슈퍼스칼라 프로세서에서 동적 분류 능력을 갖는 혼합형 데이타 값 예측기의 설계

(Design of a Hybrid Data Value Predictor with Dynamic Classification Capability in Superscalar Processors)

박 희 룡 † 이 상 정 †
(Hee-Ryong Park)(Sang-Jeong Lee)

요 약 슈퍼스칼라 프로세서에서 명령어 수준 병렬성(Instruction Level Parallelism)을 적극적으로 활용하기 위해서는 명령들 사이에 존재하는 제어 종속관계 및 데이타 종속관계를 극복하는 것이 필수적이다. 데이타 값 예측은 하나의 명령 결과가 생성되기 전에 미리 결과 값을 예측하고 이 예측된 결과를 사용하여 데이타 종속관계가 있는 명령들을 투기적으로 실행(speculative execution)하는 기법이다. 본 논문에서는 동적 분류 능력을 갖는 혼합형 데이타 값 예측기를 제안한다. 제안된 예측기는 최근 값 예측기, 스트라이드 예측기 및 2-단계 예측기를 결합한 혼합형으로 구성되며, 예측되는 명령은 하드웨어에 의한 동적 분류에 의해 각 예측기로 할당된다. 각 명령들의 특성에 따라 각 예측기로 실행 시에 동적 분류됨으로써 각 예측기는 기존의 혼합형 방식보다도 더욱 효과적으로 활용될 수 있다. 제안된 방식의 타당성 검증을 위해 실행구동방식(execution-driven) 시뮬레이터를 사용하여 SPECint95 벤치마크를 시뮬레이션하여 비교한다. 실험 결과 Instruction Per Cycle 비교실험에서 2-단계 예측기 보다 0.36, 혼합형 예측기 보다 0.018의 성능을 보였고, 제안된 방식이 기존의 혼합형 방식보다 예측 정확도가 평균 16%가 향상되었고, 하드웨어 비용을 측정한 결과 45%의 감소효과를 얻었다.

Abstract To achieve high performance by exploiting instruction level parallelism aggressively in superscalar processors, it is necessary to overcome the limitation imposed by control dependences and data dependences which prevent instructions from executing parallel. Value prediction is a technique that breaks data dependences by predicting the outcome of an instruction and executes speculatively its data dependent instruction based on the predicted outcome. In this paper, a hybrid value prediction scheme with dynamic classification mechanism is proposed. We design a hybrid predictor by combining the last predictor, a stride predictor and a two-level predictor. The choice of a predictor for each instruction is determined by a dynamic classification mechanism. This makes each predictor utilized more efficiently than the hybrid predictor without dynamic classification mechanism. To show performance improvements of our scheme, we simulate the SPECint95 benchmark set by using execution-driven simulator. The results show that our scheme effect reduce of 45% hardware cost and 16% prediction accuracy improvements comparing with the conventional hybrid prediction scheme and two-level value prediction scheme.

1. 서 론

최근 개발되고 있는 고성능 슈퍼스칼라 프로세서는 성능향상을 위해 여러 개의 명령을 동시에 이슈하고 처리하는 명령어 수준 병렬성(Instruction Level Parallelism, ILP)을 이용하고 있다. [5] 이러한 ILP 처리의 주요장애는 명령간의 종속관계인데 이는 명령의 병렬처리를 방해한다. 즉, 현재의 한 명령이 이전명령과 종속관계가 있으면 이전명령의 결과가 나올 때까지 현재의

† 비 회 원 : 김천대학 컴퓨터사무정보계열 교수
hrpark@kimcheon.ac.kr
†† 정 회 원 : 순천향대학교 컴퓨터학부 교수
sjlee@asan.sch.ac.kr
논문접수 : 1999년 12월 28일
심사완료 : 2000년 6월 12일

명령은 이슈(issue)되어 병렬로 실행할 수 없게된다. 이러한 명령어간의 종속관계는 이름종속(name dependency), 제어종속(control dependency), 데이타 종속(data dependency)관계의 3가지로 구분할 수 있다. 이름 종속관계는 레지스터나 메모리의 데이타 저장위치의 재사용으로 기인한 것으로 정적 또는 동적 재 명명(renaming)기법으로 제거할 수 있다. 제어종속 관계는 분기의 결과에 따라 프로그램의 제어흐름이 변경되는 조건분기에 기인해 발생한다. 제어 종속관계를 극복하는 방식으로는 조건분기 명령을 제거하고 제어 종속관계가 있는 명령을 조건실행(predicated execution)하여 제거하는 방식과 조건분기의 결과를 예측하여 제어 종속관계가 있는 명령들을 미리 패치(fetch)하여 투기적으로 실행(speculative execution)하는 방식이 있다.[13] 데이타 종속관계는 현재의 명령이 이전명령의 수행결과를 참조할 때 발생하고 이전명령의 결과가 발생할 때까지 현재의 명령은 실행할 수 없게된다. 데이타 종속관계는 명령간에 빈번히 발생하기 때문에 ILP 처리의 주요 장애가 되어 최근의 고성능 프로세서의 성능저하의 주된 요인이 되고 있다. 지금까지 데이타 종속관계를 극복하는 대표적인 방법으로는 최적화 컴파일러가 데이타 종속관계가 없는 독립적인 명령들을 아직 사용하지 않은 이슈슬롯(issue slot)으로 코드를 이동하여 정적 스케줄하였으나 명령의 이슈 수가 4이상으로 커짐에 따라 각 이슈슬롯에 채울 더 많은 독립적인 명령들을 채우는 것이 어렵게 되었다. 따라서 최근에는 실행되는 명령의 결과 값을 미리 예측하고 이후 데이타 종속관계가 있는 명령들에게 조기에 공급하여 실행시킴으로써 성능향상을 꾀하는 데이타 값 예측(data value prediction)방식

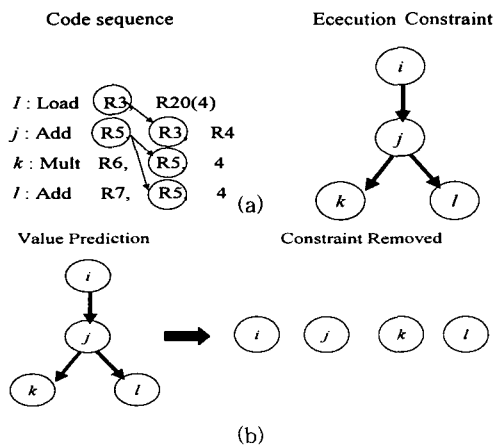


그림 1 명령에서의 데이타 종속관계

은 데이타 종속관계를 보여주는 그림으로 [그림 1]의 에 관하여 활발히 연구가 진행되고 있다.[5,6] [그림 1] (a)는 j의 R3는 i의 R3 결과가 생성되어야만 사용이 가능하며, j가 실행된 후 R5의 결과가 생성되면 명령 k, l이 실행될 수가 있다. 이러한 데이타 종속관계가 있는 명령들을 [그림 1]의 (b)와 같이 i, j의 결과 값 R3, R5를 미리 예측하여 i, j와 데이타 종속관계를 갖는 명령 k, l들에 대한 종속관계를 제거함으로써 이들 명령들을 투기적으로 실행할 수 있음을 보여 주는 그림이다.

본 논문에서는 동적 분류 능력을 갖는 혼합형 데이타 값 예측기를 제안한다. 제안된 예측기는 최근 값 예측기 [5], 스트라이드 예측기 [8] 및 2-단계 예측기 [6]를 결합한 혼합형으로 구성되며, 예측되는 명령은 하드웨어에 의한 동적 분류에 의해 각 예측기로 할당된다. 각 명령들의 특성에 따라 각 예측기로 실행 시에 동적 분류됨으로써 각 예측기는 기존의 혼합형 방식보다도 더욱 효과적으로 활용될 수 있다. 제안된 방식은 각 명령들을 만날 때 최근의 두 개의 값을 저장하고 그 차이(stride)를 구함으로써 명령들은 동적 분류된다. 즉, 그 차이가 0이면 값이 변하지 않는 상수임을 인식하고 최근 값 예측기로 할당된다. 그 차이가 같으면 일정한 증감 폭에 의해 변하는 스트라이드형 명령으로 분류하고, 만약 그 차이가 같지 않으면 증감 폭이 일정하지 않은 시퀀스를 갖는 명령으로 가정하여 2-단계 데이타 값 예측기에 할당한다. 제안된 동적 분류방식의 혼합형 예측기는 기존의 분류 방식 [3]과 달리 분류를 위해 별도의 테이블을 사용하지 않고 이미 존재하는 스트라이드 예측기를 이용하고, 비 스트라이드형 시퀀스의 예측을 위해 비현실적인 테이블 크기를 갖는 FCM(Finite Content Method) 예측기 [11] 대신 보다 현실적인 2-단계 예측기 [6]를 사용한다. 제안된 방식의 타당성 검증을 위해 실행구동방식(execution-driven) 시뮬레이터인 Simplescalar/Alpha 3.0 tool set [12]에 데이타 값 예측기를 추가하고, 이를 이용하여 SPECint95 벤치마크를 시뮬레이션하여 비교한다. 본 논문의 구성은 제 2장에서는 데이타 값 예측을 위한 데이타 값들의 패턴과 데이타 값 예측기를 소개하고, 3장에서는 본 논문에서 제안한 동적 분류 능력을 갖는 혼합형 데이타 값 예측기의 구조와 데이타 분류동작 등을 소개한다. 4장에서는 실험에 사용된 각 예측기의 파라미터와 벤치마크 프로그램 및 시뮬레이터를 소개하고, 5장에서는 성능분석을 위한 IPC 및 예측 정확도와 하드웨어 비용, 동적 분류에 대한 데이타 분류 유형 등의 성능분석을 하고, 마지막 6장에서는 실험에 대한 결론으로 구성되었다.

2. 데이타 값 예측 패턴 및 예측기

2.1 데이타 값 예측 시퀀스

프로세서에 의해 반복 수행되어지는 명령들은 다음과 같은 상수(constant)나 스트라이드(stride) 또는 비 스트라이드(non-stride)와 같은 데이타 값의 유형에 대한 시퀀스(data value sequence)패턴을 갖는다.

데이타 값의 유형	일반적인 데이타 값
상수(C)	5 5 5 5 5 . . .
	상수의 증감 폭이 모두 0임
스트라이드(S)	1 2 3 4 5 . . .
	상수의 증감 폭이 모두 +1임
비 스트라이드(NS)	28 12 -13 99 456 . . .
	상수의 증감 폭이 일정하지 않음

즉, 상수(Constant) 시퀀스는 0값이 증감되는 특수한 형태의 스트라이드(Stride) 시퀀스로 볼 수 있고, 스트라이드 시퀀스는 명령이 수행될 때마다 일정한 값만큼의 폭 차이로 증.감되는 패턴으로 프로그램에서 배열(array)이나 루프 인덱스(loop index)등을 참조할 때 빈번히 발생하는 패턴들이다. 또한 비 스트라이드(Non-Stride)시퀀스는 위의 상수 시퀀스 또는 스트라이드 시퀀스에 속하지 않은 모든 시퀀스를 의미한다. 이러한 데이타 값 시퀀스에서 특별히 상수 시퀀스는 같은 값이 반복되어 사용되는 패턴으로 Lipasti등은 이러한 패턴이 자주 발생함을 주목하여 최근 값 예측기를 개발하였다.[5] 또한 스트라이드와 비 스트라이드의 혼합된 형태의 반복 시퀀스(repeated sequences)로는 다음과 같은 패턴도 발생할 수 있다.

스트라이드 유형	루프내의 반복 데이타 값
반복 스트라이드(RS)	1 2 3 , 1 2 3 ...
	내부 루프의 반복 스트라이드값(1, 2, 3)
반복 비 스트라이드(RNS)	1 34 -3 76 , 1 34 -3 76 ...
	내부 루프의 반복 비 스트라이드 값(1 34 -3 76)

이와 같은 반복 시퀀스는 중첩된 루프에서 내부 루프(inner loop)가 스트라이드 또는 비 스트라이드 시퀀스를 생성하고 외부 루프(outer loop)가 이 시퀀스를 반복하는 경우가 발생한다. 이상과 같은 데이타 값 시퀀스 분류에 의해 데이타 값 예측기를 계산형 예측기(Computational Predictor)와 내용형 예측기(Context-based predictor)로 구분된다. 본 논문에서는 제안한 동적 분류 능력을 갖는 혼합형 데이타 값 예측기는 최근 값 예측(Last value prediction)과 스트라이드 예측(Stride value prediction), 데이타 값들의 문맥을 근간으로 한 데이타 값 예측(context-based prediction)및 혼합된 예측(Hybrid value prediction)을 포함하고 있는데, 최근 값 예측은 한 명령에서 가장 최근에 생성된 데이타 값의 결과 값을 예측하는 것이고, 스트라이드 예측은 가장 최근에 생성된 두 개의 데이타 값에 대한 차(difference)를 계산하여 계산된 결과 값을 스트라이드로 하여 다음 값을 예측한다. 또한 데이타 값의 문맥을 근간으로 한 예측기는 이전의 여러 데이타 값들을 관찰하여 데이타 값들 사이의 반복되어지는 패턴들을 이용하여 다음 값을 예측방법으로 FCM(Finite Content Mechanism) [11]과 2-단계 데이타 값 예측기 [6]가 있다.

2.2 최근 값 예측기

Lipasti에 의해 개발된 최근 값 예측기[5]는 프로그램의 실행동안 대부분의 명령들의 값의 변화가 적음을 주목하고 레지스터(register) 값을 변경하는 명령들의 재사용되는 데이타 값의 국한성(locality)을 실험하여 이를 근거로 최근 데이타 값 예측기(last value predictor)를 제안하였다. 최근 값 예측기는 명령의 주소에 의해 인덱스되는 테이블에 이 명령의 최근 수행된 결과를 저장하고 다음에 이 명령 패치 시 저장된 최근 값으로 결과 값을 예측한다. 이러한 최근 값 예측기는 최초의 본격적인 데이타 값 예측기로써 의의가 있고, 예측기의 구성이 단순하다는 장점이 있으나, 상수 시퀀스를 갖는 명령들만을 예측할 수 있다는 단점이 있다.

2.3 스트라이드 예측기

Gabbay [1]와 Wang [6]등은 상태, 데이타 값, 스트라이드등이 저장된 VHT(Value History Table)로 구성된 스트라이드 예측기를 설계하였다. 스트라이드 데이타 값 예측기는 한 명령어의 연속적인 인스턴스(instances)의 결과들이 변화하는 스트라이드를 모니터(monitor)하여 이들 결과 값들이 일정한 폭(stride)으로 변하면 그 명령어의 장래 인스턴스의 결과의 예측이 용이하다. 순환 명령(loop induction variables)이나 일정하게 배열(arrays)의 값이 증감하는 프로그램과 데이타를 캐시(cache)로 선행인출(prefetch)하기 위해 주소를 생성하는데 매우 잘 수행된다. 스트라이드를 근거로 한 예측기에서는 한 명령에 대해 최근에 수행된 두 개의 결과 값의 차인 스트라이드를 구하고 다음에 이 명령의 수행 시 이전의 결과 값에 스트라이드를 더하여 예측한다. [

그림 2]는 간단한 스트라이드를 근거로 한 데이타 값 예측기의 구조를 보여준다.

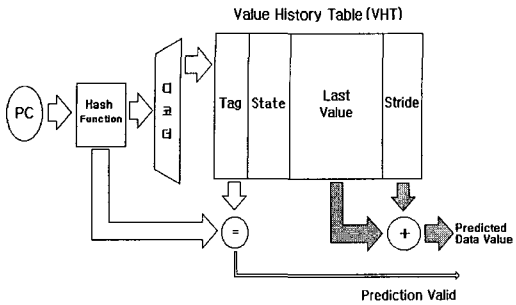


그림 2 스트라이드 데이타 값 예측기 구조

또한 스트라이드를 근거로 한 예측기 에서 다음 스트라이드를 검출하기 위한 기본적인 상태변환을 보여준다. 이러한 상태변화는 스트라이드 시퀀스를 검출하기 위한 목적으로 명령어를 만나게되는 첫 번째 시간에는 어떤 예상도 일어나지 않지만, 그 명령어가 결과를 생성할 때 VHT에 엔트리가 할당되고 다음의 동작이 발생한다. 결과는 그 엔트리의 value 필드에 저장되고 그 엔트리의 state 필드는 초기상태(initial state)으로 설정된다. "Init" 상태에 있는 동안에 동일한 명령의 다른 인스턴스를 만날 때에는 예상은 일어나지 않고, 스트라이드 시퀀스의 처음 데이타 값의 결과인 첫 번째 데이타 S_{D1}이 생성되고, 다음 예측에 사용될 스트라이드 S_{s1}은 S_{D1} - [VHT엔트리 내의 이전 데이타 값]으로 계산되며 S_{D1}과 S_{s1}은 value와 stride 필드에 저장된다. 이때 state 필드는 천이상태(Transient state)로 설정되고, "Transient" 상태에 있는 동안에 동일한 명령어의 다른 인스턴스를 만나면 예상은 일어나지 않으며, 그 인스턴스의 결과인 두 번째 데이타 S_{D2}를 생성할 때 다음 동작이 발생한다. 따라서 다음 스트라이드 S_{s2}는 S_{D2} - [VHT엔트리 내의 이전 데이타 값]으로 계산되고, 두 번째 인스턴트의 결과 데이타 S_{D2}는 value 필드로 들어가고 S_{s2}가 이전의 스트라이드 S_{s1}과 동일하면 state 필드는 안정상태(Steady state)로 설정이 되지만 그렇지 않으면 S_{s2}가 stride 필드로 들어가고 여전히 state 필드는 "Transient" 상태로 설정된다. 만약 S_{s1}과 S_{s2}가 동일하다면 "Steady" 상태에서 동안 value와 stride 필드를 함께 추가하여 다음의 예상이 만들어진다. 이러한 간단한 3-상태 구성은 대부분의 스트라이드를 검출할 수 있다. 이러한 스트라이드 예측기는 상수와 스트라이드 시퀀스를 갖는 명령의 데이타 값을 예측할 수가 있어서 최근 값 예측기 보다 예측 정확도가 높지만 반복형 비 스트라이드 시퀀스와 같은 규칙적인 패턴을 결과 값으로 생성하는 명령들에 대해서는 예측하지 못한다는 단점이 있다.

2.4 2-단계 데이타 값 예측기

Wang등은 기존의 분기예측(branch prediction)에서 사용된 2-단계 적응 분기예측(2-level adaptive branch prediction)방식[13]을 데이타 값 예측기에 적용한 2-단계 데이타 값 예측기(2-level data value predictor)를 개발하였다 [6]. 2-단계 데이타 값 예측기는 VHT와 PHT(Pattern History Table) 2개의 테이블로 구성된다. VHT테이블은 예측되는 명령의 주소에 의해 인덱스되며 최근에 사용된 n개의 데이타 값과 마지막에 수행된 p개의 명령 결과 값을 가리키는 인덱스의 비트 패턴이 저장된다. 인덱스의 비트 패턴은 PHT를 인덱스하는데 사용된다. PHT의 각 엔트리는 VHT에 저장된 n 데이타 값과 일치하는 증감 카운터들로 구성된다. lookup 동작은 PC에 의해 VHT가 인덱스되고, 히트되면 VHP에 의해 PHT가 인덱스 된다. PHT의 카운터 중에서 최대 값을 선택하고, 이 값이 임계값(Threshold)보다 크면 이 카운터의 위치에 대응되는 VHT의 데이타 값이 예측되고, 임계값 보다 작은 경우에는 예측을 하지 않는다.

2.5 혼합형 데이타 값 예측기

앞에서 언급한 각 예측기는 각 예측기에 적합한 특정 데이타 값의 시퀀스에서는 좋은 결과를 보이지만 다른 형태의 데이타 값 시퀀스에 대해서는 예측률이 떨어지는 문제점이 있다. 따라서 이들 예측기의 장점을 결합한 혼합형 예측기들이 개발되었다. Wang등은 스트라이드 예측기와 2-단계 데이타 값 예측기를 결합한 혼합형 예측기를 제안하였다.[6] [그림 3]은 혼합형 데이타 값 예측기의 구조를 보여준다. 이러한 혼합형 데이타 값 예측기는 일정한 값으로 증감하는 명령을 예측하는 스트라이드 예측기와 2-단계 데이타 값 예측기를 결합한 것으로 확신 카운터(Confidence counter)의 값에 의한 예측 방법이다. 명령어가 두 개의 예측기에서 모두 엔트리를 갖고 있다면, 카운터의 값이 높은 예측기의 예측 값을 선택한다. 따라서 혼합형 예측기는 스트라이드의 특성을 갖는 명령과 다양한 패턴을 갖는 명령어를 모두 예측하게 됨으로 높은 예측 정확도를 얻을 수 있지만 하나의 명령어가 두 개의 테이블 엔트리에 중복 저장되어 높은 하드웨어의 비용을 필요로 한다는 단점을 갖고 있다. 그러나 위에서 제안한 많은 방법들이 소개되었지만 만족할 만한 예측 정확도를 갖지 못하고있는 실정이다.

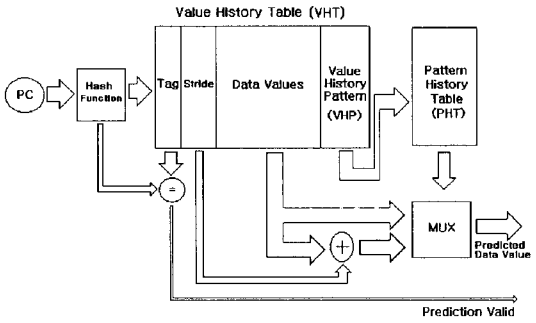


그림 3 혼합형 데이타 값 예측기 구조

3. 동적 분류 능력을 갖는 혼합형 데이타 값 예측기

본 논문에서 제안되는 Rychlik등은 동적 분류(dynamic classification)를 위하여 별도의 분류기능을 갖는 테이블인 CT(Classified instruction Table)와 UT(Unclassified instruction Table)에 최근에 실행된 3개의 값을 저장하고 이들의 각 스트라이드 차를 이용하여 최근 값(Popular Last Value), 스트라이드(Stride+), fcm 예측기로 분류하고 예측하는 혼합형 데이타 값 예측기를 제안하였다.[3] Rychlik등의 동적 분류는 새로운 명령을 만나면 바로 예측기를 지정하는 것이 아니라 일정한 학습기간(learning period)을 갖는다. 이러한 학습단계에서 어느 한 예측기가 해당 명령에 잘 맞는지를 선택한다. 명령이 한 예측기로 분류되었다면 해당 예측기의 테이블을 할당하여 예측을 시도하고, 실제 값으로 갱신하고 내부적으로 예측 확신정도를 관리한다. prediction confidence가 충분히 떨어졌으면 명령이 해당 예측기에 대해서 더 이상 예측할 수 없는 것으로 간주하여 해당 예측기로부터 퇴거(evict)하고 분류과정을 다시 하여 다른 예측기로 다시 지정하거나 어떤 예측기로도 예측할 수 없는 것으로 분류된다. fcm 예측기가 가장 일반적인 형태의 예측기이므로 fcm 예측기로부터 퇴거된 명령은 예측 불가능한 것으로 분류한다. 따라서 이러한 분류의 개념적 동작은 CT(Classified Instruction Table)와 UT(Unclassified Instruction Table)를 이용하였는데, CT는 명령 주소에 의해 인덱스 되고 각 분류된 명령의 pID(prediction ID)가 저장된다. pID는 "Don't Predict," "Use Popular Last Value," "Use Stride+," "Use FCM" 중 하나의 값을 갖는다. UT는 각 명령에 대하여 3개의 최근 데이타 값을 저장한다. history value는 학습기간동안에 채워지고 각 명령에 대한 예측기를 선택하기 위하여 분류 메커니즘에 의해 사용된다. CT와 UT테이블은 "Predict," "Update," "Classify," "Evict"의 4가지 오퍼레이션으로 나누어진다. 예측 오퍼레이션은 명령이 인출될 때 PC에 의하여 CT를 참조한다. pID가 "Don't Predict"이거나 CT 미스(miss)이면 예측을 시도하지 않는다. 이러한 경우가 아니면 해당 예측기를 참조하여 predicted value와 prediction confidence를 구한다. prediction confidence가 높으면 예측을 하고 낮으면 예측을 하지 않는다. 만약 prediction confidence가 0이면 "Evict" 오퍼레이션이 수행된다. "Update" 오퍼레이션은 명령이 종료되면 pID를 결정하기 위해 CT가 다시 참조되어 predicted value를 갱신하기 위한 예측기를 선택한다. CT가 미스이면 명령은 분류되지 않고 UT가 참조된다. UT에서 미스가 발생한 경우에는 UT에 한 엔트리를 할당하고, 성공이면 생성된 값으로 갱신된다. UT에 모든 history value가 채워졌으면 분류 오퍼레이션이 수행된다. "Classify"는 UT에 있는 명령에 대하여 history value를 조사하고 값들 사이에 차(difference)를 구하여 pID를 구한다. 모든 차가 0이면 Popular Last Value, 모든 차가 같으면 Stride+, 이외의 경우에는 fcm으로 지정된다. 마지막으로 CT의 엔트리에 pID를 저장하고 UT에서 제거한다. "Evict" 오퍼레이션은 예측기가 0값의 confidence를 돌려주면 해당 명령이 예측기로부터 제거되고 만약 pID가 FCM이면 "Don't Predictor"로 설정된다. fcm이 아니면 CT의 엔트리가 제거되고 다음에 명령이 수행될 때 새로운 명령으로 재분류된다. 그러나 본 논문에서 제안한 예측기는 Rychlik등과 같이 UT나 CT와 같은 테이블을 이용하지 않고 기존의 스트라이드 테이블을 이용하여 동적으로 테이블을 분류한다.

3.1 예측기 구조 및 동작

[그림 4]는 본 논문에서 제안한 동적 분류 능력을 갖는 혼합형 데이타 값 예측기의 구성도를 나타낸 그림으로써, LVT(Last Value Table), SVT(Stride Value Table), 2VT(2-level Value Table)의 3개의 테이블로 구성하고 각 예측기의 분류동작은 SVT에 의해 분류 예측된다. 이들 각 예측 테이블은 모두 명령의 주소(PC)에 의해 인덱스 되는데, 명령의 처음 데이타 값의 예측은 예측분류 테이블인 SVT의 상태필드를 초기화하고 초기 데이타 값과 스트라이드를 계산하여 저장, pID필드의 초기 값으로 "stride"를 설정한다. 다음 동일한 명령의 주소에 의해 인덱스 되어지면 처음과는 달리 분류예측 테이블인 SVT의 pID필드를 참조하여 DeMUX(demultiplex)에 의하여 각 예측 테이블을 지정

하고, 데이타 값 유형에 따른 분류를 시도한다. 즉, pID 필드의 값을 "Stride," "Last," "2level," "Unknown"와 같은 분류정보를 갖고 데이타 값에 따른 예측을 수행하게 되는데, 이러한 동적 분류 동작은 본 논문의 3.2절에서 자세히 소개한다.

LVT는 상수 시퀀스의 반복 패턴을 갖는 명령들의 값을 예측하기 위한 테이블로 가장 최근의 결과 값필드와 확신 카운터필드로 구성된다. 예측은 확신 카운터 값이 임계값(threshold) 이상이면 저장된 결과 값으로 예측을 한다. 명령의 실행 후 예측의 올바른 것으로 판명되면 확신 카운터를 증가시키고, 그렇지 않으면 감소한다.

SVT는 새로운 명령에 대해 어느 예측기로 예측할 것인가를 결정하여 분류하여 분류된 결과를 pID 필드에 저장하는 기능과 스트라이드 시퀀스의 반복 패턴을 갖는 명령들의 결과 값을 예측하는 기능이 있다. SVT는 상태, 스트라이드, 결과 값, 확신 카운터 필드, pID필드로 구성된다. 상태 필드는 SVT의 현재 동작 상태를 나타내며 [그림 5]와 같이 "Init," "Transient," "Steady" 상태를 갖는다. 스트라이드는 연속된 이전의 두 결과 값의 차이이며, 결과 값은 최근의 수행된 결과 값을 나타낸다. 인출되는 각 명령은 데이타 값 예측을 위해 어느 테이블을 사용하여 예측할 것인가를 결정하는 동적 분류 정보를 위해 SVT의 pID필드의 값에 따라 분류된다. pID는 2 비트 필드로 각각 "Unknown," "Last," "Stride," "2Level"의 4가지 정보를 지니고 있다. SVT의 예측은 확신 카운터(confidence counter)의 값이 임계값 이상이면 결과 값에 스트라이드를 더한 값으로 한다.

2VT는 2-단계 예측기로 VHT와 PHT로 구성된다. 첫 번째 단계인 VHT는 과거의 실행 결과를 기록하는 테이블로 최근에 실행된 4개의 데이타 값을 기록하는 데이타 값 필드와 명령의 마지막 p 개의 실행을 추적하기 위한 2*p 비트의 패턴으로 표시되는 과거 패턴(VHT) 필드로 구성된다. (이때 p 개의 비트는 과거의 발생한 값과 일치하는 데이타 값 필드의 위치로 표시된다.) 두 번째 단계인 PHT는 VHT의 각 과거 패턴 비트에 대응되는 4개의 확신 카운터로 구성된다. 이 예측기의 동작은 먼저 예측되는 명령의 주소에 의해 인덱스되는 VHT의 과거 패턴 필드를 참조하여 PHT의 엔트리를 선택한다. 선택된 PHT의 엔트리 중에서 가장 큰 값을 갖는 카운터의 값이 특정 임계값을 초과한 경우 이 카운터와 대응되는 VHT의 데이타 값 필드에 저장된 값으로 예측한다. 만약 한계 값을 넘지 않은 경우 예측을 하지 않는다. 예측 후에 실행된 결과 값에 따라 VHT의 과거 패턴은 2 비트 왼쪽으로 시프트 된다. 또한 예측이 맞으면 선택된 PHT의 카운터는 증가되고 다른 카운터는 감소된다.

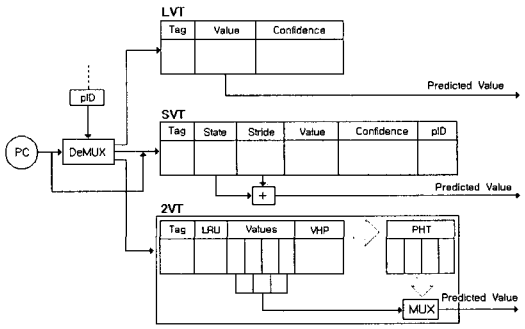


그림 4 동적 데이타 값 예측기의 구조

3.2 동적 분류동작

본 논문에서 제안한 동적 분류 능력을 갖는 혼합형 예측기에서의 동적 분류의 목적은 데이타 값 예측을 위해 어느 한 예측기를 선택하고 갱신(update)을 위해 올바른 값으로 갱신할 예측기를 선택하는 것이다.

SVT를 이용한 예측기의 분류는 [그림 5]와 같이 상태천이를 하는데, 먼저 새로운 명령이 들어오면 상태 필드의 값이 "Init" 상태가 되고 결과 값으로 value1을 저장한다. 다음에 이 명령을 다시 수행되면 "Init" 상태에서 "Transient" 상태로 천이 되고 현재 결과 값 value2와 이전 값 value1의 차인 stride1을 계산하여 스트라이드 필드에 저장한다. 다음에 수행 시에는 "Steady" 상태로 천이 되고, 현재 값 value3과 이전 값 value2로부터 stride2를 계산한다. 먼저 stride1과 stride2를 서로 비교하여 두 값이 서로 동일하고 스트라이드 값이 0인가를 조사하여 만족하면 pID를 "Last"로 설정하고 최근 값 예측기(LVT)로 예측한다. 또한 비교한 스트라이드가 만일 0이 아니면 두 개의 스트라이드가 같은가를 조사하여 같으면 pID를 "Stride"로 설정하고 계속하여 스트라이드 예측기(SVT)로 예측한다. 그리고 스트라이드가 서로 같지 않으면 pID를 "2level"로 설정하고 이후 2-단계 예측기(2VT)로 분류하여 예측을 시도한다. 그러나 최근 값 예측기와 2-단계 예측기에서 확신 카운터 값이 0이면 각 테이블에서 퇴거(Evict)동작을 수행하게되고 SVT의 pID를 "Unknown"으로 설정한다. 퇴거 동작에서는 마치 새로운 명령처럼 SVT를 이용하여 상태필드가 "Init" 상태에서부터 명령을 재분

류하여 처음부터 다시 분류동작을 계속하게 된다.

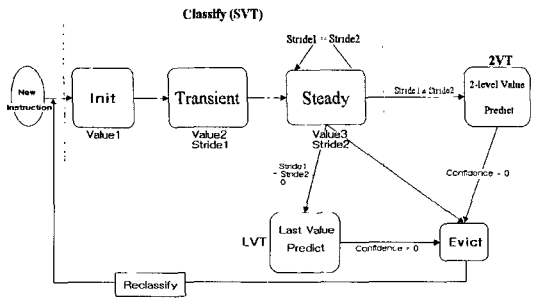


그림 5 동적 데이타 값 예측기의 분류 동작

4. 실험 방법

4장에서는 성능을 평가하기 위하여, 실험에 사용된 기본 시스템 구조의 파라미터 및 기존의 혼합형 데이타 값 예측기와 동적 분류 능력을 갖는 혼합형 데이타 값 예측기에 대한 각 테이블의 설정 값들을 소개하고, 본 논문에서 사용된 벤치마크 프로그램과 실험에 사용된 시뮬레이터인 Simplescalar Tool Set에 대하여 소개한다.

4.1 실험에 사용된 데이타 값 예측기 모델

제안된 동적 분류 능력을 갖는 혼합형 데이타 값 예측기와 기존의 혼합형 데이타 값 예측기를 비교 측정하기 위하여 4~16 이슈의 명령의 이슈 크기(issue-width)에 대한 3가지 시스템의 기본 구조를 사용한다. [표 1]에서는 기본 구조에서 사용된 파라미터와 같이 사이클(cycle)당 4~16 이슈를 기본 값으로 하였고, 분기 예측(Branch predict)을 위해서 2-way의 2K 엔트리를 갖는 BTB(Branch Target Buffer)와 정확한 분기예측을 가정한 Combine 분기 예측기를 사용하였다. 그리고 4K 크기를 갖는 명령 캐시(instruction cache)와 데이타 캐시(data cache)를 실험에 사용하였다.

[표 2]는 기존의 혼합형 데이타 값 예측기와 동적 분류 능력을 갖는 혼합형 데이타 값 예측기의 파라미터로써 혼합형 데이타 값 예측기는 8K 엔트리의 VHT과 4개의 히스토리를 갖고, 8K 엔트리로 구성된 PHT로 구성된 총 1.6M비트의 테이블 크기로 구성되고, 제안된 동적 분류 능력을 갖는 혼합형 데이타 값 예측기는 4K의 LVT와 4K의 SVT, 그리고 2K 엔트리로 구성된 2VT로 4개의 히스토리로 구성된 총 0.9M비트의 테이블 크기를 갖는다. 이러한 각 테이블들의 크기는 예측된 명령들의 동적 명령 비율로부터 구동된다.

표 1 실험에 사용된 기본 시스템 구조의 Configurations

	파라미터 (Parameter)	값(Value)	의미(Comments)
Processor core	RUU size	256	instruction windows
	LSQ size	128	Load-Store Queue
	Fetch width	4-16 instr/cycle	4-16 issue baseline
	Decode width	4-16 instr/cycle	4-16 issue baseline
	Issue width	4-16 instr/cycle	4-16 issue baseline
	Commit width	4-16 instr/cycle	4-16 issue baseline
	Functional Unit	8 int ALU(1) 2 int MUL(3) 4 fp ALU(1) 2 fp MUL(4)	()은 Latency 임
Branch Predictor	BTB	2K, 2-way	Branch target buffer
	Combine Branch Predictor	2-level 16K, 14bit history bimodal 16K	2-level + bimodal
	RAS	32	return addr. stack
Cache	L1 D-cache	4K, 2-way, 32 byte blocks	Data cache
	L1 I-cache	1-cycle latancy 4K direct map,	Instruction cache
	L2 cache	128 byte block 4-way,	secondary cache

표 2 실험에 사용된 데이타 값 예측기의 Configurations

데이타 값 예측기	테이블 구성	테이블 크기
Two-level data Value predictor	8K VHT(Value History Table) 8K PHT(Pattern History Table) 4 history data values	1,310,720 65,536
Hybrid Value predictor	8K VHT(Value History Table) 8K PHT(Pattern History Table) 4 history data values	1,589,248 65,536
Hybrid Value Predictor with Dynamic Classification Capability	4K LVT(Last Value Table) 4K SVT(Stride Value Table) 2K 2VT(2-level Value Table)	212,991 360,448 344,064

표 3 SPECint95 벤치마크 프로그램의 입력자료

벤치마크 프로그램	입력자료	명령의 크기	기본 IPC		
			4-이슈	8-이슈	16-이슈
compress	10000 e 2231	35,726,662	2.1091	2.6898	2.6711
gcc	jump.i	39,980,547	1.5303	1.8231	1.9188
ijpeg	tinyrose.ppm	72,608,027	2.4839	3.1860	3.3690
li	queen6.lsp	41,732,243	1.7259	2.2214	2.3096
m88ksim	new.100	95,476,020	2.3579	3.0578	3.2831
perl	scrabbl.in	40,450,590	1.9842	2.6992	2.7978
vortex	persons.250	66,130,658	2.0742	3.1136	3.6511
평 균		391,104,747	2.0379	2.6844	2.8572

4.2 벤치마크 프로그램

시뮬레이션을 수행하기 위해 사용되어진 벤치마크 프로그램은 [표 3]과 같이 시뮬레이션에 사용된 입력자료를 생성하기 위하여 GNU's gcc2.7.2를 UNIX 플랫폼상에 컴파일된 SPECint95 벤치마크 프로그램을 사용하였고, 수행 명령 결과들의 평균은 레지스터 기록(register-writing) 명령들이다.

4.3 시뮬레이터

본 연구에서 사용된 시뮬레이터는 데이타 값 예측기의 최적 설계 파라미터의 추출과 검증을 위해서 Simplescalar 아키텍쳐의 시뮬레이터인 Simplescalar Tool set[5]에 데이타 값 예측기 모델을 삽입하여 실험하였다. 또한, 시뮬레이터 입력의 벤치마크 프로그램(benchmark program)은 C 프로그램이나, "f2c 변환기"에 의해 C로 변환되는 FORTRAN프로그램으로 Simplescalar 아키텍쳐의 C 컴파일러인 "Gcc C 컴파일러"를 통해 어셈블리 언어를 생성한다. 생성된 Simplescalar 어셈블리 프로그램은 "GAS 어셈블러"와 "GLD 로더"를 거쳐 시뮬레이션 실행파일을 생성하며, 이 실행 파일이 Simplescalar 시뮬레이터에 입력으로 사용되어 시뮬레이션을 수행하고 시뮬레이션 결과를 출력하였다.

5. 성능분석

5장에서는 본 논문에서 제안한 동적 분류 능력을 갖는 혼합형 데이타 값 예측기를 사용하여 데이타 값 예측의 성능과, 기존의 혼합형 데이타 값 예측기와 동적 분류 능력을 갖는 혼합형 데이타 값 예측기의 예측 정확도 및 하드웨어의 비용을 실험 분석한다. 또한 동적 분류 능력을 갖는 혼합형 데이타 값 예측기에 대하여는 동적 분류기에 의한 데이타 값들을 각 유형별로 분류하여 이들 분류된 예측 데이타들의 특성들을 비교. 분석하여 각 벤치마크 프로그램에서 예측되어지는 데이타 값들의 유형들을 살펴보았다.

5.1 하드웨어의 비용효과

[그림 6]은 기존의 혼합형 데이타 값 예측기와 동적 분류 능력을 갖는 혼합형 데이타 값 예측기의 하드웨어 비용을 실험한 결과로써 혼합형 예측기는 8K의 VHT엔트리와 8K의 PHT로 구성되고 히스토리의 크기는 4로 구성되었다. 그리고 본 논문에서 제안한 동적 분류 능력을 갖는 혼합형 데이타 값 예측기는 4K의 LVT와 4K의 SVT, 2K의 2VT와 4개의 히스토리를 갖는다. 따라서 전체적인 테이블의 크기는 혼합형 데이타 값 예측기의 크기는 1.6M의 크기를 갖고, 동적 분류 능력을 갖는 데이타 값 예측기의 크기는 총 0.9M의 크기로 구성되어 약 45%의 하드웨어 비용의 감소효과를 나타내었다.

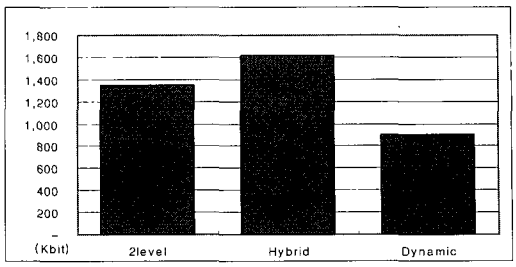
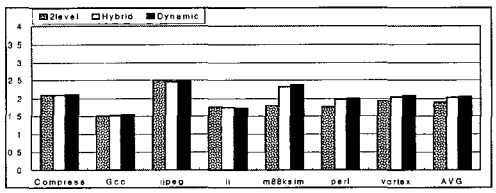


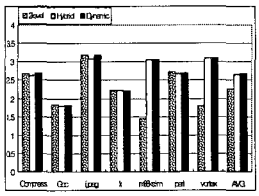
그림 6 혼합형 예측기의 하드웨어 비용

5.2 IPC 효과

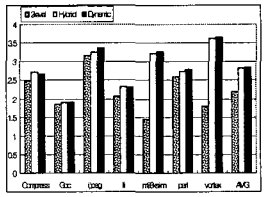
[그림 7]은 실험에 사용된 기본 구조에서 4~16 이슈 IPC에 대한 벤치마크 프로그램별 성능측정의 결과들을 보여준다. [그림 7]에서 보이는 것과 같이 4-이슈 (a)에서 평균 IPC가 기존의 혼합형 데이타 값 예측기 보다 0.98%정도 동적 분류 능력을 갖는 혼합형 데이타 값 예측기에서 성능향상을 보이고, 8-이슈 (b) 에서는 0.74%, 16-이슈 (c) 에서는 기존의 혼합형 데이타 값 예측기보다 0.71%를 나타내고 있다. 그러나 2-단계 예측기에서는 4-이슈에서 4.43%, 8-이슈에서 15%, 16-이슈에서 21.5%의 성능향상을 보였다. 따라서 IPC의 실험 결과 기존의 2-단계 예측기 및 혼합형 데이타 값 예측기와 본 논문에서 제안한 동적 분류 능력을 갖는 데이타 값 예측기와의 IPC가 평균 혼합형 예측기에서 0.8%, 2-단계 예측기에서 14% 정도의 IPC결과를 보였다.



(a) 4-이슈



(b) 8-이슈



(c) 16-이슈

그림 7 예측기별 IPC 비교

[그림 8]은 4~16이슈별 데이타 값 예측기의 IPC를 실험한 것으로 역시 기존의 데이타 값 예측기 보다는 동적 분류능력을 갖는 혼합형 예측기의 IPC성능이 2-단계 데이타 값 예측기에서 평균 0.36, 혼합형 데이타 값 예측기 에서 평균 0.018정도로 측정되어 기존의 예측 방법보다 동적 분류 능력을 갖는 혼합형 데이타 값 예측기에서 IPC가 다소 높게 나타내 보이고 있다.

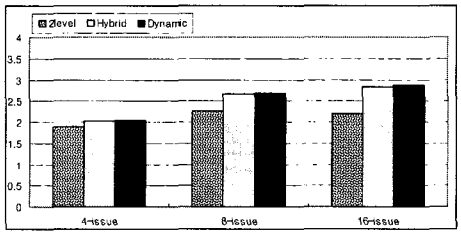


그림 8 기본 구조에서 명령 이슈별 성능비교

5.3 데이타 값 예측에 대한 예측 정확도

[그림 9]는 명령어에 대한 데이타 값을 예측하는데 있어서 예측을 시도한 값들 가운데 예측 실패로 인한 예측 패널티를 보여주고 있는데, 2-단계 예측기에서 11%, 혼합형 예측기에서 24%, 제안된 동적 예측기에서 평균 8%의 예측 실패시 발생하는 예측 패널티로서 기존의 예측기에 비교하여 동적기에서의 예측실패가 적은 것으로 나타났다. 따라서 본 논문에서 제안한 예측기의 IPC는 비슷하게 측정 되었지만, 예측 패널티의 감소(3% ~ 16%)로 인한 예측 정확도가 높게 나타난다. IPC를 실험한 결과 각 벤치마크 프로그램에서 성능 향상이 기존의 예측기와 비슷한 결과를 보였습니다. 또한 8-이슈에서 li와 16-이슈에서 li가 다소 성능이 떨어지는 경향을 보이고 있는데, 이는 기존의 혼합형 예측기에서 처리되지 않는 즉, 본 논문에서 제안한 동적 능력을 갖는 혼합형 데이타 값 예측기에서 데이타 패턴에 따른 테이블 분류 동작시 "Unknown"으로 처리된 명령은 재분류되어 또다시 예측을 시도하게 되는데, 재분류 시 발생되는

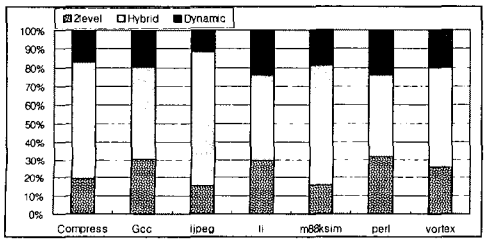


그림 9 데이타 값 예측기의 예측 패널티

지연(delay)으로 인하여 성능이 다소 낮은 결과를 초래하게 된다. 그러나 전체적으로 성능에는 커다란 영향을 주지 않으면서도 하드웨어의 비용이 낮으면서도 예측 정확도는 평균 16%의 높은 정확도를 보였다.

[그림 10]은 데이타 값에 대하여 올바르게 예측되었는지를 실험한 것으로, 기존의 혼합형 데이타 값 예측기와 동적 분류 능력을 갖는 혼합형 데이타 값 예측기에 대한 데이타 값의 예측정확도를 실험한 결과이다. 실험 내용에서 예측 율(Prediction rate)은 데이타 값 예측을 수행하는 모든 명령들에 대하여 올바르게 예측된 결과들의 비율을 의미하는 것이고, 예측 정확도(prediction accuracy)는 실제 예측 시도된 데이타 값들 중에서 올바르게 예측된 결과에 대한 비율을 의미한다. 따라서, 본 논문에서는 데이타 값 예측기의 성능측정을 위하여 예측 정확도를 기초로 하여 성능을 연구하였다. 이러한 예측정확도를 연구하여 성능을 비교 분석한 이유는 실제로 컴퓨터의 성능에 영향을 미치는 것은 데이타 값들에 대한 예측 정확도이지 예측율이 아니기 때문이다. [그림 10]은 기존의 혼합형 데이타 값 예측기와 제안된 동적 분류 능력을 갖는 혼합형 데이타 값 예측기에 대한 예측 정확도를 보여주는데, 여기에서 각 데이타 값 예측기의 평균 예측 정확도는 기존의 혼합형 예측기에서 67%, 동적 분류 능력을 갖는 혼합형 데이타 값 예측기에서 83%로 측정되어, 기존의 혼합형 데이타 값 예측기 보다는 동적 분류 능력을 갖는 혼합형 데이타 값 예측기에서 평균 16%의 높은 예측 정확도의 결과를 보여 성능이 향상되어짐을 보여주었다. 특히 ijpeg인 경우에는 기존의 혼합형 데이타 값 예측기에 비교하여 동적 분류 능력을 갖는 혼합형 데이타 값 예측기가 전체명령 가운데 예측 정확도가 무려 34%의 높은 예측정확도를 보였다.

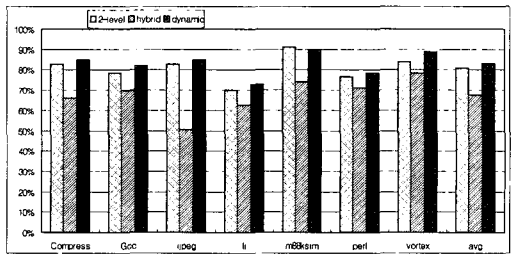


그림 10 데이타 값 예측기의 예측 정확도

5.4 동적 분류의 데이타 예측유형 분포

[그림 11]은 제안된 동적 분류 능력을 갖는 혼합형

데이타 값 예측기를 사용한 동적 명령 스트림(stream)에서 데이타 값 예측유형의 분포된 결과를 보여준다. 동적 분류 능력을 갖는 혼합형 데이타 값 예측기에서 데이타 값의 유형별 분류형태를 실험한 결과 최근 값 예측 유형이 평균 29%, 스트라이드 형태의 데이타 값 예측유형에서 1.92%, 2-단계 데이타 값 유형에서 56.1%, 그리고 "Unknown"인 경우가 16.3%로 측정되었다. 이러한 결과는 2-단계 데이타 값 예측의 유형이 기타 다른 데이타 값 예측유형 보다 많은 예측을 시도하고 있음을 보이는 것이다. 특히 compress와 ijpeg에서 2-단계 데이타 값 유형이 각각 68.8%와 68%로 측정되었다. 따라서 이러한 결과는 2-단계 데이타 값으로 분류된 명령들이 최근 데이타 값 예측의 유형과 스트라이드 데이

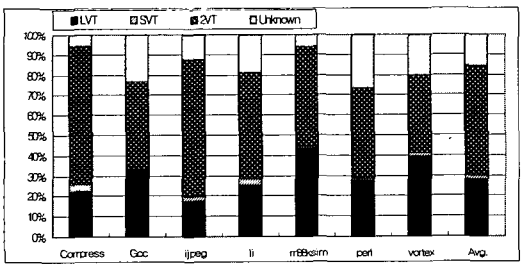


그림 11 동적 데이타 값 예측기에서 데이타 값의 예측 유형별 분포

터 값 예측의 유형이 모두 실패되었을 때 2-단계 데이타 값 예측으로 분류되어, 최근 값이나 스트라이드 데이타 값과 같은 특수한 유형의 예측분류 형태보다는 다양한 유형(non-stride sequence)의 2-단계 데이타 값들이 프로그램에서 많이 사용되어 제안한 동적 분류 능력을 갖는 혼합형 데이타 값 예측기에서 보다 많은 데이타 값들을 예측할 수 있음을 보여주는 것이다.

6. 결 론

본 논문에서는 다중 이슈(wide-issue) 슈퍼스칼라 프로세서(superscalar processor) 구조에서 다중 데이타 값 예측에 대한 수행이슈를 살펴보았다. 이 것은 다중 데이타 값 예측 테이블이 제공된 것처럼 명령 패치(Instruction fetch) 단계로부터 동적으로 예측을 하게되어 각 데이타 값 예측 테이블에 접근된 수를 상당히 감소시킬 수 있다. 위의 실험은 Simplescalar Tool Set을 사용, 실험결과에 대한 성능 얻었는데, 본 논문에서 제안한 동적 분류 능력을 갖는 혼합형 데이타 값 예측기는 SPECint95 벤치마크 프로그램의 기본 설정된 구조에서 데이타 값에 대한 IPC와 예측 정확도 및 하드웨어의 비용과 데이타 값 유형에 따른 성능실험을 한 결과, IPC는 기존의 혼합형 예측기와 비슷한 결과를 보였으나, 기존의 혼합형 데이타 값 예측기에 비해 평균 16%의 예측 정확도를 얻었고, 두 개의 혼합형 예측기에 대한 하드웨어의 비용효과를 실험한 결과 동적 분류 능력을 갖는 혼합형 예측기가 45%의 하드웨어 비용 감소효과를 얻었다. 또한 동적 분류 능력을 갖는 혼합형 데이타 값 예측기에서 데이타 값의 유형별 분포를 실험한 결과 특수한 데이타 시퀀스를 갖는 값들보다는 다양한 비 스트라이드 유형의 데이타 값들이 프로그램에서 최대 68%이상을 차지하고있어 2-단계 데이타 값 예측에 대한 연구가 필요함을 알았다. 본 논문의 데이타 값 예측 테이블에서 높은 대역폭(bandwidth)을 가진 전형적인 데이타 예측 구조를 사용하여 높은 성능을 얻었는데, 실험결과 레지스터 기록명령들은 오직 데이타 값 예측을 필요로 하는 명령들임을 확인할 수 있었는데, 이것은 불필요한 데이타 값 예측을 감소시킬 수가 있음을 보였다. 결과적으로 본 논문에서 제시된 동적 분류 능력을 갖는 혼합형 데이타 값 예측기가 다중 이슈 슈퍼스칼라 프로세서에서 다중 데이타 값 예측에 효과적임을 입증하였다.

참 고 문 헌

[1] F.Gabbay and A.Mendeson, "Can Program Profiling Support Value Prediction?," *Proc. of 30th Annual ACM/IEEE International Symposium on Microarchitecture*, Dec. 1997.

[2] Sang-Jeong Lee, Yuan Wang, and Pen-Chung Yew, "Decoupled Value Prediction on Trace Processors," *IEEE 6th international Symposium on high performance computer architecture*, 2000.

[3] B. Rychlik, J.Faistl, B.Krug, A.Kurland, J.Jung, Miroslav, N.Velev, and J.Shen, "Efficient and Accurate Value Prediction Using Dynamic Classification," *Technical Report of Micro-architecture Research Team in Dept. of Electrical and Computer Engineering*, Carnegie Mellon Univ., 1998.

[4] B.Rychlik, J.Faistl, B.Krug, and J.Shen, "Efficacy and Performance Impact of Value Prediction," *Parallel Architectures and Compilation Techniques*, Paris, Oct. 1998.

[5] M.H.Lipasti and J.P.Shen, "Exceeding the Limit via Value Prediction," *Proc. of 29th Annual ACM/IEEE International Symposium on Microarchitecture*, Dec. 1996.

[6] Kai Wang, Manoj Franklin, "Highly Accurate data value Predictions using hybird predictor," *Proc. of*

30th Annual ACM/IEEE International Symposium on Micro architecture, Dec. 1997.
[7] B.Calder, G.Reinman and D.Tullsen, "Selective Value Prediction," *Proc. of the 26th International Symposium on Computer Architecture(ISCA-26)*, May. 1999.
[8] F.Gabbay and A.Mendelson, "Speculative Execution Based on Value Prediction," *EE Department TR 1080, Technion-Israel Institute of Technology*, Nov. 1996.
[9] M. Johnson, *Superscalar Microprocessor Design*, Prentice Hall, 1991.
[10] F.Gabbay and A.Mendelson, "The Effect of Instruction Fetch Bandwidth on Value Prediction," *Proc. of the 25th International Symposium on Computer Architecture(ISCA-25)*, p.272-281, 1998.
[11] Yiannakis Sazeides and James E. Smith, "The Predictability of Data Values," *Proc. of 30th Annual ACM/IEEE International Symposium on Microarchitecture*, Dec. 1997.
[12] D.Burger and T.Austin, "The Simplescalar Tool Set, Version 2.0," *Technical Report CS-TR-97-1342, University of Wisconsin, Medison*, June. 1997.
[13] T.Y.Yeh and Y.N.Patt, "Two-level Adaptive Branch Prediction," *Proc. of the 24th Annual ACM/IEEE International Symposium on Microarchitecture*,pp. 51-61, 1991.
[14] M.H.Lipasti, C.B.Wilkerson and J.P.Shen. "Value Locality and Load Value Prediction," *Proc. of the 7th International Conference on Architectural Support for Programming Languages and Operating Systems(ASPLOS-VII)*, Oct. 1996.
[15] 박희룡, 전병찬, 이상정 "ILP프로세서에서 데이타 값 예측기의 성능평가", 98가을학술발표논문집(III)제25권 2호, pp.21-23, 한국정보과학회 1998. 9.
[16] 박희룡, 전병찬, 이상정 "ILP프로세서의 혼합형 데이타 값 예측기 모델", 99봄학술발표논문집(A) 제26권 1호, pp.18-20,. 한국정보과학회 1999. 4.

박 희 룡
1990년 대전산업대학교 전자계산학과 졸업(공학사). 1993년 수원대학교 전자계산학과 졸업 (이학석사). 2000년 순천향대학교 전자계산학과 (공학박사). 1996년 ~ 1997년 중앙대학교 산업교육전산원 교학부장. 1997년 ~ 현재 김천대학 컴퓨터사무정보계열 전임강사. 김천대학 사회교육원 소장. 관심분야는 고성능 프로세서 설계, 마이크로 프로세서 응용, 최적화컴파일러.

이 상 정
1983년 2월 한양대학교 전자공학과 졸업(공학사). 1985년 2월 한양대학교 대학원 전자공학과 졸업(공학석사). 1988년 8월 한양대학교 대학원 전자공학과 졸업(공학박사). 1988년 9월 ~ 현재 순천향대학교 공과대학 컴퓨터학부 교수. 관심분야는 고성능 프로세서 설계, 최적화 컴파일러, 마이크로프로세서 응용