슈퍼스칼라 프로세서에서 값 예측기의 성능평가

전 병 찬*, 박 희 룡**, 이 상 정*
* 순천향대학교 정보기술공학부
*김천대학 컴퓨터정보처리계열
e-mail:leebc-66@hanmail.net, hrpark@kimcheon.ac.kr, sjlee@sch.ac.kr

A Performance Evaluation of Value Predictors in a Superscalar Processor

Byung-Chan Jeon , Hee-Ryong Park, Sang-Jeong Lee
Div. of Information Technology Engineering, Soonchunhyang University
Div. of Computer Information Process, Kimcheon College

요 약

와이드 이슈 슈퍼스칼라 프로세서에서 값 예측기는 한 명령어의 결과를 미리 예측하여 명령들 간의 데이터 종속관계를 극복하고 실행함으로써 명령어 수준 병렬성(Instruction Level Parallesim, ILP)을 향상시키는 기법이다. 본 논문에서는 명령어 수준 병렬성을 이용하여 성능을 향상시키기 위하여 데이터 값을 미리 예측하여 병렬로 이슈하고 수행하는 값 예측기의 성능을 비교분석 한다. 먼저 각 예측기 종류별로 성능을 측정한다. 그리고 테이블의 갱신시점, 트레이스 캐시 유무 및 명령윈도우 크기에 따른 값 예측기의 성능영향을 평가분석 한다. 성능분석 결과 최근 값 예측기가 간소한 하드웨어 구성에도 불구하고 우수한 성능을 보였다. 그리고 예측테이블 갱신시점과 트레이스캐시의 사용이 값 예측기의 성능향상에 영향을 주었다.

1. 서론

와이드 이슈 슈퍼스칼라 프로세서에서 명령어 수준의 병렬성 처리는 한 프로세스에서 여러 개의 명령을 이슈하여 병렬로 실행하는 기법이다. 이러한 ILP처리에서 주요한 장애요소로는 명령들 간의 데이터 종속성 문제인데, 이러한 데이터 종속관계의 문제는 명령의 병렬처리를 방해한다. 즉, 현재의 명령이 이전 명령과 종속관계가 있으면 이전 명령의 결과가 나올 때까지 현재의 명령은 이슈되어 병렬로 실행할 수 없게 된다.[1] 따라서 ILP의 장애요소의 명령들간의 데이터 종속문제를 극복하기 위한 방법으로 데이터 값 예측 기법들이 활발히 연구되고 있다[2,3,4,5,7,9]. 데이터 값 예측기법은 최근에 실행되는 명령들의 결과 값을 미리 예측하여 모험적으로 실행(speculative execution)하므로써 명령들과 데이터 종속관계가 있는 명령들이 중지되지 않고 계속 이슈하여 실행되게 함으로써 성능을 향상시키는 기법이다. 대부분의 고성능 와이드 이슈 슈퍼스칼라 프로세서는 명령어 수준 병렬성을 이용하여 다수의 명령을 동시에 이슈하고 처리하여 성능을 향상시키고 있다. 이들 프로세서에서 높은 명령어 수준 병렬성을 유지하려면 사이클 당 많은 명령을 페치하여 명령 윈도우에 공급하는 것이 필수적이다. 그러나 대부분의 프로그램들이 빈번한 조건분기 명령들의 수행으로 제어흐름이 빈번하게 변하는 실정이다. 따라서 한 개의 캐시 블럭에 하나의 기본 블럭만을 페치하는 전통적인 페치방식으로는 명령의 공급이 한계에 도달하고 있다. 최근에는 이와 같은 문제점을 극복하기 위하여 실행되는 여러 개의 경로(트레이스)를 트레이스 캐시에 저장하여 명령들을 공급하는 기법들이 개발되었다[6].

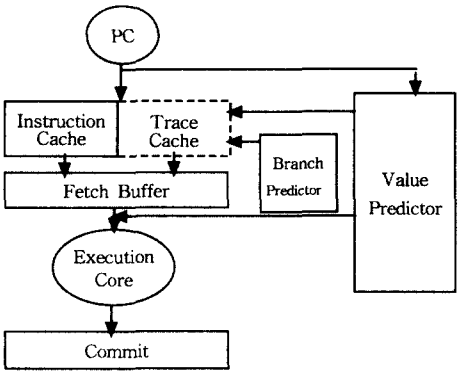
본 논문에서는 명령어 수준 병렬성을 이용하여 프로세서의 성능을 향상시키기 위하여 데이터 값을 미리 예측하여 병렬로 이슈하고 수행하는 값 예측기의 성능을 비교분석 한다. 먼저 각 예측기 종류별로 성능을 측정한다. 그리고 테이블의 갱신시점, 트레이스 캐시 유무 및 명령윈도우 크기에 따른 값 예측기의 성능영향을 평가분석 한다. 성능측정 및 평가를 위해 실행구동방식 시뮬레이터인 Simplescalar/Alpha 3.0 tool set[1]에 이용하여 SPECint95 벤치마크 프로그램을 시뮬이션하여 비교한다.

2. 값 예측기

값 예측기는 예측되는 데이터 값 시퀀스의 분류에 따라 최근 값 예측기, 스트라이드 값 예측기, 2-단계 값 예측기, 혼합형 예측기등이 있다. 최근 값 예측기(Last Value Predictor)[5]는 한 명령의 최근에 수행된 결과를 테이블에 저장하고 다음에 이 명령 수행 시 이 값을 사

본 연구는 한국과학재단 목적기초연구(2000-1-30300-008-3)지원으로 수행되었음.

용하여 예측하는 기법으로 가장 단순한 데이터 값 예측 기법이다. 스트라이드 예측기(Stride Predictor)[9]는 한 명령의 최근에 수행된 두 개의 결과 값의 차인 스트라이드를 구하고 이를 테이블에 저장한다. 이후 이 명령 수행 시 최근의 결과 값과 스트라이드 값을 더한 결과를 예측 값으로 사용하는 기법이다. 2-단계 값 예측기(Two-Level Value Predictor)[9]는 2-단계 분기 예측기 법과 유사한 구조를 갖고 있다. 즉, 최근에 수행된 데이터 값의 시퀀스를 저장하고 이 시퀀스 패턴에 따라 테이블을 인덱스하여 값을 예측한다. 혼합형 값 예측기(Hybrid Value Predictor)[9]는 스트라이드와 2-단계 데이터 값 예측기를 결합한 예측기이다.



[그림 1] 프로세서 구조

3. 실험 방법

본 논문의 성능측정은 [그림 1]과 같은 8 이슈의 프로세서 구조를 사용하였다. 분기예측을 위해서 2-웨이의 2K 엔트리를 갖는 BTB와 gshare와 bimodal 방식을 혼합한 분기 예측기를 사용하였다. 그리고, 512KB 크기를 갖는 명령 캐시와 데이터 캐시를 사용하였다(트레이스 캐시를 사용하는 경우 트레이스 캐시는 512KB와 블록당 16개 명령을 가진 2-웨이 어소시에티브로 구성된다).

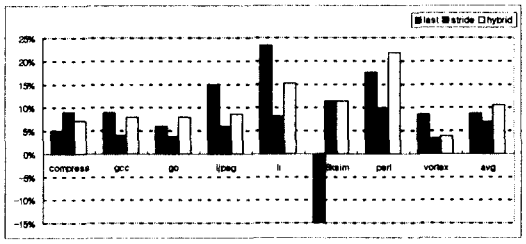
[표1]SPECint95벤치마크프로그램의입력자료

compress	10000 e 2231	35,261,714	2.7743
gcc	jump.i	38,772,635	1.7702
go	5 9	81,530,040	1.5199
ijpeg	tinyrose.ppm	63,415,601	3.2253
li	queen6.lsp	41,524,887	2.5914
m88ksim	dhry.big.100	240,738,844	5.5759
perl	scrabbl.in	40,353,136	2.359
vortex	persons.250	66,740,617	2.6992
평 균		76,042,184	2.8144

혼합형 값 예측기는 8K 엔트리의 VHT와 4개의 히스토리를 갖고 , 8K 엔트리로 구성된 PHT의 테이블 크기로 구성된다. 시뮬레이션을 수행하기 위해 사용되어진 벤치마크 프로그램은 [표 1]과 같이 SPECint95 벤치마크 프로그램 중 8개의 프로그램에 대해 시뮬레이션 하였다. [표 1] 에서 기본 IPC(Instructions Per Cycles)는 값 예측을 적용하지 않은 경우의 IPC로써 성능향상 측정 시 기본 값이 된다.

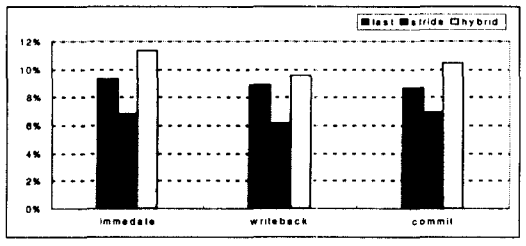
4. 성능측정 및 평가

[그림 2]는 예측 테이블의 갱신을 명령 실행 후 writeback 스테이지에서 하는 경우의 각 예측기별 성능향상을 보인 그림이다.



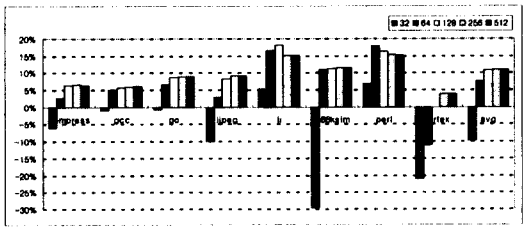
[그림 2] 값 예측기 별 성능향상

그림에 나타난 바와 같이 최근 값(last), 스트라이드(stride) 및 혼합형(hybrid) 예측기의 평균 성능향상이 각각 8.6%, 6.9%, 10.4% 임을 알 수 있다. 혼합형 예측기의 평균 성능이 가장 우수하였지만 m88ksim의 벤치마크에서의 최근 값 예측기의 큰 성능 저하를 배제하면 하드웨어가 간소한 최근 값 예측기의 성능이 전반적으로 우수함을 알 수 있다. 이는 와이드 이슈 프로세서에서의 높은 명령 폐치율 때문에 테이블의 갱신 속도가 새로운 예측을 위한 참조 속도를 따라가지 못한데서 기인한다.[3,4] 즉, 스트라이드 및 혼합형 예측기는 예측을 위해 갱신된 테이블의 값이 요구되는 반면에 최근 값 예측기는 갱신을 요구하지 않고 고정된 값으로 예측하기 때문이다.

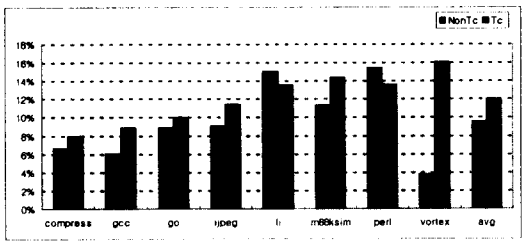


[그림 3] 테이블 갱신 시점에 따른 예측기별 평균 성능향상

[그림 3]은 각 테이블 갱신 시점에 따른 예측기별 평균 성능 향상을 보여 주는 그림이다. immediate는 명령 페치 시 예측 값의 참조를 위해 예측 테이블을 참조하자마자 새로운 값으로 갱신하는 이상적인 경우를 가정한 것이다. 예측 테이블의 갱신을 위해서는 명령의 실행 결과 값이 요구되므로 이는 실제 머신 상에서는 발생할 수 없다. writeback은 명령 실행 후 결과 값이 산출되자마자 예측 테이블을 갱신하는 것을 나타내며 commit은 예측 명령 이전의 모든 명령이 실행을 하여 non-speculative state가 되었을 때 예측 테이블을 갱신한다. [그림 3]에서 나타난 바와 같이 immediate인 경우가 성능이 가장 좋음을 알 수 있다. 특히 혼합형 예측기의 경우 immediate의 성능이 크게 우수함을 알 수 있다. 이는 immediate인 경우 예측 테이블의 갱신 지연이 없기 때문이다. writeback과 commit을 비교할 때 최근 값 예측기의 경우가 writeback이, 혼합형 예측기의 경우 commit의 성능이 약간 우수함을 알 수 있다. [그림 4]는 명령 윈도우 크기가 32, 64, 128, 256, 512인 경우에 혼합형 예측기의 성능 향상을 보여주는 그림이다. 명령 윈도우의 크기가 32인 경우에 값 예측이 대부분의 벤치마크에서 오히려 성능 감소를 보임을 알 수 있다. 이는 예측실패로 인한 재 이슈된 명령으로 인한 하드웨어 자원의 점유에 기인한다. 값 예측을 위한 명령 윈도우의 크기는 128이 적당함을 알 수 있다. [그림 5]는 트레이스 캐시를 사용하지 않은 경우(NonTc)와 사용한 경우(Tc)에 혼합형 예측기에서 성능향상을 비교한 그림이다. NonTc인 경우 평균 9.5%, Tc의 경우 11.9%의 성능 향상을 보여 트레이스 캐시를 사용한 경우에 다소 성능이 향상됨을 알 수 있다.



[그림 4] 명령윈도우 크기에 따른 혼합형 예측기 성능향상



[그림 5] 트레이시 캐시 사용 효과

5. 결론

본 논문에서는 최근 값, 스트라이드, 혼합형 값 예측기의 각 예측기별 성능과 예측 테이블의 갱신시점, 트레이스 캐시 유무 및 명령윈도우 크기에 따른 값 예측기의 성능영향을 평가 분석하였다. 성능평가 결과 와이드 이슈 프로세서에서는 예측 테이블의 갱신 지연 효과 때문에 하드웨어가 비교적 간소한 최근 값 예측기의 성능이 복잡한 혼합형 예측기의 성능에 근접하거나 우수함을 알 수 있었다. 그리고 명령윈도우의 크기도 값 예측기의 성능향상에 크게 영향을 주는 것으로 분석되었다. 또한 트레이스 캐시를 사용하면 명령들의 원활한 공급으로 성능이 다소 향상되었다.

[참고문헌]

[1] D.Burger and T.Austin, "The Simplescalar Tool Set, Version 2.0",Technical Report CS-RT-97-1342, University of Wisconsin,Madison, June 1997

[2] S.lee and P.Yew, "Decoupled Value Prediction on Trace Processors",Proceedings of the 6th International Symposium on High Performance Computer Architecture (HPCA-6), 2000.

[3] S.Lee and P.Yew "On Some Implementation Issues for Value Prediction on Wide-Issue ILP Processors," International Conference on Parallel Architectures and Compilation Techniques(PACT 2000), Oct. 2000.

[4] S.Lee, and Pen-Chung Yew, "On Table Bandwidth and Its Update Delay for Value Prediction on Wide-Issue ILP Processors", IEEE Transaction on Computers, Vol.50 No.8, Aug. 2001

[5] M.Lipasti, and J.Shen, "Exceeding th Limit via Value Prediction", Exceedings of the 29th International Symposium on Microarchitecture (MICRO-29), Dec. 1996

[6] E. Rotenberg, S.B., and J.Smith, "Trace Cache: a Low Latency Approache to High Bandwidth Instruction Fetching", Technical Report: University of Wisconsin-Madison Technical Report #1310, Apr. 1996.

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

[8] K.Skadron, M.Martonosi, and D.Clark. "Speculative Updates of Local and Global Branch History" A Quantitative Analysis," Journal of Instruction-Level Parallelism, vol.2, Jan. 2000.

[9] K.Wang, M.Franklin, "Highly Accurate data value Predictions using Hybrid Predictor,"Proceedings of the 30th International Symposium on Microarchitecture (MICRO-30), Dec. 1997.