(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2011년08월17일
(11) 등록번호 10-1057163
(24) 등록일자 2011년08월09일

(51) Int. Cl.
G06F 9/38 (2006.01)
(21) 출원번호 10-2005-7020786
(22) 출원일자(국제출원일자) 2004년01월09일
심사청구일자 2008년12월30일
(85) 번역문제출일자 2005년11월01일
(65) 공개번호 10-2006-0004974
(43) 공개일자 2006년01월16일
(86) 국제출원번호 PCT/US2004/000483
(87) 국제공개번호 WO 2004/099978
국제공개일자 2004년11월18일
(30) 우선권주장
10/429,159 2003년05월02일 미국(US)
(56) 선행기술조사문헌
WO2001061480 A1*
WO2002042902 A2*
US5781752 A
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
글로벌파운드리즈 인크.
케이만 아일랜드 케이와이1-1104 그랜드 케이만 어그랜드 하우스 피.오.박스 309 메이플즈 코포레이트 서비시즈 리미티드
(72) 발명자
필리포 마이클 에이.
미국 텍사스 78652 만차카 차파랠 로드 2030
피켓 제임스 케이.
미국 텍사스 78733 오스틴 팔로미노 리지 드라이브 1700 #2
샌더 벤자민 티.
미국 텍사스 78735 오스틴 메디슨 크릭 5701
(74) 대리인
박장원

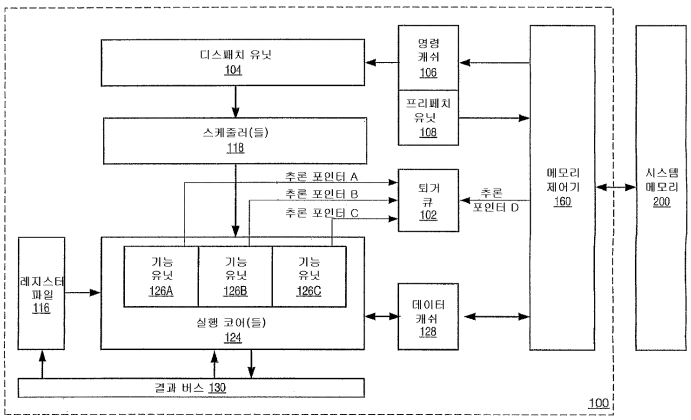
전체 청구항 수 : 총 10 항 심사관 : 박지은

(54) 마이크로프로세서에서 데이터-추론 연산들을 식별하는 추론포인터들

(57) 요 약

마이크로프로세서(100)는 퇴거 큐(102)와 하나 이상의 데이터 추론 검증 유닛을 포함할 수 있다. 상기 데이터 추론 검증 유닛들 각각은 연산 상에서 수행되는 데이터 추론을 검증하도록 구성된 것이다. 각 데이터 추론 검증 유닛은 데이터 추론이 데이터 추론 검증 유닛에 의해 검증되는 미결 연산들을 식별하는 각 추론 포인터를 생성한다. 상기 퇴거 큐(102)는 연산들을 선택적으로 퇴거시키도록 구성되며, 여기서 연산들은 상기 데이터 추론 검증 유닛들 각각으로부터 수신한 상기 추론 포인터에 의존된다.

대 표 도



특허청구의 범위

청구항 1

마이크로프로세서(100)로서,

하나 이상의 데이터 추론 검증 유닛들과, 여기서 상기 하나 이상의 데이터 추론 검증 유닛들은 연산들에 대해 수행되는 데이터 추론을 검증하고 그리고 상기 하나 이상의 데이터 추론 검증 유닛들 중 하나에 의해 데이터 추론이 검증된 미결의 연산들(outstanding operations)을 식별하는 각각의 추론 포인터를 생성하며; 그리고

상기 하나 이상의 데이터 추론 검증 유닛들 각각으로부터 추론 포인터를 수신하도록 연결된 퇴거 큐(retire queue)(102)를 포함하며, 여기서 상기 퇴거 큐(102)는 상기 하나 이상의 데이터 추론 검증 유닛들 각각으로부터 수신되는 상기 추론 포인터에 의존하여 연산들을 선택적으로 퇴거시키는 것을 특징으로 하는 마이크로프로세서(100).

청구항 2

제 1항에 있어서, 로드 저장 유닛(126)이 상기 하나 이상의 데이터 추론 검증 유닛들 중 하나를 포함하며, 상기 로드 저장 유닛(126)에 포함되는 상기 데이터 추론 검증 유닛은 의존성 예측을 검증하는 것을 특징으로 하는 마이크로프로세서(100).

청구항 3

제 1항에 있어서, 비-메모리 연산들을 실행하는 기능 유닛(126)이 상기 하나 이상의 데이터 추론 검증 유닛들 중 하나를 포함하며, 상기 기능 유닛(126)에 포함되는 상기 데이터 추론 검증 유닛은 비-메모리 연산의 결과를 예측하는 데이터 예측을 검증하는 것을 특징으로 하는 마이크로프로세서(100).

청구항 4

제1 항에 있어서,

상기 하나 이상의 데이터 추론 검증 유닛들 중 하나는, 상기 하나 이상의 데이터 추론 검증 유닛들 중 하나에 의해 검증되는 타입의 데이터 추론이 수행된 복수의 미결 연산들 각각을 식별하는 정보를 수신하도록 되어있으며, 그리고

상기 하나 이상의 데이터 추론 검증 유닛들 중 하나는, 데이터 추론이 정확한 것으로서 검증된 최신의 미결 연산(the youngest outstanding operation)보다는 나중의 연산(younger operatin)이면서, 상기 타입의 데이터 추론이 수행되었으되 아직 정확한 것으로서 검증되지 않은 가장 오래된 미결 연산(the oldest outstanding operation)보다는 나중의 것이 아닌(not younger) 연산을 식별하도록 각각의 추론 포인터를 진행시킴으로써, 어떤 미결 오퍼레이션들이 상기 하나 이상의 데이터 추론 검증 유닛들 중 하나에 의해 정확한 것으로서 검증되었는지를 식별하도록 되어있는 것을 특징으로 하는 마이크로프로세서(100).

청구항 5

컴퓨터 시스템(900)으로서,

메모리(200)와; 그리고

상기 메모리(200)에 연결된 프로세서(100)를 포함하여 구성되며;

여기서 상기 프로세서(100)는 하나 이상의 데이터 추론 검증 유닛을 포함하며, 상기 하나 이상의 데이터 추론 검증 유닛은 연산들에 대해 수행되는 데이터 추론을 검증하고 그리고 상기 하나 이상의 데이터 추론 검증 유닛에 의해 데이터 추론이 검증된 미결의 연산들을 식별하는 각각의 추론 포인터를 생성하며; 그리고

상기 프로세서(100)는 퇴거 큐(102)를 더 포함하며, 상기 퇴거 큐(102)는 상기 하나 이상의 데이터 추론 검증 유닛들 각각으로부터 추론 포인터를 수신하도록 연결되며 상기 하나 이상의 데이터 추론 검증 유닛들 각각으로부터 수신된 상기 추론 포인터에 의존하여 연산들을 선택적으로 퇴거시키는 것을 특징으로 하는 컴퓨터 시스템(900).

청구항 6

제 5항에 있어서, 로드 저장 유닛(126)이 상기 하나 이상의 데이터 추론 검증 유닛들 중 하나를 포함하며, 상기 로드 저장 유닛(126)에 포함되는 상기 데이터 추론 검증 유닛은 의존성 예측을 검증하는 것을 특징으로 하는 컴퓨터 시스템(900).

청구항 7

제 5항에 있어서, 비-메모리 연산들을 실행하는 기능 유닛(126)이 상기 하나 이상의 데이터 추론 검증 유닛들 중 하나를 포함하며, 상기 기능 유닛(126)에 포함되는 상기 데이터 추론 검증 유닛은 비-메모리 연산의 결과를 예측하는 데이터 예측을 검증하는 것을 특징으로 하는 컴퓨터 시스템(900).

청구항 8

마이크로프로세서에서 데이터 추론이 수행된 연산들을 선택적으로 퇴거시키는 방법으로서, 상기 마이크로프로세서는 검증 유닛 및 퇴거 큐(retire queue)를 포함하며,

검증 유닛이, 연산에 대해 수행된 데이터 추론이 정확하다는 것을 검증하는 단계와;

상기 검증 유닛이, 상기 연산이 상기 검증 유닛에 대하여 데이터-추론적(data-speculative)이 아님을 표시하는 추론 포인터를 생성하는 단계와; 그리고

상기 퇴거 큐가, 상기 검증 유닛에 의해 생성된 상기 추론 포인터를 수신함에 응답하여, 상기 오퍼레이션을 선택적으로 퇴거시키는 단계를 포함하는 것을 특징으로 하는 마이크로프로세서에서 데이터 추론이 수행된 연산들을 선택적으로 퇴거시키는 방법.

청구항 9

제 8항에 있어서, 상기 마이크로프로세서는 다른 검증 유닛(other verification unit)을 더 포함하며,

상기 마이크로프로세서에서 데이터 추론이 수행된 연산들을 선택적으로 퇴거시키는 방법은,

상기 다른 검증 유닛이 다른 연산(other operation)에 대해 수행되는 데이터 추론을 검증하는 단계와; 그리고

상기 다른 검증 유닛이 상기 다른 연산이 상기 다른 검증 유닛에 대하여 데이터-추론적이 아님을 표시하는 다른 추론 포인터(other operation pointer)를 생성하는 단계를 더 포함하며;

상기 연산을 퇴거시키는 단계는 상기 추론 포인터 및 상기 다른 추론 포인터 모두에 의존하는 것을 특징으로 하는 마이크로프로세서에서 데이터 추론이 수행된 연산들을 선택적으로 퇴거시키는 방법.

청구항 10

제 8항에 있어서, 상기 연산이 상기 검증 유닛에 대하여 데이터-추론적이 아님을 표시하는 추론 포인터를 생성하는 단계는, 상기 연산이 상기 검증 유닛에 의해 검증되는 임의의 타입의 데이터 추론이 수행된 최신의 연산임에 의존하는 것을 특징으로 하는 마이크로프로세서에서 데이터 추론이 수행된 연산들을 선택적으로 퇴거시키는 방법.

명 세 서

기 술 분 야

[0001] 본 발명은 마이크로프로세서 분야에 관한 것으로서, 특히 마이크로프로세서에서 데이터 추론을 실행하는 방법에 관한 것이다.

배 경 기 술

[0002] 슈퍼스칼라 마이크로프로세서들은, 다수의 명령(instructions)을 동시에 수행함으로써, 그리고 상기 마이크로프로세서들의 설계에 부합하는 가능한한 가장 짧은 클록 사이클을 이용함으로써 고성능을 달성한다. 그러나, 명령들 간의 데이터 및 제어 흐름 의존성들은 임의의 주어진 시간에 얼마나 많은 명령들을 발행할 수 있을 지를 제약할 수 있다. 결과적으로, 일부 마이크로프로세서들은, 부가적인 성능 이득을 달성하기 위하여 추론적 실행

(speculative execution)을 지원한다.

[0003] 추론의 일 타입은 제어 흐름 추론이다. 제어 흐름 추론은 프로그램 제어가 진행하는 방향을 예측하는 것이다. 예를 들어, 분기 예측(branch prediction)이 사용되어 어느 분기가 택해질(taken) 것인지를 예측할 수 있다. 많은 타입의 분기 예측들이 이용될 수 있는데, 그 타입의 범위는 매시간 단순히 동일한 예측을 하는 방법들로부터 프로그램에서 이전 분기들의 정교한 이력(history)을 유지하여 이력-기반 예측을 하는 방법들에까지 이른다. 분기 예측은 하드웨어 최적화, 컴파일러 최적화, 또는 이들 모두를 통해 용이해질 수 있다. 상기 분기 예측 메커니즘에 의해 제공되는 예측을 기반으로, 명령들을 추론하여 페치(fetch) 및 실행할 수 있다. 상기 분기 명령이 최종적으로 평가(evaluation)될 때에, 상기 분기 예측은 검증될 수 있다. 만약 상기 예측이 부정확하다면, 그 부정확한 예측에 기반하여 추론적으로 실행되었던 모든 명령들이 폐기될 수 있다.

[0004] 추론의 또 다른 타입으로서 데이터 추론이 있는데, 이는 데이터 값들을 예측하는 것이다. 제안된 데이터 추론 타입들은 메모리 연산들의 주소들을 추론적으로 생성하는 것과, 컴퓨터 연산들에서 사용하는 데이터 값들을 추론적으로 생성하는 것을 포함한다. 제어추론의 경우와 마찬가지로, 값을 추론 생성하는데 사용되는 근본 조건들(underlying conditions)이 최종적으로 평가되며, 이에 따라 추론이 검증(verify)되게 하거나 취소(undo)되게 한다.

발명의 상세한 설명

[0005] 어떤 연산들이 마이크로프로세서 내의 여러가지 검증 유닛들에 대하여 데이터 추론적인지를 식별하는 추론 포인터들에 의존하여 연산들을 퇴거(retirement)시키는 여러 가지 방법들 및 시스템들의 실시 예들이 개시된다. 본 발명의 일 실시 예에서, 마이크로프로세서는 퇴거 큐(retire queue) 및 하나 이상의 데이터 추론 검증 유닛들을 포함할 수 있다. 상기 데이터 추론 검증 유닛들 각각은 연산들 상에서 수행되는 데이터 추론을 검증한다. 각 데이터 추론 검증 유닛은 상기 데이터 추론 검증 유닛에서 데이터 추론이 검증된 미결 연산들(outstanding operations)을 식별하는 각각의 추론 포인터를 생성한다. 상기 퇴거 큐는 연산들, 즉 상기 추론 검증 유닛들 각각으로부터 수신한 추론 포인터에 의존하여 연산들을 선택적으로 퇴거시킨다.

[0006] 본 발명의 일 실시 예에서, 데이터 추론 검증 유닛 중 하나는 로드 저장 유닛에 포함될 수 있다. 이러한 데이터 추론 검증 유닛은 의존성 예측, 주소 예측, 및/또는 데이터 예측과 같은 데이터 예측 타입을 검증할 수 있다. 예컨대, 로드 저장 유닛에 포함된 데이터 추론 검증 유닛은 나중의(younger) 로드 연산이 계산되지않은 주소를 갖는 오래된(older) 저장 연산에 의존되지 않음을 예측하는 의존성 예측을 검증하도록 구성될 수 있다. 유사하게는, 로드 저장 유닛에 포함되는 데이터 추론 검증 유닛은 나중의 로드 연산이 오래된 저장 연산의 소스에 일치할 것이라 예측하는 의존성 예측을 검증하도록 구성될 수 있다.

[0007] 다른 하나의 데이터 추론 유닛은 비-메모리 연산을 실행하는 기능 유닛에 포함될 수 있다. 이러한 데이터 추론 검증 유닛은 비-메모리 연산의 결과를 예측하는 데이터 예측을 검증할 수 있다. 또 다른 데이터 추론 검증 유닛은 메모리 제어기에 포함되며, 메모리 예측을 검증할 수 있다.

[0008] 일 실시예에서, 데이터 추론 검증 유닛 중 하나는 상기 데이터 추론 검증 유닛에 의해 검증되는 일 타입의 데이터 추론이 수행된 각각의 미결 연산을 식별하는 정보를 수신할 수 있다. 이 데이터 추론 검증 유닛은 어느 미결 연산이 이 데이터 추론 검증 유닛에 의해 검증되었는지를 식별할 수 있는바, 이는 일 타입의 데이터 추론이 검증된 최신의 미결 연산보다 새로우며, 일 타입의 데이터 추론이 수행된 다른 미결 연산보다 오래된 연산을 식별하기 위해 자신의 각 추론 포인터를 진행함으로써 식별된다. 만일 어떠한 연산도 현재 이 데이터 추론 검증 유닛에 대해 데이터-추론적인 것으로서 식별되지 않는 경우에, 이 데이터 추론 검증 유닛은 현재 어떠한 미결 연산도 이 데이터 추론 검증 유닛에 대하여 데이터-추론적이 아님을 표시하기 위해 자신의 각 추론 포인터의 값을 설정할 수 있다.

[0009] 퇴거 큐가 여러가지 추론 포인터를 수신하는 실시예에서, 퇴거 큐는, 연산이 모든 추론 포인터에 의해 비-데이터-추론적(non-data-speculative)인 것으로서 식별되는 최신 연산보다 오래된 것인지를 결정함으로써 상기 연산이 퇴거가능한지를 결정할 수 있다.

[0010] 추론 포인터에 의존하여 연산을 퇴거시키는 퇴거 큐, 및 추론 포인터를 생성하는 하나 이상의 데이터 추론 검증 유닛을 포함하는 마이크로프로세서가 컴퓨터 시스템에 포함될 수 있다.

[0011] 일부 실시예에서, 방법은: 연산에 대한 데이터 추론을 수행하는 단계와; 검증 유닛이 연산에 대해 수행되는 데이터 추론을 검증하는 단계와; 상기 검증 유닛이 상기 검증에 응답하여 연산이 상기 검증 유닛에 대하여 데이터

-추론적이 아닌 것으로서 표시하는 추론 포인터를 생성하는 단계와; 연산이 검증 유닛에 대하여 데이터 추론적이 아닌 것으로 표시하는 추론 포인터에 응답하여, 상기 연산을 퇴거시키는 단계를 포함할 수 있다.

[0012] 상기 방법은 또한 다른 연산들에 대해 수행되는 데이터 추론을 검증하고 다른 연산들이 이 검증 유닛에 대하여 데이터-추론적이 아님을 표시하는 다른 추론 포인터를 생성하는 하나 이상의 다른 검증 유닛을 포함할 수 있다. 연산을 퇴거시키는 것은 모든 추론 포인터에 의존될 수 있다. 각 검증 유닛은 다양한 타입의 데이터 추론을 검증할 수 있다.

실 시 예

[0020] 도 1은 마이크로프로세서(100)의 하나의 실시 예를 보여주는 블록도이다. 마이크로프로세서(100)는 시스템 메모리(200)에 저장된 명령들을 실행하도록 구성된 것이다. 이러한 많은 명령들은 시스템 메모리(200)에 저장된 데이터에 대해 연산한다. 주목할 점으로서, 시스템 메모리(200)는 컴퓨터 시스템 도처에 물리적으로 분산될 수 있으며 하나 이상의 마이크로프로세서(100)에 의해 액세스된다.

[0021] 마이크로프로세서(100)는 명령 캐시(106) 및 데이터 캐시(128)를 포함할 수 있다. 마이크로프로세서(100)는 상기 명령 캐시(106)와 연결될 프리페치 유닛(prefetch unit)(108)을 포함할 수 있다. 디스패치 유닛(dispatch unit)(104)은 명령 캐시(106)로부터 명령들을 수신하고 연산들을 스케줄러(들)(118)로 보내도록 구성될 수 있다. 하나 이상의 스케줄러들(118)이 연결되어, 디스패치 유닛(104)에서 보내진 연산들을 수신하여 하나 이상의 실행 코어들(124)에 연산들을 발행할 수 있다. 실행 코어(들)(124) 각각은 로드/저장 유닛을 포함할 수 있는데, 이 로드/저장 유닛은 데이터 캐시(128)에 액세스을 실행하도록 구성된 것이다. 실행 코어(들)(124)에 의해 생성된 결과들은 결과 버스(130)로 출력될 수 있다. 이러한 결과들은 계속해서 발행되는 명령들에 대한 오퍼랜드(operand) 값들로 사용되고 및/또는 레지스터 파일(116)에 저장될 수 있다. 퇴거 큐(102)는 스케줄러(들)(118) 및 디스패치 유닛(104)에 연결될 수 있다. 상기 퇴거 큐(102)는 발행된 각 연산이 언제 퇴거될 수 있는지를 결정하도록 구성될 수 있다. 본 발명의 일 실시 예에서, 상기 마이크로프로세서(100)를 x86 아키텍처와 호환되도록 설계할 수 있다. 또한 알아야 할 것으로, 마이크로프로세서(100)는 수많은 다른 구성요소들을 포함할 수 있다. 예를 들면, 마이크로프로세서(100)는 분기 예측 유닛(미도시)을 포함할 수 있다.

[0022] 명령 캐시(106)는 디스패치 유닛(104)이 명령을 수신하기 이전에 그 명령들을 임시적으로 저장할 수 있다. 명령 코드를 명령 캐시(106)에 제공할 수 있는데, 이는 프리페치 유닛(108)을 통하여 상기 시스템 메모리(200)로부터의 코드를 프리페칭함으로써 이루어진다. 명령 캐시(106)는 여러 가지 구성들(예를 들면, 세트-연관(set-associative), 완전-연관(fully-associative), 직접-맵핑(direct-mapped))으로 실행될 수 있다. 본 발명의 몇몇 실시 예에서, 다중 레벨의 명령 및/또는 데이터 캐시(106 및 108)가 있을 수 있다. 몇몇 레벨의 캐시들은, 도시된 바와 같이, 상기 마이크로프로세서(100)로 통합될 수 있으며, 한편 다른 레벨들의 캐시는 상기 마이크로프로세서의 외부에 있을 수 있다.

[0023] 프리페치 유닛(108)은 상기 시스템 메모리(200)로부터의 명령 코드를 명령 캐시(106) 내에 저장하기 위해 프리페치할 수 있다. 본 발명의 일 실시 예에서, 프리페치 유닛(108)은 상기 시스템 메모리(200)로부터의 코드를 명령 캐시(106)로 버스트(burst)하도록 구성될 수 있다. 프리페치 유닛(108)은 다양한 특정 코드 프리페칭의 기술들 및 알고리즘들을 이용할 수 있다.

[0024] 디스패치 유닛(104)은 실행 코어(들)(124)에 의해 실행가능한 비트-엔코딩된 연산뿐만 아니라 오퍼랜드 주소 정보, 즉시 데이터(immediate data), 및/또는 변위 데이터를 포함하는 신호를 출력할 수 있다. 일부 실시예에서, 디스패치 유닛(104)은 특정 명령을 실행 코어(들)(124) 내에서 실행가능한 연산으로 디코딩하기 위한 디코딩 회로(미도시)를 포함할 수 있다. 단순 명령은 단일 연산에 대응할 수 있다. 일부 실시예에서, 더 복잡한 명령이 복수의 연산에 대응할 수 있다. 만일 연산이 레지스터의 갱신을 포함한다면, 레지스터 파일(116) 내의 레지스터 위치는 추론 레지스터 상태(대안 실시예에서, 재배열 버퍼가 각 레지스터에 대해 하나 이상의 추론 레지스터 상태를 저장하기 위해 사용될 수 있음)를 저장하기 위해 (예를 들어, 이 연산의 디코딩과 동시에) 보유될 수 있다. 레지스터 맵은 레지스터 재명명을 용이하게 하기 위해 소스 및 목적지 오퍼랜드의 논리적 레지스터 명칭을 물리적 레지스터 명칭으로 변환할 수 있다. 레지스터 맵(register map)은 레지스터 파일(116) 내의 레지스터가 현재 할당되었는지를 추적할 수 있다.

[0025] 도 1의 마이크로프로세서(100)는 순서를 벗어난(out-of-order) 실행을 지원한다. 퇴거 큐(102)는 레지스터 판독 및 기입 연산에 대한 원래의 프로그램 시퀀스를 추적하며, 추론 명령 실행 및 분기 오예측 복구를 허용하며, 그리고 정확한 예외를 용이하게 한다. 퇴거 큐(102)는 선입선출 구성으로 구현될 수 있는데, 여기서 연산은 유효

화될 때에 버퍼의 "하부"로 이동하여, 큐의 "상부"에 새로운 엔트리를 위한 공간을 만든다. 퇴거 큐(102)는, 검증될 때에 이 연산 완료 실행 및 임의의 연산들 상에서 수행되는 임의의 데이터 또는 제어 추론, 프로그램 순서에서 이 연산까지 및 이 연산을 포함하는 것에 응답하여 연산을 퇴거시킬 수 있다. 퇴거 큐(102)는 물리적 레지스터에 값을 생성한 연산이 퇴거될 때에 물리적 레지스터의 추론 상태를 마이크로프로세서(100)의 아키텍처 상태에 위임할 수 있다. 일부 실시예에서, 퇴거 큐(102)는 재배열 버퍼의 일부로서 구현될 수 있다. 또한, 이러한 재배열 버퍼는 레지스터 재명명을 지원하기 위해 추론 레지스터 상태을 위한 데이터 값 저장소를 제공할 수 있다. 다른 실시예에서, 퇴거 큐(102)는 어떤 데이터 값 저장소를 제공하지 않을 수 있음을 주목해야 한다. 대신에, 연산이 퇴거될 때에, 퇴거 큐(102)는 더 이상 추론 레지스터 상태를 저장하는데 불필요한 레지스터 파일(116)의 레지스터를 할당해제(deallocate)하고 레지스터들이 현재 비어있는 것으로 표시하는 신호를 레지스터 맵에 제공할 수 있다. 이 상태들을 생성한 연산이 유효화될 때까지, 레지스터 파일(116) 내(또는 대안 실시예에서, 재배열 버퍼 내)의 추론 레지스터 상태를 유지함으로써, 오예측된 경로를 따라 추론으로 실행된 연산들의 결과들은 분기 예측이 부정확한 경우 레지스터 파일(116)에서 무효화될 수 있다.

[0026] 만일 특정 연산의 요구되는 오퍼랜드가 레지스터 위치인 경우에, 레지스터 주소 정보가 레지스터 맵(또는 재배열 버퍼)에 라우팅될 수 있다. 예컨대, x86 아키텍처에서, 8개의 32비트 논리적 레지스터(예를 들어, EAX, EBX, ECX, EDX, EBP, ESI, EDI 및 ESP)들이 있다. 물리적 레지스터 파일(116)(또는 재배열 버퍼)은 이들 논리적 레지스터들의 내용(content)을 변경하는 결과를 위한 저장소를 포함하며, 이는 순서를 벗어난 실행을 허용한다. 레지스터 파일(116)의 물리적 레지스터는 논리적 레지스터들 중 하나의 내용을 수정하기 위해 결정되는 각 연산의 결과를 저장하도록 할당될 수 있다. 따라서, 특정 프로그램을 실행하는 동안의 다양한 지점들에서, 레지스터 파일(116)(또는 대안 실시예에서, 재배열 버퍼)은 소정의 논리적 레지스터의 추론으로 실행되는 내용을 포함하는 하나 이상의 레지스터들을 구비할 수 있다.

[0027] 레지스터 맵은 물리적 레지스터를 연산에 대한 목적지 오퍼랜드로서 명시되는 특정 논리적 레지스터에 할당할 수 있다. 디스패치 유닛(104)은, 레지스터 파일(116)이 소정의 연산에서 소스 오퍼랜드로서 명시된 논리적 레지스터에 지정되는 하나 이상의 이전에 할당된 물리적 레지스터를 갖는 것으로 결정할 수 있다. 레지스터 맵은 가장 최근에 이 논리적 레지스터에 할당된 물리적 레지스터에 태그(tag)를 제공할 수 있다. 이 태그는 레지스터 파일(116)에서 오퍼랜드의 데이터 값에 액세스하거나 결과 버스(130)상에서 결과 포워딩을 통해 데이터 값을 수신하는데 사용될 수 있다. 만일 오퍼랜드가 메모리 위치에 일치한다면, 오퍼랜드 값은 (결과 포워딩 및/또는 레지스터 파일(118)에 저장을 위해) 로드/저장 유닛(222)을 통해 결과 버스상에 제공될 수 있다. 오퍼랜드 데이터 값은 연산이 스케줄러(들)(118) 중 하나에 의해 발행되는 때에 실행 코어(들)(124)에 제공될 수 있다. 대안 실시예에서, 오퍼랜드 값은 (연산이 발행될 때에 대응 실행 코어(124)에 제공되는 대신에) 연산이 디스패치될 때에 대응 스케줄러(118)에 제공될 수 있음을 주목해야 한다.

[0028] 디스패치 유닛(104)의 출력에 제공된 비트-엔코딩 연산 및 즉시 데이터는 하나 이상의 스케줄러(118)에 라우팅될 수 있다. 본원에서 사용된 바와같이, 스케줄러는 연산이 준비되는 때를 검출하여 하나 이상의 기능 유닛에 준비 연산을 발행하는 장치임을 주목해야 한다. 예컨대, 예약 스테이션(prediction station)이 스케줄러이다. 또한, 스케줄러에서나 스케줄러 그룹에서의 연산은 명령 또는 연산 윈도우 또는 스케줄링 윈도우에서 연산으로 지칭될 수 있다. 각 스케줄러(118)는 실행 코어(124)에 발행되기를 대기하는 여러가지 미정의 연산에 대한 연산 정보(예를 들어, 비트 엔코딩 실행 비트뿐만 아니라 오퍼랜드 값, 오퍼랜드 태그, 및/또는 즉시 데이터)를 유지할 수 있다. 일부 실시예에서, 각 스케줄러(118)는 오퍼랜드 값 저장소를 제공하지 않을 수 있다. 대신에, 각 스케줄러는 (레지스터 파일(116) 또는 결과 버스(130)로부터) 기능 유닛(126)에 의해 판독되도록 오퍼랜드 값이 이용가능한 때를 결정하기 위해 레지스터 파일(116)에서 이용가능한 발행된 연산 및 결과를 모니터링할 수 있다. 일부 실시예에서, 각 스케줄러(118)는 전용 기능 유닛(126)에 관련될 수 있다. 다른 실시예에서, 단일 스케줄러(118)가 하나 보다 많은 기능 유닛(126)에 연산을 발행할 수 있다.

[0029] 스케줄러(118)는 실행 코어(들)(124)에 의해 실행될 연산 정보를 임시적으로 저장하도록 제공될 수 있다. 상술된 바와같이, 각 스케줄러(118)는 미정 연산에 대한 연산 정보를 저장할 수 있다. 추가적으로, 각 스케줄러는 이미 실행되었지만 여전히 재발행될 수 있는 연산에 대한 연산 정보를 저장할 수 있다. 실행을 위한 시간에 이용가능해지는 임의의 요구되는 오퍼랜드(들) 값에 응답하여, 연산들은 실행을 위해 실행 코어(들)(124)에 발행된다. 따라서, 연산들이 실행되는 순서는 원래의 프로그램 명령 시퀀스의 순서와 같지 않을 수 있다. 데이터 추론을 포함하는 연산은 상기 데이터 추론이 부정확한 경우 재발행되기 위하여 비-추론이 될 때까지 스케줄러(들)(118)에 남을 수 있다.

[0030] 일 실시예에서, 각 실행 코어(들)는 여러 기능 유닛(126)(예를 들어, 도 1에 도시된 바와같이 기능 유닛(126A

내지 126C))을 포함할 수 있다. 일부 기능 유닛, 예를 들어 (126A)은 덧셈 및 뺄셈의 정수 산술 연산들뿐만 아니라 시프트, 회전, 논리적 연산, 및 분기 연산들을 수행하도록 구성될 수 있다. 다른 기능 유닛, 예를 들어 (126B)은 부동 소수점 연산을 용이하게 하도록 구성될 수 있다. 하나 이상의 기능 유닛, 예를 들어 (126A)은 데이터 캐시(128) 및/또는 시스템 메모리에 저장된 데이터에 액세스하기 위해 로드 및 저장 연산을 수행하는 기능 유닛, 예를 들어 126C에 의해 수행되는 로드 및 저장 메모리 연산에 대한 주소 생성을 수행하도록 구성될 수 있다. 일 실시예에서, 이러한 기능 유닛(126)은 미정 로드 및/또는 저장소에 대한 데이터 및 주소 정보에 대한 여러 저장 위치들을 갖는 로드/저장 버퍼로 구성될 수 있다.

[0031] 하나 이상의 기능 유닛(126)은 또한 분기 예측 유닛에 조건 분기 명령의 실행에 관한 정보를 제공하여, 분기가 오예측된 경우, 상기 분기 예측 유닛은 명령 처리 파이프라인에 입력된 오예측 분기에 후속하는 명령을 플러쉬(flush)하고, 프리페치 유닛(108)으로 방향을 지정(redirect)한다. 이후에, 이 프리페치 유닛(108)은 명령 캐시(106) 또는 시스템 메모리(200)로부터 정확한 명령 세트를 페칭하기 시작할 수 있다. 이러한 환경에서, 오예측 분기 명령 이후에 발생한 원래의 프로그램 시퀀스에서 명령들의 결과는 버려질 수 있는데, 이는 추론적으로 실행되고 레지스터 파일(116)에 임시적으로 저장된 명령들을 포함한다.

[0032] 실행 코어(들)(124) 내의 기능 유닛(126)에 의해 발생된 결과들은 레지스터 값이 갱신되는 경우 레지스터 파일(116)로의 결과 버스(130) 상에 출력될 수 있다. 만일 메모리 위치의 내용이 변경되는 경우에, 실행 코어(들)(124) 내에 발생된 결과들은 로드/저장 유닛(126C)에 제공될 수 있다.

[0033] 데이터 캐시(128)는 실행 코어(들)(124)와 시스템 메모리(200) 사이에서 전달되는 데이터를 임시적으로 저장하도록 제공되는 캐시 메모리이다. 상술된 명령 캐시(106)와 유사하게도, 데이터 캐시(128)는 세트 연관 구성을 포함하는, 여러가지 특정 메모리 구성에서 구현될 수 있다. 추가적으로, 데이터 캐시(128) 및 명령 캐시(106)는 일부 실시예에서 단일 캐시에서 구현될 수 있다.

[0034] 일부 실시예에서, 마이크로프로세서(100)는 통합된 메모리 제어기(160)를 포함하는데, 이는 마이크로프로세서가 직접적으로 시스템 메모리(200)에 인터페이싱하도록 허용한다. 다른 실시예에서, 메모리 제어기(16)는 마이크로프로세서(100)를 시스템 메모리(200)에 간접적으로 연결하는 버스 브릿지에 포함될 수 있다.

[0035] 데이터 추론

[0036] 본원에서 설명된 바와같이, 데이터 값이 부정확하며, 결과적으로 재연산되어야 할 것으로 발견될 가능성이 있는 경우에 상기 데이터 값은 추론적(speculative)이다. 추론적 데이터 값은 정확하거나 부정확한 것으로 확실하게 식별될 수 없는 것이다. 데이터 값은, 그 데이터 값이 어떤 데이터 추론이 수행된 연산의 결과인 경우 또는 그 데이터 값이 다른 추론적 데이터 값에 의존하는 경우(예를 들어, 데이터 값이 하나 이상의 추론 오퍼랜드를 갖는 연산의 결과로서 생성되는 경우) 재연산 될 수 있다.

[0037] 마이크로프로세서(100) 내의 다양한 메커니즘들이 데이터 추론을 수행할 수 있다. 예컨대, 디스패치 유닛(104), 메모리 제어기(160), 및/또는 하나 이상의 기능 유닛(126) 각각이 특정 연산에 대한 데이터 추론을 수행할 수 있다. 디스패치 유닛(104)은 한 연산의 결과가 다른 연산에 대한 추론적 오퍼랜드로서 사용될 수 있다는 것을 검출할 수 있다. 예컨대, 디스패치 유닛은 로드 연산이 전의 저장 연산에 의해 데이터 캐시(128)에 저장된 데이터를 액세스할 것임을 예측할 수 있다. 디스패치 유닛(104)은 이에 응답하여 로드 연산의 추론적 결과로서 저장 연산의 소스로서 사용된 레지스터에 저장된 데이터 값을 식별할 수 있다. 이러한 타입의 데이터 추론은 본원에서 의존성 예측(dependency prediction)으로서 지칭된다. 의존성 예측은, 저장 연산의 소스를 로드 연산의 결과를 오퍼랜드로서 명시한 연산들에 대한 추론적 오퍼랜드 소스로서 링크함으로써 디스패치 유닛(104)에서 확장될 수 있다. 다른 타입의 의존성 예측은 특정 로드가 계산되지않은 주소를 갖는 저장을 우회(bypass)하게 함으로써, 즉 나중의 로드(younger load)가 전의 저장에 의존하지 않음을 예측함으로써 로드 저장 유닛(126C)에서 수행될 수 있다.

[0038] 멀티프로세서 시스템에서, 메모리 제어기(160)는 캐시 코히어런시(coherency)를 유지하기 위해 코히어런시 검사를 수행할 수 있다. 메모리 제어기(160)는 다른 마이크로프로세서의 캐시와의 코히어런시 검사가 완료되기 이전에 시스템 메모리(200)로부터의 캐시 라인의 사본을 추론적으로 리턴할 수 있다. 만일 코히어런시 검사가 후속적으로 검색할 캐시 라인의 정확한 사본이 현재 다른 프로세서의 캐시에 저장되어있다고 결정하는 경우에, 시스템 메모리(200)로부터 추론적으로 검색된 캐시 라인의 상기 사본은 무효화 될 수 있다. 따라서, 그 캐시 라인에 액세스하여 생성된 임의의 로드 연산 결과들은 코히어런시 검사가 종료될 때까지 추론적일 것이다. 이러한 타입의 추론은 본원에서 메모리 예측으로서 지칭된다.

[0039] 디스패치 유닛(104)은 연산 결과를 예측함으로써 데이터 추론을 수행할 수 있다. 예컨대, 몇몇 연산들은 동일한 결과를 생성하는 경향이 있으며, 따라서 이들 연산들 중 하나가 처리될 때마다, 그 결과는 기능 유닛(126)에 의한 연산의 실제 실행 전에 디스패치 유닛(104)에 의해 추론적으로 생성될 수 있다. 이러한 타입의 데이터 추론은 본원에서 데이터 예측으로서 지칭된다. 데이터 예측이 또한 마이크로프로세서의 다른 부분(예를 들어, 로드 저장 유닛(126C))에서 수행될 수 있음에 주목해야 한다.

[0040] 로드 저장 유닛(126C)은 추론적 주소에 근거한 주소, 및 조기에 처리된(earlier-handled) 로드들의 패턴에 근거하여 그 주소가 아직 연산되지않은 로드 명령의 결과를 추론적으로 생성할 수 있다. 예컨대, 이전의 N 로드 연산이 서로에 대해 일정한 오프셋(C)으로 이격된 타겟 주소(A1 내지 AN)(예를 들어, A1; A2=A1+C;...; AN=A(N-1)+C)를 갖는 경우, 로드 저장 유닛(126C)은 로드 연산의 결과로서 가장 최근에 액세스된 주소 AN에 일정한 오프셋(C)을 더한 곳에서 데이터를 추론으로 복귀시킬 수 있다. 이러한 타입의 데이터 추론은 본원에서 어드레스 예측으로서 지칭된다. 다른 형태의 주소 예측이 다른 실시예에서 이용될 수 있음을 주목해야 한다.

[0041] 또한, 데이터 추론이 수행된 연산 결과에 의존하는 연산들은 추론 결과를 생성할 수 있다. 예컨대, 만일 주소 예측이 로드 연산의 추론 결과를 생성하는데 사용되는 경우, 로드 추론 결과를 오퍼랜드로서 사용하여 실행하는 임의의 의존 연산은 추론 결과를 발생할 수 있는바, 이는 차례로 다른 의존 연산에 의해 오퍼랜드로서 사용될 수 있다. 따라서, 로드 연산에서 만일 기본적인 추론이 부정확한 것으로서 결정된다면, 의존 연산의 결과도 또한 부정확할 수 있으며, 따라서 정확한 결과를 생성하기 위하여 상기 로드에 의존하는 연산의 전체 의존성 체인이 재실행될 수 있다. 반면에, 만일 근본적인 추론이 정확한 것으로 발견되는 경우, 의존 연산의 결과는 (이 결과가 임의의 다른 추론값에 기초하지 않은 것으로 가정한 경우에) 정확할 수 있다.

[0042] 데이터 추론이 수행된 많은 연산들은 이들 연산들이 기능 유닛에 의해 실행되는 때에 검증될 수 있다. 예컨대, 연산 결과를 추론적으로 생성하는데 사용되는 데이터 예측은 실제 연산 결과를 추론 결과와 비교함으로써 이 연산을 실행하는 기능 유닛(126)에 의해 검증될 수 있다. 정확한 결과가 이미 이용가능하므로, 데이터 예측이 잘못되어 있어도, 그러한 연산은 재실행될 필요가 없다. 다른 연산들은 완전하게 실행됨이 없이도 검증될 수 있다. 예컨대, 연산되지 않은 주소를 갖는 로드가 (예를 들어, 의존성 또는 주소 예측 때문에) 이전 저장으로부터 그 결과를 포워딩하는 경우, 로드의 추론 결과는 로드 주소가 계산될 때에 검증될 수 있다. 만일 데이터 추론이 부정확한 경우, 이러한 연산은 정확한 결과를 생성하기 위해 (적어도 부분적으로) 재실행될 필요가 있다.

[0043] 데이터 추론이 수행된 연산들 및 상기 연산들에 의존하는 연산들이 재실행될 필요가 있기 때문에, 퇴거 큐(102)는 임의의 근본적 데이터 추론(underlying data speculation)이 해결된 연산만을 퇴거시키도록 구성될 수 있다. 도 1에 도시된 바와같이, 데이터 추론을 검증하기 위한 각각의 수단(본 실시예에서, 메모리 제어기(160) 및 기능 유닛(126))은 데이터 추론이 검증된 연산들을 식별하는 추론 포인터를 퇴거 큐(102)에 제공하도록 구성될 수 있다. 각 추론 포인터는, 이 검증 수단에 의해 검증된 최신 연산의 태그에 일치하는 값을 가짐으로써 특정 검증 수단에 대하여 데이터 추론이 검증된 연산들을 식별할 수 있다. 퇴거 큐(102)는 어느 연산들이 퇴거될 수 있는지를 식별하기 위해 추론 포인터를 사용할 수 있다. 하나 이상의 타입의 데이터 추론을 검증하도록 구성된 마이크로프로세서(100) 내의 다양한 요소들이 본원에서 데이터 추론 검증 유닛으로서 지칭된다.

[0044] 각 추론 포인터는 기본적인 명령 스트림에서 어느 포인트가 마이크로프로세서의 특정 부분에 대하여 비-추론적(non-speculative)인지를 식별할 수 있다. 예컨대, 메모리 제어기(160)에 의해 생성되는 추론 포인터(D)는, 메모리 예측을 검증하는 메모리 제어기(160)에 대한 최신의 비-추론적 연산(yongest non-speculative operation)을 식별할 수 있다. 일 실시예에서, 메모리 제어기(160)는 메모리 예측이 수행된 가장 최근에 검증된 연산을 포인팅하도록 추론 포인터(D)를 생성할 수 있다. 다른 실시예에서, 메모리 제어기(160)는 메모리 예측이 수행된 가장 오래된 검증되지 않은 연산 직전의 연산을 포인팅하도록 추론 포인터(D)를 생성할 수 있다. 일반적으로, 추론 포인터(D)는 어떤 연산들이 메모리 제어기(160)의 관점에서 더 이상 추론적이 아닌지를 퇴거 큐(102)에 표시한다.

[0045] 기능 유닛(126A 및 126B) 각각은 일 실시예에서 정수 및 부동 소수점 연산을 수행할 수 있다. 기능 유닛(126A 및 126B) 각각은 데이터 예측을 검증할 수 있다. 추론 포인터(A) 및 추론 포인터(B)는 각각 도 1에 도시된 실시예에서 어떤 연산들이 기능 유닛(126A 및 126B)에 의해 검증되었는지를 식별한다. 추론 포인터(C)는 로드 및 저장 연산을 수행하는 기능 유닛(126C)에 의해 생성된다. 추론 포인터(C)는 어떤 연산들이 로드 저장 유닛(126C)에 의해 검증되었는지를 식별할 수 있다. 기능 유닛(126C)은 주소, 데이터, 및/또는 의존성 예측을 검증할 수 있다.

[0046] 상술된 바와같이, 각 추론 포인터의 값은 어느 연산이 각각의 데이터 추론 검증 유닛에 의해 검증되었는지에 의

존한다. 일부 실시예에서, 또한, 마이크로프로세서(100)의 검증 부분들에 의해 생성되는 추론 포인터들의 값들은 데이터 추론이 수행된 연산들을 식별하는 정보에 의존할 수 있다. 예를 들어, 일 실시예에서, 메모리 제어기(160)는 메모리 예측이 수행된 각각의 연산을 추적(track)할 수 있다. 메모리 제어기(160)가 각각의 연산을 검증함에 따라, 메모리 제어기(160)는 메모리 예측이 수행된 후차 최신 연산(next youngest operation)까지의 모든 연산들이 메모리 제어기(160)에 대하여 비-데이터-추론적임을 나타내도록 추론 포인터(D)를 진행시킬 수 있다. 일 실시예에서, 메모리 제어기(160)는 메모리 제어기(160)에 의해 추적된 후차 최신 추론적 연산(next youngest speculative operation)을 포인팅하도록 추론 포인터(D)를 진행시킴으로써 그러한 연산들을 식별할 수 있다. 마찬가지로, 다른 타입의 데이터 추론을 수행하는 메커니즘들(예를 들어, 디스패치 유닛(104) 및/또는 로드 및 저장 연산을 수행하는 기능 유닛(126C))은 또한 데이터 추론이 수행된 연산을 추적할 수 있다. 일부 실시예에서, 이 데이터 추론 메커니즘들의 적어도 일부는 이 정보를 그 타입의 데이터 추론을 검증하도록 구성된 데이터 추론 검증 유닛(들)(예를 들어, 로드 저장 유닛(126C) 및/또는 하나 이상의 다른 기능 유닛(126))에 제공할 수 있다. 만일 현재 데이터 추론 검증 유닛들 중 특정 검증 유닛에 의해 검증될 데이터-추론적 연산들이 없는 경우, 그 검증 유닛에 의해 생성되는 추론 포인터는, 모든 미결 연산들이 그 특정 검증 유닛에 대하여 비-데이터-추론적임을 표시하는 값으로 설정될 수 있다.

[0047] 퇴거 큐(102)는 추론 포인터들에 의해 식별되는 연산 스트림의 부분들을 비교함으로써 어느 연산들이 퇴거될 수 있는지를 식별할 수 있다. 모든 추론 포인터들에 의해 비-추론적인 것으로서 식별된 가장 오래된 연산은 퇴거 큐(102)에 의해 퇴거가능한 가장 오래된 연산일 수 있다. 예컨대, 연산(0 내지 10)(연산(0)은 프로그램 순서에서 가장 오래된 연산이며 연산(10)은 프로그램 순서에서 가장 최신 연산이며, 여기서 프로그램 순서는 마이크로프로세서(100) 내에서 재순서화 또는 순서를 벗어난 프로세싱이 수행된 임의의 연산 이전에 실행되는 프로그램에서의 명령 순서임)이 디스패치 유닛(104)에 의해 디스패치된 것으로 가정한다. 만일 추론 포인터(A)가 연산(6)까지의 연산들이 비-데이터-추론적인 것으로 표시하고, 추론 포인터(B)는 연산(5)까지의 연산들이 비-데이터-추론적인 것으로 표시하며, 추론 포인터(C)가 연산(3)까지의 연산들이 비-데이터-추론적인 것으로 표시하며, 추론 포인터(D)가 모든 미결 연산들이 비-데이터-추론적인 것으로 표시하는 경우에(예를 들어, 어떤 메모리 예측도 임의의 연산(0 내지 10)에 대해 수행되지 않았기 때문에 그러함), 퇴거 큐(102)는 연산(3)까지의 모든 연산을 퇴거될 수 있는 연산 세트로서 식별할 수 있다. 연산 퇴거는 또한 이들 연산이 이미 기능 유닛(126)에 의해 실행되었는지 및 이들 연산에 영향을 미치는 임의의 제어 예측이 성공적으로 결정되었는지와 같은 전형적인 연산 제약에 의존할 수 있음을 주목해야 한다. 예컨대, 데이터-추론 연산이 부정확한 것으로 결정되며, 재실행될 필요가 있는 경우, 연산은 하나 이상의 추론 포인터에 의해 비-추론적으로 식별될 수 있지만, 연산이 재실행될 때까지 퇴거되지 않아야 한다. 어느 연산이 재실행되었는지, 어느 연산이 재실행되지 않았는지, 또는 필요한 경우 재실행되었는지에 관한 정보는 일부 실시예에서 스케줄러(118)에 의해 퇴거 큐(102)에 제공될 수 있다.

[0048] 도 2A는 추론 포인터를 생성하는데 사용되는 방법의 일 실시예를 도시한다. 이러한 방법은 적어도 부분적으로 메모리 제어기(160) 및 기능 유닛(126)과 같은 데이터 추론 검증 유닛들 중 하나에 의해 구현될 수 있다. 단계(201)에서, 하나의 연산에 대해 데이터 추론이 수행된다. 단계(201)에서의 데이터 추론은 일부 실시예에서 데이터 추론을 검증할 동일 부분의 마이크로프로세서에 의해 수행될 수 있다. 이 데이터 추론이 단계(203)에서 검증된 경우, 추론 포인터는 단계(205)에서 보여진 바와같이 데이터 추론이 단계(201)에서 비-추론적으로 수행되는 연산을 식별하기 위해 진행될 수 있다.

[0049] 일 실시예에서, 단계(205)에서 추론 포인터를 진행하는 것은 가장 최근에 검증된 연산을 식별하도록 추론 포인터를 진행시키는 것을 포함할 수 있으며, 이는 그 연산을 포함하여 그 연산까지의 모든 연산들이 데이터 추론을 검증하기 위한 특정 수단에 대하여 비-데이터-추론적임을 나타낸다. 다른 실시예에서, 단계(205)에서 추론 포인터를 진행시키는 것은, 프로그램 순서상 그 특정 검증 수단에 의해 검증될, 다음의 데이터-추론적 연산(next data-speculative operation) 직전의 연산을 식별하도록 추론 포인터를 진행하는 것을 포함할 수 있다. 예컨대, 로드 저장 유닛(126)은 로드 저장 유닛이 데이터 추론을 수행한 연산들을 추적할 수 있다. 이 로드 저장 유닛(126)이 이들 데이터-추론 연산들 중 하나를 검증할 때마다, 로드 저장 유닛(126)은 로드 저장 유닛이 데이터 추론을 수행한 후차 연산까지의 모든 연산들이 로드 저장 유닛에 대하여 비-데이터-추론인 것으로 표시도록 그것의 추론 포인터를 진행시킬 수 있다. 다른 실시예는 다른 방식으로 포인터를 진행시킬 수 있다. 예컨대, 기능 유닛은 디스패치 유닛에 의해 수행된 데이터 추론 타입을 검증할 수 있다. 일부 실시예들에서, 기능 유닛은 디스패치 유닛이 이러한 타입의 데이터 추론을 수행한 미결 연산들의 전체 세트를 알지 못할 수 있다. 대신에, 기능 유닛은 기능 유닛 내의 현재의 미결 연산들 중 어느 연산이 데이터-추론적인지만을 알 수 있다. 따라서, 특정 연산에 대한 데이터 추론의 검증에 응답하여, 기능 유닛은, (만약 있다면) 이 기능 유닛 내의 미결의 가장 오래된 데이터-추론적 연산 직전의 연산을 식별하도록 추론 포인터를 진행시킬 수 있다. 만일 이 기능 유닛 내

의 현재의 미결 연산들 중 어떤 연산도 데이터-추론적이 아닌 경우, 기능 유닛은 이 기능 유닛에 대하여 현재 데이터-추론적인 미결 연산이 없음을 표시하기 위하여 자신의 추론 포인터값을 갱신할 수 있다.

[0050] 도 2B는 미결 연산을 퇴거시키는 방법의 일 실시예의 흐름도이다. 단계(211)에서, 하나 이상의 추론 포인터가 수신된다. 만일 복수의 추론 포인터가 수신되는 경우, 각 추론 포인터는 미결 연산들의 서로 다른 부분을 비-데이터-추론적인 것으로 식별할 수 있다. 만일 임의의 추론 포인터가 특정 연산이 여전히 데이터-추론적임을 표시한다면, 연산은 단계(213)에서 나타난 바와같이 퇴거되지 않을 수 있다. 하지만, 어떠한 연산 포인터도 이 연산을 데이터-추론적일 수 있는 연산으로 식별하지 않는 경우, 이 연산의 퇴거에 대한 다른 모든 필요조건이 충족된 것으로 가정하면, 연산은 단계(213-215)에 표시된 바와 같이 퇴거될 수 있다.

[0051] 예시적 컴퓨터 시스템

[0052] 도 3은 버스 브릿지(902)를 통해 다양한 시스템 요소들에 연결되는 프로세서(100)를 포함하는 컴퓨터 시스템(900)의 일 실시예의 블록도이다. 프로세서(100)는 상술된 바와같이 추론 포인터를 생성하도록 구성된 하나 이상의 데이터 검증 유닛 및 추론 포인터에 의해 비-데이터-추론으로 식별되는 연산을 퇴거하도록 구성된 퇴거 큐를 포함할 수 있다. 다른 실시예의 컴퓨터 시스템이 가능하며 고려된다. 도시된 시스템에서, 메인 메모리(200)가 메모리 버스(906)를 통해 버스 브릿지(902)에 연결되며, 그래픽 제어기(908)가 AGP 버스(910)를 통해 버스 브릿지(902)에 연결된다. 여러 PCI 디바이스(912A 내지 912B)가 PCI 버스(914)를 통해 버스 브릿지(902)에 연결된다. 보조 버스 브릿지(916)가 또한 EISA/ISA 버스(920)를 통해 하나 이상의 EISA 또는 ISA 디바이스(918)에 전기적 인터페이스를 용이하게 하기 위해 제공될 수 있다. 본 예에서, 프로세서(100)는 CPU 버스(924)를 통해 버스 브릿지(902)에, 및 선택적 L2 캐시(928)에 연결된다. 일부 실시예에서, 프로세서(100)는 통합 L1 캐시(미도시)를 포함할 수 있다.

[0053] 버스 브릿지(902)는 프로세서(100), 메인 메모리(200), 그래픽 제어기(908), 및 PCI 버스(914)에 부착되는 장치 사이에 인터페이스를 제공한다. 연산이 버스 브릿지(902)에 연결되는 장치들 중 하나로부터 수신되는 때에, 버스 브릿지(902)는 연산의 타겟(예를 들어, 특정 장치, 또는 PCI 버스(914)의 경우에 타겟은 PCI 버스(914)상에 있음)을 식별한다. 버스 브릿지(902)는 연산을 타겟 장치에 라우팅한다. 버스 브릿지(902)는 일반적으로 소스 장치 또는 버스에 의해 사용되는 프로토콜에서 타겟 장치 또는 버스에 의해 사용되는 프로토콜로 연산을 전환한다.

[0054] PCI 버스(914)에 ISA/EISA 버스에 대한 인터페이스를 제공하는 것 이외에, 보조 버스 브릿지(916)는 추가의 기능성을 포함할 수 있다. 또한, 외부로부터 또는 보조 버스 브릿지(916)와 통합된 입력/출력 제어기(미도시)가 컴퓨터 시스템(900) 내에 포함될 수 있는바, 이는 키보드 및 마우스(922)를 위한, 다양한 직렬 및 병렬 포트를 위한 연산 지원을 제공하기 위함이다. 또한, 외부 캐시 유닛(미도시)이 다른 실시예에서 프로세서(100)와 버스 브릿지(902) 사이의 CPU 버스(924)에 연결될 수 있다. 대안적으로, 외부 캐시는 버스 브릿지(902)에 연결될 수 있으며, 외부 캐시에 대한 캐시 제어 로직이 버스 브릿지(902) 내로 통합될 수 있다. L2 캐시(928)는 프로세서(100)의 후방 구성에서 도시된다. L2 캐시(928)는 프로세서(100)와 별개이며, 프로세서(100)와 함께 카트리지(예를 들어, 슬롯 1 또는 슬롯 A)로 통합되거나, 심지어는 프로세서(100)와 함께 반도체 기판상으로 통합될 수 있다.

[0055] 메인 메모리(200)는 여기에 응용 프로그램이 저장되며, 이로부터 프로세서(100)가 주로 실행되는 메모리이다. 적합한 메인 메모리(200)는 DRAM(동적 랜덤 액세스 메모리)을 포함할 수 있다. 예컨대, 복수의 뱅크의 SDRAM(동기 DRAM) 또는 램버스 DRAM(RDRAM)이 적합할 수 있다.

[0056] PCI 디바이스(912A 내지 912B)는 네트워크 인터페이스 카드, 비디오 가속기, 오디오 카드, 하드 또는 플로피 디스크 드라이브 또는 드라이브 제어기, SCSI(소형 컴퓨터 시스템 인터페이스) 어댑터 및 전화 카드와 같은 다양한 주변 장치의 예시이다. 유사하게는, ISA 장치(918)는 모뎀, 사운드 카드, 및 GPIB 또는 필드 버스 인터페이스 카드와 같은 다양한 데이터 획득 카드와 같은 다양한 타입의 주변 장치의 예시이다.

[0057] 그래픽 제어기(908)는 디스플레이(926)상에 텍스트 및 이미지의 표시(rendering)를 제어하도록 제공된다. 그래픽 제어기(908)는 메인 메모리(200)로/로부터 효율적으로 이동될 수 있는 3차원 데이터 구조를 표시하기 위해 본 기술분야에서 일반적으로 알려진 전형적인 그래픽 제어기를 구체화할 수 있다. 따라서, 그래픽 제어기(908)는 AGP 버스(910)의 마스터가 될 수 있는바, 이는 버스 브릿지(902) 내의 타겟 인터페이스에 액세스를 요구하며 수신할 수 있으며, 이에 따라 메인 메모리(200)에 액세스를 얻을 수 있기 때문이다. 전용 그래픽 버스는 메인 메모리(200)로부터 급속 데이터 검색을 용이하게 한다. 특정 연산에 대해, 그래픽 제어기(908)는 AGP 버스(91

0)상에 PCI 프로토콜 트랜잭션을 생성하도록 더 구성될 수 있다. 따라서, 버스 브릿지(902)의 AGP 인터페이스는 AGP 프로토콜 트랜잭션뿐만 아니라 PCI 프로토콜 타겟 및 개시자 트랜잭션을 지원하는 기능성을 포함할 수 있다. 디스플레이(926)는 그 상에 이미지 또는 텍스트가 제공될 수 있는 임의의 전자 장치이다. 적합한 디스플레이(926)는 음극선관("CRT"), 액정 표시 장치("LCD") 등을 포함한다.

[0058] AGP, PCI, 및 ISA 또는 EISA 버스가 상세한 설명에서 예로서 사용되었지만은, 바람직한 경우에, 임의의 버스 아키텍처로 대체될 수 있다. 컴퓨터 시스템(900)은 추가 프로세서(예를 들어, 컴퓨터 시스템(900)의 선택적 요소로서 도시된 프로세서(100a))를 포함하는 멀티프로세싱 컴퓨터 시스템이 될 수 있다. 프로세서(100a)는 프로세서(100)와 유사하다. 특히, 프로세서(100a)는 프로세서(100)의 동일한 복제가 될 수 있다. 프로세서(100a)는 (도 3에 도시된 바와같이) 개별 버스를 통해 버스 브릿지(902)에 연결되거나 프로세서(100)와 CPU 버스(924)를 공유할 수 있다. 더욱이, 프로세서(100a)는 L2 캐시(928)와 유사한 선택적 L2 캐시(928a)에 연결될 수 있다.

[0059] 도 4를 참조하면, 프로세서(100)를 포함하는 컴퓨터 시스템(900)의 다른 실시예가 도시되는바, 프로세서(100)는 하나 이상의 추론 포인터를 생성하도록 구성된 데이터 추론 검증 유닛 및 상기 추론 포인터에 의존하는 연산을 퇴거시키는 퇴거 큐를 구비한다. 다른 실시예들이 가능하며 고려된다. 도 4의 실시예에서, 컴퓨터 시스템(900)은 여러 프로세싱 노드(1012A, 1012B, 1012C 및 1012D)를 포함한다. 각 프로세싱 노드는 각 프로세싱 노드(1012A 내지 1012D) 내에 포함된 메모리 제어기(1016A 내지 1016D)를 통해 각 메모리(200A 내지 200D)에 연결된다. 추가적으로, 프로세싱 노드(1012A 내지 1012D)는 프로세싱 노드들(1012A 내지 1012D) 간에서 통신하는데 사용되는 인터페이스 로직을 포함한다. 예컨대, 프로세싱 노드(1012A)는 프로세싱 노드(1012B)와 통신하기 위한 인터페이스 로직(1018A), 프로세싱 노드(1012C)와 통신하기 위한 인터페이스 로직(1018B), 및 또 다른 프로세싱 노드(미도시)와 통신하기 위한 제 3 인터페이스 로직(1018C)을 포함한다. 유사하게는, 프로세싱 노드(1012B)는 인터페이스 로직(1018D, 1018E, 및 1018F)을 포함하며; 프로세싱 노드(1012C)는 인터페이스 로직(1018G, 1018H, 및 1018I)을 포함하며; 프로세싱 노드(1012D)는 인터페이스 로직(1018J, 1018K, 및 1018L)을 포함한다. 프로세싱 노드(1012D)는 인터페이스 로직(1018L)을 통해 복수의 입/출력 장치(예를 들어, 연쇄 구성에서, 장치 1012A 내지 1020B)와 통신하도록 연결된다. 다른 프로세싱 노드는 유사한 방식으로 다른 I/O 장치와 통신할 수 있다.

[0060] 프로세싱 노드(1012A 내지 1012D)는 프로세싱 노드간의 통신을 위해 패킷-기반 링크를 구현한다. 본 실시예에서, 링크는 단방향 라인 세트로서 구현된다(예를 들어, 라인(1024A)은 프로세싱 노드(1012A)에서 프로세싱 노드(1012B)로 패킷을 전송하는데 사용되며, 라인(1024B)은 프로세싱 노드(1012B)에서 프로세싱 노드(1012A)로 패킷을 전송하는데 사용된다). 다른 라인 (1024C 내지 1024H) 세트는 도 4에 도시된 바와 같이 다른 프로세싱 노드들 간에서 패킷을 전송하는데 사용된다. 일반적으로, 각 라인(1024) 세트는 하나 이상의 데이터 라인, 데이터 라인에 대응하는 하나 이상의 클록 라인, 및 전달되는 패킷의 타입을 나타내는 하나 이상의 제어 라인을 포함할 수 있다. 이 링크는 프로세싱 노드들 간의 통신을 위한 캐시 코히어런트 방식 또는 프로세싱 노드와 I/O 장치(또는 PCI 버스 또는 ISA 버스와 같은 종래 구성의 I/O 버스로의 버스 브릿지) 간의 통신을 위한 비-코히어런트 방식에서 연산할 수 있다. 더욱이, 링크는 도시된 바와같이 I/O 장치들 간의 연쇄 구성(daisy-chain structure)을 사용하여 비-코히어런트 방식으로 연산할 수 있다. 일 프로세싱 노드로부터 다른 노드로 전송되는 패킷이 하나 이상의 중개 노드를 통해 전달될 수 있음을 주목해야 한다. 예컨대, 프로세싱 노드(1012A)에 의해 프로세싱 노드(1024D)로 전송되는 패킷은 도 4에 도시된 바와같이 프로세싱 노드(1012B) 또는 프로세싱 노드(1012C)를 통해 전달될 수 있다. 임의의 적합한 라우팅 알고리즘이 사용될 수 있다. 다른 실시예의 컴퓨터 시스템(900)은 도 4에서 도시된 실시예 보다 많은 프로세싱 노드 또는 보다 적은 노드를 포함할 수 있다.

[0061] 일반적으로, 패킷은 노드들 간의 라인(1024) 상에서 하나 이상의 비트 타임으로 전송될 수 있다. 일 비트 타임은 대응 클록 라인 상의 클록 신호의 상승 또는 하강 에지가 될 수 있다. 패킷은 트랜잭션을 개시하기 위한 지령(command) 패킷, 캐시 코히어런시를 유지하기 위한 프로브 패킷, 및 프로브와 지령에 응답하기 위한 응답 패킷을 포함할 수 있다.

[0062] 메모리 제어기 및 인터페이스 로직 이외에, 프로세싱 노드(1012A 내지 1012D)는 하나 이상의 프로세서를 포함할 수 있다. 일반적으로, 프로세싱 노드는 적어도 하나의 프로세서를 포함하며, 바람직한 경우에 선택적으로 메모리 및 다른 로직과 통신하기 위한 메모리 제어기를 포함할 수 있다. 좀더 구체적으로, 각 프로세싱 노드(1012A 내지 1012D)는 하나 이상의 프로세서(100)의 복제를 포함할 수 있다. 외부 인터페이스 유닛은 노드 내에 인터페이스 로직(1018)뿐만 아니라 메모리 제어기(1016)를 포함할 수 있다.

[0063] 메모리(200A 내지 200D)는 임의의 적합한 메모리 장치를 포함할 수 있다. 예컨대, 메모리(200A 내지 200D)는 하

나 이상의 RAMBUS DRAMs(RDRAMs), 동기 DRAMs(SDRAMs), 정적 RAM 등을 포함할 수 있다. 컴퓨터 시스템(900)의 주소 공간은 메모리들(200A 내지 200D) 사이에서 분배된다. 각 프로세싱 노드(1012A 내지 1012D)는 어느 주소들이 어느 메모리들(200A 내지 200D)에, 이에 따라 특정 주소에 대한 메모리 요구가 라우팅되어야 하는 어느 프로세싱 노드(1012A 내지 1012D)에 맵핑되는지를 결정하는데 사용되는 메모리 맵을 포함할 수 있다. 일 실시예에서, 컴퓨터 시스템(900) 내의 주소에 대한 코히어런시 포인트는 주소에 대응하는 바이트를 저장하는 메모리에 연결된 메모리 제어기(1016A 내지 1016D)이다. 바꾸어 말하면, 메모리 제어기(1016A 내지 1016D)는 대응 메모리들(200A 내지 200D)에 각 메모리 액세스가 캐시 코히어런시 방식으로 일어나도록 보장하는 것을 담당한다. 메모리 제어기(1016A 내지 1016D)는 메모리들(200A 내지 200D)에 인터페이싱하기 위한 제어 회로를 포함할 수 있다. 추가적으로, 메모리 제어기(1016A 내지 1016D)는 메모리 요구를 큐잉하기 위한 요구 큐를 포함할 수 있다.

[0064] 인터페이스 로직(1018A 내지 1018L)은 링크로부터의 패킷을 수신하기 위한, 링크상으로 전송될 패킷을 버퍼링하기 위한 여러가지 버퍼를 포함할 수 있다. 컴퓨터 시스템(900)은 패킷 전송을 위한 임의의 적합한 흐름 제어 메커니즘을 이용할 n 있다. 예컨대, 일 실시예에서, 각 인터페이스 로직(1018)은, 인터페이스 로직이 이에 연결되는 링크의 다른 종단에서의 수신기 내에 각 타입의 버퍼의 개수에 대한 카운트를 저장한다. 인터페이스 로직은 수신 인터페이스 로직이 패킷을 저장할 빈(free) 버퍼를 갖지 않는 경우에 패킷을 전송하지 않는다. 수신 버퍼가 전송될 패킷을 라우팅함으로써 비어지는 때에, 수신 인터페이스 로직은 버퍼가 비어졌음을 나타내기 위해 송신 인터페이스에 메시지를 전송한다. 이러한 메커니즘은 "쿠폰-기반" 시스템으로서 지칭될 있다.

[0065] I/O 장치(1020A 내지 1020B)는 임의의 적합한 I/O 장치가 될 수 있다. 예컨대, I/O 장치(1020A 내지 1020B)는 장치들이 연결될 수 있는 다른 통신 시스템(예를 들어, 네트워크 인터페이스 카드 또는 모뎀)과 통신하기 위한 장치를 포함할 수 있다. 더욱이, I/O 장치(1020A 내지 1020B)는 비디오 가속기, 오디오 카드, 하드 또는 플로피 디스크 드라이브 또는 드라이브 제어기, SCSI(소형 컴퓨터 시스템 인터페이스) 어댑터 및 전화 카드, 사운드 카드, 및 GPIB 또는 필드 버스 인터페이스 카드와 같은 다양한 데이터 획득 카드를 포함할 수 있다. 용어 "I/O 인터페이스" 및 용어 "주변 장치"는 본원에서 동의어로 의도됨을 주목해야 한다.

[0066] 본원에서 사용된 바와같이, 용어 "클록 사이클"은 명령 프로세싱 파이프라인의 다양한 스테이지가 작업을 완료하는 시간 간격을 의미한다. 명령들 및 연산될 값들은 클록 사이클을 정의하는 클록 신호에 따라 (레지스터들 또는 어레이들과 같은) 메모리 요소들에 의해 포획된다. 예컨대, 메모리 요소는 클록 신호의 상승 또는 하강 에지에 따라 값을 포획할 수 있다.

[0067] 상기 내용은 신호를 "표명(assert)되는" 것으로서 설명한다. 신호는 특정 정보를 나타내는 값을 전달할 때에 표명되는 것으로서 정의될 수 있다. 특정 신호는 이진수 1 값을 전달하거나, 대안적으로, 이진수 0 값을 전달할 때에 표명되는 것으로서 정의될 수 있다.

[0068] 수많은 변화 및 변형은 상기 개시내용이 완전히 이해되는 경우에 당업자에게 자명하게 될 것이다. 하기 청구범위는 이런 모든 변화 및 변형을 포함하는 것으로서 해석되어야 한다.

산업상 이용 가능성

[0069] 본 발명은 일반적으로 마이크로프로세서 기술분야에 적용가능하다.

도면의 간단한 설명

[0013] 이하 도면들과 관련하여 상세한 설명을 할 때 본 발명을 더 잘 이해할 수 있을 것이다.

[0014] 도 1은 본 발명에 따른 일 실시 예로서, 마이크로프로세서를 도시한 것이다.

[0015] 도 2A는 본 발명에 따른 일 실시 예로서, 추론 포인터들이 어떻게 진행되는지를 도시한 흐름도이다.

[0016] 도 2B는 본 발명에 따른 일 실시 예로서, 현재 추론 포인터들 상에서 연산의 퇴거를 조절하는 방법을 도시한 흐름도이다.

[0017] 도 3은 본 발명에 따른 일 실시 예로서, 예시적인 컴퓨터 시스템이다.

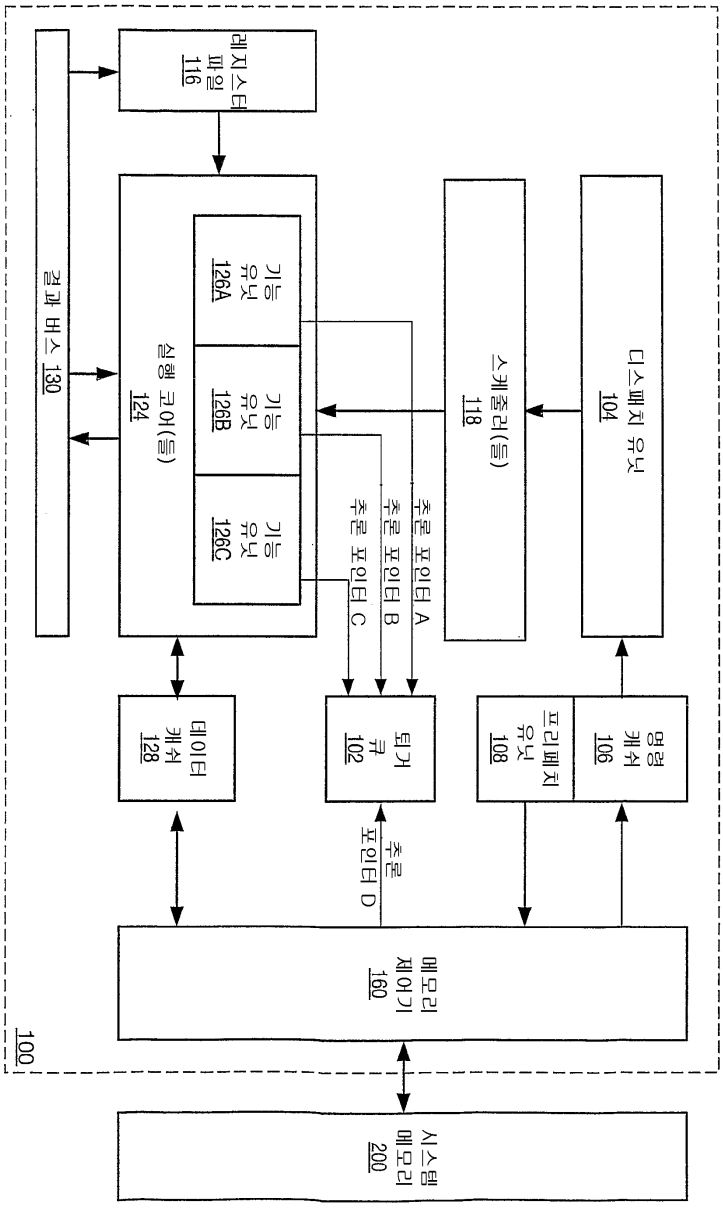
[0018] 도 4는 본 발명에 따른 또 다른 실시 예로서, 컴퓨터 시스템의 또 다른 예시이다.

[0019] 본 발명은 여러 가지 변형들 및 대안적인 형태들을 수용할 수 있는 한편, 이들의 특정 실시 예들이 상기 도면들에서 예시적으로 도시된 것이며, 이하에서 상세히 설명될 것이다. 그러나, 이해하여야할 사항으로서, 상기 도면

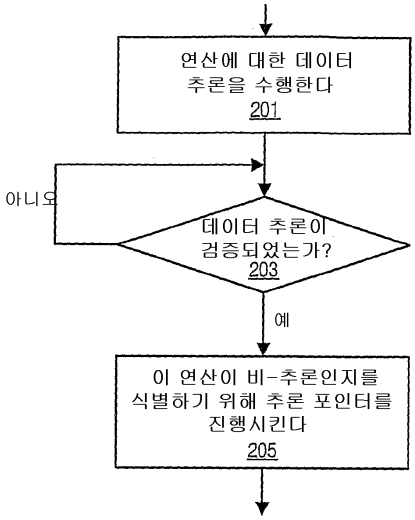
들 및 그 도면들의 상세한 설명은 설명하는 특정 형태로 본 발명은 한정하지 않으며, 첨부된 특허청구범위에 의해 정의되는 바와 같이, 본 발명의 기술적 사상 및 권리범위에 속하는 모든 변형들, 균등물, 그리고 대안 사항을 포함한다. 알아야 할 것으로, 제목은 단지 구성상의 목적을 위한 것이지, 상세한 설명 또는 특허청구범위를 한정하거나 해석하기 위한 것이 아니다. 또한 단어 "할 수 있다"는 본 출원을 통하여 허용될 수 있다는 의미(즉, 잠재 능력이 있는 것, 할 수 있다) 로 사용된다. 단어 "포함한다" 및 이 단어의 파생어는 "~를 포함하는(그러나, ~에만 한정되지 않음)"를 의미한다. 단어 "연결된"은 "직접 또는 간접으로 연결된"를 의미한다.

도면

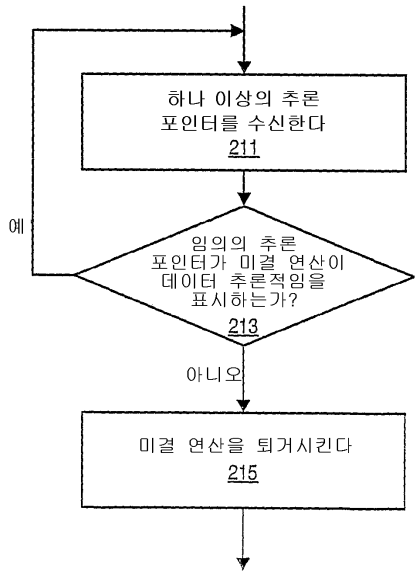
도면1



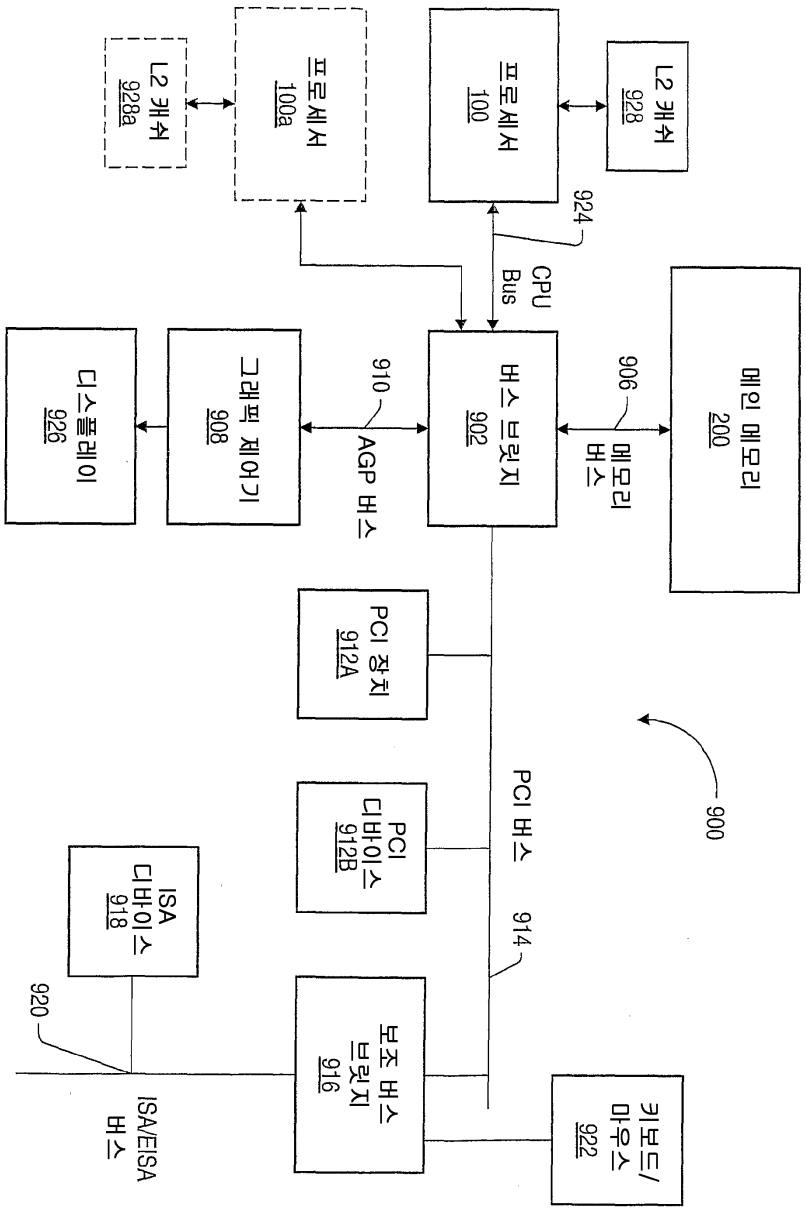
도면2a


연산에 대한 데이터
추론을 수행한다
201
아니오
데이터 추론이
검증되었는가?
203
예
이 연산이 비-추론인지를
식별하기 위해 추론 포인터를
진행시킨다
205

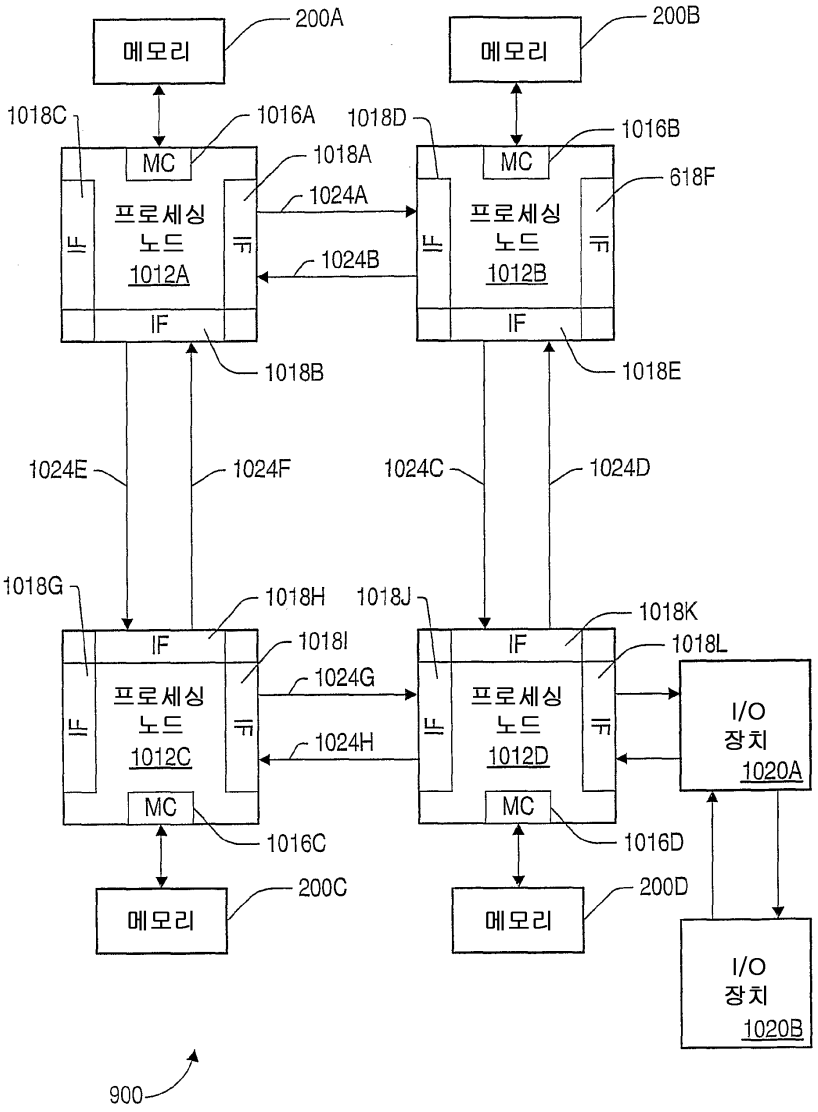
도면2b


하나 이상의 추론
포인터를 수신한다
211
예
임의의 추론
포인터가 미결 연산이
데이터 추론적임을
표시하는가?
213
아니오
미결 연산을 퇴거시킨다
215

도면3


L2 캐쉬
928
프로세서
100
924
CPU
Bus
L2 캐쉬
928a
프로세서
100a
메인 메모리
200
906
메모리
버스
버스 브릿지
902
910
AGP 버스
그래픽 제어기
908
디스플레이
926
PCI 장치
912A
900
PCI 버스
914
PCI
디바이스
912B
ISA
디바이스
918
920
보조 버스
브릿지
916
키보드/
마우스
922
ISA/EISA
버스

도면4


메모리
200A
메모리
200B
1018C
1016A
1018D
1016B
MC
MC
1018A
618F
1024A
프로세싱 노드 1012A
프로세싱 노드 1012B
1024B
IF
IF
1018B
1018E
1024E
1024F
1024C
1024D
1018G
1018H
1018J
1018K
IF
IF
1018I
1018L
1024G
프로세싱 노드 1012C
프로세싱 노드 1012D
I/O 장치 1020A
1024H
MC
MC
1016C
1016D
메모리
200C
메모리
200D
I/O 장치 1020B
900