韓國 컴퓨터情報學會 論文誌
第10卷 第2號, 2005. 5.

2005-10-2-1-10

고성능 마이크로프로세서에서 값 예측기의 성능평가

전 병 찬*, 김 혁 진**, 류 대 희***

Performance Evaluation of Value Predictor in High Performance Microprocessors

Byoung-Chan Jeon*, Hyeock-Jin Kim**, Dae-Hee Ryu***

요 약

고성능 마이크로프로세서에서 값 예측기는 한 명령어의 결과를 미리 예측하여 명령들 간의 데이터 종속관계를 극복하고 실행함으로써 명령어 수준 병렬성(Instruction Level Parallelism, ILP)을 향상시키는 기법이다. 본 논문에서는 ILP 프로세서 명령어 수준 병렬성의 성능향상을 위하여 값을 미리 예측하여 병렬로 이슈하고 수행하는 값 예측기를 비교 분석하여 각 테이블 갱신 시점에 따른 예측기별 평균 성능향상과 예측률 및 예측정확도를 측정하여 평가한다. 이러한 타당성을 검증하기 위해 실행구동방식 시뮬레이터를 사용하여 SPECint95 벤치마크를 시뮬레이션하여 비교한다.

Abstract

value prediction in high performance micro processors is a technique that exploits Instruction Level Parallelism(ILP) by predicting the outcome of an instruction and by breaking and executing true data dependences. In this paper, the mean performance improvements by predictor according to a point of time for update of each table as well as prediction accuracy and prediction rate are measured and assessed by comparison and analysis of value predictor that issues in parallel and run by predicting value, which is for performance improvements of ILP in micro processor. For the verification of its validity the SPECint95 benchmark through the simulation is compared by making use of execution driven system.

▸ Keyword : Value Prediction, Instruction-Level Parallelism, Microprocessors

• 제1저자 : 전병찬
• 접수일 : 2005.04.08, 심사완료일 : 2005.05.20
* 청운대학교 컴퓨터학과 전임강사, ** 청운대학교 컴퓨터학과 부교수, *** 청운대학교 컴퓨터학과 교수

Ⅰ. 서 론

최근 하드웨어 기술의 급속한 발달로 많은 고성능 프로세서들이 개발되고 있으며, 이들 프로세서들은 고성능의 실현을 위해 ILP를 이용하고 있다. ILP 프로세서들은 다수의 데이타 패스와 기능장치들을 구비하여 여러개의 명령들이 동시에 이슈되어 병렬로 수행된다. ILP는 한 프로세서에서 여러개의 명령을 이슈하여 병렬로 실행하는 기법으로 최근에 개발되는 고성능 프로세서들은 모두 ILP를 이용하여 성능의 극대화를 꾀하고 있다. ILP처리의 주요장애는 명령들간의 데이타 종속성(data dependency)관계인데 이는 명령의 병렬처리를 방해한다. 지금까지 데이타 종속관계를 극복하는 대표적인 방법으로는 최적화 컴파일러가 데이타 종속관계가 없는 독립적인 명령들을 아직 사용하지 않은 이슈 슬릇(issue slot)으로 코드를 이동하여 정적으로 스케쥴하였으나 명령이슈가 4이상으로 커짐에 따라 각 이슈 슬릇에 채울 더 많은 독립적인 명령들을 채우는 것이 어렵게 되었다. 따라서, 최근에는 실행되는 명령들의 결과 값을 미리 예측하고 이후 데이타 종속관계가 있는 명령들에게 조기에 공급하여 실행시킴으로서 성능향상을 꾀하는 데이타 값 예측 방식에 관하여 활발히 연구가 진행되고 있다. 데이타 값 예측기는 하나의 명령을 실행하기 전에 실행결과를 하드웨어에 의해 미리 예측하고 모험적(speculative)으로 실행하여 이 명령과 데이타 종속관계가 있는 명령들이 중지(stall)되지 않고 계속 이슈하여 실행하므로써 프로세서의 성능 향상을 시킨다. (그림 1)은 데이타 종속관계를 보여주는 그림이다. 그림의 (a)는 j의 R3는 i의 R3 결과가 생성되어야만 사용이 가능하며, j가 실행된 후 R5의 결과가 생성되면 명령 k, l이 실행될 수가 있다. 이러한 데이타 종속관계가 있는 명령들은 데이타 값 예측기를 사용하여 그림의 (b)와 같이 i, j의 결과 값 R3, R5를 미리 예측하여 i, j와 데이타 종속관계를 갖는 명령 k, l들에 대한 종속관계를 제거하고 이들 명령들을 투기적으로 실행하여 한 클럭 사이클에 이슈할 수 있다[1]. 본 논문에서는 고성능 마이크로 프로세서에서 ILP을 이용하여 프로세서의 성능을 향상시키기 위하여 데이타 값을 미리 예측하여 병렬로 이슈하고 수행하하는 테이블 갱신에 따른 데이타 값 예측기의 성능을 측정한다. 그리고 테이블 갱신에 따른 예측율 및 예측 정확도를 측정하여 평가한다. 실험되는 데이타 값 예측기는 최근 데이타 값 예측기[2], 스트라이드 데이타 값 예측기[3], 2-단계 데이타 값 예측기[4], 2-단계 값을 결합한 혼합형 데이타 값 예측기[5]로 구성되어 있다. 실험의 타당성 검증을 위해 실행 구동방식 시뮬레이터인 Simplescalar/Alpha 3.0 tool set[1]에 이용하여 SPECint95 벤치마크를 시뮬이션하여 비교한다[6].

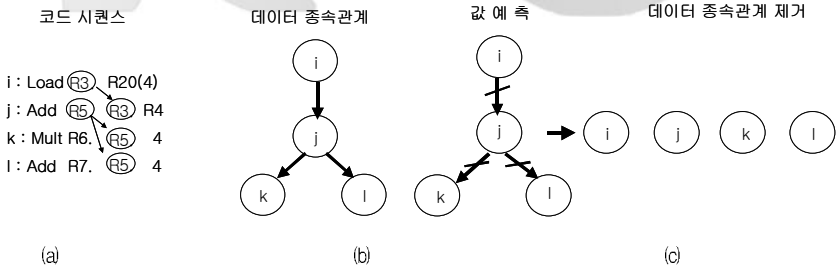

그림 1. 명령에서의 데이타 종속관계
Fig1. Data dependences in ILP

Ⅱ. 데이타 값 시퀀스

데이타 값 예측기는 프로세서에 의해 반복 수행되어지는 명령들의 반복되는 과거의 결과 값의 유형에 근거하여 값을 예측한다. 데이타 값 예측기의 동작은 대개 비슷한데 명령의 값 예측은 다음에 수행될 명령의 PC를 사용하여 명령을 페치할 때 동시에 예측기의 테이블을 참조하여 값을 예측한

다. 그리고 이 명령의 수행 후에 결과 값이 생성되면 이 값과 예측의 성공여부에 근거하여 예측 테이블을 갱신하고 이후 예측을 준비한다. 일반적으로 반복 수행되는 명령은 <표 1>과 같은 상수, 스트라이드와 비-스트라이드의 값 시퀀스 패턴을 갖는다[1]. 상수 시퀀스는 명령의 수행 결과가 변하지 않고 계속 같은 값을 생성하는 유형이다. 이 유형의 값 예측은 최근의 수행된 결과 값을 예측 테이블에 저장하고 다음에 해당 명령의 페치 시에 이 결과 값을 참조하여 예측하면 된다. Lipasti 등은 이러한 패턴이 자주 발생함을 주목하여 최근 데이타 값 예측기를 개발하였다[1,2]. 스트라이드 시퀀스는 명령이 수행될 때마다 일정한 값만큼의 간격 차이(이를 스트라이드라 한다)로 증가 감소되는 패턴으로 프로그램에서 배열이나 루프 인덱스들을 참조할 때 빈번히 발생하는 패턴들이다. 이러한 유형의 명령들의 예측을 위해서 스트라이드 데이타 값 예측기가 개발되었다[1,7]. 즉, 이 데이타 값 예측기는 3개의 명령의 수행결과를 추적하여 각 명령들 간의 결과 값의 차이인 두 개의 스트라이드를 구한다. 이 두 개의 스트라이드 값이 같으면 스트라이드 시퀀스로 간주하여 최근에 수행된 결과 값과 스트라이드를 예측 테이블에 저장한다. 다음에 해당 명령을 다시 페치하여 수행하는 경우 예측 테이블에 있는 최근의 결과 값과 스트라이드를 더하여 구해진 값을 예측 값으로 한다. 상수 시퀀스의 경우는 스트라이드가 0인 경우로 해석할 수 있기 때문에 스트라이드 데이타 값 예측기는 최근 데이타 값 예측기를 포함한다. 비-스트라이드 시퀀스는 위의 상수 시퀀스 또는 스트라이드 시퀀스에 속하지 않는 유형으로 <표 1>에서와 같이 두 가지로 분류될 수 있다. 반복패턴은 여러 값들의 시퀀스가 반복되는 경우이다. 구문형 데이타 값 예측기(context -based value predictor)는 이러한 반복되는 값들을 모두 테이블에 저장하고 최근의 값들의 패턴으로 근거하여 다음 값을 예측한다[8,9,10,11,12,13,14]. 이 데이타 값 예측기는 비-스트라이드 시퀀스의 반복되는 패턴도 예측

표 1. 값 시퀀스 유형
Table 1. Type of value sequential

값의 유형	데이타 시퀀스 예
상수	5 5 5 5 . . .
	같은 값 반복
스트라이드	1 3 5 7 . . .
	스트라이드가 2
비-스트라이드	6 11 22 -5 6 11 22 -5 6 . . .
	반복 패턴
	17 9 -13 5 15 . . .
	비 반복 패턴

할 수 있어서 스트라이드 보다는 성능이 우수하지만 예측 테이블에 반복되는 값들을 모두 저장해야 하기 때문에 부가되는 하드웨어의 경비가 크다는 단점이 있다. 비 반복 패턴은 값 예측을 위해 어떤 패턴도 추출할 수 없는 경우로 예측이 불가능하다. 최근에는 최근 데이타 값 예측기, 스트라이드 데이타 값 예측기와 구문형 예측기를 혼합한 혼합형 데이타 값 예측기(hybrid data value predictor)도 제안되었다[4,5,7,9]. 혼합형 데이타 값 예측기는 여러 유형의 반복되는 명령에 대해서 예측이 가능하여 성능이 우수하지만 예측 테이블의 크기 증가로 높은 하드웨어 경비가 요구된다.

III. 데이타 값 예측기

데이타 값 예측기는 예측되는 데이타 값 시퀀스의 분류에 따라 최근 데이타 값 예측기, 스트라이드 데이타 값 예측기, 2-단계 데이타 값 예측기, 혼합형 데이타 값 예측기 등이 있다.

3.1 최근 데이타 값 예측기(Last Data Value Predictor)

최근 데이타 값 예측기는 데이타 값 예측장치를 위해 2-단계 예측 구조를 갖는다. 첫번째 단계에서는 예측값을 생성하고, 두번째 단계에서는 예측의 정확성 여부를 결정한다. CT(Classification Table)와 VPT(Value Prediction Table)는 모두 예측되는 명령의 주소(PC)에 의해 인덱스 되고 직접 매핑 되어진다. CT의 엔트리는 가용 엔트리를 나타내는 1 비트이거나 PC의 상위 비트를 구성된 태그를

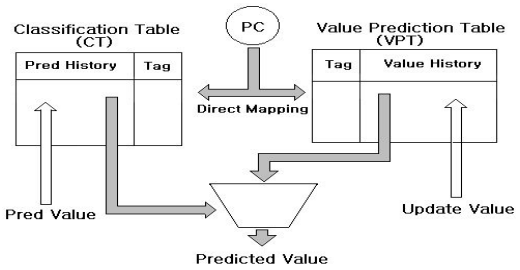

그림 2. 최근 데이타 값 예측기
Fig. 2 Last data value predictor

저장하는 유효필드와 한개 또는 여러 개의 비트들로 구성된 포화 계수기(saturating counter)인 예측 히스토리(prediction history)로 구성된다. 예측 히스토리는 예측의 값에 따라 증가감소 한다. CT는 특정명령이 예측 가능한지를 결정하고, VPT는 유효필드와 LRU 정책에 따라 유지되는 하나 이상의 32-64 bits values로 구성된 값 히스토리 필드로 구성된다. 값 히스토리 필드는 예측이 틀렸을 경우 교체되고, 교체정책(replacment policy)은 CT 예측 히스토리에 의해 좌.우된다. (그림 2)는 간단한 최근 데이타 값 예측기의 구조를 보여준다.

3.2 스트라이드 데이타 값 예측기(Stride Data Value Predictor)

스트라이드 데이타 값 예측기는 한 명령어의 연속적인 인스턴스(instances)의 결과들이 변화하는 스트라이드를 모니터하여 이들 결과들이 일정한 폭으로 변하면 그 명령어의 장래 인스턴스의 결과를 예측하는 것이 용이하므로 순환 명령이나 일정하게 배열의 값이 증.감하는 프로그램에서 잘 동작되는 방법으로써 데이타를 캐시로 선행인출(prefetch)하기 위해 주소를 생성하는데 성공적으로 수행된다. 이러한 스트라이드를 근간으로하는 모험적 실행 방법을 이용하여 데이타 값 "Locality"를 얻게 된다.

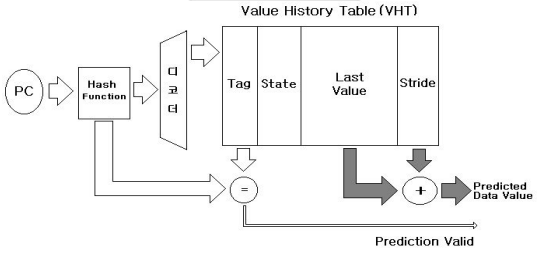

그림 3. 스트라이드 데이타 값 예측기
Fig. 3 Stride data value predictor

(그림 3)은 간단한 스트라이드를 근거로한 데이타 값 예측기의 구조를 보여준다. 또한 (그림 4)는 스트라이드를 근거로 한 예측기에서 다음 스트라이드를 검출 하기위한 기본적인 상태 변환을 보여준다. 이러한 상태변화는 스트라이드 시퀀스를 검출하기 위한 목적으로 명령어를 만나게 되는 첫 번째 시간에는 어떤 예상도 일어나지 않지만, 그 명령어가 결과를 생성할 때 VHT에 엔트리가 할당되고 다음의 동작이 발생한다.

결과 값은 해당 엔트리의 값 필드에 저장되고, 엔트리의 상태 필드의 값은 초기상태(Initial state : "Init")로 설정된다. "Init" 상태에 있는 동안에는 동일한 명령의 다른 인스턴스를 만날 때에는 결과 값을 생성하기 위하여 예측시도는 하지 않고 스트라이드 시퀀스의 처음 값의 결과인 첫 번째 데이타 S_{D1}(first data in stride prediction)이 생성

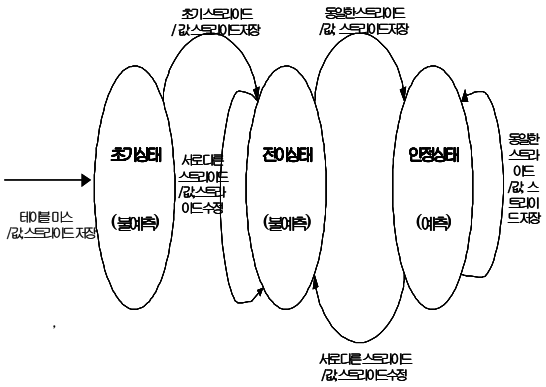

그림 4. 스트라이드 상태도
Fig. 4 Stride state

되고, 다음 예측에 사용될 스트라이드 S_{s1}(first stride in stride prediction)은 S_{D1}-2HT엔트리 내의 이전 값]으로 계산되며, S_{D1}과 S_{s1}은 값 필드와 스트라이드 필드에 저장된다. 이때 상태 필드는 전이 상태(Transient state : "Transient")로 설정되고, "Transient" 상태에 있는 동안에 동일한 명령어의 다른 인스턴스를 만나면 예상은 일어나지 않으며, 그 인스턴스의 결과인 두 번째 데이타 S_{D2}를 생성할 때 다음 동작이 발생한다. 따라서 다음 스트라이드 S_{s2}는 S_{D2} - [VHT엔트리 내의 이전 값]으로 계산되고, 두 번째 인스턴트의 결과 데이타 S_{D2}는 값 필드에 저장되고 S_{s2}가 이전의 스트라이드 S_{s1}과 동일하면 상태 필드는 안정 상태(Steady state :"Steady")로 설정이 되지만 그렇지 않으면 S_{s2}가 스트라이드 필드로 저장되고 여전히 상태 필드는 "Transient"상태로 남아있게 된다. 만약 S_{s1}과 S_{s2}가 동일하다면 "Steady" 상태 동안 값 필드와 스트라이드 필드를 함께 추가하여 다음의 예상이 만들어진다.

3.3 단계 데이타 값 예측기 (Two-Level Data Value Predictor)

2-단계 데이타 값 예측기는 VHT와 PHT(Pattern History Table)로 구성되는데, VHT는 4개의 필드들을 포함하고 있으며, 2-단계 데이타 값 예측기의 구성은 (그림 5)와 같다. VHT테이블은 예측되는 PC에 의해 인덱스 되어지며, 테이블에 직접 맵핑 되어 진다. 또한 VHT의 엔트리는 PC의 상위 비트들로 구성된 tag필드와 저장된 데이타의 최근 사용여부의 순서 정보를 갖는 LRU정보, 그리고 최근에 사용된 n개의 데이타 값과 마지막 p개의 명령 결과를 비트 패턴으로 저장하여 PHT를 인덱스 하는 VHP필드로 구성된다. 또한 데이터 값 필드의 크기는 history의 크기에 따라 결정된다. 또한 VHP는 마지막 p개의 명령을 $\log_2(hist) * p$의 비트 패턴으로 표시되어지며, PHT는 VHP로 인덱스 되어지고, PHT의 각 엔트리는 history 갯수의 독립적인 증.감 카운터로 구성된다. 따라서 lookup동작은 PC에 의해 VHT가 인덱스 되고 히트(hit)되면 VHP에 의해 PHT가 인덱스 된다. PHT에서 history 크기만큼의 카운터 중에서 최대값을 선택하고, 이 값이 임계값(threshold)보다 크면 이 카운터의 위치에 대응되는 VHT의 데이타 값이 예측되고, 임계값 보다 작은 경우에는 예측을 하지 않는다. 또한 Update동작은 명령 수행 후 생성된 결과가 VHT의 데이타 값 필드에 없는 경우 LRU정보를 참조하여 최근에 가장 적게 사용된 데이타를 새로운 데이타로 대체한다.

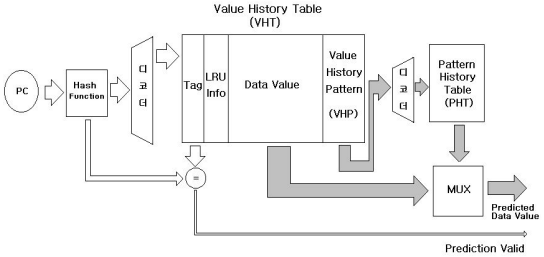

그림 5. 2-단계 데이타 값 예측기
Fig. 5 Two-level data value predictor

즉, history 크기 만큼의 데이타 값 필드 배열에 가장 적게 사용된 순으로 데이타 값을 해당위치에 저장하고, 가장 최근에 사용된 데이타의 값을 배열의 마지막에 저장한 후에 최근에 사용된 데이타 이후의 배열 값을 하나씩 이동한다. 그리고 참조된 VHT엔트리의 VHP에 명령 수행 후 생성된 데이타 위치를 나타내는 $\log_2(hist)$비트 패턴이 삽입이 되면서 이전의 비트들은 $\log_2(hist)$비트만큼 왼쪽으로 이동(left shift)된다. 따라서 참조된 PHT 엔트리의 카운터 값은 예측이 맞는 경우 3을 증가시키고, 값이 다른 카운터에는 1씩 감소시키면서 갱신한다.

3.4 혼합형 데이타 값 예측기(Hybrid Data Value Predictor)

혼합형 데이타 값 예측기[21]는 스트라이드와 2-단계 데이타 값 예측기를 결합한 예측기이다. (그림 6)은 혼합형 데이타 예측기의 구조를 보여준다. 2-단계 데이타 값 예측기와 비교하여 VHT에 스트라이를 저장하는 필드가 추가되었다. 혼합형 데이타 값 예측기는 먼저 데이타 값 예측을 위해 2-단계 데이타 값 예측 방식을 적용을 시도하고, PHT의 카운터 값들을 참조한다. 만약 카운터의 최대값이 설정된 임계값보다 크거나 같으면 2-단계 데이타 값 예측을 적용하여 이에 해당하는 VHT의 값으로 예측하고, 그렇지 않으면 스트라이드 예측기를 사용하여 최근값에 스트라이드를 더 한 값으로 예측한다.

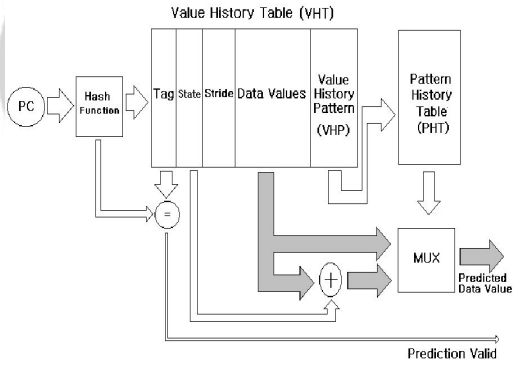

그림 6. 혼합형 데이타 값 예측기 구조
Fig. 6 Hybrid data value predictor

Ⅳ. 실험 방법

본 논문의 성능측정은 8 이슈의 명령의 시스템의 기본 구조를 사용하였다. <표 2>는 기존의 최근 데이타 값 예측기, 스트라이드 데이타 값 예측기, 2-단계 데이타 값 예측기, 혼합형 데이타 값 예측기는 8K 엔트리의 VHT와 4개의 히스토리를 갖고, 8K 엔트리로 구성된 PHT의 테이블 크기로 구성된다. 트레이스 캐시는 512KB 와 블록당 16개 명령을 가진 2-way 어소시에티브를 가진다.

표 2. 실험에 사용된 데이터 값 예측기
Table 2. Data value predictor using experiment

	테이블 구성
Last Value Predictor	8K VHT(Last Value Table)
Stride Value Predictor	8K SVT(Stride Value Table)
Two-level data value predictor	8K VHT(Value History Table) 8K PHT (Pattern History Table) 4 history data values
Hybrid Value predictor	8K VHT(Value History Table) 8K PHT(Pattern History Table) 4 history data values

<표 3>에서는 기본구조에서 사용된 머신의 파라미터를 보여준다. 분기예측을 위해서 2-way의 2K 엔트리를 갖는 BTB와 gshare와 bimodal 방식을 혼합한 분기 예측기를 사용하였다. 그리고 512K 크기를 갖는 명령 캐시와 데이타 캐시를 사용하였다.

시뮬레이션을 수행하기 위해 사용되어진 벤치마크 프로그램은 <표 4>와 같이 SPECint95 벤치마크 프로그램 중 8개의 프로그램에 대해 시뮬레이션 하였다. <표 4> 에서 기본 IPC(Instructions Per Cycles)는 데이타 값 예측을 적용하지 않은 경우의 IPC로써 speed up 측정 시 기본 값이 된다. 본 연구에서 사용된 시뮬레이터는 Simplescalar 3.0 Tool set[1]에 데이타 값 예측기 모델을 삽입하여 실험 하였다.

표 3. 실험에 사용된 기본 시스템
Table 3. Basic system using experiment

	파라미터 (Parameter)	값(Value)	의미(Comments
Processor core	RUU size	256	instruction windows
	LSQ size	128	Load-Store Queue
	Fetch width	8 instr/cycle	8 issue baseline
	Decode width	8 instr/cycle	8 issue baseline
	Issue width	8 instr/cycle	8 issue baseline
	Commit width	8 instr/cycle	8 issue baseline
	Functional Unit	8 int ALU(1) 2 int MUL(3) 4 fp ALU(1) 2 fp MUL(4)	()은 Latency 임
Branch Predictor	BTB	2K, 2-way	Branch target buffer
	Combine Branch Predictor	2-level 16K, 4bit history bimodal 16K	2-level + bimodal
	RAS	32	return addr. stack
Cache	L1 D-cache	512K, 2-way, 32 byte blocks	Data cache
	L1 I-cache	1-cycle latancy 512K	Instruction cache
	L2 cache	direct map, 128 byte block 4-way,	secondary cache

표 4. SPECint95 벤치마크 프로그램의 입력자료
Table 4. Input data of SPECint95 benchmark

벤치마크 프로그램	입력자료	명령의 크기	기본 IPC(8이슈)		
			imm	commit	wb
compress	10000 e 231	35,261,714	2.7743	2.7743	2.7743
gcc	jump.i	38,772,635	1.7702	1.7692	1.7701
go	5 9	81,530,040	1.5199	1.5203	1.5198
ijpeg	tinyrose.ppm	63,415,601	3.2253	3.2241	3.2253
li	queen6.lsp	41,524,887	2.5914	2.5914	2.5914
m88ksim	dhry.big.100	240,738,844	5.5759	5.5757	5.5759
perl	scrabbl.in	40,353,136	2.359	2.3620	2.3590
vortex	persons.250	66,740,617	2.6992	2.6990	2.6992
평 균		7,604,218,425	2.8144	2.8145	2.8143

Ⅴ.성능분석 및 평가

본 논문에서 실험한 데이타 값 예측기는 각 테이블 갱신 시점에 따른 예측기의 성능향상과 예측 정확도, 예측률의 실험 결과를 얻었다. (그림 7)은 예측 테이블의 갱신을 명령 실행 후 writeback 단계에서 하는 경우의 각 예측기별 성능향상을 보인 그림이다. (그림 7)에서 나타난 바와 같이

최근 데이타 값 예측기, 스트라이드 데이타 값 예측기, 2-단계 데이타 값 예측기 및 혼합형 데이타 값 예측기의 평균 성능향상이 각각 8.9%, 6.1%, 12.4%, 9.5%임을 알 수 있었다. 2-단계 데이타 값 예측기의 평균 성능향상이 가장 우수하였지만 특히, m88ksim의 벤치마크에서는 가장 큰 저하를 보여주고 있다.

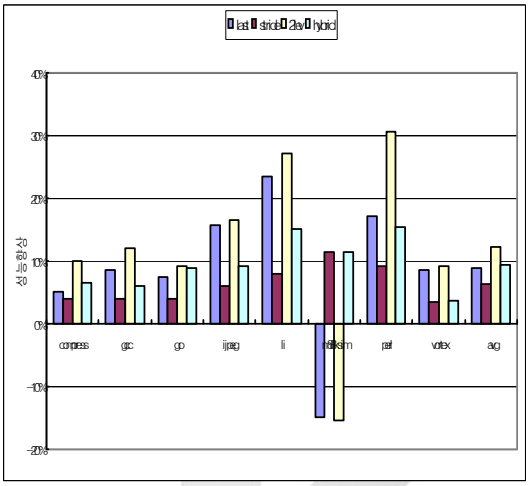

그림 7. 데이타 값 예측기별 성능향상
Fig. 7 Performance improvements of data value predictors

(그림 8)은 각 테이블 갱신 시점에 따른 예측기별 평균 성능향상을 보여주는 그림이다. immediate는 명령 페치시 예측값의 참조를 위해 예측 테이블을 참조하자마자 새로운 값으로 갱신하는 이상적인 경우를 가정한 것이다. writeback은 명령 실행 후 결과 값이 산출하자마자 예측 테이블을 갱신하는 것을 나타내며, commit은 예측 명령 이전의 모든 명령이 실행을 하여 비-투기적 상태가 되었을 때 예측 테이블을 갱신한다. (그림 8)에서 나타난 바와 같이 immediate인 경우가 가장 좋음을 알 수 있다. 특히, 2-단계 데이타 값 예측기인 경우에 immediate의 성능이 우수함을 알 수 있었다. 이는 immediate인 경우 예측 테이블의 갱신지연이 없기 때문이다. writeback 과 commit을 비교할 때 전반적으로 commit이 우수함을 알 수 있었지만, 최근 데이타 값 예측기인 경우에는 writeback 이 약간 우수함을 알 수 있다.

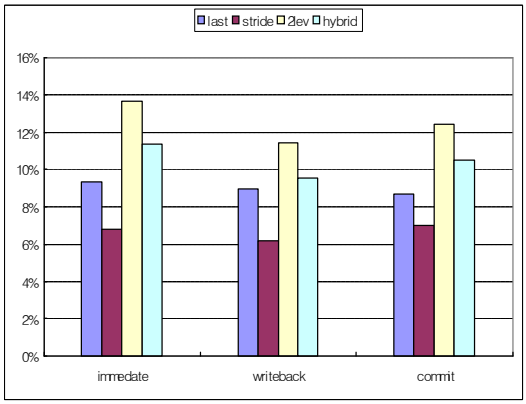

그림 8. 테이블 갱신 시점에 따른 예측기별 평균 성능향상
Fig. 8 Performance improvements by predictor according to a point of time for update of each table

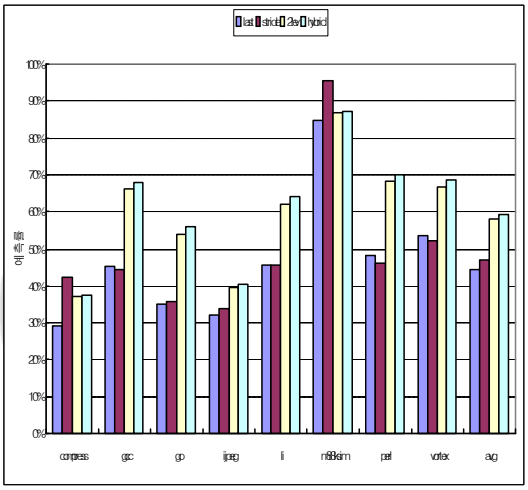

그림 9. 데이타 값 예측기별 예측률
Fig. 9 Prediction rate of data value predictor

(그림 9)에서 데이타 값 예측기별 예측률은 데이타 값 예측을 올바르게 시도한 명령들의 비율인데 (그림 9)에서 보듯이 최근 데이타 값 예측기, 스트라이드 데이타 값 예측기, 2-단계 데이타 값 예측기, 혼합형 데이타 값 예측기의 평균 예측률은 각각 48%, 49%, 50%, 65%임을 알 수 있다. 혼합형 데이타 값 예측기가 가장 예측률이 높음을 알 수 있다.

(그림 10)은 기존의 테이블 갱신 시점에 따른 데이타 값 예측기의 예측정확도를 보여주고 있다. 예측정확도는 레지스터에 결과 값을 저장하는 예측이 가능한 명령어 중에서 안정 상태에 도달하여 실제로 값을 예측한 명령 중에서 올바르게 값을 예측한 비율이다. (그림 10)에서 나타난 바와

같이 commit인 경우가 가장 정확하게 예측함을 알 수 있었다. writeback과 commit을 비교할 때 최근 데이타 값 예측인 경우 commit이 0.2%정도의 예측정확도의 차이를 보였고, 혼합형 데이타 값 예측기인 경우는 0.1%의 예측정확도의 차이를 보였다.

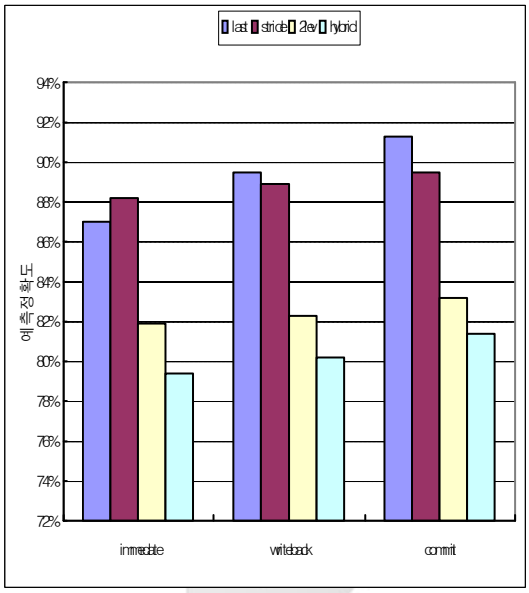

그림 10. 테이블 갱신 시점에 따른 평균 예측정확도
Fig. 10 Average prediction accuracy according to a point of time for update of each table

Ⅵ. 결론

본 논문에서는 최근 데이타 값 예측기, 스트라이드 데이타 값 예측기, 2-단계 데이타 값 예측기, 혼합형 데이타 값 예측기의 각 예측기별 성능과 예측률, 예측 테이블의 갱신 시점 및 명령윈도우 크기에 따른 데이타 값 예측기의 성능 영향을 분석하였다. 성능분석결과 고성능 슈퍼스칼라 프로세서는 예측 테이블 갱신에 따라 2-단계 데이타 값 예측기가 성능이 우수함을 보였고, 예측률은 혼합형 데이타 값 예측기가 예측률이 높음을 알 수 있었다. 그리고, 예측 정확도는 예측 테이블 갱신 시점인 경우에 commit이 가장 정확하게 예측됨을 알 수 있었다.

향후 연구과제는 혼합형 데이타 값 예측기에 대한 연구가 필요하다.

참고문헌

[1] 전병찬, 이상정 "와이드 이슈 프로세서를 위한 스트라이드 값 예측기의 모험적 갱신", 한국정보과학회 논문지, 제28권 제12호, pp.601-612. 2001.

[2] M.Lipasti, and J.Shen, "Exceeding th Limit via Value Prediction", Proceedings of the 29th International symposium on Microarchitecture (MICRO-29), Dec. pp.226-237, 1996

[3] Kai Wang, Manoj Franklin, "Highly Accurate data value Predictions using hybird predictor," Proc. of 30th Annual ACM/IEEE International Simposium on Microarchitecture, December, pp.281-290, 1997.

[4] T. Yeh and Y. Patt, "Two-Level Adaptive Branch Prediction," Proceedings of the 24th International Symposium microarchitecture (MICRO-24), Nov., pp.51-61, 1991.

[5] 전병찬, 박희룡, 이상정, "와이드 이슈 슈퍼스 칼라 프로세서의 혼합형 값 예측기의 성능평가," 대한전자공학회호서지부 추계학술발표대회논문집 제2권 제1호, pp.136-141, 2000.

[6] D.Burger and T.Austin, "The Simplescalar Tool Set, Version3.0", Technical Report CS-RT-97-1342, University of Wisconsin, Madison, Jun., 1997.

[7] F.Gabbay, and A.Mendelson, "Speculative Execution Based on Value Prediction", EE Department TR 1080, Technion-Israel Institute of Technology, Nov., 1996.

[8] M.H.Lipasti, C.B.Wilkerson and J.P.Shen."Value Locality and Load Value Predictio-n," Proc of the 7th International Conference on Architectural Support for Programming Languages and Operating Systems(ASPLOS-Ⅶ), Oct, pp138-147, 1996.

[9] B.Rychlik, J.Faistl, B.Krug, and J.Shen, Efficacy and Performance Impact of Value Prediction,"Parallel

Architectures and Compilation Techniques, Paris, Oct. 1998.

[10] Y.Sazeides, and J.Smith, "The Predicatability of Data Values", Proceedings of the 30th International Symposium on Microarchitecture (MICRO-30), Dec. pp.248-258, 1997.

[11] N.P.Jouppi, D.W.Wall,"Available Instruction-level Parallelism for Superscalar and Superpipeline Machines," Proc. of the 3th International Conference on Architectural Support for Program Languages and Operating Systems, April, pp.272-282, 1989.

[12] 박두열, "HDL을 이용한 파이프라인 프로세서의 테스트 벡터의 구현에 의한 시뮬레이션", 한국컴퓨터정보학회 논문지 제5권 제 3호, pp.16-28, 2000.

[13] 김혁진, "B-spline에 대한 근사변환의 실험분석",한국컴퓨터정보학회 논문지, 제10권 제1호,pp.35-43, 2005.

[14] 이상정, 전병찬, "값 예측을 위한 순차적이고 선택적인 복구방식", 한국정보과학회 논문지 제31권 제1·2호, pp.67-77, 2004.

저 자 소 개

전 병 찬

순천향대학교 대학원 전산학과 박사
2002~2004 청운대학교 컴퓨터학과 강의전담
2005년~현재 청운대학교 컴퓨터학과 전임강사
<관심분야> 컴퓨터구조, 홈네트워크, 모바일,마이크로프로세서 등

김 혁 진

아주대학교 대학원 컴퓨터공학과 석·박사
1992~1997년 김천대학 사무자동화과 조교수
1997년~현재 청운대학교 컴퓨터학과 부교수
<관심분야> CG, CAGD, 웹기술등

류 대 희

원광대학교 대학원 이학박사
1995~현재 청운대학교 컴퓨터학과 교수
<관심분야> 알고리즘, 퍼지이론,확률응용 등