제어 독립성을 이용한 분기 예상 실패 복구 메커니즘

윤성룡○ 신영호 박홍준 조영일
수 원 대 학 교 전 자 계 산 학 과
slyoun@cs.suwon.ac.kr hjpark@cs.kdc.ac.kr yicho@mail.suwon.ac.kr

A Branch Misprediction Recovery Mechanism using Control Independence

Sung-LyongYoon○ Young-Ho Shin Hong-Jun Park Young-Il Cho
Dept. of Computer Science, The University of Suwon

요 약

제어 독립성(Control Independence)은 슈퍼스칼라 프로세서에서 명령어 수준 병렬성(Instruction-Level Parallelism)을 향상시키기 위한 중요한 요소로 작용하고 있다. 분기 예상 기법(Branch Prediction Mechanism)에서 잘못 예상될 경우에는 예상한 분기 방향의 명령어들을 제거하고 올바른 분기 방향의 명령어들을 다시 반입하여 수행해야 한다.

본 논문에서는 컴파일 시 프로파일링을 통한 정적인 방법과 프로그램상의 제어 흐름을 통해 동적으로 제어 독립적인 명령어를 탐지함으로써 분기 명령어의 잘못된 예상으로 인해 제거되는 명령어를 효과적으로 감소시켜 프로세서의 성능을 향상시키는 메커니즘을 제안한다.

SPECint95 벤치마크 프로그램에 대해 기존의 방법과 본 논문에서 제안한 방법 사이의 사이클 당 수행된 명령어 수를 분석한 결과, 4-width 프로세서에서 4%~6%, 8-width 프로세서에서 11%~18%, 16-width 프로세서에서 15%~27%의 성능 향상을 보이고 있다.

1. 서 론

고성능 마이크로프로세서는 여러 개의 명령어를 동시에 병렬로 수행시켜 성능을 향상시키고 있다. 이런 명령어 수준 병렬성을 저해시키는 요소로 제어 종속성(Control Dependence)이 있다. 분기 명령어의 결과가 구해질 때까지 분기 명령어 다음에 수행될 명령어의 반입을 지연시키기는 제어 종속성을 제거하기 위해서 많은 분기 예상기법(Branch Prediction Mechanism)이 연구되고 있다[1,2].

현재 사용중인 분기 예상기법에서는 분기 명령어의 예상이 틀렸을 경우 이후에 반입되어 수행되고 있는 명령어들을 모두 무효화시키고 올바른 방향의 명령어들을 재반입해서 수행시키고 있다. 그러나, 이미 수행된 제어 독립적인 명령어들을 재반입하여 수행해야 하므로 하드웨어 자원의 낭비가 심해져 프로세서의 성능 감소를 가져올 수 있다.

본 논문에서는 컴파일러와 추가적인 하드웨어를 통해 잘못된 분기 예상의 경우에도 분기의 결과에 상관없는 명령어, 즉 제어 독립적인 명령어를 제거시키지 않음으로서 성능을 향상시키는 메커니즘을 제안한다.

2. 관련연구

분기 예상이 틀렸을 경우 페널티를 줄이는 방법으로 예상이 어려운 명령어의 제어 종속을 데이터 종속으로 변환을 시키는 Predication 기법[3]이 제안되었으며, 하드웨어적인 방법으로는 dual-path execution 기법과 multiple path execution 기법[4]이 제안되었다.

Shen은 페널티를 줄이기 위해 잘못 예상된 분기 명령어 이후의 명령어 중 제어 종속적인 명령어만 제거하는 기법을 제안하였다[5,6]. 그러나 Shen의 기법은 제어 독립적인 명령어를 탐지하고, 여러 개의 분기 명령어가 pending될 경우를 처리하기 위해 많은 하드웨어 비용이 요구된다. 또한, 제어 독립적인 부분 중에서 데이터 종속 관계를 탐지하는 하드웨어 비용도 큰 비중을 차지하게 된다.

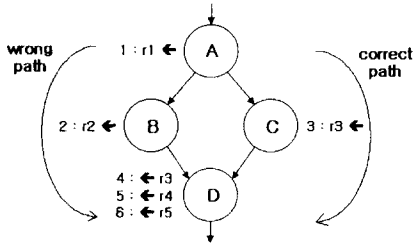
따라서, 본 논문에서는 프로그램의 구조를 사용하여 제어 독립적인 부분을 탐지함으로써 아주 적은 하드웨어의 추가로 기존의 방법보다 우수한 성능을 보여주는 방법을 제안한다.

3. 제어독립성을 이용한 분기예상 실패 복구 메커니즘

3.1 제어 독립적인 명령어

분기 명령어의 결과에 따라 분기 방향이 결정되고, 분기 방향에 따라 분기 명령어 이후에 수행될 명령어가 결정된다. 분기 명령어의 결과에 종속되는 명령어를 제어 종속적인 명령어라 하고, 반대로 분기 명령어의 결과에 상관없이 항상 수행되는 명령어를 제어 독립적인 명령어라고 한다.

[그림1]의 블록 A에 있는 분기 명령어가 예상되어 블록 B,D가 수행됐다고 가정하자. 이후에 분기 명령어가 완료되어 예상이 잘못된 것으로 판명되면, 블록 B,D에 있는 명령어를 무효화시키고, 블록 C,D를 반입하여 수행시켜야 한다. 그러나 분기 방향에 상관없이 수행되는 블록 D에 있는 모든 명령어를 무효화시킬 필요가 없고, 단지 3번 명령어와 데이터 종속관계를 가지고 있는 4번 명령어만 재반입 및 수행시키면 된다.

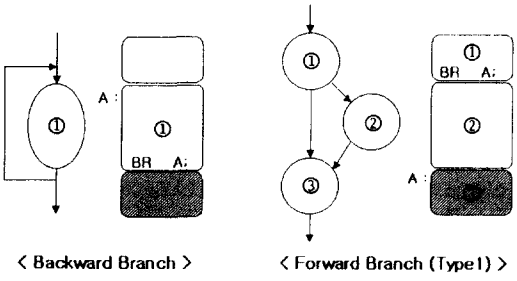


[그림1] 제어 독립적 명령어의 흐름도

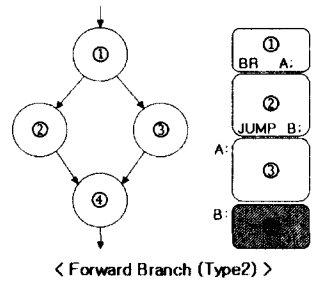
3.2 프로그램 상에서 제어 독립적인 명령어의 탐지

본 논문에서 제안한 기법은 프로그램의 순서상에서 보여지는 패턴을 이용해 제어 독립적인 명령어를 탐지해 낸다. 즉, 컴파일 시 각각의 분기 명령어에 해당 분기의 목적 주소 정보를 첨가시킨다. 이렇게 정적으로 구해진 정보를 이용해 프로그램의 수행시 동적으로 분기 명령어에 대응하는 제어 독립적인 명령어를 탐지한다.

[그림2]는 반복문과 같이 프로그램의 앞쪽으로 분기하는 Backward Branch의 경우, 분기 명령어의 결과에 상관없이 항상 수행되는 명령어 즉, 제어 독립적인 명령어는 Backward Branch의 다음 명령어가 된다. 뒤쪽으로 분기를 하지만 단방향 Forward Branch(Type1) 경우의 제어 독립적인 명령어는 분기 명령어의 목적 명령어(target instruction)가 된다.



[그림2] 프로그램 상에서의 제어 독립성(1)



[그림3] 프로그램 상에서의 제어 독립성(2)

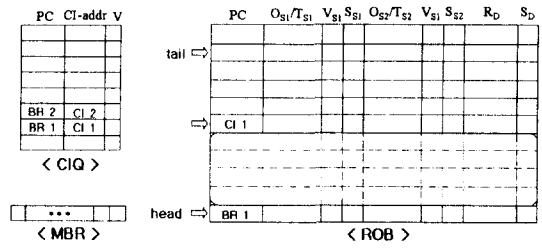
[그림3]의 Forward Branch(Type2)는 분기가 양방향으로 나타나는 유형이다. 이런 경우에는 목적 명령어의 바로 이전의 명령어가 무조건 분기 명령어가 된다. 따라서, Forward Branch(Type2)의 제어 독립적인 명령어는 무조건 분기 명령어의 목적 명령어가 된다.

위와 같은 방법으로 구해진 정보를 이용해 동적으로 조건 분기 명령어의 제어 종속적인 명령어를 찾아낼 수 있다.

3.3 분기 예상 실패 복구 메커니즘의 구현

분기 명령어가 반입될 때 분기 예상을 위해 예측기에 접근하는 동시에 정적으로 구해진 분기 명령어와 제어 독립적인 명령어의 정보와 분기 명령어의 지시자를 [그림4]의 CIQ(Control Independence Queue)에 입력한다. 예측기에서 구해진 분기 방향과 목적 명령어로 분기해서 수행을 진행한 후, 분기 명령어의 수행이 완료되었을 때 분기 예상의 정확성 여부를 확인한다.

예상된 분기가 틀렸을 경우에는 분기 명령어 이후의 명령어를 ROB에서 제거하고 올바른 분기 방향의 명령어를 반입해야 한다. 이때, 모든 명령어를 제거하지 않고 해당 분기 명령어에서부터 CI로 표시되어 있는 명령어 이전의 제어 종속적인 명령어만을 제거한다.



[그림4] 분기 예상 복구 메커니즘의 구조

잘못 예상된 분기의 명령어들이 제거될 때 해당 분기 명령어에 대한 MBR(Mispredicted Branch Register)의 지시자를 세트시킨다. 올바른 분기 방향의 명령어 수행이 완료된 후에 이전에 제거되지 않았던 제어 독립적인 명령어는 MBR을 검색해서 제어 종속적인 명령어와 데이터 종속관계가 있는지의 여부를 조사한다. 이는 데이터 종속관계가 있는 명령어는 재수행을 시켜야하기 때문이다.

이와 같은 방법으로 분기 명령어의 예상이 잘못되었을 경우에 불필요하게 제거되는 명령어를 줄임으로써 하드웨어 자원의 낭비를 줄일 수 있고, 프로세서의 성능을 향상시킬 수 있다.

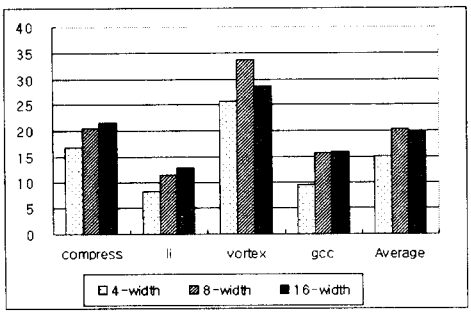
4. 실험 결과

4.1 실험 환경

본 절에서는 분기 예상이 틀렸을 경우 모든 명령어를 제거하는 기존의 방법과 선택적으로 제어 종속적인 명령어만을 제거하는 본 논문에서 제안한 방법을 비교 분석한다. 실험은 명령어 레벨 시뮬레이터인 SimpleScalar 3.0 툴셋[7] 중 MIPS-4 명령어 셋을 기반으로 비순서적(out-of-order)인 방법으로 시뮬레이션을 수행하였다.

4.2 제어 독립적인 명령어의 비율

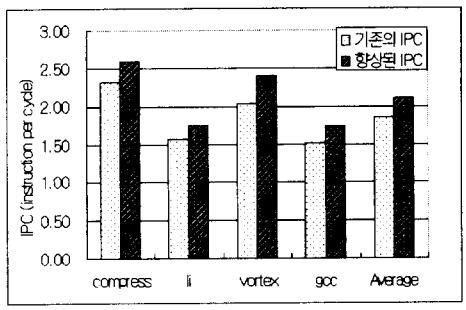
[그림5]는 분기 예상기법에 의한 분기 예상이 틀렸을 경우 무효화되는 명령어 중에서 제어 독립적인 명령어가 차지하는 비율을 나타낸다. 8-width 프로세서에서 전체 무효화되는 명령어 중 평균 20%의 명령어가 제어 독립적인 명령어이다.



[그림5] 제어 종속적인 명령어의 무효화 비율

4.3 제어 독립성의 성능 향상

[그림5]의 결과에서 20%의 제어 독립적인 명령어는 무효화되지 않으므로 이후에 다시 반입되지 않고 수행되는 명령어이다. 따라서, 반입 및 수행되는 명령어를 줄일 수 있고 전체적인 IPC(Instruction Per Cycle)를 향상시킬 수 있다. [그림6]은 8-width 프로세서에서 기존 방법과 본 논문에서 제안한 방법의 IPC를 비교하였다. 본 논문에서 제안한 기법이 11~18%, 평균 14%의 성능 향상을 보이고 있다.



[그림6] 8-width 프로세서의 성능 향상

5. 결론

슈퍼스칼라 프로세서에서 명령어 수준 병렬성을 향상시키기 위해서는 분기 예상 기법이 필수적이다. 그러나 분기 예상 실패시 발생하는 페널티는 프로세서의 성능에 심각한 영향을 준다.

본 논문에서는 컴파일 시 프로파일링을 통한 정적인 방법과 프로그램상의 제어 흐름을 통해 동적으로 제어 독립적인 명령어를 탐지함으로써 분기 명령어의 잘못된 예상으로 인해 제거되는 명령어를 효과적으로 감소시켜 프로세서의 성능을 향상시키는 메커니즘을 제안하였다.

6. 참고 문헌

[1] E. Jacobsen, E Rotenberg and J. E. Smith, "Assigning Confidence to Conditional Branch Predictors", in Proc. of 29th MICRO, pp 142-131, 1998.

[2] D. I .August, W. W. Hwu and S. A. Mahlke, "A Framework for Balancing Control Flow and Predication", in Proc. of 30th MICRO, pp 92-103, 1997.

[3] T. Heil and J. Smith, "Selective Dual Path Execution", Univ. of Wisconsin-Madison, Nov. 1996.

[4] Artur Klauser, "Reducing Branch Misprediction Penalty through Multipath Execution", Ph.D. Thesis, Univ. of Colorado at Boulder, May 1999.

[5] E. Rotenberg, Q. Jacobson and J. Smith, " A Study of Control Independence in Superscalar Processors", in Proc. of 5th HPCA, 1999.

[6] Y. Chou, J. Fung, and John P. Shen, "Reducing Branch Misprediction Penalties Via Dynamic Control Independence Detection", ICS-99, June 1999.

[7] D. Burger, T.M. Austin, S. Bennett, "Evaluating Future Microprocessors: The SimpleScalar Tool Set", CS-TR-96-1308, Univ. of Wisconsin, 1996.