세트 연관 캐쉬를 사용한 2단계 적응적 분기 예측

심 원†

요 약

조건부 분기 명령어는 분기 벌칙을 야기함으로써 명령어 수준의 병렬도 향상에 제약을 가한다. 고성능 슈퍼스칼라 프로세서의 등장으로 인해, 정확한 분기 예측의 중요성은 더욱 높아지고, 이를 위해 동적 분기 예측의 일종인 2단계 적응적 분기 예측(2-level adaptive branch prediction) 방식이 개발되었다. 그러나 2단계 적응적 분기 예측이 상당히 높은 예측 정확도를 보여주고 있음에도 불구하고, 정확도에 따른 비용이 기하급수적으로 증가하는 등의 문제점을 가지고 있다. 본 논문에서는 2단계 적응적 분기 예측의 이러한 문제점을 개선하기 위하여 세트 연관 캐쉬를 이용한 캐쉬 상관 분기 예측기(cached correlated branch predictor)를 제안하고, 기존의 방식에 비해 예측의 정확도는 증가하고, 비용은 줄어든 것을 시뮬레이션을 통하여 확인한다. 세트 연관 예측기의 경우에 전역과 지역 방식의 가장 좋은 예측 실패율은 각각 5.99%, 6.28%이며, 이는 종래의 2단계 적응적 분기 예측 방식에서의 가장 좋은 결과인 9.23%, 7.35%에 비해 각각 54%, 17% 향상된 결과이다.

2-Level Adaptive Branch Prediction Based on Set-Associative Cache

Won Shim†

ABSTRACT

Conditional branches can severely limit the performance of instruction level parallelism by causing branch penalties. 2-level adaptive branch predictors were developed to get accurate branch prediction in high performance superscalar processors. Although 2-level adaptive branch predictors achieve very high prediction accuracy, they tend to be very costly. In this paper, set-associative cached correlated 2-level branch predictors are proposed to overcome the cost problem in conventional 2-level adaptive branch predictors. According to simulation results, cached correlated predictors deliver higher prediction accuracy than conventional predictors at a significantly lower cost. The best misprediction rates of global and local cached correlated predictors using set-associative caches are 5.99% and 6.28% respectively. They achieve 54% and 17% improvements over those of the conventional 2-level adaptive branch predictors.

키워드 : 분기 예측(Branch Prediction), 동적 분기 예측(Dynamic Branch Prediction), 2단계 적응적 분기 예측(2-Level Adaptive Branch Prediction), 캐쉬 상관 분기 예측(Cached Correlated Branch Prediction)

1. 서 론

고성능 프로세서는 분기 명령어(branch instruction)가 수행될 때 발생하는 파이프라인 처리 중지를 피하기 위해 동적 분기 예측(dynamic branch prediction)을 주로 사용한다. 과거 분기 명령어의 처리 결과에 바탕을 둔 BTC(branch target cache)를 이용한 종래의 방식의 예측 정확도는 80~95% 정도이다[1]. 최근 슈퍼스칼라(superscalar) 프로세서의 등장으로 분기 예측의 정확도는 더욱 중요하게 되었다. 종래의 스칼라 프로세서의 경우에는 분기 예측이 잘못되었을 때 한 두 사이클 정도를 손해보게 되어 한 두 명령어만 다시 실행하면 되는데 반해, 슈퍼스칼라 프로세서의 경우 훨씬 많은 명령어를 다시 수행하게 되어 성능의 손실이 크게 늘어나게 된다. 따라서, 1990년대 Yeh & Patt와 Pan, et al에 의해 제안된 2단계 적응적 분기 예측(2-level adaptive branch prediction)을 이용한 동적 분기 예측 방식이 각광을 받게 되었다[2, 3].

2단계 적응적 분기 예측은 상당히 높은 예측 정확도를 보여주고 있음에도 불구하고, 몇가지 문제점을 가지고 있다. 첫째로, 2단계 적응적 분기 예측의 정확도를 올리려면 대량의 예측 계수기(prediction counter)와 과거의 분기 패턴표 즉, 패턴 히스토리 테이블(PHT, pattern history table)이 필요한데 이는 대단히 큰 용량의 메모리를 필요로 한다. 둘째로, 대개의 경우 메모리를 줄이기 위해, 몇 개의 분기 명령어가 예측 계수기를 공유함으로써 분기간에 간섭이 발생한다는 것이다. 셋째로, 대량의 예측 계수기는 과도한 초기화 과정이 필요하다는 것이다. 과거 분기의 내용을 저장한 히스토리(history)가 증가하면 할수록 초기화 과정도 더욱 늘어나게 되므로 초기화 부담으로 인해 분기 내용의 히스토리를 제한하게 된다.

이러한 세 가지 단점 즉 비용, 간섭, 초기화의 문제를 개선하기 위해 캐쉬 상관 분기 예측기(cached correlated branch predictor)가 제안되었다[4, 5]. 그러나, [4, 5]에서는 비용이 많이 들어 현실적으로 구현하기 어려운 완전 연관(fully-associative) 캐쉬가 사용되었다.

본 논문에서는 완전 연관 캐쉬 대신 현실적으로 구현 가능한 세트 연관(set-associative) 캐쉬를 이용한 예측기가 기존의 2단계 적응적 분기 예측 방식에 비해 예측의 정확도는 높이고, 비용은 줄어든 것을 시뮬레이션을 통하여 확인한다. 캐쉬를 이용한 분기 예측기에서는 첫째, 실제로 사용

† 종신회원 : 서울산업대학교 컴퓨터공학과 교수
논문접수 : 2002년 8월 22일, 심사완료 : 2002년 10월 22일

되는 예측 계수기만 유지하므로 비용을 절약할 수 있다. 둘째, 분기 예측에 분기 명령어간의 간섭을 배제하고 그 분기 명령어만의 과거 분기 정보에 의해 예측한다. 셋째, 단 하나의 분기만 일어나면 초기화될 수 있는 디폴트(default) 예측기를 사용함으로 인해 예측 계수기의 초기화 과정에서 발생하는 부정확한 예측을 방지할 수 있다.

2. 2단계 적응적 분기 예측기

2단계 적응적 분기 예측은 이미 처리된 분기 명령어의 처리 결과에 따라 현재의 분기 여부가 영향을 받는다는 사실에 바탕을 두고 있다. 2단계 적응적 분기 예측은 분기 예측을 위해 두 단계의 분기 히스토리 정보를 이용한다. 첫 번째 정보는 최근에 실행된 k개의 분기 명령어의 처리 결과로, 분기 히스토리 레지스터(branch history register)에 비트 패턴으로 저장된다. 예측의 두 번째 단계는 k개의 분기 명령어에 대한 j비트로 구성된 특정한 비트 패턴의 히스토리를 가지고 있으며, 보통 PHT라고 부른다. 첫 번째 정보는 실제 분기 예측을 제공하는 두 번째 단계의 PHT에 들어 있는 예측 계수기(prediction counter)를 지정하는 인덱스로 사용된다. 여기서 예측 계수기는 2비트 포화 계수기(saturation counter)로 구성한다.

2단계 적응적 분기 예측기는 보통 Yeh & Patt가 제안한 분류법으로 구별한다[2]. 자주 사용되는 여섯 가지 기본 구성은 GAg, GAp, GAs, PAg, PAp, PAs 등이다. 첫 문자는 첫째 단계의 동작 구조를 나타내고, 끝의 문자는 둘째 단계의 동작 구조를 나타내며, 가운데 A는 동적인 예측을 의미하는 적응성(adaptive)을 나타낸다. GAg, GAp, GAs는 전역 분기 히스토리(global branch history)를 사용하고, PAg, PAp, PAs는 지역(local) 분기 히스토리를 사용한다.

GAg 방식은 최근의 k개의 분기 명령어의 처리 결과를 기록하는 하나의 전역 히스토리 레지스터와 예측 계수기로 구성된 하나의 PHT를 사용한다. 예측을 하기 위해서는 1단계 전역 히스토리 레지스터가 2단계 PHT에 있는 예측 계수기를 지정하는 인덱스로 사용된다. 예측 계수기는 2비트 포화 계수기로 구성된다. 분기 예측을 하기 위해서는 최근 k개의 분기 명령어의 처리 결과와 다음 분기 명령어 사이의 상관 관계를 찾아야한다. 전역 히스토리 레지스터와 PHT에 있는 예측 계수기는 실제 분기 결과가 나오는대로 갱신(update)된다. 또한 분기 명령어가 분기했을 때 수행할 명령어를 가리키는 BTC(branch target cache)를 유지하여야 한다. GAg에서는 모든 분기 명령어가 PHT를 공유함으로 한 분기 명령어의 처리 결과가 다른 분기 명령어의 처리에 영향을 미치는 분기 명령어 사이의 간섭이 발생한다. 이러한 간섭이 전역 예측기의 성능을 저하시키는 요인인데 이를 개선하려면 히스토리 레지스터의 길이를 늘려야 한다. 그러면 PHT의 계수기 수가 증가하게 되고, 따라서 PHT의 비용과 초기화 과정으로 인해 부정확한 예측이 초기에 증가하게 된다. 결과적으로 초기의 부정확한 예측 증가로 인해 히스토리 레지스터의 증가 효과를 상쇄하고 더 이상 예측의 정확도가 증가하지 않게 된다.

GAp는 [3]에서 처음 제안하였으며, 상관 분기 예측(correlated branch prediction)이라고도 한다. GAp 방식은 GAg 방식과 같이, 최근 k개의 분기 명령어의 처리 결과를 기록하는 하나의 전역 히스토리 레지스터만 가지고 있다. 모든 분기 명령어가 PHT를 공유함으로 생기는 분기 명령어 사이의 간섭을 배제하기 위해, 하나의 PHT 대신에 각 분기 명령어마다 독립된 PHT를 가지고 있다. 따라서, 프로그램 카운터(PC, program counter)와 히스토리 레지스터가 PHT에 있는 예측 계수기를 지정하는 인덱스로 사용된다. 이러한 방식은 분기 명령어 사이의 간섭을 배제할 수는 있지만 굉장히 큰 PHT를 필요로 한다. 예를 들어, 30비트 PC의 경우, 2^{30+k}개의 2비트 계수기가 필요하다. 실제로는 PHT의 크기를 줄이기 위해 몇 개의 분기 명령어가 PHT를 공유하도록 하고, 이러한 구성을 GAs 방식이라고 한다. GAs 방식에서는 간섭이 완전히 제거되지는 못하고 부분적으로 제거된다.

1991년에 Yeh & Patt에 의해 제안된 2단계 적응적 분기 예측은 PAg 방식으로 분류된다[6]. PAg 방식은 분기 명령어마다 각각의 지역 히스토리 레지스터와 공통의 PHT를 가지고 있다. 따라서 각 분기 명령어의 처리 결과에 따라 해당하는 각각의 분기 예측이 이루어진다. PAg 방식도 서로 다른 분기 명령어가 하나의 PHT를 공유하므로 간섭이 발생한다. 이러한 간섭은 PAg 방식에 여러개의 PHT를 사용함으로써 제거할 수 있다. 분기 명령어마다 지역 히스토리 레지스터와 독자의 PHT를 갖추면 PAp 구성이 된다.

PAp 방식도 GAp와 마찬가지로 PHT의 크기가 증가하면 초기화 문제가 생긴다. 이 때문에 몇 개의 분기 명령어를 묶어서 하나의 PHT를 제공하는 PAs 구성이 사용된다. PAg나 PAs 방식은 모두 두 번의 순차적인 메모리 접근이 필요하다. 첫 번째는 지역 히스토리 레지스터를 구하기 위한 BTC 접근이고, 두 번째는 분기 예측을 위한 PHT에 대한 접근이다. 예측기의 성능을 높이기 위해서는 일단 한 분기 명령어가 정해지면 한 사이클 내에 예측을 완료할 수 있으면 좋다. 이를 위해 현재의 분기 명령어 처리 결과를 BTC에 저장한다면, 다음 번 분기 명령어의 예측은 현재 분기 명령어의 갱신 과정에서 결정되며, 이의 처리가 종료되는 즉시 알 수 있다[7].

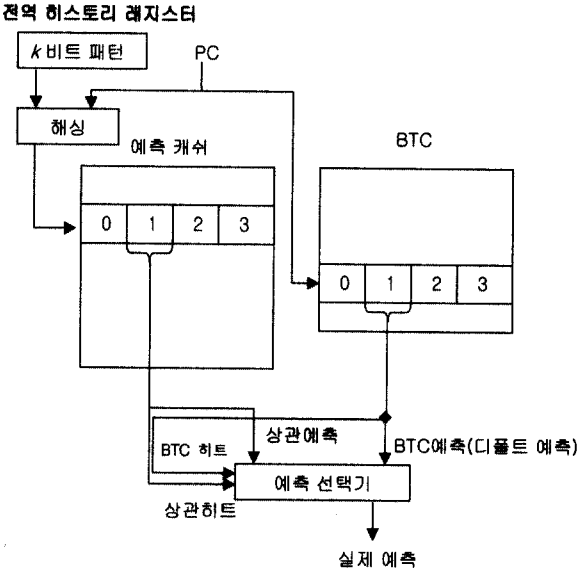
3. 캐쉬 상관 예측

2단계 적응적 분기 예측기를 구성하는데 비용이 많이 드는 것은 과도한 2차 PHT를 구성해야 하기 때문이다. 캐쉬 상관 예측기에서 1차 단계는 2단계 적응적 분기 예측기와 변동이 없고, 2차 단계의 PHT를 캐쉬로 대체한 것이다. 예측 캐쉬에서는 실제 예측에 사용되는 계수기만 유지함으로써, 종래의 분기 예측기의 PHT에서 사용되지 않는 계수기에 들어가는 비용을 절약할 수 있다. 그러므로 캐쉬 상관 예측기에서는 종래의 2단계 적응적 분기 예측기와는 달리, 예측 캐쉬(prediction cache)의 크기는 히스토리 레지스터 길이의 함수가 아니고, 실제 예측에 사용되는 예측 계수기의 크기에 의해 결정된다.

본 논문에서는 전역 히스토리 레지스터와 지역 히스토리 레지스터를 사용하는 두 가지 방식의 캐쉬 상관 예측기를 구성한다. 예측 캐쉬를 완전 연관(fully-associative) 사상으로 구성하면 비용이 너무 많이 들어가므로, 현실적인 4원 세트 연관(4-way set-associative)으로 구성한다. 이 경우에는 고려해야 할 사항이 있다. 보통 BTC는 PC의 하위 비트를 이용하여 접근하는데, 예측 캐쉬에 이를 적용하는 경우에는 문제가 생긴다. 예를 들어, 4원 세트 연관 캐쉬의 경우, 각 PC당 4엔트리로 제한된다. k비트의 히스토리 레지스터가 사용된다면, 각 PC당 최대 2^k개의 엔트리가 필요하게 된다. 물론, 히스토리 레지스터 패턴의 많은 부분이 발생하지 않는다 하더라도 과도한 충돌로 인해 문제가 생긴다. 대안으로 히스토리 레지스터를 이용해 예측 캐쉬에 접근하더라도 문제는 있다. 히스토리 레지스터의 비트 수가 적을 때는 예측 캐쉬의 아주 일부만을 사용하게 된다. 히스토리 레지스터의 비트 수가 커지면 예측 캐쉬에서 충돌이 자주 일어나게 된다. 따라서 본 논문에서는 PC와 히스토리 레지스터를 해싱(hashing)하여 인덱스를 구했는데, 해싱 함수로는 XOR(exclusive OR)를 사용했다. 또한 예측 캐쉬의 크기가 변함에 따라 예측 캐쉬와 같은 크기의 효율적인 인덱스를 구하기 위해 초기 인덱스를 구하여 예측 캐쉬의 크기에 맞게 절단하고 XOR를 반복하여 처리하였다. 예를 들어, 히스토리 레지스터가 16비트, PC가 32비트, 예측 캐쉬의 용량이 4K 엔트리라면 먼저 히스토리 레지스터와 PC를 연결(concatenate)하여 48비트 데이터로 만든다. 이 48비트 데이터를 예측 캐쉬의 용량에 맞게 12비트 단위로 자른 다음 반복적인 XOR 연산을 하여 최종의 12비트 인덱스를 구한다.

3.1 전역 캐쉬 상관 예측기

(그림 1)은 4원 세트 연관(4-way set-associative) 캐쉬를 이용한 전역 캐쉬 상관 예측기(global cached correlated predictor)의 구성이다. 예측 캐쉬의 각 엔트리는 PC, 히스토리 레지스터, 2비트 예측 계수기, 유효비트, LRU(least recently used) 부분 등으로 구성된다. 분기 목적지 주소(branch target address)를 나타내기 위한 BTC도 제공된다. BTC의 각 엔트리는 분기 목적지 주소, 2비트 예측 계수기, 유효비트, LRU(least recently used) 부분으로 구성된다. BTC는 PC 값으로 접근하고, 예측 캐쉬(prediction cache)의 인덱스는 PC와 전역 히스토리 레지스터를 해싱하여 얻는다. BTC에 미스(miss)가 나면 이미 처리한 분기 결과가 아직 없다는 뜻이며, 미리 정해진 '분기안함'(not taken)을 취한다. BTC에 히트(hit)가 되면, 다시 예측 캐쉬에 접근한다. 이때 히트가 되면 예측 캐쉬의 2비트 포화 계수기(saturation counter)로부터 예측을 구하고, 미스가 나면 디폴트 예측기(default predictor)인 BTC로부터 예측을 얻는다. 즉, BTC보다 예측 캐쉬에 우선을 두는 우선순위(priority) 메카니즘을 사용한다. BTC로부터의 예측은 과거에 처리한 모든 분기 명령어의 결과에 입각한 것이며, 예측 캐쉬로부터의 예측은 현재의 히스토리 레지스터 패턴에 따른 예측을 얻는 것이다. 일단 분기 명령어가 실행되어 결과가 최종 결정되면, 즉시 BTC와 예측 캐쉬에 있는 예측 계수기를 갱신한다. 어느 쪽이든 미스가 났던 경우는 이 새로운 엔트리를 추가하고 최종적으로 전역 히스토리 레지스터를 갱신한다. 이때 예측 캐쉬가 포화 상태이면 교체를 해야 하는데, 교체할 엔트리는 LRU 알고리듬에 의해 정한다.



(그림 1) 전역 캐쉬 상관 예측기

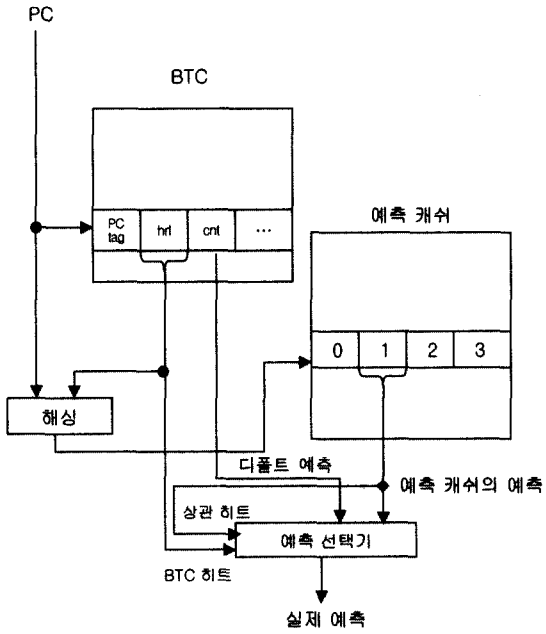
디폴트 예측기를 두게 되면 몇가지 이점이 있다. 첫째로 디폴트 예측기는 단 하나의 분기 명령어만 만나도 초기화된다. 반면에, k비트를 사용하는 히스토리 레지스터의 경우, 완전한 예측을 하기 위해서는 2^k개의 계수기가 모두 초기화되어야 한다. 디폴트 예측기를 둠으로써 초기의 잘못된 예측을 줄일 수 있다. 두 번째로 디폴트 예측기는 예측 캐쉬의 미스에 대한 영향을 최소화한다.

개별 예측이 가능한 예측기를 두개 이상 사용하는 혼합 예측기(hybrid predictor)도 있기는 하다[8, 9]. 혼합 예측기에서 예측을 얻을 경우, 과거의 성공률에 따른 동적인 선택 메카니즘이 필요하다. 그러나 본 시스템에서는 BTC는 예측 캐쉬에 미스가 났을 때만 이용하고, 언제나 예측 캐쉬에 우선을 두는 우선 순위 메카니즘을 사용한다.

3.2 지역 캐쉬 상관 예측기

지역 캐쉬 상관 예측기에서도 (그림 2)와 같이 PHT를 예측 캐쉬로 대체할 수 있다. 이 경우는 각 분기 명령어마다 히스토리 레지스터가 필요하므로, 각 BTC의 엔트리에 지역 히스토리 레지스터를 추가하여야 한다. BTC는 PC의 하위 비트들을 이용하여 접근한다. BTC에 히트가 되면 관련 히스토리 레지스터와 디폴트 예측을 얻을 수 있다. 예측 캐쉬는 히스토리 레지스터와 PC를 해싱을 하여 접근한다. 지역 예측기를 이용할 때의 한가지 문제는 BTC를 접근할 때 한번, 예측 캐쉬를 접근할 때 또 한번 등, 도합 두 번의 순차적인 접근이 필요하다는 것이다. 이 문제는 지역 예측기의 결과를 BTC에 바로 저장하면 해결할 수 있다. 분기 명령어의 실행 결과가 확정되

는 즉시, 히스토리 레지스터와 예측 캐쉬는 바로 갱신된다. 바로 이 순간에 다음에 만날 분기 명령어의 예측이 확정된다. 따라서 이 결과를 BTC에 저장하여 다음 분기 명령어의 예측에 사용한다면 한 사이클에 분기 예측이 가능하게 된다.



(그림 2) 지역 캐쉬 상관 예측기

4. 시뮬레이션 결과 및 검토

시뮬레이션을 통해 세트 연관 캐쉬 상관 예측기와 종래의 2단계 예측기의 성능을 전역과 지역의 경우로 나누어 비교하였다. Stanford 벤취마크(benchmark)를 사용하여 예측 실패율을 구하였는데, Stanford 벤취마크 프로그램은 bubble, matrix, perm, puzzle, queens, sort, tower, tree 등 8개의 정수 프로그램으로 되어 있다. 이는 SPEC 벤취마크보다 짧은 프로그램이므로, 각 분기 명령어의 실행 횟수가 적기 때문에 예측이 더 어려워지고 예측기의 초기화 문제가 더욱 중요하게 된다. 먼저 8개의 각 정수 프로그램별로 예측 실패율을 구하고, 각 구성 방식에 따라 이 8개의 값을 평균하여 히스토리 레지스터 크기에 따른 전체 예측 실패율을 구하였다. 시뮬레이션은 영국 Hertfordshire 대학에서 개발된 HSA (Hatfield Superscalar Architecture)를 이용하였다[10, 11].

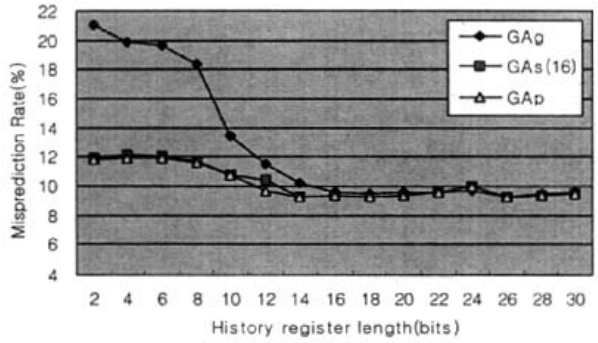
4.1 전역 예측기

<표 1>과 (그림 3)은 GAg, GAp, GAs 등 종래의 방식에 대하여, 히스토리 레지스터의 크기에 따른 예측 실패율을 나타낸 표와 그래프이다. 평균적인 예측 실패율은 히스토리 레지스터의 크기가 증가함에 따라 줄어들다가 9.5% 근처에서 더 이상 향상되지 않는다.

가장 좋은 예측 실패율은 히스토리 레지스터가 26비트인 GAs(16 세트) 방식의 9.23%이다. 일반적으로, GAg의 경우 히스토리 레지스터 16비트, GAp와 GAs의 경우 히스토리 레지스터 14비트부터는 레지스터 길이 증가의 효과가 별로 없다.

〈표 1〉 종래의 전역 2단계 적응적 예측기의 예측 실패율

HR	GAg	GAs(16)	GAp
2	21.01	11.91	11.79
4	19.85	12.11	11.98
6	19.69	12.03	11.91
8	18.36	11.68	11.65
10	13.47	10.70	11.76
12	11.46	10.44	9.69
14	10.19	9.29	9.29
16	9.54	9.34	9.33
18	9.48	9.28	9.28
20	9.58	9.40	9.40
22	9.62	9.52	9.52
24	9.72	10.00	10.00
26	9.30	9.23	9.23
28	9.45	9.39	9.39
30	9.58	9.51	9.51



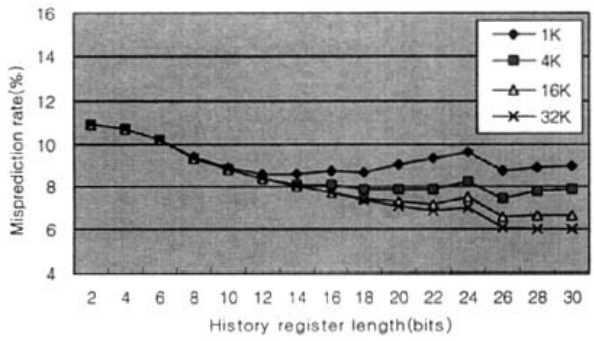
(그림 3) 종래의 전역 2단계 적응적 예측기의 예측 실패율

히스토리 레지스터의 길이가 증가하는 데에도 예측의 정확도가 증가하지 않는 포화 현상은 다음의 두 가지 이유로 설명할 수 있다. 첫째로, 히스토리 레지스터의 길이가 길어지면 패턴의 종류도 급격히 증가하게 된다. 패턴의 종류가 증가하면 초기화 과정에서 예측 실패의 가능성이 증가하고 이는 예측의 정확도를 높이지 못하는 요인으로 작용한다. 둘째로, 패턴의 수가 증가함에 따라 예측 캐쉬의 엔트리 교체(replacement)도 빈번하게 일어나게 되어 이 또한 예측의 정확도를 높이지 못하는 요인으로 작용한다.

제안한 4원 세트 연관 전역 캐쉬 상관 예측기의 평균 예측 실패율은 <표 2>와 같고, 이를 그래프로 나타내면 (그림 4)와 같다. 예측 캐쉬는 1K 엔트리에서 32K 엔트리까지 변화시켰다. 처음에는 히스토리 레지스터의 크기가 증가함에 따라 예측 정확도는 꾸준히 증가한다. 예측 캐쉬의 크기가 1K 엔트리인 경우, 히스토리 레지스터가 12비트가 되면 더 이상 정확도는 증가하지 않는다. 반면에 예측 캐쉬의 크기가 1K 엔트리보다 훨씬 큰 경우, 히스토리 레지스터가 26비트가 될 때까지 예측의 정확도가 증가한다. 당연히 예측 캐쉬의 크기가 커질수록 예측 실패율은 낮아진다. 가장 좋은

〈표 2〉 전역 4원 세트 연관 예측기의 예측 실패율

HR	1K	4K	16K	32K
2	10.89	10.89	10.89	10.89
4	10.69	10.69	10.69	10.69
6	10.19	10.17	10.17	10.17
8	9.36	9.31	9.31	9.31
10	8.91	8.84	8.82	8.82
12	8.62	8.40	8.37	8.36
14	8.60	8.10	8.03	8.03
16	8.72	8.08	7.72	7.71
18	8.65	7.88	7.45	7.35
20	9.03	7.86	7.32	7.08
22	9.29	7.86	7.18	6.85
24	9.58	8.21	7.53	7.04
26	8.75	7.45	6.60	6.05
28	8.91	7.82	6.69	6.02
30	8.96	7.85	6.68	5.99



(그림 4) 전역 4원 세트 연관 예측기의 예측 실패율

결과는 예측 캐쉬 32K 엔트리이고, 히스토리 레지스터 30비트일 때, 예측 실패율 5.99%이며, 이는 종래의 2단계 방식에서 가장 좋은 결과인 히스토리 레지스터가 26비트인 GAs (16 세트) 방식의 9.23% 비해 54% 향상된 결과이다.

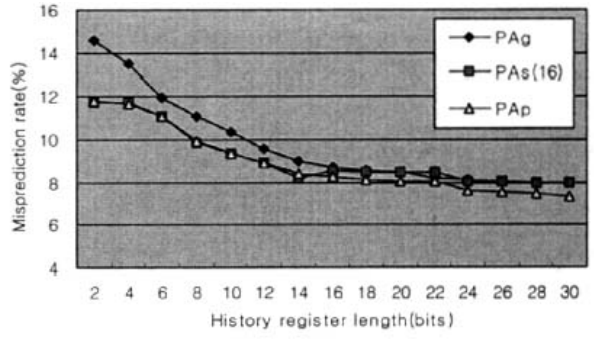
디폴트 예측기를 보조로 사용하면, 사용하지 않는 경우에 비해 예측 캐쉬의 용량이 4K 엔트리일 때, 최대 8.5% 정도의 예측 정확도가 증가한다.

4.2 지역 예측기

<표 3>과 (그림 5)는 PAg, PAp, PAs 등 종래의 방식의 예측 실패율에 대한 표와 그래프이다. 종래의 지역 예측기의 평균 예측 실패율은 GAg, GAs 방식보다는 7.5% 정도 향상되었으나, 가장 좋은 전역 캐쉬 상관 예측기의 결과보다는 못하다. 가장 좋은 예측 실패율은 히스토리 레지스터가 30 비트인 PAp 방식의 7.35%이다. 지역 예측기는 히스토리 레지스터 길이가 증가할수록 전역 예측기에 비해 정확도가 향상된다. 제안한 4원 세트 연관 지역 캐쉬 상관 예측기의 평균 예측 실패율은 <표 4>와 같고, 이를 그래프로 나타내면 (그림 6)과 같다. 예측 캐쉬의 용량은 1K 엔트리에서 32K 엔트리까지 변화시켰다. 예측 캐쉬의 크기가 증가함에 따라 예측 정확도는 꾸준히 증가하다가 16K 엔트리가 넘으면 더 이상 정확도는 증가하지 않는다.

〈표 3〉 종래의 지역 2단계 적응적 예측기의 예측 실패율

HR	PAg	PAs(16)	PAp
2	14.59	11.77	11.81
4	13.45	11.72	11.69
6	11.97	11.11	11.08
8	11.12	9.88	9.90
10	10.35	9.33	9.36
12	9.55	8.90	8.89
14	8.96	8.21	8.44
16	8.73	8.59	8.24
18	8.57	8.51	8.14
20	8.50	8.45	8.07
22	8.17	8.45	8.03
24	8.12	8.07	7.60
26	8.02	8.03	7.57
28	7.97	7.96	7.50
30	7.96	7.95	7.35



(그림 5) 종래의 지역 2단계 적응적 예측기의 예측 실패율

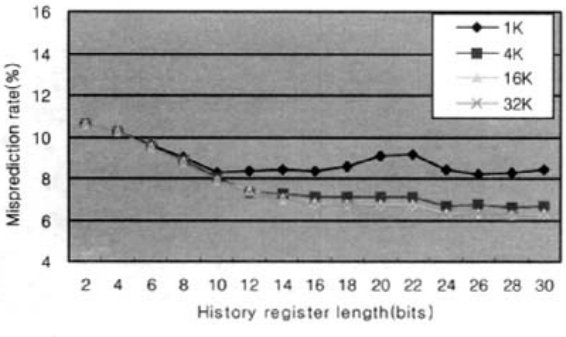
〈표 4〉 지역 4원 세트 연관 예측기의 예측 실패율

HR	1K	4K	16K	32K
2	10.64	10.64	10.64	10.64
4	10.27	10.27	10.27	10.27
6	9.62	9.53	9.53	9.53
8	8.99	8.89	8.83	8.83
10	8.31	8.06	7.94	7.93
12	8.35	7.36	7.39	7.33
14	8.45	7.28	7.04	7.03
16	8.39	7.11	6.85	6.79
18	8.60	7.10	6.78	6.74
20	9.11	7.14	6.74	6.70
22	9.16	7.10	6.74	6.67
24	8.47	6.68	6.34	6.28
26	8.23	6.75	6.36	6.32
28	8.26	6.65	6.28	6.29
30	8.43	6.71	6.31	6.29

반면에 예측 캐쉬의 크기가 1K 엔트리보다 훨씬 큰 경우, 히스토리 레지스터가 26비트가 될 때까지 예측의 정확도가 증가한다. 당연히 예측 캐쉬의 크기가 커질수록 예측 실패율은 낮아진다. 가장 좋은 결과는 예측 캐쉬 16K 엔트리이고, 히스토리 레지스터 30비트일 때, 예측 실패율 6.28%이고, 이는 종래의 2단계 PAp 방식의 가장 좋은 결과인 히스토리 레지스터

가 30 비트일 때의 7.35%에 비해 17% 향상된 결과이다.

디폴트 예측기를 사용하면, 사용하지 않는 경우에 비해 예측 캐쉬의 용량이 4K 엔트리일 때, 최대 17% 정도의 예측 정확도가 증가한다.



(그림 6) 지역 4원 세트 연관 예측기의 예측 실패율

4.3 비용 비교

예측의 정확도 못지 않게 중요한 요소는 예측기를 구성하는 비용이다. 여기서의 비용은 구체적인 회로 구성 방식에 따른 차이는 무시하고, 계량화할 수 있는 두 방식의 메모리 용량을 중심으로 비교한다. 캐쉬 상관 예측기에서 가장 정확도가 높을 때의 메모리 양을 바이트로 환산하면 전역의 경우 87KByte, 지역의 경우 93.75KByte이다. 반면에 종래의 방식에서 정확도가 가장 높을 때 필요한 메모리의 양은 PAp 방식의 268GByte이다. 따라서 캐쉬 상관 예측기를 이용하면 종래의 방식에 비해 천분의 1이하의 메모리 용량으로 동일한 효과를 얻을 수 있다는 것이다. 캐쉬 상관 예측기에 필요한 메모리는 히스토리 레지스터의 크기에 선형적으로 비례하여 증가하므로, 따라서 비용도 선형적으로 증가한다. 반면에 종래의 2단계 예측기는 히스토리 레지스터의 크기가 증가하게 되면 PHT를 구성하는 메모리는 기하급수적으로 증가하게 되어, 비용도 폭발적으로 늘어난다.

전역 예측기의 메모리를 250KB정도로 제한했을 때의 각 구성 방식에 따른 성능을 비교하여 보면, 4K 엔트리 캐쉬 상관 예측기는 종래의 어떤 전역 예측기 구성보다도 비용 효용성이 뛰어나며, 16K 엔트리 캐쉬 상관 예측기는 비슷한 비용의 종래의 방식에 비해 월등하다.

5. 결 론

시뮬레이션 결과, 캐쉬 상관 예측기는 종래의 2단계 예측기에 비해 예측의 정확도가 더 높고 실리콘 면적도 절약할 수 있음을 확인했다.

전역 예측기의 경우에 가장 좋은 결과는 예측 캐쉬 32K 엔트리이고 히스토리 레지스터 30비트 일 때 예측 실패율 5.99%이며, 이는 종래의 2단계 방식에서 가장 좋은 결과인 히스토리 레지스터가 26비트인 GAs(16세트) 방식의 9.23%에 비해 54% 향상된 결과이다.

지역 예측기의 경우에 가장 좋은 결과는 예측 캐쉬 16K 엔트리이고 히스토리 레지스터 30비트 일 때 예측 실패율 6.28%이고, 이는 종래의 2단계 PAp 방식의 가장 좋은 결과인 히스토리 레지스터가 30비트 일 때의 7.35%에 비해 17% 향상된 결과이다.

이러한 특성은 다음 사실에 기인한다. 첫째, 예측은 최소한 한번이라도 초기화된 예측 계수기에 의해 결정된다. 둘째, 서로 다른 분기 예측 사이에 간섭이 일어나지 않는다. 셋째, BTC에 디폴트 예측기를 추가함으로써 예측 캐쉬가 초기화될 동안에 BTC를 이용해 예측할 수 있다. 더욱이 디폴트 계수기는 예측 캐쉬의 미스 충격을 완화시켜준다.

참 고 문 헌

[1] J. L. Hennessy and D. A. Pattern, Computer Architecture, Morgan Kaufmann, 1996.

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

[3] S. Pan, K. So and J. T. Rahmeh, "Improving the Accuracy of Dynamic Branch Prediction Using Branch Correlation," ASPLOS-V, pp.76-84, 1992.

[4] G. B. Steven, C. Egan and L. Vintan, "A Cost Effective Cached Correlated Two-Level Adaptive Branch Predictor," International Conf. on Applied Informatics, pp.208-212, Feb., 2000.

[5] C. Egan, "Dynamic Branch Prediction in High Performance Superscalar Processors," PhD Thesis, Univ. of Hertfordshire, pp.125-140, Aug., 2000.

[6] T. Yeh and Y. Patt, "Two-Level Adaptive Training Branch Prediction," Micro-24, pp.51-61, Nov., 1991.

[7] T. Yeh and Y. Patt, "A Comparison of Dynamic Branch Predictors that use Two Levels of Branch History," ISCA-20, pp.257-266, May, 1993.

[8] S. McFarling, Combining Branch Predictors, Western Research Lab. Technical Report TN-36, Jun., 1993.

[9] P. Chang, E. Hao and Y. Patt, "Alternative Implementation of Hybrid Branch Predictors," Micro-28, pp.252-257, Nov., 1995.

[10] G. B. Steven, D. B. Christianson, R. Collins, R. D. Potter and F. L. Steven, "A Superscalar Architecture to Exploit Instruction Level Parallelism," Microprocessors and Microsystems, pp.391-400, 1997.

[11] F. L. Steven, An Introduction to the Hatfield Superscalar Scheduler, Univ. of Hertfordshire Technical Report 316, 1998.

심 원

e-mail : wonshim@snut.ac.kr

1974년 서울대학교 공과대학 공업교육학과 (전자)학사

1978년 서울대학교 공업교육학과 석사

1995년 동국대학교 컴퓨터공학과 박사

1978년~1982년 삼성전자(구 KTC)

1982년~현재 서울산업대학교 컴퓨터공학과 교수

2000년~2001년 영국 Hertfordshire 대학 방문교수

관심분야 : 병렬처리, 마이크로프로세서