# 동적 분류를 이용한 하이브리드 결과 값 예측기

## (Hybrid Value Predictor using Dynamic Classification)

신 영 호 † 윤 성 룡 † 조 영 일 ††
(Young-Ho Shin)(Sung-Lyong Yoon)(Young-Il Cho)

**요 약** 슈퍼스칼라 프로세서의 성능을 향상시키기 위해서는 데이타 종속성에 의한 장애를 제거해야 한다. 최근 여러 논문들은 이러한 데이타 종속성을 제거하기 위해서 명령어의 결과 값을 예상하는 메커니즘을 제안하였다.

이러한 예상 메커니즘 중 여러 예측기를 혼합해서 사용하는 하이브리드 방법은 각각 하나의 예측기만을 사용하는 방법보다 더 좋은 성능을 얻을 수 있다. 그러나 그러한 하이브리드 예측기는 명령어를 중복해서 저장하여 많은 하드웨어 크기를 요구한다.

본 논문에서는 여러 예측기의 장점을 이용하여 높은 성능을 얻을 수 있는 새로운 하이브리드 예측 메커니즘을 제안한다. 또한 예상이 자주 틀리는 명령어를 동적으로 찾아내어 예상하지 않음으로서 잘못 예상 시 발생하는 misprediction 페널티를 낮추고 예상 정확도를 높인다. 시뮬레이션 결과 SPECint95 벤치마크 프로그램에 대해 제안한 하이브리드 예측기에서 예측율은 평균 79%에서 90%로 향상하였고, misprediction rate는 평균 12%에서 2%로 낮추었다.

***Abstract*** To improve the performance of Superscalar microprocessors, the serialization constraints imposed by true data dependences must be removed. Several related works have suggested that the limits imposed by data dependences can be overcome to some extent with the use of data value prediction.

Hybrid value predictors can obtain high prediction accuracy using advantages of various value predictors, but they have a defect that same instruction has overlapping entries in all predictors.

In this paper, we proposes a new hybrid value predictor which combines three prediction mechanisms, such as last value predictor, stride value predictor and two-level value predictor, into a hybrid value predictor. We then propose a dynamic classification mechanism which selects the most adequate value predictor for an instruction. Simulation results show that we achieve the average prediction rate of 90% and the average mispredicton rate of 2% for the SPECint95 benchmark set.

## 1. 서 론

현재 마이크로프로세서 구조는 여러 개의 실행 유닛을 사용해서 다수의 명령어를 병렬로 처리하여 프로세서의 성능을 향상할 수 있게 설계되고 있다. 이러한 구조의 프로세서에서 높은 성능을 얻기 위해 높은 명령어 이슈율과 명령어 수준 병렬성(Instruction Level Parallelism: ILP)을 가능한 한 최대로 이용하는 것이 중요하다. ILP를 처리 할 때 주요 장애 요소로는 제어종속과 데이타 종속이 있다. 제어종속은 조건분기 명령이 반입될 때 분기 방향과 다음에 수행할 명령어 주소를 예상하는 분기 예상기법으로 문제를 해결할 수 있다. 데이타 종속에는 false-data 종속과 true-data 종속이 있다. false-data 종속에 의한 장애는 소프트웨어 및 하드웨어 재명명(rename) 방법으로 완전히 제거될 수 있다. 그러나 선행 명령어의 결과를 입력으로 사용하는 true-data 종속관계의 명령어는 병렬로 수행할 수 없다. 이러한

† 학생회원 : 수원대학교 컴퓨터과학과
yhsin@cs.suwon.ac.kr
slyoun@cs.suwon.ac.kr
†† 정 회 원 : 수원대학교 컴퓨터과학과 교수
ycho@mail.suwon.ac.kr
논문접수 : 2000년 3월 6일
심사완료 : 2000년 10월 13일

true-data 종속성을 제거하기 위해 결과 값(value) 예상 기법을 사용하게 된다. 결과 값 예상기법은 선행 명령어의 결과를 예상하고, 종속적인 명령어가 예상 값을 사용하여 빨리 수행함으로서 true-data 종속 연결관계를 제거하는 하드웨어 메커니즘이다[1,2,3,4,5,6,7,8,14,15].

결과 값 예상기법 중 명령어의 결과를 바로 이전에 수행된 결과로 예상하는 방법으로 수행 결과가 변하지 않는 경우에 정확히 예상할 수 있는 Last 결과 값 예측기[1,2]가 있다. 이 방법은 마지막으로 수행된 결과 값(Last value) 한 개만 예상 테이블에 저장함으로 적은 하드웨어를 사용하고, 구현이 쉽다는 장점이 있다. 또한 명령어의 결과가 마지막으로 수행된 결과 값에 일정한 차이 값(stride)만큼 변한다는 사실에 기인하여 해당 명령어의 다음 결과를 예상하는 Stride-based 결과 값 예측기[6,7]가 있다. 이 방법은 반복문의 카운터 변수나 결과 값이 일정하게 증감하는 명령어를 갖는 프로그램에서 좋은 성능을 발휘하게 된다. 또한 Two-level 결과 값 예측기[5]는 이전에 실행된 네 개의 결과 중에 하나를 다음 값으로 예상하는 방법으로 높은 예상 정확도를 가지나 네 개의 결과 값을 저장함으로 구현 시 많은 하드웨어 비용을 필요로 한다.

기존의 결과 값 예측기는 다양한 특성을 갖는 여러 명령어들을 모두 예상할 수 없기 때문에 이러한 단점을 극복하기 위해 여러 예측기를 혼합해서 사용하는 하이브리드 예측기[6,8]가 제안되었다.

하이브리드 예측기는 하나의 예측기만 사용하는 방법보다 높은 예측정확도를 갖는다. 그러나 제한된 예측기 테이블에서 동일한 명령어를 여러 예측기의 엔트리에 할당함으로서 예측 가능한 다른 명령어가 예측기의 엔트리에 할당되지 못하는 문제점을 갖고 있다.

이러한 문제점을 해결하기 위해 Shen[14]은 명령어들을 가장 잘 예상할 수 있는 예측기를 동적으로 분류하는 방법을 제안하였다. 그러나 Shen이 제안한 방법은 명령어를 동적으로 분류시키는 분류 테이블(Classified Instruction Table : CT)을 위한 추가적인 하드웨어 오버헤드가 요구되고, 동적으로 적합한 예측기를 찾아 분류시키는 학습시간이 필요하다.

본 논문에서는 앞서 기술한 Last 결과 값 예측기, Stride-based 결과 값 예측기, Two-level 결과 값 예측기의 장점을 혼합해서 사용하며, 추가적인 하드웨어 없이 명령어들의 수행 결과의 패턴을 조사하여 각 패턴에 적합한 결과 값 예측기를 동적으로 분류하는 하이브리드 결과 값 예측기를 제안한다. 제안된 방법은 기존의 하이브리드 예측기와 동일한 하드웨어 비용에서 높은 예상 정확도를 갖는다. 또한, 명령어가 이전 패턴과 관련된 예측기의 테이블만을 사용하게 하여 여러 예측기의 테이블 엔트리에 중복 사용을 피하게 함으로써 제한된 하드웨어 테이블에 보다 많은 명령어를 할당할 수 있다. 그리고 예상이 자주 틀리는 명령어들을 동적으로 찾아내어 그러한 명령어들은 예측기에 할당시키지 않음으로서 예측기를 보다 효율적으로 사용하여 기존의 하이브리드 예측기보다 더 높은 예상 정확도를 가질 수 있다.

## 2. 관련 연구

이 장에서는 기존의 대표적인 결과 값 예측기인 Last 결과 값 예측기, Stride-based 결과 값 예측기, Two-level 결과 값 예측기, 하이브리드 예측기, Shen의 하이브리드 예측기를 설명하고 그들의 장단점을 고찰한다.

### 2.1 Last 결과 값 예측기

Last 결과 값 예측기는 명령어가 최종적으로 수행된 결과 값을 저장해서 다음에 동일 명령어를 만났을 때, 바로 이전의 수행 시 저장된 결과 값을 예상 값으로 사용하는 방법이다. Last 결과 값 예측기는 VPT(Value Prediction Table)의 각 엔트리에 마지막 수행 결과 값 하나만 저장하기 때문에 적은 하드웨어 비용을 요구하지만 변하지 않는 값(상수 값)과 같은 단순한 패턴만 예상할 수 있어서 예상 정확도는 40% ~ 50%로 낮다는 단점을 갖는다. Last 결과 값 예측기는 3개의 필드로 구성되어 있다 -태그 필드(Tag), 한 개의 결과 값 필드(Last Value), 포화카운터 필드(Saturating Counter). [그림 1]은 Last 결과 값 예측기의 엔트리 구조를 보여주고 있다. 태그 필드는 유효한 엔트리를 표시하기 위해서 명령어 주소의 상위 비트 일부를 저장해 놓은 부분이고, Last 결과 값 필드는 마지막으로 수행된 명령어의 결과 값을 저장하는데 사용되며, 포화카운터 필드는 예상을 결정하기 위한 2비트 필드로 {'0','1','2','3'}중 하나의 값을 가지고 '2'와 '3'일 경우만 예상을 수행한다.

| Tag | Last Value | Saturating Count |
|---|---|---|
| 20 | 32 | 2 |

그림 1 Last 결과 값 예측기의 엔트리 구조

### 2.2 Stride-based 결과 값 예측기

Stride-based 결과 값 예측기[6]는 명령어의 마지막 수행된 결과 값과 마지막 두 번의 수행된 결과 값의 차

(stride)를 VPT에 저장하여 명령어의 반입(fetch)시 마지막 수행 결과 값과 stride의 합으로 예상한다.

| Tag | State | Last Value | Stride |
|---|---|---|---|
| 20 | 2 | 32 | 8 |

그림 2 Stride-based 결과 값 예측기의 엔트리 구조

이를 변형시킨 Two-delta stride 결과 값 예측기[9]는 해당 명령어의 stride의 변경을 효율적으로 하기 위해 두 개의 stride(predicted stride, new stride)를 VPT에 저장하여, new stride가 연속적으로 두 번 나타나면 predicted stride를 new stride로 치환하여 stride가 변화하는 상황에 보다 빨리 적응하도록 한 예상방법이다. 이 방법은 Last 결과 값 예측기보다 좋은 성능을 가진다. Stride-based 결과 값 예측기는 4개의 필드로 구성되어있다 -태그 필드(Tag), 상태 필드(State), Last 결과 값 필드(Last Value), 차이 값 필드(Stride). [그림 2]는 Stride-based 결과 값 예측기의 엔트리 구조를 보여주고 있다. 상태필드는 2비트로 되어있으며 3개의 상태(Init, Transient 그리고 Steady)를 가진다. 상태필드가 Init('0') 혹은 Transient('1')를 나타내는 경우는 예상을 위해 준비중인 단계를 나타내며 예상을 수행하지는 않는다. 상태필드의 값이 Steady('2')를 나타내는 경우에만 Last 결과 값 필드와 차이 값 필드의 합을 예상 값으로 제공한다.

### 2.3 Two-level 결과 값 예측기

Two-level 결과 값 예측기[5]는 다른 예측기와 같이 value-locality[1]를 기반으로 하고 있고 명령어가 수행한 마지막 4개의 결과 값을 VHT(Value History Table)에 저장하여 이를 바탕으로 결과 값을 예상하는 방법이다. Two-level 결과 값 예측기는 명령어의 주소(PC)로 VHT를 인덱스하고, 해당 엔트리에 있는 4개의 결과 값의 수행 패턴으로 PHT(Pattern History Table)를 인덱스하고 PHT 엔트리의 카운터에 의해 4개의 값 중 하나를 결과 값으로 예상한다. Two-level 결과 값 예측기는 높은 예상 정확도를 갖지만 4개의 결과 값을 저장하기 때문에 많은 하드웨어 비용을 요구한다는 단점을 갖는다.

Two-level 결과 값 예측기 중 VHT는 4개의 필드로 구성되어 있다 -태그 필드(Tag), LRU 필드(LRU_info), 네 개의 마지막 결과 값 필드(Four Values), VHP(Value History Pattern)필드(VHP). [그림 3]은 Two-level 결과 값 예측기의 엔트리 구조를 보여주고 있다. 결과 값 필드는 최근의 사용된 결과 값들 중에서 중복되지 않은 서로 다른 네 개의 결과 값을 저장한다. LRU 필드는 결과 값 필드에 저장된 값들의 사용된 순서에 대한 정보를 가지고 있다. 즉 결과 값 필드 중 어떤 필드의 값이 가장 오래 전에 사용된 것인지 어떤 필드의 값이 가장 최근에 사용된 것인지를 나타낸다. 만약 결과 값 필드의 네 개의 결과 값과 다른 새로운 결과 값이 들어왔을 경우, LRU 필드에서 가장 오래전에 사용된 결과 값으로 표시된 결과 값을 제거하고 새로운 결과 값을 그 자리에 저장한다. VHP 필드는 결과 값이 중복되는지에 상관없이 단순히 결과가 나타난 순서대로의 위치가 저장된다.

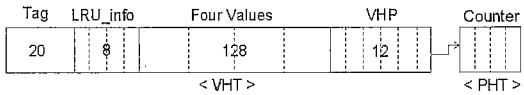


그림 3 Two-level 결과 값 예측기의 엔트리 구조

PHT에는 VHT의 결과 값 필드에서 어떤 값으로 예상을 수행할지 결정하기 위해서 4개의 카운터 필드를 사용한다. 이 카운터 필드 중에서 가장 큰 값으로 설정되어 있는 위치의 결과 값을 예상 값으로 사용하게 된다.

### 2.4 하이브리드 결과 값 예측기

명령어의 패턴은 여러 가지로 나타난다. 일정한 상수 값을 갖는 명령어, 일정한 값으로 증감하는 명령어, 일정한 값은 아니지만 규칙적인 값을 생성하는 명령어, 예상하기 어려운 불규칙적인 값을 생성하는 명령어가 있다. 하이브리드 결과 값 예측기[6]는 일정한 값으로 증감하는 명령어를 잘 예상하는 Stride-based 결과 값 예측기와 네 개의 결과 값을 저장하는 Two-level 결과 값 예측기를 결합한 것으로 신뢰성 카운터(Confidence Counter)에 의해 어느 한 예측기로부터 예상 값을 선택하는 방법이다. 명령어가 두 개의 예측기에서 모두 엔트리를 갖고 있다면, 높은 신뢰성 카운터 값을 갖는 예측기의 예상 값을 선택한다. 하이브리드 결과 값 예상방법은 stride 특성을 갖는 명령어와 규칙적인 패턴을 갖는 명령어를 모두 예상하기 때문에 높은 예상 정확도를 갖는다는 장점을 갖지만 명령어가 두 개의 예측기의 엔트리에 중복됨으로써 많은 하드웨어 비용을 요구한다는 단점을 갖는다. [그림 4]는 하이브리드 결과 값 예측기의 엔트리 구조를 보여주고 있다.

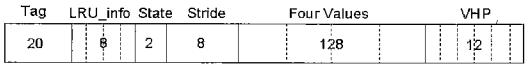


그림 4 하이브리드 결과 값 예측기의 엔트리 구조

### 2.5 Shen의 하이브리드 결과 값 예측기

Shen의 하이브리드 예측기[14]는 Last 결과 값 예측기, Stride-based 결과 값 예측기, FCM(Finite Context Method) 결과 값 예측기[15]를 혼합한 하이브리드 예측기다. Shen이 제안한 방법은 분류 테이블(Classified Instruction Table : CT)을 이용하여 실행 시간동안 명령어들의 패턴을 분류하여 가장 잘 예상할 수 있는 예측기에 할당하는 예측기다. 그러나 분류 방법에 있어서 분류 테이블을 따로 가짐으로써 하드웨어 오버헤드가 증가되고, 패턴을 분류하기 위해서는 분류 테이블에서 몇 번의 분석을 거쳐야 하는 학습시간이 필요한 단점이 있다. 본 논문에서 사용한 예측기는 Shen의 하이브리드 예측기에서 FCM 결과 값 예측기 대신 Two-level 결과 값 예측기를 사용한다.

위에서 보듯이 명령어의 결과 값을 예상하기 위한 여러 가지 예상방법이 제안되었으나 아직 만족할 만한 예상 정확도를 갖지 못하고 있다. 또한 예상이 틀릴 경우, 회복(recovery)을 위한 페널티가 예상을 수행하지 않고 순서적으로 수행했을 때보다 훨씬 크므로 예상이 자주 틀리는 명령어는 예상하지 않는 것이 성능 향상에 도움이 된다.

## 3. 제안된 하이브리드 결과 값 예상방법

이장에서는 본 논문에서 제안한 예측기의 구조와 명령어에 가장 적합한 예측기를 선택하기 위해 명령어들을 동적으로 분류하는 방법에 대해 알아본다.

### 3.1 제안된 하이브리드 결과 값 예측기 구조

본 논문에서는 이전에 제안된 하이브리드 결과 값 예상방법이 동일한 명령어에 대해 여러 예측기에 엔트리를 할당함으로서 많은 하드웨어 비용을 요구하는 단점을 해결하기 위해 명령어의 결과 값을 예상 할 때 어떤 예측기를 선택할 것인지를 동적으로 분류하고 가장 적합한 예측기에만 엔트리를 할당하여 중복을 허용하지 않음으로서 하드웨어 비용을 감소시키는 하이브리드 결과 값 예상방법을 제안한다. 또한 예상이 자주 틀리는 명령어들은 UIT(Unpredicted Instruction Table)에 저장하여 반입된 명령어가 UIT에 있을 경우 명령어의 결과 값을 예상하지 않고, 예측기의 엔트리도 할당하지 않음으로서 예측율과 예상정확도를 향상시키게 한다.

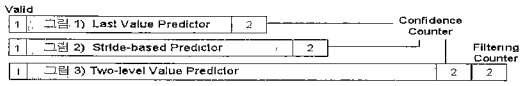


그림 5 제안한 하이브리드 결과 값 예측기의 엔트리구조

본 논문에서 제안한 하이브리드 예측기의 구조[그림 5]는 이전의 결과 값 예측기에 추가된 필드가 나타나고 있다. Valid 비트는 해당 엔트리에 명령어가 할당되어 있는지를 나타낸다. Valid 비트가 '1'로 설정된 예측기로 예상을 하고, 어느 예측기에도 할당되지 않은 상태를 나타낼 경우에는 명령어의 동적 분류를 위해 Two-level 결과 값 예측기에 할당한다. 또한, 신뢰성 카운터는 예상이 어려운 명령어들을 찾아내는데 사용한다. 여과 카운터(Filtering Counter)는 Two-level 결과 값 예측기에서 명령어를 동적 분류하기 위해 사용된다.

Last 결과 값 예상 방법을 사용하면 예상 하드웨어는 적게 들지만, 결과가 stride로 증감하는 명령어나 반복되는 패턴을 갖는 명령어는 예상할 수 없으므로 예상 정확도가 감소한다. Stride-based 결과 값 예상 방법을 사용하면, 예상 하드웨어가 Two-level 결과 값 예상방법보다는 적게 들지만, 결과가 반복되는 패턴을 갖는 명령어는 예상할 수 없으므로 예상 정확도가 감소한다. 또한 Two-level 결과 값 예상방법을 사용하면 zero-stride(last 결과 값)를 갖는 많은 명령어들과 stride로 변하는 명령어들에 대해 불필요한 필드들에 의한 하드웨어 낭비가 발생한다. 또한 예상이 자주 틀리는 명령어는 잘못 예상하여 성능을 감소시키는 원인이 되며 불필요하게 예측기의 엔트리를 차지하기 때문에 예상 정확도가 높은 명령어들이 예측기의 엔트리에 할당될 기회를 감소시켜 예상율과 예상정확도를 감소시킨다.

본 논문에서는 Last 결과 값 예상 방법, Stride-based 결과 값 예상 방법, Two-level 결과 값 예상방법의 장단점에 착안하여, 세 개의 예상 방법을 결합시킨 하이브리드 결과 값 예상 방법을 제안한다.

프로그램에서 결과 값을 예상할 수 있는 명령어의 유형은 3개의 서브세트로 분할된다. 첫째는 last 결과 값을 사용하는 명령어 서브세트가 전체 명령어 중 60%를 차지하고, 둘째는 stride를 이용하여 예상하는 명령어 서브세트가 10%, 셋째는 이전 수행된 4개의 결과 중 하나로 예상되는 명령어 서브세트가 30%를 차지한다[4]. 따라서, 상대적으로 많은 명령어를 예상할 수 있도록 Last 결과 값 예측기에는 가장 많은 엔트리를 할당하고 Stride-based 결과 값 예측기에는 적은 엔트리를, Two-level 결과 값 예측기에는 중간 크기의 엔트리를

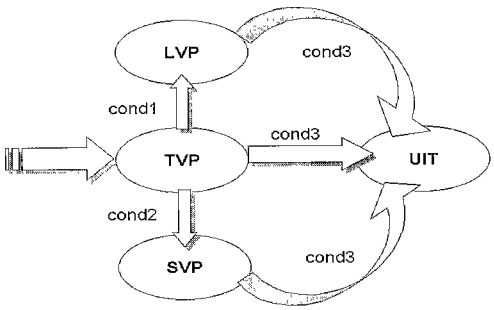


그림 6 제안한 하이브리드 결과 값 예측기에서 동적 분류를 위한 상태 전이도

갖는 하이브리드 결과 값 예상 방법을 제안한다. 예상이 자주 틀리는 명령어들은 UIT에 할당하여 반입된 명령어가 UIT에 있으면 예상을 하지 않으며 예측기의 엔트리에도 할당하지 않는 메커니즘을 제안하여, Kai Wang [6]에서 제안한 하이브리드 예측기와 동일한 하드웨어 비용으로 높은 예상 정확도를 갖도록 한다.

### 3.2 명령어의 동적 분류 방법

동적 분류방법은 예상이 자주 틀리는 명령어와 명령어의 예상을 위해 가장 예상 정확도가 높은 예측기를 동적으로 선택하게 한다. 명령어는 최초 Two-level 결과 값 예측기(TVP)에 할당하고, 여과 카운터는 '2'로 설정한다. 명령어가 다시 반입되면(fetch) 여과 카운터를 확인, '2'로 설정되어 있으면 두 개의 결과 값을 비교하여 동일하면 Last 결과 값 예측기(LVP)로 전이시키고, 같지 않을 경우 여과 카운터를 '1'로 수정하고 Stride필드와 결과 값 필드에 값을 넣는다. 다시 명령어가 반입되면 여과 카운터를 확인, '1'로 되어 있으면 세 개의 결과 값을 비교하여 Stride-based 결과 값 패턴이면 Stride-based 결과 값 예측기(SVP)로 전이시킨다. Last 결과 값 패턴도 Stride-based 결과 값 패턴도 아니면 여과 카운터를 '3'으로 설정하여 다음부터 반입된 명령은Two-level 결과 값 패턴으로 처리하게 한다. 이렇게 함으로서 초기에 패턴을 알아낼 수 있고 계속해서 패턴을 검사할 필요가 없어 Two-level 결과 값 예측기의 오버헤드를 줄인다. 또한 각각의 예측기는 처음 엔트리에 할당될 때 신뢰성 카운터를 임계치 값으로 초기화하고, 예상이 틀리면 신뢰성 카운터를 '1'씩 감소시킨다. 신뢰성 카운터가 '0'이 되면(즉 연속해서 임계치만큼 틀리면) 예상하기 어려운 명령어로 판단하여 이러한 명

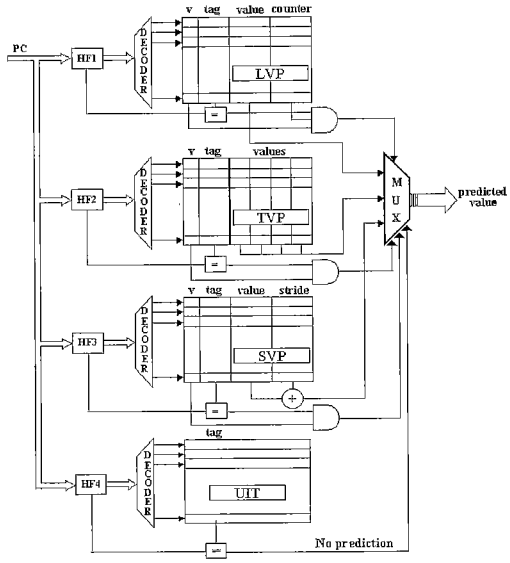


그림 7 제안한 하이브리드 결과 값 예측기의 블록도

령어들은 UIT로 전이하여 해당 명령어는 예상하지 못하도록 한다. 중간에 한번이라도 예상이 맞으면 신뢰성 카운터를 다시 임계치로 변경하여 예상을 계속할 수 있게 한다. 명령어의 예측기를 동적으로 선택하는 상태 전이도를 [그림 6]에 나타내었다.

### 3.3 제안된 하이브리드 예측기의 메커니즘

제안한 하이브리드 예측기의 구조는 [그림 7]과 같다. 명령어가 반입되면 그 명령어의 PC로 해쉬 함수(HF1, HF2, HF3, HF4)를 사용하여 인덱스를 해서 3개의 예측기 중 그 명령어가 할당된 엔트리를 선택하고, 선택된 예측기의 엔트리에 있는 값을 예상 값으로 사용한다. 만약 반입된 명령어가 UIT에 있으면 예상을 하지 않는다. 명령어에 대한 엔트리가 3개의 예측기에 할당되지 않았거나 UIT에 없다면 예상은 이루어지지 않으며, 명령어의 동적 분류를 위해 Tow-level 예측기에 해당 명령어를 할당하고 명령어의 수행이 완료되면 수행 결과 값을 할당된 엔트리의 결과 값 필드에 저장한다. 동일 명령어가 다시 반입할 때를 위해 valid 비트(v)를 세트시켜 계속적으로 분류를 수행하도록 한다.

반입된 명령어가 3개의 예측기나 UIT에 있을 경우 예상과 갱신 및 예측기의 동적 전이는 다음과 같다.

(1) 반입된 명령어가 LVP에 엔트리를 갖고 있을 경우

명령어 반입시 LVP 엔트리의 포화 카운터 값이 '2'이

상이면 결과 값 필드에 있는 값으로 예상하고, 명령어 수행이 완료 후 예상 결과가 맞으면 포화 카운터 값을 증가시키고, 신뢰성 카운터를 임계치 값으로 세트시켜 예상을 할 수 있는 기회를 오래 가지도록 한다.

예상 결과가 틀렸으면 수행 결과를 결과 값 필드에 저장하고 포화 카운터와 신뢰성 카운터를 감소시킨다. 신뢰성 카운터가 '0'이 되면 해당 명령어를 UIT로 전이하고 해당 엔트리의 valid 비트를 '0'으로 리셋시켜 초기화한다.

(2) 반입된 명령어가 SVP에 엔트리를 갖고 있을 경우

명령어가 반입시 SVP 엔트리의 상태 필드가 steady('2')이면 결과 값 필드와 stride필드의 합으로 예상한다. 명령어 수행 완료 후 정확한 결과 값과 stride 값을 해당 엔트리로 갱신하고 연속해서 예상이 틀려 신뢰성 카운터가 '0'이 되면 명령어를 UIT로 전이시키고 해당 엔트리의 valid 비트를 '0'으로 리셋시켜 초기화한다.

(3) 반입된 명령어가 UIT에 엔트리를 갖고 있을 경우

LVP, SVP, TVP에서 예상이 어려운 명령어들은 UIT로 보내진다. 따라서 UIT에 있는 명령어들은 예상을 하지 않고 일반적인 명령어처럼 수행이 완료될 때까지 이후의 종속적인 명령어들이 결과를 기다리게 된다. 이는 예상이 어려운 명령어를 잘못 예상해서 생기는 페널티를 줄이기 위해서이다.

(4) 반입된 명령어가 TVP에 엔트리를 갖고 있을 경우

본 논문에서 제안한 예측기에서 TVP는 결과 값을 예상하는 예측기 역할은 물론 결과 값들의 패턴을 조사하여 적합한 예측기를 선택하는 여과기의 역할로도 사용된다. 우선 예측기의 역할을 보면, 명령어 반입 시 TVP 엔트리의 VHP 필드를 사용하여 PHT 엔트리를 인덱스하고, PHT 엔트리의 선택된 카운터 값이 임계치 이상이면 해당 카운터에 대응하는 결과 값 필드의 값으로 예상한다. 여과기의 역할을 보면, 명령어 수행 완료후 수행 결과 값과 마지막 값이 같으면 수행 결과 값을 LVP의 결과 값 필드로 전이하고 LVP의 포화 카운터 값을 '2'로, valid 비트를 '1'로 설정한다. 또, 패턴이 Stride 패턴이면 수행 결과 값과 차이 값을 SVP의 결과 값 필드와 차이 값 필드로 전이하고 상태 필드를 steady('2')로, valid 비트를 셋트시키고 TVP의 해당 엔트리의 valid 필드를 리셋시킨다. TVP에서도 마찬가지로 예상을 해서 틀리면 신뢰성 카운터는 감소하고 신뢰성 카운터가 '0'이 되면 해당 명령어를 UIT로 전이하고 valid 비트를 리셋시킨다.

## 4. 실험 환경

본 실험에서는 앞서 언급한 Kai Wang[5]의 하이브리드 예측기(Hybrid)와 제안된 하이브리드 예측기(Filtered Hybrid : F-Hybrid)의 상대적인 예상빈도(Prediction rate) 및 예상 정확도(Prediction accuracy)를 비교, 분석하기 위해 5개의 SPECint 95 벤치마크에 대해 execution-driven 시뮬레이터인 SimpleScalar(버전 2.0) 툴셋[10]을 사용하여 시뮬레이션을 수행하였다. [표 1]은 실험에 사용한 벤치마크이다. 첫 번째 열은 실험에 사용된 SPECint 95 벤치마크 프로그램이고, 두 번째 열은 프로그램에 사용한 입력 파일들이고, 마지막 열은 실제 프로그램을 수행했을 때의 명령어 수를 나타내며 단위는 M($10^6$)이다.

표 1 벤치마크 프로그램과 입력 데이타

| Benchmark | Input set | Dynamic Instruction ($10^6$) |
|---|---|---|
| go | 2stone9.in | 100M |
| m88ksim | tiny.in | 100M |
| vortex | vortex.in | 65M |
| li | queen6.lsp | 41M |
| compress | test.in | 35M |

## 5. 실험 결과

이 장에서는 Kai Wang[6]의 하이브리드 예측기(Hybrid)와 제안된 하이브리드 예측기(F-Hybrid)의 엔트리 수, 예상율 및 예상정확도를 비교 분석한다.

### 5.1 예상 테이블의 엔트리 수

[그림 8]는 LVP, SVP, TVP 테이블 크기를 제한하지 않았을 때의 각각의 예측기가 정확하게 예측한 부분을 그래프로 나타낸 것이다. 그림에서 보듯이 LVP 65%, SVP 15%, TVP 20%의 비율로 결과가 나왔다.

[표 2]는 [그림 8]의 결과를 기준으로 하이브리드 예측기(Hybrid)와 동일한 하드웨어 비용으로 본 논문에서 제안된 하이브리드 예측기(F-Hybrid)가 가질 수 있는 테이블 엔트리 수를 나타내고 있다. [표 2]에서 보듯이 동일한 하드웨어 비용을 가정하면 Hybrid 예측기의 테이블은 F-Hybrid 보다 적은 엔트리 수를 갖는다. 테이블의 엔트리 수가 적다는 것은 예측기에 할당될 수 있는 명령어가 적다는 것이다. 따라서, 여러 개의 명령어가 같은 엔트리를 사용하게 되고 명령어의 할당 시 대체(replacement)가 발생하게 된다. 이렇게 용량부족으로

인한 실패(capacity misses)를 유발하고, 명령어 대체가 이루어진 후에는 학습시간 동안에는 예상을 할 수 없으므로 예상할 수 있는 명령어의 빈도수를 감소시킨다.

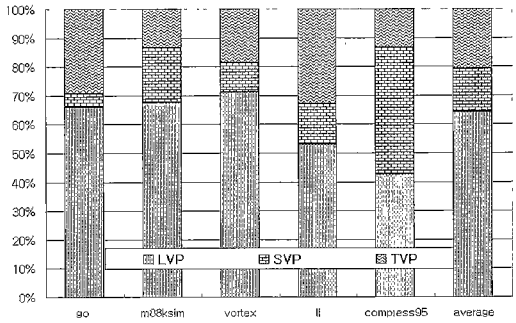


그림 8 LVP, SVP, TVP의 예측 분포도 (각각 8K 엔트리)

표 2 각 하이브리드 예측기의 테이블 엔트리의 비교

| Hybrid | F-Hybrid | | | |
|---|---|---|---|---|
| | LVP | SVP | TVP | UIT |
| 1024 | 2048 | 256 | 512 | 1024 |
| 2048 | 4096 | 512 | 1024 | 2048 |

제안된 하이브리드 예측기(F-Hybrid)는 기존의 하이브리드 예측기(Hybrid)의 엔트리 수 보다 동일한 하드웨어 비용으로 3배정도 많은 엔트리 수를 가진다. 이것은 제안된 하이브리드 예측기에 예상 가능한 명령어를 할당할 수 있는 엔트리가 더 많이 존재한다는 사실을 나타내고 있다. 제안된 하이브리드 예측기의 장점은 동일 하드웨어 비용에서 기존의 하이브리드 예측기보다 많은 명령어를 테이블 엔트리에 할당할 수 있으므로 명령어 대체를 줄여 예상을 할 수 있는 빈도수를 증가시키는 것이다.

이는 기존의 하이브리드 예측기보다 많은 명령어를 예상할 수 있으며, 또한 정확히 예상되는 명령어의 대체를 줄임으로서 예상 빈도수를 높일 수 있다.

### 5.2 예상 정확도

제안된 하이브리드 예측기를 사용하여 명령어의 결과 값을 예상하였을 경우에 나타나는 실험 결과와 예상 정확도를 [그림 9], [그림 10]에서 보여주고 있다.

[그림 9]는 제안된 하이브리드 예측기의 실험 결과로서 Table miss는 예측기의 테이블에 해당 엔트리가 할당이 되지 않아서 예상을 수행하지 못한 경우를 나타내고, Not prediction은 엔트리가 존재하더라도 포화 카운터 값이 임계치보다 작거나(LVP 또는 TVP), 상태 필드의 값이 예상을 수행할 수 없는 상태(SVP)를 나타내서 예상을 수행하지 못한 명령어 비율을 나타내고, Correct prediction은 예측기의 테이블에 할당된 엔트리가 존재하고, 해당 엔트리의 포화 카운터 값이 임계치보다 크거나(LVP 또는 TVP의 경우) 상태필드가 예상 가능한 상태(SVP의 경우)를 나타내서 예상이 이루어진 명령어 중 정확한 예상이 이루어진 비율을 나타낸다. Incorrect prediction은 예상이 이루어진 명령어 중에서 예상이 틀린 비율을 나타낸 것이다. 그리고 No prediction은 각 예측기에서 예상이 어려운 명령어로 판단된 명령어들로써 UIT에 할당되어 예상을 하지 있는 명령어의 비율을 나타낸다.

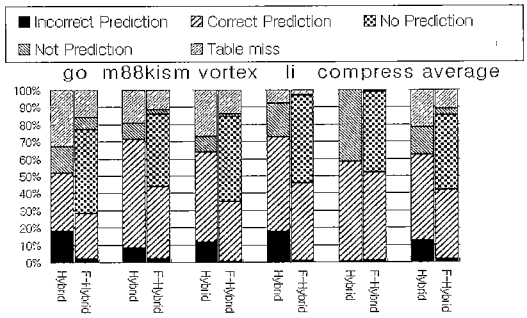


그림 9 각 하이브리드 예측기의 실험 결과

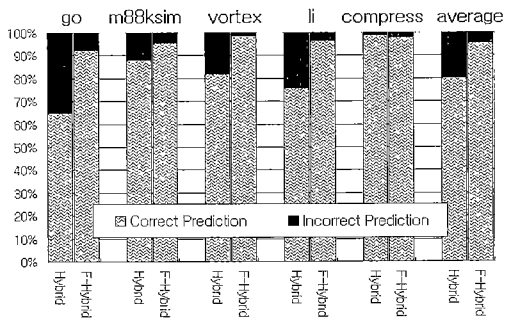


그림 10 각 하이브리드 예측기의 예상 정확도

실험 결과, Table miss가 기존의 하이브리드 예측기인 Hybrid보다 현저히 감소하여 제안된 예측기인 F-Hybrid가 더 많은 명령어를 예측기의 엔트리에 할당하고 있음을 알 수 있다. 또한, Incorrect Prediction이 기존의 하이브리드 예측기보다 평균 12%에서 2%로 현저히 줄어들므로써 잘못된 예상에 의한 페널티를 감소 시

킬 수 있는 잠재력을 보이고 있다.

[그림 10]은 본 실험에서 비교한 두 개의 하이브리드 예측기의 예상 정확도를 보여주고 있다. Correct Prediction은 예상을 수행한 명령어 중에서 정확하게 예상된 비율이고, Incorrect Prediction은 예상이 이루어진 명령어 중에서 예상이 틀린 비율이다. 예상 정확도가 평균 80%에서 95%로 향상된 것을 볼 수 있다.

F-Hybrid가 일반적으로 Hybrid보다 정확한 예상의 절대적인 비율이 감소하는 것으로 나타나고 있지만, 틀린 예상의 비율을 보다 효과적으로 감소시키고 있다. 즉, 상대적인 감소율을 측정했을 경우, 정확한 예상보다 틀린 예상을 훨씬 효과적으로 제거함으로써 예상 정확도를 향상시킬 수 있다. 결과 값 예상방법의 경우, 정확한 예상의 절대적인 비율이 50% 내외로 나타나고 있다. 이는 그만큼 틀릴 예상의 가능성도 많다는 것이다. 고성능의 프로세서에서는 이렇게 틀린 예상에 대한 페널티가 아주 큰 부작용으로 나타나고 있다. 잘못 예상된 명령어와 종속적인 관계를 가진 명령어들이 하드웨어 자원을 낭비하게되므로 다른 명령어들이 자원을 사용할 수 있는 기회를 잃게 되며, 종속적인 관계를 가진 명령어들을 반입 및 재수행시켜야 하는 오버헤드가 발생하기 때문에 예상이 틀릴 가능성이 높은 명령어(No Prediction)는 예상을 하지 못하게 하여 잠재적인 성능 감소를 피할 수 있다.

따라서, UIT를 사용해 예상이 자주 틀리는 명령어는 예상하지 않음으로서 예상 정확도를 향상시키는 것은 물론이고 예상이 틀릴 경우 발생하는 많은 페널티를 피할 수 있다.

## 6. 결 론

본 논문은 슈퍼스칼라 프로세서에서 ILP를 향상시키기 위해 결과 값을 생성하는 명령어와 그 값을 사용하는 명령어 사이의 데이타 종속성을 제거하기 위해 명령어의 결과 값을 예상하는 새로운 하이브리드 예상기법을 제안하였다. 기존의 하이브리드 예상기법은 각각의 예측기에 예상할 명령어를 중복하여 할당함으로서 많은 하드웨어 비용이 든다. 그러나, 제안한 하이브리드 예측기는 명령어에 대해 가장 예상 정확도가 높은 예측기를 추가적인 하드웨어 부담없이 동적으로 선택함으로서 중복된 할당을 피할 수 있고, 따라서 보다 많은 명령어를 예측기에 할당하여 예상 빈도수와 예상 정확도를 높였다. 또한 예상이 어려운 명령어들을 동적으로 탐색하는 메카니즘을 제안하였고 그러한 명령어들을 UIT에 저장하여 반입된 명령어가 UIT에 있을 경우 명령어의 결과 값을 예상하지 않고, 예측기의 엔트리도 할당하지 않음으로서 SPECint95 벤치마크 프로그램에서 잘못 예상되는 율(Incorrect Prediction Rate)을 평균 12%에서 2%로 현저히 낮추었고, 예측율은 평균 79%에서 90%로, 예상정확도는 평균 80%에서 95%로 향상시켰다.

향후 과제로는 예상을 정확히 하기 위해서는 좀더 정확한 패턴 분석에 대한 연구가 필요하다. 즉, UIT에 할당되는 명령어를 최적화시켜 예상이 가능한 명령어가 초기의 잘못된 패턴으로 인해서 예상을 수행하지 못하는 경우를 최소화 시켜 절대적인 예상율도 향상시킬 수 있을 것이다. 또한 본 논문의 실험은 반입된 명령어의 실행은 순서적(in-order)으로 처리하여 구현하였으나, 순서에 관계없이 실행 가능한 명령어부터 처리하는 비순서적(out-of-order) 방법으로 구현해서 틀린 예상으로 인해 나타나는 페널티의 부작용과 전체적인 실행속도의 성능 향상도 측정해 보아야 할 것이다.

## 참 고 문 헌

[1] M. H. Lipasti, C. B. Wilderson, J. P. Shen, "Value Locality and Load Vaule Prediction," ASPLOS-VII, pp. 138-147, October 1996.

[2] M. H. Lipasti and J. P. Shen, "Exceeding the Dataflow limit via Value Prediction," MICRO-29, pp. 226-237, December 1996.

[3] Y. Sazeides and J. E. Smith, "The Predict-ability of Data Values," MICRO-29, pp. 248-258, December 1996.

[4] F. Gabbay and A. Mendelson, "Can Program Profiling Support Value prediction?," MICRO-30, pp. 270-280, December 1997.

[5] T-Y Yeh and Y. N. Patt, "Alternative Impementations of Two-Level Adaptive Branch Prediction," ISCA-19, pp.124-134, 1992.

[6] K. Wang and M. Franklin, "Highly Accurate Data Value Prediction using Hybrid Predictors," MICRO-30, pp. 281-290, December 1997.

[7] J. Gonzalez and A. Gonzalez, " The potential of data value speculation to boost ilp," ICS-12, 1998

[8] G. Reinman and B. Calder, "Predictive techniques for aggressive load speculation," MICRO-31, 1998

[9] T. Nakra, R. Gupta and M.L. Soffa, "Global Context-Based Value Prediction," HPCA-5, January 1999.

[10] D.C. Burger and T.M. Austin , "The simplescalar tool set, version 2.0" Technical Report CS-TR-97-1342, University of wisconsin, Madison, June 1997.

[11] Calder B., Feller P. and Eustace A., "Value Profiling," MICRO-30, pp. 259-269, December 1997.

[12] F. Dahlgren and P. Stenstrom, "Evaluation of Hardware-Based Stride and Sequential Prefetching in Shared-Memory Multiprocessors," IEEE Transactions on Parallel and Distributed Systems. vol. 7. no. 4. pp. 385-398. April 1996.
[13] B. Calder, G. Reinman, D. M. Tullsen, "Selective Value Prediction," ISCA-26, May 1999.
[14] B. Rychlik, J. W. Faistl, B. P. Krug, A. Y. Kurland, J. J. Sung, M. N. Velev, J. P. Shen, "Efficient and Accurate Value Prediction Using Dynamic Classification" Tech. rep. CMUART-1998-0
[15] Y.Sazeides, J.E. Smith, "The Predictability of Data Values," Micro-30, December 1997.

신 영 호
1999년 수원대학교 전자계산학과 이학사. 1999년 3월 ~ 현재 수원대학교 컴퓨터과학과 석사과정. 관심분야는 ILP 프로세서의 분기예상 메커니즘 및 결과값 예상 메커니즘, ILP프로세서의 정적, 동적 명령어 스케쥴링 등임.

윤 성 룡
1999년 수원대학교 전자계산학과 이학사. 1999년 3월 ~ 현재 수원대학교 컴퓨터과학과 석사과정. 관심분야는 최적화 컴파일러 설계, ILP 프로세서의 분기예상 메커니즘 및 결과값 예상 메커니즘, ILP 프로세서의 정적, 동적 명령어 스케쥴링 등임.

조 영 일
1980년 한양대학교 전자공학과 공학사. 1982년 한양대학교 전자공학과 공학석사. 1985년 한양대학교 전자공학과 공학박사. 1986년 3월 ~ 현재 수원대학교 컴퓨터과학과 부교수. 관심분야는 최적화 컴파일러 설계, ILP 프로세서의 분기예상 메커니즘 및 결과값 예상 메커니즘, ILP프로세서의 정적, 동적 명령어 스케쥴링 등임.