論文
7-3-5

슈퍼스칼라 프로세서에서 명령어 이슈 길이를 고려한 값 예측기의 성능분석
(Performance Analysis of Value Predictor considering instruction issue width in Superscalar processor)

전 병 찬(Byoung-Chan Jean)[1] 김혁진(Hyeock-Jin Kim)[2]

요 약

슈퍼스칼라 프로세서에서 명령어 이슈 길이 값 예측방식은 명령의 결과 값을 미리 예측하고, 그 이후에 데이터 종속관계가 있는 명령들에게 값을 조기에 공급하므로써 이들 명령들을 모험적으로 실행하여 성능을 향상시키는 방식이다. ILP 프로세서는 명령어 수준 병렬성의 성능향상을 위하여 값을 미리 예측하여 병렬로 이슈하고 수행한다[4]. 본 논문에서는 이를 수행하기 위한 값 예측기의 명령어 이슈 길이(4,8,16)의 성능분석을 위한 예측률, 예측정확도, 성능향상 등을 측정하여 평가한다. 실험결과 8이슈의 성능향상이 높음을 보였다.

Abstract

Value prediction of instruction issue width in superscalar processor is a technique to obtain performance gains by supplying earlier source values of its data dependent instructions using predicted value of a instruction. In this paper, the mean performance improvement by predictor as well as prediction accuracy and prediction rate are meaned and assessed by comparison and analysis of value predictor that instruction issue width(4,8,16) in parallel and run by predicting value, which is for performance improvements of ILP[4]. The experiment result show the superiority hight performance of 8-issue.

▶Keywords: superscalar processor, value prediction, Instruction-Level Parallelism, issue

논문접수 : 2006. 5. 15.
심사완료 : 2006. 6. 10.

1)정회원 : 청운대학교 컴퓨터학과 전임강사
2)정회원 : 청운대학교 컴퓨터학과 부교수
*본 논문은 청운대학교 학술연구조성비 지원에 의하여 연구되었음

1. 서 론

슈퍼스칼라 프로세서(superscallar processor)에서 높은 성능을 도달하기 위해서는 명령어 수준 병렬성(Instruction Level Parallelism, ILP)을 이용하고 있다. ILP 프로세들은 다수의 데이터 패스와 기능장치들을 구비하여 여러개의 명령들이 동시에 이슈되어 병렬로 수행된다. ILP는 한 프로세서들에서 여러개의 명령을 이슈하여 성능을 극대화 하고 있다. ILP를 이용하는 주요장애는 명령어간의 종족이 있는데 이는 제어종속(control dependency)과 데이터 종속(data dependency) 관계이다[4].

제어종속 관계는 프로그램의 제어흐름이 변경되는 분기 명령에 의해 발생한다. 데이터종속관계는 현재 명령이 이전 명령의 수행결과를 참조할 때 발생하고, 이전 명령의 결과가 생성될 때까지 현재의 명령은 실행할 수가 없다[4]. 값 예측기는 하나의 명령을 실행하기 전에 실행결과를 하드웨어에 의해 미리 예측하고 모험적(speculative)으로 실행하여 이 명령과 데이터 종속관계가 있는 명령들이 중지(stall)하지 않고 계속 이슈하여 실행하므로써 프로세서의 성능향상을 시킨다.

본 논문에서는 슈퍼스칼라 프로세서 ILP를 이용하여 프로세서 성능을 향상시키기 위하여 값을 미리 예측하여 병렬로 이슈하고 수행하여 값 예측기의 명령어 이슈 길이(4,8,16)의 예측률, 예측정확도, 성능향상을 측정하여 평가한다. 실험되는 값 예측기는 최근 값 예측기[5], 스트라이드 값 예측기[6], 2-단계 값 예측기[3], 2-단계 값을 결합한 혼합형 값 예측기[2]로 구성되어 있다. 실험의 타당성 검증을 위해실행구동방식 시뮬레이터인 Simplescalar/Alpha 3.0 tool set[8]에 이용하여 SPECint95 벤치마크를 시뮬이션하여 비교한다.

2. 이론적 배경

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

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

즉, 상수(constant) 시퀀스는 0값이 증감되는 특수한 형태의 스트라이드 시퀀스로 볼 수 있고, 스트라이드 시퀀스는 명령이 수행될 때마다 일정한 값만큼의 간격 차이로 증감되는 패턴으로 프로그램에서 배열(array)이나 루프 인덱스(loop index)등을 참조할 때 빈번히 발생하는 패턴이다. 또한, 비 스트라이드(non-stride) 시퀀스는 위의 상수 시퀀스 또는 스트라이드 시퀀스에 속하지 않은 모든 시퀀스를 의미한다. 이러한 값 시퀀스에서 특히 상수 시퀀스는 같은 값이 반복되어 사용되는 패턴으로 Lipasti 등은 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 데이터 값
반복 스트라이드	1 2 3, 1 2 3 ...
	내부 루프의 반복 스트라이드값 (1, 2, 3)
반복 비 스트라이드	1 34 -3 7 6, 1 34 -3 7 6 ...
	내부 루프의 반복 비 스트라이드 값 (1 34 -3 7 6)

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 값 시퀀스 분류에 의해 값 예측기를 계산형 예측기(computational predictor)와 내용형 예측기(context-based predictor)로 구분된다[2,8,10,12].

3. 값 예측기의 분류

슈퍼스칼라 프로세서에서 값 예측기로는 여러 가지 기법들로 제안 되었다. 대표적인 예측기는 예측되는 값의 유형에 대한 시퀀스 패턴의 분류에 따라 최근 값 예측기, 스트라이드 값 예측기, 2-단계 값 예측기, 혼합형 값 예측기가 있다[4].

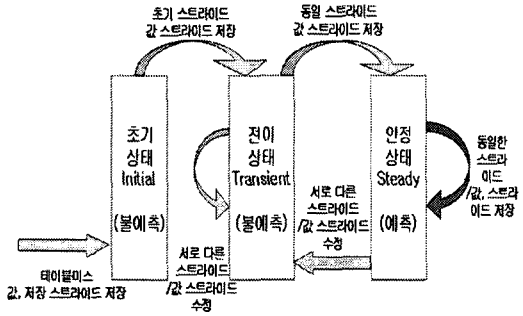
3.1 최근 값 예측기(Last Value Predictor)

Lipasti에 의해 개발된 최근 값 예측기는 프로그램 실행동안 대부분의 명령들 값의 변화가 적음을 주목하고 레지스터 값을 변경하는 명령들이 재사용되는 값의 국한성(locality)을 실험하여 이를 근거로 최근 값 예측기를 제안 하였다[4,5]. 최근값 예측기는 명령의 주소에 의해 인덱스되는 테이블에 이 명령의 최근 수행된 결과를 저장하고 다음에 이 명령 페치시 저장된 최근 값으로 결과 값을 예측한다. 이러한 최근 값 예측기는 최초의 값 예측기로써 의의가 있고, 예측기의 구성은 단순하다는 장점이 있으나, 상수 시퀀스를 갖는 명령들만을 예측할수 있다는 단점이 있다.

3.2 스트라이드 값 예측기(Stride Value Predictor)

Gabby[5]와 Wang[6]등은 상태, 값, 스트라이드등이 저장된 VHT로 구성된 스트라이드 예측기를 설계 하였다. 스트라이드 값 예측기는 한 명령의 연속적인 인스턴스(instance)의 결과들이 변화하는 스트라이드를 모니터(monitor)하여 이들 결과 값들이 일정한 폭으로 변하면 그 명령어의 장래 인스턴스의 결과 예측이 용이하다.

[그림 1]은 스트라이드 예측기의 값 예측을 위한 3 상태 천이도를 보여주는 그림이다. 상태의 값은 예측 테이블의 상태 필드에 저장된다. 먼저 예측될 명령이 처음 테이블에 등록될 때 상태는 초기상태가 되고 명령의 값을 예측하지 않는다. 이 명령의 실행 후 예측 테이블 갱신 시에 실행 결과 값(value1)을 테이블의 값 필드에 저장한다. 이 후 같은 명령이 다시 페치되고 상태가 초기상태이면 상태를 전이상태로 변환하고 값은 예측되지 않는다. 또한, 현재 실행된 결과 값(value2)에서 테이블의 값 필드에 저장된 값(value1)의 차를 구하여 스트라이드(stride1)를 스트라이드 필드에 저장하고 최근의 실행 값(value2)도 값 필드에 저장한다. 또 한번 동일한 명령이 페치되어 상태가 전이상태이므로 값은 예측되지는 않는다. 테이블이 갱신될 때 현재 실행 결과 값(value3)과 테이블의 값 필드에 저장된 값(value2)의 차로 새로운 스트라이드(stride2)를 구하고, 새로운 스트라이드(stride2)와 이전 스트라이드(stride1)의 값이 같으면 상태는 전이상태에서 안정상태로 변환된다. 만약 같지 않으면 상태는 그대로 전이상태로 남아있게 된다. 그리고 예측 테이블의 값과 스트라이드 필드에 각각 새로운 값으로 값(value2)와 스트라이드(stride2)를 저장한다. 명령의 페치 시 예측 테이블의 상태의 값이 안정상태일 경우에만 값 필드와 스트라이드를 더한 값으로 예측한다.



[그림 1] 스트라이드 예측기의 상태도
[Fig.1] state of predictor stride

3.3. 2-단계 값 예측기(Two-Level Data Value Predictor)

Wang등은 기존의 분기예측에서 사용된 2-단계적응 분기예측(two-level adaptive branch predictor)방식을 값 예측기에 적용한 2-단계 값 예측기를 개발 하였다[4,6]. 2-단계 값 예측기는 VHT와 PHT(Pattern History Table) 두

개의 테이블로 구성된다. VHT 테이블은 예측되는 명령의 주소에 의해 인덱스 되며, 최근에 사용된 n개의 값과 마지막에 수행된 p개의 명령 결과 값을 가리키는 인덱스의 비트 패턴이 VHP에 저장된다. VHP의 인덱스 비트패턴 PHT를 인덱스 하는데 사용된다.

PHP의 각 엔트리는 VHT에 저장된 n개 데이터 값과 일치하는 증감 카운터들로 구성된다. Lookup 동작은 PC에 의해 VHT가 인덱스되고, 히트(hit) 되면 VHP에 의해 PHT가 인덱스 된다. PHT의 카운터 중에서 최대값을 선택하고, 이 값이 임계값 보다 크면 이 카운터의 위치에 대응되는 VHT의 데이터 값이 예측되고, 임계값 보다 작은 경우에는 예측을 하지 않는다.

3.4 혼합형 값 예측기(Hybrid Data Value Predictor)

Wang등은 스트라이드 예측기와 2-단계 값 예측기를 결합한 혼합형 예측기를 제안하였다[5]. 혼합형 값 예측기는 일정한 값으로 증감하는 명령을 예측하는 스트라이드 예측기와 2-단계 값 예측기를 결합한 것으로 확신 카운터(confidence counter)의 값에 의해 예측하는 방법이다. 명령어가 두개의 예측기에서 모드 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 얻을수 있지만 하나의 명령어가 두개의 테이블 엔트리에 중복 저정되어 높은 하드웨어 비용을 필요로 한다는 단점을 가지고 있다.

4. 실험 방법

본 논문의 성능측정은 명령어 이슈 길이(4, 8, 16)의 명령의 시스템의 기본 구조를 사용하였다. 기존의 예측기인 최근 값 예측기, 스트라이드 값 예측기, 2-단계 값 예측기, 혼합형 값 예측기를 사용하였으며[14], 또한, <표 1>에서는 기본구조인 시뮬레이션된 명령어 이슈 길이(4, 8, 16) 프로세서 구성에 대한 각 머신의 파라미터를 보여준다. 분기예측을 위해서 2-way의 2K 엔트리를 갖는 BTB와 gshare와 bimodal 방식을 혼합한 분기 예측기를 사용하였다. 그리고 512K 크기를 갖는 명령 캐시와 데이타 캐시를 사용하였다. 시뮬레이션을 수행하기 위해 사용되어진 벤치마크 프로그램은 <표 2>와 같이 SPECint95 벤치마크 프로그램 중 8개의 프로그램에 대해 시뮬레이션 하였다. <표 2>에서 기본 IPC(Instructions Per Cycles) 즉, 명령어 이슈 길이(4, 8, 16)는 값 예측을 적용하지 않은 경우의 IPC로써 speed up 측정 시 기본 값이 된다. 본 연구에서 사용된 시뮬레이터는 Simplescalar 3.0 Tool set[8]에 값 예측기 모델을 삽입하여 실험 하였다.

<표 1> 실험에 사용된 기본 시스템
<Table 1> basic system using experiment

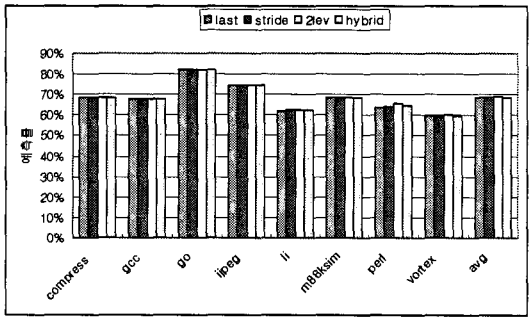
	파라미터 (Parameter)	값(Value)	의미(Comments
Process or core	RUU size LSQ size Fetch width Decode width Issue width Commit width Functional Unit	256 128 4,8,16 instr/cycle 4,8,16 instr/cycle 4,8,16 instr/cycle 4,8,16 instr/cycle 8 int ALU(1) 2 int MUL(3) 4 fp ALU(1) 2 fp MUL(4)	instruction windows Load-Store Queue 4,8,16 issue baseline 4,8,16 issue baseline 4,8,16 issue baseline 4,8,16 issue baseline ()은 Latency 임
Branch Predictor	BTB Combine Branch Predictor RAS	2K, 2-way 2-level 16K, 14bit history bimodal 16K 32	Branch target buffer 2-level + bimodal return addr. stack
Cache	L1 D-cache L1 I-cache L2 cache	512K, 2-way, 32 byte blocks 1-cycle latancy 512K direct map, 128 byte block 4-way,	Data cache Instruction cache secondary cache

<표 2> SPECint95 벤치마크 프로그램의 입력자료
<Table 2> input data of SPECint95 benchmark

벤치마크 프로그램	입력자료	명령의 크기	기본 IPC		
			4이슈	8이슈	16이슈
compress	10000e2231	35,261,714	2.1289	2.6580	2.8914
gcc	jump.i5 9	38,772,635	1.4720	1.4664	1.8439
go	tinyrose.pp	81,530,040	1.3630	1.2704	1.5592
ijpeg	m	63,415,601	2.2726	2.8987	3.6440
li	queen6.lsp	41,524,887	1.8413	2.2399	2.8703
m88ksim	dhry.big.100	240,738,844	3.2316	4.6006	6.7654
perl	scrabbl.in	40,353,136	1.8720	2.2182	2.5354
vortex	persons.250	66,740,617	1.9815	2.2779	3.0164
평 균		7,604,218,425	2.0203	2.4537	3.1407

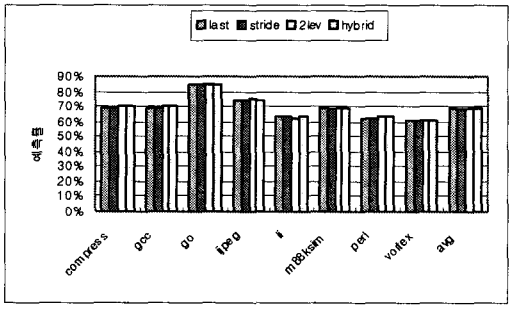
5. 성능분석 및 평가

본 논문에서 실험한 값 예측기는 테이블 갱신에 따른 명령어 이슈 길이(4, 8, 16)의 예측률, 예측 정확도, 성능향상 등의 실험 결과를 얻었다. 예측률은 전체 실행된 명령어들 중에서 예측기에 예측되어질 명령들의 비율이다. [그림 2]는 4이슈 머신 예측률을 보여주고 있다.



[그림 2] 4 이슈 머신 예측률
[Fig.2] 4-issue machine predictor rate

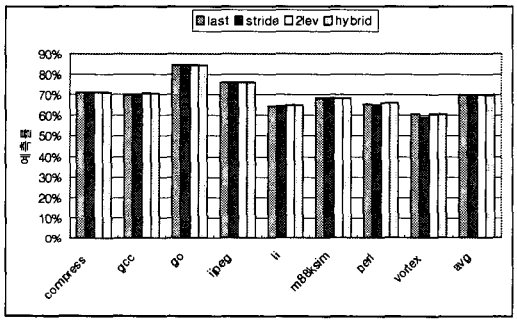
[그림 2]에서 보이는 것처럼 last 예측기, stride 예측기, 2lev 예측기, hybrid 예측기의 예측률은 각각 68.4%, 68.4%, 68.7%, 68.6% 임을 알수 있다. 2lev 예측기가 근소한 차이로 다른 예측기보다 예측률이 높음을 알수 있다. [그림 3]은 8이슈 머신 예측률을 보여 주고 있다. 8이슈 머신 일 때는 last 예측기, stride 예측기, 2lev 예측기, hybrid 예측기의



[그림 3] 8 이슈 머신 예측률
[Fig.3] 8-issue machine predictor rate

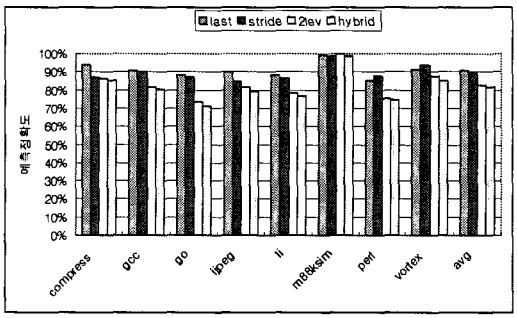
예측률은 각각 68.8, 68.8%, 69.3%, 69.7% 임을 알수 있다. 8이슈 머신 예측률은 4이슈 머신 예측률과는 달리 hybrid 예측기가 근소한 차이로 높은 예측률을 보여주고 있다.

[그림 4]는 16이슈 머신 예측률을 보여 주고 있다. 16이슈일 때는 last 예측기, stride 예측기, 2lev 예측기, hybrid 예측기의 예측률은 각각 70.1%, 69.8%, 70,3%, 70,4% 임을 알수 있다. 특히, go인 경우 다른 벤치마크 프로그램보다 예측기들이 높은 예측률을 보여 주고 있다.



[그림 4] 16 이슈 머신 예측률
[Fig.4] 16-issue machine predictor rate

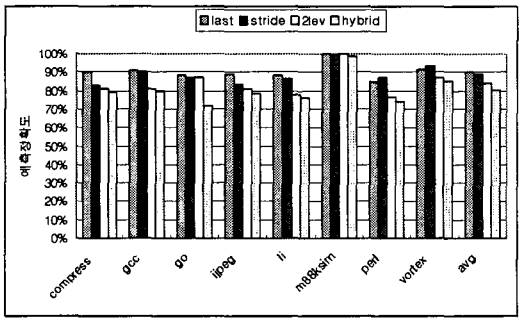
예측 정확도는 값 예측이 가능한 명령 중에 안정상태에 도달하여 실제 값 예측을 한 명령 중에서 올바르게 값을 예측한 비율로써, [그림 5]는 4 이슈일 때의 예측 정확도를 보여주고 있다.



[그림 5] 4 이슈 머신 예측정확도
[Fig.5] 4-issue machine predictor accuracy

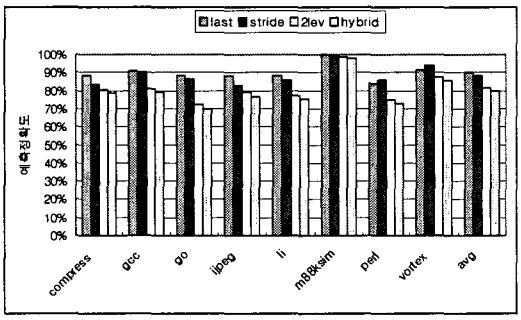
[그림 5]에서 보면 last 예측기, stride 예측기,

2lev 에측기, hybrid 예측기의 예측정확도는 각각 90.8%, 89.5%, 83.1%, 81.4% 임을 알수 있다. 4이슈 예측정확도는 4 이슈 예측률과는 달리 조금 상반된 결과가 나타남을 알수 있다. 또한, m88kim인 경우 2lev 예측기 가장 높게 예측정확도를 보여 주고 있다. [그림 6]은 8이슈의 예측정확도를 보여주고 있다. [그림 6]에서도 4이슈 예측정확도와 유사한 형태의 결과를 보이고 있다. 즉, last 예측기, stride 예측기, 2lev 에측기, hybrid 예측기의 예측정확도는 각각 90%, 88.6%, 83.8%, 80.4%임을 알수 있다.



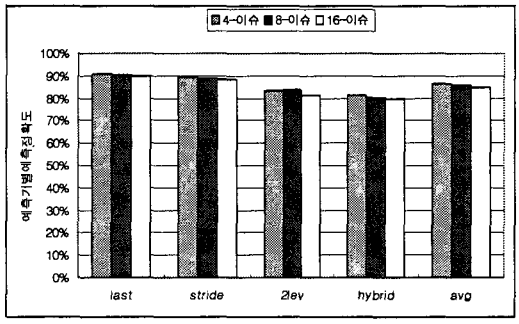
[그림 6] 8이슈 머신 예측정확도
[Fig.6] 8-issue machine predictor accuracy

[그림 7]에서는 16이슈의 예측정확도를 보여주고 있는데, last 예측기, stride 예측기, 2lev 예측기, hybrid 예측기의 예측정확도는 각각 8 9.8%, 88.6%, 81.3%, 79.6% 임을 알수 있다.



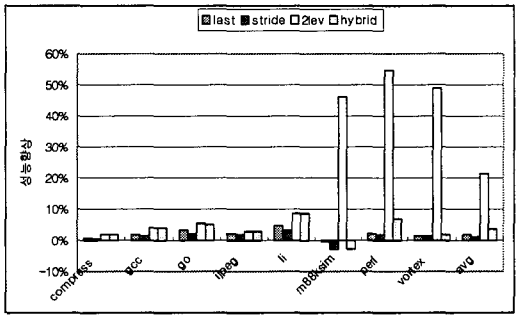
[그림 7] 16 이슈 머신 예측정확도
[Fig.7] 16-issue machine predictor accuracy

16 이슈 예측정확도에서도 4 이슈, 8 이슈의 예측정확도와 비슷한 결과가 나옴을 알수 있었다. [그림 8]은 예측기별 평균 예측 정확도를 나타내는 그림이다.4 이슈, 8 이슈, 16 이슈의 예측정확도는 각각 86.2%, 85.6%, 84,8% 임을 알수 잇다. 예측기별로 볼때 전반적으로 4 이슈가 가장 높게 예측정확도를 보여 주고 있다. 특히, 2lev 예측기에서 4 이슈의 예측정확도가 8 이슈 보다 0.8% 정도 낮게 예측정확도를 보여 주고 있다.



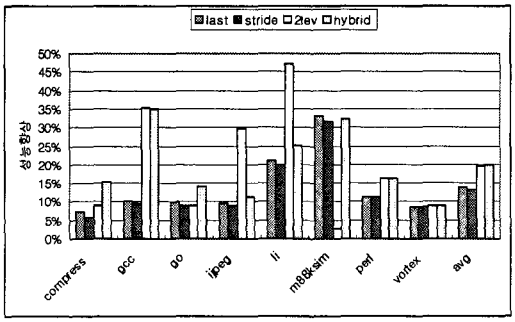
[그림 8] 예측기별 평균 예측정확도
[Fig.8] average predictor accuracy of predictors

성능향상은 값 예측을 시도하지 않은 경우를 기준으로 값 예측을 하였을 경우의 성능 변화를 나타내는 것이다. [그림 9]에서 나타난 바와 같이 4 이슈인 경우 last 예측기, stride 예측기, 2lev 에측기, hybrid 예측기 성능향상은 각각 1.9%, 1.1%, 21%, 3.5%임을 알수 있다.



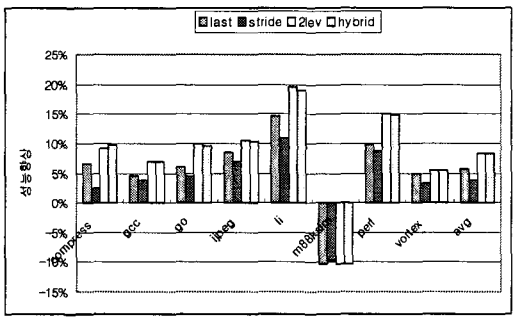
[그림 9] 4이슈 머신 성능향상
[Fig.9] 4-issue machine performance improvement

2lev 예측기가 가장 성능향상이 높음을 알수 있다. 특히, m88ksim, perl, vortex인 경우에 2lev의 성능향상이 높게 나타났으며, m88ksim은 2lev을 제외한 나머지 예측기들은 감소를 보이고 있다. [그림 10]은 8 이슈인 경우의 성능향상을 보여주고 있다.



[그림 10] 8 이슈 머신 성능향상
[Fig.10] 8-issue machine performance improvement

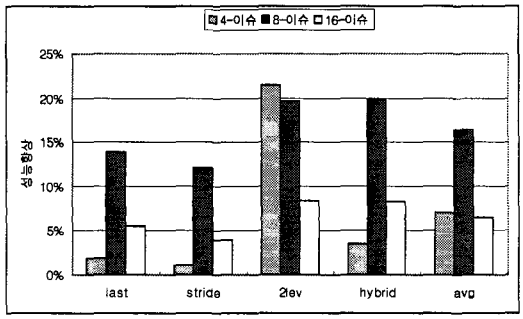
8 이슈인 경우에 last 예측기, stride 예측기, 2lev 예측기, hybrid 예측기의 성능향상은 각각 13.8%, 13%, 19.%, 19,8% 임을 알수 있다. 8 이슈인 경우 2lev 예측기 보다 hybrid 예측기가 약간 성능향상이 높음을 알수 있다. 유독 m88ksim에서 2lev 예측기가 가장 낮은 성능향상을 보여 주고 있다. [그림 11]은 16이슈인 경우의 성능향상을 보여 주고 있다. 16 이슈인 경우도 last 예측기, stride 예측기, 2lev 예측기



[그림 11] 16이슈 머신 성능향상
[Fig.11] 16-issue machine performance improvement

, hybrid 예측기의 성능향상은 5.5%, 3.8%, 8.2% 임을 알수 있다. 2lev 예측기가 근소한 차이로 다른 예측기보다 높은 성능향상이 높음을 알수 있다. m88ksim인 경우 모든 예측기들이 가장 큰 성능저하를 보여주고 있다.

[그림 12]는 예측기별 평균 성능향상을 보여주고 있다. [그림 12]에서 보는 것처럼 4 이슈, 8 이슈, 16 이슈의 성능향상은 각각 7.0%, 16.0%, 6.4%의 임을 알수 있다. 예측기에서는 8 이슈의 성능향상이 가장 높음을 알 수 있다. 또한, 2lev 예측기를 제외한 나머지 예측기에서 8이슈가 높은 성능향상을 보여 주고 있다.



[그림 12] 예측기별 평균 성능향상
[Fig.12] average performance improvement of predictors

6. 결론

본 논문에서는 슈퍼스칼라프로세서에서 값 예측기인 최근 값 예측기, 스트라이드 값 예측기, 2-단계 값 예측기, 혼합형 값 예측기를 테이블 갱신시점에 따른 명령어 이슈 길이(4,8,16)의 예측률, 예측정확도, 성능향상 등을 분석 하였다. 성능 분석결과 예측률에서는 명령어 이슈 길이인 16 이슈가 예측률이 월등히 높아 보임을 알수 있었으며, 예측정확도에서는 4이슈, 8이슈, 16이슈 순으로 예측정확도가 높음을 알수 있었다. 예측정확도에서 특이한 점은 2lev에서 4이슈의 예측정확도가 떨어지는 것을 알 수 있었다. 예측기별 성능향상에서는

8이슈가 월등히 높음을 알 수 있었다.

참고문헌

[1] 전병찬, 이상정, "와이드 이슈 프로세서를 위한 스트라이드 값 예측기의 모험적 갱신", 한국정보과학회 논문지, 제28권,제12호, pp.601-612. 2001.

[2] 전병찬, 박희룡, 이상정, "와이드 이슈 슈퍼 스칼라 프로세서의 혼합형 값 예측기의 성능평가", 대한전자공학회호서지부 추계 학술발표대회 논문집 제2권 제1호, pp.136-141, 2000.

[3] 이상정, 전병찬, "값 예측을 위한 순차적이고 선택적인 복구방식", 한국정보과학회 논문지 제31권 제1·2호, pp.67-77, 2004.

[4] 전병찬, 김혁진, 류대희, "고성능 마이크로프로세서에서 값 예측기의 성능평가", 한국컴퓨터정보학회 논문지 제10권 제2호, pp88-95, 2005.

[5] M.Lipasti and J.Shen, "Exceeding th Limit via Value Prediction", Proceedings of the 29th International symposium on Microarchitecture (MICRO-29), pp.226-237, Dec. 1996

[6] Kai Wang and Manoj Franklin, "Highly Accurate data value Predictions using hybird predictor ,"Proc. of 30th Annual ACM/IEEE International Simposium on Microarchitecture, pp.281-290, Dec. 1997.

[7] T.Yeh and Y.Patt, "Two-Level Adaptive Branch Prediction, "Proceedings of the 24th International Symposium microarchitecture (MICRO-24), pp.51-61, Nov. 1991.

[8] D.Burger and T.Austin, "The Simplescalar Tool Set, Version3.0", Technical Report CS-RT-97-1342, University of Wisconsin, Madison, June 1997.

[9] F.Gabbay and A.Mendelson, "Speculative Execution Based on Value Prediction", EE Department TR 1080, Technion-Israel Institute of Technology, Nov. 1996.

[10] 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), pp138-147, Oct. 1996.

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

[12] Y.Sazeides and J.Smith, "The Predicatability of Data Values", Proceedings of the 30th International Symposium on Microarchitecture (MICRO-30), pp.248-258, Dec. 1997.

전 병 찬
순천향대학교 대학원 전산학과 박사
현 청운대학교 컴퓨터학과
전임강사
관심분야 : 컴퓨터구조, 홈네트워크, 모바일, 마이크로프로세서 등

김 혁 진
아주대학교 대학원 컴퓨터공학과
석·박사
김천대학 사무자동화과 교수
현 청운대학교 컴퓨터학과 부교수
관심분야 : CG, CAGD, 웹기술 등