VLIW型計算機KIDOCH用Cコンパイラにおけるグローバル最適化

4E−2

本郷 哲[+]、安倍 正人[++]、中鉢 憲賢[+]、城戸 健一[+++]
([+]東北大学工学部、[++]東北大学大型計算機センター、[+++]千葉工業大学)

1.はじめに

我々は信号処理を主目的としたVLIW型計算機KIDOCH III[1]のための Cコンパイラ[2]を開発している。さらにこのKIDOCH III で培った経験を生かし、様々の改良を施したKIDOCH IV を開発中である[3][4]。本報告では、このKIDOCH IV 用のCコンパイラについて実行効率を向上させるための様々な最適化について述べている。本文では特に、中間コードにおけるグローバルな最適化に関して検討を行った。

2.VLIW型計算機の最適化

VLIW型計算機の特徴を生かすためには、高度のコンパイラ技法が必要である[5]。その中で、我々は一般のプログラムにおいて処理能力が落ちる可能性のない汎用性のある最適化技法だけを行ってCコンパイラを設計している。

一般の計算機と同様にVLIW型計算機でも、分岐命令はパイプラインを乱し結果として演算速度が落ちる。つまり、演算速度を上げるためには、分岐命令がない連続して処理が行える1つのまとまり(ブロック)の長さをできるだけ長くする必要がある。

これらの分岐命令に対する最適化のうちif文については前回報告した[6]。しかし、従来開発した分岐命令最適化は1つの分岐に対しては有効であるが例えば、if文が2つ以上続く場合には、それらにまたがる最適化は行えなかった。そこで、本文では"for文の中にif文がある"の場合を例にして、このような場合のグローバルな最適化について検討を行った。

3.中間コードのグローバル最適化(ループにおける最適化)

図1に示すサンプルプログラムに対して筆者らが前回までに開発したKIDOCH C コンパイラで得られた中間言語(最適化処理後)を図2に示す。図2において、 if文の後の演算とelse文の後の演算はどちらも分岐命令(if文)の前に移動されており、(15,16行)分岐命令の後には退避命令(18-20行)があるだけである(図中T1 -T26は中間結果で、これは後で使う必要があればALU中のレジスタに退避され、その場限りの場合にはそのまま必要な演算器に送られる)。ところが、その前の for文においては評価式の演算をする際に同じ値(iの値)を2回ロードしており、さらに、INC処理がif文の後にきており、ブロックのまとまりが欠けている。また、本文では省略しているが、if文の後にif文に無関係の変数に対しての演算があるときにも、その処理がif文の後で行なわれており、小さ

Global Optimization in the C Compiler for VLIW Computer KIDOCH
S. Hongo[+], M. Abe[++], N. Chubachi[+], and K. Kido[+++]
[+]Tohoku University, [++]Computer Center,Tohoku University, [+++]Chiba Institute of Technology

なブロックに分かれてしまう。一方今回行ったオプティマイズ方式では、図3に示すように無駄なLOADはなくなり、また、INC もif文の前に行っており(11行)、ブロックとしてのまとまりも良くなっている。また、if文の後の命令の処理もif文に無関係の場合には、できるだけ前の段階でまとめて処理するようになるため、ブロックは長くなり、結果として演算速度は向上する。図1のプログラムの例では、ブロック長が短いため、VLIWの特徴をそれほど有効には表せていないが、長いブロックのあるプログラムにおいては、本文の最適化手法の効果が一段と発揮できる。

4.むすび

VLIW型計算機KIDOCH IV用Cコンパイラの最適化について特に分岐処理のグローバルな最適化に関して検討した結果を報告した。

参考文献

[1] 安倍他:音響ディジタル信号処理を主目的とする高速演算装置μKIDOCH,情報処理学会論文誌, 28,12,pp.1306-1317(1987)

[2] 永田他:VLIW型計算機KIDOCH用Cコンパイラ, 第15回東北大学応用情報学研究センターシンポジウム予稿集, pp.77-82(1989)

[3] 安倍他:VLIW型計算機KIDOCHのアーキテクチャ, 第15回東北大学応用情報学研究センターシンポジウム予稿集, pp.61-76(1989)

[4] 安倍他:VLIW型計算機KIDOCHのメモリ管理機構, 電子情報通信学会研究報告 CPSY89-40, pp.55-59(1989)

[5] 横田 :次世代RISC,並列処理を導入し、CMOSで10MIPSをねらう, 日経エレクトロニクス, 11.27(No.487), pp.191-200(1989)

[6] 安倍他:VLIW型計算機KIDOCH用Cコンパイラにいおけるif文の最適化, 第40回情報処理学会全国大会講演論文集(III), pp1265-1266(1990)


```
main()
{
        int     i,a[10],b;

        for(i=0;i<10;i++){
                if (a[i]<0)
                        b -= a[i];
                else
                        b += a[i];
        }

}
```

図1

```
L       LOAD    #0              T1      Ld imm #0
        LOAD    #10             T2      Ld imm #10
        ADD     &i              T3
        ADD     T3      fp      T4      Cal add "i"
        LEA     &a              T5
        ADD     T5      fp      T6      Cal add "a[0]"
        LEA     &b              T7
        ADD     T7      fp      T8      Calculate address of "b"
        SAVE    T4      T1              Save "i" = #0
L#0     LOAD    T4              T11     Load i
        ADD     T6      T11     T12     Calculate address of "a[i]"
        LOAD    T12             T13     Load a[i]
        ELT     T13     T1      T14     Compare T13 and T1
        LOAD    T8              T15     Load b
        SUB     T15     T13     T16     Calculate "b - a[i]"
        ADD     T15     T13     T17     Calculate "b + a[i]"
        JF      T14             L#1     Jump to Label #1 if T14 is false
        SAVE    T8      T16             Save T8("b") to T16
        JMP                     L#2     Jump Label #2
L#1     SAVE    T8      T17             Save T8("b") to T17
L#2     LOAD    T4              T24     Load "i"
        INC     T24             T25     Inc "i"
        ELT     T25     T2      T26     Compare T25 and T2
        SAVE    T4      T25             Save "i+1"
        JT      T26             L#0     Jump Label #0 if T26 is true
        END
```

図2

```
L       LOAD    #0              T1      Load immediate data #0
        LOAD    #10             T2      Load immediate data #10
        LEA     &i              T3
        ADD     T3      fp      T4      Calculate address of "i"
        LEA     &a              T5
        ADD     T5      fp      T6      Calculate address of "a[0]"
        LEA     &b              T7
        ADD     T7      fp      T8      Calculate address of "b"
        SAVE    T4      T1              Save "i" = #0
L#0     LOAD    T4              T11     Load i
        INC     T11             T12     Inc "i"
        ADD     T6      T11     T13     Calculate address of "a[i]"
        LOAD    T13             T14     Load a[i]
        ELT     T14     T1      T15     Compare T13 and T1
        LOAD    T8              T16     Load b
        SUB     T16     T14     T17     Calculate "b - a[i]"
        ADD     T16     T14     T18     Calculate "b + a[i]"
        JF      T15             L#1     Jump to Label #1 if T14 is false
        SAVE    T8      T17             Save T8("b") to T16
        JMP                     L#2     Jump Label #2
L#1     SAVE    T8      T18             Save T8("b") to T17
L#2     ELT     T12     T2      T25     Compare T25 and T2
        SAVE    T4      T12             Save "i+1"
        JT      T25             L#0     Jump Label #0 if T26 is true
        END
```

図3