第4章 記号表と中間語

記号表 (symbol table)

　　はソースプログラム中の識別子 (identifier) に関する情報を集めた表である。

- 記号表のデータ構造がコンパイラーの性能に大きく影響する。
    - 配列、リスト、二分木 — これらは適さない。
    - 　　(hash table) を用いることが多い。 《教 p.81 6.2.3》

      「ハッシュ」は「切り刻む」という意味で、ハッシュドビーフのハッシュと同じ語源らしい。
- 識別子はこの表のエントリーのインデックス（あるいはポインター）として表す。

中間語 (intermediate language) 《教 p.84》

　　では、ターゲット (CPU) の種類によらない最適化を行なう（中間語を用いるとターゲットの追加が容易になる）。

木（あるいはグラフ）構造、逆ポーランド記法、　　、　　などがある。 《教 p.85》 どれも本質は構文木である。

ただし簡単な 1 パスコンパイラーでは中間語を生成せず直接ターゲットを生成する。

四つ組み (quadruple)

四つ組みとは

　　

という形である。ただし　オペランドは　　。

例

ソース	四つ組み
a + b * c	

- オペランドに複雑な式を書くことができない。
- 途中計算の結果を必ず変数に代入しなければいけない。

第5章 誤り処理

要するに難しい。（＿＿＿＿＿＿＿＿, ＿＿＿＿＿＿＿＿、と言われる。）

第6章 実行時環境とレジスター割り付け

できるだけレジスター（高速なメモリー）を効率的に利用できるようにする

第7章 コード生成

7.1 変換の基本パターン

基本パターンにあてはめ、再帰的にコード生成する (教 p.108)。以下の生成コード例は Oolong のニーモニックを使用している。

① 代入文: `v = 式1;`

```
（式1を計算するコード）        ← 再帰的に
                              ; 整数型の場合
```

③ 変数の参照（式）: `v`

```
                              ; 整数型の場合
```

定数の参照（式）: `c`

```
```

⑤ 二項演算（式）: `式1 op 式2`

```
（式1を計算するコード）
（式2を計算するコード）
（op に対応する命令）     ; 整数の足し算 "+" なら
  ; "-" isub, "*" imul, "/" idiv, "%" irem
```

問 7.1.1 `x, y, z` がそれぞれ 1, 2, 3 という番号に対応するとき、 `x = y + z * 2;` のコードは、どうなるか?

⑥ if 文: `if (式1) 文1 else 文2`

```
（式1を計算するコード）
                              ; … 0（偽）ならば L1 にジャンプする
（文1のコード）

（文2のコード）
```

ここで L_1, L_2 はフレッシュな（他と異なる）名前である。機械語などで、ラベルでなくオフセット（番地の差）を出力する必要がある場合は、＿＿＿＿＿＿（後戻りして修正）する

問 7.1.2 else のない if 文に対しては、どのようなコードを生成するべきか?

⑦ while 文: `while` （式$_1$） 文$_1$

```
（式1を計算するコード）
                                   ; … 0（偽）ならば L2 にジャンプする
（文1のコード）
```

問 7.1.3 do ～ while 文: `do` 文$_1$ `while` （式$_1$） ; に対しては、どのようなコードを生成するべきか?
ヒント: Oolong の場合 `ifne` を使うと簡単になる。

⑧ 関数呼出し式: *f*（式$_1$, 式$_2$, …）

```
（式1を計算するコード）
（式2を計算するコード）
        ⋮
（f の呼出し命令） ;                    f など
```

呼出し命令は次のような＿＿＿＿＿＿(prologue, プロローグ) を行う (教 p.97)

- レジスターや局所変数を＿＿＿＿＿＿する
- ＿＿＿＿＿＿をスタックに退避する
- 関数のコードへジャンプする

⑨ return 文: `return` 式$_1$;

```
（式1を計算するコード）
                  ; 整数型の場合
```

return 命令は次のような＿＿＿＿＿＿(epilogue, エピローグ) を行う (教 p.97)

- スタックに退避していた＿＿＿＿＿＿＿＿＿＿＿＿する
- 戻り番地をスタックから取り出して、そこへジャンプする