슈퍼스칼라 프로세서에서 예상 테이블의 모험적 갱신과 명령어 실행 유형의 정적 분류를 이용한 혼합형 결과값 예측기

(A Hybrid Value Predictor using Speculative Update of the Predictor Table and Static Classification for the Pattern of Executed Instructions in Superscalar Processors)

박 홍 준 † 조 영 일 ††
(Hong Jun Park) (Young Il Jo)

요 약 데이타 종속성을 제거하기 위해서 명령어의 결과값을 예상하는 여러 결과값 예측기의 장점을 이용하여 높은 성능을 얻을 수 있는 새로운 혼합형 예측 메커니즘을 제안한다. 제안된 혼합형 결과값 예측기는 예상 테이블을 모험적으로 갱신할 수 있기 때문에 부적절한(stale) 데이타로 인해 잘못 예상되는 명령어의 수를 효과적으로 감소시킨다. 또한 정적 분류 정보를 사용하여 명령의 반입시 적절한 예측기에 할당함으로써 예상 정확도를 더욱 향상시키며, 하드웨어 비용을 효율적으로 감소시키도록 하였다. 5개의 SPECint 95 벤치마크 프로그램에 대해 SimpleScalar/PISA 3.0 툴셋을 사용하여 실험하였다. 16-이슈 폭에서 모험적 갱신을 사용한 평균 예상 정확도는 73%의 실험 결과가 나왔으며, 정적 분류 정보를 사용하였을 경우 예상 정확도가 88%로 증가된 결과를 얻었다.

키워드 : 명령레벨 병렬성, 결과값 예측 방법, 결과값 예측기, 혼합형 결과값 예측, 예상 정확도

Abstract We propose a new hybrid value predictor which achieves high performance by combining several predictors. Because the proposed hybrid value predictor can update the prediction table speculatively, it efficiently reduces the number of mispredicted instructions due to stale data. Also, the proposed predictor can enhance the prediction accuracy and efficiently decrease the hardware cost of predictor, because it allocates instructions into the best-suited predictor during instruction fetch stage by using the information of static classification which is obtained from the profile-based compiler implementation. For the 16-issue superscalar processors, simulation results based on the SimpleScalar/PISA tool set show that we achieve the average prediction rates of 73% by using speculative update and the average prediction rates of 88% by adding static classification for the SPECint95 benchmark programs.

Key words : Instruction-level parallelism, ILP, data value prediction, data value predictor, hybrid value prediction, prediction accuracy

1. 서 론

최근 개발되고 있는 고성능 슈퍼스칼라 프로세서는 여러 개의 실행 장치를 사용하여 다수의 명령어를 동시에 이슈하고 병렬로 처리함으로서 프로세서의 성능을 향상할 수 있게 설계되고 있다. 이러한 고성능 프로세서에서 성능을 향상시키기 위해서는 높은 명령어 이슈율과 명령어 수준 병렬성(ILP: Instruction Level Parallelism)을 가능한 최대로 이용하는 것이 중요하다[2]. 그러나 명령어간의 종속관계는 ILP를 저해시키는 주요 장애요소가 되며, 선행 명령어의 결과값을 후행 명령어가 입력으로 사용하는 데이타 종속관계의 명령어는 선행 명령어의 결과가 구해질 때까지 지연되어야 한다. 이러한 문

† 정 회 원 : 곡동정보대학 전산정보처리과 교수
hjpark@cs.kdc.ac.kr
†† 정 회 원 : 수원대학교 컴퓨터과학과 교수
ycho@mail.suwon.ac.kr
논문접수 : 2001년 5월 11일
심사완료 : 2001년 8월 7일

제를 해결하기 위해서 결과값 예상기법(data value prediction)을 사용하게 된다. 결과값 예상기법은 데이타 종속적인 명령어가 선행 명령어의 결과값이 나올 때까지 기다리는 것이 아니라, 선행 명령어의 결과를 미리 예측하여 모험적으로 실행(speculative execution)함으로써 데이타 종속 연결관계를 제거하는 하드웨어 메커니즘이다[1,2,3,4,5,6,7,8].

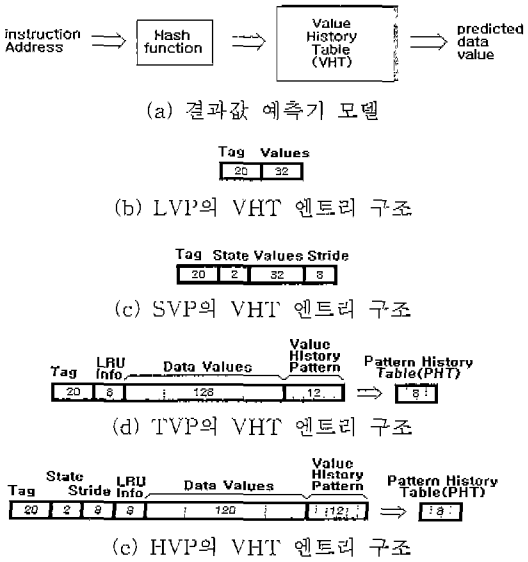
결과값 예상기법 중 명령어의 결과를 바로 이전에 수행된 결과로 예상하는 최근 결과값 예측기(Last Value Predictor)[1,2]는 마지막으로 수행된 결과값만을 예상 테이블에 저장하므로 적은 하드웨어를 사용하고 구현이 쉽다는 장점이 있으나, 수행 결과가 변하지 않는 경우에만 사용할 수 있다는 단점이 있다. 명령어의 결과가 마지막으로 수행된 결과값에 일정한 값만큼 변한다는 사실을 이용하여 명령어의 다음 결과를 예상하는 스트라이드 결과값 예측기(Stride Value Predictor)[6,7]는 반복문의 카운터 변수나 결과값이 일정하게 증감하는 명령어를 갖는 프로그램에서 좋은 성능을 발휘하게 된다. 또한 2-단계 결과값 예측기(Two-level Value Predictor)[6]는 이전에 실행된 n개의 결과값 중에 하나를 다음 값으로 예상하는 방법으로, 높은 예상 정확도를 가지나 n개의 결과값을 저장함으로서 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 혼합형 결과값 예측기(Hybrid Value Predictor)가 제안되었다[6,8].

고성능 프로세서에서는 성능향상을 위해 명령어의 반입폭과 이슈폭이 계속해서 증가하고 있다. 이로 인해 명령어가 예상된 후 실제 결과값으로 예상 테이블을 수정하기 전에 다시 그 명령어를 예상하는 경우에는 부적절한 데이타의 사용이 증가하게 된다. 부적절한 데이타 사용은 올바르지 못한 예상을 발생시키게 되며, 프로세서의 성능을 저하시킨다. 특히 스트라이드 결과값 예측기와 2-단계 결과값 예측기에서는 이러한 예상 실패로 인해 성능에 미치는 영향이 심각하다.

본 논문에서는 명령어의 결과값 예상 시 부적절한 데이타를 사용함으로서 발생되는 예상 실패를 최소화하기 위해 명령어의 결과값을 예상하는 동시에 예상 테이블을 모험적으로 갱신하는 혼합형 결과값 예측기를 제안한다. 결과값이 구해진 후 예상 테이블을 갱신하기 전에 동일 명령어의 값을 다시 예상할 때, 부적절한 데이타의 사용을 방지함으로써 예상 정확도를 향상시킬 수 있고, 부적절한 데이타로 인해 잘못 예상되는 명령어의 수와 잘못예상 시의 페널티를 감소시켜 프로세서의 성능을 향상시킨다. 또한 제안한 혼합형 결과값 예측기는 프로파일링으로 얻어진 명령어의 실행 형태에 따른 정적 분류(static classification) 정보를 사용함으로써 명령어 반입 시 명령어를 적절한 예측기에 할당함으로써 예상 정확도를 더욱 향상시킬 수 있고, 하드웨어 비용을 효율적으로 감소시키도록 구성하였다.

2. 관련 연구

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



(a) 결과값 예측기 모델

(b) LVP의 VHT 엔트리 구조

(c) SVP의 VHT 엔트리 구조

(d) TVP의 VHT 엔트리 구조

(e) HVP의 VHT 엔트리 구조

그림 1 결과값 예측기의 VHT 엔트리 구조

2.1 최근 결과값 예측기(LVP: Last Value Predictor)

최근 결과값 예측기는 명령어의 최근 수행된 결과값을 예상 테이블(VHT: Value History Table)에 저장하고 다음에 동일 명령어를 반입할 때, 저장된 결과값을 예상값으로 사용하는 방법이다[1,2].

[그림 1-(b)]는 최근 결과값 예측기의 VHT 엔트리 구조를 보여주고 있다. 예측되는 명령어의 주소로 인덱스되는 VHT의 각 엔트리는 명령어 주소의 상위 비트 일부를 저장하는 태그(Tag) 필드와 마지막으로 수행된 명령어의 결과값을 저장하는 결과값(Value) 필드로 구성된다.

최근 결과값 예측기는 VHT의 각 엔트리에 마지막 수행 결과값 하나만을 저장하기 때문에 적은 하드웨어 비용을 요구하지만 변하지 않는 값(상수 값)을 생성하는 명령어들만 예상할 수 있어서 예상 정확도가 40%~50%로 낮다는 단점을 갖는다.

2.2 스트라이드 결과값 예측기(SVP: Stride Value Predictor)

스트라이드 결과값 예측기[6]는 명령어의 마지막 수행된 결과값과 마지막 두 번의 수행된 결과값의 차이값을 VHT에 저장하여 동일 명령어의 반입 시 마지막 수행 결과값과 차이값의 합으로 예상한다.

[그림 1-(c)]는 스트라이드 결과값 예측기의 VHT 엔트리 구조를 보여주고 있다. VHT의 각 엔트리는 태그(Tag) 필드, 상태(State) 필드, 결과값(Value) 필드, 차이값(Stride) 필드로 구성된다. 상태 필드는 2비트로 구성되며 3개의 상태(Init, Transient, Steady)를 가진다. 상태 필드가 Init('0') 혹은 Transient('1')를 나타내는 경우는 예상을 위해 준비중인 단계로 예상을 수행하지 않는다. 상태 필드의 값이 Steady('2')를 나타내는 경우에만 결과값 필드와 차이값 필드의 합을 예상값으로 제공한다.

스트라이드 결과값 예측기는 일정한 값을 갖는 패턴(zero stride)과 일정하게 증감하는 패턴 모두를 예상할 수 있어서 최근 결과값 예측기보다 좋은 성능을 가진다.

2.3 2-단계 결과값 예측기(TVP: Two-level Value Predictor)

2-단계 결과값 예측기[5]는 명령어가 수행한 마지막 네 개의 결과값을 VHT에 저장하여 결과값을 예상하는 방법이다.

[그림 1-(d)]는 2-단계 결과값 예측기의 VHT 엔트리 구조를 보여주고 있다. VHT의 각 엔트리는 태그(Tag) 필드, LRU(LRU_Info) 필드, 네 개의 마지막 결과값(Data Values) 필드, VHP(Value History Pattern) 필드로 구성되어 있다. 결과값 필드는 최근에 사용된 결과값들 중에서 중복되지 않은 서로 다른 네 개의 결과값을 저장한다. LRU 필드는 결과값 필드에 저장된 값들의 사용 순서 정보를 가지고 있으며, VHP 필드는 마지막 6번의 결과값이 나타난 순서 위치를 저장한다. VHP로 인덱스되는 PHT(Pattern History Table)는 4개의 2비트 카운터 필드를 사용하며, 이 카운터 필드 중에서 최대값을 선택하고 이 값이 임계값보다 크면 이 카운터 위치에 대응되는 위치의 결과값을 예상 값으로 사용하게 된다.

2-단계 결과값 예측기는 높은 예상 정확도를 갖지만 네 개의 결과값을 저장하기 때문에 많은 하드웨어 비용을 요구하는 단점을 갖는다.

2.4 혼합형 결과값 예측기(Hybrid Value Predictor)

Wang 등의 혼합형 결과값 예측기[6]는 스트라이드 결과값 예측기와 2-단계 결과값 예측기를 결합한 형태의 예측기로, 신뢰성 카운터(Confidence Counter)에 의해 데이타 값을 예측하는 방법이다. 신뢰성 카운터가 임계값보다 크면 2-단계 예상 방법으로, 임계값보다 작으면 스트라이드 예상 방법으로 예상 값을 선택하게 된다. 또한, 명령어가 예측기의 두 엔트리에 있다면 신뢰성 카운터의 값이 높은 엔트리의 예상 값을 선택하게 된다. [그림 1-(e)]는 혼합형 결과값 예측기의 VHT 구조를 보여주고 있다.

Wang의 혼합형 결과값 예상방법은 스트라이드 특성을 갖는 명령어와 규칙적인 패턴을 갖는 명령어를 모두 예상하기 때문에 높은 예상 정확도를 얻을 수 있다. 그러나, 각 엔트리에 스트라이드 패턴을 예상하기 위한 필드와 2-단계 패턴을 예상하기 위한 필드를 포함하여 많은 하드웨어 비용을 요구하고, 다른 엔트리에 같은 데이타 값이 중복 저장된다는 단점을 갖는다.

3. 제안된 혼합형 결과값 예측기

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

3.1 모험적 갱신의 필요성

명령어의 반입폭과 이슈폭이 증가함에 따라 명령어가 예상된 후 실제 결과값으로 예상 테이블을 수정하기 전에 다시 그 명령어를 예상하는 경우가 증가하고 있다. 이런 경우 부적절한 데이타를 사용하게 되어 올바르지 못한 예상을 발생시키게 되며 프로세서의 성능을 저하시키게 된다.

[그림 2]는 8-이슈폭과 16-이슈폭을 갖는 프로세서에서 모험적으로 테이블을 갱신하지 않는 2-단계 결과값 예측기를 사용하여 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 명령어의 비율을 나타낸다. "age=2"는 명령어가 예상된 후 결과가 구해지기 전에 그 명령어를 두 번 예상하는 비율을 나타내고, "age>=3"은 결과가 구해지기 전에 그 명령어를 세 번 이상 다시 예상하는 비율을 나타낸다.

실험 결과를 보면 전체 예상 명령어 중 부적절한 데이타로 예상되는 비율이 8-이슈 폭에서 평균 19%, 16-

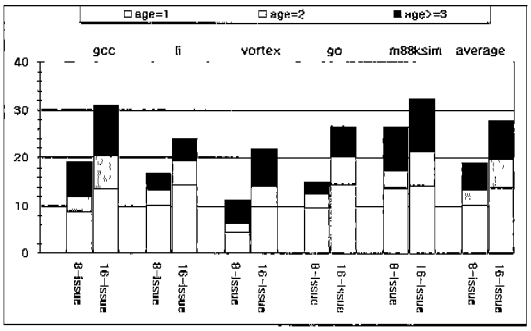


그림 2 2-단계 결과값 예측기에서 부적절한 데이타 사용 비율

이슈 폭에서 평균 28%로 나타난다. 이러한 결과는 이슈 폭이 증가함에 따라 부적절한 데이타를 사용하는 비율이 증가한다는 것을 알 수 있으며, 프로세서의 성능을 향상시키기 위해서는 명령어 반입 시 결과값을 예상할 때 예상 테이블을 모험적으로 갱신해야 하는 필요성을 입증하고 있다.

3.2 명령어의 분류

Wang 등의 혼합형 예측기[6]는 높은 예상 정확도를 갖으나, 예측기의 엔트리에 예상할 명령어를 중복하여 할당한다는 문제점을 갖고 있다. 명령어의 실행 패턴을 분석[17]할 수 있다면 가장 적절한 예측기를 선택하여 할당함으로써 예측기의 엔트리에 중복 할당되는 문제점을 해결할 수 있다.

[그림 3]은 각 벤치마크 프로그램에서 프로파일링을 수행함으로써 최근 결과값과 스트라이드 결과값을 사용하는 명령어들의 실행 패턴 분포를 분석한 결과이다. 명령어들이 생성하는 결과값들을 분석하여 전체 실행 결과값 중 최근 결과값(last) 또는 스트라이드(st)의 실행 패턴이 나타나는 비율을 x축에 나타내었고, y축은 전체 실행 명령어 중 최근 결과값 또는 스트라이드 실행 패턴으로 실행되는 명령어의 비율을 나타낸다. 예를 들어, vortex 벤치마크 프로그램의 경우 명령어의 실행 결과값 중 67%의 결과값이 최근 결과값 유형(vortex_last)으로 실행되는 명령어는 전체 명령어 중 6.3%이고, 67%의 결과값이 스트라이드 유형(vortex_st)으로 실행되는 명령은 전체 명령어 중 0.07%가 된다.

명령어의 유형별 분포를 살펴보면 최근 결과값 실행 패턴을 사용하는 명령어는 전체 명령어 중 평균 57.9%, 스트라이드 결과값 실행 패턴을 사용하는 명령어는 평

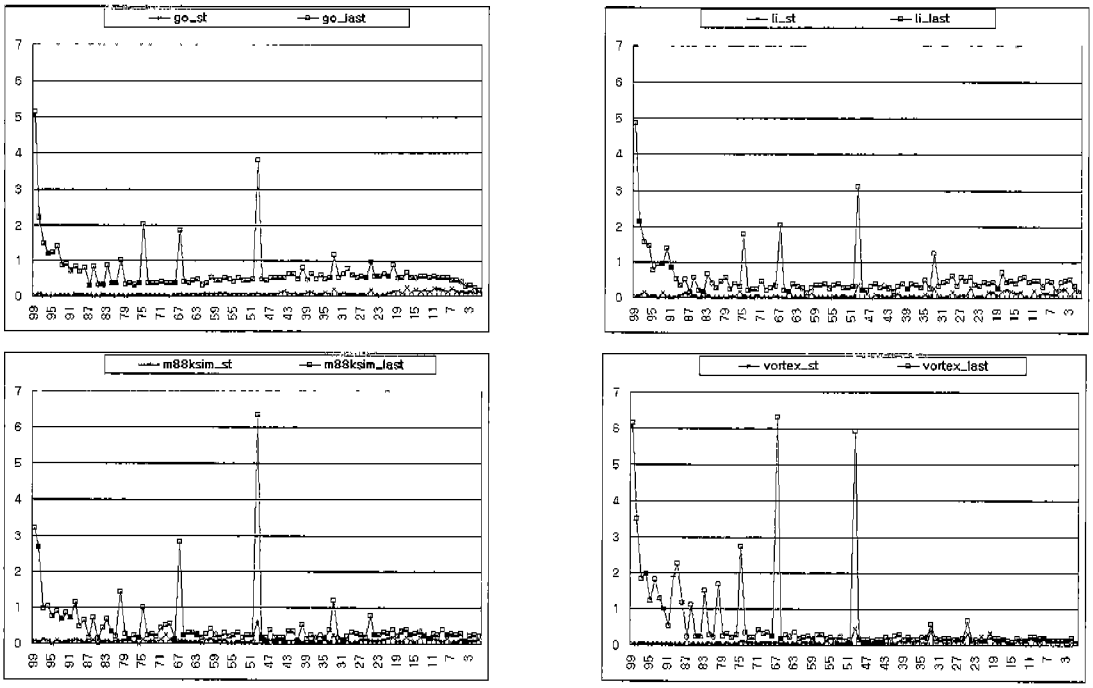


그림 3 벤치마크 프로그램에서의 명령어 실행 패턴의 분포

균 7.7%, 실행 패턴 유형을 판단할 수 없는 명령어들이 전체 실행 명령어 중 평균 34.4%를 차지한다. 실행 패턴 유형을 판단할 수 없는 명령어들 중에는 한번 수행 등으로 예상할 필요가 없는 명령어가 평균 20% 포함되어 있다. 이러한 분석 결과를 이용하면 각 명령어의 반입시 적절한 예측기에만 할당하여 예상을 수행할 수 있으며, 예측이 불가능한 명령어는 예상을 수행하지 않으므로써 전체 성능을 향상시킬 수 있다.

3.3 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 혼합형 결과값 예측기를 제안한다.

제안된 혼합형 결과값 예측기는 [그림 4]와 같이 최근 결과값 예측기(LVP), 스트라이드 결과값 예측기(SVP), 2-단계 결과값 예측기(TVP)로 구성된 혼합형 예측기로, 프로파일링을 통한 정적 분류 정보를 사용하여 명령어 반입시 적절한 예측기에만 명령어를 할당하도록 하였다. 또한, 명령어 반입시 모험적 갱신을 수행하도록 스트라이드 결과값 예측기의 VHT 엔트리에는 S_age 카운터 필드를 추가하였으며, 2-단계 결과값 예측기의 VHT 엔트리에는 T_age 카운터 필드와 S_VHP(Speculative VHP) 필드를 추가하였다.

두 개의 age 카운터의 초기값은 '0'으로 설정하고, 명령어 반입 시 예측기의 VHT를 조사(lookup)할 때 예상된 예측기의 age 카운터 값을 '1'씩 증가시키고 재기록 때 '1'씩 감소시킨다. 2-단계 결과값 예측기에 추가된 S_VHP 필드는 6 비트로 구성되며, 모험적 갱신 시 예상 값으로 사용된 결과값 필드 위치를 저장한다.

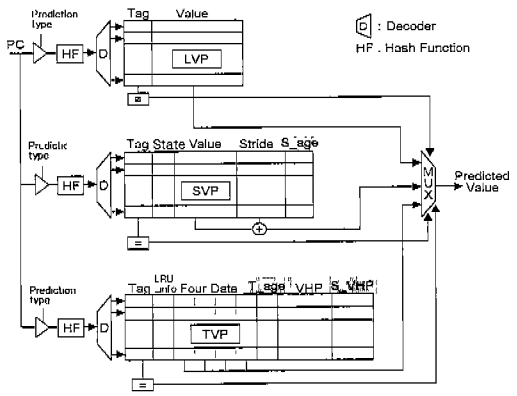


그림 4 제안된 혼합형 결과값 예측기의 구조

3.4 예측기의 예상 메커니즘

3.4.1 명령어의 분류 정보 사용

명령어 반입시 적절한 예측기를 선택하기 위해 [표 1]과 같이 명령어의 실행 패턴에 따라 "Last" 유형, "Stride" 유형, "Not Prediction" 유형, "Unknown" 유형으로 분류한 정적 분류 정보를 사용하였다.

표 1 명령어의 실행 패턴 정보와 선택 예측기

실행 유형	할당 명령어	선택 예측기
Last	최근값 실행 패턴을 갖는 명령어	최근 결과값 예측기
Stride	스트라이드 실행 패턴을 갖는 명령어	스트라이드 결과값 예측기
Unknown	혼합된 패턴을 갖는 명령어	2단계 결과값 예측기
Not Prediction	한 번 수행으로 예상하지 않은 명령어	예상하지 않음

효율적인 분류 정보를 얻기 위해 실행 유형을 결정하는 실행 패턴 비율을 변화시키며 8K VHT 엔트리의 제안한 혼합형 예측기에서 예상 정확도를 실험하였다. [표 2]는 li 벤치마크 프로그램에서 실행 패턴 비율의 변화에 따라 측정된 명령어 유형의 분포와 예상 정확도이다.

표 2 실행 패턴 비율에 따른 실행 유형 분포와 예상 정확도

실행 패턴 비율	유형별 명령어 비율			예상 정확도
	Last 유형	Stride 유형	Unknown 유형	
40% 이상	48.4%	3.1%	21.7%	86.17%
50% 이상	45.5%	2.1%	25.6%	88.64%
60% 이상	39.3%	1.8%	32.2%	87.75%

실행 패턴의 비율을 50% 이상으로 하여 명령어의 실행 유형을 결정한 경우의 예상 정확도가 각각 40% 이상과 60% 이상인 경우의 86.17%와 87.75%에 비해 88.64%로 가장 높게 측정되었다. 이 실험 결과는 40% 이상의 경우에는 낮은 실행 패턴 비율을 갖는 명령어의 증가로, 60% 이상의 경우는 "Unknown" 유형의 증가로 잘못된 예상이 증가하기 때문이다. 따라서 본 논문에서는 명령어가 생성하는 전체 실행 결과값 중 50% 이상의 최근 결과값 실행 패턴을 사용하는 명령어를 "Last" 유형으로, 50% 이상의

스트라이드 결과값 실행 패턴을 사용하는 명령어를 "Stride" 유형으로 분류하고, "Not Prediction" 유형을 제외한 나머지 명령어를 "Unknown" 유형으로 분류한 정보를 사용하여 예측기를 선택하도록 하였다.

3.4.2 모험적 갱신 메커니즘

반입된 명령어가 2단계 결과값 예측기를 조사할 때 T_age 카운터가 '1'이상이면 이전에 반입된 명령어가 갱신되기 전에 조사한 경우로, 부적절한 결과값을 사용하게 되는 경우이다. [그림 5]와 같이 T_age 카운터의 값에 따라 올바른 VHP를 사용하여 PHT를 인덱스 하도록 하면 부적절한 결과값의 사용을 방지할 수 있다. 즉 T_age 카운터가 '0'일 경우는 예상한 후 실제 결과값으로 예상 테이블이 갱신된 다음 명령어가 반입된 경우이므로 $P_0 \sim P_5$의 VHP 값을 사용하여 PHT를 인덱스하고, 예상된 결과값의 위치를 추가된 S_VHP의 P_6에 저장한다. T_age 카운터가 '1'일 경우는 반입된 명령어가 한 번 수정되기 전에 명령어가 다시 반입된 경우이므로 $P_1 \sim P_6$의 VHP값을 사용하여 PHT를 인덱스하고, 예상된 값의 위치를 P_7에 저장한다. T_age 카운터가 '2'일 경우는 반입된 명령어가 두 번 수정되기 전에 명령어가 반입된 경우이므로 $P_2 \sim P_7$의 VHP값을 사용하여 PHT를 인덱스한다. 명령어 수행이 완료되면 재기록 사이클 동안 T_age는 '1'을 감소시키고, 나머지 필드는 기존의 2-단계 결과값 예측기에서와 같은 방법으로 갱신한다.

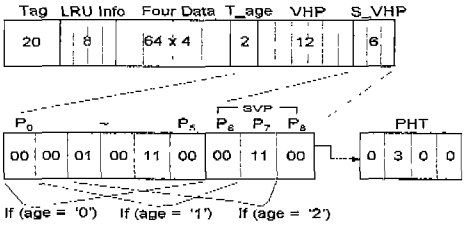


그림 5 제안된 예측기에서 TVP의 테이블 엔트리 구조

반입된 명령어가 스트라이드 결과값 예측기를 조사할 때는 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.

4. 실험 및 평가

4.1 실험 환경

제안된 모험적 갱신을 수행하는 정적 분류 혼합형 결과값 예측기와 모험적 갱신을 수행하지 않는 혼합형 결과값 예측기를 비교 측정하기 위해서 8-이슈와 16-이슈의 명령어 이슈 크기를 갖는 환경에서 실험하였다. 실험 모델은 높은 명령어 반입폭과 용량이 큰 명령어 윈도우를 가지고 있으며, 분기 예측을 위해 1K 엔트리를 갖는 2단계 예측기를 사용하였다.

실험에 사용된 혼합형 결과값 예측기의 PHT를 4K 엔트리로 고정하고, VHT 엔트리를 각각 8K 엔트리와 4K 엔트리로 변화시키며 예상 빈도(Prediction rate) 및 예상 정확도(Prediction accuracy)를 비교, 분석한다.

본 논문의 실험을 위해 사용된 벤치마크 프로그램은 SPECint 95 벤치마크 중 5개를 실험하였으며 시뮬레이터는 실행 구동 시뮬레이터인 SimpleScalar/ PISA 3.0 툴셋[10]을 사용하였다. [표 3]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 3 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

4.2 성능 평가

이 장에서는 8-이슈 폭과 16-이슈 폭을 갖는 머신 모델에서 모험적 갱신을 수행하는 제안된 정적 분류 혼합형 결과값 예측기의 예상 정확도를 살펴본다.

4.2.1 모험적 갱신 적용시 예상 정확도

[그림 6]은 8-이슈 폭과 16-이슈 폭을 갖는 각 모델에서 8K 엔트리 VHT를 사용하여 명령어 예상 시 모험적으로 예상 테이블을 갱신하지 않은 기존의 혼합형 예측기(NS_H_8: Non-Speculative 8-issue, NS_H_16: Non-Speculative 16-issue)와 모험적으로 예상 테이블을 갱신하는 제안된 혼합형 예측기(S_H_8: Speculative 8-issue, S_H_16: Speculative 16-issue)에 대한 예상 빈도를 보여준다.

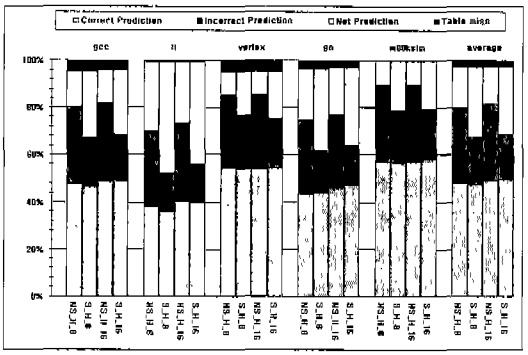


그림 6 8-이슈/16-이슈 폭 8K 엔트리의 예상 빈도

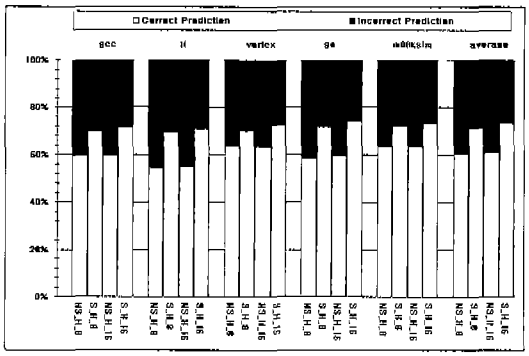


그림 7 8-이슈/16-이슈 폭 8K 엔트리의 예상 정확도

"Correct Prediction"은 예측기의 테이블에 할당된 엔트리가 존재하고, 해당 엔트리의 포화 카운터 값이 임계값보다 크거나(LVP 또는 TVP의 경우) 상태필드가 예상 가능한 상태(SVP의 경우)를 나타내서 예상이 이루어진 명령어 중 정확한 예상이 이루어진 비율을 나타낸다. "Incorrect Prediction"은 예상이 이루어진 명령어 중에서 예상이 틀린 비율을 나타낸 것이다. "Table miss"는 예측기의 테이블에 해당 엔트리가 할당되지 않아서 예상을 수행하지 않는 경우를 나타내고, "Not Predicton"은 엔트리가 존재하더라도 포화 카운터 값이 임계값보다 작거나(LVP 또는 TVP의 경우) 상태 필드의 값이 예상을 수행할 수 없는 상태(SVP의 경우)를 나타내서 예상을 수행하지 못한 명령어 비율을 나타낸다. 이 부분 역시 테이블 적중(table hit) 부분에 포함되는 부분이다.

실험결과에서 보듯이 8-이슈 폭과 16-이슈 폭 모두에서 기존의 혼합형 결과값 예측기보다 모험적 갱신을 수행하는 제안된 혼합형 결과값 예측기의 "Incorrect Prediction" 비율이 감소되고 "Not Prediction" 비율이 증가된 것을 보여주고 있다. 이 결과는 모험적 갱신을 수행함으로써 부적절한 데이타 사용으로 인한 잘못된 예상을 줄이게 되어 전체 프로세서의 성능을 향상시킴을 보여준다.

[그림 7]은 8-이슈 폭과 16-이슈 폭을 갖는 각 모델에서 8K 엔트리 VHT를 갖는 경우에 대해 각 벤치마크의 예상 정확도를 보여 주고 있다. "Correct Prediction"은 예상을 수행한 명령어 중에서 정확하게 예상된 비율이고, "Incorrect Prediction"은 예상이 이루어진 명령어 중에서 예상이 틀린 비율이다. 모험적 갱신을 사용함으로서 8-이슈 폭과 16-이슈 폭에서 각각 예상 정확도가 평균 61%에서 평균 71%, 평균 61%에서 평균 73%로 향상되었다.

4.2.2 분류 정보 적용시 예상 정확도

[그림 8]는 16-이슈 폭에서 모험적 갱신을 수행하는 제안된 혼합형 예측기에 정적 분류 정보를 사용했을 경우의 예상 빈도를 보여주고 있다.

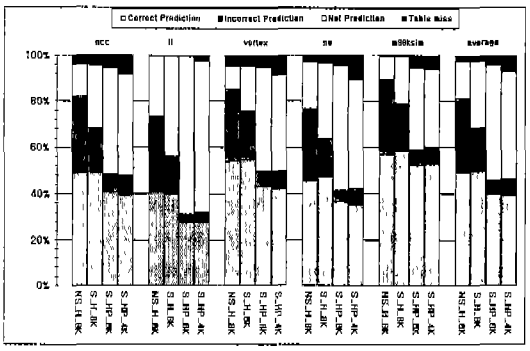


그림 8 16-이슈 폭에서 정적 분류 정보를 사용한 예상 빈도

8K 엔트리 VHT를 사용하여 모험적 갱신을 하며 정적 분류 정보를 사용한 혼합형 예측기 S_HP_8K는 "Correct Prediction"이 8K 엔트리 VHT를 사용하는 모험적 갱신 혼합형 예측기(S_H_8K)에 비해 감소하나, 정확하지 않은 예상은 수행하지 않기 때문에 "Not Prediction"이 상대적으로 크게 증가하게 된다. 또한 분류 정보에 따라 적절한 예측기에만 명령어를 할당함으로써 잘못된 예상을 하는 "Incorrect Prediction"이 높은 비율로 감소하여 전체적인 성능은 향상된다. 또한 VHT의 크기를 4K 엔트리로 축소한 S_HP_4K도 8K 엔트리의 S_HP_8K와 비슷한 성능을 보이고 있다.

[그림 9]는 16-이슈 폭에서 모험적 갱신을 수행하

는 제안된 혼합형 예측기에 정적 분류 정보를 적용시켰을 경우의 예상 정확도를 보여주고 있다.

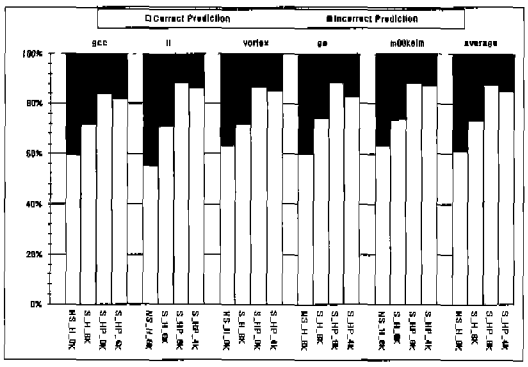


그림 9 16-이슈 폭에서 정적 분류 정보를 사용한 예상 정확도

모험적 갱신을 수행하는 혼합형 예측기에 정적 분류 정보를 사용함으로써 예상 정확도가 기존의 혼합형 예측기(NS_H_8K)의 평균 61% 예상 정확도와 모험적 갱신만을 수행하는 혼합형 예측기(S_H_8K)의 평균 73% 예상 정확도에 비해 평균 87%로 크게 향상되었다. 또한 VHT를 4K 엔트리로 하드웨어 비용을 크게 축소하여도 평균 85%로 비슷한 예상 정확도를 나타내고 있다.

이 실험 결과는 정적 분류 정보를 사용하여 각 명령어를 적절한 예측기에만 할당함으로써 중복된 할당을 피할 수 있고, 따라서 보다 많은 명령어를 예측기에 할당할 수 있어 예상율과 예상 정확도를 향상시켰음을 보여주는 것이다.

5. 결 론

슈퍼스칼라 프로세서에서 비순서적으로 명령어가 이슈 및 수행될 때 명령어를 예상한 후 명령어가 완료되어 예상 테이블을 수정하기 전에 다시 그 명령어를 예상 시 부적절한 데이타 사용으로 인한 예상 실패를 방지하는 혼합형 결과값 예상 메커니즘을 제안하였다. 제안한 예측기는 명령어 반입 시 예상과 예측기의 갱신을 동시에 수행함으로서 부적절한 데이타 사용을 줄여 예상 정확도를 향상시킬 수 있었다. 또한, 프로파일링으로 얻어진 정적 분류 정보를 사용함으로써 명령어 반입시 명령어를 적절한 예측기에만 할당함으로써 예상 정확도를 더욱 향상시킬 수 있고, 예측기의 하드웨어를 절반 크기로 축소하여도 동등한 성능을 얻을 수 있었다.

실험 결과 SPECint95 벤치마크 프로그램에서 명령어를 예상 후 결과값이 구해지기 전에 그 명령어를 다시 예상하는 비율이 8-이슈 폭에서 평균 19%, 16-이슈 폭에서는 평균 28%로 측정되었으며, 이는 제안된 예측기를 사용하면 예상 정확도가 향상될 것을 증명해 주었다. 모험적 갱신을 하지 않은 혼합형 예측기보다 예상 정확도가 8-이슈폭에 대해 평균 61%에서 71%로, 16-이슈폭에 대해 평균 61%에서 73%로 증가하였다. 제안한 예측기에 정적 분류 정보를 사용함으로써 16-이슈에서 예상 정확도가 평균 88%로 향상되었고, VHT 엔트리 수를 절반 크기로 축소하여도 평균 85%의 예상 정확도를 얻었다.

향후 과제로 명령어에 적합한 예측기를 선택할 수 있는 컴파일러 기반 분류 알고리즘을 개발하여 분류 정보를 명령어에 적용하여 명령어 수행시 예측기에 전달하는 방법에 대한 연구가 필요하다.

참 고 문 헌

[1] M. H. Lipasti, C. B. Wilderson, J. P. Shen, *"Value Locality and Load Vaule Prediction,"* ASPLOS-VII, pp. 138-147, October 1996.

[2] M. H. Lipasti and J. P. Shen, *"Exceeding the Dataflow limit via Value Prediction."* MICRO-29, pp. 226-237, December 1996

[3] Y. Sazeides and J. E. Smith, *"The Predictability of Data Values,"* MICRO-29, pp. 226-237, December 1996.

[4] F. Gabbay and A. Mendelson, *"Can Program Profiling Support Value prediction?,"* MICRO-30, pp. 270-280, December 1997.

[5] T-Y Yeh and Y. N. Patt, *"Alternative Implementations of Two-Level Adaptive Branch Prediction,"* ISCA-19, pp.124-134, 1992.

[6] K. Wang and M. Franklin, *"Highly Accurate Data Value Prediction using Hybrid Predictors,"* MICRO30, pp. 281-290, December 1997.

[7] J. Gonzalez and A. Gonzalez, *"The potential of data value speculation to boost ilp,"* ICS-12, 1998

[8] G. Reinman and B. Calder, *"Predictive techniques for aggressive load speculation,"* MICRO-31, 1998

[9] T. Nakra, R Gupta and M.L. Soffa, *"Global Context-Based Value Prediction,"* HPCA-5, 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] Calder B., Feller P. and Eustace A., *"Value*

Profiling," MICRO-30, pp. 259-269, December 1997.
[12] 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.
[13] B. Calder, G. Reinman, D. M. Tullsen, *"Selective Value Prediction,"* ISCA-26, May 1999.
[14] B. Rychlik, J. W. Faistl, B. P. Krug, A. Y. Kurland, J. J. Sung, M. N. Velev, J. P. Shen, *"Efficient and Accurate Value Prediction Using Dynamic Classification"* Tech. rep. CMUART-1998-0
[15] J. Huang and D. Lilja, *"Exploiting Basic Block Value Locality with Block Reuse,"* in Procs. of 5th Int. Symp. on High-Performance Computer Architecture, 1999.
[16] S. J. Lee, P. C. Yew, *"On Some Implementation Issues for Value Prediction on Wide-Issue ILP Processors"* IEEE/ACM International Conference on Parallel Architectures and Compilation Techniques (PACT 2000), Oct. 2000.
[17] Q. Zhao, D. J. Lilja, *"Compiler-Directed Static Classification of Value Locality Behavior,"* Laboratory for Advanced Research in Computing Technology and Compilers Technical Report No. ARCTiC 00-07, July, 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

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