정확하고 효율적인 간접 분기 예측기 설계

*백경호, 김은성
순천향대학교 공과대학 정보기술공학부
E-mail : pkhtow@hitel.net

Design of Accurate and Efficient Indirect Branch Predictor

*Kyoung-Ho Paik, Eun-Sung Kim
Div. of Information Technology Engineering,
Soonchunhyang University

요 약

Modern superscalar processors exploit Instruction Level Parallelism to achieve high performance by speculative techniques such as branch prediction. The indirect branch target prediction is very difficult compared to the prediction of direct branch target and branch direction, since it has dynamically polymorphic target.

We present a accurate and hardware-efficient indirect branch target predictor. It can reduce the tags which has to be stored in the Indirect Branch Target Cache without a sacrifice of the prediction accuracy. We implement the proposed scheme on SimpleScalar and show the efficiency running SPEC95 benchmarks.

I. 서론

현대 슈퍼스칼라 프로세서에서의 더 깊은 파이프라인과 명령어 이슈율의 증가로 인해 발생되는 명령어 수준의 병렬성에 대한 제약을 제거 또는 완화시켜 보다 좋은 성능을 위해 투기적인 실행은 필수적인 요소가 되고 있다. 따라서 이러한 투기적인 실행을 유지하기 위해서는 정확한 분기 예측이 필요하다. 직접 분기 명령의 타겟은 BTB를 사용하여 거의 정확하게 예측할 수 있으며[1], 조건 분기 명령의 분기 조건 결정을 위해 분기 방향 예측기가 사용되는데 이에 대한 많은 연구로 인해 아주 높은 정확도를 얻을 수 있다[2]. 그러나 간접 분기의 타겟은 동적으로 다형태의 특성을 보이기 때문에 예측하기 매우 어려워 이에 대한 많은 연구가 있었으나 분기 방향에 대한 것보다 낮은 예측 정확도를 보이고 또한 예측기에 대한 하드웨어 부담이 크다[3-5]. 더욱이 간접 분기를 많이 포함하고 있는 C++나 Java와 같은 객체지향언어의 사용이 현격히 증가하고 있기 때문에 예측 정확도가 아주 높으면서도 하드웨어 측면에서 효율적인 간접 분기 예측기가 절실히 필요한 실정이다[6,7].

우리는 이전 방식과는 전혀 다를 뿐만 아니라 예측 정확도가 아주 뛰어난 간접 분기 예측 방식 즉, 이미 간접 분기 명령과 이와 데이터 종속 관계를 가지고 있는 이 명령어 보다 훨씬 앞서 수행되는 명령어의 레지스터 내용을 결합하여 간접 분기의 타겟을 예측해내는 방식을 제안하였으나[9], 기존의 예측기들과 마찬가지로 하드웨어 오버헤드가 크다는 단점을 안고 있다. 따라서 본 논문에서는 예측 정확도에 대한 손실이 없으면서 예측기의 IBTC(간접 분기 타겟 캐시)에 저장되는 태그의 크기를 최소화할 수 있는 방식을 제안한다. 또한 제안된 방식을 심플스칼라[8] 상에서 구현하고, SPEC95 벤치마크를 사용하여 그 효율성을 검증한다.

II. 데이터 종속성의 간접 분기 예측기

2.1 간접 분기 예측기의 메커니즘

본 논문의 예측기는 간접 분기 명령과 이와 데이터 종속 관계를 가지고 있는 이 명령어 보다 훨씬 앞서 수행되는 명령어의 레지스터 내용을 결합하여 간접 분기의 타겟을 예측하는 메커니즘이다[9].

프로그램 코드에는 간접 분기 명령어와 절대적 상관관계가 있는 명령어의 해당 레지스터를 명확히 명시해 두고, 이 명령을 해독하여 수행할 때 그 명시된 레지스터 즉, 최종 절대적 상관관계에 있는 레지스터를 레지스터 FAR(Final Absolutely-correlated Register)로 불러온다. 그런 다음, 그림 1에서와 같이 FAR과 간접 분기 주소 PC를 사용하는 해싱 함수에 의해 생성된 인덱스로 예측 분기 타겟 주소를 저장해 놓은 IBTC를 액세스한다[10].

IBTC의 해싱 방법으로는 적절한 엔트리 선택을 위해 FAR과 PC를 연쇄연결(concatenate)하는 방식을 사용한다. 앨리어싱을 기능한 한 피하기 위해 제한된 크기의 인덱스에 분기 주소와 FAR의 비트를 각각 얼마만큼 포함시키는 지에 따라 성능에 다양한 편차를 보이게 된다. 실험에 따르면 각각의 비트를 절반씩 포함시키는 것이 가장 좋은 성능 결과를 보였다.

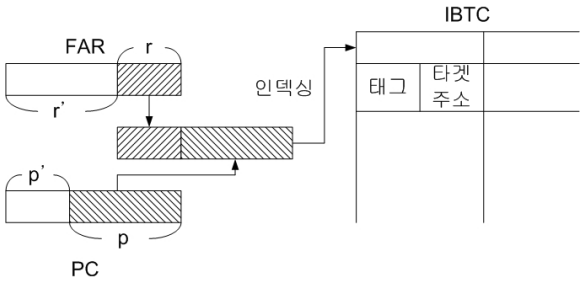

그림 1. 데이터 종속 기반의 간접 분기 예측기

2.2 IBTC의 태그 구성 방식

IBTC를 액세스하기 위한 해싱 방법과 더불어 IBTC에 저장되는 태그의 구성도 성능에 영향을 미치게 된다. 일반적으로 성능을 고려하여 IBTC는 집합-연관 사상 방식으로 구현하고, IBTC에는 예상 타겟 주소와 함께 태그 정보를 저장해야 하므로 IBTC의 하드웨어 부담이 커지게 되어 성능 손실을 최소화하면서 동시에 태그 매칭에 사용되는 정보를 최적화하는 방법이 필요하다.

그림 2에서처럼 IBTC에 저장되는 태그는 부정적인 간섭을 피하기 위해서 사용할 수 있는 다양한 태그 방식들을 보여준다.

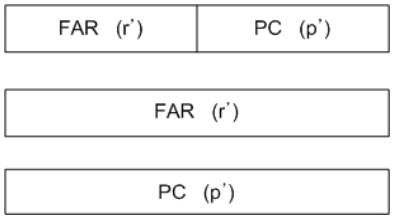

그림 2. IBTC의 태그 구성 방식

첫 번째 방식은 가장 일반적인 방식으로, 그림 1에서 설명한 것처럼 IBTC의 인덱스를 레지스터 FAR의 하위 부분 r 비트와 분기 주소 PC의 하위 부분 p 비트로 구성하고, 태그로는 인덱스로 사용되지 않은 그 나머지 상위 비트 부분 즉, r′ + p′로 구성하는 것이다. 두 번째는 레지스터 FAR에서 인덱스로 사용되지 않은 상위 비트들(r′)로만 태그를 구성하는 것이고, 세 번째는 분기 주소에서 인덱스로 사용되지 않은 상위 비트들(p′)로만 태그를 구성하는 것이다. 예를 들어 메모리 주소와 레지스터가 각각 4 바이트인 현재 가장 일반적인 32 비트 프로세서에서 1K 엔트리의 IBTC를 사용하고 인덱스로 FAR과 PC에서 각 5 비트씩을 사용한다고 가정하면, 첫 번째 방법으로 했을 경우 저장되는 태그는 52 비트가 되고(대부분의 프로세서는 4 바이트 메모리 정렬을 사용하기 때문에 명령어 주소는 30 비트로 구분할 수 있다), 두 번째와 세 번째와 같이 태그를 구성하는 경우에는 각각 27 비트와 25 비트의 태그의 크기를 가진다.

실험 결과에 따르면 분기 주소에서 사용되지 않은 비트들로만 태그를 구성하는 세 번째 방식의 예측 정확도가 다른 두 가지 방식보다 우수하게 나타났다.

2.3 태그 크기의 최소화

이전 절에서 알 수 있듯이 분기 주소로만 태그를 구성하는 p′ 방식이 예측 정확도도 우수할 뿐만 아니라 특히 가장 일반적인 형태인 p′ + n′ 방식에 비해 하드웨어 오버헤드를 크게 줄일 수 있다.

그러나 간접 분기 예측기의 IBTC에는 태그뿐만 아니라 예상 타겟 주소도 함께 저장해 주어야 하기 때문에 하드웨어 부담이 커지므로, 본 논문에서는 저장되는 태그 비트 수를 최소화시키는 방법을 제시한다. 즉, p′ 방식의 태그 구성을 사용하는 경우에, IBTC에 태그 비트를 p′ 보다 적게 저장하도록 한다. 정상적인 경우보다 저장되는 태그가 적어질 때 통상적으로 부정적인 간섭으로 인해 성능이 떨어질 수 있다. 저장되는 태그 크기와 성능과는 일반적으로 trade-off 관계가 성립된다. 따라서 본 논문에서는 태그에 저장되는 비트수를 가변적으로 줄여가면서 나타나는 예측 정확도를 측정

표 1. IBTC의 태그 비트 수의 변화에 따른 예측 정확도

		IBTC의 크기 집합-연관 사상								
		64			1 K			16 K		
		32 × 2	16 × 4	8 × 8	512 × 2	256 × 4	128 × 8	8 K × 2	4K × 4	2 K × 8
벤치마크	태그에 저장되는 비트수	태그에 저장되는 비트수에 대한 예측 정확도(%) (IBTC에 저장되는 최대 태그 비트 수)								
Gcc	1	61.67	33.28	32.77	88.18	58.45	40.81	57.61	47.30	55.49
	3	72.07	73.22	61.67	96.07	95.02	89.31	81.56	78.53	77.69
	5	72.91	77.36	76.82	96.65	97.99	98.08	91.33	91.28	95.05
	7	**72.98**	77.53	78.01	**96.80**	98.26	98.28	95.88	95.92	99.69
	9	72.98	**77.54**	**78.10**	96.80	**98.28**	**98.67**	**99.82**	**99.85**	**99.88**
	Max	72.98 (26)	77.54 (26)	78.09 (26)	96.80 (24)	98.28 (23)	98.67 (23)	99.82 (22)	99.85 (21)	99.88 (21)
Perl	1	92.83	9.33	18.04	91.66	9.35	27.88	39.11	17.18	93.43
	3	98.87	98.91	92.83	99.58	99.59	90.62	99.99	98.94	99.99
	5	**98.76**	99.55	98.91	**100.00**	**100.00**	99.59	**100.00**	99.99	**100.00**
	7	98.76	**99.90**	**99.90**	100.00	100.00	**100.00**	100.00	**100.00**	100.00
	9	98.76	99.90	99.90	100.00	100.00	100.00	100.00	100.00	100.00
	Max	98.76 (26)	99.90 (26)	99.90 (26)	100.00 (22)	100.00 (22)	100.00 (22)	100.00 (22)	100.00 (22)	100.00 (22)
Vortex	1	95.37	63.20	53.03	84.69	76.08	67.19	54.74	79.31	76.08
	3	96.45	93.21	74.01	**99.58**	99.44	98.95	95.49	95.07	99.41
	5	**96.65**	**94.12**	75.57	99.58	**99.88**	99.61	95.65	97.39	**99.96**
	7	96.63	94.04	**82.30**	99.58	99.88	**99.91**	**99.96**	97.39	99.96
	9	96.63	94.04	82.30	99.58	99.88	99.91	99.96	**99.96**	99.96
	Max	96.63 (24)	94.04 (25)	82.30 (26)	99.88 (23)	99.88 (23)	99.91 (22)	99.96 (20)	99.96 (20)	99.96 (20)

하였다. 만약 태그에 저장되는 비트 수를 줄여도 예측 정확도가 크게 떨어지지 않거나 비슷하다면, 하드웨어 오버헤드를 크게 줄일 수 있을 것이다.

Ⅲ. 성능 분석

본 논문의 실험 결과는 제안한 간접 분기 예측기를 심플스칼라 상에서 구현하고, 다양한 IBTC 크기에 대해 분기 주소와 FAR를 연쇄 연결하는 해싱 방식과 분기 주소의 상위 비트들만을 태그로 저장하는 p′ 방식을 적용하고 태그에 저장되는 p′ 비트 수를 가변적으로 변화시켜가면서 나타나는 간접 분기 타겟 예측율을 측정하였다. 벤치마크 프로그램으로는 SPEC95를 사용하였고, 실험 결과는 지면 관계 상 Gcc와 Perl 및 Vortex만 제시하였다.

표 1은 IBTC의 크기를 64, 1 K 및 16 K로 하고 다양한 집합-연관 사상을 사용하였을 때 IBTC에 저장되는 태그 비트 수에 따른 예측 정확도를 보여준다. 또한 인덱스로 사용되는 분기 주소의 비트 수는 이전의 논문에서 가장 높은 예측 정확도를 보인 것으로 사용하였다. 예를 들어 Gcc의 경우, IBTC의 크기가 512 × 2로 사용하면, 인덱스에 사용되는 분기 주소의 비트수는 5비트(나머지 4 비트는 FAR에서 사용)이고 태그에 저장되는 최대 비트수는 24비트임을 나타낸다. 4 바이트 메모리 정렬을 사용하는 대부분의 32 비트 프로세서는 메모리 주소를 30 비트로 구분이 가능하지만 심플스칼라에서는 annotation field를 포함하여 opcode가 3 바이트로 구성되어 있기 때문에 3 비트의 LSB를 제외시켜 29 비트로 메모리 주소를 구분한다. 따라서 분기 주소 29 비트 중에서 하위 5 비트를 인덱스에 이용하고 나머지 24 비트가 태그로 저장되는 것이다.

Perl에 대한 예측 정확도 중에 100%라는 의미는 99.999%이상을 말한다. 즉, IBTC에 최초에 저장된 후에는 항상 정확한 예측이 가능하게 된다는 것이다.

표 1에서 보듯이 각 벤치마크에 대해 다양한 크기와 집합-연관 사상 방식을 적용하였을 때 저장되는 태그 비트 수를 10 비트까지 줄여도 전체 비트를 저장하는 방식과 동일한 예측 정확도를 보임을 알 수 있다. 따라서 저장하는 태그 비트 수를 10 비트로 하면 사용하는 IBTC의 크기와 집합-연관 사상에 따라 50%에서 61.5%까지 절약할 수 있다.

그림 3은 1K 크기의 IBTC에 대해 여러 가지 집합-연관 사상을 사용하였을 때의 각 벤치마크에 대한 예측 정확도를 부분적으로 확대하여 자세히 나타내었다.

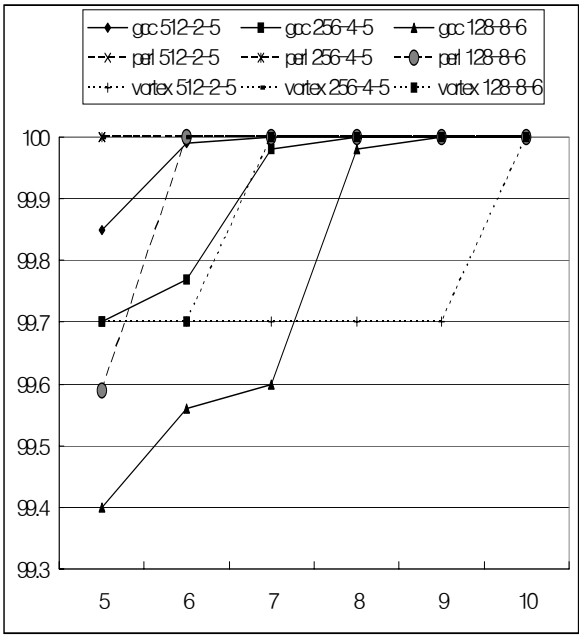

그림 3. 1 K의 IBTC에 대한 부분적인 예측 정확도

여기서는 태그에 저장되는 비트가 최대일 경우의 예측 정확도 100%로 했을 때 태그에 저장되는 각 비트 수에 따른 상대적인 예측 정확도를 보였다. 표 1에서 알 수 있듯이 태그로 저장되는 비트 수가 4 비트 이하가 되면 예측 정확도는 급격히 나빠진다. 그러므로 그림 3에는 태그에 저장되는 비트 수가 5 비트 이상인 경우만을 나타내었다. 태그에 저장되는 비트 수를 6 비트로 하여도 각 벤치마크에 대해 예측 정확도가 최소 99.56% ~ 100.00%까지의 상대적인 예측 정확도를 보임을 알 수가 있다. 따라서 태그에 저장되는 비트 수는 크게 줄어들어 하드웨어 측면에서 매우 효율적인 예측기의 구현이 가능하게 된다.

V. 결론

본 논문에서는 기존에 제안하였던 간접 분기 예측기의 하드웨어 오버헤드를 줄일 수 있는 방식을 제안하였다. 즉, 예측 정확도에 대한 손실이 없으면서 동시에 예측기의 IBTC(간접 분기 타겟 캐시)에 저장되는 태그의 크기를 최소화할 수 있는 방식을 제시하였다. 실험 결과로 나타낸 바와 같이 IBTC의 태그를 구성함에 있어서 인덱스로 사용되지 않은 분기 주소의 그 나머지 상위 부분을 모두 사용하는 경우보다 그 일부분을 태그로 사용하여도 예측 정확도에 거의 손실이 없음을 알 수 있었다. 저장하는 태그 비트 수를 10 비트로 하면 예측 정확도의 손실이 전혀 없이 IBTC의 태그 크기를 50%에서 61.5%까지 절약할 수 있다. 또한 태그 비트 수를 6 비트로 제한할 경우 상대적인 예측 정확도를 99.56% ~ 100%까지 유지할 수 있었다. 따라서 태그에 저장되는 비트 수는 크게 줄어들어 하드웨어 측면에서 매우 효율적인 예측기의 구현이 가능하여 실제적인 상용화가 가능할 것이다.

참고문헌

[1] C. Perleberg and A. J. Smith, “Branch Target Buffer Design and Optimization”, IEEE Transactions on Computers, 42(4), pp. 396-412, Apr. 1993.

[2] D. A. Jimenes and C. Lin, “Dynamic Branch Prediction with Perceptron”, 7th Int'l Symp. on High Performance Computer Architecture, pp. 197-206, Monterrey, Mexico, Jan. 2001.

[3] P. Y. Chang, E. Hao and Y. N. Patt, “Target Prediction for Indirect Jumps”, 24th Int'l Symp. on Computer Architecture, pp. 274-283, Denver, U.S.A., June 1997.

[4] K. Driesen and U. Holzle, “Accurate Indirect Branch Prediction”, 25th Int'l Symp. on Computer Architecture, pp. 167-178, Barcelona, Spain, July 1998.

[5] K. Driesen and U. Holzle, “The Cascaded Predictor: Economical and Adaptive Branch Target Prediction”, Micro '98 Conference Proceedings, Dallas, Texas, Dec 1998

[6] O. J. Santana, A. Falcon, E. Fernandez, P. Medina, A. Ramirez and M. Volero, “A Comprehensive Analysis of Indirect Branch Prediction”, 4th Int'l Symp. on High Performance Computing, pp. 133-145, Kansay Science City, Japan, May 2002.

[7] M. Anton Ertl and David Gregg, “Optimizing Indirect Branch Prediction Accuracy in Virtual Machine Interpreters”, Proc. of the ACM SIGPLAN 2003 Conference on PPLDI, pp. 278-288, San Diego, California, June 2003.

[8] D. Burger and T. M. Austin, “The SimpleScalar Tool Set, Version 2.0”, Technical Report CS-RT-97-1342, University of Wisconsin, Madison, June 1997.

[9] 백경호, 김은성, “간접 분기의 타형태 타겟 주소의 정확한 예측”, 전자공학회논문집(CI) 제41권 6호, pp. 511-522‘, 2004. 11.

[10] 백경호, 김은성, “정확한 간접 분기 예측기의 태그 특성 분석”, 대한전자공학회 하계학술대회 논문집 제28권 제1호, pp. 107-111 2005, 6