슈퍼스칼라 프로세서에서 모험적 갱신을 사용한 하이브리드 결과값 예측기

(A Hybrid Value Predictor using Speculative Update in Superscalar Processors)

박 홍 준 † 신 영 호 †† 조 영 일 †††
(Hong-Jun Park) (Young-Ho Shin) (Young-Il Cho)

요 약 슈퍼스칼라 프로세서는 성능향상을 위해 명령어 반입폭과 이슈율을 증가시키고 있다. 데이타 종속성은 ILP(Instruction-Level Parallelism)를 향상시키는데 주요 장애요소가 되고 있으며, 최근 여러 논문에서 데이타 종속성을 제거하기 위해서 명령어의 결과값을 예상하는 메커니즘이 연구되었다. 그러나 이러한 예측기들은 예상한 명령어의 실제 결과값으로 예상 테이블을 갱신하기 전에 그 명령어를 다시 예상할 때 부적절(stale)한 데이타를 사용함으로써 예상 실패율이 증가하여 프로세서의 성능은 감소시킨다.

본 논문에서는 부적절 데이타 사용을 줄여 높은 성능을 얻을 수 있는 새로운 하이브리드 예측 메커니즘을 제안한다. 제안된 하이브리드 결과값 예측기는 예상 테이블을 모험적으로 갱신할 수 있기 때문에 부적절 데이타로 인해 잘못 예상되는 명령어의 수를 효과적으로 감소시킨다. 16-이슈폭 슈퍼스칼라 프로세서에서 SPECint95 벤치마크 프로그램에 대해 모험적 갱신을 사용함으로써 모험적 갱신을 사용하지 않은 경우의 평균 예상 정확도 59%에 비해 평균 예상 정확도가 72%로 크게 향상되었다.

Abstract To improve the performance of wide-issue Superscalar microprocessors, it is essential to increase the width of instruction fetch and issue rate. Data dependences are major hurdle to exploit ILP(Instruction-Level Parallelism) efficiently, so several related works have suggested that the limits imposed by data dependences can be overcome to some extent with the use of the data value prediction. But the suggested mechanisms may access the same value prediction table entry again before they have been updated with a real data value. They will cause incorrect value prediction by using stale data and incur misprediction penalty and lowering performance.

In this paper, we propose a new hybrid value predictor which achieve high performance by reducing stale data. Because the proposed hybrid value predictor can update the prediction table speculatively, it efficiently reduces the number of mispredicted instruction due to stale data. For SPECint95 benchmark programs on the 16-issue superscalar processors, simulation results show that the average prediction accuracy increase from 59% for non-speculative update to 72% for speculative update.

1. 서 론

현재 고성능 마이크로프로세서는 여러 개의 실행 유닛을 사용하여 다수의 명령어를 병렬로 처리함으로서 프로세서의 성능을 향상할 수 있게 설계되고 있다. 이러한 고성능 프로세서에서 성능을 향상시키기 위해서는 높은 명령어 이슈율(Issue rate)과 명령어 수준 병렬성(ILP: Instruction Level Parallelism)을 가능한 최대로 이용하는 것이 중요하다. 그러나 ILP를 저해시키는 주요 장애요소에는 제어 종속과 데이타 종속 등이 있다.

제어 종속은 조건분기 명령이 반입될 때 분기 방향과 다음에 수행할 명령어 주소를 예상하는 분기 예상기법으로 해결할 수 있다. 데이타 종속에는 False-데이타 종속과 True-데이타 종속이 있으며, False-데이타 종속

† 비 회 원 : 극동정보대학 전산정보처리과 교수
hjpark@cs.kdc.ac.kr
†† 비 회 원 : (주)에어미디어 기술연구소 연구원
yhshin@airmedia.co.kr
††† 정 회 원 : 수원대학교 컴퓨터과학과 교수
ycho@mail.suwon.ac.kr
논문접수 : 2001년 1월 3일
심사완료 : 2001년 8월 27일

에 의한 장애는 소프트웨어 및 하드웨어 재명명(rename) 방법으로 완전히 제거될 수 있다. 그러나 선행 명령어의 결과를 후행 명령어가 입력으로 사용하는 True-데이타 종속관계의 명령어는 병렬 수행을 제약한다. 이러한 문제를 해결하기 위해서 결과값 예상기법을 사용하게 된다. 결과값 예상기법은 True-데이타 종속적인 명령어가 선행 명령어의 결과값이 나올 때까지 기다리는 것이 아니라, 예상된 선행 명령어의 결과를 모험적으로 사용하여 True-데이타 종속 연결관계를 제거하는 하드웨어 메커니즘이다[1-8].

결과값 예상기법 중 명령어의 결과를 이전에 수행된 결과로 예상하는 최근 결과값 예측기가 있다[1,2]. 이 방법은 마지막으로 수행된 결과값(Last Value) 한 개만 예상 테이블에 저장하므로 적은 하드웨어를 사용하고 구현이 쉽다는 장점이 있으나, 수행결과가 변하지 않는 경우에만 사용될 수 있다는 단점이 있다. 또한 명령어의 결과가 마지막으로 수행된 결과값에 일정한 값만큼 변한다는 사실을 이용하여 명령어의 다음 결과를 예상하는 스트라이드 결과값 예측기가 있다[6,7]. 이 방법은 반복문의 카운터 변수나 결과값이 일정하게 증감하는 명령어를 갖는 프로그램에서 아주 좋은 성능을 발휘하게 된다. 또한 2-단계 결과값 예측기[6]는 이전에 실행된 n개의 결과값 중에 하나를 다음 값으로 예상하는 방법으로 높은 예상 정확도를 가지나 n개의 결과값을 저장함으로서 구현 시 다른 예측기들 보다 많은 하드웨어 비용을 필요로 한다. 기존의 결과값 예측기는 다양한 특성을 갖는 여러 명령어들을 모두 예상할 수 없기 때문에 이러한 단점을 극복하기 위해 여러 예측기를 혼합해서 사용하는 하이브리드 예측기가 제안되었다[6,8,12,13].

고성능 프로세서에서 명령어의 반입폭과 이슈폭은 계속해서 증가하고 있다. 이로 인해 명령어가 예상된 후 실제 결과값으로 예상 테이블을 수정하기 전에 다시 그 명령어를 예상하면 부적절한 데이타를 사용하게 된다. 이러한 부적절한 데이타 사용은 올바르지 못한 예상을 발생시키게 되고, 이러한 잘못된 예상은 프로세서의 성능을 저하시킨다. 특히 스트라이드 결과값 예측기와 2-단계 결과값 예측기에서는 이러한 예상 실패로 인해 성능에 미치는 영향이 심각하다. 이러한 문제점을 부분적으로 해결하기 위해 Lee[13]등은 스트라이드 결과값 예측기를 모험적 갱신하는 방법을 제안하였으나 2-단계 결과값 예측기는 모험적 갱신을 하지 않으므로 부적절한 데이타에 의한 예상 실패가 여전히 문제점으로 남아있다.

본 논문에서는 부적절한 데이타를 사용함으로서 발생되는 예상 실패를 최소화하기 위해 명령어의 결과값을 예상하는 동시에 예상 테이블을 모험적으로 갱신하는 하이브리드 결과값 예측기를 제안한다. 제안한 예측기는 결과값이 구해진 후 예상 테이블을 갱신하기 전에 동일 명령어의 값을 다시 예상할 때, 문제점으로 나타나는 부적절한 데이타의 사용을 방지함으로써 예상 정확도를 향상시킬 수 있고, 부적절한 데이타로 인해 잘못 예상되는 명령어의 수와 예상실패 페널티를 감소 시켜 프로세서의 성능을 향상시킨다.

2. 관련 연구

이 장에서는 기존의 대표적인 결과값 예측기인 최근 결과값 예측기, 스트라이드 결과값 예측기, 2-단계 결과값 예측기, 하이브리드 결과값 예측기를 설명하고 그들의 장단점을 고찰한다.

2.1 최근 결과값 예측기

최근 결과값 예측기는 명령어가 최종적으로 수행된 결과값을 저장해서 다음에 동일한 명령어를 만났을 때, 바로 이전의 수행 시 저장된 결과값을 예상 값으로 사용하는 방법이다[1,2]. [그림 1]은 최근 결과값 예측기의 구조를 보여주고 있다. VHT(Value History Table)의 각 엔트리는 태그(Tag) 필드, 한 개의 결과값(Value) 필드, 신뢰성 카운터(Confidence Counter) 필드로 구성되어 있다. 태그 필드는 명령어 주소의 일부를 저장해 놓은 부분이고, 결과값 필드는 마지막으로 수행된 명령어의 결과값을 저장하는데 사용되며, 예상을 결정하기 위한 신뢰성 카운터 필드는 2비트로 4가지 상태('0','1','2','3')를 가지며 '2'와 '3'일 경우만 예상을 수행한다.

최근 결과값 예측기는 적은 하드웨어 비용을 요구하지만 상수값과 같은 단순한 패턴만 예상할 수 있어서 예상 정확도는 40%~50%로 낮다는 단점을 갖는다[9].

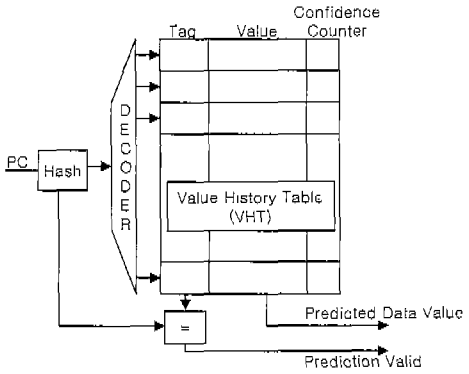

그림 1 최근 결과값 예측기의 구조

2.2 스트라이드 결과값 예측기

스트라이드 결과값 예측기[6,11]는 명령어의 마지막 수행된 결과값과 마지막 두 번의 수행된 결과값의 차(stride)를 VHT에 저장하여 명령어의 반입(fetch)시 마지막 수행 결과값과 스트라이드의 합으로 예상한다.

[그림 2]는 스트라이드 결과값 예측기의 VHT 엔트리 구조를 보여주고 있다. 상태(State) 필드는 2비트로 3개의 상태(Init, Transient, Steady)를 갖는다. 상태 필드가 Init('0') 혹은 Transient('1')를 나타내는 경우는 예상을 위해 준비중인 단계로 예상을 수행하지는 않으며, Steady('2')를 나타내는 경우에만 결과값 필드와 차이값 필드의 합을 예상 값으로 제공한다. 이 방법은 최근 결과값 예측기보다 좋은 성능을 가진다.

Tag	State	Value	Stride
20	2	32	8

그림 2 스트라이드 결과값 예측기의 VHT 엔트리 구조

2.3 2-단계 결과값 예측기

2-단계 결과값 예측기[5]는 명령어가 수행한 마지막 4개의 결과값을 VHT에 저장하여 이를 바탕으로 결과값을 예상하는 방법이다. 2-단계 결과값 예측기의 VHT 엔트리 구조는 [그림 3]과 같다. 결과값(Four values) 필드는 최근의 결과 값들 중에서 중복되지 않은 서로 다른 네 개의 결과값을 저장한다. LRU(LRU_info) 필드는 결과값 필드에 저장된 값들의 사용된 순서 정보를 가지고 있으며, VHP(Value History Pattern) 필드는 결과가 나타난 순서의 위치가 저장된다.

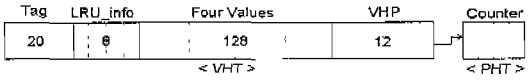

그림 3 2-단계 결과 값 예측기의 엔트리 구조

VHP로 인덱스되는 PHT(Pattern History Table)는 VHT의 4개의 결과값 필드에 대응되는 4개의 카운터 필드를 사용하며, 이 카운터 필드 중에서 최대값으로 설정되어 있는 위치의 결과값을 예상값으로 사용하게 된다.

2-단계 결과값 예측기는 높은 예상 정확도를 갖지만 4개의 결과 값을 저장하기 때문에 많은 하드웨어 비용을 요구한다는 단점을 갖는다.

2.4 Wang의 하이브리드 결과값 예측기

하이브리드 결과값 예측기[6]는 스트라이드 결과값 예측기와 2-단계 결과값 예측기를 결합한 것으로 신뢰성 카운터(Confidence Counter)에 의해 어느 한 예측기로부터 예상값을 선택하는 방법이다. 명령어가 두 개의 예측기에 모두 엔트리를 갖고 있다면, 높은 신뢰성 카운터 값을 갖는 예측기의 예상값을 선택한다. [그림 4]는 하이브리드 결과 값 예측기의 VHT 엔트리 구조를 보여주고 있다.

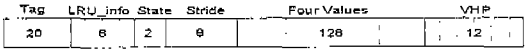

그림 4 하이브리드 결과 값 예측기의 엔트리 구조

하이브리드 결과값 예상방법은 스트라이드 특성을 갖는 명령어와 규칙적인 패턴을 갖는 명령어를 모두 예상하기 때문에 높은 예상 정확도를 갖는다는 장점을 갖지만, 명령어가 두 개의 예측기의 엔트리에 중복됨으로써 많은 하드웨어 비용을 요구한다는 단점을 갖는다.

2.5 Lee의 하이브리드 결과값 예측기

명령어가 예상 테이블을 갱신하기 전에 다시 그 명령어를 예상하는 경우에 발생하는 부적절한 데이타 사용은 예측기가 잘못된 값으로 예상하게 되어 프로세서의 성능을 저하시킨다. 이러한 문제점을 해결하기 위해 Lee[13]등은 명령 캐시와 결합된 PVC(Prediction Value Cache)에 예상 횟수를 기록하는 age 카운터를 사용하였다. age 카운터를 사용함으로써 스트라이드 결과값 예측기에서 예상 테이블을 모험적으로 갱신하는 방법을 제안하였으나 2-단계 결과값 예측기는 모험적 갱신을 하지 않으므로 부적절한 데이타에 의한 예상 실패가 여전히 문제점으로 남아있다.

3. 제안한 하이브리드 결과값 예상방법

이장에서는 본 논문에서 모험적 갱신의 필요성과 제안한 예측기의 구조 그리고 명령어에 가장 적합한 예측기를 선택하여 명령어들을 예상하는 예상 메커니즘에 대해 알아본다.

3.1 모험적 갱신의 필요성

본 논문에서는 모험적 갱신의 필요성을 입증하기 위해 예상 테이블을 모험적으로 갱신하지 않는 2-단계 결과값 예측기에서 부적절한 데이타로 예상하게 되는 명령어들의 비율을 실험을 통해 조사하였다. [그림 5]는 2-단계 결과값 예측기를 사용하여 각각 8-이슈폭과 16-이슈폭을 갖는 프로세서에서 전체 예상 명령어 중 부적절한 데이타로 예상하는 비율을 나타낸다. "age=1"은

전체 예상한 명령어 중 명령어가 예상된 후 결과가 구해지기 전에 그 명령어를 한 번 다시 예상하는 비율을 나타낸다. "age=2"는 전체 예상한 명령어 중 명령어가 예상된 후 결과가 구해지기 전에 그 명령어를 두 번 예상하는 비율을 나타내고, "age>=3"은 명령어가 예상된 후 결과가 구해지기 전에 그 명령어를 세 번 이상 다시 예상하는 비율을 나타낸다.

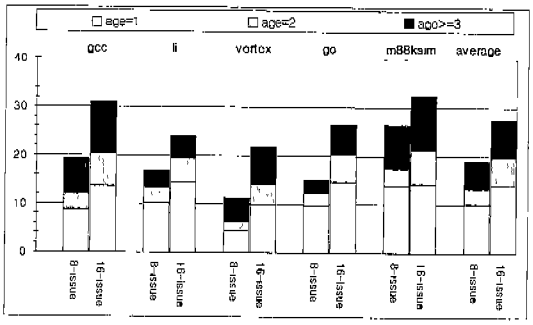

그림 5 2-단계 결과값 예측기의 부적절한 데이타 사용 비율

실험 결과를 보면 전체 예상 명령어 중 부적절한 데이타로 예상되는 비율이 8-이슈폭에서 평균 19%, 16-이슈폭에서 평균 28%로 나타난다. 이러한 결과는 이슈폭이 증가함에 따라 부적절한 데이타를 사용하는 비율이 증가한다는 것을 알 수 있으며, 명령어 반입 시 결과값을 예상하는 시간에 예상 테이블을 갱신할 필요성을 입증하고 있다.

3.2 하이브리드 결과값 예측기 구조

기존의 예측기들에서는 결과값이 예상된 후 실제 결과값이 생산되어 예상 테이블을 수정하기 전에 다시 그 명령어를 예상하여 부적절한 데이타를 사용하게 되는 문제점을 가지고 있다. 본 논문에서는 앞 절에서 분석한 결과를 바탕으로 부적절한 데이타에 의한 예상 정확도의 감소를 줄이기 위해 명령어 반입 시 결과값을 예상하는 동시에 예상 테이블을 모험적으로 갱신하는 하이브리드 결과값 예측기를 제안한다.

제안한 하이브리드 결과값 예측기는 최근 결과값 예측기(LVP), 스트라이드 결과값 예측기(SVP), 2-단계 결과값 예측기(TVP)로 구성된 하이브리드 예측기로, [그림 6]과 같이 기존의 하이브리드 예측기에서 스트라이드 결과값 예측기와 2-단계 결과값 예측기가 모험적 갱신을 수행하도록 수정하였다.

제안한 하이브리드 결과값 예측기는 기존의 하이

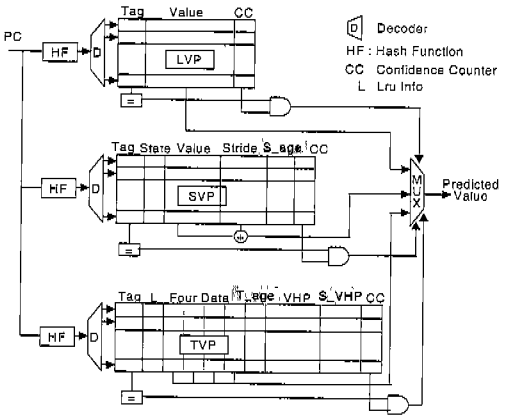

그림 6 제안한 하이브리드 결과값 예측기의 구조

브리드 결과값 예측기의 스트라이드 결과값 예측기에 S_age(Stride Age) 카운터 필드를 추가하였으며, 2-단계 결과값 예측기에 T_age(Two Age) 카운터 필드와 S_VHP(Speculative VHP) 필드를 추가하였다. 두 Age 카운터의 초기 값은 '0'으로 설정하였다. 그리고 명령어 반입 시, 결과값 예측기의 VHT를 조사(lookup)할 때 예상된 예측기의 Age 카운터 값을 '1'씩 증가시키고, 재기록할 때 '1'씩 감소한다. 2-단계 결과값 예측기의 S_VHP 필드는 6 비트로 구성되어 있으며, 모험적 갱신 시 예상된 결과값의 결과값 필드에 대응하는 위치를 저장한다.

3.3 하이브리드 예측기의 메카니즘

제안한 예측기의 예상 메커니즘을 설명하기 위해 먼저 2-단계 결과값 예측기 부분에 대해 설명한다. 신뢰성 카운터는 세 개의 예측기(최근 결과값 예측기, 스트라이드 결과값 예측기, 2-단계 결과값 예측기) 중 가장 큰 값을 갖는 예측기로 예상을 할 때 사용된다. 각 예상 테이블을 갱신할 때 올바른 값을 갖고 있는 예측기는 신뢰성 카운터를 증가하고 잘못된 값을 갖고 있는 예측기는 감소시켜 정확한 값을 갖는 예측기가 예상될 수 있도록 한다.

반입된 명령어가 결과값 예측기를 조사할 때 Age 카운터가 '1'이상이면 이전에 반입된 명령어가 갱신되기 전에 조사한 경우이다. 즉 부적절한 결과값을 사용하게 되는 경우가 발생한다. 그러나, S_VHP를 사용하여 PHT를 인덱스하면 문제를 해결할 수 있다. [그림 7]에서 보듯이 T_age 카운터의 값에 따라 올바른 VHP를 사용하여 PHT를 인덱스 함으로써 부적절한 결과값을 사용하지 않을 수 있다. 즉 T_age 카운터가 '0'일 경우

는 예상한 후 실제 결과값으로 예상 테이블이 갱신된 다음 명령어가 반입된 경우이므로 VHP값($P_0 \sim P_5$)을 사용하여 PHT를 인덱스하고, 예상된 결과값의 위치를 P_6에 저장하여 결과값이 유해지기 전에 그 명령어를 다시 예상 시 부적절한 데이타를 사용하여 예상되는 것을 방지한다. T_age 카운터가 '1'일 경우는 반입된 명령어가 한 번 수정되기 전에 명령어가 다시 반입된 경우임으로 VHP값($P_1 \sim P_6$)을 사용하여 PHT를 인덱스하고, 예상된 값의 위치를 P_7에 저장한다. T_age 카운터가 '2'일 경우는 반입된 명령어가 두 번 수정되기 전에 명령어가 반입된 경우이므로 VHP값($P_2 \sim P_7$)을 사용하여 PHT를 인덱스한다.

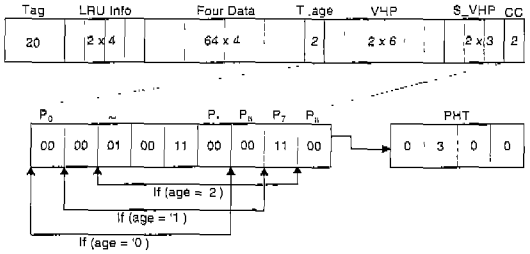

그림 7 제안한 예측기에서 TVP의 테이블 엔트리 구조

스트라이드 결과값 예측기에서 예상 테이블의 엔트리 구조는 [그림 8]과 같다. 명령어 반입 시 S_age를 '1' 증가시키고 예상값은 " $value + (S\_age * Stride)$ "로 예상한다. 또한 명령어 수행이 완료되면 재기록 시킬 동안 S_age는 '1' 감소시키고, 나머지 필드는 기존의 스트라이드 결과값 예측기에서와 같은 방법으로 갱신한다.

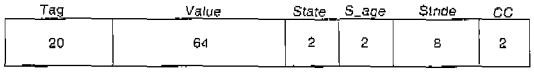

그림 8 제안한 예측기에서 SVP의 테이블 엔트리 구조

4. 실험 환경

제안한 모험적 갱신을 수행하는 하이브리드 결과값 예측기와 모험적 갱신을 수행하지 않는 하이브리드 결과값 예측기를 비교 측정하기 위해서 8-이슈와 16-이슈의 명령어 이슈 크기를 갖는 환경에서 실험하였다.

본 실험에서는 8K/16K 엔트리를 갖는 하이브리드 예측기에 대해 예상 빈도(Prediction rate) 및 예상 정확도(Prediction accuracy)를 비교, 분석한다. 본 논문의 실험을 위해 사용된 벤치마크 프로그램은 SPECint95 벤치마크 중 8개를 실험하였으며 시뮬레이터는 실행 구동 시뮬레이터인 SimpleScalar/PISA 3.0 툴셋[10]을 사용하였다.

[표 1]은 실험에 사용한 벤치마크 프로그램에 대한 설명이다. 첫 번째 열은 실험에 사용된 SPECint 95 벤치마크 프로그램들이고, 두 번째 열은 프로그램에 사용한 입력 파일들이며, 세 번째 열은 실제 프로그램을 수행했을 때의 동적으로 수행된 명령어 수를 $M(10^6)$단위로 나타내었다. [표 2]는 베이스라인 구조에서 사용된 실험 모델의 파라미터들을 요약해 놓은 것이다.

표 1 벤치마크 프로그램과 입력 데이타

Benchmark	Input set	Dynamic Instruction(10^6)
go	2stone9.in	100
m88ksim	tiny.in	100
vortex	vortex.tiny.in	65
li	queen6.lsp	42
gcc	jump.i	40
ijpeg	tinyrose.ppm	75
perl	scrabbl.in	40

표 2 베이스라인 구조의 구성

	인수	값	비고
Processor Core	RUU size	256	Instruction window
	LSQ size	256	Load-Store queue
	Fetch width	8/16 instr/cycle	8/16 issue baseline
	Decode width	8/16 instr/cycle	8/16 issue baseline
	Issue width	8/16 instr/cycle	8/16 issue baseline
	Commit width	8/16 instr/cycle	8/16 issue baseline
	Functional units	10/20 int alu(1) 4/8 fp add(2) 2/4 int mult(3) div(12) 2/4 fp mult(4) div(12)	8/16 issue baseline ( ) is latency
Branch Predictor	2lev	1K entry 8 history size	
Cache	L1 D- cache	512K, 2-way, 32 byte block	Data cache
	L1 I- cache	3-cycles latency	Instruction cache
	L2 cache	512K, direct map, 32 byte blocks unified, 1M, 4-way, 64blocks	Secondary cache

5. 실험 결과

이 장에서는 8-이슈폭과 16-이슈폭을 갖는 머신 모델에서 모험적 갱신을 수행하지 않는 하이브리드 예측기와 모험적 갱신을 수행하는 제안한 하이브리드 예측기를 사용하여 명령어의 결과값을 예상하였을 경우에 예상 정확도와 성능향상을 비교, 분석한다.

5.1 8-이슈폭에서 실험결과

[그림 9]는 8-이슈폭에서 8K 엔트리를 갖는 VHT를 사용하여 명령어 예상 시 모험적으로 예상 테이블을 갱신하지 않은 하이브리드 예측기(NS_Hybrid: Non-Speculative Hybrid predictor)와 모험적으로 예상 테이블을 갱신하는 제안된 하이브리드 예측기(S_Hybrid: Speculative Hybrid predictor)에 대한 실험결과를 보여준다.

Table miss는 예측기의 테이블에 해당 엔트리가 할당되지 않아서 예상을 수행하지 못한 경우를 나타내고, Not predicton은 엔트리가 존재하더라도 신뢰성 카운터 값이 임계치보다 작거나(LVP 또는 TVP의 경우) 상태 필드의 값이 예상을 수행할 수 없는 상태(SVP의 경우)를 나타내서 예상을 수행하지 못한 명령어 비율을 나타낸다. Correct prediction은 예측기의 테이블에 할당된 엔트리가 존재하고, 해당 엔트리의 포화 카운터 값이 임계치보다 크거나(LVP 또는 TVP의 경우) 상태 필드가 예상 가능한 상태(SVP의 경우)를 나타내서 예상이 이루어진 명령어 중 정확한 예상이 이루어진 비율을 나타낸다. Incorrect prediction은 예상이 이루어진 명령어 중에서 예상이 틀린 비율을 나타낸 것이다.

[그림 10]은 8-이슈폭에서 16K 엔트리의 VHT를 갖는 환경에서 NS_hybrid 와 S_hybrid 예측기에 대한 실험결과를 보여주고 있다.

8K 엔트리와 16K 엔트리의 실험결과에서 보듯이 기존의 하이브리드 결과값 예측기보다 모험적 갱신을 수행하는 제안된 하이브리드 결과값 예측기의 Incorrect Prediction 비율이 감소된 것을 보여주고 있다. 이것은 misprediction 페널티를 줄임으로서 프로세서의 성능을 향상시킬 수 있음을 나타낸다.

[그림 11]과 [그림 12]는 8-이슈폭을 갖는 8K엔트리와 16K엔트리의 VHT를 갖는 경우에 대한 각각의 예상 정확도를 보여 주고 있다. Correct Prediction은 예상을 수행한 명령어 중에서 정확하게 예상된 비율이고, Incorrect Prediction은 예상이 이루어진 명령어 중에서 예상이 틀린 비율이다. 예상 정확도는 8K 엔트리에서 평균 59%에서 68%로 향상되었고, 16K 엔트리에서 평균 59%에서 69%로 향상되었다.

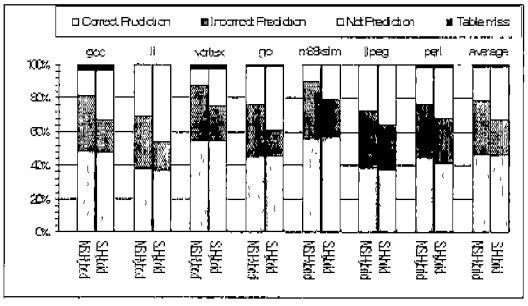

그림 9 8-이슈폭 16K 엔트리 실험 결과

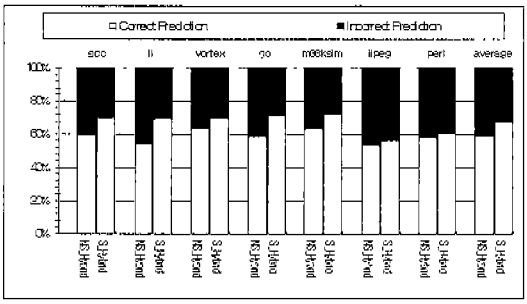

그림 11 8-이슈폭 8K 엔트리 예상 정확도

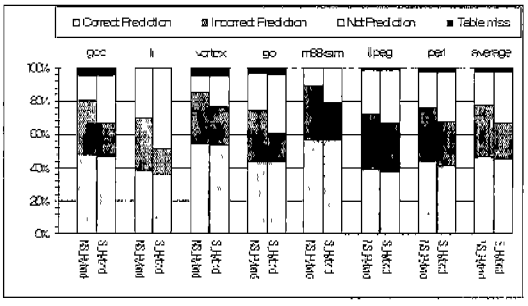

그림 10 8-이슈폭 8K 엔트리 실험 결과

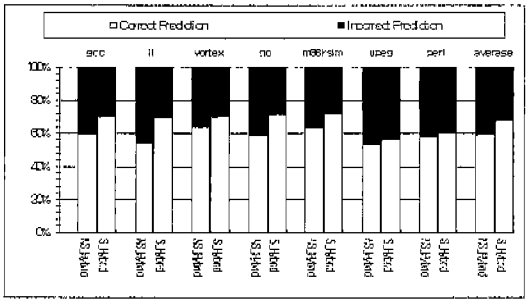

그림 12 8-이슈폭 16K 엔트리 예상 정확도

5.2 16-이슈폭에서 실험결과

[그림 13]과 [그림 14]는 16-이슈폭을 갖는 8K 엔트리의 VHT와 16K 엔트리의 VHT에 대한 각각의 실험 결과를 보여 주고 있다. 8-이슈폭에서와 마찬가지로 제안된 모험적 갱신을 수행하는 하이브리드 결과값 예측기에서 잘못된 예상을 하는 Incorrect Prediction이 감소하여 전체적인 성능은 향상되었다.

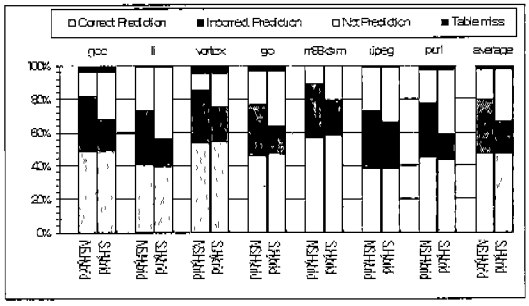

그림 13 16-이슈폭 8K 엔트리 실험 결과

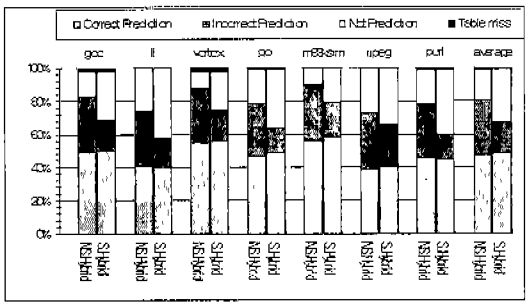

그림 14 16-이슈폭 16K 엔트리 실험 결과

[그림 15]와 [그림 16]은 16-이슈폭을 갖는 8K 엔트리와 16K 엔트리에 대한 각각의 예상 정확도를 보여 주고 있다. NS_Hybrid와 제안된 S_Hybrid 예측기에

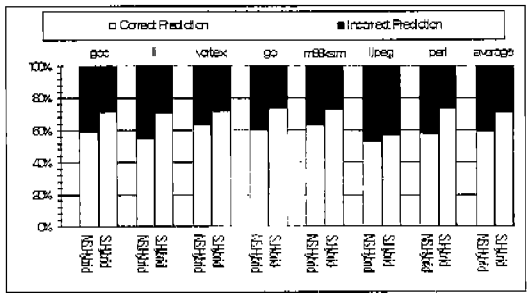

그림 15 16-이슈폭 8K 엔트리 예상 정확도

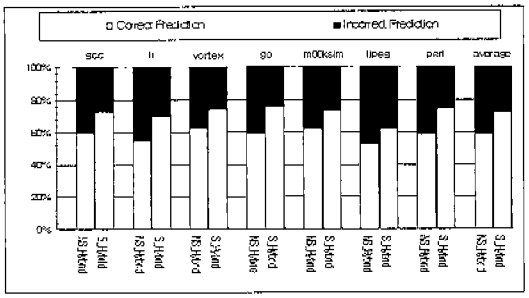

그림 16 16-이슈폭 16K 엔트리 예상 정확도

대한 예상정확도는 평균 8K 엔트리를 갖는 예측기에서 59%에서 71%로, 16K 엔트리를 갖는 예측기에서는 59%에서 72%로 향상되었다.

5.3 성능비교

모험적 갱신을 수행하지 않은 결과값 예측기와 제안한 결과값 예측기에 대한 성능비교를 [그림 17]과 [그림 18]에서 보여주고 있다. 실험 결과 제안한 예측기는 베이스라인 구조보다 8-이슈폭에서 성능향상이 평균 6.2%, 16-이슈폭에서 성능향상이 5.4% 증가하였고 모험적 갱신을 수행하지 않은 결과값 예측기 구조보다 8-이슈폭에서 2%, 16-이슈폭에서는 1.9% 성능이 증가하였다.

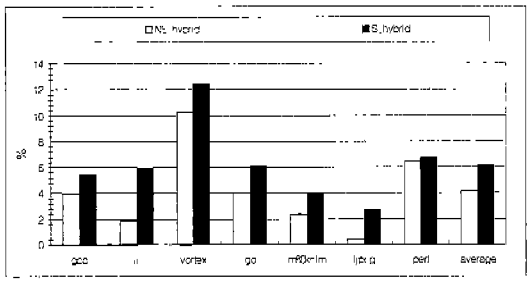

그림 17 8-이슈폭 16K 엔트리에서의 성능향상

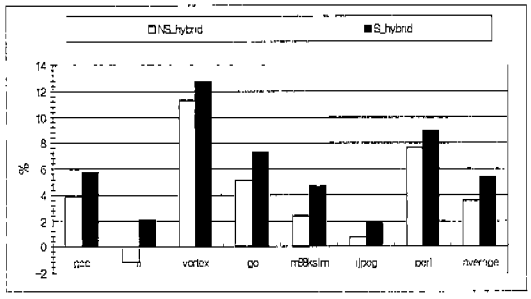

그림 18 16-이슈폭 16K 엔트리에서의 성능향상

잘못 예상된 명령어들은 올바른 결과값으로 다시 수행되어야 한다. 이렇게 잘못 예상된 명령어들을 복구하기 위해서는 많은 패널티가 요구된다. 본 논문에서 제안한 모험적 갱신을 수행하는 하이브리드 결과값 예측기는 예상 정확도가 높을 뿐만 아니라, 예상실패 비율이 모험적 갱신을 수행하지 않는 하이브리드 결과값 예측기보다 감소하여 예상실패 패널티를 줄임으로서 전체적인 프로세서의 성능향상을 보이고 있다.

6. 결 론

본 논문에서는 슈퍼스칼라 프로세서에서 비순서적(out-of-order)으로 명령어가 이슈 및 수행될 때 명령어를 예상한 후 명령어가 완료되어 예상테이블을 수정하기 전에 다시 그 명령어를 예상 시 부적절한 데이타 사용으로 인한 예상 실패를 방지하는 하이브리드 결과값 예상 메커니즘을 제안하였다. 제안한 예측기는 최근 결과값 예측기, 스트라이드 결과값 예측기, 2-단계 결과값 예측기를 하이브리드 시킨 예측기에 명령어 반입 시 예상과 예측기의 갱신을 동시에 수행함으로서 부적절한 데이타 사용을 줄여 예상 정확도를 향상시킬 수 있었다. 실험 결과 SPECint95 벤치마크 프로그램에서 명령어를 예상 후 결과값이 구해지기 전에 그 명령어를 다시 예상하는 비율이 8-이슈폭에서는 평균 19%, 16-이슈폭에서는 평균 28%로서, 이는 제안된 예측기를 사용하면 예상 정확도가 향상될 것을 증명해 주었다.

제안된 예측기의 예상 정확도는 모험적 갱신을 하지 않은 하이브리드 예측기보다 8-이슈폭에서 평균 59%에서 69%로, 16-이슈폭에서는 평균 59%에서 72%로 증가하였고, 잘못된 예상비율은 8-이슈폭에서 평균 41%에서 31%로, 16-이슈에서는 41%에서 28%로 감소하였다. 본 논문에서 제안된 모험적 갱신을 허용하는 메커니즘은 기존에 제안된 대부분의 하이브리드 결과값 예측기에서도 사용할 수 있음으로 기존의 하이브리드 결과값 예측기에 적용 시 예상 정확도와 성능향상에 기여 할 것이라 기대된다. 향후 연구과제로 반입되는 명령어에 적합한 예측기를 선택하여 예측기의 전체 하드웨어 비용을 줄일 수 있는 정적, 동적 분류 방법을 개발하고 제안된 기법에 적용시킴으로써 예상 정확도를 더욱 향상시키도록 하는 연구가 필요하다.

참 고 문 헌

[1] M. H. Lipasti, C. B. Wilderson, J. P. Shen, *"Value Locality and Load Vaule Prediction,"* Proc. of the 7th Intl. Conference on Architectural Support for Programming Languages and Operating Systems, pp.138-147, October 1996.

[2] M. H. Lipasti and J. P. Shen, *"Exceeding the Dataflow limit via Value Prediction,"* Proc. of the 29th Intl. Symp. on Microarchitecture, pp.226-237, December 1996.

[3] Y. Sazeides and J. E. Smith, *"The Predictability of Data Values,"* Proc. of the 29th Intl. Symp. on Microarchitecture, pp.226-237, December 1996.

[4] F. Gabbay and A. Mendelson, *"Can Program Profiling Support Value prediction?,"* Proc. of the 30th Intl. Symp. on Microarchitecture, pp.270-280, December 1997.

[5] T-Y Yeh and Y. N. Patt, *"Alternative Implementations of Two-Level Adaptive Branch Prediction,"* Proc. of the 19th Intl. Symposium on Computer Architecture, pp.124-134, 1992.

[6] K. Wang and M. Franklin, *"Highly Accurate Data Value Prediction using Hybrid Predictors,"* Proc. of the 30th International Symp. on Microarchitecture, pp.281-290, December 1997.

[7] J. Gonzalez and A. Gonzalez, *"The potential of data value speculation to boost ilp,"* in 12th International Conference on Supercomputing, 1998

[8] G. Reinman and B. Calder, *"Predictive techniques for aggressive load speculation,"* Proc. of the 31th International Symp. on Microarchitecture, 1998

[9] T. Nakra, R. Gupta and M.L. Soffa, *"Global Context-Based Value Prediction"*, Proc. of the 5th Intl. Symposium on High Performance Computer Architecture, January 1999.

[10] D.C. Burger and T.M. Austin, *"The simplescalar tool set, version 2.0"*, Technical Report CS-TR-97-1342, University of wisconsin, Madison, June 1997.

[11] F. Dahlgren, F. Dahlgren and P. Stenstrom, *"Evaluation of Hardware-Based Stride and Sequential Prefetching in Shared-Memory Multiprocessors,"* IEEE Transactions on Parallel and Distributed Systems. vol. 7. no. 4. pp. 385-398. April 1996.

[12] B. Calder, G. Reinman, D. M. Tullsen, *"Selective Value Prediction,"* Proc. of the 26th Intl. Symposium on Computer Architecture, May 1999.

[13] S. J. Lee, P. C. Yew, *"On Some Implementation Issues for Value Prediction on Wide-Issue ILP Processors"*, Intl. Conference on Parallel Architectures and Computer Technique, 2000.

박 홍 준

1984년 아주대학교 전자계산학과 공학사. 1987년 한양대학교 전자계산학과 공학석사. 1997년 ~ 현재 수원대학교 전자계산학과 박사과정. 1994년 ~ 현재 극동정보대학 전산정보처리과 조교수. 관심분야는 ILP 프로세서의 분기예상 메커니즘 및 결과값 예상 메커니즘, ILP 프로세서의 정적 및 동적 명령어 스케줄링 등

신 영 호

1999년 수원대학교 전자계산학과 이학사. 2001년 수원대학교 컴퓨터과학과 석사. 현재 (주)에어미디어 기술연구소 연구원. 관심분야는 ILP 프로세서의 분기예상 메커니즘 및 결과값 예상 메커니즘, ILP 프로세서의 정적 및 동적 명령어 스케줄링, 네트웍 프로그래밍 등

조 영 일

1980년 한양대학교 전자공학과 공학사. 1982년 한양대학교 전자공학과 공학석사. 1985년 한양대학교 전자공학과 공학박사. 1986년 3월 ~ 현재 수원대학교 컴퓨터과학과 부교수. 관심분야는 최적화 컴파일러 설계, ILP 프로세서의 분기예상 메커니즘 및 결과값 예상 메커니즘, ILP프로세서의 정적 및 동적 명령어 스케줄링, Internet real-time and multimedia services and protocols 등