혼합 예측기를 사용하는 효율적인 적재 명령어의 오퍼랜드 참조 기법

최 승 교[†] · 조 경 산[††]

요 약

프로세서들이 더 높은 클록에서 동작하고 동일 클록에 여러 명령어를 실행하게 됨에 따라, 적재 명령어의 참조 지연과 자료 의존성은 프로세서의 성능에 큰 영향을 주게 되었다. 오퍼랜드의 페치 지연을 감소시키고 적재 후에 사용되는 값의 자료 의존성을 해결하여 ILP를 증가시키기 위하여, 본 논문에서는 기존의 값 예측기법과 주소 예측 기법을 혼합한 새로운 예측 기법을 제안하고 성능 개선을 분석하였다. 시뮬레이션을 통해 5개 벤치마크 프로그램에서 적재 명령어의 수행을 분석한 결과, 제안 기법은 512항목 예측 버퍼의 경우에 값 예측 기법에 비해 전체 적재 명령어 실행에 대한 예측 성공율이 평균 50.08%에서 62.72%로 향상되었고, 기존의 주소 예측 기법에 비해 비용 면에서 효율적으로 구현됨을 보였다. 또한 스트라이드(stride)의 관리 및 과거의 예측 기록에 의한 성능 개선을 분석하였다.

An Improved Load Operand Referencing Scheme Using A Hybrid Predictor

Sung-Kyo Choi[†] · Kyung-San Cho[††]

ABSTRACT

As processor's operational frequency increases and processors execute multiple instructions per cycle, the processor performance becomes more dependent on the load operand referencing latency and the data dependency. To reduce the operand fetch latency and to increase ILP by breaking the data dependency, we propose a value-address hybrid predictor using a reasonable sized prediction buffer and analyse the performance improvement by the proposed predictor. Through the extensive simulation of 5 benchmark programs, the proposed hybrid prediction scheme accurately predicts 62.72% of all loads which are 12.64% higher than the value prediction scheme and shows its cost-effectiveness compared to the address prediction scheme. In addition, we analyse the performance improvement achieved by the stride management and the history of previous predictions.

1. 서 론

최근의 프로세서들은 더욱 높은 클록 주파수에서 동작하게 되었고, 또한 동일 클록 주기에 여러 명령어들을 발생(issue)하고 실행하는 수퍼스칼라 구조를 가지게 되었다[6, 7, 14]. 따라서, 명령어를 처리하는 각 단계의 신속한 처리와 명령어 수준의 병렬성 증대는 프로세서의 성능 향상을 위해 필수적이 되었다. 따라서, 프로세서에 비해 상대적으로 느린 메모리를 참조하는 적재 명령어의 오퍼랜드 참조에 소요되는 지연과 오퍼랜드에 의한 자료 의존성은 프로세서의 명령어 처리 시간 및 처리 대역폭에 큰 영향을 주게되었다[2]. 이 문

※ 본 논문은 삼척대학교 2000학년도 지체 학술연구비 지원에 의하여 연구되었음
†정 회 원 : 삼척대학교 컴퓨터공학과 교수
††종신회원 : 단국대학교 전산통계학과 교수
논문접수 : 2000년 5월 22일, 심사완료 : 2000년 7월 7일

제를 해결하기 위해 프로세서들은 보다 큰 용량의 온칩 캐쉬와 계층적 다중 캐쉬 구조를 사용하였으나 근본적인 해결책은 되지 못하였으며, 최근에는 적재 명령어가 참조하는 오퍼랜드에 대한 여러 예측 기법들이 제안되었다[12, 6, 13].

실제 다양한 프로그램의 실행 특성을 분석한 결과에 의하면 참조의 시간적 지역성과 유사하게, 일부 적재 명령어들은 반복 수행되며 반복 수행될 때 참조되는 오퍼랜드의 값 또는 저장 주소는 규칙성을 갖는다[7]. 적재 명령어의 페치 과정에서 반복 참조되는 오퍼랜드의 특성을 미리 예측하여 신속하게 오퍼랜드를 제공할 수 있다면 명령어 처리를 더욱 효율적으로 할 수 있다. 만약 제시된 예측이 성공한다면 적재 명령어가 오퍼랜드를 페치하는 시간을 줄일 뿐 아니라 오퍼랜드에 의한 자료 의존성을 조기에 해결하여 참조후 사용까지의 시간을 줄이고 명령어 수준의 병렬성을 높일 수 있다. 하지만 예측이 실패하면 그에 따른 페널티(시간 지연)가 발생한다.

예측 기법은 프로세서의 명령어 처리 환경, 메모리 참조 특성, 예측 기법의 구조 특성 및 운영 방법에 따라 성능 결과가 다르다. 따라서, 본 논문에서는 오퍼랜드 참조 특성과 오퍼랜드 예측 기법에 대한 기존 연구를 분석하여, 적재 명령어의 효율적인 오퍼랜드 참조를 위한 예측 방법을 제안한다 제안된 구조는 기존의 오퍼랜드 값의 예측 기법과 유효 주소의 예측 기법을 혼합한 예측기이며, 제안 기법에 의한 예측 개선을 분석하여 제시한다.

본 논문의 구성은 다음과 같다. 2장에서는 오퍼랜드 참조의 특성과 기존의 다양한 예측 기법을 분석하고, 3장은 기존 기법들을 혼합하여 개선된 예측 기법을 제시하고, 4장에서는 시뮬레이션을 통한 분석으로 제안 기법에 의한 성능개선을 검증하고, 5장에서 결론을 맺고 향후 연구과제를 제시한다.

2. 적재 명령어의 오퍼랜드 참조 특성과 기존 예측 기법의 분석

적재 명령어에 대한 오퍼랜드 예측은 오퍼랜드의 참조 특성에 따라 예측하는 대상, 예측 기법의 구조와 운영 방법이 달라질 수 있다 오퍼랜드 값의 참조 특성 및 이를 활용하여 예측할 수 있는 방법에 대해 다음과 같은 여러 규칙성이 분석되었다

1) 일부 적재 명령어는 항상 동일한 오퍼랜드 값을 참조하며, 이 특성은 참조의 값 지역성(value locality of reference)이라고도 불린다[12]. 직전에 참조된 오퍼랜드 값을 이용하여 예측할 수 있다.

2) 일부 오퍼랜드 값은 적재 명령어가 참조할 때마다 일정한 값만큼 변동된다. 이 경우에는 마지막 참조 값과 그 이전 참조 값과의 차이(값의 스트라이드)를 값의 예측에 이용할 수 있다[8].

3) 어떤 적재 명령어가 참조하는 오퍼랜드 값은 특정한 규칙을 갖는 패턴을 갖는다. 따라서, 반복 수행되는 참조 패턴을 이용하여 예측할 수 있다[15].

앞에서 제시한 값의 참조 특성과 유사하게 오퍼랜드가 저장된 유효 주소의 참조 특성 및 예측에의 활용도 다음과 같이 분석되었다.

1) 각 적재 명령어의 오퍼랜드 저장 주소는 항상 일정하다. 이 경우에는 한 번 참조된 오퍼랜드의 주소를 저장하여 다음 참조의 예측 시에 사용할 수 있다.

2) 각 적재 명령어의 오퍼랜드가 저장된 주소는 각 참조시 마다 일정한 값(주소의 스트라이드)만큼 변동된다 따라서 마지막 참조 주소에 그 값을 더하여 다음 참조 주소를 예측할 수 있다[11, 17].

3) 각 명령어의 오퍼랜드 참조 주소는 특정하게 반복되는 규칙적인 패턴을 갖는다. 따라서, 반복 수행되는 참조 패턴을 이용 유효 주소를 예측할 수 있다[2]. 이의 변형으로 포인터(간접 주소 지정 모드에 의한) 접근의 규칙성을 갖는 경우도 있다[4].

위와 같은 참조의 규칙성 분석으로부터 적재 명령어의 효율적 페치를 위한 예측의 대상은 적재할 오퍼랜드의 값 또는 유효 주소가 적절함이 알려졌으며, 명령어 페치 과정에서 이들을 미리 예측하여 오퍼랜드를 효율적으로 수행하는 것이 가능하다.

앞에서 분석된 참조 특성을 활용한 적재 명령어의 다양한 예측 기법은 다음과 같다. [9]에서는 CISC 구조를 갖는 x86 마이크로프로세서에서 수행되는 40억 개의 오퍼랜드 참조 주소를 분석한 결과 실제 참조된 오퍼랜드 주소의 대부분은 1000개 이하라는 결과를 이용하여, 명령어의 주소로 직접 오퍼랜드 값을 예측할 수 있는 오퍼랜드 프리페치 캐쉬(OPC : Operand Prefetch Cache) 구조를 제안하였다. 8-way 어소시어티브 매핑의 512 항목으로 구성된 OPC에 대해 전체 적재 명령

어의 35.59%에 대해 예측을 수행하여 95.3%의 예측 정확도를 보였다. 하지만 예측의 검증을 위해 캐쉬로부터 실제 오퍼랜드를 페치 해야 하므로 캐쉬의 참조 수는 변화가 없고, 예측이 실패로 인한 성능 손실이 크다는 문제점이 발생한다 [12]에서는 적재 명령어의 주소로 직접 그 명령어의 오퍼랜드 값을 예측할 수 있는 적재 값 예측 기법을 제시하였다. 예측 구조는 적재 값 예측표(LVPT : Load Value Prediction Table), 적재 분류표(LCT : Load Classfication Table), 검증장치(CVU : Constant Verification Unit)등으로 구성되며, 시뮬레이션을 통해 전체 오퍼랜드 참조 중에서 50%의 적재 값 예측이 가능함을 보였다 [18]에서는 기존의 연구에 검증의 시기를 다양화하고 주소 변환 기능을 추가한 오퍼랜드 참조 예측 캐쉬 ORPC(Operand Reference Prediction Cache) 구조를 제안하였다. 유효 주소 비교의 검증 방법을 사용하여 예측 실패시의 성능 저하를 최소화시킨 512 항목의 ORPC에 대한 시뮬레이션 결과 전체 적재 명령어에 대해 45%의 예측 성공률(예측된 경우에 대하여는 91%의 예측 성공률)을 보였다.

앞에서 설명된 값의 예측 기법과는 달리, [8]에서는 스트라이드를 이용한 값 예측 기법을 제시하고 이에 의한 명령어 수준 병렬성의 향상도를 분석하였다. 제시된 스트라이드 기반의 예측기는 적재 명령어 또는 산술 명령어의 오퍼랜드 예측에 이용되며, 각 열은 마지막 참조된 값, 연속 참조된 값의 차이인 스트라이드, 신뢰도의 3 항목으로 구성된다. 적재 명령어의 주소로 예측기를 참조하여 해당 신뢰도 값이 기준 이상이면 마지막 참조된 값에 스트라이드를 더한 값을 제공한다. 적재 및 산술 명령어에 대한 값의 스트라이드에 의한 예측의 정확도의 분석에 의하면 산술 명령어에 대해 정확도가 더 높다. 또한, [15]에서는 일정한 참조 패턴을 이용한 예측 방법으로 배열과 연쇄 링크 구조의 자료에 대한 접근시 LSC(Load/Store Cache)구조에 의한 예측을 효율적으로 할 수 있는 기법을 제안하였으며, 자료 캐쉬에서 미스 하는 경우만을 저장하며 시스템 모델을 통한 분석결과 항목수를 512에서 8로 줄이고도 실패율은 단지 3.75%만이 감소함을 보였다.

앞에서 설명된 바와 같이 오퍼랜드 값을 직접 예측하는 경우에는 별도의 버퍼에 예측 값을 저장하고 명령어 참조시에 이를 동시에 참조하여 직접 그 값을 가져오는 예측 버퍼 기법이 사용되었다. 기존 오퍼랜드 값의 예측을 위한 예측 버퍼 기법은 보다 신속히 오퍼랜드를 제공하지만, 전체 적재 명령어에 대한 예측의 정확도가 상대적으로 낮고 자료 캐쉬이외의 별도의 저장 장치를 구현하고 관리해야 한다는 부담이 있다. 또한 예측이 실패하면, 이로 인해 처리된 내용을 무효화해야 하는 페널티가 발생할 수 있다.

오퍼랜드의 유효 주소를 예측하는 경우는 적재 명령어의 주소로 다음 접근될 오퍼랜드의 주소를 예측하고 예측된 주소에 저장된 오퍼랜드를 미리 자료 캐쉬로 가져오는 프리페치 기법이 많이 사용되었다. 프리페치 기법은 자료 캐쉬의 히트율을 높일 수 있다. [17]에서는 메모리 참조의 정보를 참조 예측표(RPT : Reference Prediction Table)에 저장하여, 이전 참조 주소와 최근 두 번의 참조 주소의 차이인 스트라이드를 이용하여 다음 참조 주소를 계산하는 방법을 제안하였다. [11]에서도 유사하게 각 적재 명령어에 대해 최근에 참조된 두 오퍼랜드 주소의 차이를 이용하여 그 적재 명령어의 새로운 오퍼랜드 주소를 예측할 수 있는 버퍼인 LTB(Load Target Buffer)를 제안하였다. 또한, [4]에서는 배열과 같은 선형적인 자료들의 참조 주소뿐 아니라 링크 리스트, 인덱스 배열 또는 트리 구조로 연결된 포인터 참조를 하는 자료들의 주소를 예측할 수 있는 참조 값 예측 캐쉬 RVPC(Reference Value Prediction Caching)를 제안하였으며 레지스터 값이 반복적으로 변하는 것을 예측할 수 있도록 오퍼랜드 주소 계산에 사용되는 각 레지스터들의 예측값을 추적하여 저장한다. 대부분의 프로그램과는 달리 전체 오퍼랜드 참조 중에서 스트라이드 값이 일정하지 않은 적재 명령어의 비율이 높은 프로그램에 좋은 성능을 보이게된다.

최근에는 한가지 오퍼랜드 참조의 특성을 예측하는 기법과는 달리, 적재 명령어의 참조 특성들을 예측할 수 있는 여러 기법들을 사용하는 혼합 예측 기법이 제시되고 있다. [1]은 적재 명령어의 유효 주소 예측을 위하여 동일한 유효 주소를 예측하는 마지막 주소 예측기, 일정한 값의 차이를 갖는 유효 주소를 예측하는 스트라이드 주소 예측기, 반복되는 주소 패턴을 예측하는 전역 동적 예측기의 3개의 예측기를 이용하여 예측 기록과 신뢰도에 따라 가장 확률이 높은 결과를 예측하는 기법을 제안하였다. [10]은 오퍼랜드 값에 대한 스트리이드와 패턴을 혼합한 혼합 기법으로 예측하여 자료 캐쉬로 오퍼랜드를 프리페치 하는 기법을 제시하였다. 또한, [2]는 주소에 대한 패턴에 스트라이드 기법

을 혼합한 혼합 기법을 제시하고, 성능을 분석하였다.

각 프로그램마다 참조 특성이 다르므로, 하나의 예측 기법이 다양한 프로그램에 대해 모두 좋은 결과를 보일 수는 없다. 따라서, 기존 예측 기법의 단점을 상호 보완하여 더욱 높은 예측율을 얻을 수 있는 예측 기법이 연구 방향이 된다.

3. 혼합 예측 기법의 제안

예측 기법에 의한 명령어 처리 시간의 개선은 다음과 같은 식으로 표현될 수 있다.

평균 명령어 처리 시간 개선 = $P_h \times T_h - (1-P_h) \times T_m$

P_h : 예측이 성공할 확률
T_h : 예측 성공에 의한 시간 단축
T_m 예측 실패에 의한 시간 지연

예측 기법을 사용하여 처리 시간을 줄이고 신속한 오퍼랜드 제공으로 처리 대역폭 향상을 위해서는 예측이 성공할 확률(P_h)을 높이고, 예측 시점을 앞당겨 수행을 가능한 빨리 시작하여 T_h를 늘이고, 검증 기법을 통해 예측 확률을 높이고 실패에 의한 지연(T_m)을 감소시켜야 한다.

본 연구에서는 기존의 단일 예측 기법에서 발생하는 문제들을 해결하고, 위에서 제시한 개선의 조건들을 만족시키기 위하여 오퍼랜드 값의 예측 기법과 유효 주소의 예측 기법을 혼합한 기법을 제안한다. 기존 오퍼랜드 값의 예측 기법에서 발생하는 문제점으로는 전체 수행된 적재 명령어 중에서 예측이 가능한 경우의 수가 적은 것인데, 이는 작업부하 자체의 특성이므로 값의 예측에서 구조적으로는 해결될 수 없다 또한, 값의 검증을 실제 오퍼랜드 값을 이용할 경우에는 예측 기법은 검증만을 위해 캐쉬 접근이 필요하며, 오퍼랜드 페치로 인한 지연으로 검증단계가 늦어져 예측 실패에 따른 지연이 상당히 커진다.

기존 연구에 의하면, 적재 명령어에 대한 예측 가능 범위 및 정확도는 주소의 스트라이드(연속 참조된 유효 주소의 차이)를 사용한 오퍼랜드의 유효 주소 예측이 오퍼랜드 값의 예측보다 더 높음이 알려져 있다 [16]. 하지만, 단순히 유효 주소만을 예측하여 프리페치하는 것은 성능 개선 면에서 큰 의미가 없다. 따라서, 유효 주소의 예측 기법에서는 예측된 유효 주소에 저장된 값을 미리 캐쉬로부터 가져와 신속히 공급할 수 있는 방법이 제시되어야 한다.

따라서, 본 연구에서는 오퍼랜드 값의 예측과 유효 주소의 예측을 혼합한 예측 버퍼 구조를 제안하고, 유효 주소에 의한 동일한 검증 방법을 사용하여 제안된 구조가 예측의 가능 범위를 높이고 예측 실패에 의한 성능 저하를 감소시킬 수 있음을 제시한다. 제안한 기법을 구현하기 위해서는 예측을 위한 관련 정보를 저장하기 위한 예측 버퍼 구조를 사용하며, 이는 적재 명령어의 주소로 직접 참조가 가능하도록 한다.

3.1 제안 구조 및 운영

적재 명령어 주소로 접근되는 예측 버퍼 (그림 1)의 각 저장 위치는 다음 항목들로 구성되며, 본 연구에서 제안된 예측 구조 및 동작은 (그림 2)와 같다.

태그	유효 주소	스트라이드 0	스트라이드 1	오퍼랜드 값	유효 비트	예측 기록	LRU

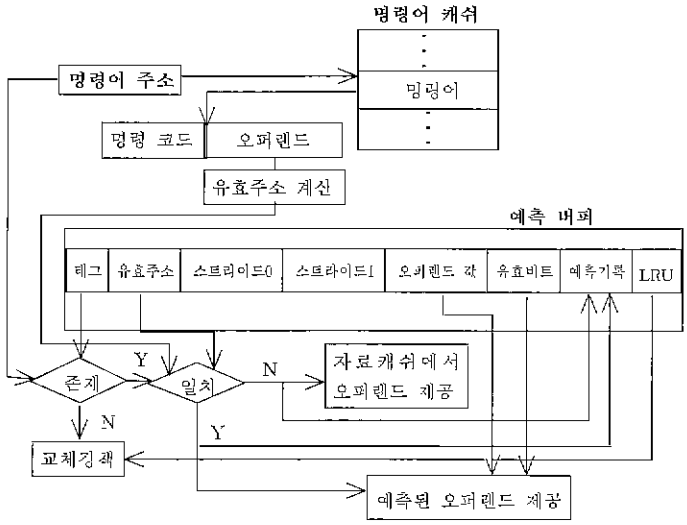
1) 태그 : 해당 적재 명령어의 주소(일부)
2) 유효주소 : 다음에 페치할 예측된 오퍼랜드 유효 주소
3) 스트라이드0 : 유효주소의 계산에 사용되는 스트라이드 값
4) 스트라이드1 : 최근에 참조된 두 유효주소의 차이 값
5) 오퍼랜드 값 : 적재 명령어의 예측된 오퍼랜드 값
6) 유효 비트 오퍼랜드 값의 유효성 표시
7) 예측 기록 : 최근 2번의 예측 정보의 성공/실패에 대한 기록
8) LRU : 예측 버퍼의 교체정책에 사용

(그림 1) 예측버퍼의 항목 구성

예측 버퍼를 통해 오퍼랜드를 제공하기 위해서, 적재 명령어의 페치 시에 동시에 예측 버퍼에 접근하여 해당 항목을 참조한다. 새로운 스트라이드 값에 따라 값 또는 주소에 의한 예측을 수행한다.

또한, 예측 기법의 문제점인 예측 실패의 페널티(시간 지연)를 해결하기 위해, 검증 시점을 디코딩 시점으로 앞당기고, 검증이 완료된 경우만 예측하도록 한다.

매번 예측이 실패할 때마다 계산된 스트라이드 값을 새로운 값으로 수정하게 되면, loop 문의 수행 시에는 좋지 않은 결과를 낼 수 있다. 따라서, 최근에 2번 연속해서 예측이 실패할 경우에만 스트라이드0 값을 직전에 페치한 두 오퍼랜드들의 유효주소의 차이 값인 스트라이드1로 대체 수정한다. 이를 위하여 기존 연구에서 많이 사용되던 포화 카운터가 아닌 과거 예측에 대한 기록을 사용한다. 포화 카운터 기법은 누적된 정보를 최대로 얻을 수 있다는 장점은 있지만 가장 최신의 정보는 아니라는 단점이 있다. 하지만, 과거 기록은 직전 두 번의 예측들의 성공 여부를 기록하여 예측에



(그림 2) 제안 예측구조 및 동작

대한 가장 최근의 정보를 나타낼 수 있다.

3.2 제안된 예측 버퍼의 운영 기법

제안된 기법은 적재 명령어 주소로 해당 오퍼랜드를 제공하도록 (그림 2)에서 보이는 과정대로 다음과 같이 운영된다

1) 적재 명령어 주소로 예측 버퍼를 참조한다.
2) 예측 버퍼가 히트(적재 명령어 주소와 예측 버퍼의 태그값이 동일한 경우)이면, 예측을 수행한다. 아니면, 예측 버퍼를 교체한 후에 일반적인 오퍼랜드 참조과정을 수행하고 종료한다.
3) 유효주소 히트(예측 버퍼의 유효주소 값과 계산된 유효주소가 동일한 경우)이면 다음과 같이 예측을 수행하여 오퍼랜드 값을 제공하고 종료한다.
   3-1) 스트라이드0=0 이면, 값에 의한 오퍼랜드 예측을 수행하고 예측된 오퍼랜드 값을 제공한 후 종료한다.
   3-2) 스트라이드0≠0 이면, 유효주소(직전 유효주소+스트라이드0)에 의한 예측을 수행하고 예측된 주소에서 오퍼랜드를 페치하여 제공하고 종료한다.

유효주소 미스이면, 일반적인 오퍼랜드 참조과정을 수행하고 종료한다. 이때, 해당여부 버퍼 항목이 연속2회 실패인 경우에는 스트라이드1의 값을 스트라이드0에 저장한다.

4. 시뮬레이션 및 성능 분석

본 연구에서는 제안된 오퍼랜드 참조 예측 버퍼 구조에 의한 성능 개선을 trace-driven 시뮬레이션을 통해 분석하였다. 시뮬레이션에 사용된 각 벤치마크에서 적재 명령어의 수행에 관한 정보는 SUN사에서 제공하는 shade를 사용하여 다음의 수행 환경을 통해 추적하였다[3, 5].

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

시뮬레이션을 통해 본 연구에서 제시된 값-주소 혼합 예측 기법은 앞 절에서 제시된 특성에 대해 다음과 같은 성능 분석을 보인다.

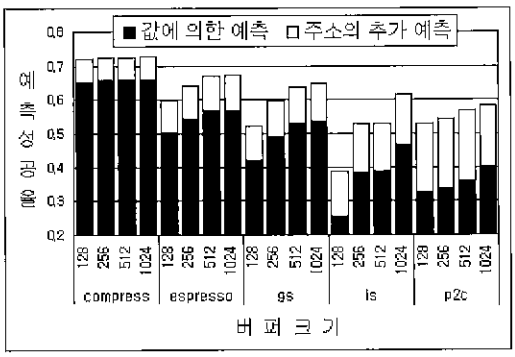
1) <표 1>에서 적재 명령 비율은 서로 다른 주소에 저장된 전체 적재 명령어의수에 대한 비율이고, 실행 명령비율은 실제 실행된 전체 적재 명령어의 수에 대한 비율이다. 실제 예측에 효율적인 명령어는 반복 실행 횟수가 높은 명령어인데, 적재 명령어의 수행 특성을 요약한 <표 1>로부터 전체 실행의 대부분은 1001번 이상 반복 실행되는 경우이다. <표 1>에서 1001번 이상 반복되는 적재 명령 비율(명령어수)은 각 벤치마

〈표 1〉 적재 명령어의 실행 특성

	compress		espresso		gs		ls		p2c	
	적재 명령비율	실행 명령비율	적재 명령비율	실행 명령비율	적재 명령비율	실행 명령비율	적재 명령비율	실행 명령비율	적재 명령비율	실행 명령비율
1	0.3174	0.0002	0.2447	0.0004	0.1728	0.0005	0.2459	0.0005	0.2277	0.0006
2~10	0.4788	0.0017	0.3073	0.0020	0.2659	0.0034	0.3352	0.0035	0.2675	0.0038
11~100	0.1028	0.0028	0.1588	0.0090	0.3494	0.0310	0.0958	0.0056	0.3095	0.0349
101~1000	0.0429	0.0143	0.1565	0.1088	0.1486	0.1485	0.2077	0.1537	0.1610	0.1276
1001~10000	0.0395	0.0701	0.1228	0.5870	0.0590	0.5136	0.1063	0.5022	0.0274	0.1886
10001~50000	0.0143	0.1902	0.0094	0.2199	0.0039	0.2506	0.0088	0.2938	0.0056	0.2441
50001~무한대	0.0043	0.7207	0.0005	0.0730	0.0003	0.0523	0.0003	0.0408	0.0014	0.4005
명령어 수	2306	3154130	3843	2549904	6642	2498445	3538	1796678	6461	2507381

크 프로그램에 대해 5.81%(134), 13.27%(509), 6.32%(420), 11.54%(408), 3.44%(222), 평균 8.076%로 전체 적재 명령어에 대한 비율이 매우 적고 이들의 적재 명령어수도 500여개 이하이다. 하지만, 이들에 의해 실행된 명령어 비율은 각각 98%, 88%, 82%, 84%, 83%으로 전체 실행수의 80%를 초과한다. 따라서, 적재 명령어에 대한 예측은 이들을 효과적으로 예측할 수 있다면 좋은 성능을 얻을 수 있다. 또한 예측 버퍼의 저장 항목 수를 무조건 늘이는 것보다 적절한 항목 수를 선정하고 반복 수행되는 명령어를 선택하여 저장하는 방법의 채택이 더욱 효율적이다.

각 벤치마크에 대해 예측 버퍼의 크기에 따른 예측 성공률을 보이는 (그림 3)에서 예측 버퍼의 저장 항목 수를 2배씩 증가(128에서 256, 512, 1024 항목으로)하여도 예측율이 이에 비례적으로 증가하지 않는 이유는 반복 횟수가 높은 소수의 명령어들이 전체 실행된 명령어의 대부분을 차지하고 또한 그들이 동시에 수행되지는 않기 때문이다.

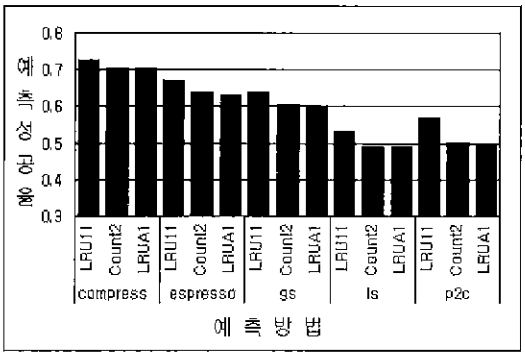


(그림 3) 제안 구조의 예측 성능분석
(버퍼 크기별 예측 성공률)

2) 제안된 기법에서 유효 주소의 변화가 없는 경우는 값에 의한 예측과 동일한 효과를 갖게 되므로, 이 경우에는 주소에 의한 예측을 하는 것은 비효율적이다. 즉, 이 경우에는 별도로 값에 의한 예측을 하는 것이 경제적이며, (그림 3)에 의하면 전체 예측 중에서 평균 50%는 동일한 값을 다시 사용하므로 이들을 값으로 예측하면 그 만큼 비용면에서 효율성을 높일 수 있게된다. 따라서, 값 예측 구조와 주소 예측 구조를 별도로 구현하는 것보다는 이들을 통합하여 운영하는 것이 비용 및 관리면 에서 유리하다. (그림 3)에서 예측 버퍼의 크기가 512항목인 경우에, 직전에 참조된 오퍼랜드가 다시 사용되는 값에 의한 예측의 평균 정확도 50.08%와 비교하여 제안된 혼합 예측 기법은 평균 62.72%의 예측 정확도를 보여 평균 12.64%의 예측 정확도 향상을 보인다. compress의 경우는 66%에서 73%로, p2c는 36%에서 57%로 향상된다. 다른 벤치마크 프로그램들은 이들의 중간치를 보인다 즉, 스트라이드 항목의 추가 비용만으로 기존 값에 의한 예측율을 주소의 스트라이드에 의해 평균 12.64% 향상시킬 수 있다. 다른 예측 버퍼의 크기에 대하여도 유사한 결과를 보인다. 따라서, 제안된 통합 예측 기법은 적절한 예측 버퍼의 크기 선정으로 가격 대비 효율성을 높일 수 있다.

3) 예측이 실패할 때마다 스트라이드 값을 수정하는 방법(그림 4에서 LRUA1로 표시)은 512항목의 예측 버퍼에서 평균 58.52%의 예측 정확도를 보이며, 최근 두 번의 예측 결과를 고려하여 루프 반복문에서의 예측을 개선하도록 본 연구에서 사용된 방법(그림 4에서 LRU11로 표시)은 이보다 4.2% 향상된 62.72%의 예측 정확도로 성능 개선을 보였다. 또한, 포화 카운터를 사용하는 방법(그림 4에서 Count2로 표시)은 512항목의

예측 버퍼에서 평균 58.74%의 예측 정확도를 보이며, 본 연구에서 제안된 예측에 대한 기록 정보를 사용하는 방법(그림 4에서 LRU1로 표시)은 62.72%의 예측 정확도로 3.98%의 성능 개선을 보였다.



(그림 4) 여러 기법의 예측 성능 분석

4) 제안된 혼합 예측 기법을 수행하기 위해서 요구되는 구현 비용은 예측 버퍼 크기로 분석할 수 있다. (그림 1)에서 예측 버퍼의 항목당 용량이 21 Byte가 되며 항목별 버퍼크기는 128 항목은 2.6KByte (128×21/1024), 256 항목은 5.3KByte, 512항목은 10.5KByte, 1024 항목은 21KByte의 크기의 메모리가 필요하며, 추가적인 제어회로나 비교기의 구현 비용은 무시할만하다

앞의 분석으로부터, 제안된 기법으로 주소 예측 기법보다는 저 비용으로 동일한 효과를, 또한 값의 예측보다는 약간의 비용 추가로 주소 예측 기법의 높은 예측 정확도를 얻을 수 있다.

5. 결 론

최근의 프로세서들은 보다 빠른 클록 주파수와 클록 주기에 여러 명령어들을 발생하고 실행하는 수퍼스칼라 구조를 가지게 되었다. 하지만, 급속히 발전하는 프로세서에 비해 상대적으로 느린 메모리를 참조하는 적재 명령어의 오퍼랜드 참조에 소요되는 시간과 오퍼랜드에 의한 자료 의존성은 프로세서의 명령어 처리 성능에 큰 영향을 주게되었다 이 문제를 해결하기 위해 적재 명령어가 참조하는 오퍼랜드에 대한 여러 예측 기법들이 제안되었으며, 이는 일부 적재 명령어들은 반복 수행되며 반복 수행될 때 참조되는 오퍼랜드의 값 또는 저장 주소는 규칙성을 갖는다는 특성의 활용이다

본 논문에서는 적재 명령어의 오퍼랜드 참조를 효율적으로 수행하기 위해, 기존의 값의 예측과 유효 주소의 예측 기법을 혼합한 새로운 예측 기법을 제안하고 이에 의한 성능 개선을 분석하였다 즉, 벤치마크 프로그램들에 대한 시뮬레이션을 통해 적재 명령어의 수행 특성을 분석하여 제안 기법의 예측 정확도를 제시하였다. 먼저, 예측 버퍼를 512항목으로 구성하면 실행된 전체 적재 명령어의 63%이상을 저장할 수 있음을 보였다 제안된 구조는 기존의 주소 예측 기법에서 스트라이드 값이 0인 경우는 별도로 직전에 참조된 값을 직접 제공하도록 처리하여 동일한 예측 성능을 보다 저렴하게 구현할 수 있음을 보였다. 또한, 오퍼랜드 값의 예측 구조에 주소 스트라이드 항목 비용의 추가로 예측 성공율을 50.08%에서 62.72%까지 12.64% 향상시킴을 보였다. 또한 스트라이드의 관리 및 과거 예측 기록의 활용에 의한 예측율의 향상을 보였다. 제안 기법은 예측 성능의 개선으로 명령어창 내의 의존성 문제를 조속히 해결하여 ILP를 증가시키며, 예측 실패에 의한 시간 지연을 줄이고, 주소 예측시 발생하는 불필요한 캐쉬 참조 회수를 줄일 수 있다. 따라서, 제안된 기법으로 주소 예측 기법보다는 저비용으로 동일한 성능을, 또한 값의 예측보다는 약간의 비용 추가로 높은 예측 정확도를 얻을 수 있다.

예측 기법의 향후 연구 과제는 보다 다양한 오퍼랜드 참조 특성의 분석과 이를 활용하여 제한된 예측 버퍼의 크기에서 더 높은 예측 성공율을 올릴 수 있는 기법의 개발이다. 이를 위하여, 분기 명령어와 적재 명령어의 상호 관계를 분석하여 예측 구조에 활용할 수 있도록 연구를 확장하고 있다.

참 고 문 헌

[1] B. Black, et al., "Load Execution Latency Reduction," *Procs. of 1998 ICS*, pp.29-35, 1998.

[2] B. Calder, G Reinman, D Tullsen, "Selective Value Prediction," *Procs. of 26th ISCA '99*, pp.64-74, 1999.

[3] Bob Cmelik and David Keppel, Shade, "A Fast Instruction-Set Simulator for Execution Profiling," *Sigmetrics*, ACM, pp.128-137, 1994.

[4] C. Chi and C. Cheung, "Hardware-Driven Prefetching For Pointer Data References," *Procs. of 1998 ICS*, pp 377-384, 1998.

[5] David L. Weaver and Tom Germond, *The SPARC Architecture Manual*, Prentice Hall, 1994.

[6] Digital Equipment Corporation, *Digital Semiconductor 21164 Alpha Microprocessor*, Data Sheet, Digital Equipment Corporation, 1997.

[7] Intel, *Optimization's For Intel's 32-Bit Processors*, available at http://ftp://download.intel.com, Intel Corporation, 1996.

[8] Jose Gonzalez and Antonio Gonzalez, "The Potential of Data Value Speculation to Boost ILP," *Proc. of 1998 ICS*. pp.21-28, 1998.

[9] L. Widigen, E. Sowadsky and K. McGrath, "Eliminating Operand Read Latency," *Computer Architecture News*, Vol.24. No.5, pp 18-22, 1996.

[10] M. Bekerman, et al., "Correlated Load-Address Predictor," *Proc. of 26th ISCA '99*, pp.54-63, 1999.

[11] M. Golden and Trevor Mudge, "Hardware Support for Hiding Cache Latency," CSE-TR-152-93, available at www.cs.umich.edu, U of Michigan, 1993

[12] M. Lipasti and et al , "Value Locality and Load Value Prediction," *Proc. of ASPLOS*. pp.138-147, 1996.

[13] Mikko. H. Lipasti, "Value Locality and Speculative Execution," *PhD Dissertation*, Carnegie Mellon University, 1997.

[14] MIPS Technologies, *MIPS R10000 Microprocessor User's Manual, Version 2.0*, MIPS Technologies Inc., available at http://www.mips.com. 1996.

[15] P. Ibanez, et al., "Characterization and Improvement of Load/Store Cache-based Prefetching," *Proc. of the 1998 ICS*, pp.369-376, 1998.

[16] Tien-Fu Chen and Jean-Loup Baer, "Effective Hardware-Based Data Prefetching for High-Performance Processors," *IEEE Transactions on Computer*, Vol.44, No.5, pp.609-623, 1995.

[17] Tien-Fu Chen, "Data Prefetching For High-Performance Processors," *Ph.D Dissertation*, University of Washington, 1993

[18] 김홍준, 조경산, "오퍼랜드 참조 예측 캐쉬(ORPC)를 활용한 오퍼랜드 페치의 성능개선", *한국 정보처리 학회 논문지*, 제5권 제6호, pp.1652-1659, 1998.

최 승 교

e-mail : skchoi@samchok.ac.kr

1982년 단국대학교 전기공학과 졸업(학사)

1992년 단국대학교 대학원 전산통계학과 전산교육(석사)

1996년~현재 단국대학교 대학원 전산통계학과 박사과정 수료

1994년~현재 삼척대학교 컴퓨터공학과 부교수

관심분야 : 컴퓨터 구조론, 성능평가, 시뮬레이션, 컴퓨터 통신

조 경 산

e-mail : kscho@dankook.ac.kr

1979년 서울대학교 전자공학과 졸업(학사)

1981년 한국과학원 전기 및 전자공학과 졸업(공학석사)

1988년 텍사스대학교(오스틴) 전기전산공학과 졸업(Ph.D.)

1988년~1990년 삼성전자 컴퓨터 부문 책임 연구원

1990년~현재 단국대학교 전산통계학과 교수

관심분야 : 시스템 구조론 및 성능 분석, 컴퓨터 통신, 시뮬레이션