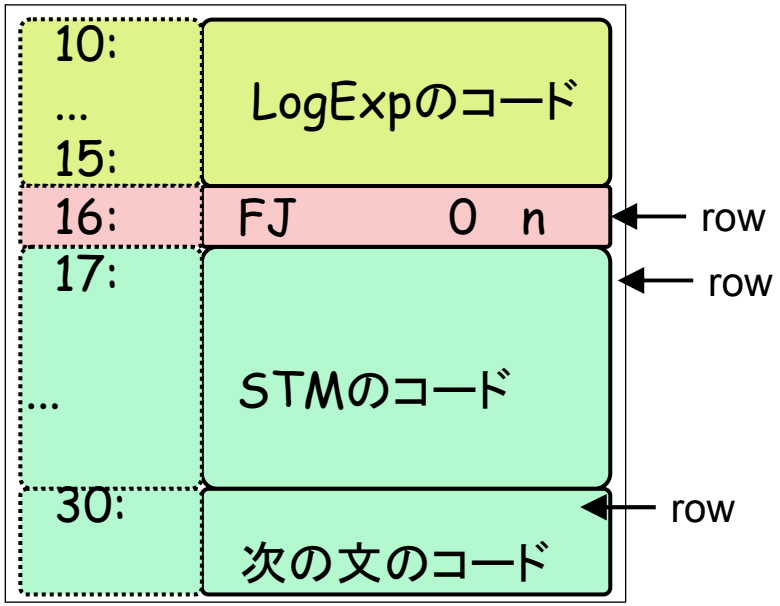

STM
次の文のコード
If
(
LOGEXP
)
STM
条件式
FJ 0 n
then節
LogExpのコード
STMのコード
後にここの行のqオペランドを決定することを覚えておくポインタ
LogExpのコード
FJ 0 100
nの値=100
STMのコード
100: 次の文のコード


l:
...
m-1:
LogExpのコード
m:
FJ 0 n
m+1:
...
STMのコード
n:
次の文のコード

JavaCCでの実装法

```
<STATEMENT>::= <SUBSTITUTION> '=' <EXPRESSION> ';'
          | '{' <STATEMENTLIST> '}'
          | row1=<IFPREFIX>() <STATEMENT> <IFPOSTFIX>(row1)
                16                                          16
（一部省略）

int <IFPREFIX>()::= 'if' '(' <LOGICALEXPRESSION> ')'
        {CodeTableのrow行目にInst(FJ, 0, #)を登録;
         return row++;}  16
int <IFPOSTFIX>(int row1){
    'else'
     {CodeTableのrow行目をInst(J, 0, #)とする;
              row2=row;row++;
              CodeTableのrow1行目のqオペランドをrowに;}
     <STATEMENT>
      {CodeTableのrow2番目のqオペランドを=rowに;}  // ここまでが
else節がある場合の処理。
              | /* empty */ {CodeTableのrow1行目のqオペランドをrow
に;}                                    16                  30
```

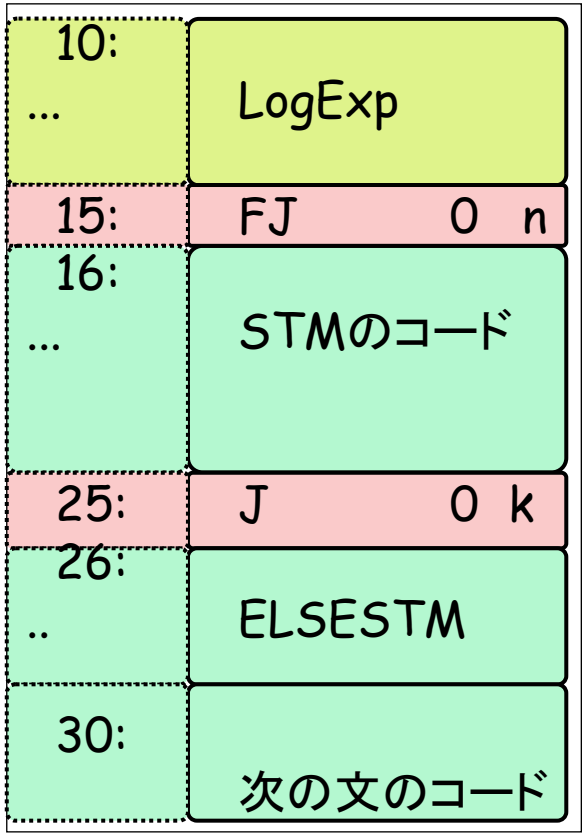


練習問題

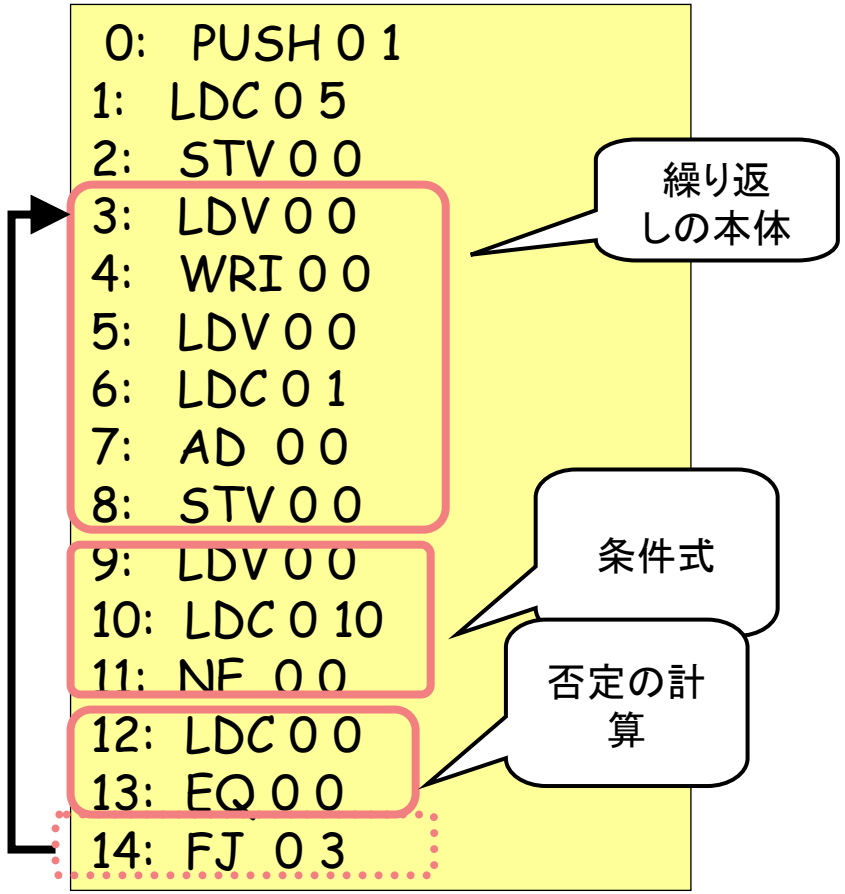
```
<STATEMENT>::= <SUBSTITUTION> '=' <EXPRESSION> ';'
          | '{' <STATEMENTLIST> '}'
          | row1=<IFPREFIX>() <STATEMENT> <IFPOSTFIX>(row1)

（一部省略）

int <IFPREFIX>()::= 'if' '(' <LOGICALEXPRESSION> ')'
         {CodeTableのrow行目にInst(FJ, 0, #)を登録;
          return row++;}
int <IFPOSTFIX>(int row1){
    'else'
      {CodeTableのrow行目をInst(J, 0, #)とする;
               row2=row;row++;
               CodeTableのrow1行目のqオペランドをrowに;}
     <STATEMENT>
       {CodeTableのrow2番目のqオペランドを=rowに;}  // ここまでが
else節がある場合の処理。
               | /* empty */ {CodeTableのrow1行目のqオペランドをrow
に;}
```

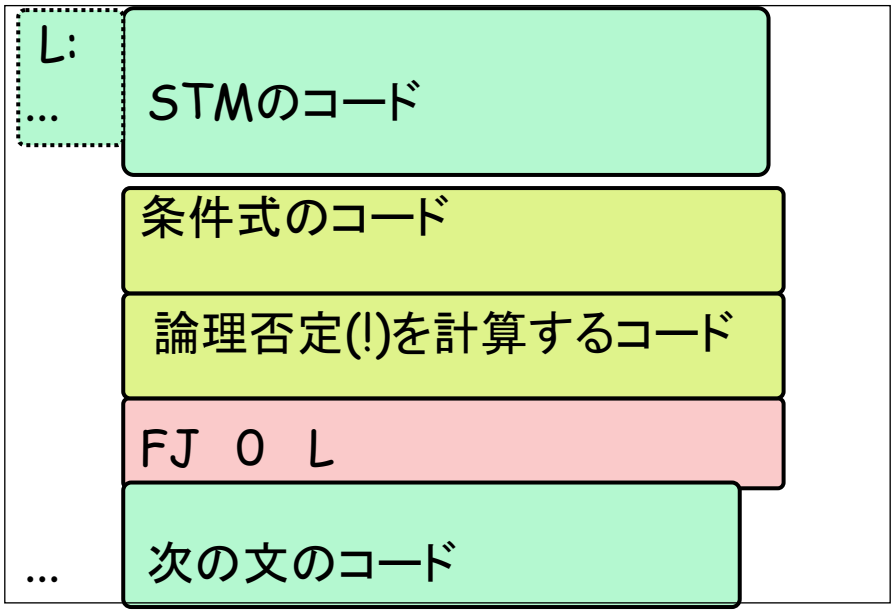


制御文の例(do-while)



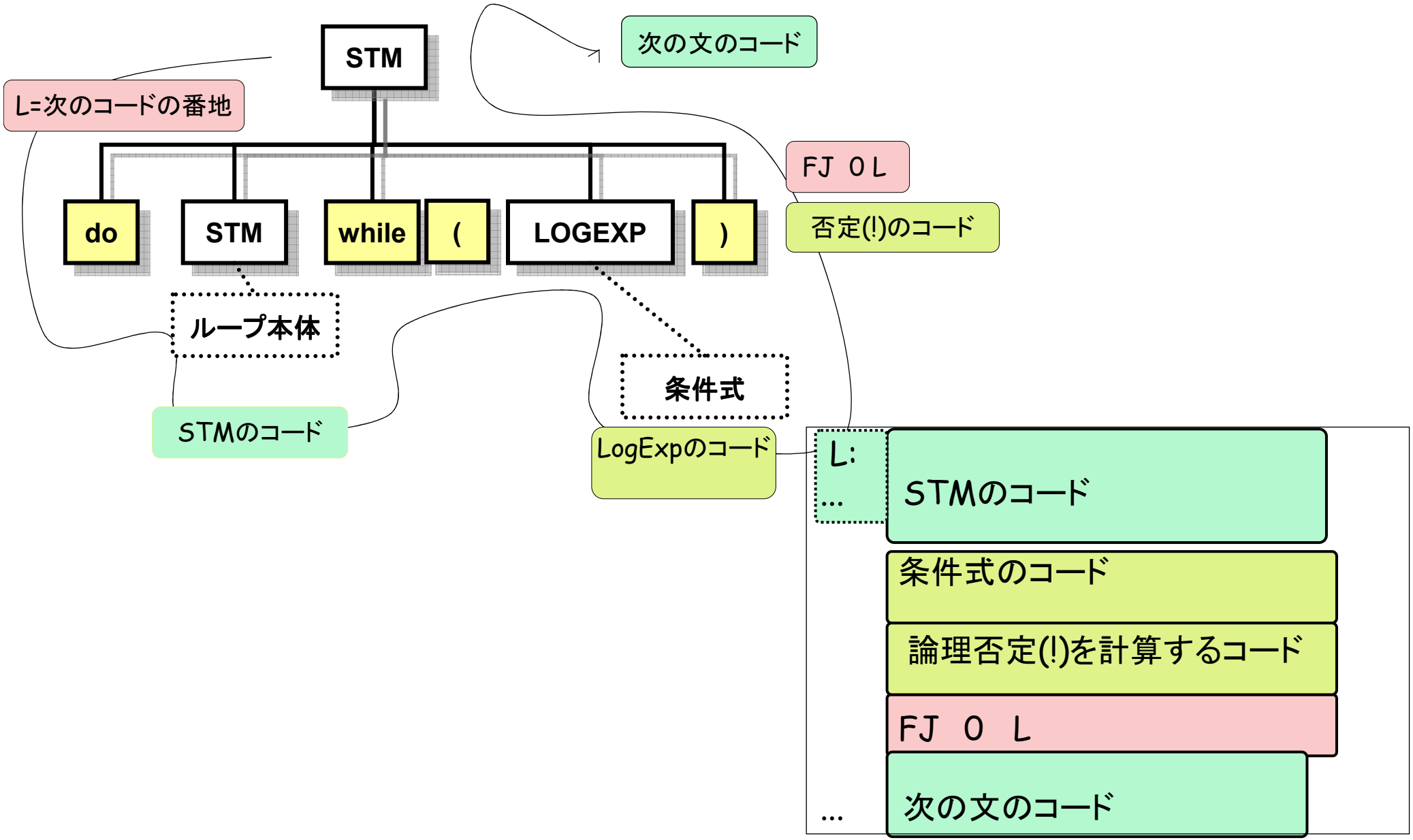

```
int main(){
  int i;
  i = 5;
  do
  {
    putint(i);
    i = i + 1;
  }
  while (i != 10);
}
```

do-while文




```
int main(){
  int i;
  i = 5;
  do
  {
    putint(i);
    i = i + 1;
  }
  while (i != 10);
}
```

do-while文

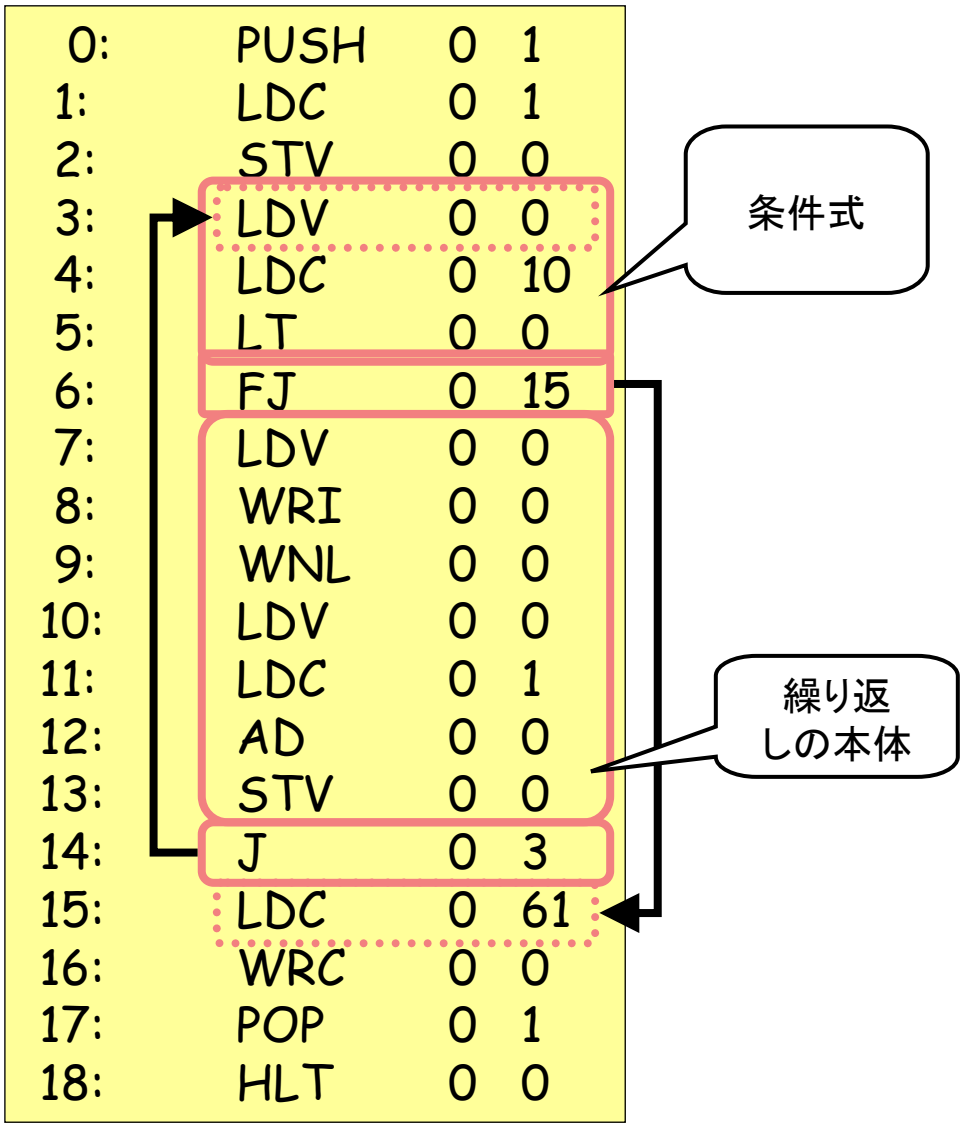

STM
次の文のコード
L=次のコードの番地
FJ 0 L
do
STM
while
(
LOGEXP
)
否定(!)のコード
ループ本体
条件式
STMのコード
LogExpのコード
L:
...
STMのコード
条件式のコード
論理否定(!)を計算するコード
FJ 0 L
次の文のコード
...

練習問題(do-while)

```
int main(){
  int i;
  i=1;
  while(i <10){
    putint(i);
    putnl;
    i=i+1;
  }
  putchar('=');
}
```

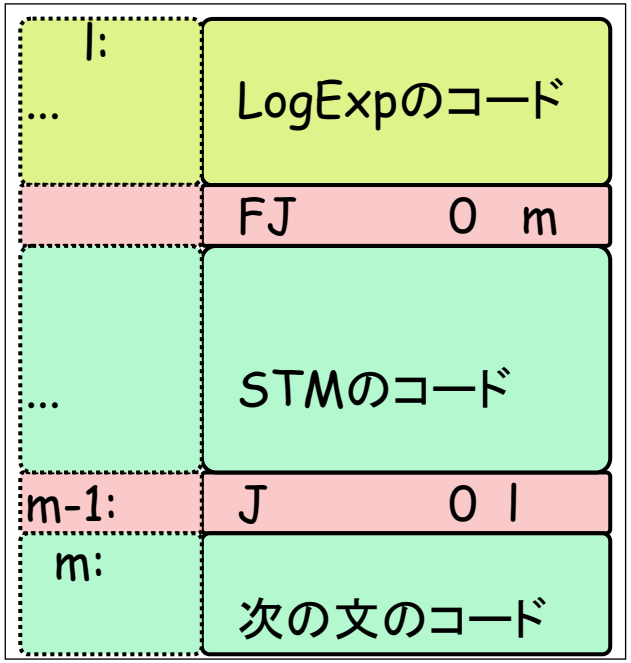
```
int main(){
  int i;
  i=1;
  do{
    putint(i);
    putnl;
    i=i+1;
  } while(i <10)
  putchar('=');
}
```

制御文の例(while)



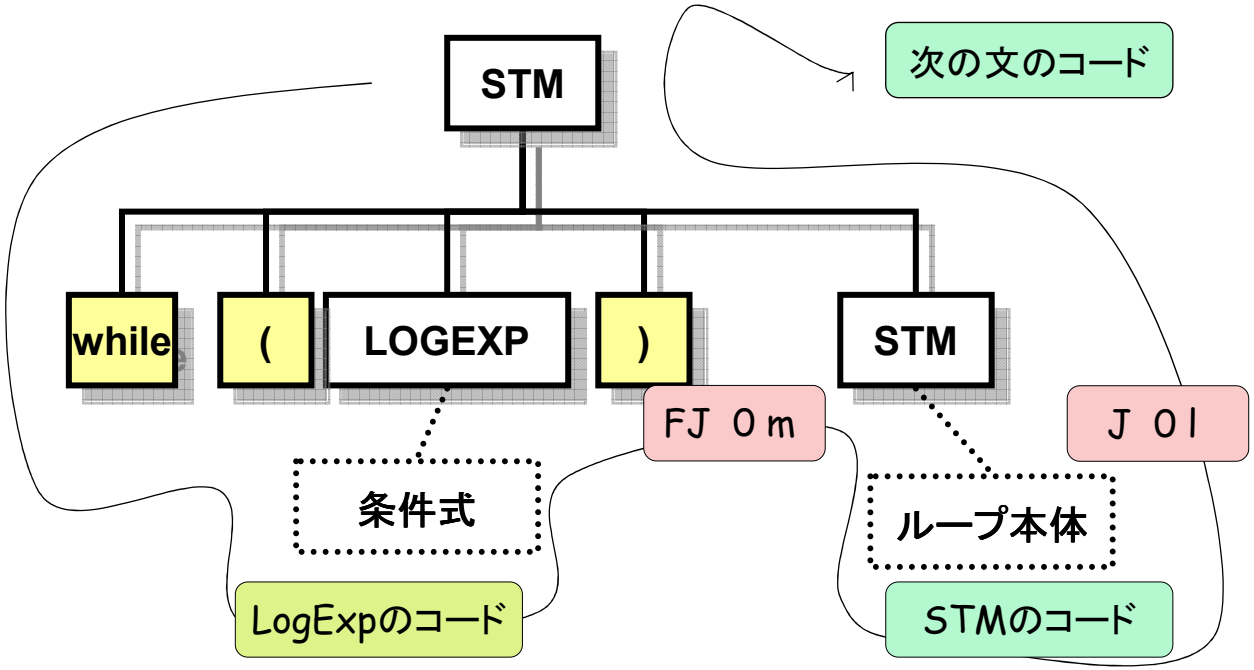

```
int main(){
  int i;
  i=1;
  while(i <10){
    putint(i);
    putnl;
    i=i+1;
  }
  putchar('=');
}
```

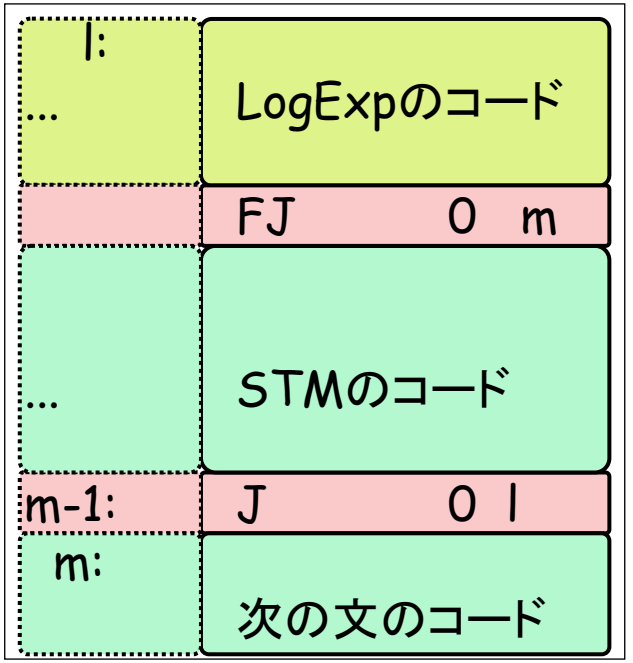
while文




```
int main(){
  int i;
  i=1;
  while(i <10){
    putint(i);
    putnl;
    i=i+1;
  }
  putchar('=');
}
```

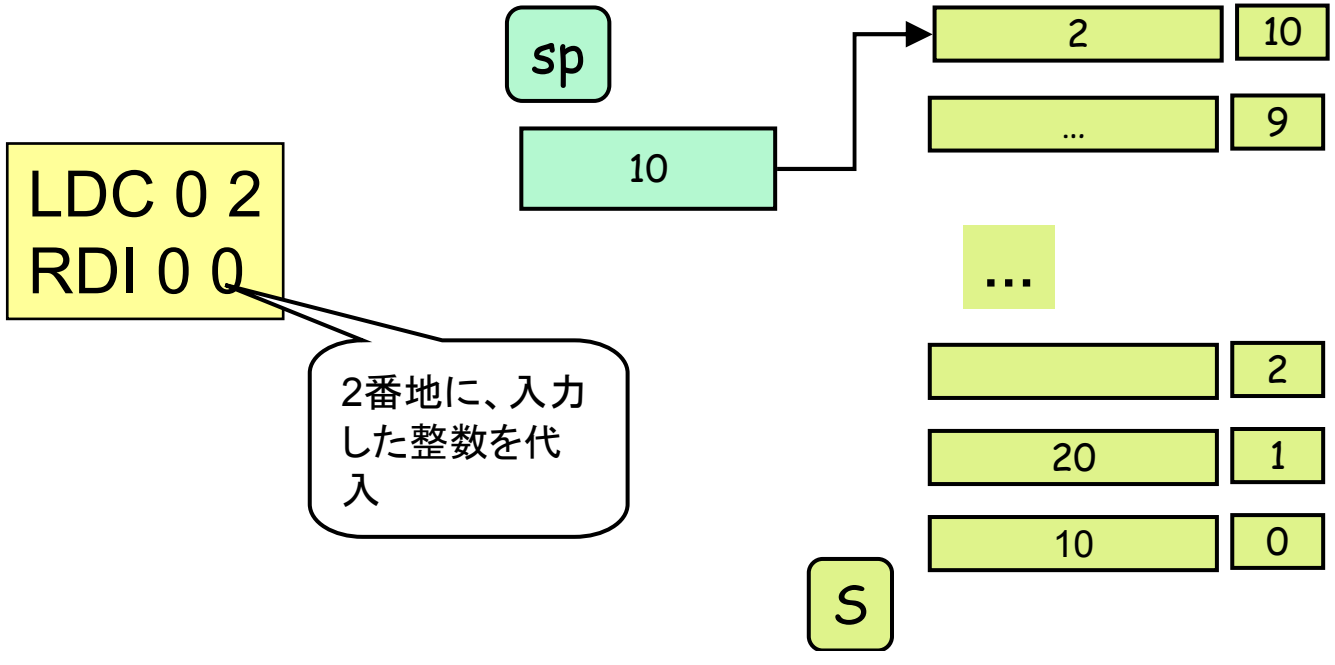
while文


STM
次の文のコード
while
(
LOGEXP
)
STM
FJ 0 m
J 0 l
条件式
ループ本体
LogExpのコード
STMのコード


l:
...
LogExpのコード
FJ 0 m
...
STMのコード
m-1:
J 0 l
m:
次の文のコード

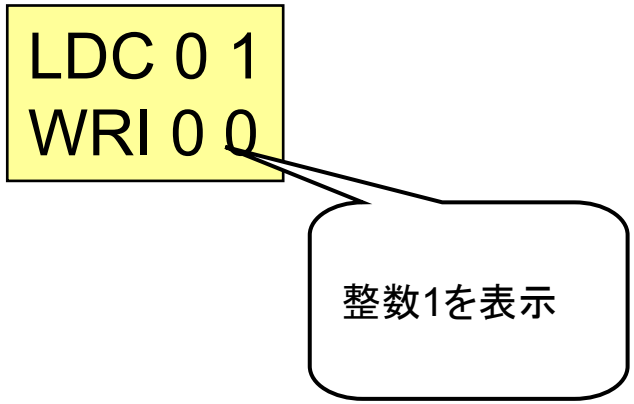
入力命令

RDI 0 0　　整数入力 (Read Integer)
　　整数入力→i;　S[S[t]] ← i; sp--;　pc++;
RDC 0 0　　文字入力 (Read Character)
　　文字入力→c ; S[S[t]] ← c; sp--;　pc++;
　　(cは入力された文字の文字コード)



出力命令

```
WRI  0  0     整数表示:
                  S[sp]を表示; sp--;  pc++;
WRC 0 0   文字表示:
                  文字コードS[sp]に対する文字の表示;
                  sp--; pc++;
WNL 0 0   改行表示:
                  改行; pc++;
```



課題4(Problem4)

- 作成問題4
  http://cis.k.hosei.ac.jp/~asasaki/lect/compiler/2009A/problem/problem4.htm

  提出状況

  http://cis.k.hosei.ac.jp/~asasaki/lectureCompiler/status.htm