논문 2009-2-14

슈퍼스칼라 프로세서에서 명령 윈도우 크기에 따른 혼합형 값 예측기

Hybrid Value Predictor in Wide-Issue Superscalar Processor

전병찬*, 최규석**

Byoung-Chan Jeon, Gyoo-Seok Choi

요 약 본 논문에서는 슈퍼스칼라에서 윈도우 크기에 따른 명령 페치율에 따라 혼합형 값 예측기의 성능을 평가한다. 일반적으로, 명령의 데이터 의존성은 명령의 페치수에 따라 증가된다. 그러므로, 명령 페치율이 증가할 때 값 예측기의 성능이 높다고 본다. 이러한 성능은 명령 페치 메카니즘인 컬랩싱 버퍼 와 트레이스 캐쉬로 연구한다. 실험결과는 명령 윈도우 크기에 따른 명령 페치율 증가와 혼합형에서 non-tc 와 tc을 적용한 IPC와 예측률의 값 예측기의 성능 효과를 평가한다.

Abstract In this paper, the performance of a hybrid value predictor according to the instruction fetch rate on window size superscalar processors is evaluated. In general, the data dependency relations of instructions are increased with the number of the fetched instructions. Therefore, it is expected that the performance of a value predictor will be higher when the instruction fetch rate is increased. The performance is studied for the machine with collapsing buffer and he one with trace cache as an instruction fetch mechanism. As a result of experiment, it is showed that the performance effect of a value predictor is higher as the instruction fetch rate of instruction window size, IPC, predict rate on apply with non-tc and tc is increased.

Key Words : Superscalar, Data dependency, Trace cache,

Ⅰ. 서 론

최근에 개발되고 있는 슈퍼스칼라 프로세서는 명령어 수준 병렬성(instruction level parallelism, ILP)을 이용하여 다수의 명령을 동시에 이슈하고 처리하여 성능을 향상시키고 있다. 이들 프로세서에서 높은 명령어 수준 병렬성을 유지하려면 사이클 당 많은 명령을 페치하여 명령 윈도우에 공급하는 것이 필수적이다. 그러나 대부분의 프로그램들이 빈번한 조건분기 명령들의 수행으로 제어흐름이 빈번하게 변하는 실정이다. 따라서 한 개의 캐시블록에 하나의 기본블록 만을 페치하는 전통적인 페치방식으로는 명령의 공급이 한계에 도달하고 있다. 최근에는 이와같은 문제점을 극복하기 위하여 사이클 당 한 개 이상의 조건분기 결과를 예측(multiple branch prediction)하고 이를 토대로 한 개 이상의 기본블럭을 페치하여 공급하는 기법들이 개발되었다[2,6,8].이 중 컬랩싱 버퍼(collapsing buffer)[2]는 명령 캐시블록에 2개의 기본 블록을 페치하여 명령을 공급해주며, 트레이스 캐시(trace cache)[6]는 명령 캐시 외에 최근에 실행된 명령들의 순서(instruction sequence)를 저장하고 공급해준다.

명령어 수준 병렬처리의 주요장애는 명령 간의 종속관계인데 이는 명령의 병렬처리를 방해한다. 즉, 현재 한 명령이 이전명령과 종속관계가 있으면 이전명령의 결과가 나올 때까지 현재 명령은 이슈되어 병렬로 실행 할 수가 없게 된다. 이러한 명령어 간의 종속관계는 이름종속

*정회원, 청운대학교 방송영상학과
**종신회원, 청운대학교 컴퓨터학과
접수일자 2009.3.18, 수정완료 2009.4.8

(name dependency),제어종속(control dependency), 데이타 종속(data dependency)관계의 3가지로 구분 할 수가 있다. 이름 종속관계는 레지스터나 메모리의 저장위치의 재사용으로 기인한 것으로 정적 또는 동적 재 명명(renaming) 기법으로 제거한다. 제어종속관계는 분기의 결과에 따라 프로그램의 제어흐름이 변경되는 조건분기에 의해 발생한다. 제어종속관계를 극복하는 방식으로는 조건 분기명령을 제거하고 제어종속관계가 있는 명령을 조건실행(predicated execution)하여 제거하는 방식과 조건분기의 결과를 예측하여 제어종속관계가 있는 명령들을 미리 페치하여 투기적 실행(speculative execution)하는 방식이 있다[9]. 데이타 종속관계는 현재의 명령이 이전명령의 수행결과를 참조할 때 발생하고 이전 명령의 결과가 발생할 때까지 현재의 명령은 실행할 수가 없다. 데이터 종속관계는 와이드 이슈 프로세서에서 명령간에 빈번히 발생하기 때문에 명령어 수준 병렬처리의 주요 장애가 되어 최근의 와이드 이슈 고성능 슈퍼스칼라 프로세서의 성능 저하의 주된 요인이 되고 있다. 따라서, 최근에 실행되는 명령의 결과 값을 미리 예측하고 이후 데이터 종속관계가 있는 명령들에게 조기에 공급하여 이들 명령들을 투기적으로 실행하여 성능향상을 꾀하는 데이타 값 예측(data Value prediction)방식에 관하여 활발히 연구가 진행되고 있다.[3,4,7]

본 논문에서는 명령 윈도우 크기에 따른 명령 페치율 증가와 혼합형에서 non-tc 와 tc을 적용한 IPC와 예측률의 값 예측기의 성능 효과를 평가한다.

즉, 높은 페치율과 와이드 이슈로 인하여 공급되는 명령이 많아질 경우에 데이타 종속관계가 많아지므로 데이타 값 예측으로 인한 큰 성능향상이 된다고 전제하고 이를 측정하고 평가분석 한다. 사용된 명령 패치 메카니즘으로는 2개의 기본블록을 페치 공급할 수 있는 컬랩싱 버퍼와 트레이스를 공급하지 않는 트레이스 캐시와 트레이스를 공급하는 트레이스 캐시에 혼합형 데이터값 예측기를 적용하여 성능을 평가한다.

Ⅱ. 다수의 기본블록 페치 메카니즘

그림 1은 파이프라인 구조에서 슈퍼스칼라 프로세서의 명령어 실행을 보여 주고 있다. 따라서, 각 단계의 기능에 대해 언급한다면 IF(Instruction Fetch:명령어 인출) 단계는 PC(Program Counter)에 있는 주소를 사용하여 메모리에 명령어를 읽고 IF/ID 파이프라인 레지스터에 저장한다. ID(Instruction Decode:명령어 해석)단계에서는 IF/ID 파이프라인 레지스터에 있는 명령어를 읽어 레지스터 번호 두 개를 제공하며 값들을 ID/EX 파이프라인 레지스터에 저장한다. 또한, EX(Execution:실행)단계는 유효주소 값을 EX/MEM 파이프라인 레지스터에 기억하며, MEM(Memory)단계에서는 저장될 데이터를 가지고 있는 레지스터는 전 단계에서 읽혀져 ID/EX에 저장한다. 마지막 단계인 WB(Write Back)단계에서는 계산된 결과 값을 레지스터에 저장하고 모든 명령은 이 단계에서 끝을 맺는다.

clock →						
명령수	1	2	3	4	5	6
i	IF	ID	EX	MEM	WB	
i+1	IF	ID	EX	MEM	WB	
i+2	IF	ID	EX	MEM	WB	
i+3	IF	ID	EX	MEM	WB	

그림 1. 슈퍼스칼라 프로세서 명령어 실행
Fig 1. Superscalar processor instruction execution

그림 2는 컬랩싱 버퍼를 사용한 명령페치 메카니즘을 보여주는 그림이다. 이 방식은 BTB(Branch Target Buffer)로부터 지정된 두 개의 명령캐시 뱅크로부터 페치된 기본 블록들을 버퍼에 저장하고 분기 사이에 사용하지 않는 명령을 제거한 후 실제 사용된 명령들의 시퀀스를 제공해 준다.

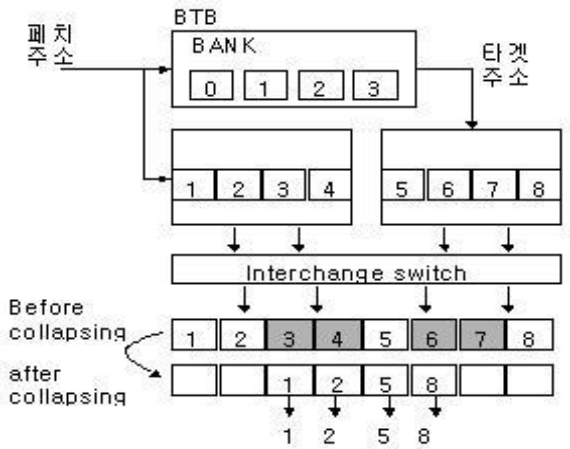


그림 2. 컬랩싱 버퍼 기법
Fig 2. collapsing buffer technique

트레이스 캐시를 이용한 페치 메카니즘은 명령캐시 외에 프로그램의 실행에 따라 논리적으로 연속적인 명령을 캐싱할 수 있는 트레이스 캐시로부터 명령들을 페치하는 방식이다. 이후 같은 트레이스를 수행 시에는 명령캐시로부터 페치하지 않고 트레이스 캐시에 저장된 명령들을 페치한다. 보통 하나의 트레이스에는 최대 3개까지의 기본블록까지 저장된다. 그림 3은 트레이스 캐시 명령페치 구조를 보여주는 그림이다. 명령페치 시 트레이스 캐시와 명령캐시가 참조된다. 만약 트레이스 캐시가 히트되면 명령들은 트레이스 캐시로부터 페치된다. 트레이스 캐시가 미스되면 명령캐시로부터 페치되고 이 후 이들명령들은 트레이스를 형성하기 위해서 필유닛(fill unit)에 보내진다. 필유닛은 명령들을 모아 트레이스를 만들고 이를 트레이스 캐시에 전송한다. 다음에 같은 트레이스가 실행되면 명령캐시가 아닌 트레이스 캐시로부터 페치된다.

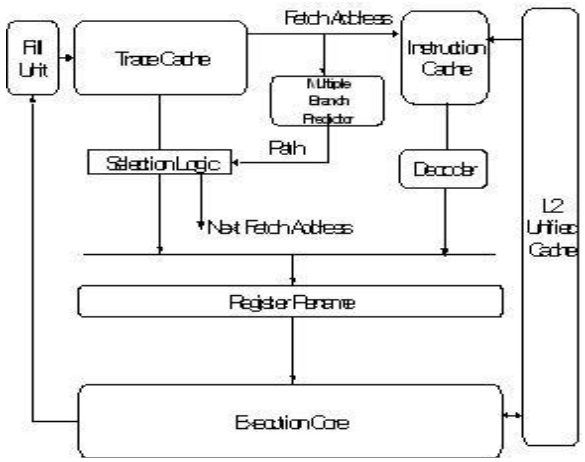


그림 3. 트레이스 캐시 명령페치 기법
Fig 3. trace cache instruction fetch technique

Ⅲ. 데이터 값 예측기

데이타 값 예측기는 예측되는 데이타 값 시퀀스의 분류에 따라 최근 값 예측기, 스트라이드 예측기, 2-단계 값 예측기, 혼합형 예측기등이 있다.

최근 값 예측기(Last Value Predictor)[4]는 한 명령의 최근에 수행된 결과를 테이블에 저장하고 다음에 이 명령 수행 시 이 값을 사용하여 예측하는 기법으로 가장 단순한 데이타 값 예측기법이다. 그림 4는 최근값 예측기를 보여주고 있다.

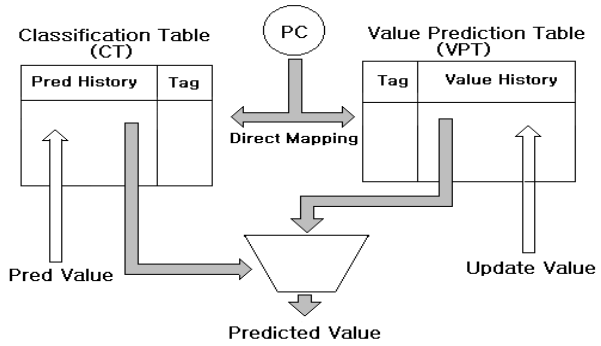


그림 4. 최근 데이타 값 예측기
Fig 4. Last data value predictor

스트라이드 예측기(Stride Predictor)[3,7]는 한 명령의 최근에 수행된 두 개의 결과값의 차인 스트라이드를 구하고 이를 테이블에 저장한다. 이후 이 명령 수행시 최근의 결과값과 스트라이드 값을 더한 결과를 예측값으로 사용하는 기법이다. 그림 5는 스트라이드 데이터 값 예측기를 보여주고 있다.

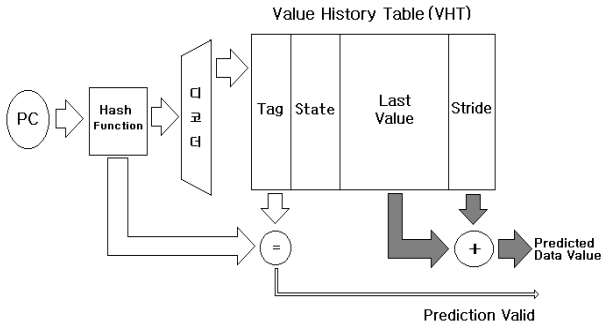


그림 5. 스트라이드 데이타 값 예측기
Fig 5. Stride data value predictor

2-단계 값 예측기(Two-Level Value Predictor)[7]는 2-단계 분기 예측기법과 유사한 구조를 갖고 있다. 즉 VHT(Value History Table)와 PHT(Pattern History Table) 2개의 테이블로 구성된다. VHT 테이블은 예측되는 명령 주소에 의해 인덱스 되며 최근에 사용된 n개의 데이타 값과 마지막에 수행된 p개의 명령 결과 값을 가리키는 인덱스의 비트패턴인 VHP(Value History Pattern)가 저장된다. PHT는 VHT의 VHP로부터 인덱스 되며 VHT 테이블에 저장된 데이타 값과 연관된 2비트 카운터로 구성된다. 만약 명령의 실행된 데이타 값이 VHT에 있으면 이에 해당되는 카운터가 증가하고, 다른 카운터들은 감소한다. 데이타 값 예측 시에는 최대 카운터 값을 갖는 VHT의 데이타 값을 예측값으로 사용한다.

그림 6은 2단계 값 예측기를 보여 주고 있다.

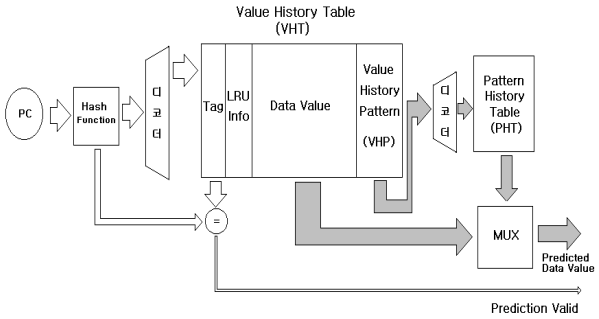


그림 6. 2-단계 데이터 값 예측기
Fig 6. two-stage value pridictor

혼합형 데이타 값 예측기(hybrid Value Predictor)[7]는 스트라이드와 2-단계 데이타 값 예측기를 결합한 예측기이다. 그림 7은 혼합형 데이터 예측기의 구조를 보여준다. 2-단계 데이타 값 예측기와 비교하여 VHT에 스트라이를 저장하는 필드가 추가되었다.

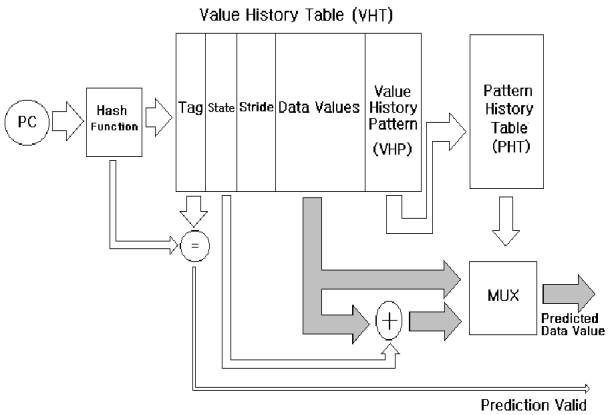


그림 7. 혼합형 데이타 값 예측기 구조
Fig 7. hybrid value predictor structure

혼합형 데이타 값 예측기는 먼저 데이타 값 예측을 위해 2-단계 데이타 값 예측 방식을 적용을 시도하고, PHT의 카운터 값들을 참조한다. 만약 카운터의 최대값값이 설정된 임계값(threshold value)보다 크거나 같으면 2-단계 데이타 값 예측을 적용하여 이에 해당하는 VHT의 값으로 예측하고, 그렇지 않으면 스트라이드 예측기를 사용하여 최근값에 스트라이드를 더한값으로 예측한다.

본 논문에서는 컬랩싱 버퍼 및 트레이스 캐시(non-trace cache, trace cache) 페치 메카니즘에 혼합형 데이타 값 예측기를 적용하여 명령 윈도우 크기에 따른 페치율의 값 예측의 성능 측정은 그림 8과 같은 머신 구조를 모델링하여 성능을 측정 분석 하였다.

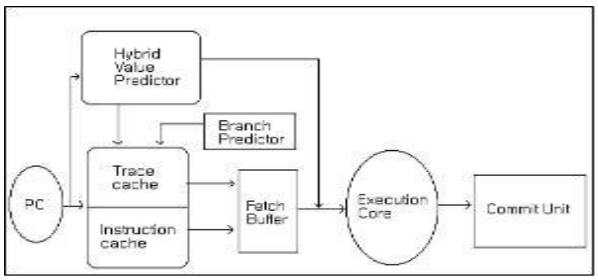


그림 8. 시뮬레이션 머신 구조
Fig 8. simulation machine structure

IV. 실험방법

본 논문의 성능측정은 명령 윈도우 크기와 4~16 이슈에 대한 3가지 시스템의 기본 구조를 사용하였다. 표 1에서는 기본구조에서 사용된 머신의 파라미터를 보여준다. 분기예측을 위해서 2-way의 2K 엔트리를 갖는 BTB와 gshare와 bimodal 방식을 혼합한 분기 예측기를 사용하였다. 그리고, 512K 크기를 갖는 명령 캐시와 데이터 캐시를 사용하였다.

표 1. 실험에 사용된 기본 시스템 구조의 Configuration
Table 1. Configuration base system structure on using example

	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	512K, 2-way, 32 byte blocks	Data cache
	L1 I-cache	1-cycle latancy 512K direct map,	Instruction cache
	L2 cache	128 byte block 4-way,	secondary cache

표 2는 기존의 혼합형 데이타 값 예측기와 트레이스 캐시의 파라미터로써 혼합형 데이타 값 예측기는 4k, 8K 엔트리의 VHT와 4개의 히스토리를 갖고 , 4k, 8K 엔트리로 구성된 PHT의 테이블 크기로 구성된다. 트레이스 캐시는 512KB 와 블록당 16개 명령을 가진 2-way 어소시에티브를 가진다.

표 2. 실험에 사용된 데이타 값 예측기 구성
Table 2. composition on value predictor using example

	테이블 구성
Non-Trace cache, Trace cache	512K, 2-way 16 instruction trace cache block
Hybrid Value predictor	4k,8K VHT(Value History Table) 4k,8K PHT(Pattern History Table) 4 history data values

시뮬레이션을 수행하기 위해 사용되어진 벤치마크 프로그램은 표 4와 같이 SPECint95 벤치마크 프로그램 중 8개의 프로그램에 대해 시뮬레이션 하였다.

표 3 에서 기본 IPC(Instructions Per Cycles)는 데이타 값 예측을 적용하지 않은 경우의 IPC로써 speed up 측정 시 기본 값이 된다. 본 연구에서 사용된 시뮬레이터는 Simplescalar 3.0 Tool set[9]에 데이타 값 예측기 모델을 삽입하여 실험 하였다.

표 3. SPECint95 벤치마크 프로그램의 입력자료
Table 3. Input data of SPECint95 benchmark program

벤치마크 프로그램	입력자료	명령의 크기	기본 IPC		
			4이슈	8이슈	16이슈
compress	10000 e 2231	35,261,714	2.1289	2.6580	2.8914
gcc	jump.i	38,772,635	1.4720	1.4664	1.8439
go	5 9	81,530,040	1.3630	1.2704	1.5592
ijpeg	tinyrose.ppm	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

V. 성능평가 및 분석

본 논문에서 제안한 실험 결과는 IPC(혼합형에서 non-trace와 trace), 성능향상(혼합형에서 non-trace와 trace), 예측률(혼합형에서 non-trace와 trace), 명령어 윈도우 크기의 성능향상을 실험 하였다.

그림 9는 혼합형 예측기 에서 non-tc 와 tc의 IPC에 대한 결과를 보여주고 있다. IPC는 한 사이클 당 명령들을 공급할 수있는 능력으로서 한 사이클당 많은 명령을 공급할수 있으므로 성능을 향상 시킬수 있다. 그림9에서 보여주는 바와같이 평균적으로 non-tc와 tc는 각각 3.09%와3.16%로 나타내어 tc가 0.05%가 높음을 알수 잇었다. 또한 벤치마크별로 보면 m88ksim과 vertex에서도 tc가 IPC 높음을 보여주고 있다.

그림 10은 혼합형에서 non-tc와 tc에 대한 예측률을 보여주고 있다. 그림에서 보여주는 것처럼 평균적으로 non-tc는 6.6%이고 tc는 6.2%로 나타내었다. 따라서, 예측률은 평균적으로 non-tc가 높음을 보였다.

그림 11은 혼합형 예측기 에서 non-tc와 tc에 대한 성능향상을 보여주고 있다. 앞에서 IPC는 다소 향상된 면을 보였으나, 예측율은 낮은 결과를 보였다. 그렇지만, 성능향상은 평균적으로 non-tc는 9.5%, tc는 11.9%으로 나타내어 tc가 2.4% 차이의 성능향상을 보여 주었다.

그림 12는 명령 윈도우 크기(4k,8k)에 따른 이슈별로 나타 내엇다. 1b는 1개의 기본블록만을 명령캐시로부터 페치하는 메카니즘을 의미하며, 2b는 2개의 기본블록을 페치할 수 있는 컬랩싱 버퍼를, tc는 최대 3개의 기본블록을 페치할 수 있는 트레이스 캐시 메카니즘을 나타낸다. (a)는 4-이슈에서 1b인 경우 4k는 6.5%, 8k는 6.4%이고, 2b 인 경우 4k는 5.6%, 8k는 5.7%, tc인겨우 4k는 5.5%, 8k는 5.8%의 성능향상을 나타 내었다 (b)의 8-이슈에서는 1b인 경우 4K는 9.2%, 8k는 9.4%이고, 2b인 경우 4k는 18,1%, 8k는 17,5%, tc인 경우 4k는 19.8%, 8k는 19.1%의 성능향상을 나타 내었다. (c)는 16-이슈에서 1b인 경우 4k는 11.4%, 8k는 12.9%, 2b인 경우 4k는 19.6%, 8k는 20.0%, tc인 경우 4k는 24.1%, 8k는 24.8%의 성능향상을 나타 내었다. 따라서, 일반적으로 이슈율이 증가함에 따라 명령 윈도의 크기에서도 성능향상이 증가 됨을 알았다.

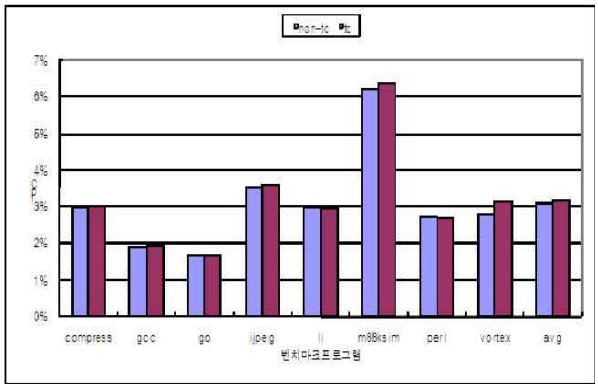


그림 9. 혼합형 예측기에서 non-tc 와 tc의 IPC 비교
Fig 9. IPC compare of hybrid predictor non-tc and tc

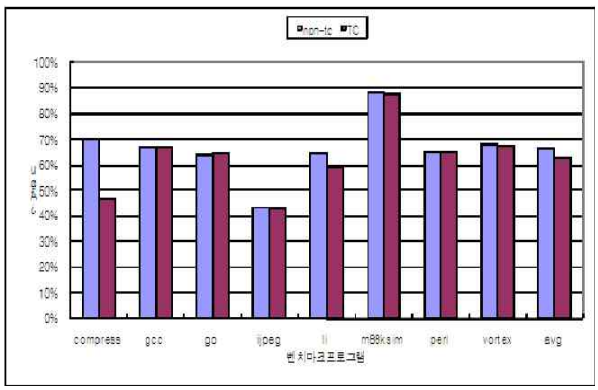


그림 10. 혼합형 예측기에서 non-tc 와 tc의 예측률
Fig 10. predictor rate of hybrid predictor non-tc and tc

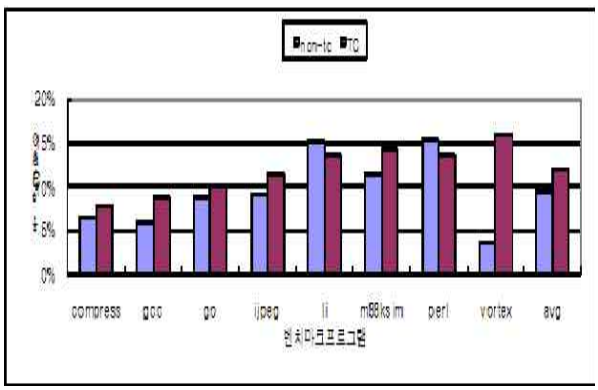
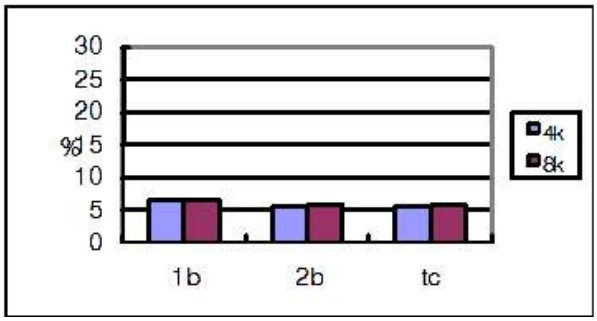
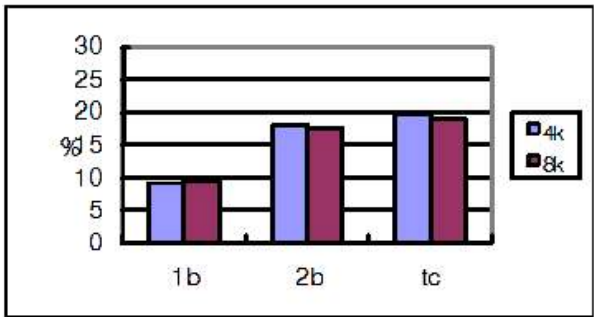


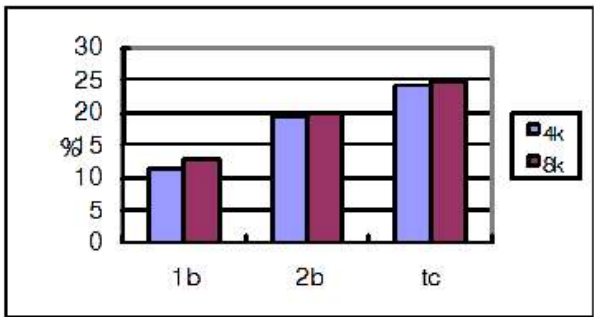
그림 11. 혼합형 예측기에서 non-tc 와 tc의 성능향상
Fig 11. speed-up of hybrid predictor non-tc and tc



(a) 4-이슈



(b) 8-이슈



(c) 16-이슈

그림 12. 명령 윈도우 크기의 성능향상
Fig 12. speed-up of instruction window size

VI. 결론

본 논문에서는 명령어 윈도우 크기에 따른 이슈율(4이슈,8이슈,16이슈)인 명령어 수준 병렬성을 이용하는 슈퍼스칼라 프로세서에서 데이타 값 예측기의 성능을 분석하고 평가를 위해 컬랩싱 버퍼 및 non-trace cache와 trace cache 페치 메카니즘에 혼합형 데이타 값 예측기를 적용하여 IPC, 예측률,성능향상을 측정하고 분석하였다.

실험 결과 혼합형 예측기에서 non-tc와 tc의 IPC는 각각 3.09%,3.16%나타내었다. 또한, 예측률은 각각 6.65%,6.2%의 예측률로 나타내었다. 성능향상은 각각 9.5%,11.9%의 성능향상을 보여 주었다. 결과적으로 혼합

형 예측기에서 non-tc보다는 tc를 적용하는 것이 값 예측으로 인한 약간의 성능향상이 됨을 알수 있었다. 또한, 명령어 크기에 따른 성능향상도 8-이슈를 제외한 나머지 4-이슈와 16-이슈에서 성능향상이 약간 높음을 알 수 있었다.

참 고 문 헌

[1] D.Burger and T.Austin, "The Simplescalar Tool Set, Version 3.0," Technical Report CS-TR -97 -1342, University of Wisconsin, Medison, June. 1997.

[2] T.M.Conte, K.N.Menezes, P.M.Mills, and B.A. Patel., "Optimization of instruction fetch mechanisms for high issue rate", Proc. of 22th Annual international Symposium on Computer Architecture, pp. 333-344,1994

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

[4] 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.

[5] S.J.Lee, Y.Wang, and P.C.Yew, "Decoupled Value Prediction on Trace Processors," IEEE 6th international Symposium on high performance computer architecture, 2000.

[6] E.Rotenberg, S.Bennett, and J.E.Smith, "Trace cache : a low latency approach to high bandwidth instruction fetching", proc. of 29th Annual ACM/IEEE international Symposium on Microarchitecture, 1996

[7] K.Wang,M.Franklin, "Highly Accurate data value Predictions using hybird predictor," Proc. of 30th Annual ACM/IEE E International Symposium on Microarchitecture,Dec.1997.

[8] T.Y.Yeh, D.T.Marr, and Y.N.Patt, "Increasing the Instruction Fetch Rate via Multiple Branch Prediction and a Branch Address Cache," in International Conference on Supercomputing '93, 1993, pp 67-76.

[9] T.Y.Yeh and Y.N.Patt, "Two-level Adaptive Branch Prediction," Proc. of 24th Annual ACM/IEEE International Symposium on Microarchitecture, pp. 51-61, 1991.

[10] 박희룡, 전병찬, 이상정 "ILP프로세서에서 데이타 값 예측기의 성능평가," 98가을학술발표논문집 (Ⅲ) 제25권 2호, pp.21-23, 한국정보과학회 1998. 9.

[11] 박희룡, 전병찬, 이상정 "ILP프로세서의 혼합형 데이터 값 예측기 모델," 99봄학술발표논문집(A) 제26권 1호, pp.18-20, 한국정보과학회 1999. 4.

저자 소개

전 병 찬(정회원)

- 1992년 한밭대학교 전산학과 공학사
- 1994년 수원대학교 전산학과 공학석사
- 2002년 순천향대학교 대학원 전산학과 박사
- 2008년 현재 청운대학교 방송영상학과 조교수

<관심분야 : 컴퓨터구조, 홈 네트워크, 모바일, 마이크로프로세서 등>

최 규 석(종신회원)

제8권 제6호 참조

현 청운대학교 컴퓨터학과 교수

<주관심분야 : 이동통신, 인공지능, 인공생명, 지능형 교통체계(ITS), 이동 컴퓨팅>