『新風』プロセッサの条件分岐方式

7L－6

原 哲也　久我守弘　村上和彰　富田眞治
(九州大学大学院総合理工学研究科)

1. はじめに

我々は，スーパースカラ・プロセッサ・アーキテクチャとしてSIMP（単一命令流／多重命令パイプライン）方式を提案し，これに基づく試作プロセッサ『新風』を開発している．[1] 『新風』プロセッサの命令セット・アーキテクチャは，我々が提案しているBISC（Balanced Instruction Set Computer：均衡命令セット・コンピュータ）[2]に基づいており，その1具現化例である．『新風』の条件分岐は，“先行条件決定命令”と“分岐命令”との組合せで行うが，これは『新風』の命令セット・アーキテクチャの特長の1つである．

本稿では，従来のプロセッサにおける条件分岐方式の問題点を整理し，『新風』プロセッサの条件分岐方式について述べる．

2. 条件分岐処理過程

条件分岐は，一般に以下の処理から成る．

①コンディション生成：条件分岐の元になるコンディション（状態）を生成する．このコンディションは，算術・論理演算等を実行することにより生成される．

②条件決定（conditioning）：①で生成したコンディションを分岐条件でテストし，分岐するか否かを決定する．

上記①②は分岐するしないに関わらず行う．分岐する場合は，さらに以下の処理が必要である．

③分岐先アドレス生成：分岐先の命令アドレスをアドレッシング・モードに従って計算する．なお，アドレッシング・モード次第で，この処理は省ける．

④分岐処理：③で生成した分岐先アドレスをプログラム・カウンタ（PC）に設定する．

上記の4処理には，その先行制約に関し，①→②→④，および，③→④という半順序関係（→：先行制約）が存在する．

3. 従来の条件分岐方式

従来のプロセッサで主に用いられている条件分岐方式には，compare-and-branch方式とbranch-on-condition方式がある．

3.1 compare-and-branch方式

表1に示すように，条件分岐処理過程の①②③④のすべてを1個のcompare-and-branch命令で行う．次に述べるbranch-on-condition方式における問題点がない反面，クリティカルパスが①→②→④と長いため分岐コストが大きくなる欠点がある．

3.2 branch-on-condition方式

コンディション・コード（CC：Condition Code）を用いて，①と②③④とを別々の命令で行うようにする．表1に示すように，①は通常の算術・論理演算命令におけるCC設定で，②③④は1個のbranch-on-condition命令で行う．branch-on-condition命令自身のクリティカルパスが②→④ないし③→④と短くなるため，compare-and-branch命令に比べて分岐コストは小さくなる．しかし，CCを導入したことで，以下の問題点が生じる．

ⓐCCに関するフロー依存関係：CCを設定する演算命令とCCを参照するbranch-on-condition命令との間にフロー依存関係が生じるので，一般の命令間依存関係と同様にこれを保証する必要がある．単純なパイプライン・インターロックでこれは解決可能だが，パイプラインに乱れが生じる．乱れを抑えるには，CCスコアボーディングなどの機構が必要である．

ⓑCCに対するアクセス競合：CCを設定／参照する命令間で，CCに対するアクセス競合が生じ得る．よって，競合関係にある命令間の逆依存および出力依存関係を保証するために，上記ⓐ同様パイプラインに乱れが生じる．アクセス競合の頻度を軽減するには，CCを複数個設けるなどの手段が必要となる．ただし，この場合，CCを設定／参照するすべての命令にCC番号を指定するフィールドが必要となる．

ⓒ静的コードスケジューリングへの影響：branch-on-condition方式は，CC設定を行う演算命令とCC参照を行うbranch-on-condition命令とを離せる場合にのみ，compare-and-branch方式に対する優位性がある（図1（a）（b）参照）．この場合，当該2命令の間に1個以上の命令が存在することになるが，そのいかなる命令もCCを変更してはいけない．このとき，CC設定方式の相違により，次の影響が生じる；

・暗黙的設定：CCを設定するか否かを暗黙的に定める．CCを設定しない命令しか当該2命令間に挿入できないので，コードスケジューリングの自由度が低くなる欠点がある．

・明示的設定：命令フィールド中でCCを設定するか否かを明示的に指定する．暗黙的設定に比べると自由度は高いが，CCが1個しかない場合には，CC設定－参照関係の入れ子ないし交差が許されないという制限がつく．

4. 『新風』における条件分岐方式

4.1 先行条件決定（advanced-conditioning）方式

3章で述べた条件分岐方式以外に，先行条件決定（advanced-conditioning）方式と呼ぶ方式が提案されている．[3] 本方式は表1に示すように，①はbranch-on-condition方式同様，通常の算術・論理演算命令におけるCC設定で行うが，②と③④とは別々の命令で行う．②の条件決定を④の分岐に先行して行う命令を“先行条件決定（advanced-conditioning）命令”と呼ぶ．③④は1個の“分岐（branch）”命令で行い，そのクリティカルパスは③→④と短い（さらに，アドレッシング・モード次第では，③が省けて④のみとなる）．

本方式では，先行条件決定命令の実行結果は「分岐するか否か（TF：True／False）」の1ビット情報で，それはレジスタ（TFレジスタと呼ぶ）に格納されて分岐命令に伝達される．このTFは，branch-on-condition方式のCCに比べて，以下の特徴がある．

ⓐTFに関するフロー依存関係：CCの場合と同様．

ⓑTFに対するアクセス競合：複数のTFレジスタを設けることで，アクセス競合を緩和できる．この場合，TFレジスタを設定／参照する命令は先行条件決定命令と分岐命令だけであるから，他の命令のフォーマットには影響を与えない．

ⓒ静的コードスケジューリングへの影響：複数のTFレジスタを設けることで，TF設定－参照関係の入れ子および交差を含めて，自由度の高いコードスケジューリングが可能である（図1（c）（d）参照）．

4.2 改良されたCC

『新風』における条件分岐処理は先行条件決定方式に基づく．しかしながら，本方式でも，①のコンディション生成と②の条

表1．条件分岐方式

方式＼処理	①コンディション生成	②条件決定	③分岐先アドレス生成	④分岐
compare-and-branch	compare-and-branch命令			
branch-on-condition	算術・論理演算命令	branch-on-condition命令		
advanced-conditioning	算術・論理演算命令	先行条件決定命令	branch命令	

A Conditional Branch Architecture for the 【ʃɪmpuː】 Processor
Tetsuya HARA, Morihiro KUGA, Kazuaki MURAKAMI and Shinji TOMITA
Kyushu University

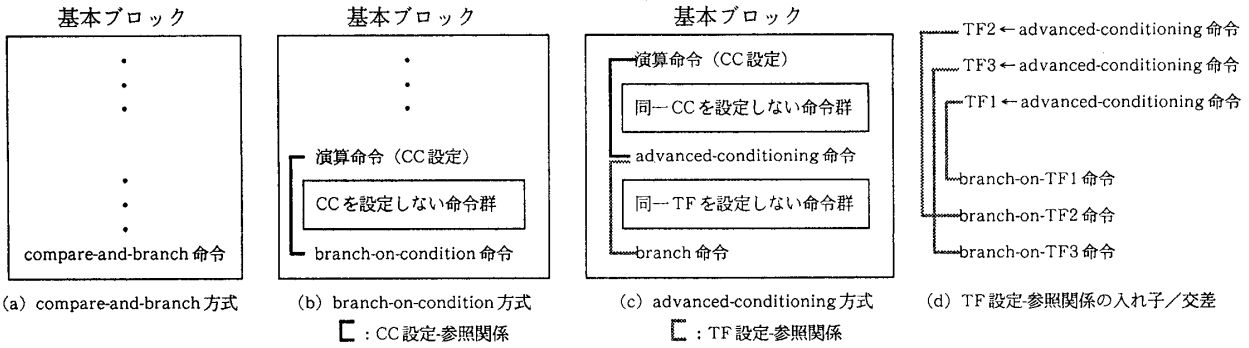

図1. 静的コード・スケジューリングへの影響

件決定との間ではCCを用いるので，3. 2節で述べたCCに関する問題はそのまま残る．この問題は，次のようにCCを改良することで解決している．

すなわち，CCを独立した単一のレジスタとしてではなく，個々の汎用／浮動小数点レジスタに付随するタグとして定義する．よって，CCを設定する命令を実行すると，そのデスティネーション・レジスタのタグにCCが設定される．この改良されたCCは，通常のCCに比べて以下の特徴がある．

ⓐ CCに関するフロー依存関係：レジスタ・スコアボーディング機構により同時に解決される．

ⓑ CCに対するアクセス競合：汎用／浮動小数点レジスタと同数個のCCが存在し，アクセス競合が緩和される．しかも，CCの指定はデスティネーション・フィールドを兼用するので，新たなフィールドは必要ない．

ⓒ 静的コードスケジューリングへの影響：CCへの設定は暗黙的に行う．ただし，branch-on-condition命令と異なり先行条件決定命令は任意の位置に置けることから（branch-on-condition命令は基本ブロックの底部のみ），CCが変更される前に条件決定を済ませることが可能である（図1（c）参照）．

5. 『新風』の条件分岐処理の詳細

5.1 関連レジスタ

条件分岐に関連して，次の2種類のレジスタを定義する．

i) 汎用／浮動小数点レジスタのタグ：算術・論理演算命令の実行により生成されたCCを保持する．

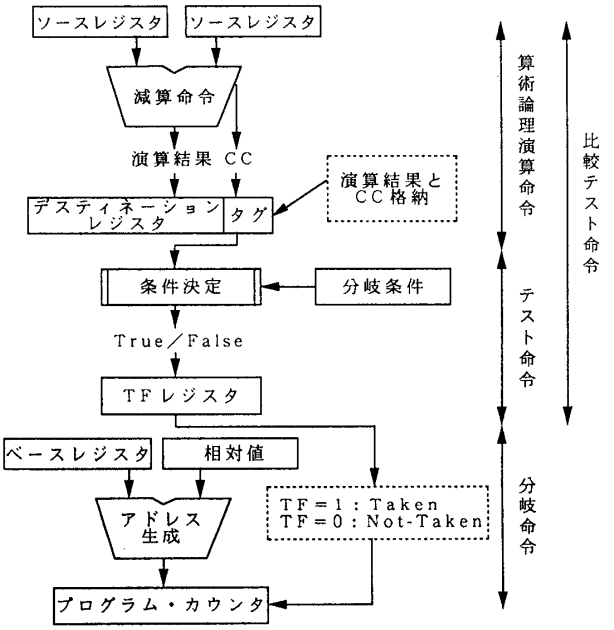

図2.『新風』における分岐処理命令

ii) TFレジスタ（TFR）：32個の1ビット長レジスタで，「分岐するか否か（0：Not-Taken／1：Taken）」を保持する．ただし，TFR0の値は常に1（Taken）で，無条件分岐の際にはこれを指定する．

5.2 関連命令

条件分岐に関連して，次の3種類の命令を定義する．

i) 算術・論理演算命令：実行により生成したCCをデスティネーション・レジスタのタグに設定する．

ii) 先行条件決定命令：分岐条件を用いて条件決定を行い，その結果をTFRに設定する．これには，次の2命令がある；

・テスト命令：ソースレジスタのタグ内のCCに対して分岐条件と合致するか否かをテストする．

・比較＆テスト命令：2つのソースオペランド間の算術・論理比較を行い，その結果に対して分岐条件と合致するか否かをテストする．

iii) 分岐命令：ソースTFRの値が1（Taken）の場合，プログラム・カウンタ（PC）に分岐先アドレスを設定する．分岐先アドレスはアドレッシング・モード（ベース相対／PC相対）に従って計算する．

5.3 条件分岐処理過程

条件分岐処理過程は，以下のようになる（図2参照）．

① コンディション生成：算術・論理演算命令により，そのデスティネーション・レジスタのタグにCCを設定する．あるいは，比較＆テスト命令により，算術・論理比較を行う．

② 条件決定（conditioning）：先行条件決定命令（テスト命令および比較＆テスト命令）により，①で生成したCCを分岐条件でテストし，分岐するか否かを決定する．その結果をデスティネーションTFRに設定する．

③ 分岐先アドレス生成：分岐命令により，分岐先の命令アドレスを計算する．なお，ソースTFRの値が0（Not-Taken）の場合，あるいは，相対値が0の場合，この処理は省く．

④ 分岐処理：ソースTFRの値が1（Taken）の場合，③で生成した分岐先アドレスをPCに設定する．

6. おわりに

以上，『新風』プロセッサの条件分岐方式について述べた．本方式は，分岐関連レジスタへのアクセス競合の緩和，および，自由度の高い静的コード・スケジューリングを可能としている．

参考文献

[1] K. Murakami et al. : SIMP (Single Instruction stream／Multiple instruction Pipelining) : A Novel High-Speed Single-Processor Architecture, Proc. 16th ISCA, pp. 78-85, May 1989.

[2] 久我ほか：『新風』プロセッサの命令セット・アーキテクチャ，情処38全大論文集，5T-8（1989年3月）．

[3] W. G. Rosocha and E. S. Lee : Performance Enhancement of SISD Processors, Proc. 6th ISCA, pp. 216-231, April 1979.