패턴/패스 통합 분기 예측 전략의 성능 분석

Performance Analysis of Pattern/Path Hybrid Branch Prediction Strategy

조경산*, 주영상*
Kyungsan Cho, Youngsang Joo

Abstract

Recently studies have shown that conditional branches can be accurately predicted by recording the path leading up to the branch. But path predictors are more complex and uncompatible with existing pattern branch predictors.
In order to solve these problems, we propose a simple path branch predictor(SPBP) that hashes together two most recent branch instruction addresses. In addition, we propose a pattern/path hybrid branch predictor composed of the SPBP and existing pattern branch predictors. Through the trace-driven simulation of six benchmark programs, the performance improvement by the proposed pattern/path hybrid branch prediction is analysed and validated. The proposed predictor can improve the prediction accuracy from 94.21% to 95.03%.

* 단국대학교 전산통계학과

1. 서론

파이프라인 프로세서의 성능은 실행 흐름을 변경하는 실행 제어 명령어들에 의해 큰 영향을 받는다. 실행 제어 명령어에는 조건 분기 명령어, 무조건 분기 명령어, 간접 분기 명령어, 함수 호출 등이 있는데, 조건 코드 값에 의해 분기 방향이 결정되는 조건 분기 명령어가 파이프라인의 성능 저하에 가장 큰 원인이 된다[1].

파이프라인 프로세서에서 명령어가 명령어 페치(IF), 명령어 해석(DE), 오퍼랜드 페치(OF), 실행(EX)의 순서로 실행된다고 가정하면, 조건 분기 명령어는 실행 단계까지 분기될지 여부를 정할 수 없으므로 분기가 발생하는 경우에는 분기 지연(delay)이 발생한다. 그러나 분기 방향을 예측하여 분기 목적 명령어를 미리 페치 할 수 있다면, 분기 지연을 줄일 수 있다[12].

분기 여부를 실행 단계 이전에 결정하기 위해서는 분기 방향을 미리 예측하는 기법을 사용하는데, 이를 분기 예측 전략(Branch Prediction Strategy)이라 한다. 분기 예측 전략은 정적 예측(*static prediction*)과 동적 예측(*dynamic* prediction)의 두 가지 방법으로 구현된다. 정적 예측은 모든 분기 명령어에 대해 항상 분기됨(taken) 또는 분기되지 않음(not taken)으로 예측하며, 동적 예측은 분기 명령어의 과거 분기 실행 기록(history)을 사용하여 과거 분기 패턴에 따라 명령어 수행 시에 예측한다.

조건 분기 명령어의 분기 특성은 90%이상 분기되거나 분기되지 않는 강편향 분기(strongly biased branch)가 전체 분기 명령어의 60% 이상이며, 패턴 길이 1(과거 분기 기록이 1111 또는 0000과 같이, 항상 분기되거나 또는 분기되지 않는 경우)인 분기 명령어의 비율은 50% 정도로 알려져 있다[2][3]. 이러한 강편향 분기 명령어들은 정적 예측 전략을 적용하는 것이 예측 정확도가 높게된다.

약편향 분기(weakly biased branch)인 경우는 정적 예측 전략보다는 동적 분기 예측 전략이 보다 높은 정확도를 얻을 수 있다. 예를 들면, 분기 패턴이 101010인 경우 정적 예측으로는 50%만 정확히 예측 정확할 수 있다. 그러나 동적 예측을 사용할 경우 분기됨과 분기되지 않음을 번갈아 예측할 수 있으므로 정확한 예측이 가능하다.

따라서, 한가지 분기 예측 전략으로 강편향 분기 명령어와 약편향 분기 명령어들을 모두 정확히 예측하는 것은 불가능하므로, 여러 예측 방법들을 동시에 구현하고 각 분기 명령어의 특성에 따라 가장 적합한 예측 방법을 선택하여 예측 정확도를 높이는 통합 분기 예측(Hybrid Branch Prediction) 방법들이 개발되었다[4].

그러나 기존의 통합 분기 예측 방법들은 비용 면에서 비효율적인 구조이고, 다음의 명령어 열과 같이 과거 분기 패턴을 사용해서는 분기 여부를 정확히 예측 할 수 없는 경우가 있다.

분기 명령어 A: if(aa == 0) M
분기 명령어 B: if(aa == 2) M
분기 명령어 M: if( ) Y
분기 명령어 Y: if(aa > 0) Z

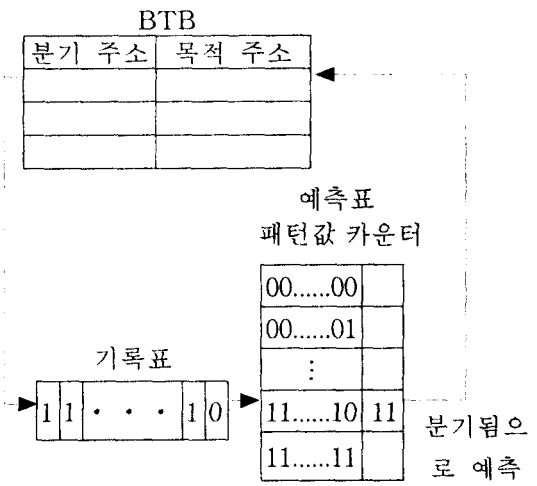
위의 예에서 경로 AMY인 경우에는 Y는 항상 분기되지 않고, BMY인 경우에는 항상 분기됨을 알 수 있다. 패턴은 모두 tt이므로 예측표의 동일한 항목으로 매핑되는 간섭이 발생하므로 정확한 예측을 할 수 없지만, 현재 분기 명령어에 이르기까지의 패스를 사용하면 AMY와 BMY의 두 개의 패스로 구분되므로 간섭이 발생하지 않는다. 따라서, 이 경우에는 패스 히스토리를 사용하는 것이 유리하다[5]. 그러나 패스 예측 방법은 실제 구현이 복잡하고 패턴 예측 방법과 결합하여 사용하기 어렵다는 문제점이 있다.

따라서, 본 연구에서는 패스 예측 방법의 문제점들을 보완한 단순한 패스 예측기(SPBP: Simple Path Branch Predictor)를 기존의 패턴 예측 구조와 결합한 패턴/패스 통합 예측 구조를 제안한다. 제안한 통합 구조는 패턴을 사용하여 정확히 예측이 어려운 명령어들은 SPBP를 사용하여 예측하고, 나머지 명령어들은 패턴 예측 구조를 사용하여 예측하는 운영 방법을 사용한다. 제안 구조가 기존의 통합 분기 예측 방법들에 비해 비용 면에서 효율적임을 trace driven 시뮬레이션을 통해 분석한다.

본 연구의 구성은 다음과 같다. 2장에서는 패턴

이나 패스를 사용하는 기존의 분기 예측 방법들을 분석하고, 3장에서는 본 연구에서 제안하는 기법의 검증 및 분석에 사용된 벤치마크 프로그램들의 특성과 시뮬레이션 및 파이프라인 구조 등을 소개하고, 4장에서는 패턴과 패스를 결합한 새로운 통합 분기 예측 전략을 제안하고 성능을 분석하고, 5장에서 결론을 내린다.

2. 관련 연구



<그림 1> 2-단계 예측기 구조

일반적인 동적 예측 방법은 기록표(history table)와 예측표(prediction table)로 구성된 2-단계 예측기(predictor)를 사용한다. 기록표는 분기 명령어들의 과거 분기 기록인 패턴을 저장하며, 예측표는 특정한 분기 기록을 갖는 분기 명령어의 과거 분기 패턴을 이용하여 분기를 예측한다. 2-단계 예측기의 구현 시에 기록표와 예측표가 각 분기 명령어에 대해 공유되는 경우를 전역 예측(Global Prediction)이라 하고, 각기 따로 유지하는 경우를 지역 예측(Local 또는 Private Prediction)이라 한다[6]. <그림 1>은 전역 예측기의 구조이다.

2-단계 예측기에서 기록표 또는 예측표가 분기 명령어들에 의해 공유되는 경우에, 2 개 이상의 분기 명령어가 하나의 패턴표로 매핑될 수 있으며, 이를 간섭(interference) 이라 한다. 간섭은 서로 다른 주 분기 방향 (majority direction)을 갖는 두 개의 분기 명령어가 동일한 패턴표의 항목으로 매핑되는 경우에 예측이 실패하는 주요 요인이 된다[2].

따라서, 간섭에 의한 예측 실패를 줄이기 위해 여러 예측 방법들을 동시에 구현하고 각 분기 명령어의 특성에 따라 가장 적합한 예측 방법을 선택하여 예측 정확도를 높이는 통합 분기 예측(Hybrid Branch Prediction) 방법들이 개발되었다[4][7].

이의 대표적인 예로, Chang은 filtering이라는 정적 예측 전략을 사용하고, 동적 예측을 위해 gshare의 전역 예측 및 pshare의 지역 예측을 사용하는 통합 분기 예측 구조를 제안하였다[3]. gshare는 하나의 기록 레지스터(history register)와 예측표를 모든 분기 명령어들이 공유하는 전역 예측 방법의 하나로 기록 레지스터와 분기 명령어의 하위 주소를 XOR한 값을 인덱스로 예측표에 매핑된다. pshare는 분기 명령어마다 개별적인 기록 레지스터를 갖는다는 점만 gshare와 다르다.

이 구조에서는 filtering을 위한 정적 예측 필드를 4 비트 크기의 포화 카운터와 1 비트 크기의 방향 비트(Direction bit)로 구성하였으며, 카운터 값이 1111이 되는 경우에 방향 비트 값이 1이면 분기됨으로, 0이면 분기되지 않음으로 예측한다. 그 외의 분기 명령어는 BPST(Branch Predictor Selection Table)라는 선택기(selector)를 통해 gshare 또는 pshare 중 하나로 예측한다. 그러나 이 방법은 전체 분기 명령어 중에서 10% 정도인 패턴길이 2-4인 분기 명령어[3]를 위해 지역 예측 방법을 사용하는데, 지역 예측 방법은 각 분기 명령어마다 과거 분기 기록을 별도로 유지해야하고 각 분기 명령어마다 가장 적합한 예측 방법을 결정하기 위해 선택기를 사용해야 하므로 고비용이 요구된다.

따라서, 본 연구팀은 Chang의 구조에서 지역 예측 방법인 pshare를 제외한 구조를 제안하였다[13]. 제안된 구조에서 gshare만을 사용하여 예측하는 방법과 비교해 gshare+filtering은 총 용량이 63% 증가할 때 예측 정확도가 24.42% 개선되었

고, gshare+filtering+pshare는 총 용량이 206% 증가할 때 예측 정확도가 25.45% 개선되었다[13]. 즉, gshare와 filtering만을 사용한 구조가 Chang의 구조보다 비용 면에서 효율적임을 알 수 있다.

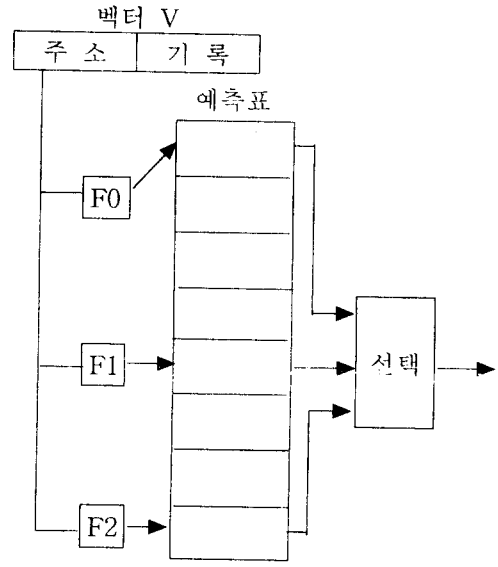
Michaud등은 간섭을 줄이는 방법으로 홀수개의 뱅크로 구성된 예측 표를 사용하는 다중-뱅크 스큐드 분기 예측기(multi bank Skewed Branch Predictor)를 제안하였다[8].

스큐드 분기 예측기는 각 뱅크마다 각기 다른 해시 함수를 사용하여 다수결에 따라 예측 방향을 결정하는 구조를 말한다. 다중 뱅크 스큐드 분기 예측기는 기존의 예측기에 비해 뱅크의 수에 비례하여 비용이 증가하고, gshare등과는 다른 해시 함수를 사용하기 때문에 기존의 예측기에 바로 적용하는데 어려움이 있다[14].

그래서 본 연구팀은 기존의 스큐드 분기 예측기에 비해 비용의 증가가 적은 다중 포트 스큐드 분기 예측기(multi port Skewed Branch Predictor)를 이전에 제안하였다[14]. 다중 포트 스큐드 분기 예측기는 기존의 스큐드 분기 예측기와는 달리 단지 홀수개의 포트를 포함한 하나의 예측표로 구성된다. 다중 포트를 위한 비용을 고려하지 않으면 전역 예측기와 동일한 구조를 가지므로 gshare와 예측 비용은 동일하다. 제안 구조는 각 포트마다 각기 다른 해시 함수를 사용하여 다수결에 따라 <그림 2> 같이 예측 방향을 결정한다.

<그림 2>의 3 포트 스큐드 분기 예측기에서 사용하는 해시 함수들은 하나의 히스토리 레지스터와 예측표를 모든 분기 명령어들이 공유하고, 단지 예측표로 인덱스되는 방법만 각기 틀린 전역 예측 방법들이다. 이 연구에 의하면 3-포트 구조가 3-뱅크 구조에 비해 예측 정확도가 높게 분석되므로 다중 포트 스큐드 분기 예측기 구조가 기존의 스큐드 분기 예측기 보다 비용 면에서 효율적임을 알 수 있다[14].

Yale N. Patt등은 분기 명령어의 목적 명령어로 패스 히스토리를 구성하였으며, 이 패스의 길이를 동적으로 변화하는 가변 길이 패스 분기 예측(Variable Length Path Branch Prediction) 구조를 제안하였다[9].



<그림 2> 3 포트 스큐드 분기 예측기 구조

기본 구조는 목적 명령어 주소의 하위 k 비트씩을 저장하는 Target History Buffer(THB)와 2비트 포화 카운터들로 구성된 예측표를 갖는다. 해시 함수로는 THB의 목적 명령어 주소들을 모두 XOR하는 방법을 사용하며, 이 값을 사용하여 예측표의 한 항목으로 매핑 된다. 이때, 예측표의 카운터 값이 2 이상이면 분기로 2 미만이면 분기되지 않음으로 예측한다. 여기에 각 분기 명령어마다 적절한 패스의 길이를 profiled한 방법을 통해 구하여 그 길이를 Hash Function Number Table(HFNT)에 저장한다.

따라서 실제로 예측되는 과정은 다음과 같다. 분기 명령어가 페치되면 HFNT를 통해 적절한 패스 길이를 구한 후 THB의 목적 명령의 주소들을 XOR 하여 예측표를 통해 예측한다.

패스를 사용한 예측 전략은 패턴으로는 정확히 예측 할 수 없는 분기 명령어들을 예측하는데 유리하다. 그러나 여러 개의 분기 명령어 주소 또는 목적 명령어 주소를 저장하고 이들을 모두 XOR 해야 하므로 패턴 예측 전략에 비해 구조가 복잡하고, 추가적인 비용을 요구한다. 또한 기존의 패턴을 사용한 예측 전략과 결합하여 사용하기 어렵다는

문제점이 있다.

본 연구에서는 패스 예측 방법의 문제점을 개선한 패스 예측 구조를 2 장에서 분석한 패턴을 사용하는 통합 분기 예측 방법들과 결합하여 분기 예측 정확도를 높일 수 있는 방법을 4 장에서 제안한다.

3. 작업부하 및 trace-driven 시뮬레이션

본 연구에서는 제안된 운영 구조에 의한 예측 정확도를 분석하기 위해 trace-driven 시뮬레이션을 수행한다.

시뮬레이션에 사용된 분기 명령어 실행에 대한 입력 자료는 SUN사에서 제공하는 shade를 사용하여 trace 하였다. shade는 SPARC v8과 SPARC v9에서 수행되는 분기 명령어에 관한 정보를 trace 하고 시뮬레이션 할 수 있는 C library를 제공한다 [10][11]. shade에서 제공되는 추적 정보를 기록하기 위한 구조체 Trace의 주요 멤버 변수와 라이브러리 함수는 다음과 같다.

Trace 멤버 변수
tr_pc : 명령어의 주소
tr_i : 명령어 텍스트
tr_annulled : 중단된 시점의 명령어를 추적
tr_taken : 분기 명령어의 분기 여부
tr_ih : 명령어 종류를 구분하는 정수 값
tr_ea : 유효 주소

Sade/Sipx 함수
shade_main() : 추적 프로그램의 시작함수
shade_trctl_trsize() : Trace 구조체의 크기 정보를 shade에 알려준다.
shade_trictl_ih() : 각 명령어에 대해 수집해야 할 항목 설정
shade_step() : 명령어 단위로 추적을 수행하는 함수
shade_shell() : 사용자 명령을 해석하여 수행하는 함수
is_ihuncond() : 조건 분기 명령 구분 함수
is_ihload() : 적재 명령 구분 함수

본 연구에서는 shade의 여러 함수들을 사용하여 분기 명령어 주소, 분기 목적 주소, 분기 방향 등을 추출하도록 trace 프로그램을 작성하였다. 다음은 trace 프로그램의 초기화 루틴의 일부분이다.

```
void      initialize(argc, argv, envp)
{
   if (argc != 1) usage ("");
   shade_trctl_trsize(sizeof(Trace));
   shade_trctl_it((uint32) IT_ANY, 1, 1,
   (uint32)  (TC_ANNULLED  |  TC_PC  |
TC_EA | TC_IH | TC_TAKEN));
}
```

작업부하는 shade를 활용한 trace 프로그램과 benchmark 프로그램들을 다음의 실행 환경을 통해 각 응용 프로그램에 대해 수행 조건 분기 명령어

<표 1> 사용된 벤치마크 프로그램

벤치마크 프로그램	프로그램 특성	작업 정도	명령어 수	조건 분기 명령어 비율
espresso	PLA 논리 최적화	mlp4	85,539,946	15.3%
gs	포스트 스크립트 페이지-묘사 언어에 대한 해석 수행	large	231,865,158	14.0%
make	GNU make 프로그램	perl	79,996,383	16.4%
p2c	파스칼을 C로 변환	mf	406,976,984	20.0%
ls	디렉토리 나열 프로그램	ls-lr	80,865,200	17.76%
compress	자료 압축 프로그램	compress	583,945,522	15.23%

중에서 1000 만개씩을 수집하였다.

커널 구조: Sun4m
응용 구조: Sparc
커널 버전: SunOS 5.5.1

작업부하의 생성을 위해 사용된 벤치마크 프로그램은 espresso, gs, make, p2c, ls, compress를 사용하였다. 각 벤치마크의 프로그램 특성과 수행 특성은 <표 1>과 같다.

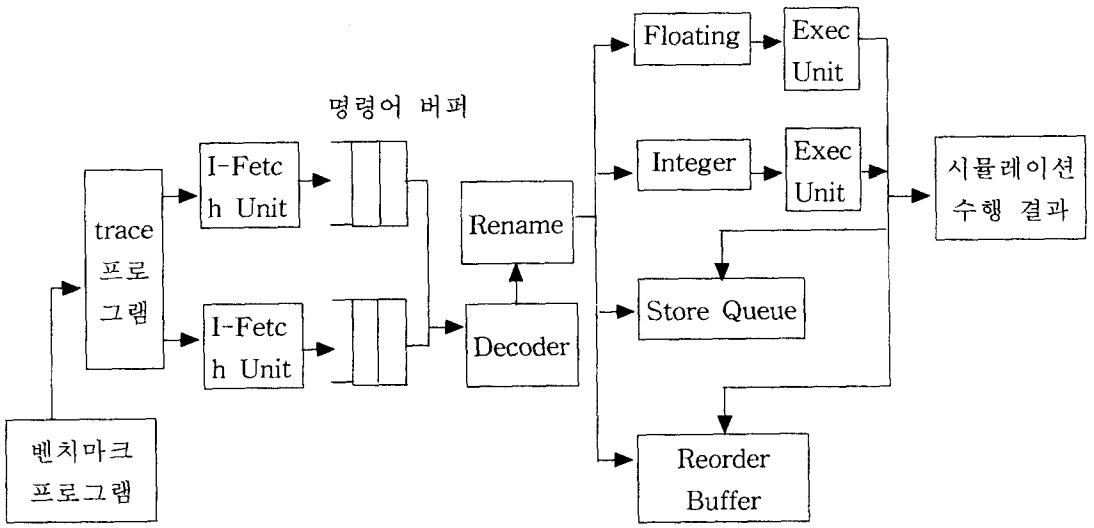
본 연구에서 제안하는 예측 방법의 예측 정확도를 분석하기 위해 <그림 3>과 같은 구조에서 시뮬레이션을 수행하였다.

trace 프로그램을 통해 벤치마크 프로그램의 분기 명령어 주소, 목적 명령어 주소들이 파이프라인으로 패치되면 제안하는 예측 방법을 통해 분기 방향을 예측한다. 예측 결과는 미리 trace된 분기 결과와 비교하여 예측 성공 빈도를 누적하고, 성공된 수를 수행된 전체 분기 명령어 수로 나누어 예측 정확도를 산출한다.

4. 패턴/패스 통합 분기 예측 구조와 시뮬레이션 결과 분석

2장에서 분석된 바와 같이 패스를 사용한 예측 전략은 패턴으로는 정확히 예측 할 수 없는 분기 명령어들을 예측하는데 유리하다. 그러나 패턴에 비해 구조가 복잡하고, 추가적인 비용이 요구되며, 또한 기존의 예측 전략과 결합하여 사용하기 어렵다는 문제점이 있다. 따라서 본 장에서는 현재 분기 명령어와 이전의 분기 명령어만을 패스로 사용하는 단순한 패스 예측 구조(SPBP)를 제안하고, 이 구조가 기존의 패턴을 사용한 예측 전략과 결합되어 예측 정확도를 높일 수 있음을 제시한다.

기존의 패턴을 사용한 예측 전략으로는 정확히 예측 할 수 없는 분기 명령어들을 분류하기 위해, gshare와 GAg가 동일하게 예측하는 경우와 서로 틀리게 예측하는 경우의 예측 정확도를 분석하면 다음과 같다.



<그림 3> 제안 구조 분석을 위한 시뮬레이션 수행 대상 구조

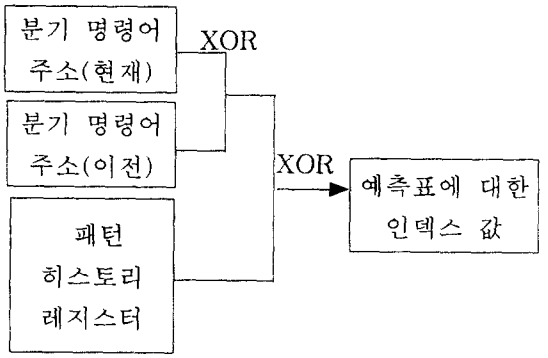
<표 2> gshare와 GAg가 동일하게 예측 되는 예측 정확도(%)

벤치마크	동일하게 예측	gshare_only	path_only
espresso	95.415	41.656	68.140
gs	96.517	55.406	84.276
make	98.723	85.888	91.771
p2c	96.883	44.566	71.077
ls	97.049	41.068	80.360
compress	91.148	51.593	58.444
평균	95.956	53.363	75.678

<표 2>를 분석해보면 gshare와 GAg가 동일하게 예측하는 분기 명령어들의 예측 정확도는 평균 95.956%로 높은 값을 보임을 알 수 있다. 그러나 서로 틀리게 예측하는 분기 명령어들에 대해 gshare로 예측하면, 예측 정확도가 평균 53.363%로 상대적으로 낮은 값을 보임을 알 수 있다. 따라서, 본 연구에서는 이런 명령어들의 예측 정확도를 분석하기 위해 패턴 히스토리가 아닌 패스 히스토리를 사용한 예측 기법을 제안한다.

패스 히스토리는 경로 ABY와 CDY 같이 현재 분기 명령어에 이르는 경로가 다르고, 그 경로에 따라 분기 여부가 서로 틀린 경우에 유용하다. 즉, 경로가 다른 경우에는 이전 분기 명령어가 경로에 따라 서로 다를 수 있다. 만일 이전 분기 명령어의 주소를 저장하여 현재 분기 명령의 주소와 XOR한 값을 주소 값으로 사용한다면, 서로 다른 경로로 현재 분기 명령어에 이르는 경우를 구분할 수 있다. 따라서 <그림 4>과 같은 간단한 방법으로 패스를 사용한 SPBP를 구현할 수 있다.

<표 2>의 path_only 항목은 gshare와 GAg가 서로 틀리게 예측하는 분기 명령어들을 <그림 4>에서 보인 인덱스 값을 사용하는 패스 예측 구조를 사용한 예측 정확도를 보인 것이다. gshare만으로 예측한 예측 정확도가 평균 53.363%인데 반해, 패스를 사용한 예측 정확도는 평균 75.678%로 예측 정확도가 높음을 알 수 있다. 본 연구에서 제안한 SPBP는 기존의 패턴 히스토리 예측 구조에 이전 분기 명령어의 주소를 저장하는 레지스터 하나를 추가하여 간단히 구현할 수 있고, 기존의 패턴 히스토리 예측 구조들과 결합하여 사용할 수 있다. 또, 패턴으로는 예측이 어려운 분기 명령어들을 비교적 높은 정확도로 예측 할 수 있다.



<그림 4> SPBP 구조의 인덱싱 방법

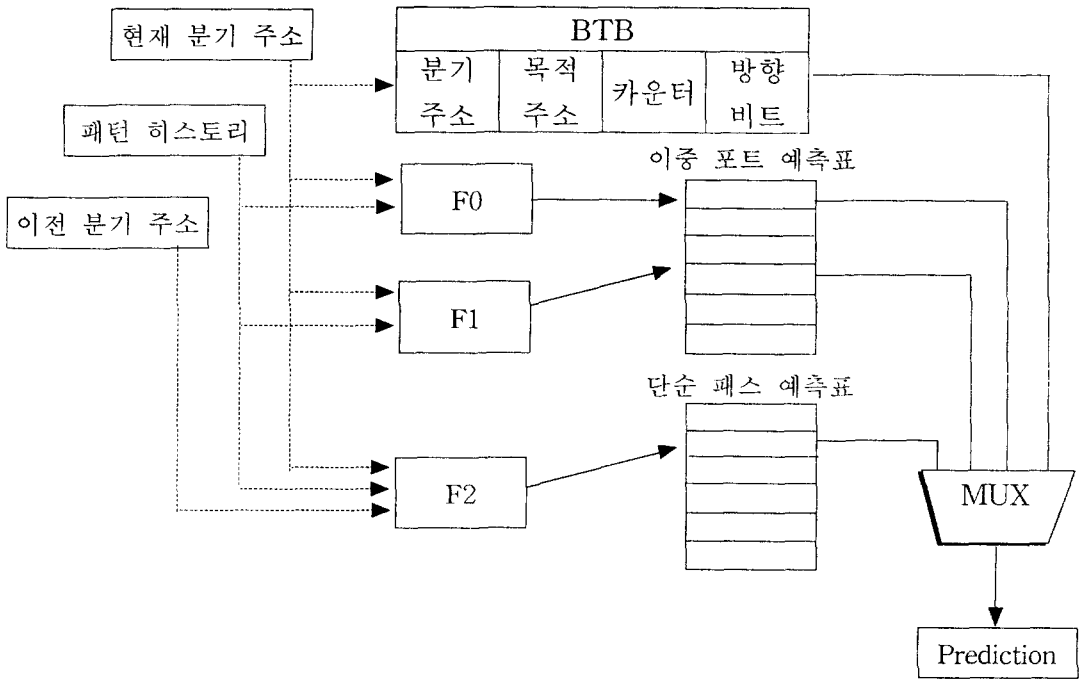
따라서 본 장에서는 패스의 장점을 살린 SPBP 예측 구조를 2장에서 분석된 filtering과 gshare를 사용한 통합 분기 예측 구조와 다중 포트 스큐드 분기 예측기와 결합하여 예측 정확도를 높이는 패턴/패스 통합 예측 구조를 제안하고 성능 개선 정도를 분석한다.

패턴/패스 통합 예측 구조는 filtering과 이중 포트(dual port)된 패턴 예측표를 공유하는 gshare와 GAg 그리고 SPBP로 구성된다. SPBP는 gshare등과는 달리 패스 히스토리를 사용하므로 gshare등과 예측 표를 공유해서는 기대하는 성능을 얻을 수 없으므로 독립된 패스 예측표를 사용한다.

<그림 5>는 패턴/패스 통합 분기 예측 구조를 나타낸 것이다.

<그림 5>의 구조에서 예측에 필요한 각 필드의 구성과 비용은 다음과 같다.

filtering을 위한 4 bit 카운터와 1 비트 크기의 방향 비트는 총 1K 개의 항목을 갖는 4-way 셋 연관 매핑 BTB(Branch Target Buffer)에 포함된다. 패턴 예측을 위해서는 12 비트 전역 기록 표와



<그림 5> 패턴/패스 통합 분기 예측 구조

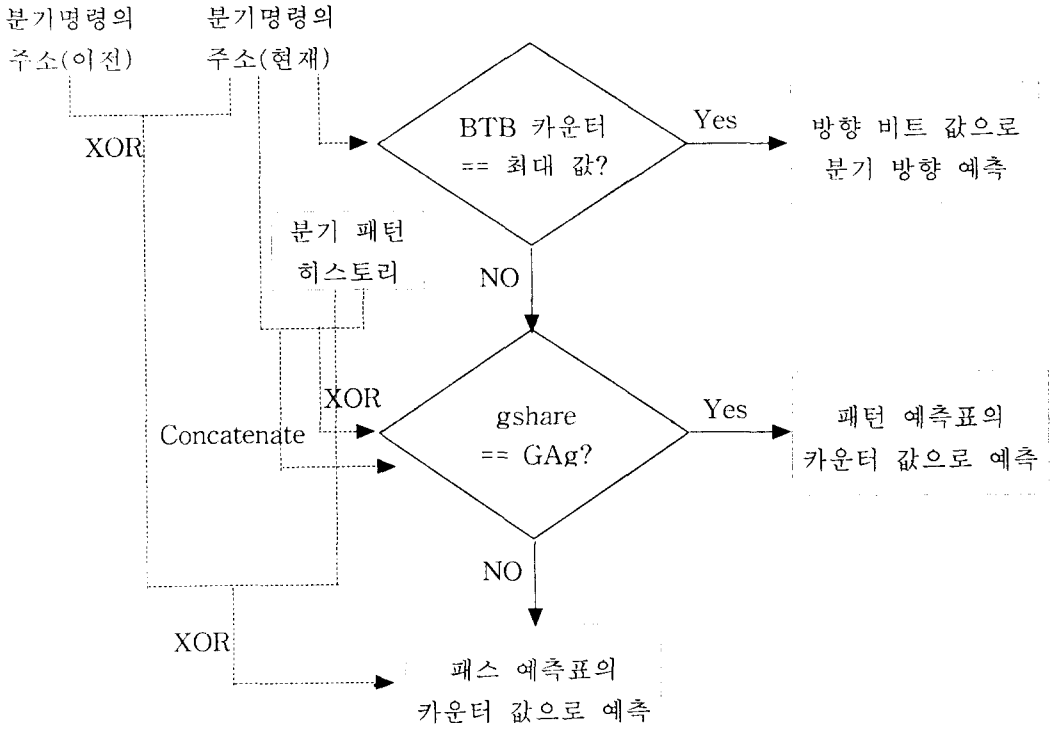
2 비트 카운터로 구성된 이중 포트 예측표를 사용하고, 패스 예측을 위해 이전 분기 명령어의 주소를 저장하는 레지스터, 2 비트 카운터로 구성된 패스 예측표를 별도로 유지한다. filtering과 패턴 예측과 패스 예측에 사용되는 비용은 각각 5K 비트, 8K 비트, 8K 비트이므로 예측에 필요한 총 비용은 대략 5K+8K+8K = 21K 비트가 된다.

다음은 <그림 5>에서 사용되는 해시 함수들을 사용 예를 포함하여 보인 것이다.

F0 : gshare(XOR)
분기주소 = 0xff4576f4, history = 0xff8
--->Index = 011011110100 ⊕ 111111111000
= 011000001100
F1 : 변형 GAg
분기주소 = 0xff4576f4, history = 0xff8
---> Index = 011011 concatenate 111000
= 011011111000
F2 : SPBP
분기주소 = 0xff4576f4, history = 0xff8,
이전 분기 주소 = 0xff4578f4
---> Index = Index(F0)⊕100011110100
= 111011111000

패턴/패스 통합 분기 예측 구조의 운영 방법은 <그림 6>과 같다. BTB의 카운터 값이 최대 값이면, BTB의 방향 비트 값으로 분기 방향을 예측하고, 카운터 값이 최대 값이 아니면 예측표를 사용하여 동적으로 예측한다. 동적 예측 방법은 gshare와 변형 GAg에서 예측한 값이 동일하면 그 값으로 예측하고, 다르면 이전 분기 명령어와 현재 분기 명령어 주소 그리고 패턴 히스토리를 XOR한 값을 인덱스로하여 패스 예측표의 카운터 값에 따라 예측한다.

제안 구조는 정적 예측과 3 가지 동적 예측기가 결합된 구조이므로 예측에 사용되는 각 필드들을 갱신하는 방법에 따라 성능이 달라질 수 있다. 본 절에서는 정적 예측을 위한 필드들은 항상 갱신



<그림 6> 패턴/패스 통합 분기 예측 구조의 운영 방법

하고, 동적 예측에 사용되는 예측표들은 부분적으로 갱신하는 방법을 사용한다. 즉, gshare와 GAg가 공유하는 예측표의 카운터는 동적으로 예측되는 경우만 갱신한다. SPBP에서 사용되는 예측표의 카운터는 gshare와 GAg가 다르게 예측하는 경우만 갱신된다.

<표 3>은 제안 구조에서 filtering을 제외한 나머지 예측기의 예측 정확도를 보인 것이다. 표의 첫 번째 항목은 gshare와 GAg가 동일하게 예측하는 경우의 예측 정확도이고, 두 번째와 세 번째 항목은 gshare와 GAg가 다르게 예측하는 경우 gshare로 예측한 것과 SP 예측 구조로 예측한 것의 예측 정확도를 보인 것이다. gshare로 예측한 것에 비해 SP 예측 구조는 예측 정확도가 최소 3.65%에서 최대 21.89% 평균적으로 14.18% 높은 것을 알 수 있다.

<표 3> 제안 구조를 구성하는 각 예측기의 예측 정확도

벤치마크	gshare ==GAg	gshare_only	path_only
espresso	87.958	49.873	71.763
gs	93.524	68.371	85.953
make	96.639	87.455	91.100
p2c	90.610	52.542	72.448
ls	90.878	63.048	74.068
compress	80.734	54.604	65.647
평균	90.05	62.64	76.82

<표 4>는 패턴/패스 통합 분기 예측 구조의 예측 정확도와 3 뱅크 스큐드 분기 예측기의 예측 정확도를 분석한 것이다. 제안 구조(21K 비트)의 예측 정확도는 평균 95.03%로, 비슷한 비용(24K 비

트)으로 구현된 3 뱅크 스큐드 예측기 비해 최소 0.3%에서 최대 1.1%로 평균적으로 0.82% 예측 정확도가 개선된다.

<표 4> 패턴/패스 통합 분기 예측 구조와 3 뱅크 스큐드 분기 예측기의 예측 정확도

벤치마크	패턴/패스 예측 구조	3 뱅크 스큐드 분기 예측기
espresso	94.122	93.490
gs	96.154	95.041
make	98.328	97.975
p2c	95.754	94.645
ls	96.507	96.107
compress	89.355	88.028
평균	95.03	94.21

<표 3>과 <표 4>의 분석으로부터, 제안 구조는 패턴을 사용하여 예측하기 힘든 명령어들을 보다 정확히 예측하여 전체적인 예측 정확도를 개선할 수 있고, 3 뱅크 스큐드 분기 예측기에 비해 적은 비용으로 예측 정확도를 개선하는 비용 면에서 효율적인 구조임을 알 수 있다. 또, <표 2>에서 gshare와 GAg가 동일하게 예측하는 분기 명령어들의 예측 정확도는 95.956%이고, 이 값은 비교적 정확히 예측할 수 있는 분기 명령어들만의 예측 정확도이므로 제안한 패턴/패스 통합 분기 예측 구조가 이 값에 가까운 예측 정확도를 보이는 예측 구조임을 알 수 있다.

5. 결 론

분기 예측 전략을 사용하는 최근의 연구들에서는 독립적인 한가지 예측 방법만 사용하는 것이 아니라 여러 가지 예측 방법들을 결합한 통합 분기 예측 방법들이 연구되고 있다. 그러나 패턴을 사용하는 기존의 통합 분기 예측 방법들로는 정확한 예측이 어려운 분기 명령어들이 존재한다. 따라서, 패턴이 아니라 현재 분기 명령어에 이르기까지의 분기 명령어나 목적 명령어들을 기록한 패스를 사용하는 예측 전략이 연구되고 있다. 그러나 기존의 패스 예측 전략은 패턴 예측 전략에 비해 복잡한 구조를 요구하고 기존의 패턴 예측 방법과 결합하여 사용하기 어렵다는 문제점을 갖고 있다.

따라서, 본 연구에서는 기존의 패스 예측 전략의 문제점들을 보완한 단순한 패스 예측 구조(SPBP)를 제안하고, 이 구조를 패턴을 사용한 통합 분기 예측 방법들과 결합한 패턴/패스 통합 분기 예측 구조를 제안하였다.

패턴/패스 통합 분기 예측 구조는 BTB의 카운터 값이 최대 값이면, BTB의 방향 비트 값으로 분기 방향을 정적으로 예측하고, 카운터 값이 최대 값이 아니면 동적으로 예측한다. 동적 예측 방법은 gshare와 변형 GAg에서 예측한 값이 동일하면 그 값으로 예측하고, 다르면 패스 예측표의 카운터 값에 따라 예측하는 구조이다.

이 구조를 기존의 3 뱅크 스큐드 분기 예측기의 예측 정확도와 비교하면, 제안 구조의 예측 정확도는 3 뱅크된 스큐드 예측기 비해 평균적으로 94.21%에서 95.03% 예측 정확도가 개선되었다.

따라서, 패턴을 사용하여 정확히 예측이 어려운 명령어들은 패스를 사용하여 예측하고 나머지 명령어들은 패턴을 사용하여 예측하는 패턴/패스 통합 예측 구조는 기존의 통합 분기 예측 방법들의 고비용 문제와 패스 예측 방법의 문제점들을 보완할 수 있는 구조이다.

참고 문헌

[1] Brad Calder and Dirk Grunward, "Fast & Accurate Instruction Fetch and Branch Prediction," Proc. 21th Ann. Symp. Computer Architecture, pp. 2-11, 1994.

[2] Cliff Young et. al, "A Comparative Analysis of Schemes for Correlated Branch Prediction," Proc. 22th Ann. Symp. Computer Architecture, pp. 276-286, 1995.

[3] Po-Young Chang, "Classification-Directed Branch Predictor Design," The University of Michigan, available at http://www.cs.umich.edu., 1997.

[4] S. MCFarling, "Combining branch predictors," Technical Report TN-36, Digital Western Research Laboratory, June 1993.

[5] C. Young, N. Gloy, and M. D. Smith, " A comparative analysis of schemes for correlated branch prediction," Proc. 22nd Ann. Symp. Computer Architecture, pp. 276-286, 1995.

[6] Tse-Yu Yeh and Tale N Patt, "A Comparison of Dynamic Branch Predictors that use Two Levels of Branch History," Proc. 20th Ann. Symp. Computer Architecture, pp. 257-266, 1993.

[7] Eric Sprangle et. al, "The Agree Predictor: A Mechanism for Reducing Negative Branch History Interference," Proc. 24th Ann. Symp. Computer Architecture, pp. 284-291, 1997.

[8] Pierre Michaud et. al, "Trading Conflict and Capacity Aliasing in Conditional Branch Predictors," Proc. 24th Ann. Symp. Computer Architecture, pp. 292-303, 1997.

[9] Yale N. Patt et. al, "Variable Length Path Branch Prediction", in Proc. 8th Ann. Conf. Architectural Support for Programming Languages and Operating Systems, pp. 170-179, 1998.

[10] Bob Cmelik and David Keppel, "Shade: A Fast Instruction-Set Simulator for Execution Profiling," Sigmetrics, ACM, pp. 138 -137, 1994.

[11] Sun microsystem, "shade user's manual," available at http : //sw.sun.com/shade.

[12] 주 영상, 조 경산, "분기 예측과 이중 경로 전략을 결합한 파이프라인 구조에 관한 연구," 정보 처리 논문지, 제3권 제1호, pp. 181 - 190, 1996.

[13] 주 영상, 조 경산, "Victim BTB를 활용한 히트율 개선과 효율적인 통합 분기 예측," 정보 처리 논문지, 제5권 제10호, pp. 2676 - 2685 , 1998.

[14] 주 영상, 조 경산, "다중 포트된 스큐드 분기 예측기의 성능 분석" 정보 처리 학회 추계 학술 발표 논문집 제5권 제2호, pp. 1215 - 1218, 1998.

저자소개

조 경 산

1979년 서울대학교 전자공학과 졸업 (학사)
1981년 한국과학원 전기 및 전자공학과 졸업(공학석사)
1988년 텍사스대학교(오스틴소재) 전기전산공학과 졸업(Ph.D.)
1988년~1990년 삼성전자 컴퓨터 부문 책임 연구원
1990년~현재 단국대학교 전산통계학과 교수
관심분야 : 구조론 및 성능평가, 컴퓨터 통신, 시뮬레이션

주 영 상

1990년 고려대학교 재료공학과 졸업 (학사)
1995년 단국대학교 전산통계학과 졸업 (이학석사)
1995년~현재 단국대학교 전산통계학과 박사과정
관심분야 : 구조론 및 성능평가