농업정보기술을 위한 ILP 프로세서에서 새로운 복구 메커니즘 적용 분기예측기

A Branch Predictor with New Recovery Mechanism in ILP Processors for Agriculture Information Technology

고광현* · 조영일**

Kwang Hyun Ko, Young Il Cho

ABSTRACT

To improve the performance of wide-issue superscalar processors, it is essential to increase the width of instruction fetch and the issue rate. Removal of control hazard has been put forward as a significant new source of instruction-level parallelism for superscalar processors and the conditional branch prediction is an important technique for improving processor performance. Branch mispredictions, however, waste a large number of cycles, inhibit out-of-order execution, and waste electric power on mis-speculated instructions. Hence, the branch predictor with higher accuracy is necessary for good processor performance. In global-history-based predictors like gshare and GAg, many mispredictions come from commit update of the branch history. Some works on this subject have discussed the need for speculative update of the history and recovery mechanisms for branch mispredictions. In this paper, we present a new mechanism for recovering the branch history after a misprediction. The proposed mechanism adds an age_counter to the original predictor and doubles the size of the branch history register. The age_counter counts the number of outstanding branches and uses it to recover the branch history register. Simulation results on the SimpleScalar 3.0/PISA tool set and the SPECINT95 benchmarks show that gshare and GAg with the proposed recovery mechanism improved the average prediction accuracy by 2.14% and 9.21%, respectively and the average IPC by 8.75% and 18.08%, respectively over the original predictor.

Key Words : branch prediction, misprediction, speculative update, recovery mechanism

* 한국농수산대학교 정보교육센터, E-mail: ko@af.ac.kr

** 수원대학교 컴퓨터학과, E-mail: yicho@suwon.ac.kr:

1. 서 론

농업정보의 수집, 분석, 축적, 검색 등 농업관련 분야와 연구에서 프로세서의 사용이 일반화됨에 따라 프로세서 성능에 크게 관심이 모아지고 있다(고광현 & 조영일, 2003; Kwang-Hyun Ko, Young-Il Cho, 2008). 특히, 유전체구조해석 등과 같은 연구 분야에서 필요한 대용량 데이터처리 시 데이터의 검색과 처리가 최단시간에 정확하게 이루어 질 수 있도록 하기 위해 많은 농업관련 연구자들은 점점 더 고성능 프로세서의 활용을 희망하고 있으며, 이와 같은 요구는 앞으로도 계속될 것이다. 표1은 현재 국립농업과학원의 연구용프로세서들 중 일부에 대한 성능을 조사한 것으로 유전체구조해석분야등의 연구에서 고성능 프로세서가 사용되고 있음을 보여주고 있으며, 표2는 프로세서 성능 발전의 최근 현황을 알아보기 위해 인텔의 경우만을 예로 들어 조사한 결과이다. 프로세서들의 성능이 이와 같은 추세로 계속 개선 되어간다면 앞으로 농업정보 및 연구 분야 발전에도 크게 기여할 것으로 예측된다.

〈표 1〉 국립농업과학원 연구용 Processor 현황

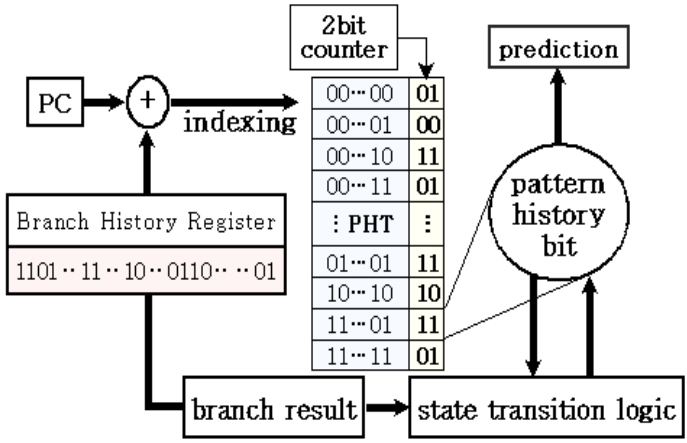
구 분	OS	CPU수	속도	용 도
IBM	AIX 5L 5.1	12	1.3G	EnsEMBL, 연구정보
IBM	AIX 5L 5.3	4	1.3G	유전체 분석
Compaq	Windows 2000	4	900M	식물 유전자원DB
HiServer	Linux 9	4	1.5G	BAC 분석
Compaq	Linux 7.1	2	1.1G	Vector NTI
MSI	FC2(Linux)	2	1.8G	유전자 검색
MSI	FC5(Linux)	2	1.8G	배추 유전체 분석
MSI	FC3(Linux)	2	1.8G	EST 분석
MSI	Linux 9	2	1.8G	TRANSFAC
SUN	Solaris 8	2	400M	생물자원DB
Compaq	Linux 7.1	2	1.1G	유전자 검색
SGI(64bit)	SuSE Ent. 10	20(2Core)	1.6G	유전체 구조해석
SGI(64bit)	SuSE Ent. 9	4(2Core)	2.6G	Pedant-Pro
SGI(64bit)	SuSE Ent. 9	4(2Core)	2.6G	Pedant-Pro

〈표 2〉 Processor성능 발전의 최근 현황

구 분	Processor	Transfer GT/s	Cache L3,MB	Clock GHz	Power W	Core	Thread
Intel® Xeon® Processor 5000 Sequence Specification (32n mm)	X5680	6.4	12	3.33	130	6	12
	X5677			3.46		4	8
	X5670			2.93	95	6	12
	X5667			3.06		4	8
	X5660			2.8		6	12
	X5650			2.66			
	L5640	5.86		2.26	60		
	L5630			2.13	40	4	8
	L5609	4.8		1.86			4
	E5640	5.86		2.66	80		8
	E5630			2.53			
	E5620			2.4			
Intel® Xeon® Processor 6000 Sequence Specification (45n mm)	X6550	6.4	18	2	130	8	16
	E6540				105	6	12
	E6510	4.8	12	1.73		4	8
Intel® Xeon® Processor 7000 Sequence Specification (45n mm)	X7560	6.4	24	2.266	130	8	16
	X7560		18	2			
	X7542	5.86		2.266			6

최근 많은 논문에서 프로세서의 성능향상을 위한 여러 가지 방법들 중 명령어수준 병렬성(ILP : Instruction Level Parallelism)의 성능을 저해시키는 “데이터종속”과 “제어종속”에 의한 장애를 제거시키는 “결과 값 예측기법”과 “분기예측기법”에 대한 연구가 활발히 진행되고 있다. 사이클 당 여러 개의 명령어를 수행시키는 슈퍼스칼라 프로세서는 유용한 명령어를 계속 공급하기 위해서 정확한 분기예측기가 필요하다. 프로세서의 성능을 향상시키기 위해 파이프라인의 깊이(depth)와 이슈 폭(width)을 증가 시킬수록 분기예측실패(branch misprediction)가 발생하면 올바른 경로의 명령어를 수행하기 전 잘못된 경로상의 명령어들을 수행하게 되므로 많은 사이클이 낭비되어 프로세서의 성능이 저

하되는 문제가 발생한다. 이 문제를 해결하기 위해 예측정확도가 높은 "동적 분기결과 예측기술"에 대한 연구가 계속되고 있다. 동적 분기예측 방법은 실행시간동안 얻어지는 정보를 이용하여 예측기를 학습시키고, 학습된 지식을 이용하여 예측정확도를 향상시키는 방법이다. 동적 분기예측 방법 중 선행 분기명령들의 분기결과를 예측하려는 분기명령과 상호연관(correlation)시켜 예측하는 2-단계적응분기예측기(two-level adaptive branch predictor)의 효과가 입증(T.-Y. Yeh., & Y. N. Patt, 1992)되었으며 이를 변형시킨 방법이 적용된 예측기가 여러 상용 프로세서에 적용되고 있다(K. Diefendorff, 1998; R. E. Kessler., E. J. McLellan., & D. A. Webb, 1998). 또, 분기명령의 지역 분기 히스토리(local branch history)와 전역 분기 히스토리(global branch history)를 선택적으로 사용하여 예측하는 하이브리드 분기예측기(G.H. Loh., & D.S. Henry, 2002)와 지역 히스토리와 전역 히스토리를 결합하여 예측하는 결합형 예측기(Z. Lu., J. Lach., M. Stan., & K. Skadron, 2003)등 여러 가지 방법의 예측기도 연구되었다.



〈그림 1〉 Two-level adaptive branch predictor

파이프라인 프로세서에서 분기명령어는 반입시간(fetch stage)에 분기결과를 예측하고, 해당 분기명령의 실제결과는 실행시간(execute stage)에 결정되며, 분기예측이 실패한 경우 명령어 완료시간에 잘못된 경로에서 수행된 명령어들을 무효화시키고 올바른 경로의 명령어들을 반입하여 실행시킨다. 분기를 예측하고, 올바른 결과가 결정되는 시간에 여러 사이클이 경과된다. 파이프라인 깊이

와 이슈 폭이 큰 프로세서에서는 이 시간동안 상당수의 분기명령들이 반입되어 예측된다. 분기예측기가 명령어 완료시간에 갱신된다면 수행중인(in-flight) 분기명령들의 결과는 부정확한 분기 히스토리(stale branch history)로 예측된 것이다. 이것은 특히 2-단계적응예측기 등에서 문제가 된다(A. R. Talcott., W. Yamamoto., M. J. Serrano., R. C. Wood., & M. Nemirovsky, 1994; E. Hao., P.-Y. Chang., & Y. N. Patt, 1994). 이들 방법은 이전 분기들의 동작에 대한 정확한 분기 히스토리에 의존하고 있기 때문이다(M. Evers., S. J. Patel., R. S. Chappell., & Y. N. Patt. 1998). 만약, 예측기가 실제결과(resolved outcome)대신 예측결과(predicted outcome)로 모험적 갱신(speculative update)을 할 수 있다면 이 문제를 해결할 수 있다. 예측이 올바르다면 모험적 갱신은 어떠한 손실도 발생시키지 않는 반면, 분기예측실패(branch misprediction)의 경우 모험적 갱신은 분기 히스토리에 잘못된 정보를 삽입하였기 때문에 오염된 분기 히스토리를 분기명령을 예측하기 직전의 상태로 복구시켜야 한다. 이를 위해 본 논문 이전에 제안되었던 다른 복구 메커니즘(recovery mechanism)들은 분기명령 예측 시 큐(queue)나 리오더 버퍼(reorder buffer)에 분기 히스토리를 저장하였다가 분기예측실패 시 저장되었던 값으로 복구하는 복잡한 메커니즘을 사용하였으나(K. Skadron., & M. Martonosi, 2000; S. Jourdan, J. Stark., T.-H. Hsing, & Y. N. Patt, 1997; K. Skadron., P. S. Ahuja., M. Martonosi., & D. W. Clark, 1998), 본 논문에서는 분기예측실패 후에 분기 히스토리를 복구하는 새로운 메커니즘을 제안한다. 분기예측기에 미해결 분기명령어 수를 저장하는 age_counter를 추가하고, 분기 히스토리 레지스터 크기를 2배로 확장시킨 후 age_counter를 이용하여 분기예측실패 발생즉시 분기 히스토리 레지스터를 복구시키는 방법을 사용한다. SimpleScalar 3.0/PISA 툴셋으로 시뮬레이션 한 결과, 복구 메커니즘을 기존의 전역 히스토리 기반 분기예측기에 적용하였을 때 예측 정확도와 프로세서 성능을 개선시킬 수 있었음을 확인하였다.

2. 연구배경

명령어의 반입 폭과 이슈 폭이 증가함에 따라 GAg(T.-Y. Yeh., & Y. N. Patt, 1992), 또는 gshare(M. Evers., S. J. Patel., R. S. Chappell., & Y.

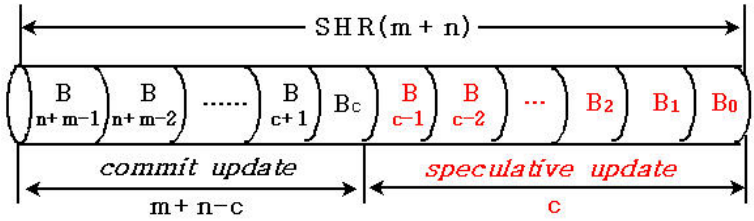
N. Patt, 1998)와 같이 전역 히스토리를 사용하는 분기예측기에서는 분기명령어가 예측된 후 실제 분기결과 값으로 예측기의 전역 히스토리 레지스터(GHR:Global History Register)를 수정하기 전에 다른 분기명령어를 예측하는 경우가 대부분이다. 이런 경우, 전역 히스토리 레지스터는 최근의 분기명령들에 대한 히스토리를 포함시키지 못하므로 최근의 분기명령들과 강한 상호관계(strong correlation)를 갖는 분기명령을 예측하려할 때 예측정확도가 떨어져 프로세서의 성능을 저하시키게 된다(E. Hao., P.-Y. Chang., & Y. N. Patt, 1994). 분기예측기는 프로그램이 어떤 상태에 있는지를 식별하기 위해 전역 히스토리 레지스터를 사용하고, 예측은 해당 상태를 기초로 한다. 그러나 프로그램 수행 시 예측은 되었으나 결과가 아직 전역 히스토리 레지스터에 반영되지 않은 미해결 분기명령어 수의 다양한 동적 변화는 전역 히스토리 레지스터가 더 이상 특정 프로그램 상태의 모든 동적발생(every dynamic occurrences)에 대해 동일한 히스토리를 갖는 것을 보장할 수 없다. 따라서 전역 히스토리 레지스터는 프로그램에서 상태를 식별하는 예측기의 능력이 감소되게 된다. 다시 말해, 동일 분기명령의 다른 동적 발생들을 예측할 때 캐시미스, 다른 분기명령의 예측실패, 자원부족 등과 같이 동적으로 변하는 상태에 따라 미해결 분기명령어 수는 달라질 수 있다. 두 개의 동적 발생에 대한 실제 분기 히스토리가 같다면, 동일한 예측을 해야 하지만 예측할 때 미해결 분기명령어 수가 다르기 때문에 사용되는 전역 히스토리 레지스터의 내용이 달라진다. 이것은 예측기가 각 예측을 위해 다른 PHT 엔트리를 사용하게 하여 잠재적으로 다른 예측을 만들 수 있으며, 결과적으로 예측정확도를 떨어지게 할 수 있다(E. Hao., P.-Y. Chang., & Y. N. Patt, 1994). 이런 문제들은 분기명령을 예측한 후 그 예측값으로 전역 히스토리 레지스터를 모험적으로 갱신하면 해결될 수 있다(E. Hao., P.-Y. Chang., & Y. N. Patt, 1994; K. Skadron., & M. Martonosi, 2000). 모험적으로 갱신한 예측 값이 실제 결과 값과 동일하면 어떤 해도 미치지 않는다. 그러나 예측 값이 실제 결과와 다른 경우 전역 히스토리 레지스터는 잘못된 경로상의 분기명령들의 모험적 갱신으로 오염되어 있으므로 이것을 복구시키지 않으면 예측실패 후 프로세서 상태가 복구된 뒤 올바른 경로상의 분기명령을 예측하게 될 때 오염된 전역 히스토리 레지스터를 사용하게 되므로 예측정확도가 떨어지는 결과를 낳게 된다. 따라서 분기예측실패 후 프로세서의 상태복구와 더불어 전역 히스토리 레지스터를 분기예측실패 직전의 히스토리로 복구시키는 메커니즘이 필요하다.

3. 새로운 복구 메커니즘

기존의 메커니즘은 대부분 큐를 사용하므로 복잡한 하드웨어가 요구되는데, 본 논문에서는 기존의 메커니즘과 같이 전역 히스토리 레지스터를 복구하면서 간단한 하드웨어로 구현시키는 메커니즘을 제안한다. 기존의 복구 메커니즘인 history-based 메커니즘과 future-based 메커니즘에서 OBQ의 크기는 PHT의 엔트리 수가 8K, 명령어 윈도우(instruction window)의 엔트리 수가 128, 최대 미해결 분기수를 20개라 가정했을 때 OBQ의 한 엔트리 크기는 21bit(전역 히스토리 레지스터의 내용을 저장하기 위한 13bit, 한 분기 명령이 OBQ에 여러 엔트리를 가질 수 있으므로 이를 식별하기 위해 명령어 윈도우 엔트리 식별자를 OBQ 엔트리에 태그(tag)하기 위한 8bit)이므로 총 420bit와 큐를 구동시키는 로직(logic)을 필요로 하는 반면, 새롭게 제안한 메커니즘은 SHR(13bit)과 age_counter(5bit)만 추가 시키므로 18bit만 소요된다.

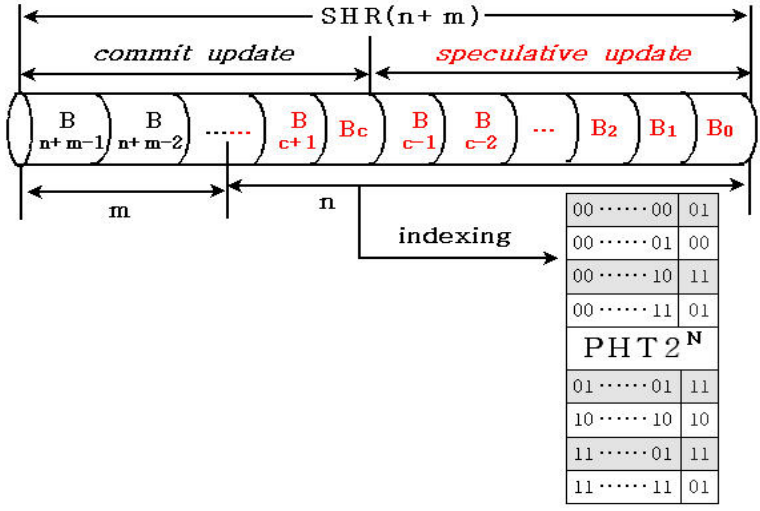
3.1. 분기예측과 모험적 갱신

분기예측기에 모험적 갱신과 분기예측실패 시 분기 히스토리 복구를 위한 age_counter를 추가하고, 전역 히스토리 레지스터를 사용하지 않고 SHR (Speculative History Register)을 사용한다. age_counter는 현재 반입되어 예측되었으나 아직 실제결과로 분기 히스토리를 갱신하지 못한 미해결 분기명령어의 수를 나타낸다. 새로운 분기명령이 반입되면 미해결 분기명령어가 한 개 증가하므로 age_counter를 '1' 증가시키고, 분기명령어의 완료시간에 분기명령어가 올바로 예측이 되었다면 미해결 분기명령어가 한 개 줄어들게 되므로 age_counter를 '1' 감소시킨다. 기존의 예측기에서 전역 히스토리 레지스터는 반입된 순서대로 완료시간에 실제 분기결과로 갱신된 분기 히스토리를 가지나, 본 논문의 분기예측기에 있는 SHR은 완료시간에 갱신된 히스토리와 현재 수행중인 분기명령들의 예측 값으로 모험적 갱신한 히스토리를 포함한다. 그림2는 age_counter가 c일 경우 즉, 미해결 분기명령이 c개일 때 SHR의 상태를 나타낸다. B0에서 BC-1은 수행중인 분기명령들의 모험적 갱신 히스토리를 나타내고, BC에서 Bn+m-1은 완료시간 갱신 히스토리를 나타낸다. SHR은 전역 히스토리 레지스터와는 다르게 모험적 갱신을 위해 m bit를 추가하였다. 시뮬레이션 시 m의 값은 n과 동일한 값을 갖도록 하였다.



〈그림 2〉 age_counter가 c일 경우 SHR의 상태

전역 히스토리 레지스터의 모험적 갱신을 허용하는 gshare 분기예측기에서 조건 분기명령에 대한 예측은 그림3과 같이 SHR에서 최근의 마지막 n개(PHT 엔트리 수가 2n 일 때)의 분기명령어 결과를 분기명령어의 프로그램 카운터와 exclusive-or 하여 PHT를 인덱스하고(GAg의 경우 최근 마지막 n개의 분기명령어 결과를 사용하여 PHT 인덱스) 선택된 PHT 엔트리(포화 카운터)의 값이 '2' 이상이면 taken으로 '1' 이하면 not-taken으로 분기를 예측한다. 조건 분기명령을 예측할 때 age_counter의 내용을 '1' 증가시켜 현재 미해결 분기명령어의 수가 '1' 증가되었음을 나타낸다. 모험적 갱신은 분기명령에 대한 분기예측결과를 사용하여 SHR을 갱신하게 되는데, SHR을 좌로 '1' bit 이동시키고, B0에 예측한 결과를 저장한다.



〈그림 3〉 gshare에 적용한 복구 메커니즘

3.2. 분기예측실패 발생과 복구

SHR을 모험적으로 갱신한 예측 값이 실제결과 값과 동일하다면 프로세서는 정상적으로 계속 동작한다. 그러나 예측 값이 실제결과와 다를 경우 SHR은 잘못 예측된 경로상의 분기명령들에 대한 모험적 갱신으로 오염되어 있으므로 SHR을 복구시키지 않는 다면 예측실패 후 완료시간에 프로세서 상태가 복구되고 올바른 경로상의 분기명령을 예측하려할 때 오염된 SHR을 사용할 수밖에 없으므로 예측정확도는 낮아진다. 따라서 오염된 SHR에 대해 예측이 실패된 분기명령의 예측 직전 히스토리로 SHR을 복구시켜야 한다. 본 논문에서 제시한 복구 메커니즘은 age_counter를 사용하여 완료시간에 SHR을 예측 직전의 히스토리로 복구시킨다. 그림2에서 BC는 현재 마지막으로 완료된 분기명령의 분기결과를 나타낸다. 만약 어떤 분기명령의 실제결과가 구해졌다면 완료시간에 그 분기명령이 모험적으로 갱신된 결과인 BC-1과 실제결과를 비교한다. 만약 동일하다면 올바로 예측한 경우이므로 프로세서는 정상적으로 진행된다. 이때 age_counter를 '1' 감소시켜 분기 히스토리에서 미해결 분기명령의 수가 하나 줄었음을 나타낸다. 그러나 두 결과가 다르다면 분기예측실패가 검출된 경우이고, SHR에서 BC-2로부터 B0까지는 잘못 예측된 경로상의 분기명령들이 모험적으로 갱신됨에 따라 분기 히스토리를 오염시키게 된 것이다. 따라서 SHR을 c-1 bit만큼 오른쪽으로 이동시켜 예측이 실패된 분기명령의 예측동작 직전의 상태로 SHR을 복구시키고, 예측 실패한 분기명령 이후에 반입되어 모험적으로 갱신되었던 히스토리를 무효화 시키고 예측실패한 분기명령의 실제결과를 B0에 저장한다. 이때 age_counter의 값은 '0'으로 초기화시켜, 현재 미해결 분기명령어가 없음을 나타낸다.

4. 성능측정 및 분석

4.1. 실험환경

성능 측정을 위해 슈퍼스칼라 프로세서의 사이클 수준 시뮬레이터 Simple Scalar 3.0/PISA 툴셋(D. Burger., T. M. Austin., & S. Bennett, 1996)에서 모험적 갱신을 허용하는 gshare와 GAg 분기예측기에 본 논문에서 제안한

분기예측실패 복구 메커니즘을 적용하기 위해 bpred와 sim-outorder를 수정하였다. 표3은 시뮬레이션 된 프로세서의 머신 파라메타를 나타낸다. 8이슈 프로세서를 기반으로 1~8K 엔트리 PHT를 갖는 gshare와 GAg 분기예측기로 실험하였다. 모든 분기예측기의 BTB(Branch Target Buffer)는 512sets 4-way로 고정하였다. 128KB L1 데이터 캐시, 512KB L1 명령어 캐시, 1MB L2 통합 캐시를 사용하였다. 표4는 시뮬레이션에서 사용된 SPECINT95 벤치마크 프로그램과 입력 데이터이며, 입력 데이터는 ref input을 사용하였고 시뮬레이션 시간을 줄이기 위해 명령어수를 200M(million)로 제한하였다.

〈표3〉 시뮬레이션 프로세서의 머신 파라미터

구 분	인 수	값
Processor Core	RUU size	128 entries
	LSQ size	64 entries
	Fetch width	8 instructions/cycle
	Decode width	8 instructions/cycle
	Issue width	8 instructions/cycle
	Commit width	8 instructions/cycle
	Functional units	8 i-ALU(1), 8 f-ALU(2) 2 i-MULT/DIV(3/12) 2 f-MULT/DIV(4/12)
Memory	Memory ports	2 ports, 8byte bus
	Memory access latency	first_chunk(18), inter_chunk(2)
Branch predictor	gshare	(1K/2K/4K/8K) entries
	GAg	(1K/2K/4K/8K) entries
	BTB	512 sets, 4 way
Cache	L1 data cache	128K, 32B block, 4 way, LRU, 1 cycle latency
	L1 instruction cache	512K, 32B block, d-map, LRU, 1 cycle latency
	L2 unified cache	1M, 64B block, 4 way, LRU, 6 cycle latency

〈표4〉 벤치마크 프로그램과 입력 데이터

벤치마크	입력 데이터	수행된 명령어수 (million)
gcc	cccp.i	200
li	train.lsp	183
vortex	persons.250	200
go	50 9 2stone9.in	200
m88ksim	dcrand.lit	200
compress	100000 e 2231	200
perl	primes.pl	200
ijpeg	penguin.ppm	200

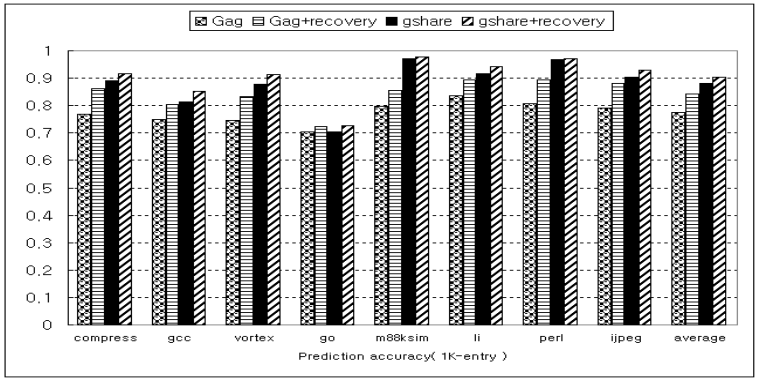
4.2. 성능 분석

1~8K 엔트리를 갖는 PHT와 모험적 갱신을 허용하는 gshare와 GAg 분기예측기에 제안된 복구 메커니즘을 비적용(GAg, gshare)과 적용(GAg + recovery, gshare + recovery)의 상태로 구분하여 예측정확도와 성능 향상(speedup)을 측정하였다.

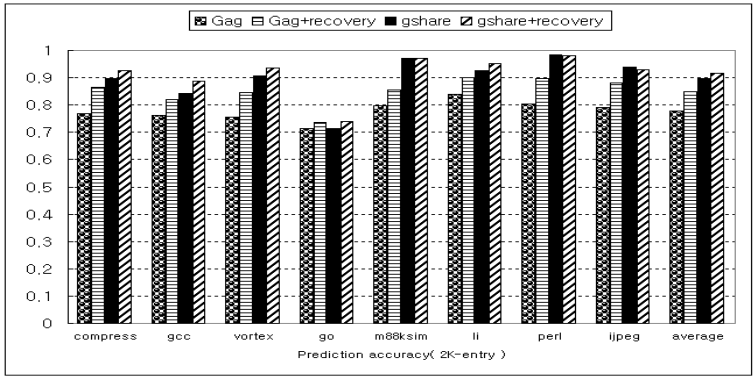
적용된 복구 메커니즘은 간단한 하드웨어 추가만으로 복잡한 하드웨어를 요구하고 있는 기존의 복구 메커니즘의 결과와 같은 예측정확도와 성능의 개선을 확인할 수 있었다. 그림4~8은 1K, 2K, 4K, 8K엔트리의 PHT를 갖는 GAg와 gshare에 분기예측실패 복구 메커니즘 적용 전과 적용 후의 예측정확도를 나타낸다. 그림4~13의 범례에서 GAg는 분기예측실패 복구 메커니즘 적용 전의 예측정확도, GAg + recovery는 GAg에 분기예측실패 복구 메커니즘 적용 후의 예측정확도를 나타낸다. 또, gshare는 분기예측실패 복구 메커니즘 적용 전의 예측정확도, GAg+recovery는 gshare에 분기예측실패 복구 메커니즘 적용 후의 예측정확도를 나타낸다. GAg에 분기예측실패 복구 메커니즘을 적용한 결과 예측정확도는 최소 8.89%(1K엔트리 PHT), 최대 9.63%(8K 엔트리 PHT), 평균 9.21% 개선되었다. GAg에서는 PHT 엔트리 수가 증가할수록 분기예측실패 복구 메커니즘이 예측정확도를 좀 더 개선시키는 것으로 나타났다. gshare에 분기예측실패 복구 메커니즘을 적용한 결과 예측정확도는 최소 1.9%(8K 엔트리 PHT), 최대 2.65%(1K 엔트리 PHT), 평균 2.14%개선되었다. gshare에서

는 엔트리 수가 적을 때 분기예측실패 복구 메커니즘에 의한 개선이 약간 좋은 것으로 나타났다.

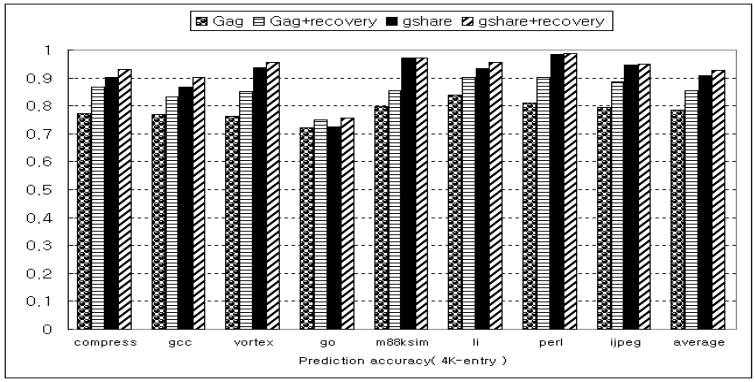
GAg + recovery가 gshare+recovery보다 예측정확도가 많이 개선된 것은 GAg는 PHT를 인덱스할 때 전역 히스토리 레지스터만 사용하지만 gshare는 전역 히스토리 레지스터와 분기명령의 주소를 exclusive-or하여 인덱스 하므로 전역 히스토리 레지스터에 대한 영향이 GAg보다 적기 때문이라 생각된다. 또, 그림 8과 같이 PHT엔트리 수 변화에 따른 예측정확도의 변화를 측정한 결과 PHT의 엔트리 수를 증가시켜도 예측정확도에 큰 변화가 없었으므로 복구메커니즘은 PHT 엔트리 수에 크게 영향 받지 않는다는 것을 확인할 수 있었다.



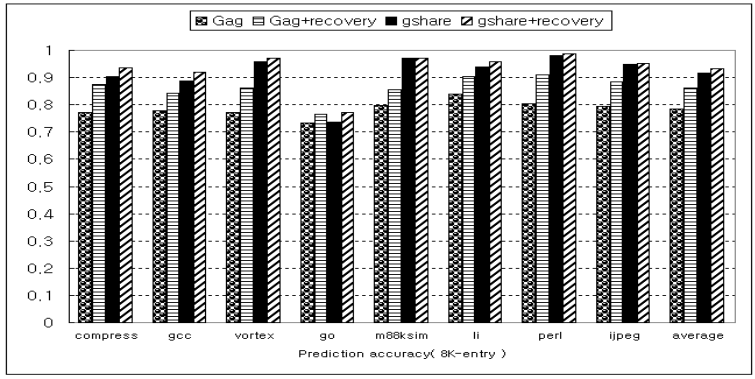
〈그림 4〉 GAg와 gshare에 PHT 1K엔트리 적용 예측정확도



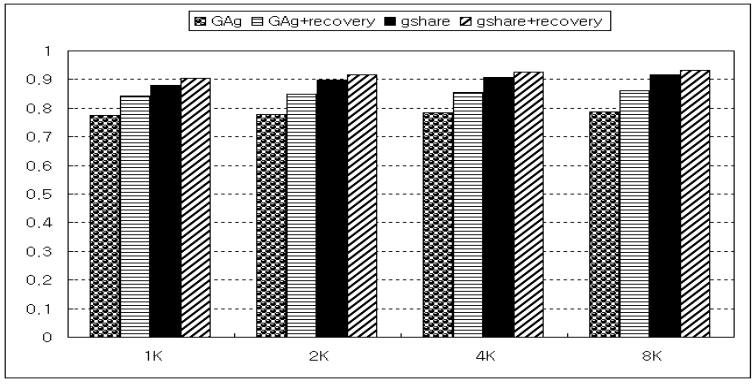
〈그림 5〉 GAg와 gshare에 PHT 2K엔트리 적용 예측정확도



〈그림 6〉 GAg와 gshare에 PHT 4K엔트리 적용 예측정확도

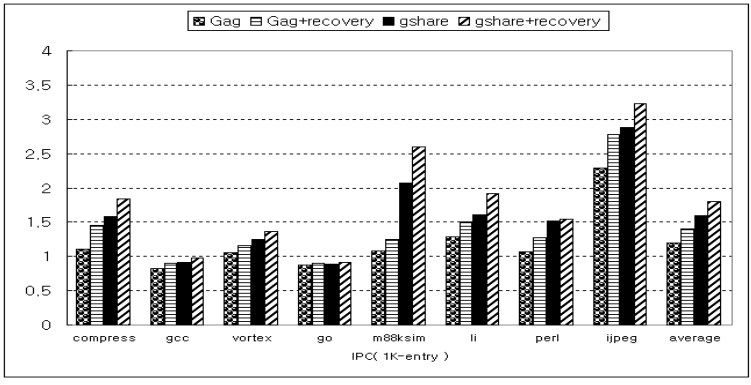


〈그림 7〉 GAg와 gshare에 PHT 8K엔트리 적용 예측정확도

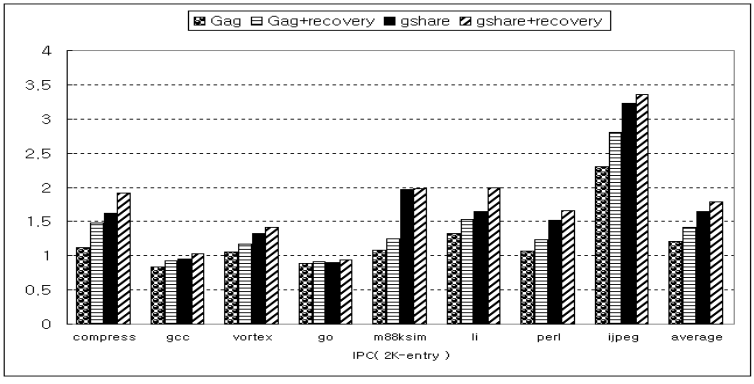


〈그림 8〉 PHT엔트리 수 변화에 따른 예측정확도 변화

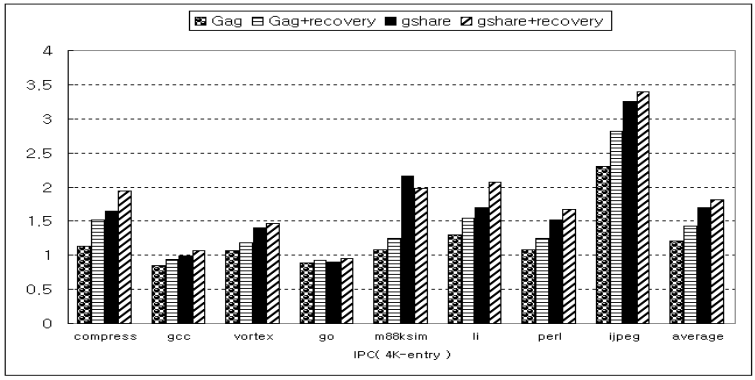
그림9~13은 GAg와 gshare에 분기예측실패 복구 메커니즘을 적용하기 전과 후의 성능 향상(speedup)을 IPC로 나타내었다. 분기예측실패 복구 메커니즘을 GAg에 적용한 결과 IPC는 최소 17%(2K 엔트리PHT), 최대 20.20%(8K 엔트리 PHT), 평균 18.08%개선되었다. IPC도 GAg에서는 PHT 엔트리 수가 증가하면 IPC가 약간 더 개선되는 것으로 나타났다. 또, 분기예측실패 복구 메커니즘을 gshare에 적용한 결과 IPC는 최소 6.36%(8K 엔트리 PHT), 최대 13.09%(1K 엔트리 PHT), 평균 8.75% 개선되었다. 이는 GAg가 gshare보다 예측정확도를 더 크게 개선시킬 수 있었으므로 프로세서의 성능이 gshare보다 더 많이 개선된 결과라 생각된다. 또, 그림 13과 같이 PHT 엔트리 수 변화에 따른 IPC변화를 측정한 결과, PHT의 엔트리 수를 증가시켜도 예측정확도에서와 같이 IPC에는 큰 변화가 없었으므로 복구메커니즘은 PHT 엔트리 수에 큰 영향을 받지 않는다는 것을 또다시 확인할 수 있었다.



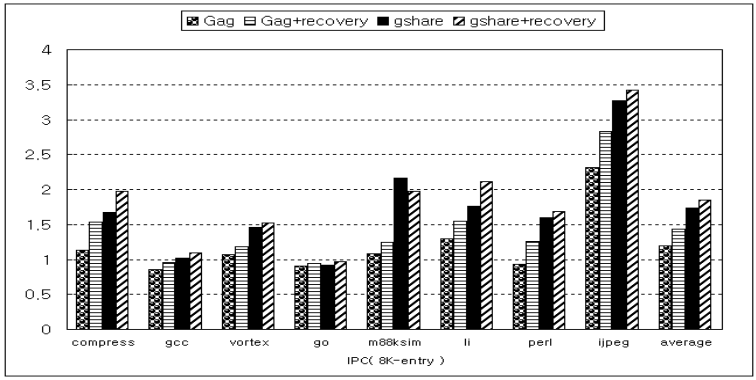
〈그림 9〉 GAg와 gshare에 PHT 1K엔트리 적용 IPC



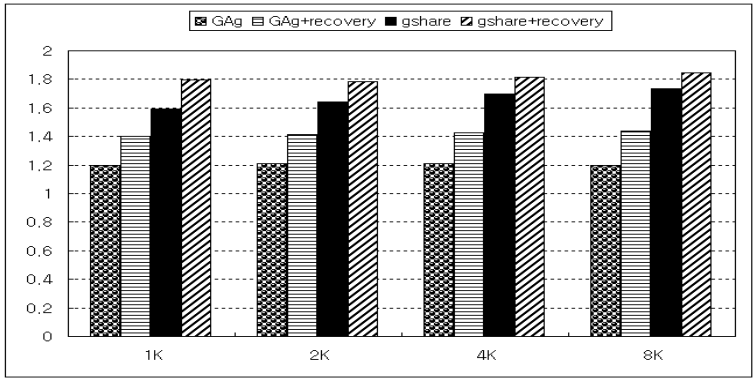
〈그림 10〉 GAg와 gshare에 PHT 2K엔트리 적용 IPC



〈그림 11〉GAg와 gshare에 PHT 4K엔트리 적용 IPC



〈그림 12〉 GAg와 gshare에 PHT 8K엔트리 적용 IPC



〈그림 13〉 PHT엔트리 수 변화에 따른 IPC의 변화

5. 결 론

본 논문에서는 전역 히스토리 레지스터를 모험적으로 갱신하는 전역 히스토리 기반 분기예측기에서 분기예측 실패 후 전역 히스토리 레지스터를 분기예측 직전의 히스토리로 복구하는 간단한 메커니즘을 제안하였다. 기존의 복구 메커니즘에서는 큐와 같은 복잡한 하드웨어의 추가가 요구되는 반면, 제안한 복구 메커니즘은 하드웨어의 특별한 추가 없이 동일한 효과를 얻었다. 전역 히스토리를 기반으로 하는 예측기인 GAg와 gshare에 제안한 복구 메커니즘을 적용하여 SimpleScalar 3.0/PISA 툴셋으로 시뮬레이션 한 결과 예측정확도의 증가와 프로세서 성능향상 효과를 확인하였다. 제안한 복구 메커니즘을 GAg 예측기에 적용한 결과 적용전보다 예측정확도는 평균 9.21%, IPC는 평균 18.08% 개선되었다. 또, gshare 예측기에 적용한 결과 예측정확도는 평균 2.14%, IPC는 평균 8.75% 개선되었다.

■ 참고문헌 ■

고광현, & 조영일. (2003). 농업정보기술을 위한 ILP 프로세서에서 정·동적 분류를 이용한 결과값 예측기, 한국농업정보과학회, 제4권 제1호, pp.43-46.

Kwang-Hyun Ko, Young-Il Cho. (2008). *A Branch Predictor with New Recovery Mechanism in ILP Processors for Agriculture Information Technology, World Conference on Agricultural Information and IT-IAALD-AFITA-WCCA 2008.*

T.-Y. Yeh., & Y. N. Patt, (1992). Alternative implementations of two-level adaptive branch prediction, in Proceedings of the 19th Annual International Symposium on Computer Architecture, pp. 124-34, May.

K. Diefendorff, (1998). K7 challenges Intel, Microprocessor Report, pp. 1, 6-11, Oct.

R. E. Kessler., E. J. McLellan., & D. A. Webb, (1998). The Alpha 21264 microprocessor architecture, in Proceedings of the 1998 International Conference on Computer Design, pp. 90-95, Oct.

G.H. Loh., & D.S. Henry, (2002). Predicting Conditional Branches with Fusion-based Hybrid Predictors, PACT2002, pp 395-405, Sep.

Z. Lu., J. Lach., M. Stan., & K. Skadron, (2003). Alloyed Branch History: Combining Global and Local Branch History for Robust Performance, International Journal of Parallel Programming, Kluwer, volume 31, number 2, Apr.

A. R. Talcott., W. Yamamoto., M. J. Serrano., R. C. Wood., & M. Nemirovsky, (1994). The impact of unresolved branches on branch prediction scheme performance, in Proceedings of the 21st Annual International Symposium on Computer Architecture, pp. 12-21, Apr.

E. Hao., P.-Y. Chang., & Y. N. Patt, (1994). The effect of speculatively updating branch history on branch prediction accuracy, revisited, in Proceedings of the 27th Annual International Symposium on Microarchitecture, pp. 228-32, Nov.

M. Evers., S. J. Patel., R. S. Chappell., & Y. N. Patt. (1998). An analysis of correlation and predictability: What makes two-level branch predictors work, in Proceedings of the 25th Annual International Symposium on Computer Architecture, pp. 52-61, June.

K. Skadron., & M. Martonosi, (2000). Speculative Updates of Local and Global Branch History: A Quantitative Analysis, JILP Vol. 2, Jan.

S. Jourdan, J. Stark., T.-H. Hsing, & Y. N. Patt, (1997). Recovery requirements of branch prediction storage structures in the presence of mispredicted-path execution, International Journal of Parallel Programming, vol. 25, pp. 363-83, Oct.

K. Skadron., P. S. Ahuja., M. Martonosi., & D. W. Clark, (1998). Improving prediction for procedure returns with return-address-stack repair mechanisms, in Proceedings of the 31st Annual ACM/IEEE International Symposium on Microarchitecture, pp. 259-71, Dec.

D. Burger., T. M. Austin., & S. Bennett, (1996). *Evaluating future microprocessors: the SimpleScalar tool set,* Tech. Report TR-1308, University of Wisconsin-Madison Computer Sciences Department, July.

The Standard Performance Evaluation Corporation, (1999). SPEC CPU95 Benchmarks, from http://www.specbench.org/osg/cpu95, Dec.