様式：課程博士用

25年 1 月 10日

電子・情報工学専攻		学籍番号	D085403	指導教員	中川 聖一
申請者氏名	下村 佳生				小林 良太郎

論 文 要 旨 (博士)

論文題目	予測に基づいて性能向上を行うマイクロプロセッサに関する研究

(要旨 1,200 字程度)

近年のマイクロプロセッサは，高速化，省電力化のために，処理の細分化，並列化が行われている．高い性能を達成するためには，プログラムに高い並列性が内在する必要があるが，それらは複数の要因（**依存関係**）によって厳しく制限される．例えば，あるデータを生成する命令と，使用する命令の間にはデータ依存関係があり，逐次的に実行することしかできない（**真の依存**）．また，分岐命令は実行が完了するまで成立するかどうかがわからず，次に続く命令の読み込みが行えない（**制御依存**）．これらの依存は，プロセッサ性能に重大な悪影響を及ぼす．それに対し，これらの依存を緩和するため，値予測，分岐予測と呼ばれる機構が提案されている．これらの機構は，命令の実行前に結果を予測することで，プロセッサに投機的な実行を促し，性能向上を達成している．

予測機構は，性能向上のための重要な機構であるが，ハードウェア量の増加，消費エネルギー増加が問題となる．そこで本論文では，値予測機構の性能向上，省電力化，分岐予測機構の省電力化に焦点をあて，より効率良く予測機構を扱うための研究を行った．値予測機構は命令の実行結果を，命令フェッチ時に予測する機構であり，真の依存を緩和することで，命令レベル並列性を高めることができる．値予測は，値履歴表(VHT: Value History Table)に過去の実行結果を保持し，再度同じ命令が出現した際に，保持している値を用いて命令の実行結果を予測する．しかし，値を生成する命令には，特定のパターンを示す，予測可能な命令と，アトランダムな値を生成する予測不可能な命令が混在しており，全ての命令が予測可能とは限らない．VHT のサイズは有限であるため，予測不可能な命令を登録すると，競合性ミスが増加する．そこで，この研究では，予測容易性を予め判定してから，値予測機構へ登録を行うことで，競合の抑制を行う機構を提案した．この機構によって，従来機構の半分のハードウェア容量で，同程度の予測性能を得ることが可能となった．

予測は命令フェッチ時に行われるため，予測機構にアクセスする命令が，予測対象となっている命令かどうかはわからない．また，予測対象となっていても，予測可能になっているとは限らない．予測可能でない命令が値予測機構にアクセスを行うと，動的エネルギーが増加する．この研究では，予測可能になっているかどうかの情報を持たせ，予測機構への無駄なアクセスを抑制することで，値予測機構，分岐予測機構の動的エネルギー削減する機構を提案し，エネルギー削減率を定量的に示した．

これらの提案機構により，値予想機構，分岐予測機構の性能向上，省電力化を達成し，より効率良く予想機構を実現することが可能となった．

year	month	day
2013	1	16

Department	Electronic and Information Engineering	ID	D085403
Name	Yoshio Shimomura		

Supervisor	Seiichi Nakagawa Ryotaro Kobayashi

A b s t r a c t

Title	Study on High-Performance Microprocessors Based on Predictions

(800 words)

The current microprocessors use the parallelism included in the program to achieve high performance. Inherent parallelism in programs is restricted by multiple factors. Among them, there are true data dependencies and control dependencies. The true data dependency occurs when an instruction depends on the result of a previous instruction. In order to mitigate them, value prediction techniques are used. The control dependency is caused by a branch instruction. It results in fetch stall until the end of the execution of the branch instruction. To mitigate this problem, branch prediction techniques are proposed.

This paper proposes three mechanisms to achieve the efficiency of these prediction mechanisms. Generally, value predictions use the relations among past and future values. Past values are stored in a table called **VHT** (Value History Table), and this history is used to predict future data. VHT is indexed by the instruction address and is referred and updated during the instruction fetch and commit stage, respectively. This study focuses on the "stride value predictor". Each entry in VHT consists of a tag field, a value field, a stride field, and a state field. The tag is the upper bits of an instruction address. The value indicates the last instruction result. The stride indicates the difference between the last two results. The state indicates the state of the value history: "Initial", "Transition", or "Steady". The update operation is described as follows: First, the state is specified as "Initial". Second, the stride is calculated and the state changes from "Initial" to "Transition". Subsequently, if the stride becomes constant, the state changes to "Steady"; otherwise, the state changes to "Transition". When an instruction is fetched, VHT is referred and the corresponding entry is read. If the state is "Steady", the predicted value is obtained by adding the value and stride. In the first mechanism, we propose the mechanism that moves only the instruction that became "Steady" to VHT. The target instruction for the value prediction is registered in **PVT** (Predictability Verification Table) in the beginning. It moves to VHT when the entry is updated, and it changes to the "Steady" state. The instruction that doesn't become "Steady" state stays in PVT. As a result, it becomes possible to register only instruction to which it can predict in VHT. In the second mechanism, we propose a more efficient mechanism for a value predictor that extends the use of an existing **BTB** (branch target buffer) to reduce the number of invalid VHT references. BTB is a buffer used to predict the target address of a branch instruction. The proposed mechanism introduces a **p-bit** (predictability bit) to identify an instruction and added a field to BTB. The proposed mechanism controls VHT references according to **p-bits** (series of p-bit). The proposed mechanism comprises the extended BTB, refer unit, update unit, and VHT. Except for VHT, all these components employ p-bits. Each bit of p-bits indicates whether the corresponding instruction following the last taken branch is predictable or not. The update unit sets the p-bit on the basis of the predictability of the committed instruction, which is predictable only if the corresponding VHT state is "Steady". The refer unit refers to VHT on the basis of the p-bit corresponding to the fetched instruction.

A branch prediction consists of the branch direction prediction and the branch target prediction. It consumes considerable energy because BTB is large and is referenced very frequently. In the third mechanism, we propose a more efficient mechanism for a branch predictor that uses a dedicated table, called **BB Table** (branch bit table) to reduce the number of invalid BTB references. The proposed mechanism introduces a **B-Bit** (branch bit) and **B-Bits** (series of B-Bit) to identify a branch instruction, and each entry of BB Table contains multiple B-Bits corresponded to statically continuous instructions. The proposed mechanism controls BTB references according to B-Bits. The proposed mechanism consists of BTB, extended refer unit, extended update unit, and BB Table. Each bit of B-Bits indicates whether the corresponding instruction following the last taken branch is branch instruction or not. The refer unit refers to BTB on the basis of the B-Bit corresponding to the fetched instruction.

We used the SimpleScalar Tool Set to simulate the architecture layer and ran eight programs from the SPECint2000 suite as benchmarks. In the first experiment, we confirmed to obtain the equivalent performance by using half the number of entries of conventional mechanisms. In the second experiment, the evaluation results showed that this technique achieved significant improvements in efficiency and small losses of performance. The proposed mechanism reduces invalid references by 42.4% and energy by 25.7% with 0.1% performance loss on average in the 32 p-bits length. In the third experiment, the evaluation results show that this mechanism reduces energy consumption without performance loss. Unnecessary references and energy consumption are reduced by 71.0% and 55.4% on average with the 16 B-Bits length, respectively.