5E-6

レジスタ割り付け最適化処理のコンパイラ構成方式

竹田栄作　松岡恭正　河内浩明　茂木強　篠崎衛
(三菱電機株式会社)

1．はじめに

コンパイラのコード生成処理では、通常レジスタ割り付けの最適化処理を行なった後でコード生成を行なう。このため、コード生成の最終段階では、オペランドのレジスタ割り付け状況をもとに、機械命令を出し分ける必要があり、その設計は、かなり複雑なものになる場合が多い。ここで述べる方式では、レジスタ割り付けの最適化処理を単純なレジスタ割り付けによるコード生成した後に行なうことにより、コンパイラのコード生成処理の負担軽減を狙うものである。レジスタ割り付けの結果として変数、テンポラリ等のスカラ値に対する演算をロード／ストアも含めて可能な限りレジスタ内で行なうことを目標とする。

2．命令リストの生成

本方式によるコンパイラの構文解析／テキスト最適化以降の構成を図１に示す。

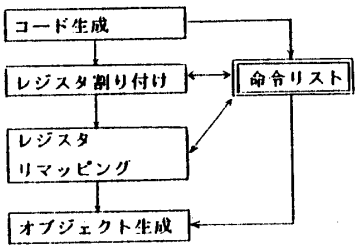

図１．コンパイラの構成

コード生成処理では、構文解析／テキスト最適化の結果をもとに、単純なレジスタ割り付け処理によって最小限のレジスタを使用して実行可能な機械命令列を生成する。生成した機械命令列は、命令リストと呼ぶ中間語として表現する。命令リストには、以下の情報が含まれる。

（１）レジスタ番号、アドレス等機械命令に関する情報。
（２）各命令が参照／定義するオペランド。
（３）基本ブロック分割及びループ構造に関する情報。

この命令リストに対し、オペランドをレジスタに割り付けることにより、命令を再構成する。

3．レジスタ割り付け

レジスタ割り付け処理では、コード生成で、使用しなかったレジスタを、変数、定数、テンポラリ等のオペランドに割り付ける。レジスタが割り付いたオペランドに対する命令リストは、メモリ参照からレジスタ参照に命令を変更する。

（例）　A＝B

```
L     Ri, A     (コード生成)
ST    Ri, B
  ⇩
LR    Ri, RA    (レジスタ割り付け)
LR    RB, Ri
```

コード生成処理では、最小限のレジスタによって命令を生成しているため、レジスタ化されないオペランドに対するコードの効率は、あまり良くない。このため、レジスタ割り付け処理では、１つのレジスタに複数のオペランドを対応させることにより、できるだけ多くのオペランドがレジスタに割り付くようにする。また、ループ内に現われるオペランドは優先的にレジスタに割り付けるよう重み付けを行なう。

レジスタ割り付け処理では、さらに、後続のレジスタ・リマッピング処理のためにプログラム内でのレジスタの使用法を調べ、各基本ブロック毎に出口でのレジスタのｌｉｖｅ情報をセットする。

4．レジスタ・リマッピング

レジスタ割り付け処理による命令リストの変形は、メモリ参照をレジスタ参照に変更するだけであり、そのままではレジスタ間の無駄なデータ転送が残る。レジスタ・リマッピング処理は、それらの無駄なデータ転送を削除するものである。レジスタ・リマッピングには、参照レジスタ・リマッピングと定義レジスタ・リマッピングがある。

Compiler Construction for Register Allocation
E.Takeda Y.Matuoka H.Kawachi T.Motegi M.Shinozaki
MITSUBISHI ELECTRIC CORPORATION

【参照レジスタ・リマッピング】

① LR　RA, RB
　ref／dref (RA)
② ref／dref (RA)
　　⇩
①′ (nop)
　ref／dref (RB)
②′ ref／dref (RB)

〔条件〕
(1) 区間内でRBの値は変更されない。
(2) 区間内にdref (RA) がある場合、RBは、LR命令の実行後deadである。
(3) ②の実行後、RAはdeadである。

【定義レジスタ・リマッピング】

① def (RB)
　ref／dref (RB)
② LR　RA, RB
　　⇩
①′ def (RA)
　ref／dref (RA)
②′ (nop)

〔条件〕
(1) 区間内にRAのref／def／drefはない。
(2) ②の実行後、RBはdeadである。

【記号】
ref (Ri) ……レジスタRiを参照する命令
def (Ri) ……レジスタRiを参照する命令
dref (ri) …レジスタRiを参照／定義する命令。

レジスタ・リマッピング処理を行なうには、個々の命令毎に命令実行後のレジスタのlive/dead 状況を知る必要がある。そのために、リマッピング処理に先立って、レジスタ割り付け処理フェイズで基本ブロック毎に求めたレジスタのlive/dead 情報をもとに、命令単位のlive/dead 情報を作成する。

5．適用例

(例1)　X＝Y
```
L     Ri, Y     (コード生成)
ST    Ri, X
   ⇩
LR    Ri, RY    (レジスタ割り付け)
LR    RX, Ri
   ⇩
(nop)           (参照レジスタ
LR    RX, RY     リマッピング)
```

(例2)　A＝B＋C
```
L     Ri, B     (コード生成)
A     Ri, C
ST    Ri, A
   ⇩
LR    Ri, RB    (レジスタ割り付け)
AR    Ri, RC
LR    RA, Ri
   ⇩
LR    RA, RB    (定義レジスタ
AR    RA, RC     リマッピング)
(nop)
```

6．結言

一旦コード生成した後で、オペランドにレジスタを割り付けることによって、コードを改良する方式を示した。本方式によれば、コード生成フェイズからレジスタ割り付けの最適化処理を除くことができ、コード生成フェイズの設計が容易になる。また、処理を機械命令に対応した中間語である命令リストで行なうため、言語依存部分が少なく、複数言語に共通のレジスタ割り付け最適化処理を設計することが可能である。

【参考文献】

1. R.G.Scarborough and H.G.Kolsky : Improved Optimization of FORTRAN Object Programs, IBM J.Res.Develop.,Vol.24 No.6 (1980)
2. 松田，河内他：広域レジスタ割り付け手法に関する一考察，情報処理学会第２８回全国大会