슈퍼스칼라 프로세서에서 동적 분류를 사용한 하이브리드 결과 값 예측기

신영호, 윤성룡, 박홍준, 이원모, 김주익, 조영일
수원대학교 전자계산학과
e-mail : yicho@mail.suwon.ac.kr

A Hybrid Value Predictor using Dynamic Classification in Superscalar Processors.

Young-Ho Shin, Sung-Lyong Yoon, Hong-Jun Park,
Won-Mo Lee, Ju-Ik Kim, Young-Il Cho
Dept of Computer Science, The University of Suwon

요약

슈퍼스칼라 프로세서의 성능을 향상시키기 위해서는 데이터 종속성에 의한 장애를 제거해야 한다. 최근 여러 논문들은 이러한 데이터 종속성을 제거하기 위해서 명령의 결과 값을 예상하는 메커니즘이 연구되고 있다.

결과 값 예상 메커니즘 중 여러 예측기를 하이브리드해서 사용하는 방법은 각각 하나의 예측기만을 사용하는 방법보다 더 좋은 성능을 얻을 수 있다. 그러나 종전의 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 예상하기 어려운 명령어를 동적으로 찾아내어 예상하지 않음으로서 잘못 예상한 misprediction 페널티를 줄이고 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

1. 서론

현재 마이크로프로세서의 구조의 성능은 높은 명령어 이슈율과 명령어 수준 병렬성(Instruction Level Parallelism : ILP)을 가능한 최대로 이용하는 것이 중요하다. ILP를 처리 할 때 주요 장애 요소로는 제어 종속과 데이터 종속이 있다. 제어종속은 조건분기 명령이 반입 될 때 분기 방향과 다음에 수행할 명령어 주소를 예상하는 분기 예상기법으로 문제를 해결할 수 있다. 데이터 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 true-data 종속성을 제거하기 위해 결과 값(value) 예상기법을 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].

명령어의 결과를 바로 이전에 수행된 결과로 예상하는 Last 결과 값 예측기[1,2], 일정한 값(stride)만큼 변하는 명령어를 예상하는 Stride-based 결과 값 예측기[5,6], 이전에 수행된 4개의 결과 중에 하나를 예상하는 Two-level 결과 값 예측기[5]가 있다. 또한 다양한 특성을 갖는 여러 명령어들을 모두 예상하기 위해 여러 예측기를 혼합하여 사용하는 하이브리드 예측기[5,7]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 예

측기의 엔트리에 할당되지 못하는 문제점을 갖고 있다.

본 논문에서는 앞서 기술한 세 가지 결과 값 예측기의 장점을 혼합해서 사용하며 명령어들의 수행 결과의 패턴을 조사하여 각 패턴에 맞는 결과 값 예측기를 사용하게 하는 하이브리드 결과 값 예측기를 제안한다. 제안된 방법은 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 어려운 명령어들을 동적으로 찾아내어 그러한 명령어들은 예측기에 할당시키지 않음으로서 예측기를 보다 효율적으로 사용하여 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 예측기를 설명하고 그들의 장단점을 고찰한다.

2.1 Last 결과 값 예측기

Last 결과 값 예측기[2]는 명령어의 마지막 수행 결과 값을 저장하는 VPT(Value Prediction Table)과 예상을 결정하는 2비트 CT(Classification Table)로 구성되어있다. 두 개의 테이블은 명령어 주소(PC)로 인덱스하고 CT테이블 값(CT Counter)이 '2'이상일 때만 VPT테이블의 값을 수행 시 예상 값으로 제공하는 방법이다.

Last 결과 값 예측기는 적은 하드웨어 비용을 요구되지만 예상 정확도는 40%~50%로 낮다는 단점을 갖는다.

2.2 Stride-based 결과 값 예측기

Stride-based 결과 값 예측기[5]는 명령어의 결과 값과 마지막 두 번의 수행 결과 값의 차(stride)를 VPT에 저장하여 명령어의 반입(fetch)시 마지막 수행 결과 값과 stride의 합으로 예상한다.

이 방법은 Last 결과 값 예측기보다 좋은 성능을 가진다.

2.3 Two-level 결과 값 예측기

Two-level 결과 값 예측기[5]는 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 이전 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 많은 하드웨어 비용을 요구한다는 단점을 갖는다.

2.4 하이브리드 결과 값 예측기

하이브리드 결과 값 예측기[5]는 Stride-based 결과 값 예측기와 Two-level 결과 값 예측기를 결합한 것으로 신뢰성 카운터(Confidence Counter)에 의해 한 예측기의 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 다양한 패턴을 갖는 명령어를 모두 예상하기 때문에 높은 예상 정확도를 갖는다는 장점을 갖지만 명령어가 두 개의 예측기의 엔트리에 중복됨으로써 많은 하드웨어 비용을 요구한다는 단점을 갖는다.

위에서 보듯이 명령어의 결과 값을 예상하기 위한 여러 가지 예상방법이 제안되었으나 아직 만족할 만한 예상정확도를 갖지 못하고 있다. 또한 예상 값이 틀릴 경우 회복(recovery)을 위한 페널티가 크므로 예상이 어려운 명령어는 예상하지 않는 것이 성능 향상에 도움이 된다.

3. 제안된 하이브리드 결과 값 예상방법

이 장에서는 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 엔

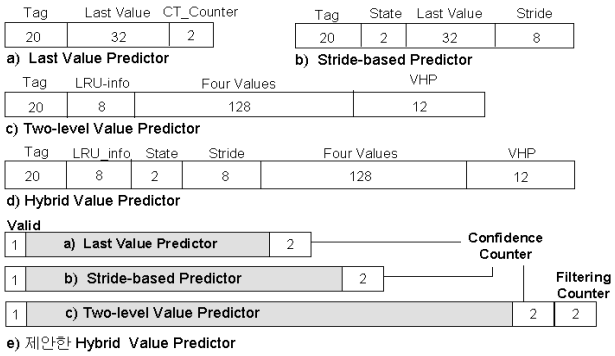


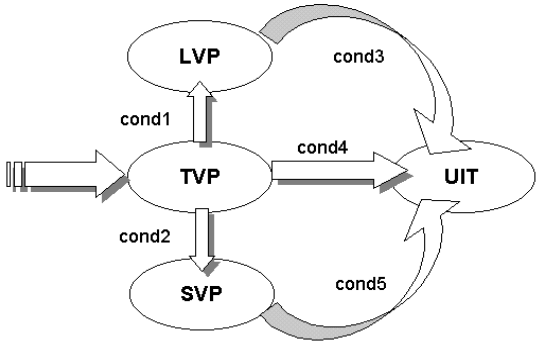
그림 1 여러 예측기에서의 테이블 엔트리 구조

트리를 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 예상하기 어려운 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 1]은 여러 예측기가 사용하는 예상테이블의 엔트리 구조를 보여주고 있다. Last 결과 값 예상방법의 테이블의 엔트리(그림 1, a)는 태그 필드, 한 개의 Last 결과 값 필드, 카운터(CT Counter)로 구성되어 있다. Stride-based 결과 값 예상방법의 엔트리(그림 1, b)는 태그필드, 상태필드, stride 필드, Last 결과 값 필드로 구성되며 Last 결과 값 예상방법보다 좀더 많은 하드웨어를 필요로 한다. Two-level 결과 값 예상방법의 엔트리(그림 1, c)는 태그필드, LRU 필드, 네 개의 마지막 결과 값 필드, 2*p 비트의 패턴 히스토리 필드와 VPT로 구성되며 다른 예측기보다 상당히 많은 하드웨어를 요구한다.

본 논문에서는 Last 결과 값 예상 방법, Stride-based 결과 값 예상 방법, Two-level 결과 값 예상 방법의 장단점에 착안하여, 세 개의 예상 방법을 결합시킨 하이브리드 결과 값 예상 방법을 제안한다. 또한 예상이 어려운 명령어들은 UIT에 할당하여 반입된 명령어가 UIT에 있으면 예상을 하지 않으며, 예측기의 엔트리도 할당하지 않는 메카니즘을 제안하여, Kai Wang[5]에서 제안한 하이브리드 예측기와 동일한 하드웨어 비용으로 높은 예상 정확도를 갖도록 한다.

본 논문에서 제안한 하이브리드 예측기의 구조는



cond1 : 엔트리의 결과 값의 차가 두 번 연속 '0' 일 경우
cond2 : 엔트리의 결과 값의 차가 두 번 연속 같을 경우
cond3 : 엔트리의 신뢰성 카운터 필드가 '0'일 경우
cond4 : 엔트리의 신뢰성 카운터 필드가 '0'일 경우
cond5 : 엔트리의 신뢰성 카운터 필드가 '0'일 경우

그림 2 제안한 하이브리드 결과 값
예측기에서 동적 분류를 위한 상태 전이도

그림 1의 e)에서 보듯이 추가된 부분이 있다. Valid 비트는 해당 엔트리에 명령어가 할당되어 있는지를 나타낸다. Valid 비트가 '1'로 설정된 예측기로 예상을 하고 모두 '0'일 때는 Two-level 결과 값 예측기에 할당한다. 또한, 신뢰성 카운터는 예상이 어려운 명령어들을 찾아내는데 사용한다. Filtering 카운터(F_Counter)는 Two-level 결과 값 예측기에서 명령어를 동적 분류하기 위해 사용된다.

3.2 명령어의 동적 분류 방법

동적 분류방법은 예상이 어려운 명령어와 명령어의 예상을 위해 가장 예상 정확도가 높은 예측기를 동적으로 선택하게 한다. 명령어는 최초 Two-level 결과 값 예측기(TVP)에 할당하고, F_Counter는 '2'로 설정한다. 명령어가 다시 반입되면(fetch) F_Counter를 확인, '2'로 설정되어 있으면 두 개의 결과 값을 비교하여 Last 결과 값 패턴이면 Last 결과 값 예측기(LVP)로 전이시키고, 같지 않을 경우 F_Counter를 '1'로 수정하고 Stride필드와 결과 값 필드에 값을 넣는다. 다시 명령어가 반입되면 F_Counter를 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 F_Counter를 '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 '3'으로 초기화하고, 예상이 틀리면 신뢰성 카운터를 '1'씩 감소시킨다. 신뢰성 카운터가 '0'이 되면 (즉 연속해서 세 번 틀리면) 예상하기 어려운 명령어로 판단하여 이러한 명령어들은 UIT로 전이하여 그 명령어는 예상하지 못하도록 한다. 명령어의 예측기를 동적으로 선택하는 상태 전이도를 [그림 2]에 나타내었다.

3.3 제안된 하이브리드 예측기의 메카니즘

제안한 하이브리드 예측기의 구조는 [그림 3]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, 먼저 Tow-level 예측기에 해당 명령어를 할당하고 명령어의 수행이 완료되면 수행 결과 값을 할당된 엔트리의 결과 값 필드에 저장한다. 해당 명령어가 다시 반입할 때를 위해 valid 비트(v)를 세트시켜 다음에 예상하도록 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 CT 카운터 값이 '2' 이상이면 결과 값 필드에 있는 값으로 예상하고, 명령어 수행이 완료 후 예상 결과가 맞으면 CT 카운터 값을 증가시키고, 신뢰성 카운터를 '3'으로 고쳐 예상을 할 수 있는 기회를 오래 가지도록 한다.

예상 결과가 틀렸으면 수행 결과를 결과 값 필드에 저장하고 CT 카운터를 감소시키며 신뢰성 카운터 값을 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(10)이면 결과 값 필드와 stride필드의 합으로 예상한다. 명령어 수행 완료 후 정확한 결과 값과

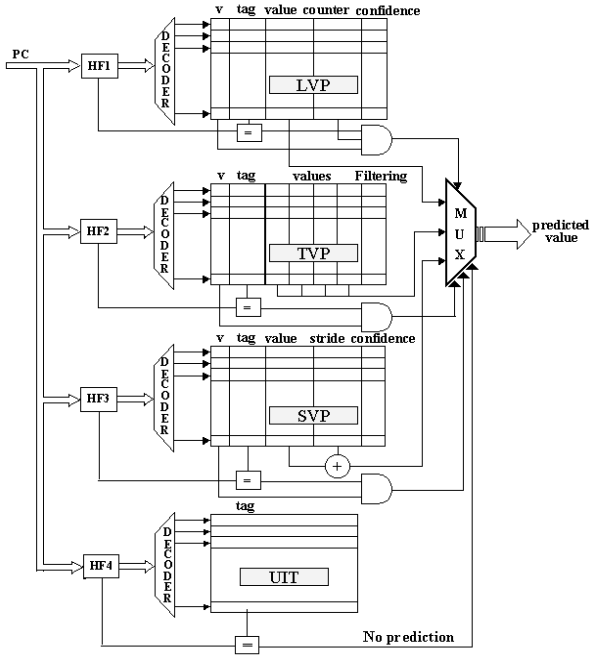


그림 3 제안한 하이브리드 결과값 예측기의 블럭도

stride 값을 해당 엔트리로 갱신하고 연속해서 예상이 틀려서 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 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 조사하는 필터기의 역할로도 사용된다. 우선 예측기의 역할을 보면, 명령어 반입 시 TVP 엔트리의 VHP 필드를 사용하여 PHT 엔트리를 인텍스하고, PHT 엔트리의 선택된 카운터 값이 임계치 이상이면 해당 카운터에 대응하는 결과 값 필드의 값으로 예상한다. 필터기의 역할을 보면, 명령어 수행 완료후 수행 결과 값과 마지막 값이 같으면 수행 결과 값을 LVP의 결과 값 필드로 전이하고 LVP의 CT 카운터 값을 '2'로,

valid 비트를 '1'로 설정한다. 또, 패턴이 Stride 경향이 있으면 수행 결과 값과 stride값을 SVP의 결과 값 필드와 stride 필드로 전이하고 state 필드를 steady(10)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 예상율(Prediction rate) 및 예상 정확도(Prediction accuracy)를 비교, 분석하기 위해 5개의 SPECint 95 벤치마크에 대해 execution-driven 시뮬레이터인 SimpleScalar(버전 2.0) 툴셋[9]을 사용하여 시뮬레이션을 수행하였다. [표 1]은 실험에 사용한 벤치마크이다. 첫 번째 열은 실험에 사용된 SPECint 95 벤치마크 프로그램이고, 두 번째 열은 프로그램에 사용한 Input 파일들이고, 마지막 열은 실제 프로그램을 수행했을 때의 명령어 수를 나타내며 단위는 M(10^6)이다.

표 1 벤치마크 프로그램과 입력 데이터

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[5]의 하이브리드 예측기(Hybrid)와 제안된 하이브리드 예측기(F-Hybrid)의 엔트리 수, 예상정확도를 비교 분석한다.

5.1 예상 테이블의 엔트리 수

[표 2]는 하이브리드 예측기(Hybrid)와 동일한 하드웨어 비용으로 본 논문에서 제안된 하이브리드 예측기(F-Hybrid)가 가질 수 있는 테이블 엔트리 수를 나타내고 있다.

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.

표 2 두 하이브리드 예측기의 테이블 엔트리의 비교

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

5.2 예상 정확도

제안된 하이브리드 예측기를 사용하여 명령어의 결과 값을 예상하였을 경우에 나타나는 실험 결과와 예상 정확도를 [그림 4], [그림 5]에서 보여주고 있다.

[그림 4]는 예상할 수 있는 명령어의 실험결과로서 Table miss는 예측기의 테이블에 해당 엔트리가 할당이 되지 않아서 예상을 할 수 없는 경우를 나타내고, Not prediction은 엔트리가 존재하더라도 카운터 값이 임계치보다 작아서 예상을 못한 명령어 비율을 나타내고, Correct prediction은 예측기의 테이블에 할당된 엔트리가 존재하고, 예상이 이루어진 명령어 중 정확한 예상이 이루어진 비율을 나타낸다. Incorrect prediction은 예상이 이루어진 명령어 중에서 예상이 틀린 비율을 나타낸 것이다. 그리고 No prediction은 본 논문에서 각 예측기에서 예상이 어려운 명령어로 판단된 명령어들이 UIT에 있는 비율을 나타낸다. Incorrect Prediction부분이 기존의 하이브리드 예측기보다 평균 12%에서 2%로 현저히 줄어든 것을 볼 수 있다.

[그림 5]은 본 실험에서 비교한 두 개의 하이브리드 예측기의 예상 정확도를 보여주고 있다. Correct Prediction은 예상한 명령어 중에서 정확하게 예상한 비율이고, Incorrect Prediction은 예상한 명령어 중에서 예상이 틀린 비율이다. 예상 정확도는 평균 80%에서 95%로 향상된 것을 볼 수 있다. 이는 UIT을 사용함으로서 예상하기 어려운 명령어는 예상하지 않음으

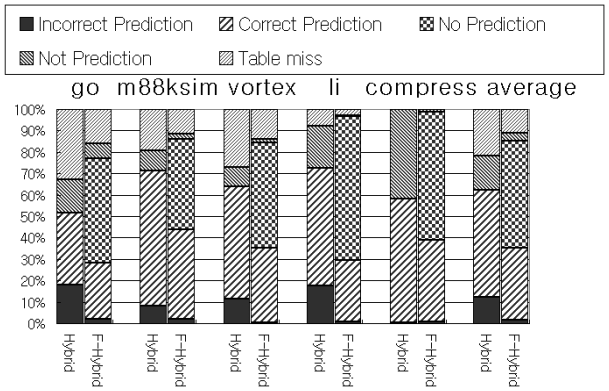


그림 4 각 하이브리드 예측기의 실험 결과

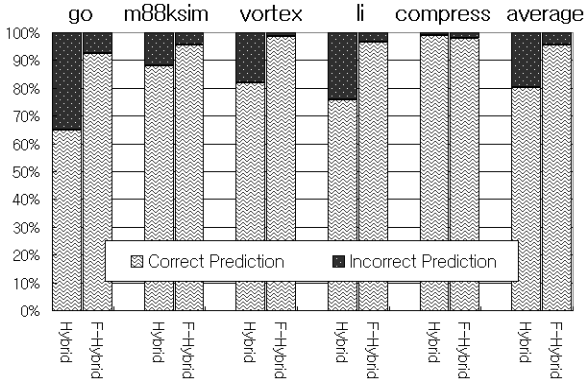


그림 5 각 하이브리드 예측기의 예상 정확도

로서 예상 정확도를 향상시키는 것은 물론이고 예상이 틀릴 경우 발생하는 높은 페널티를 피할 수 있다.

6. 결론

본 논문은 슈퍼스칼라 프로세서에서 ILP를 향상시키기 위해 결과 값을 생성하는 명령어와 그 값을 사용하는 명령어 사이의 데이터 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 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)을 평균 12%에서 2%로 현저히 낮추었고, 예측율은 평균 79%에서 90%로, 예상정확도는 평균 80%에서 95%로 향상시켰다.

향후 과제로는 본 논문의 실험은 반입된 명령어의 실행은 순서적(in-order)으로 처리하여 구현하였으나, 순서에 관계없이 실행 가능한 명령어부터 처리하는 비 순서적(out-of-order) 방법으로 사용했을 경우의 전체적인 실행속도의 성능 향상도 측정해 보아야 할 것이다.

참고문헌

[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. 226-237, December 1996.

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

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

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

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

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

[9] 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.