ILP 프로세서를 위한 부정적 간섭을 감소시키는 동적 분기예상 기법

An Dynamic Branch Prediction Scheme to Reduce Negative Interferences for ILP Processors

박 홍 준* 조 영 일**
Hong-Jun Park Young-Il Cho

요 약

ILP 프로세서는 고성능을 유지하기 위해 정확한 분기예상 방법을 요구한다. Two-Level 분기예상 방법은 높은 분기예상 정확성을 갖는 것으로 알려져 있다. 그러나, 한 분기 명령이 다른 분기 명령에 의해 갱신된 PHT 엔트리를 사용할 때 간섭이 발생하며, 간섭 중 부정적 간섭은 잘못된 예상(misprediction)을 유발하여 성능에 부정적 영향을 주게 된다. Agree 분기예상 방법에서는 BTB에 bias 비트를 추가하여 부정적 간섭을 긍정적 간섭으로 변환하여 예상 정확도를 높였으나, bias 비트를 잘못 설정하는 경우에는 오히려 부정적 간섭이 증가하게 된다.

본 논문에서는 이러한 부정적 간섭을 감소시키는 새로운 동적 분기예상 방법을 제안한다. 제안한 분기예상 방법은 수행시간에 bias 비트를 동적으로 변경시키기 위해 BTB의 엔트리에 hit 비트를 추가하였다. 그 결과 부정적 간섭을 효과적으로 감소시켜 예상 정확도를 향상시켰다. 제안된 방법의 효율성을 보여주기 위해, SPEC92int 벤치마크를 사용하여 성능을 평가한 결과, 제안된 방법이 기존의 방법보다 성능이 우수함을 확인하였다.

Abstract

ILP processors require an accurate branch prediction scheme to achieve higher performance. Two-Level branch predictor has been known to achieve high prediction accuracy. But, when a branch accesses a PHT entry that was previously updated by other branch, Two-level predictor may cause interferences. Negative interferences among all interferences have a negative effect on performance, since they can cause branch mispredictions. Agree predictor achieve high prediction accuracy by converting negative interferences to positive interferences by adding bias bits to BTB, but negative interferences may occur when bias bit is set incorrectly.

This paper presents a new dynamic branch predictor which reduces negative interferences. In the proposed predictor, we attach hit bits to entries in BTB to change bias bit dynamically during the execution time. As a result, the proposed scheme can improve the accuracy of prediction by reducing negative interferences effectively. To illustrate the effect of the proposed scheme, we evaluate the performance of this scheme using SPEC92int benchmarks. The results show that the proposed scheme can outperform traditional branch predictors.

1. 서 론

슈퍼스칼라 프로세서(Superscalar Processor)는 사이클 당 다수의 명령어를 반입, 해독, 이슈, 수행시킴으로써 명령어 수준 병렬성(Instruction Level Parallelism:ILP)을 향상시키는 고성능 프로세서이다[1]. 프로세서의 명령어 반입 능력은 분기로 인한 제어 종속(Control Dependence) 관계에 의해 제한되며, 명령어들의 많은 부분을 구성하는 분기 명령에 의한 제어 종속이 ILP 향상에 가장 큰 제약요소가 된다.

프로세서의 명령 반입 폭(fetch width)과 파이프

* 정회원 : 극동정보대학 전산정보처리과 교수
hjpark@cs.kdc.ac.kr
** 정회원 : 수원대학교 컴퓨터과학과 교수
yicho@mail.suwon.ac.kr

라인 깊이(pipeline depth)가 계속 증가함에 따라 제어 종속에 따른 제약이 더욱 심각해지고 있다. 따라서, ILP 프로세서는 고성능을 유지시키기 위하여 분기 결과와 목적 주소를 빠르고 정확하게 예상하는 메커니즘이 필수적이다[2,5,10].

정확한 분기예상을 위해서는 컴파일 시간에 수집된 분기명령에 대한 예상 데이터를 명령어의 opcode에 적용시키는 정적 분기예상 방법보다는 수행시간에 분기 동작에 대한 히스토리(history) 정보에 의해 예상을 하는 동적 분기예상이 높은 정확성을 갖는다[3,5,6].

동적 분기예상 방법으로 높은 분기예상 정확성을 갖는다고 알려진 Two-Level Adaptive Training 방법은 분기 결과에 따라 분기 히스토리 레지스터(Branch History Register:BHR)와 패턴 히스토리 테이블(Pattern History Table:PHT)에서 포화 카운터(saturating counter)의 갱신을 기초로 분기예상을 한다[2]. 선행분기 명령어들의 분기 결과를 저장한 BHR을 사용하여 PHT에 대한 인덱스를 생성한 후 첫 번째 단계의 인덱스 함수에 의해서 사상(mapping)된 PHT의 엔트리로 분기예상을 수행한다. PHT의 엔트리 수가 유한하기 때문에 서로 다른 분기 명령이 인덱스 함수에 의해 동일 PHT 엔트리를 사용하는 경우 간섭(interference)이 발생할 수 있다. 이러한 간섭 중 임의의 분기 명령에 대한 예상을 수행하게 될 때 동일 PHT 엔트리를 사용하는 이전의 분기 명령에 의해 잘못된 예상을 유발하는 부정적 간섭(negative interference)이 발생하기 때문에 분기예상 정확도가 감소하는 문제점을 가진다[1].

Agree Predictor에서는 BTB에 bias 비트를 추가함으로써 이러한 부정적 간섭을 중립적 간섭(neutral interference)이나 긍정적 간섭(positive interference)으로 변환시켜 분기예상 정확도를 향상시켰다[1]. 그러나, Agree Predictor는 bias 비트를 사용하여 분기예상 정확도를 증가시켰으나, 최초에 bias 비트의 설정을 잘못하는 경우에는 오히려 부정적 간섭이 증가하게 된다는 문제점을 갖는다[12].

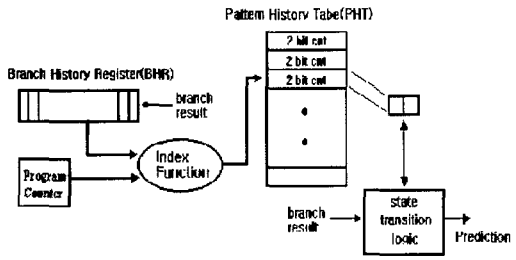
본 논문에서는 Agree Predictor의 이러한 문제점을 해결하기 위하여 bias 비트를 동적으로 변경할 수 있는 Hit Predictor라는 방법을 제안한다. 제안한 Hit Predictor 방법에서는 BTB에 2비트의 hit 비트를 추가하여 bias 비트를 동적으로 변경시켜 줌으로써 잘못 설정된 bias 비트에 의한 부정적 간섭을 감소시켜 분기예상 정확도를 향상시킨다. 본 논문에서 제안된 방법의 효율성을 확인하기 위하여 execution-driven 시뮬레이터인 Shade를 사용하여 SPEC92int 벤치마크에 대해 Two-level Adaptive Training, Agree 그리고 Hit Predictor의 성능을 비교하도록 한다.

2. 동적 분기예상 기법(Dynamic Branch Predictors)의 고찰

동적 분기예상 기법은 프로그램을 profiling한 결과로 예상을 하는 정적 분기예상 방법과는 달리 수행 중 발생한 분기결과를 바탕으로 분기예상을 수행한다. 분기 명령어로 인한 성능의 저하를 최소화하는 대표적인 동적 분기예상 방법들은 다음과 같다.

2.1 Two-Level Adaptive Training 분기예상 방법

Two-Level Adaptive Training(TLAT) 분기예상 방법은 예상 정확도가 높은 것으로 알려져 널리 사용되고 있으며, 그림 1과 같이 두 단계의 히스토리 정보(BHR과 PHT)를 사용하여 분기예상을 한다[2,10].



(그림 1) Two-Level Adaptive Training 분기예상 방법의 구조

첫 단계의 히스토리 정보를 갖는 BHR은 최근 k개의 분기 명령에 대한 분기 결과를 기록하는 k 비트 시프트 레지스터(Shift Register)로 구성된다. 분기 명령의 수행 결과가 결정되면 BHR은 한 비트씩 좌로 시프트 되고 k번째 비트에 수행 결과가 taken이면 1, not-taken이면 0이 저장된다.

첫 단계의 인덱스 함수에 의해 인덱스 되는 두 번째 단계의 패턴 히스토리 테이블(PHT)은 2^k개의 엔트리로 구성되며, 각 엔트리의 2비트 업다운 포화 카운터(2bit UP/DOWN saturating Counter)를 사용하여 분기 방향을 예상한다. 카운터의 값이 2 이상이면 taken으로, 1이하이면 not-taken으로 예상한다. 이 카운터는 분기 결과가 taken으로 결정되었을 때는 증가하고, 그렇지 않은 경우에는 감소한다.

TLAT 분기예상 방법은 첫 단계의 분기 기록 형태에 따라 모든 분기 명령들이 한 개의 BHR을 사용하는 전역 히스토리 레지스터(Global History Register) 방식과 각 분기 명령이 한 개의 BHR을 갖는 어드레스 단위 히스토리 테이블(Per Address History Table) 방식이 있다. 전역 히스토리 레지스터 방식은 한 개의 BHR을 사용하여 PHT를 인덱스하기 때문에 한 조건분기 명령에 대한 예상이 다른 분기 명령의 수행 결과에 의해 간섭(interference)을 받게 되나, 하드웨어 구조가 간단하고 분기 명령들이 연관성(correlation)을 많이 갖기 때문에 분기예상 정확도를 높일 수 있다. 그러나 연관이 없는 분기명령들에 의해서 부정적 간섭(negative interference)이 발생하여 분기예상 정확도를 감소시킨다[1].

2.2 Agree 분기예상 방법

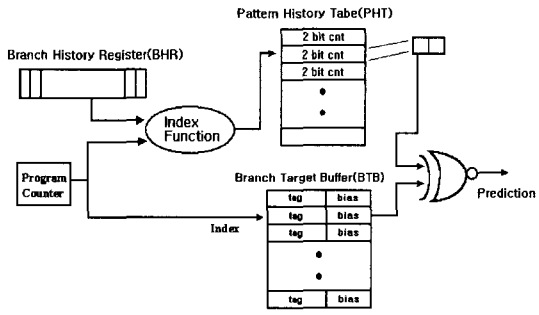
TLAT 분기예상 방법에서는 PHT의 엔트리 수가 제한되기 때문에 명령어 스트림에서 두 개 이상의 연관없는 분기 명령들이 동일 PHT 엔트리에 사상(mapping) 될 수 있다. 이러한 경우 한 분기 명령의 결과가 다른 분기 명령의 분기예상에 영향을 미치게 되는 간섭(interference)을 유발하게 된다.

한 분기 명령의 결과가 다른 분기 명령의 분기예상에 영향을 미치지 않는 경우를 중립적 간섭(neutral interference), 간섭이 없으면 잘못 예상될 것을 간섭에 의해 올바른 예상을 하도록 영향을 미치는 경우를 긍정적 간섭(positive interference), 한 분기 명령의 결과가 다른 분기예상을 틀리게 하는 경우를 부정적 간섭(negative interference)이라 한다. 부정적 간섭은 분기예상 정확도를 감소시키는 원인이 되므로 간섭의 수를 줄여주기 위한 여러 가지 연구들이 시도되었다.

기존의 연구에서는 부정적 간섭을 해결하기 위한 방법으로 3가지 방법을 사용하고 있다[1]. 첫째, PHT의 엔트리 수를 증가시켜 충돌하는 분기 명령을 PHT의 다른 엔트리에 사상시킨다. 둘째, 엔트리에 널리 분산될 수 있는 인덱싱 방법을 사용하여 부정적 간섭을 줄인다. 셋째, 분기 명령들을 분류하여 다른 예상기법을 사용함으로써 간섭을 줄이는 방법이다.

Agree 분기예상 방법은 전체 간섭의 수를 감소시켜 부정적 간섭의 수를 줄이는 기존의 방법들과는 달리 부정적 간섭을 긍정적 간섭이나 중립적 간섭으로 변환시켜 예상 정확도를 향상시키는 방법이다[1,12].

그림 2는 Agree 분기예상 방법의 구조를 나타낸다. Agree 분기예상 방법은 가장 좋은 결과를



(그림 2) Agree 분기예상 방법의 구조

예상하기 위하여 BTB의 각 엔트리에 1비트의 bias 비트를 추가하여 예상된 방향으로 분기할 것인지 아닌지를 결정한다. TLAT 분기예상 방법에서는 2비트 포화카운터가 taken/not-taken 분기 결과에 따라 증가/감소하여 부정적 간섭이 발생할 가능성이 있는 반면, Agree 분기예상 방법에서는 분기의 방향이 bias 비트와 일치하면 카운터를 증가시키고, bias 비트와 일치하지 않으면 카운터 값을 감소시킨다. 즉, 동일 엔트리를 사용하는 두 분기 명령의 bias 비트가 잘 설정되었다면 카운터를 같은 방향으로 갱신하여 부정적 간섭을 효과적으로 감소시킨다.

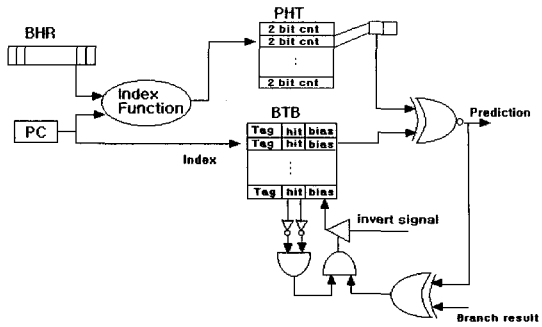
Agree 분기예상 방법에서는 bias 비트를 사용함으로써 부정적 간섭을 긍정적 간섭이나 중립적 간섭으로 변경하여 예상 정확도를 높일 수 있으나, bias 비트는 한번 설정되면 해당 분기 명령이 BTB에서 제거된 후 다시 할당되기 전까지는 동일 값을 유지하게 되므로 잘못 설정하게 되면 표 1에 보이는 바와 같이 오히려 부정적 간섭을 유발시킨다는 문제점이 있다[12].

3. 제안한 분기예상 방법

3.1 Hit 분기예상 방법

Agree 분기예상 방법에서는 부정적 간섭을 긍정적 간섭이나 중립적 간섭으로 변화시켜 예상 정확도를 개선시켰으나, bias 비트를 잘못 설정하게 되면 오히려 부정적 간섭이 증가될 수 있다는 문제점이 있다. 부정적 간섭은 bias 비트를 잘못 설정한 결과이며, 이러한 부정적 간섭을 제거하기 위해서는 bias 비트를 동적으로 변경시킬 수 있는 방법이 필요하다.

본 논문에서는 부정적 간섭을 감소시켜 주기 위해 그림 3과 같이 BTB에 hit 비트를 추가함으로써 bias 비트를 동적으로 변경시켜 예상 정확도를 향상시키는 Hit 분기예상 방법을 제안한다.

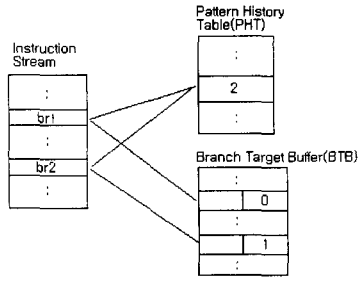


(그림 3) Hit 분기예상 방법의 구조

제안한 Hit 분기예상 방법에서는 분기 명령이 반입되는 시간에 그 명령의 주소로 BTB를 액세스하여 해당 명령의 분기 정보가 있는 엔트리를 찾는다. 분기예상은 명령의 주소와 BHR의 내용을 exclusive-OR하여 PHT에 대한 인덱스를 생성하고, 해당 엔트리의 카운터 값과 BTB의 bias 값을 비교하여 예상하게 된다. 엔트리의 카운터 값이 '2'이상일 때 BTB의 bias 값이 '1'이면 taken으로 예상하고 '0'이면 not-taken으로 예상하며, 카운터 값이 '1'이하일 때는 반대로 BTB의 bias 값이 '1'이면 not-taken으로 예상하고 '0'이면 taken으로 예상한다.

Hit 분기예상 방법에서는 BTB 엔트리에 2비트의 hit 비트를 추가하여 bias 비트를 동적으로 변경시킴으로써 bias 비트 설정의 문제점을 해결한다. 분기 명령이 BTB에 할당될 때 hit 비트 카운터는 '1'로 설정되고, 분기예상이 틀리면 '1' 감소하고 예상이 적중되면 '1' 증가시킨다. hit 비트 카운터가 '0' 값을 가지고 있을 때 분기예상이 틀리면 bias 비트가 잘못 설정된 것이므로 bias 비트를 반전시켜 올바른 bias 비트가 설정되도록 하여 부정적 간섭을 감소시킨다. 또한 제안한 Hit 분기예상 방법은 bias 비트를 사용함으로써 새로운 분기 명령의 분기예상 준비기간(warm-up) 동안에도 예상 정확도를 향상시킬 수 있다.

3.2 hit 비트의 설정



(그림 4) Agree 분기예상 방법에 대한 two-level에서의 간섭

hit 비트의 올바른 설정을 위해 다음과 같은 환경을 가정한다. 명령어 스트림에서 두 분기 명령 br1과 br2가 동일한 PHT 엔트리를 사용한다고 가정한다. 그림 4와 같이 br1과 br2의 bias 비트가 각각 '0'과 '1'로 설정되어 있고, br1과 br2의 분기 결과는 모두 taken이고 br1과 br2가 사용하는 PHT 엔트리의 2비트 포화 카운터가 '2'로 설정되었다고 가정한다.

먼저 br1이 인출되었다면 bias 비트가 '0'이고 카운터가 '2'이므로 not-taken으로 예상되므로, 분기 결과(taken)와 일치하지 않고 PHT 엔트리의 카운터는 분기 결과가 bias 비트와 일치하지 않으므로 감소하여 '1'이 된다. 이후에 br2가 인출되었다면 bias 비트가 '1'이고, 카운터가 '1'이므로 not-taken으로 예상되어, 잘못된 분기예상이 발생하고 PHT 엔트리의 카운터는 분기 결과와 bias 비트가 일치하므로 증가되어 '2'가 된다. 이후에도 br1과 br2가 동일 PHT 엔트리를 사용할 때 서로에게 부

(표 1) bias 비트의 고정적 지정

	br1					br2				
	bias bit	saturating counter	분기 예상	분기 결과	예상 적중	bias bit	saturating counter	분기 예상	분기 결과	예상 적중
1	0	2	nt	t	x	1	1	nt	t	x
2	0	2	nt	t	x	1	1	nt	t	x
3	0	2	nt	t	x	1	1	nt	t	x
4	0	2	nt	t	x	1	1	nt	t	x
5	0	2	nt	t	x	1	1	nt	t	x
6	0	2	nt	t	x	1	1	nt	t	x
7	0	2	nt	t	x	1	1	nt	t	x
8	0	2	nt	t	x	1	1	nt	t	x
9	0	2	nt	t	x	1	1	nt	t	x

(표 2) bias 비트의 동적 지정

	br1						br2					
	bias bit	saturating counter	hit bit	분기 예상	분기 결과	예상 적중	bias bit	saturating counter	hit bit	분기 예상	분기 결과	예상 적중
1	0	2	1	nt	t	x	1	1	1	nt	t	x
2	1	2	0	t	t	o	0	3	0	nt	t	x
3	1	2	1	t	t	o	0	3	1	nt	t	x
4	1	2	2	t	t	o	1	3	0	t	t	o
5	1	3	3	t	t	o	1	3	1	t	t	o
6	1	3	3	t	t	o	1	3	2	t	t	o
7	1	3	3	t	t	o	1	3	3	t	t	o
8	1	3	3	t	t	o	1	3	3	t	t	o
9	1	3	3	t	t	o	1	3	3	t	t	o

정적 간섭이 발생하여 표 1에서와 같이 잘못된 분기예상이 반복된다.

이러한 문제점을 해결하기 위해 bias 비트를 동적으로 변경시키도록 hit 비트를 사용하는 경우를 표 2에서 볼 수 있다. 제안한 Hit 분기예상 방법에서는 hit 비트를 추가하여 bias 비트를 동적으로 변경시킴으로써 예상적중을 높일 수 있음을 보여주고 있다.

hit 비트는 '1'로 초기화되었고, 분기예상과 결과가 일치하지 않으면 1씩 감소하게 된다. hit 비트가 '0' 값을 가지고 있을 때 분기예상과 결과가 일치하지 않으면 bias 비트를 반전시키게 되며 hit 비트는 '1' 값으로 변경한다. 즉, hit 비트가 '0'인 상태에서 예상이 다시 잘못된 경우에는 bias 비트가 잘못 설정된 것으로 간주해 bias 비트를 동적으로 반전시켜 올바른 예상이 이루어지도록 한다.

4. 시뮬레이션 및 실험 결과

4.1 시뮬레이션 방법

본 논문에서 제안한 Hit 분기예상 방법의 우수성을 확인하기 위해 execution-driven 시뮬레이터인 Shade를 사용하여 Sun-SPARC 시스템에서 시뮬레이션하였다. SPEC92int 벤치마크(표3)를 사용하여 성능비교를 하였으며, 모든 벤치마크는 Unix Sun-SPARC C 컴파일러에서 최적화 옵션을 사용하여 컴파일 하였다.

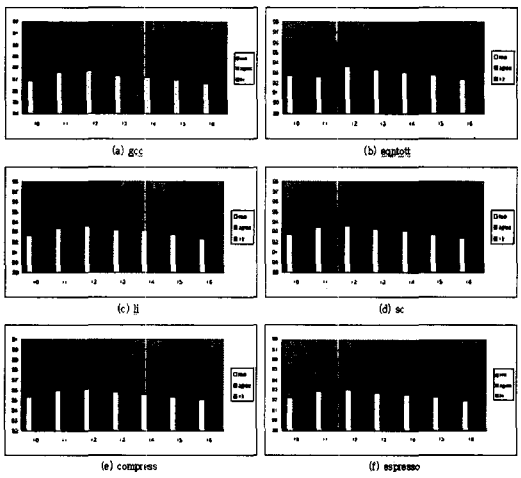
(표 3) SPEC92int 벤치마크 프로그램

benchmark	execution file	input data	Description
008.espresso	espresso	bca.in	Minimizes Boolean functions
022.li	xlisp	li_input.lisp	Lisp interpreter
023.eqntott	eqntott	int_pri_3.eqn	Translate a boolean equation into a truth table
026.compresso	compress	in	Performs data compression
072.sc	sc	loada1	Performs computations within a Unix spreadsheet
085.gcc	gcc	cexp.i	GNU C compiler

BTB 크기는 1K direct map 캐시 구조로 고정하고 BHR의 비트수를 10비트에서 16비트로 증가시키면서 본 논문에서 제안한 Hit 분기예상 방법(hit)과 TLAT 분기예상 방법(two), Agree 분기예상 방법(agree)에 대해 예상 정확도를 측정하였다. 각 분기예상 방법은 PC와 BHR의 내용을 exclusive-OR하여 PHT의 인덱스로 사용하는 gshare 방식을 사용하였으며, PHT 엔트리의 각 포화 카운터는 2로 초기화하였다.

4.2 예상 정확도의 측정

그림 5는 6개의 SPEC92int 벤치마크에 대해 BHR의 크기를 변화시키며 벤치마크 프로그램이 완료될 때까지 각 분기예상 방법의 예상 정확도를 측정한 결과이다. 그림 5에서 x축은 BHR의 비트수를, y축은 예상 정확도를 나타낸다.



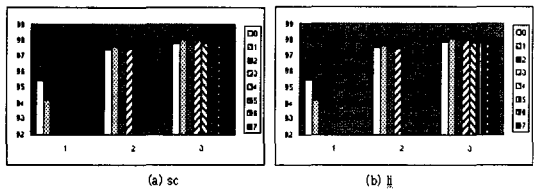
(그림 5) SPEC92int 벤치마크에서의 예상 정확도

모든 벤치마크들에서 제안된 Hit 분기예상 방법이 부정적 간섭을 효과적으로 줄여줌으로써 TLAT 분기예상 방법에 비해 최소 2.5%에서 최대 5.1%의 예상 정확도가 향상되었으며, Agree 분기예상 방법에 비해 최소 0.2%에서 최대 0.6%의 예상 정확도가 향상되었다. 또한 BHR의 비트 수를 증가시키면 PHT의 엔트리 수가 증가하므로 전체 간섭 수가 감소하게 되어 부정적 간섭이 발생할 가능성이 감소되기 때문에 예상 정확도가 높아짐을 측정할 수 있었다. 대부분의 벤치마크에서 BHR 비트 수가 13 비트 이상일 경우 Hit 분기예상 방법이 Agree 분기예상 방법보다 2 비트 적은 BHR 비트 수를 사용하여도 더 높은 예상 정확성을 나타내고 있다. 이는 1K 엔트리의 BTB사용시 hit 비트를 추가할 경우 BTB 크기가 증가(2비트*1K 증가)하지만 BHR 비트 수 감소에 따라 전체 PHT 엔트리가 1/4로 감소(2비트*3K 감소)하여 적은 하드웨어로 더 좋은 성능을 얻을 수 있었다.

그림 5에서 two로 표현된 TLAT 분기예상 방법의 경우에는 모든 벤치마크에서 12비트 이후에 오히려 예상 정확도가 감소하는 경향이 있는데 이것은 BHR의 비트 수가 증가하면 준비(warm-up) 시간이 증가함으로 예상 정확도가 감소되기 때문이다.

4.3 hit 비트의 설정

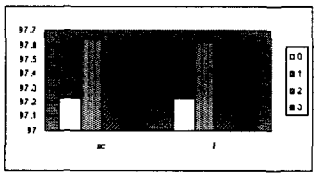
그림 6은 가장 좋은 예상 정확도를 보이는 sc와 li 벤치마크 프로그램에 대해 hit 비트 수를 1부터 3까지 변화시키며 예상 정확도를 측정한 결과이다. BHR의 비트 수는 15 비트이며, hit 비트



(그림 6) hit 비트 수를 변화시키며 측정한 예상 정확도

의 초기 설정값과 반전시 설정값을 비트 수에 따라 0부터 최대 7까지로 변화시켜 보았다.

hit 비트 수가 커지면 예상 정확도도 높아지는 것을 알 수 있으나 2 비트 이후에는 그 차이가 근소하므로 제안한 Hit 분기예상 방법에서는 hit 비트 수를 2 비트로 고정하도록 하였다. hit 비트의 초기 설정값과 반전시 설정값을 1로 지정한 경우가 가장 높은 예상 정확도를 보이고 있다. 또한, 그림 7은 hit 비트 수를 2로, hit 비트 반전값을 1로 고정했을 경우에 hit 비트 설정값을 0에서 3까지 변화시키며 측정한 예상 정확도를 보여준다. hit 비트의 초기값을 1로 지정하였을 경우의 예상 정확도가 가장 높은 것으로 측정되었다.



(그림 7) hit 비트의 설정값에 따른 예상 정확도

5. 결 론

TLAT 분기예상 방법은 분기 명령들의 연관 관계를 이용하여 높은 예상 정확도를 갖는 것으로 알려졌으나, PHT 엔트리 수의 제약 때문에 서로 다른 분기 명령이 PHT의 동일 엔트리를 사용함으로써 부정적 간섭이 발생하게 된다는 문제점이 있다. Agree 분기예상 방법에서는 이러한 부정적 간섭을 긍정적 간섭으로 변환하기 위해 BTB에 bias 비트를 추가하여 예상 정확도를 높였으나, bias 비트를 잘못 선택하는 경우에는 오히려 부정적 간섭이 증가하게 된다.

본 논문에서는 BTB 엔트리에 2비트의 hit 비트를 추가함으로써 부정적 간섭을 제거하는 새로운 동적 분기예상 기법인 Hit 분기예상 방법을 제안하였다. Hit 분기예상 방법은 hit 비트를 이용해 bias 비트가 잘못 설정되었을 경우를 검출하게 되면 수행시간에 동적으로 bias 비트를 수정하여 부정적 간섭을 제거함으로써 예상 정확도를 증가시켰다.

Sun-SPARC 시스템 상에서 SPEC92int 벤치마크 프로그램들에 대해 Shade를 사용하여 시뮬레이션한 결과 본 논문에서 제안한 Hit 분기예상 방법이 기존의 동적 분기예상 방법인 TLAT 분기예상 방법과 Agree 분기예상 방법보다 부정적 간섭을 효과적으로 줄여 각각 평균 4%, 0.4% 예상 정확도를 높일 수 있음을 보였다. 또한 대부분의 벤치마크에서 BHR 비트 수가 13 비트 이상일 경우 Hit 분기예상 방법이 Agree 분기예상 방법보다 2 비트 적은 BHR 비트 수를 사용하여도 더 높은 예상 정확성을 나타내고 있어 hit 비트를 추가하여도 전체 PHT 엔트리 수는 1/4로 감소하므로 적은 하드웨어로 더 좋은 성능을 얻을 수 있었다.

참 고 문 헌

[1] Eric Sprangle, Robert S. Chappell, Mitch Alsup, Yale N. Patt, 'The Agree Predictor : A Mechanism for Reducing Negative Branch History Interference', Proc. of 24th ISCA, Vol. 25, No.2, pp.284-291, May 1997.

[2] T.Y. Yeh and Y.N. Patt, 'Two-Level Adaptive Training Branch Prediction', Proc. of Micro-24, pp.51-61, 1991.

[3] B. K. Bray and M. J. Flynn, 'Strategies for Branch Target Buffers', Proc. of Micro-24, pp. 42-50, November, 1991.

[4] Lee, J. K. F and A. J. Smith, 'Branch Prediction Strategies and Branch Target Buffer Design', IEEE Computer, Vol. 17, pp.6-22, January, 1984.

[5] Shin-Tai Pan, Kimming So, Joseph T. Rahmeh, 'Improving the Accuracy of Dynamic Prediction Using Branch Correlation', ASPLOS V, pp. 76-84, 1992.

[6] B. Calder and D. Grunwald, 'Fast and Accurate Instruction Fetch and Branch Prediction', Proc. of 21th ISCA, pp.2-11, April, 1994.

[7] McFarling S. and J. Hennessy, 'Reducing the Cost of Branches', Proc. of 13th ISCA, pp.396-403, 1986.

[8] Talcott A. R. and R. C. Wood, 'The Influence of Branch Prediction Table Interference on Branch prediction Scheme Performance', International Conference on Parallel Architectures and Compilation Technique, 1995.

[9] Young C. and M. D. Smith, 'A Comparative Analysis of Schemes for Correlated Branch Prediction', Proc. of 22th ISCA, pp.276-286, 1995.

[10] M. Evers, S. J. Patel, R. S. Chappell, Y. N, Patt, 'An Analysis of Correlation and Predictablility : What Makes Two-Level Branch Predictors Work', In Proc .ISCA-25, pp.52-61, 1998.

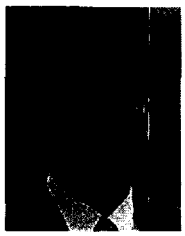
[11] Timothy H. Heil, 'Improving Branch Predictors by Correlating on Data values', Proc. 32th Intl. Symp. on Microarchitecture, Nov. 1999.

[12] Avinoam Nomik Eden, Trevor Mudge, 'The YAGS Branch Prediction Scheme', Proc. 31st Intl. Symp. on Microarchitecture, pp.69-77, Dec. 1998.

◐ 저 자 소 개 ◑

박 홍 준

1984년 아주대학교 전자계산학과 공학사
1987년 한양대학교 전자계산학과 공학석사
1997년~현재 수원대학교 전자계산학과 박사과정
1994년~현재 극동정보대학 전산정보처리과 조교수
관심분야 : 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.
관심분야 : 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