독립시행의 정리를 이용하는 수퍼스칼라 프로세서의 다중 분기 예측 성능 모델

이종복
LG 반도체 시스템 IC 사업본부 프로세서팀
tel : 02-3459-3353, fax : 02-3459-5842

The Analytic Performance Model of the Superscalar Processor Using Multiple Branch Prediction

Jongbok Lee
Processor Team, LG Semicon, Daechi-Dong 891, Gangnam-Gu, Seoul
e-mail : jongbok@lgsemicon.co.kr

Abstract

An analytical performance model that can predict the performance of a superscalar processor employing multiple branch prediction is introduced. The model is based on the conditional independence probability and the basic block size of instructions, with the degree of multiple branch prediction, the fetch rate, and the window size of a superscalar architecture. Trace driven simulation is performed for the subset of SPEC integer benchmarks, and the measured IPCs are compared with the results derived from the model. As the result, our analytic model could predict the performance of the superscalar processor using multiple branch prediction within 6.6 percent on the average.

1 서 론

복수개의 기본블럭에서 명령어 단위 병렬도를 추출해내기 위하여 다중 분기 예측법이 수퍼스칼라 마이크로 프로세서에 널리 쓰이고 있다[1]. 예를 들어, Intel의 p6 마이크로 아키텍쳐는 동적 실행을 위하여 다중 분기 예측법을 이용하며, Sun Microsystem의 Ultra-Sparc I은 한 싸이클에 실행되는 명령어의 개수를 증가시키기 위하여 두 개의 기본블럭에서 명령어를 발행한다. 따라서, 다중 분기 예측법을 사용하는 수퍼스칼라 프로세서의 성능을 올바로 이해하고 측정하며 예측하는 것이 중요하다. 비록 명령어 자취 모의실험 (trace-driven simulation)이 정확하고 널리 쓰이고 있으나 많은 시간이 소요되며, 아키텍쳐가 바뀜에 따라 매번 반복되어야 하는 단점을 지닌다. 따라서, 프로세서 아키텍쳐의 성능 모델을 개발해 내고 벤치마크에 대한 특성화 작업에 대한 필요성이 대두되고 있다[2].

본 논문에서는, 다중 분기 예측법을 이용하는 수퍼스칼라 프로세서의 간단한 성능 모델을 제시한다. 본 성능 모델은 벤치마크 프로그램의 조건부 독립 확률과 기본 블럭의 크기가 주어졌을 때, 수퍼스칼라 아키텍쳐의 다중분기 차수, 인출율 및 윈도우 크기의 함수로 IPC를 계산할 수 있다. 다중 분기 예측 차수가 각각 1, 2, 3인 경우, 8 개의 정수형 SPEC 벤치마크 프로그램의 성능을 명령어 자취 모의실험에 의하여 측정하며, 그 결과를 성능 분석 모델에 의하여 얻은 값과 비교한다.

2 본 론

2.1 수퍼스칼라 프로세서의 기본구조

본 논문에서는 기본구조로서, 그림 1에 보인 바와 같은 수퍼스칼라 프로세서 구조에 다중 분기 예측 하드웨어를 추가하였다. 최대 3 개의 분기 명령어를 한 싸이클에

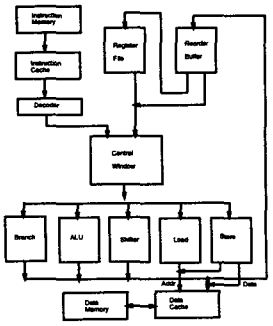


그림 1: 수퍼스칼라 프로세서의 기본구조.

인출하기 위하여 이중 적응형 다중 분기 예측법을 이용하였다. 명령어 캐쉬에서 한 싸이클에 가져올 수 있는 최대 명령어 수인 인출율은 16으로 정하였다. 윈도우의 크기는 32, 64, 및 128이며, 피연산자가 준비된 명령어는 순서와 관계없이 실행될 수 있다. 연산 유닛의 개수는 무한하고, 발행된 명령어는 단위 시간만에 실행이 완료된다고 가정하였다.

2.2 성능 분석 모델

동적 명령어의 흐름 $I_0, I_1, I_2, \cdots, I_{W-1}$로 구성된 크기가 W인 윈도우를 고찰한다. 두 개 명령어가 조건부 독립일 확률은 오직 두 명령어 간의 거리만의 함수이며, 이것이 p_δ로 일정하다고 가정한다 [2]. 이 때, 윈도우 $I_0, I_1, \cdots, I_{W-1}$에서 정확히 k개의 명령어를 실행하는 확률인 $P(W,k)$는 독립시행의 정리를 따르는 이항분포로 볼 수 있다. 윈도우의 첫번째 명령어는 항상 실행되므로 첫번째 명령어를 제외시키면, 크기 $W-1$인 윈도우에서 $k-1$개의 명령어를 실행시키는 확률로 귀착된다. 조건부 독립 확률 p_δ가 임의의 명령어가 실행되는 확률이며, 1-p_δ가 실행되지 않는 확률임을 감안하면 이 값은 다음과 같이 나타낼 수 있다.

$$P(W,k) = {}_{W-1}C_{k-1} \times p_\delta^{k-1} \times (1-p_\delta)^{W-k} \quad (1)$$

위 식에서 분기 명령어를 고려하면, 매 싸이클 당 실행가능한 명령어의 범위는 W가 아닌 기본블럭의 크기 $\lceil B_s \rceil$에 국한된다. 따라서, 단일 분기 예측법을 이용할 때 정확히 k개의 명령어를 실행할 확률은 아래 식과 같다.

$$P_1(W,k) = {}_{\lceil B_s \rceil - 1}C_{k-1} \times p_\delta^{k-1} \times (1-p_\delta)^{\lceil B_s \rceil - k} \quad (2)$$

다중 분기 예측의 차수가 2 차 및 3 차인 경우, 실행되는 명령어의 범위는 인출율과 윈도우의 크기가 충분한 경우 각각 기본블럭 크기의 2 배 및 3 배가 된다. 따라서, 이 때 각 분기 차수에 해당하는 확률인 $P_2(W,k)$와 $P_3(W,k)$는 다음식과 같이 표현된다.

$$P_2(W,k) = {}_{\lceil 2B_s \rceil - 1}C_{k-1} \times p_\delta^{k-1} \times (1-p_\delta)^{\lceil 2B_s \rceil - k} \quad (3)$$

$$P_3(W,k) = {}_{\lceil 3B_s \rceil - 1}C_{k-1} \times p_\delta^{k-1} \times (1-p_\delta)^{\lceil 3B_s \rceil - k} \quad (4)$$

한편, 기본블럭의 배수가 윈도우의 크기 W를 초과하는 경우에는, 실행되는 명령어의 범위를 W로 근사시킨다.

매 싸이클 마다 실제로 인출되는 명령어의 개수가 g_i이고 그 최대값이 인출율인 k일 때, 1 차, 2 차, 및 3 차의 다중분기예측법을 이용하는 수퍼스칼라 프로세서의 IPC는 다음과 같이 나타낼 수 있다.

$$ipc_m = g_k P_m(W,k) + \sum_{i=1}^{k} g_i P_m(W,i) \quad (m = 1,2,3) \quad (5)$$

따라서, 다중 분기 예측 차수가 m이고, 윈도우의 크기가 W이며, 인출율이 k인 수퍼스칼라 프로세서에 대한 벤치마크 프로그램의 조건부 독립 확률이 p_δ이고 기본블럭의 크기가 B_s일 때, 그 성능을 얻을 수 있다. 위 식에서, 캐쉬 미스에 및 분기 해소에 의한 지연은 무시되므로, g_i는 i에 해당한다.

2.3 모의실험 환경

명령어 자취 모의실험은 SPARC를 기반으로 하여 수행하였으며, 표 1에 실험에 사용된 8 개의 SPEC 정수형 벤치마크를 수록하였다. 이 프로그램들을 C 컴파일러에 의하여 오브젝트 코드를 얻은 후에 Shadow를 통과시켜 천만 개의 명령어 자취를 발생시켰다 [3]. 이렇게 하여 발생된 명령어 자취는 수퍼스칼라 모의실험기로 입력되었으며, 이 흐름을 그림 2에 도시하였다. 한편, 분기 예

표 1: SPEC 정수형 벤치마크.

벤치마크	기 술
eqntott	진리표 생성기
espresso	불리언 함수 최소화기
lisp	lisp 해석기
gcc1	GNU C 컴파일러의 패스 1
go	바둑 게임
ijpeg	jpeg 영상 압축
mk88sim	Motorolla 88100 모의실험기
perl	문자 및 숫자 처리기

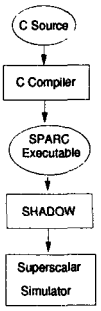


그림 2: 모의실험 과정.

측 정확도는 100 % 맞다고 가정하였으므로, 본 실험에서 다중 분기 예측에 의하여 프로세서가 얻을 수 있는 최고의 성능을 고찰할 수 있다.

각 벤치마크 프로그램의 조건부 독립 확률 p_δ와 기본 블럭의 크기 B_s를 측정하기 위하여 초기 명령어 자취 모의실험을 행하였다. p_δ의 값을 얻기 위하여 다음과 같은 방법이 사용되었다. 먼저 각 윈도우의 위치에서 명령어가 실행될 확률을 모의실험에 의하여 측정한 후, 초항이 1이고 항수가 W이며, 공비가 1보다 작은 등비수열을 가정하였다. 두 수열의 각 항을 비교하여 오차의 제곱을 구한 후에, 가정한 등비수열의 공비를 1보다 작은 범위 내에서 증가시켜 위 과정을 반복하였다. 이 때, 오차의 제곱이 최소인 공비를 각 벤치마크 프로그램의 p_δ로 정하였다.

2.4 모의실험 결과

그림 3은 모의실험에서 측정한 윈도우 내 명령어의 실행 확률과 이것과 가장 근접한 등비수열을 *perl*에 대하여 나타낸 것이다. 이 때의 공비는 0.903을 나타내었으며, 두 곡선이 비교적 잘 맞음을 볼 수있다. 실험 결과, *xlisp*와 *perl*의 등비수열은 실행 확률과 매우 근접하지만, *gcc*와 *ijpeg*는 다소의 편차를 보였다.

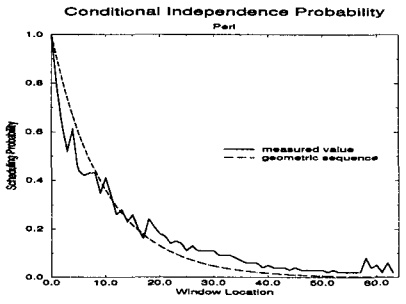
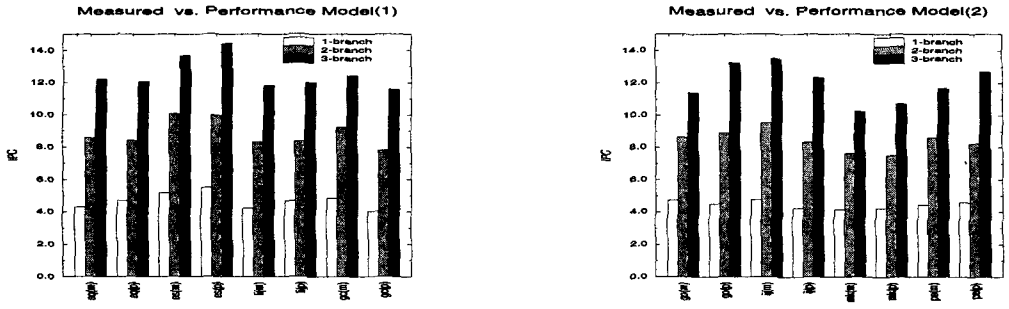


그림 3: 실행 확률 대 등비수열

실제로 p_δ의 값은 일정하지가 않으며, 윈도우의 크기가 2 또는 4와 같이 작은 값에서 증가할 때 다소의 변화를 보인다. 그러나 윈도우의 크기가 충분히 큰 경우, p_δ의 값은 포화된 값을 나타내었다. 본 논문에서는, 윈도우의 크기가 128인 초기의 명령어 자취 모의실험에서 측정된 p_δ 값을 이용하였다. 표 2에 각 벤치마크 프로그램의 조건부 독립 확률을 각 기본블럭의 크기와 함께 나타내었다.

그림 4는 명령어 자취 모의실험에 의하여 측정한 IPC와 성능 분석 모델에 의하여 계산한 IPC를 비교한 것이다. 본 결과는 1 차, 2 차, 및 3 차의 다중 분기 예측을 이용한 경우로서, 본 그림에서는 윈도우의 크기가 128인 경우만을 나타내었다. 모의실험 결과, 윈도우의 크기가 32이고 1 차에서 2 차의 분기 예측으로 이행하는 경우, 측정된 성능 향상의 기하평균은 1.79를 기록하였으며, 3 차의 분기 예측인 경우에는 이 값이 2.17로 증가하였다. 윈도우의 크기가 64인 경우에 각 값은 1.90과 2.57을, 마지막으로 128인 경우에 각각 1.92와 2.65를 기록하였다.

모의실험에 의한 측정값과 모델에 의한 계산값을 비교하면, *eqntott, espresso, xlisp*는 평균오차가 4.2 % 미



(a) (b)

그림 4: IPC의 측정값과 모델값의 비교.

표 2: 조건부 독립 확률과 기본블럭의 크기

benchmarks	basic block	p_δ
eqntott	4.29	0.923
espresso	5.25	0.896
xlisp	4.24	0.917
gcc1	4.91	0.754
go	5.00	0.812
ijpeg	4.78	0.787
mk88sim	4.15	0.811
perl	4.43	0.903

만으로 매우 낮은 편이다. 그러나, 이미 예견된 것과 같이, *gcc*는 13.3 %의 평균오차를 기록하였다. 비록 p_δ값이 비교적 낮은 오차로 근사되었어도, 기본블럭의 배수를 취하는 과정에서 자리올림 오차가 발생한다. 따라서, *perl*의 계산값은 기대된 것 만큼 정확한 결과를 가져다주지 않았다. 본 모의실험을 통하여 평균적인 오차는 6.6 %를 기록하였다.

3 결 론

본 논문에서는 다중 분기 예측법을 이용하는 수퍼스칼라 프로세서의 성능 분석 모델을 독립 시행의 정리를 이용하여 제안하였다. 이것은, 과도한 재귀식의 계산에 의존하는 기존의 방식을 개선시키고, 다중 분기 예측으로 응용 및 확장시킨 것이다.

본 모델에 의한 계산 결과는 측정결과와 평균 6.6 %의 오차로 일치하였으며, 실행 확률값과 등비수열 간에 다소 편차가 존재하여도 그 오차는 13.3 %를 초과하지 않았다.

향 후 연구과제로서, 유한한 개수의 연산유닛을 가지며 실제의 분기 예측 정확도를 반영하는 보다 실제적인 성능 모델을 들 수 있다. 또한, 실수형 벤치마크 프로그램과 같이 명령어가 단위 싸이클에 실행이 종료되지 않는 경우에 대한 효과를 아울러 고려해야한다.

참고문헌

[1] T-Y. Yeh, D. T. Marr, and Y.N. Patt, "Increasing the Instruction Fetch Rate via Multiple Branch Prediction and a Branch Address Cache," in *International Conference on Supercomputing '93*, 1993, pp. 67–76.

[2] P.K. Dubey, G.B. Adams III, and M. J. Flynn, "Instruction Window Size Trade-Offs and Characterization of Program Parallelism," *IEEE Transactions on Computers*, vol. 43, pp. 431–442, Apr. 1994.

[3] *Introduction to SHADOW*, Sun Microsystems. Inc., Jul. 1989.