(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2011년12월19일
(11) 등록번호 10-1093784
(24) 등록일자 2011년12월07일

(51) Int. Cl.
G06F 9/38 (2006.01)
(21) 출원번호 10-2005-7010852
(22) 출원일자(국제출원일자) 2004년06월02일
심사청구일자 2009년05월15일
(85) 번역문제출일자 2005년06월13일
(65) 공개번호 10-2006-0021281
(43) 공개일자 2006년03월07일
(86) 국제출원번호 PCT/US2004/017096
(87) 국제공개번호 WO 2004/111839
국제공개일자 2004년12월23일
(30) 우선권주장
10/458,457 2003년06월10일 미국(US)
(56) 선행기술조사문헌
KR1020070068351 A
US20030088760 A1
전체 청구항 수 : 총 10 항

(73) 특허권자
어드밴스드 마이크로 디바이시즈, 인코포레이티드
미국 캘리포니아 94088-3453 서니베일 원 에이엠디 플레이스 메일 스톱68
(72) 발명자
필리포 마이클 에이.
미국 텍사스 78652 맨차카 채파랠 로드 2030
피켓 제임스 케이.
미국 텍사스 78733 오스틴 #2 팔로미노 리지 드라이브 1700
(뒷면에 계속)
(74) 대리인
박장원

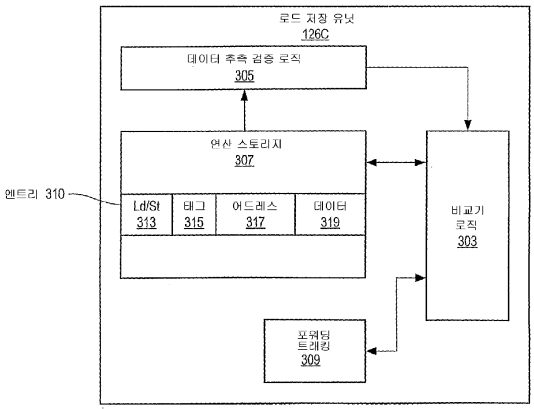
심사관 : 황승희

(54) 리플레이 메커니즘을 구비한 로드 저장 유닛

(57) 요 약

연산(operation)들을 발행하도록 구성된 스케줄러(118) 그리고, 상기 스케줄러(118)에 의해서 발행되는 메모리 연산들을 실행하도록 구성되는 로드 저장 유닛(126C)을 포함하는 마이크로프로세서(100)가 제시된다. 상기 로드 저장 유닛(126C)은 상기 로드 저장 유닛(126C)에 발행된 다수의 메모리 연산들을 식별하는 정보를 저장하도록 구성된다. 상기 발행된 메모리 연산들 중 하나에 대해 부정확 데이터 추측이 검출됨에 응답하여, 상기 로드 저장 유닛(126C)은 상기 스케줄러(118)에 표시를 제공함으로써 상기 발행된 메모리 연산들 중 적어도 하나를 리플레이(replay)하도록 구성된다. 상기 스케줄러(118) 상기 로드 저장 유닛(126C)에 의해서 식별된 메모리 연산들을 개별적으로 재발행하도록 구성된다.

대 표 도



(72) 발명자

샌더 벤자민 티.

미국 텍사스 78735 오스틴 메디신 크리크 5701

고팔 라마 에스.

미국 텍사스 78759 오스틴 미라몽테 드라이브 5906

특허청구의 범위

청구항 1

연산들을 발행하도록 구성된 스케줄러(118)와; 그리고

상기 스케줄러(118)에 의해 발행되는 메모리 연산들을 수신하도록 연결되고, 상기 메모리 연산들을 실행하도록 구성되는 로드 저장 유닛(126C)을 포함하는 마이크로프로세서(100)로서,

여기서, 상기 로드 저장 유닛(126C)은 상기 로드 저장 유닛(126C)에 발행된 다수의 메모리 연산들을 식별하는 정보를 저장하도록 구성되고, 다수의 메모리 연산들 중 하나에 대해 부정확 데이터 추측이 검출됨에 따라, 상기 로드 저장 유닛(126C)은 상기 로드 저장 유닛(126C) 내의 다수의 메모리 연산들 중 적어도 하나가 재발행되어야 함을 나타내는 리플레이 표시(replay indication)를 상기 스케줄러(118)에 제공하도록 구성되며;

상기 스케줄러(118)는 상기 로드 저장 유닛(126C)으로부터의 표시에 따라 상기 다수의 메모리 연산들 중 적어도 하나를 발행하도록 구성되는 것을 특징으로 하는 마이크로프로세서.

청구항 2

제 1 항에 있어서,

상기 로드 저장 유닛(126C)은 상기 다수의 메모리 연산들 중 하나의 어드레스와 일치하는 어드레스를 갖는 다수의 메모리 연산들 중 각각을 식별함으로써 상기 리플레이 표시를 생성하도록 구성되는 것을 특징으로 하는 마이크로프로세서.

청구항 3

제 1 항에 있어서,

상기 로드 저장 유닛(126C)은 다수의 메모리 연산들 중 하나의 어드레스의 추측 값 또는 다수의 메모리 연산들 중 하나의 어드레스의 새로운 값 중 어느 하나와 일치하는 어드레스를 갖는 상기 다수의 메모리 연산들 중 각각을 식별함으로써 상기 리플레이 표시 생성하도록 구성되는 것을 특징으로 하는 마이크로프로세서.

청구항 4

제 1 항에 있어서,

상기 로드 저장 유닛(126C)은, 로드 연산이며, 상기 다수의 메모리 연산들 중 하나의 어드레스와 일치하는 어드레스를 갖는 상기 다수의 메모리 연산들 중 각각을 식별함으로써 상기 리플레이 표시를 생성하도록 구성되는 것을 특징으로 하는 마이크로프로세서.

청구항 5

제 1 항에 있어서,

상기 로드 저장 유닛(126C)은 상기 다수의 메모리 연산에 포함된 로드 연산중 어느 연산이 저장 연산으로부터 데이터를 포워딩(forwarding)하였는지를 추적하도록 구성되고, 그리고 저장 연산의 어드레스가 부정확하게 추측된 것으로 검출되는 경우, 상기 로드 저장 유닛(126C)은 상기 저장 연산으로부터 데이터를 포워딩한 로드 연산들 중 보다 최근(younger)의 연산들을 식별함으로써 상기 리플레이 표시를 생성하도록 구성되는 것을 특징으로 하는 마이크로프로세서.

청구항 6

시스템 메모리(200)와; 그리고,

상기 시스템 메모리(200)에 연결된 마이크로프로세서(100)를 포함하는 컴퓨터 시스템(900)에 있어서,

상기 마이크로프로세서(100)는,

메모리 연산들을 발행하도록 구성된 스케줄러(118)와; 그리고,

상기 스케줄러(118)에 의해 발행된 메모리 연산을 수신하도록 연결되며, 그리고 메모리 연산들을 실행하도록 구성되는 로드 저장 유닛(126C)을 포함하고,

여기서, 상기 로드 저장 유닛(126C)은 상기 로드 저장 유닛(126C)에 발행된 다수의 메모리 연산들을 식별하는 정보를 저장하도록 구성되고, 다수의 메모리 연산들 중 하나에 대해 부정확 데이터 추측이 검출됨에 따라, 상기 로드 저장 유닛(126C)은 상기 로드 저장 유닛(126C) 내의 다수의 메모리 연산들 중 적어도 하나가 재발행되어야 함을 나타내는 리플레이 표시를 상기 스케줄러(118)에 제공하도록 구성되며;

상기 스케줄러(118)는 상기 로드 저장 유닛(126C)으로부터의 표시에 따라 다수의 메모리 연산들 중 적어도 하나를 발행하도록 구성되는 것을 특징으로 하는 컴퓨터 시스템.

청구항 7

스케줄러(118)가 실행하기 위해 연산을 로드 저장 유닛(126C)에 발행하는 단계와;

상기 로드 저장 유닛(126C)이 상기 연산에서 실행되는 데이터 추측이 부정확함을 검출하는 단계와;

상기 검출에 응답하여, 상기 로드 저장 유닛(126C)이 상기 로드 저장 유닛(126C)에 현재 발행된 적어도 하나의 연산을 식별하는 리플레이 표시를 생성하는 단계와;

상기 리플레이 표시의 생성에 응답하여, 상기 스케줄러(118)가 상기 적어도 하나의 연산을 상기 로드 저장 유닛(126C)에 재발행하는 단계를 포함하는 것을 특징으로 하는 방법.

청구항 8

제 7 항에 있어서,

상기 생성하는 단계는 상기 데이터 추측이 부정확한 연산 보다 최근의 상기 로드 저장 유닛(126C) 내의 미처리 상태의 연산들 모두를 식별하는 하나 이상의 리플레이 표시를 상기 로드 저장 유닛(126C)이 생성하는 단계를 포함하는 것을 특징으로 하는 방법.

청구항 9

제 7 항에 있어서,

상기 생성하는 단계는 상기 데이터 추측이 부정확한 연산의 어드레스와 일치하는 어드레스를 갖는 상기 로드 저장 유닛(126C) 내의 미처리 상태의 임의의 연산을 식별하는 하나 이상의 리플레이 표시를 상기 로드 저장 유닛(126C)이 생성하는 단계를 포함하는 것을 특징으로 하는 방법.

청구항 10

제 7 항에 있어서,

상기 생성하는 단계는 상기 데이터 추측이 부정확한 연산의 어드레스의 추측 값과 일치하는 어드레스를 갖는 상기 로드 저장 유닛(126C) 내의 미처리 상태의 임의의 연산을 식별하는 하나 이상의 리플레이 표시를 상기 로드 저장 유닛(126C)이 생성하는 단계를 포함하는 것을 특징으로 하는 방법.

명 세 서

기 술 분 야

[0001] 본 발명은 마이크로프로세서에 관한 것으로, 더욱 상세히는 마이크로프로세서 내에서 데이터 추측(speculation)을 실행하는 방법에 관한 것이다.

배 경 기 술

[0002] 슈퍼스칼라 마이크로프로세서는 다중 명령을 동시에 처리함으로써, 그리고 설계된 바에 따라 가능한 가장 짧은 클록 사이클을 사용함으로써 고성능을 달성하고 있다. 그러나 데이터 및 명령들 간의 제어 흐름 의존성이 주어진 시간에 얼마나 많은 명령들이 발행될 수 있는지를 제한할 수 있다. 결과적으로, 일부 마이크로프로세서들은 추가적인 성능 증대를 얻기 위해 추측 실행을 지원하고 있다.

[0003] 추측의 일 타입은 제어 흐름 추측이다. 제어 흐름 추측은 프로그램 제어가 진행되는 방향을 예측한다. 예를 들면, 분기가 취해질 것인지 여부를 예측하기 위해 분기 예측이 사용될 수 있다. 분기 예측의 타입으로 많은 것들이 이용가능한데, 즉 시간마다 동일한 예측이 단순하게 행해지게 하는 방법에서부터 이력-기반(history-based) 예측을 하기 위해 프로그램 내의 이전 분기들의 상세한 이력들을 유지하는 방법에까지 많은 타입들이 이용가능하다. 분기 예측은 하드웨어 최적화, 컴파일러 최적화, 또는 이 모두를 통하여 가능하게 될 수 있다. 분기 예측 메커니즘에 의해 제공되는 예측에 기초하여, 명령들이 추측적으로 인출되고, 실행될 수 있다. 상기 분기 예측은 상기 분기 명령이 최종적으로 평가됨에 따라, 검증될 수 있다. 만약 예측이 부정확하다면, 부정확한 예측에 기초하여 추측으로 실행된 명령들은 폐기될 수 있다.

[0004] 다른 타입의 추측으로는 데이터 추측이 있다. 제안된 다른 타입의 데이터 추측은 메모리 연산을 위한 어드레스를 추측적으로 생성하는 것을 포함하고, 그리고 계산을 요구하는 연산에서 사용하기 위한 데이터 값을 추측적으로 생성하는 것을 포함한다. 제어 추측에서와 같이, 기본적 조건은 상기 추측이 최종적으로 평가되어서, 검증되도록 하거나, 혹은 수행되지 않도록 하는 것이다.

[0005] 추측은 의존성 검사가 완료되기만을 기다리지 않고, 명령 실행이 진행될 수 있도록 하기 때문에, 만약 정확한 추측으로부터 얻은 성능이 부정확한 추측으로 인해 잃은 성능보다 큰 것이라면, 현저한 성능 증대를 얻을 수 있게 한다. 부정확한 추측으로 인한 성능 불이익을 감소시키는 것이 바람직하다.

발명의 상세한 설명

[0006] 데이터 추측 마이크로프로세서의 로드 저장 유닛에서 연산을 리플레이(replay)하기 위한 방법 및 시스템에 대한 다양한 실시예들이 제시된다. 일부 실시예에서, 마이크로프로세서는 연산을 발행하도록 구성된 스케줄러와 상기 스케줄러에 의해서 발행된 메모리 연산을 실행하도록 구성된 로드 저장 유닛을 포함할 수 있다. 상기 로드 저장 유닛은 상기 로드 저장 유닛에 발행된 메모리 연산들을 식별하는 정보를 저장하도록 구성된다. 상기 발행된 메모리 연산들 중 하나에 대해 부정확한 데이터 추측이 검출됨에 따라, 상기 로드 저장 유닛은 상기 로드 저장 유닛의 발행된 메모리 연산들 중 적어도 하나가 재발행되어야 한다는 것을 나타내는 리플레이 표시를 상기 스케줄러에 제공하도록 구성된다. 상기 스케줄러는 상기 로드 저장 유닛에 의해서 식별되는 상기 메모리 연산들을 즉각적으로 반응하여 재발행하도록 구성된다.

[0007] 일 실시예에서, 상기 로드 저장 유닛은 부정확하게 추측된 메모리 연산의 어드레스와 일치(match)하는 어드레스를 갖는 각각의 메모리 연산들을 리플레이하도록 구성된다. 또한(혹은 대안적으로), 상기 로드 저장 유닛은 부정확하게 추측된 메모리 연산의 추측적 어드레스와 일치하는 어드레스를 갖는 각각의 메모리 연산들을 리플레이하도록 구성될 수 있다. 일 실시예에서, 상기 로드 저장 유닛은 부정확하게 추측된 메모리 연산의 어드레스와 일치하는 어드레스를 갖는 오직 로드 연산만을 리플레이할 수도 있다. 일부 실시예에서는, 상기 로드 저장 유닛은 어느 로드 연산이 상기 저장 연산으로부터 데이터를 전송(forwarding)하였는 지를 추적할 수 있고, 그리고 만약 저장 연산의 어드레스가 부정확하게 추측되는 경우, 상기 로드 저장 유닛은 상기 저장 연산으로부터 데이터가 전송된 어떠한 보다 최근(younger)의 로드 연산들이라도 리플레이할 수 있다. 또한, 일부 상황에서는 상기 로드 저장 유닛은 부정확한 데이터 추측이 검출되는 메모리 연산을 리플레이하도록 구성될 수 있다.

다수의 실시예에서, 메모리 연산의 어드레스의 추측값을 메모리 연산의 어드레스의 새로운 값과 비교하고, 및/또는 메모리 연산의 추측 결과를 메모리 연산의 비추측 결과와 비교함으로써, 부정확한 데이터 추측을 검출하도록 로드 저장 유닛이 구성될 수 있다.

다양한 방법의 실시예에서, 실행하도록 로드 저장 유닛에 연산을 발행하고, 이 로드 저장 유닛은 연산에서 실행되는 데이터 추측의 오류 지시를 수신하고, 이 지시에 따라, 로드 저장 유닛은 로드 저장 유닛 내의 미처리된(outstanding) 적어도 하나의 연산을 식별하는 리플레이 지시를 생성하고, 스케줄러는 리플레이 지시에 의해 식별된 연산을 응답가능하도록 재발행한다.

실 시 예

[0017] 도 1은 마이크로프로세서(100)의 일 실시예에 대한 블록도 이다. 마이크로프로세서(100)는 시스템 메모리(200) 내에 저장된 명령들을 실행하도록 구성된다. 많은 이러한 명령들이 상기 시스템 메모리(200) 내에 저장된 데이터를 조작하게 된다. 상기 시스템 메모리(200)는 컴퓨터 시스템을 통하여 물리적으로 분산될 수 있고, 그리고 하나 이상의 마이크로프로세서(100)에 의해서 액세스 될 수도 있다.

[0018] 마이크로프로세서(100)는 명령어 캐시(106) 및 데이터 캐시(128)를 포함할 수 있다. 마이크로프로세서(100)는

명령어 캐시(106)에 결합된 프리패치 유닛(108)을 포함한다. 디스패치 유닛(104)은 명령어 캐시(106)로부터 명령을 수신하도록, 그리고 연산을 스케줄러(들)(118)에 디스패치(dispatch) 하도록 구성될 수 있다. 하나 이상의 스케줄러들(118)은 디스패치 유닛(104)으로부터 디스패치된 연산을 수신하도록, 그리고 연산들을 하나 이상의 실행 코어(124)에 발행하도록 연결될 수 있다. 실행 코어들(124)은 데이터 캐시(128)에 액세스하도록 구성된 로드/저장 유닛을 포함할 수 있다. 상기 실행 코어(124)에 의해 생성된 결과들은 결과 버스(130)에 출력될 수 있다. 이러한 결과들은 후속하여 발행되고, 그리고 레지스터 파일(116)에 저장된 명령들에 대한 피 연산자 값들로서 사용될 수 있다. 퇴거 큐(retire queue)(102)가 상기 스케줄러(118) 및 디스패치 유닛(104)에 연결될 수 있다. 상기 퇴거 큐(102)는 발행된 각각의 연산이 언제 퇴거 되는지를 결정하도록 구성될 수 있다. 일 실시예에서, 상기 마이크로프로세서(100)는 x86 아키텍처와 호환되도록 설계될 수 있다. 또한, 마이크로프로세서(100)는 많은 다른 컴포넌트들을 포함할 수 있다. 예를 들면, 마이크로프로세서(100)는 분기 예측 유닛(도시되지 않음)을 포함할 수 있다.

[0019] 명령어 캐시(106)는 디스패치 유닛(104)에 의해 명령들이 수신되기 전에 상기 명령들을 일시적으로 저장할 수 있다. 프리패치(prefetch) 유닛(108)을 통하여 상기 시스템 메모리(200)로부터 코드를 프리패치함으로써 명령 코드가 명령어 캐시(106)에 제공될 수 있다. 명령어 캐시(106)는 다양한 설정들(예컨대, 세트-어소시에이티브(set-associative), 풀리-어소시에이티브(fully-associative), 또는 직접-맵핑)로 구현될 수 있다. 다수의 실시예에서, 명령어 캐시(106) 및/또는 데이터 캐시(128)의 다수 레벨이 있을 수 있다. 다수 레벨의 캐시는 도시된 바와 같이 마이크로프로세서(100)와 결합될 수 있고, 한편 다른 레벨의 캐시는 마이크로프로세서의 외부에 확장될 수 있다.

[0020] 프리패치(prefetch) 유닛(108)은 명령어 캐시(106)에 저장하기 위해 시스템 메모리(200)로부터 코드를 프리패치할 수 있다. 일 실시예에서, 프리패치 유닛(108)은 시스템 메모리(200)로부터 코드를 명령어 캐시(106)로 버스트(burst)하도록 구성될 수 있다. 프리패치 유닛(108)은 다양한 특정 코드 프리패치 기법 및 알고리즘을 사용할 수 있다.

[0021] 디스패치 유닛(104)은 실행 코어(124)가, 피 연산자 어드레스 정보, 즉시 데이터, 및/또는 변위 데이터에 의해서 실행가능한 비트-인코딩된 연산을 포함하는 신호를 출력할 수 있다. 일부 실시예에서, 디스패치 유닛(104)은 실행 코어(124) 내에서 실행가능한 연산으로 특정 명령들을 디코딩하기 위한 디코딩 회로(도시되지 않음)를 포함할 수 있다. 단순한 명령어들은 단독의 연산에 대응될 수 있다. 일부 실시예에서, 더욱 복잡한 명령들은 다중 연산과 대응될 수 있다. 연산이 레지스터의 업데이트를 포함하는 경우, 레지스터 파일(116) 내의 레지스터 위치가 추측적인 레지스터 상태(대안적인 실시예에서는, 재정렬 버퍼가 각각의 레지스터에 대한 하나 이상의 추측적인 레지스터 상태를 저장하도록 사용될 수 있음)를 저장하도록 예약(예컨대, 상기 연산의 디코드 위에서) 될 수 있다. 레지스터 이름 바꾸기를 가능하게 하기 위해서, 레지스터 맵이 소스 및 목적지 피 연산자의 논리 레지스터 이름을 물리 레지스터 이름으로 변환할 수 있다. 레지스터 맵은 레지스터 파일(116) 내의 어느 레지스터가 현재 할당되었는지, 그리고 할당이 해제되었는지를 추적할 수 있다.

[0022] 도 1의 마이크로프로세서(100)는 오더(order)를 벗어난 실행을 지원한다. 퇴거 큐(102)는 레지스터 읽기 및 쓰기 연산에 대한 원래 프로그램 시퀀스를 계속 추적할 수 있고, 추측적으로 명령이 실행될 수 있게 하며, 분기 오 예측이 된 경우 회복될 수 있도록 하고, 그리고 정밀한 예외 처리를 가능하게 한다. 일부 실시예에서, 퇴거 큐(102)는 연산들이 유효화됨에 따라 연산이 버퍼의 "하단"으로 이동하여, 새로운 엔트리를 위한 룸을 상기 큐의 "상단"에 제공하는 선입선출(first-in-first-out) 구성으로 구현될 수 있다. 퇴거 큐(102)는 연산이 실행을 완료한 것에 응답하여 연산을 퇴거하고, 어느 연산에 대해 실행된 어느 제어 추측 및 데이터가, 그 연산을 포함하는 그 연산까지 처리된 프로그램에 관하여, 검증이 이루어진다. 퇴거 큐(102)는 상기 물리 레지스터에서 그의 값을 생성하였던 연산이 퇴거될 때, 물리 레지스터의 추측 상태를 마이크로프로세서(100)의 구성적 상태에 커미트(commit)할 수 있다. 일부 실시예에서, 퇴거 큐(102)는 재정렬 버퍼의 부분으로서 구현될 수도 있다. 또한, 그러한 재정렬 버퍼는 레지스터 이름 바꾸기를 지원할 수 있도록 하기 위해서 추측 레지스터 상태에 대한 데이터 값 스토리지를 제공할 수 있다. 주목할 점으로서, 다른 실시예에서는 퇴거 큐(102)는 어떠한 데이터 값 스토리지도 제공하지 않을 수도 있다. 대신에, 연산이 퇴거 되면, 퇴거 큐(102)는 더 이상 추측 레지스터 상태를 저장할 필요가 없게 된 레지스터 파일(116) 내에 할당된 레지스터들의 할당을 해제하여, 어느 레지스터들이 현재 비어있다는 것을 나타내는 시그널을 레지스터 맵에 제공할 수 있다. 그러한 상태들을 생성하는 상기 연산들이 유효화될 때까지, 추측적 레지스터 상태를 레지스터 파일(116) 내에(또는, 대안적으로 재정렬 버퍼 내에) 유지하고, 만약 부정확하게 분기가 예측된 경우라면, 잘못 예측된 경로를 따라 추측적으로 실행된 연산의 결과는 무효화될 수 있다.

[0023] 특정한 연산을 수행시 요구되는 피 연산자가 레지스터 위치에 존재하는 경우, 레지스터 어드레스 정보는 레지스터 맵(또는, 재정렬 버퍼)으로 라우팅될 수 있다. 예를 들어, x86 아키텍처에서는, 32비트 논리 레지스터(예컨대, EAX, EBX, ECX, EDX, EBP, ESI, EDI, 그리고, ESP)가 8개 있다. 물리 레지스터 파일(116)(또는, 재정렬 버퍼)은 이러한 논리 레지스터의 콘텐츠를 변경하는 결과에 대한 스토리지를 포함하고, 이로 인해 오더를 벗어난 실행을 가능하게 한다. 레지스터 파일(116) 내의 물리 레지스터는 논리 레지스터들 중 하나의 레지스터의 콘텐츠를 수정하도록 결정된 각 연산의 결과를 저장하기 위해서 할당될 수 있다. 따라서, 특정 프로그램의 실행시에 다양한 포인터에서, 레지스터 파일(116)(또는, 대안적인 실시예에서는 재정렬 버퍼)은 주어진 논리 레지스터의 추측적으로 실행된 콘텐츠를 포함하는 하나 이상의 레지스터를 구비할 수 있다.

[0024] 레지스터 맵은 연산에 대한 목적지 피 연산자로서 지정된 특정의 논리 레지스터에 물리 레지스터를 할당할 수 있다. 디스패치 유닛(104)은, 소정의 연산에서 소스 피 연산자로 특정된 논리 레지스터에 할당된, 하나 이상의 이전에 할당된 하나 이상의 물리 레지스터들을 레지스터 파일(116)이 가지고 있다고 결정할 수 있다. 상기 레지스터 맵은 그 논리 레지스터에 가장 최근에 할당된 물리 레지스터에 대한 태그를 제공할 수 있다. 이 태그는 상기 레지스터 파일(116) 내의 피 연산자의 데이터 값을 액세스하는 데 사용되거나, 또는 상기 결과 버스(130) 상에 전송하는 결과를 통하여 데이터 값을 수신하는 데에 사용될 수 있다. 상기 피 연산자가 메모리 위치와 대응될 경우, 상기 피 연산자 값은 로드/저장 유닛(222)을 통하여 상기 결과 버스(결과 포워딩 및/또는 레지스터 파일(118) 내의 스토리지에 대해)에 제공될 수 있다. 상기 연산이 상기 스케줄러(118)들 중 하나에 의해서 발행되었을 때, 피 연산자 데이터 값들이 실행 코어(124)에 제공될 수 있다. 대안적인 실시예에서, 피 연산자 값들은 연산이 디스패치될 때 대응 스케줄러(118)에 제공(상기 연산이 발행될 때, 대응 실행 코어(124)에 제공되는 대신에)될 수 있다.

[0025] 상기 디스패치 유닛(104)의 출력에 제공된 상기 비트-인코딩 연산들 및 즉시 데이터는 하나 이상의 스케줄러(118)로 라우팅될 수 있다. 주목할 점으로서, 여기서 사용되는 바와 같이, 스케줄러는 연산이 실행할 준비가 되었을 때를 검출하고, 하나 이상의 기능 유닛에 준비 연산을 발행하는 장치이다. 예를 들면, 예약 스테이션으로서 스케줄러가 될 수 있다. 스케줄러 내의 연산 또는 스케줄러 내의 그룹이 또한 명령어 또는 연산 윈도우 또는 스케줄링 윈도우로 언급될 수도 있다. 각 스케줄러(118)는 실행 코어(124)에 발행되도록 대기하는 다수의 계류 연산에 대한 연산 정보(예컨대, 비트 인코딩된 실행 비트 및 피 연산자 값, 피 연산자 태그, 및/또는 즉시 데이터)를 유지할 수 있다. 일부 실시예에서는, 각각의 스케줄러(118)는 피 연산자 값 스토리지를 제공하지 않을 수 있다. 대신에, 각각의 스케줄러는 피 연산자 값이 언제 기능 유닛(126)에 의해서 읽혀질 수 있는지를 결정하기 위해, 레지스터 파일(116) 내에서 이용가능한 발행된 연산 및 결과를 모니터링할 수 있다(레지스터 파일(116) 또는, 결과 버스(130)로부터). 일부 실시예에서, 각 스케줄러(118)는 전용 기능 유닛(126)과 결합될 수 있다. 다른 실시예에서는, 단독의 스케줄러(118)가 기능 유닛(126)들 중 하나 이상에 연산을 발행할 수 있다.

[0026] 스케줄러(118)는 상기 실행 코어(124)에 의해서 실행될 연산 정보를 일시적으로 저장하도록 제공될 수도 있다. 전술된 바와 같이, 각각의 스케줄러(118)는 연산을 계류시키기 위해서 연산 정보를 저장할 수 있다. 부가적으로, 각각의 스케줄러는 이미 시행되었으나 재발행될 수 있는 연산들에 대한 연산 정보를 저장할 수 있다. 실행을 위해 이용가능하게 된 필요한 임의의 피 연산자의 값에 응답하여, 연산들이 실행되도록 실행 코어(124)에 발행된다. 따라서, 연산들이 실행되는 시퀀스는 원 프로그램 명령 시퀀스와 다를 수 있다. 데이터 추측이 부정확한 경우, 연산들이 재발행될 수 있도록 상기 연산들은 추측적이지 않을 때까지, 데이터 추측을 포함하는 연산이 스케줄러(118)에 잔여하게 된다. 도 1에서 예시된 바와 같이, 로드 저장 유닛(126C)은 재발행될 하나 이상의 연산들을 식별하는 리플레이 표시를 스케줄러(118)에 제공할 수 있다(예컨대, 그러한 리플레이 표시는 리플레이될 각각의 연산의 태그를 포함할 수 있다). 스케줄러(118)는 그러한 리플레이 지시에 의해 식별된 연산을 응답가능하도록 재발행할 수 있다.

[0027] 일 실시예에서, 각각의 상기 실행 코어들(124)은 다수의 기능 유닛(126)(예컨대, 도 1에 나타난 바와 같이 기능 유닛들(126A 내지 126C))을 포함할 수 있다. 일부 기능 유닛들 예컨대, 126A는 더하기 및 빼기의 정수 수학 연산, 그리고, 이동, 회전, 논리 연산, 그리고 분기 연산을 수행하도록 구성될 수 있다. 또한, 다른 기능 유닛 예컨대, 기능 유닛(126B)이 부동 소수점 연산을 수용하도록 포함될 수 있다. 기능 유닛들 중 하나 이상은 데이터 캐시(128) 및/또는 시스템 메모리에 저장된 데이터를 액세스하기 위해 로드 및 저장 연산을 수행하는 기능 유닛 예컨대, 126C에 의해서 수행될 로드 및 저장 메모리 연산을 위해 어드레스 생성을 수행하도록 구성될 수 있다. 일 실시예에서, 그러한 기능 유닛(126C)은 계류중인 로드 및/또는 저장들에 대한 데이터 및 어드레스 정보를 위해 다수의 스토리지 위치를 갖는 로드 저장 버퍼로 구성될 수 있다.

[0028] 또한, 상기 하나 이상의 기능 유닛들(126)은 조건적 분기 명령의 실행에 관한 정보를 분기 예측 유닛에 제공할

수 있는데, 이에 따라 상기 분기가 잘못 예측된 경우에는, 상기 분기 예측 유닛은 명령어 프로세싱 파이프라인에 입력되었던 잘못 예측된 분기에 후속하는 명령어를 플러쉬(flush)할 수 있고, 그리고 상기 프리패치 유닛(106)에 리다이렉트(redirect)할 수 있다. 다음으로, 상기 리다이렉트된 프리패치 유닛(106)은 명령어 캐시(106) 또는 시스템 메모리(200)로부터 정확한 명령 세트를 인출하기 시작한다. 그러한 상황에서, 추측적으로 실행되었던 것과 레지스터 파일(116)에 일시적으로 저장된 것들을 포함하고 있는 잘못 예측된 분기 명령 이후에 발생한 원 프로그램 시퀀스의 명령어의 결과는 폐기된다.

[0029] 실행 코어(들)(124) 내의 상기 기능 유닛(126)에 의해 생성된 결과는 레지스터 값이 업데이트 되는 경우 상기 결과 버스(130) 상에서 상기 레지스터 파일(116)에 출력된다. 메모리 위치의 콘텐츠가 변경되는 경우, 상기 실행 코어(들)(124) 내에서 생성된 결과들은 상기 로드/저장 유닛(126C)에 제공될 수 있다.

[0030] 데이터 캐시(128)는 실행 코어(124)와 상기 시스템 메모리(200) 사이에 전송되는 데이터를 일시적으로 저장하기 위해 제공되는 캐시 메모리이다. 전술된 명령어 캐시(106)와 같이, 상기 데이터 캐시(128)는 연산될 수 있는 일련의 구성들을 포함하는 다양한 특정 메모리 구성들로 구현될 수 있다. 부가적으로, 데이터 캐시(106) 및 명령어 캐시(128)는 일부 실시예에서는 단일화된 캐시로 구현될 수 있다.

[0031] 일부 실시예에서, 마이크로프로세서(100)는 통합 메모리 제어기(160)를 포함할 수 있고, 상기 마이크로프로세서가 상기 시스템 메모리(200)와 직접 인터페이스할 수 있도록 한다. 다른 실시예에서는, 메모리 제어기(160)는 마이크로프로세서(100)를 시스템 메모리(200)와 간접적으로 연결하는 버스 브리지 내에 포함될 수도 있다.

[0032] **데이터 추측**

[0033] 이하 기술되는 바와 같이, 데이터 값이 부정확한 것으로 판단되고, 그리고 후속하여 재계산된다고 판단될 가능성이 존재하는 경우, 데이터 값은 추측적인 것이라고 말할 수 있다. 추측적 데이터 값은 정확한 또는 부정확한 것처럼 확정적으로 식별될 수 없는 것이다. 데이터 값은 일부 데이터 추측이 수행되는 연산의 결과인 경우, 또는 상기 데이터 값이 다른 추측 데이터 값에 의존하는 경우(예컨대, 데이터 값이 하나 이상의 추측 피 연산자를 갖는 연산의 결과로서 발생 되는 경우) 데이터 값은 재컴퓨팅 될 수 있다. 추측적이지 않은 값이라 함은 어떠한 데이터 추측에도 의존하지 않는 값을 뜻한다(그러나, 그러한 값은 여전히 추측을 제어하기 쉬울 수 있다).

[0034] 마이크로프로세서(100)의 다양한 메커니즘이 데이터 추측을 수행할 수 있다. 예를 들면, 디스패치 유닛(104), 메모리 제어기(160), 그리고/또는 하나 이상의 기능 유닛(126)이 특정 연산에 대한 데이터 추측을 개별적으로 수행할 수 있다. 디스패치 유닛(104)은 하나의 연산 결과가 또 다른 연산에 대한 추측 피 연산자로서 사용될 수 있다고 검출할 수 있다. 예를 들면, 디스패치 유닛은 로드 연산이 이전 저장 연산에 의해서 데이터 캐시(128)에 저장된 데이터를 액세스할 것을 예측할 수 있다. 이에 응답하여, 상기 디스패치 유닛(104)은 로드 연산의 추측 결과와 같은 저장 연산의 소스로서 사용되는 레지스터 내에 저장된 데이터 값을 식별할 수 있다. 데이터 추측의 이 타입은 이하 의존성 예측이라고 한다. 의존성 예측은 로드 연산의 결과를 피 연산자로 지정하는 연산을 위해 추측적 피 연산자 소스처럼 저장 연산의 소스를 연결함으로써 디스패치 유닛(104) 내에서 확장될 수 있다. 의존성 예측의 또 다른 타입은 로드가 컴퓨팅되지 않은 어드레스를 갖는 저장을 우회하도록 함으로써 즉, 보다 최근의 로드가 이전 저장에 의존하지 않는다 라고 예측함으로써 로드 저장 유닛(126C) 내에서 수행될 수 있다.

[0035] 멀티프로세서 시스템에서, 메모리 제어기(160)가 캐시 코히런시(coherency)를 유지하기 위해 코히런시 체크를 수행할 수 있다. 메모리 제어기(160)는 다른 마이크로프로세서의 캐시들과 함께 코히런시 체크가 완료되기 전에, 시스템 메모리(200)로부터 캐시 라인의 사본을 추측적으로 반환할 수 있다. 상기 코히런시 체크가 후속하여 검색될 캐시 라인의 올바른 사본이 현재 다른 프로세서의 캐시에 저장되어 있다라고 결정하는 경우, 상기 시스템 메모리(200)로부터 추측적으로 검색된 캐시 라인의 사본은 무효로 될 수 있다. 따라서, 코히런시 체크가 완료되기 전까지는 상기 캐시 라인을 액세스하는 것으로부터 발생된 어떠한 로드 연산 결과라도 추측적인 것일 수 있다. 이 추측의 타입을 이하 메모리 예측이라 한다.

[0036] 디스패치 유닛(104)은 연산의 결과를 예측함으로써 데이터 추측을 수행할 수 있다. 예를 들면, 일부 연산들은 동일한 결과를 발생시키려는 경향이 있는데, 이에 따라 그러한 연산들 중 하나가 처리될 때마다, 기능 유닛(126)에 의해 연산을 실제 실행하기 이전에 디스패치 유닛(104)에 의해서 상기 결과가 추측적으로 생성될 수 있다. 이 데이터 추측의 타입을 이하 데이터 예측이라고 한다. 주목할 사항으로서, 데이터 예측은 또한 마이크로프로세서의 다른 부분(예컨대, 로드 저장 유닛(126C))에서 수행될 수도 있다.

[0037] 로드 저장 유닛(126C)은 추측적으로 어드레스를 생성할 수 있고, 그리고 이전에 처리된 로드들의 패턴에 기초하여 아직 컴퓨팅 되지 않은 어드레스를 갖는 로드 명령어의 결과를 상기 추측적 어드레스에 기초하여 추측적으로

생성할 수 있다. 예를 들면, 이전 N 로드 연산이 서로 일정한 오프셋 C 만큼(예컨대, A1, A2 = A1+C,...., AN=A(N-1)+C) 차지한 어드레스들 A1-AN을 타겟팅(targeting)하는 경우, 상기 로드 저장 유닛(126C)은 가장 최근에 액세스된 AN + 일정한 오프셋 C 어드레스에서 로드 연산의 결과로서 데이터를 반환할 수 있다. 이 타입의 데이터 추측을 이하 어드레스 예측이라고 한다. 주목할 사항으로서, 어드레스 예측의 다른 형태가 많은 실시예에서 사용될 수 있다.

[0038] 데이터 추측의 여러 다른 타입들은 일부 추측적 결과를 생성하도록 수행될 수 있다. 예를 들면, 정수 연산의 추측적 결과는 데이터 예측을 사용하여 생성될 수 있다. 이어서, 이 추측 결과는 저장 연산에 의해서 저장될 수 있다. 로드 연산은 이 저장에 의존하는 것으로 의존성 예측을 통하여 예측될 수 있고, 이에 따라 로드 연산의 추측 결과는 정수 연산의 추측 결과이다.

[0039] 또한, 데이터 추측이 수행되는 연산의 결과에 의존하는 연산들이 추측 결과를 생성할 수 있다. 예를 들면, 어드레스 예측이 로드 연산의 추측 결과를 생성하도록 사용되는 경우에는, 피 연산자로서 로드의 추측 결과를 사용하여 실행하는 어떠한 의존형 연산일지라도 추측적 결과를 생성할 수 있으며, 이는 다른 의존형 연산에 의해 피 연산자로서 번갈아 사용될 수 있다. 따라서, 로드 연산의 기본적인 추측이 부정확한 것으로 결정되는 경우, 상기 의존형 연산의 결과는 또한 부정확한 것으로 되고, 이에 따라 상기 로드에 의존하는 연산의 전체 의존성 체인(chain)은 정확한 결과를 생성하기 위해서 재실행될 필요가 있다. 반면, 기본적인 추측이 올바른 것으로 판명되는 경우, (그러한 결과들이 어떤 다른 추측 값에도 기초하지 않다고 가정하면) 상기 의존형 연산의 결과는 정확한 것으로 될 수 있다.

[0040] 데이터 추측이 수행되어 왔던 많은 연산들은 그러한 연산들이 기능 유닛에 의해서 실행될 때 검증될 수 있다. 예를 들면, 연산의 결과를 추측적으로 생성하기 위해 사용되는 데이터 예측은 연산의 실제 결과와 추측 결과를 비교함으로써 상기 연산을 실행하는 기능 유닛(126)에 의해서 검증될 수 있다. 상기 데이터 추측이 부정확한 경우 그러한 연산들은 다시 실행될 필요가 없는데, 이는 정확한 결과가 이미 이용가능하기 때문이다. 다른 연산들은 완전히 실행되지 않고서도 검증될 수 있다. 예를 들면, 컴퓨팅 되지 않은 어드레스를 갖는 로드가 이전 저장으로부터 그것의 결과를 포워딩 되는 경우(예컨대, 의존성 또는 어드레스 예측으로 인해), 로드의 추측 결과는 로드 어드레스가 계산될 때 검증될 수 있다. 상기 데이터 추측이 부정확한 경우, 그러한 연산은 올바른 결과를 생성하기 위해 재실행될(적어도 부분적으로 라도) 필요가 있을 수 있다.

[0041] 데이터 추측이 수행되었던 연산들 및 그들의 의존형 연산은 재실행될 필요가 있는데, 이로 인해 퇴거 큐(102)는 어떠한 기본 데이터 추측이 해결된 연산들 만을 퇴거하도록 구성될 수 있다.

[0042] **리플레이 메커니즘을 구비한 로드 저장 유닛**

[0043] 데이터 추측 실행을 지원하는 마이크로프로세서에서는, 하나 이상의 연산들이 부정확한 데이터 추측으로 인해 재실행될 필요가 있을 수 있다. 마이크로프로세서(100)는 연산들이 재실행될 수 있도록 하기 위한 다양한 리플레이 메커니즘을 포함할 수 있다. 예를 들면, 스케줄러(118)는 연산의 데이터 추측 피 연산자들 중 하나가 부정확하다 라는 표시에 따라 연산을 기능 유닛(126)에 재발행하도록 구성될 수 있다. 또한, 비슷하게, 로드 저장 유닛(126C)은 부정확하게 판명된 연산의 데이터 추측에 따라 상기 로드 저장 유닛에서 어느 미처리된 연산이 리플레이될 필요가 있는지를 검출하는 리플레이 메커니즘을 포함할 수 있다. 상기 로드 저장 유닛(126C)은 미처리된 연산이 리플레이될 필요가 있는 것처럼 스케줄러에 신호할 수 있고, 이는 상기 스케줄러가 그러한 연산들을 차후에 로드 저장 유닛(126C)에 재발행할 수 있도록 한다.

[0044] 도 2는 상기 로드 저장 유닛(126C)에 의해서 부정확한 데이터 추측으로 검출됨에 따라 연산을 리플레이하도록 구성되는 로드 저장 유닛(126C)의 일 실시예를 나타낸다. 데이터 추측 검증 로직(305)은 상기 로드 저장 유닛(126C)에 의해서 수행된 데이터 추측의 각각의 타입을 검증하도록 구성될 수 있다. 예를 들면, 로드 저장 유닛이 어드레스 예측(예컨대, 로드 연산의 어드레스를 예측함으로써)과 의존성 예측(예컨대, 계산되지 않은 어드레스를 갖는 이전 저장을 로드가 우회하도록 함으로써) 모두를 수행하는 경우, 상기 로드 저장 유닛은 어드레스와 의존성 예측 모두를 검증하도록 구성될 수 있다. 또한(혹은 대안적으로), 데이터 추측 검증 로직(305)은 마이크로프로세서(100)의 다른 부분에 의해서 수행된 데이터 추측을 검증하도록 구성될 수도 있다. 예를 들면, 디스패치 유닛(104)이 의존성 예측을 수행하도록 구성되는 경우(예컨대, 로드 결과를 이전 저장의 소스에 추측적으로 연결함으로써), 데이터 추측 검증 로직(305)은 상기 의존성 예측을 검증하도록 구성될 수 있다.

[0045] 또한, 로드 저장 유닛(126C)은 상기 로드 저장 유닛(예컨대, 스케줄러(118)에 의해)에 발행은 되었지만 아직 실

행이 완료되지는 않은 연산을 위해 연산 스토리지(307)를 포함한다. 데이터 추측이 실행되었던 로드 저장 유닛에 발행된 연산은 데이터 추측 검증 로직(305)에 의해서 검증이 완료될 때까지는 퇴거되지 않을 수 있다. 연산 스토리지(307)는 로드 저장 유닛(126C) 내의 모든 미처리 상태로 있는 연산들을 추적할 수 있다. 연산 스토리지(307)은 각각의 미처리 상태로 있는 로드 및 저장을 위한 엔트리(310)를 포함할 수 있다.

[0046] 또한, 엔트리(310)는 상기 엔트리가 로드 또는 저장에 할당되는 지를 나타내는 정보(313)를 포함할 수 있다(또는, 일부 실시예에서, 엔트리가 메모리 어드레스로부터 로드된 값에 영향을 주고, 메모리 어드레스에 대한 결과를 저장하는 연산에 대응하는 경우, 그것은 로드 및 저장 모두를 포함한다 라고 엔트리가 나타낼 수 있다). 부가적으로 엔트리(310)는 태그(315)(예컨대, 상기 마이크로프로세서 내에서 연산 및 그것의 결과를 식별하는 태그), 어드레스(317), 및/또는 데이터(319)를 포함할 수 있다. 일부 실시예에서, 각각의 엔트리의 데이터 필드(319)는 추측적 데이터와 추측적이지 않은 데이터 모두를 위한 스토리지를 포함할 수 있다. 비슷하게 일부 실시예에서, 상기 어드레스 필드(317)는 연산의 어드레스의 하나 이상의 값을 위한(예컨대, 어드레스 예측에 의해서 생성된 추측적 어드레스 및 연산을 실행함으로써 생성된 새로운 어드레스 값) 스토리지를 포함할 수 있다. 일부 실시예에서, 엔트리들은 연산자 및/또는 피 연산자를 데이터 추측으로서 식별하는 부가적 필드들을 포함할 수 있다. 엔트리(310)는 로드 저장 유닛(126C)에 연산을 발행하는 스케줄러(118)에 따라 발행될 수 있고, 연산의 실행을 완료하는 로드 저장 유닛(126C)에 따라 할당 해제할 수도 있다.

[0047] 데이터 추측 검증 로직(305)은 연산의 추측 결과와 연산의 실제 결과를 비교함으로써 데이터 추측의 특정 타입들(예컨대, 데이터 예측 및/또는 의존성 예측의 일부 타입들)을 검증할 수 있다. 예를 들면, 로드 연산의 추측 결과는 연산 스토리지(307) 내의 상기 로드의 엔트리(310)에 저장될 수 있다. 로드 연산의 실제 결과가 데이터 캐시(128)로부터 수신될 때, 상기 데이터 추측 검증 로직(305)은 연산 스토리지(307) 내에 저장된 추측 결과와 실제 결과를 비교할 수 있다. 데이터 추측 검증 로직(305)은 연산의 어드레스를 하나 이상의 이전 연산의 어드레스와 비교함으로써 데이터 추측의 다른 타입들(예컨대, 의존성 예측의 일부 타입들)을 검증할 수 있다. 데이터 추측 검증 로직(305)은 연산의 새로운 값이 상기 결과 버스(130)에 브로드캐스팅됨에 따라 추측 어드레스와 상기 어드레스의 새로운 값을 비교함으로써 데이터 추측의 다른 타입들(예컨대, 어드레스 예측)을 검증할 수 있다. 만약 상기 추측 어드레스가 어드레스의 새로운 값과 일치하지 않는 경우라면, 데이터 추측 검증 로직(305)은 상기 어드레스에서 수행된 데이터 추측이 부정확한 것이라고 결정할 수 있다.

[0048] 특정 연산에 대한 데이터 추측이 부정확하다고 검출됨에 따라, 데이터 추측 검증 로직(305)은 하나 이상의 연산이 리플레이되도록 한다. 연산은 상기 연산을 식별하는 리플레이 신호를 상기 스케줄러(118)에 제공함으로써 리플레이될 수 있다. 그러한 신호에 응답하여, 상기 스케줄러(118)는 리플레이를 위한 연산을 표시할 수 있다(예컨대, 상기 연산이 반드시 리플레이되어야 한다고 표시하기 위해 상기 연산과 관련된 상태 정보를 수정함으로써). 일 실시예에서, 데이터 추측 검증 로직(305)은 상기 연산이 반드시 리플레이되어야 한다고 표시하는 플래그에 따라 상기 스케줄러(118)에 상기 연산의 태그를 제공함으로써 상기 연산이 리플레이 되도록 할 수 있다.

[0049] 만약 부정확하게 추측된 연산이 재실행될 필요가 있는 경우(예컨대, 부정확한 의존성 예측 또는 부정확한 어드레스 예측으로 인해), 데이터 추측 검증 로직(305)은 상기 연산이 리플레이되도록 할 수 있다. 만약 부정확하게 추측된 결과의 정확한 결과가 이미 이용가능한 경우(예컨대, 만약 부정확한 결과가 부정확한 데이터 또는 의존성 예측으로 인한 것인 경우), 데이터 추측 검증 로직(305)은 로드 저장 유닛(126C)이 부정확하게 추측된 연산의 정확한 결과를 마이크로프로세서의 다른 컴포넌트에 브로드캐스팅 하도록 하여서, 그에 따라 마이크로프로세서의 다른 부분들 내의 어떠한 의존형 연산이라도 정확한 값을 사용하여 재실행될 수 있도록 한다. 일부 실시예에서는 데이터 추측 검증 로직(305)은 그러한 연산이 재실행되지 않도록 할 수도 있다. 주목할 사항으로서, 상기 로드 저장 유닛(126C)이 또한 상기 연산을 리플레이되도록 하는 경우일지라도, 로드 저장 유닛(126C)은 미처리 상태로 있는 연산의 실행을 완료할 수 있다.

[0050] 부정확하게 추측된 연산을 리플레이하는 것에 부가하여, 필요시에 데이터 추측 검증 로직(305)이 또한 상기 로직 저장 유닛에서 함께 현재 미처리 상태의 다른 연산들이 리플레이되도록 할 수 있다. 예를 들면, 일부 로드 연산들은 저장 연산의 추측적 어드레스에 기초하여 저장 연산으로부터 데이터를 포워딩할 수 있다. 상기 저장 연산의 추측 어드레스가 부정확한 것으로 결정되는 경우(예컨대, 상기 저장 연산의 어드레스를 생성하는 연산의 실행이 추측 어드레스와 같지 않은 새로운 어드레스 값을 생성하는 경우), 상기 저장의 추측 어드레스에 기초하여 데이터를 포워딩한 어떠한 로드 연산의 어드레스일지라도 또한 리플레이될 수 있다. 비슷하게, 어떠한 로드 연산일지라도 저장으로부터 데이터를 포워딩할 수 있도록 하기 위해서, 저장 연산에 대해 새로운 어드레스와 일치하는 어드레스를 갖는 어떠한 로드 연산일지라도 리플레이될 수 있다. 비슷하게, 데이터 추측 검증 로직(30

5)은 저장 연산에 대한 피 연산자가 언제 재 브로드캐스팅 되는지를 검출하도록, 그리고 저장 연산의 피 연산자의 이전 값을 포워딩한 어떠한 의존형 로드 연산이 즉각 반응하여 리플레이되도록 구성될 수 있다.

[0051] 일 실시예에서, 상기 데이터 추측 검증 로직(305)은 부정확하게 추측된 연산 보다 오래되지 않은(younger) 로드 저장 유닛 내의 미처리 상태로 있는 모든 연산들이 리플레이되도록 함으로써 의존형 연산을 리플레이할 수 있다. 일부 실시예에서, 비교기 로직(303)이 리플레이하기 위한 연산 스토리지(307) 내에서 비교적 조기의 연산들을 식별할 수 있다. 비교기 로직(303)은 값들을 비교하기 위한 그리고/또는 일치하는 값들을 검출하기 위한 임의의 다양한 수단들을 포함할 수 있다(예컨대, 비교기, 콘텐츠-어드레스 지정 가능한 메모리, 등등)

[0052] 다른 실시예에서, 데이터 추측 검증 로직(305)은 특정 연산에 대한 부정확한 데이터가 검출됨에 따라 특정의 보다 최근의 연산들을 선택적으로 리플레이하기 위한 비교기 로직(303)을 사용할 수 있다. 예를 들면, 일 실시예에서, 저장 연산의 어드레스에 대한 부정확한 데이터 추측이 검출됨에 따라, 비교기 로직(303)은 저장의 어드레스의 추측 값 및 새로운 값을 연산 스토리지(307) 내에서 각각의 보다 최근의 연산의 어드레스들과 비교할 수 있다. 임의의 보다 최근의 연산의 어드레스들이 저장의 어드레스들 중 어느 것과 일치하는 경우, 상기 일치하는 보다 최근의 연산은 상기 스케줄러에 그러한 연산을 리플레이할 필요를 나타내는 표시를 제공함으로써 재실행될 수 있도록 된다.

[0053] 일부 실시예에서, 로드 저장 유닛(126C)은 미처리된 로드의 어드레스를 오래된 저장의 어드레스와 비교함으로써 저장-대-로드-포워딩을 구현할 수 있다. 만약 로드의 어드레스가 오래된 저장의 어드레스와 일치하는 경우, 상기 로드 저장 유닛(126C)은 오래된 저장 연산에 의해서 저장된 데이터를 로드 연산의 결과로서 출력할 수 있다. 로드 연산이 저장 연산으로부터 데이터를 포워딩할 때마다, 상기 로드 저장 유닛(126C)은 상기 저장 연산을 식별하는 태그를 포워딩된 로드 연산과 관련된 포워딩 트래킹(forwarding tracking) 버퍼(309) 엔트리에 저장할 수 있다. 이러한 실시예들 중 일부에서는, 비교기 로직(303)이 연산에 대한 데이터 추측이 부정확하다고 검출하는 데이터 추측 검증 로직(305)에 따라 재실행하기 위해 연산을 식별하도록 포워딩 트래킹 버퍼(309)에 의해서 제공되는 정보를 사용할 수 있다. 예를 들면, 저장 연산에 대한 어드레스 예측이 부정확하다고 검출되는 경우, 비교기 로직(303)은 저장 연산에 대한 어드레스의 새로운 값을 연산 스토리지(307) 내의 각각 보다 최근의 로드 연산의 어드레스와 비교하도록 구성될 수 있다. 어떠한 일치하는 연산일지라도 부정확하게 추측된 저장으로부터 데이터를 포워딩했어야 함에도, 부정확하게 추측된 저장 어드레스로 인해 포워딩하지 않았다. 따라서, 로드 저장 유닛(126C)은 이러한 일치하는 연산들이 재실행될 수 있도록 한다. 부가적으로, 부정확하게 추측된 저장 연산의 태그는 보다 최근의 로드에 대한 포워딩 트래킹 버퍼(309)에 저장된 태그들과 비교될 수 있다. 부정확하게 추측된 저장 연산의 태그가 보다 최근의 로드를 위해 저장된 태그와 일치하는 경우, 부정확하게 추측된 저장의 어드레스에 기초하여 보다 최근의 로드가 부정확하게 데이터를 포워딩한다고 나타내는 상기 로드 저장 유닛(126C)은 상기 보다 최근의 로드가 재실행될 수 있도록 한다.

[0054] 도 3A는 부정확한 데이터 추측이 검출됨에 따라 로드 저장 유닛 내에서 미처리 상태로 있는 연산을 리플레이하기 위한 방법의 일 실시예를 나타낸다. (501)에서, 연산에 대한 데이터 추측이 부정확하다는 표시가 검출된다. 예를 들면, 로드 저장 유닛은 연산에 대한 어드레스의 새로운 값을 (일부 실시에에서는 로드 저장 유닛의 부분일 수 있는) 어드레스 생성 유닛으로부터 수신할 수 있다. 상기 새로운 어드레스 값은 상기 연산 또는 다른 연산의 추측 결과를 생성하도록 상기 로드 저장 유닛에 의해서 사용되는 추측 어드레스 값과는 다르며, 이는 상기 어드레스에서 수행되는 데이터 추측이 부정확하다고 나타낸다. 다른 실시예에서, 로드 저장 유닛은 로드의 추측 결과(예컨대, 상기 로드 결과가 이전 저장의 소스와 동일하다고 예측하여 생성된)를 상기 데이터 캐시를 액세스하여 얻어진 로드의 실제 결과와 비교함으로써 부정확한 의존성 예측을 검출할 수 있다.

[0055] 도 3A의 실시예에서, 상기 로드 저장 유닛은 (503)에서와 같이 연산에 대한 데이터 추측이 부정확하다고 검출되는 것에 따라 상기 로드 저장 유닛 내에서 미처리 상태로 있는 보다 최근의 연산 전부가 재실행되도록 할 수 있다. 상기 로드 저장 유닛은 상기 보다 최근의 연산이 리플레이될 각각의 연산의 태그를 스케줄러에 제공함으로써 재실행될 수 있도록 한다. 상기 로드 저장 유닛에 의해서 제공된 태그에 따라, 상기 스케줄러는 상기 스케줄러 내부의 연산의 상태를 업데이트(예컨대, "발행됨"에서 "발행되지 않음", 또는 "발행될 필요가 있음")할 수 있다. 그러한 리플레이 메커니즘은 상기 보다 최근의 연산들이 마이크로프로세서의 프로세싱 파이프라인으로부터 플러시(flush)되지 않고서도, 그리고 상기 명령어 캐시(106)로부터 재인출되도록 하지 않고서도 리플레이될 수 있도록 한다.

[0056] 도 3B는 상기 로드 저장 유닛 내에서 미처리 상태로 있는 연산들을 재실행하는 방법의 다른 실시예를 나타낸다. (505)에서, 저장의 어드레스에 대한 상기 데이터 추측이 부정확하다는 표시가 검출된다. 어떠한 미처리 상태의

보다 최근의 로드들 또는 저장들이 부정확하게 추측된 저장의 어드레스의 추측 값 또는 새로운 값과 일치하는 어드레스를 갖는 경우, (507)에서 결정되는 바와 같이 일치하는 연산들이 재 샐행될 수 있다. 어떠한 보다 최근의 연산들의 어드레스라도 저장 연산의 어드레스의 추측 값 또는 새로운 값 중 어느 것과도 일치하지 않는 경우, 상기 보다 최근의 연산의 어드레스는 (509)에서와 같이 리플레이되지 않을 수 있다.

[0057] 도 3C는 로드 저장 유닛 내에서 미처리 상태의 연산을 재실행하는 방법의 다른 실시예를 나타낸다. (515)에서, 저장 연산의 어드레스에 대한 데이터 추측이 부정확하다 라는 표시가 검출된다. 어떠한 보다 최근의 로드의 어드레스라도 저장 연산의 어드레스의 새로운 값과 일치하는 경우, 상기 일치하는 보다 최근의 로드들은 리플레이될 수 있다. 어떠한 보다 최근의 로드들일지라도 부정확하게 추측된 저장으로부터 데이터가 포워딩된 경우에는 (예컨대, 상기 보다 최근의 로드가 데이터를 포워딩한 저장 연산의 태그를 부정확하게 추측된 저장의 태그와 비교함으로써 결정되는 바와 같이), (521)에서 결정되는 바와 같이, 상기 일치하는 보다 최근의 로드들이 리플레이될 수 있다.

[0058] 주목할 사항으로서, 다른 실시예들은 도 3A 내지 3C와는 다르게 동작할 수 있다. 예를 들면, 도 3B와 유사한 실시예에서, 부정확하게 추측된 저장의 어드레스의 추측 값 또는 새로운 값과 일치하는 어드레스를 갖는 오직 보다 최근의 로드 연산만이 재실행될 수 있다. 비슷하게, 도 3C와 비슷한 실시예에서는, 부정확하게 추측된 저장의 어드레스의 새로운 값과 일치하는 어드레스를 갖는 보다 최근의 로드 및 저장이 재실행될 수 있다. 많은 다른 변형도 가능하다.

[0059] **예시적인 컴퓨터 시스템**

[0060] 도 4는 버스 브리지(902)를 통하여 다양한 시스템 컴포넌트에 결합된 프로세서(100)를 포함하는 컴퓨터 시스템(900)의 일 실시예에 대한 블록도를 나타낸다. 프로세서(100)는 상기 나타난 바와 같이 로드 저장 유닛의 실시예들을 포함할 수 있다. 컴퓨터 시스템에 대한 다른 실시예들도 가능하며, 고려될 수 있는 것이다. 도시된 시스템에서, 메인 메모리(200)는 메모리 버스(906)를 통하여 버스 브리지(902)에 결합되고, 그리고 그래픽 제어기(908)는 AGP 버스(910)을 통하여 버스 브리지(902)에 결합된다. 다수의 PCI 디바이스들(912A 내지 912B)은 PCI 버스(914)를 통하여 버스 브리지(902)에 결합된다. 또한, 2차 버스 브리지(916)가 EISA/ISA 버스(920)를 통하여 하나 이상의 EISA 또는 ISA 디바이스들(918)에 대한 전기적 인터페이스를 수용하기 위해서 제공될 수 있다. 이 예에서, 프로세서(100)는 CPU 버스(924)를 통하여 버스 브리지(902) 및 선택적 L2 캐시(918)에 결합된다. 일부 실시예에서, 상기 프로세서(100)는 통합된 L1 캐시(도시되지 않음)를 포함할 수 있다.

[0061] 버스 브리지(902)는 프로세서(100), 메인 메모리(200), 그래픽 제어기(908), 그리고 PCI 버스(914)에 부착된 디바이스들 사이의 인터페이스를 제공한다. 동작이 버스 브리지(902)에 결합된 디바이스들 중 하나로부터 수신되었을 때, 버스 브리지(902)는 동작의 타깃을 식별(예컨대, 특정 디바이스, 또는 PCI 버스(914)의 경우에는, 상기 타깃이 PCI 버스(914) 상에 있는 디바이스)한다. 버스 브리지(902)는 타겟된 디바이스에 동작을 라우팅한다. 버스 브리지(902)는 일반적으로 소스 다비이스들 또는 버스에 의해서 사용된 프로토콜로, 상기 타겟 디바이스 또는 버스에 의해서 사용되는 프로토콜의 동작을 변환한다.

[0062] 인터페이스를 PCI 버스(914)에 대한 ISA/EISA 버스에 제공하는 것에 더하여 2차 버스 브리지(916)는 추가적인 기능을 통합할 수 있다. 2차 버스 브리지(916)와 외부로 된 또는 2차 버스 브리지(916)와 통합된 입력/출력 제어기(도시되지 않음)는 또한 키보드 및 마우스(922)를 위해 그리고 다양한 시리얼 또는 병렬 포트를 위해 동작 가능한 지원을 제공하기 위해서 컴퓨터 시스템(900) 내에 통합될 수 있다. 또한, 다른 실시예에서는, 외부 캐시 유닛(도시되지 않음)이 프로세서(100)와 버스 브리지(902) 사이의 CPU 버스(924)에 결합될 수 있다. 대안적으로, 상기 외부 캐시는 버스 브리지(902)에 결합될 수 있고, 그리고 외부 캐시에 대한 캐시 제어 로직이 버스 브리지(902)에 통합될 수 있다. L2 캐시(928)는 프로세서(100)에 대한 백사이드(backside) 구성으로 나타난다. 주목할 점으로는 L2 캐시(928)는 프로세서(100)와 분리되어 프로세서(100)와 함께 카트리지(예컨대, 슬롯 1 또는 슬롯 A)로 통합될 수 있거나, 혹은 심지어 프로세서(100)와 함께 반도체 기판에 통합될 수도 있다는 점이다.

[0063] 메인 메모리(200)는 어플리케이션 프로그램이 저장되고, 그리고 프로세서(100)가 초기에 실행하는 메모리이다. 적용가능한 메인 메모리(200)는 DRAM(Dynamic Random Memory)를 포함한다. 예를 들면 SDRAM(Synchronous DRAM) 또는 Rambus DRAM(RDRAM)의 다수의 뱅크들이 적용될 수 있다.

[0064] PCI 디바이스들(912A 내지 912B)은 네트워크 인터페이스 카드, 비디오 가속기, 오디오 카드, 하드, 또는 플로피

디스크 드라이브, 또는 드라이브 제어기, SCSI(Small Computer Systems Interface) 어댑터, 그리고 전화 카드들과 같은 다양한 주변 디바이스들의 예시이다. 비슷하게, ISA 디바이스들(918)은 모뎀(modem), 사운드 카드와 같은 다양한 타입의 주변 디바이스들, 그리고 GPIB 또는 필드 버스 인터페이스 카드와 같은 다양한 데이터 획득 카드들에 대한 예시이다.

[0065] 그래픽 제어기(908)는 디스플레이(926)에 텍스트 및 이미지를 렌더링하는 것을 제어하기 위해서 제공되는 것이다. 그래픽 제어기(908)는 메인 메모리(200) 내외로 효율적으로 이동될 수 있는 3차원 데이터 구조를 렌더링하기 위한 기술로서 공지된 일반적인 그래픽 가속을 구현할 수 있다. 따라서, 그래픽 제어기(908)는 메인 메모리(200)에 대한 액세스를 얻기 위해 버스 브리지(902) 내의 타깃 인터페이스에 대한 액세스를 요청 및 수신할 수 있다는 점에서 AGP 버스(910)의 마스터일 수 있다. 전용 그래픽 버스는 메인 메모리(904)로부터의 데이터를 고속으로 검색할 수 있게 한다. 특정한 동작을 위해서, 그래픽 제어기(908)는 AGP 버스(910)에 PCI 프로토콜 트랜잭션을 발생시키도록 더 구성될 수 있다. 따라서, 버스 브리지(902)의 AGP 인터페이스는 AGP 프로토콜 트랜잭션 및 PCI 프로토콜 타깃과 초기 트랜잭션을 모두 지원할 수 있는 기능을 포함할 수 있다. 디스플레이(926)는 이미지 또는 텍스트가 표현될 전기 디스플레이이다. 적용가능한 디스플레이(926)로는 CRT(cathode ray tube), LCD(liquid crystal display), 등등을 포함한다.

[0066] 주목할 점으로는 상기 AGP, PCI, 그리고 ISA 또는 EISA 버스들이 전술된 설명에서 예시적으로 사용되었으나, 임의의 버스 아키텍처라도 바람직하게 대체될 수 있다 라는 점이다. 또한, 주목할 점으로는 컴퓨터 시스템(900)은 추가적인 프로세서(예컨대, 컴퓨터 시스템(900)의 선택적 컴포넌트로서 나타난 프로세서(100a))를 포함하는 멀티프로세싱 컴퓨터 시스템일 수 있다 라는 것이다. 프로세서(100a)는 프로세서(100)와 비슷한 것일 수 있다. 더욱 상세히, 프로세서(100a)는 프로세서(100)와 동일한 프로세서일 수 있다. 프로세서(100a)는 독립 버스(도 4에 도시된 바와 같은)를 통하여 버스 브리지(902)에 결합될 수 있거나, 혹은 프로세서(100)와 함께 CPU 버스(924)를 공유할 수 있다, 더욱이, 프로세서(100a)는 L2 캐시(928)과 비슷한 선택적 L2 캐시(928a)에 결합될 수 있다.

[0067] 도 5에서는 상술된 것 같은 로드 저장 유닛의 실시예를 포함할 수 있는 컴퓨터 시스템(900)에 대한 예시적인 실시예가 나타나 있다. 다른 실시예들도 또한 가능하고, 고려될 수 있다. 도 5의 실시예에서는, 컴퓨터 시스템(900)은 다수의 프로세싱 노드들(1012A, 1012B, 1012C, 및 1012D)을 포함할 수 있다. 각각의 프로세싱 노드는 개별 프로세싱 노드(1012A 내지 1012D) 내에 포함된 메모리 제어기(1016A 내지 1016D)를 통하여 개별 메모리(200A 내지 200D)에 결합될 수 있다. 추가로, 프로세싱 노드들(1012A 내지 1012D)은 프로세싱 노드들(1012A 내지 1012D) 간에 통신할 수 있게 하기 위해 사용되는 인터페이스 로직을 포함한다. 예를 들면, 프로세싱 노드(1012A)는 프로세싱 노드(1012B)와 통신하기 위한 인터페이스 로직(1018A)을 포함하고, 프로세싱 노드(1012C)와 통신하기 위한 인터페이스 로직(1018B), 그리고 또 다른 프로세싱 노드(도시 되지 않음)와 통신하기 위한 제 3 인터페이스 로직(1018C)를 포함한다. 비슷하게 프로세싱 노드(1012B)는 인터페이스 로직(1018D, 1018E, 및 1018F)을 포함하고; 프로세싱 노드(1012C)는 인터페이스 로직(1018G, 1018H, 및 1018I)을 포함하고; 그리고, 프로세싱 노드(1012D)는 인터페이스 로직(1018J, 1018K, 및 1018L)을 포함한다. 프로세싱 노드(1012D)는 인터페이스 로직(1018L)을 통하여 다수의 입력/출력 디바이스들(예컨대, 데이지 체인 구성(daisy chain configuration)에서 디바이스들(1020A 내지 1020B))와 통신하기 위해서 결합된다. 다른 프로세싱 노드들은 비슷한 방식으로 다른 I/O 디바이스들과 통신할 수 있다.

[0068] 프로세싱 노드들(1012A 내지 1012D)은 내부-프로세싱 노드 통신을 위해 패킷 기반 결합을 구현한다. 본 예에서, 상기 결합은 단방향 결합의 세트들(예컨대, 라인들(1024A)이 프로세싱 노드(1012A)에서 프로세싱 노드(1012B)로 패킷을 전송하기 위해서 사용되고, 그리고 라인들(1024B)이 프로세싱 노드(1012B)에서 프로세싱 노드(1012A)로 패킷을 전송하기 위해서 사용된다)로 구현된다. 라인들(1024C 내지 1024H)의 다른 세트들은 도 5에 도시된 바와 같이 다른 프로세싱 노들 간에 패킷을 전송하기 위해서 사용된다. 일반적으로 라인들(1024)의 각각의 세트는 하나 이상이 데이터 라인들, 상기 데이터 라인들에 대응되는 하나 이상이 클록 라인들, 그리고 패킷이 전송되는 타입을 표시하는 하나 이상의 제어 라인들을 포함할 수 있다. 상기 결합은 프로세싱 노들 간의 통신을 위해 캐시 코히런시 방식으로 동작할 수 있거나, 혹은 프로세싱 노드와 I/O 디바이스(또는 PCI 버스 또는 ISA 버스와 같은 종래 구성의 I/O 버스에 대한 버스 브리지) 간에 통신을 위해서 넌 코히런시(non-coherency) 방식으로 동작될 수 있다. 더욱이, 상기 결합은 도시된 것과 같은 I/O 디바이스들 사이의 데이지 체인(daisy-chain) 구조를 사용하여 넌 코히런시 방식으로 동작될 수도 있다. 주목할 점으로는 하나의 프로세싱 노드에서 다른 프로세싱 노드로 전송되는 패킷은 하나 이상의 중간 노드를 통하여 전송될 수 있다라는 것이다. 예를 들면, 도 5에 나타난 바와 같이 프로세싱 노드(1012A)에 의해 프로세싱 노드(1012D)로 전송되는 패킷은 프로세싱 노드(1012B) 또는 프로세싱 노드(1012C)를 통하여 전송될 수 있다. 임의의 적용가능한 라우팅 알고리즘이 사용될 수 있다. 컴

퓨터 시스템(900)이 다른 실시예는 도 5에 나타난 바와 같은 실시예 보다 더 많거나, 혹은 더 적은 프로세싱 노드들을 포함할 수 있다.

[0069] 일반적으로, 상기 패킷들은 노드들 간의 상기 라인들(1024)에 하나 이상의 비트 타임들로 전송될 수 있다. 비트 타임은 대응 클록 라인 위의 클록 신호의 상승 또는 하강 에지일 수 있다. 상기 패킷들은 트랜잭션을 개시하기 위한 커맨드 패킷들, 캐시 코히런시를 유지하기 위한 프로브 패킷들, 그리고 프로브 및 커맨드들에 응답하는 것으로부터의 응답 패킷을 포함할 수 있다.

[0070] 프로세싱 노드들(1012A 내지 1012D)은 메모리 제어기 및 인터페이스 로직에 더하여 하나 이상의 프로세서들을 포함할 수 있다. 일반적으로, 프로세싱 노드는 적어도 하나의 프로세서를 포함하고, 그리고 선택적으로는 메모리 및 다른 로직과 통신하기 위한 메모리 제어기를 포함할 수 있다. 더욱 상세히는, 프로세싱 노드(1012A 내지 1012D)는 하나 이상의 프로세서(100) 들을 포함할 수 있다. 외부 인터페이스 유닛은 메모리 제어기(1016) 뿐 아니라, 노드 내의 인터페이스 로직(1018)을 포함할 수 있다.

[0071] 메모리들(200A 내지 200D)은 임의의 적용가능한 메모리 디바이스들을 포함할 수 있다. 예를 들면, 메모리들(200A 내지 200D)은 하나 이상의 RAMBUS DRAM들(RDRAM들), SDRAM들(synchronous DRAM들), SRAM(static RAM)을 포함할 수 있다. 컴퓨터 시스템(900)의 어드레스 공간은 메모리들(200A 내지 200D) 간에 나누어진다. 각각의 프로세싱 노드들(1012A 내지 1012D)은 어느 어드레스가 어느 메모리들(200A 내지 200D)에 맵핑되었는지를, 그리고 나아가 특정 어드레스에 대한 메모리의 요청이 어느 프로세싱 노드들(1012A 내지 1012D)에 라우팅 되어야할지를 결정하기 위해서 사용되는 메모리 맵을 포함한다. 일 실시예에서, 컴퓨터 시스템(900) 내의 어드레스에 대한 코히런시 포인트(coherency point)는 어드레스에 대응되는 바이트들을 저장하는 메모리에 결합된 메모리 제어기(1016A 내지 1016D)이다. 즉, 상기 메모리 제어기(1016A 내지 1016D)는 대응 메모리들(200A 내지 200D)에 대한 각각의 메모리 액세스가 캐시 코히런시 방식으로 발행하는지를 보증할 책임이 있다. 메모리 제어기(1016A 내지 1016D)들은 메모리들(200A 내지 200D)에 인터페이스하기 위한 제어 회로를 포함할 수 있다. 추가로, 메모리 제어기(1016A 내지 1016D)들은 메모리 요청을 큐에 넣기 위한 요청 큐를 포함할 수 있다.

[0072] 인터페이스 로직들(1018A 내지 1018L)은 상기 결합으로부터 패킷을 수신하기 위한 그리고 상기 결합에 전송될 패킷을 버퍼링하기 위한 다양한 버퍼를 포함한다. 컴퓨터 시스템(900)은 패킷들을 전송하기 위해 임의의 적용가능한 흐름 제어 메커니즘을 사용할 수 있다. 예를 들면, 일 실시예에서, 각각의 인터페이스 로직(1018)은 상기 인터페이스 로직이 연결되는 결합의 다른 종단에서 수신 내의 버퍼의 각 타입의 카운트 수를 저장한다. 상기 인터페이스 로직은 상기 수신 인터페이스 로직이 패킷을 저장할 버퍼가 비어있지 않다면 패킷을 전송하지 않는다. 수신하는 버퍼가 전송되는 패킷을 라우팅함으로써 비어짐에 따라, 상기 수신하는 인터페이스 로직은 상기 버퍼가 비어있다는 것을 나타내기 위해 메시지를 송신하는 인터페이스 로직에 전송한다. 그러한 메커니즘은 "쿠폰-기반(coupon-based)" 시스템으로 언급될 수 있다.

[0073] I/O 디바이스들(1020A 내지 1020B)은 임의의 적용가능한 I/O 디바이스들일 수 있다. 예를 들면, I/O 디바이스들(1020A 내지 1020B)은 상기 디바이스들이 연결되는 다른 컴퓨터 시스템과 통신하기 위한 디바이스들을 포함할 수 있다(예컨대, 네트워크 인터페이스 카드, 또는 모뎀). 더욱이, I/O 디바이스들 (1020A 내지 1020B)은 비디오 가속기, 오디오 카드들, 하드 또는 플로피 디스크 드라이브, 또는 드라이브 제어기, SCSI(Small Computer Systems Interface) 어댑터들, 그리고 전화 카드, 사운드 카드, 그리고 GPIB 또는 필드 버스 인터페이스 카드와 같은 다양한 데이터 획득 카드들을 포함할 수 있다. 주목할 점은 "I/O 디바이스"라는 용어 및 "주변 디바이스"라는 용어는 여기서 동의어를 의미한다.

[0074] 여기서 사용되는 바와 같이, "클록 사이클" 또는 "사이클"이라는 용어는 명령어 프로세싱 파이프라인이 다양한 단계에서 그들의 작업을 완료하는 시간 간격을 의미한다. 명령어 및 계산된 값들은 상기 클록 사이클을 정의하는 클록 신호에 따라 (레지스터 또는 배열과 같은) 메모리 요소들에 의해서 취득된다. 예를 들면, 메모리 요소는 클록 신호의 상승 또는 하강 에지에 따라 값을 취득될 수 있다.

상기의 설명은 신호를 "어서트(assert)" 되는 것으로서 설명한다. 정보의 특정적인 일부를 나타내는 값을 전송할 때, 그 신호는 "어서트"되는 것으로서 정의될 수 있다. 특정의 신호는, 이진 값 '1'의 신호를 전송할 때, 또는 이진 값 '0'의 값을 전송할 때, '어서트'되는 것으로 정의될 수 있다.

[0075] 일단 상기 개시를 충분히 이해하고 나면 수많은 변형과 수정이 이루어질 수 있다는 것은 당업자에게 자명한 사항이다. 하기의 청구범위는 이러한 모든 변형과 수정을 포함하도록 해석되어야 한다.

도면의 간단한 설명

[0008] 상세한 설명을 첨부된 도면과 함께 읽음으로써, 본 발명이 더욱 명확히 이해될 수 있다.

[0009] 도 1은 일 실시예를 따르는 마이크로프로세서를 나타낸다.

[0010] 도 2는 일 실시예를 따르는 로드 저장 유닛의 블록 도를 나타낸다.

[0011] 도 3A는 일 실시예를 따라 로드 저장 유닛에서 연산들을 리플레이하는 방법에 대한 흐름 도이다.

[0012] 도 3B는 다른 실시예를 따라 로드 저장 유닛에서 연산을 리플레이하는 방법에 대한 흐름 도이다.

[0013] 도 3C는 다른 실시예를 따라 로드 저장 유닛에서 연산을 리플레이하는 방법에 대한 흐름 도이다.

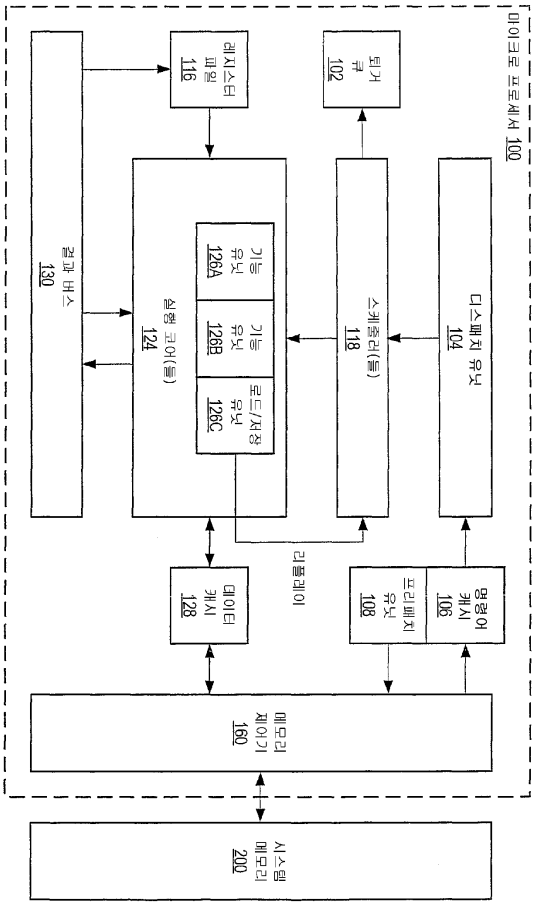
[0014] 도 4는 일 실시예를 따르는 예시적인 컴퓨터 시스템을 나타낸다.

[0015] 도 5는 또 다른 실시예를 따르는 또 다른 예시적인 컴퓨터 시스템을 나타낸다.

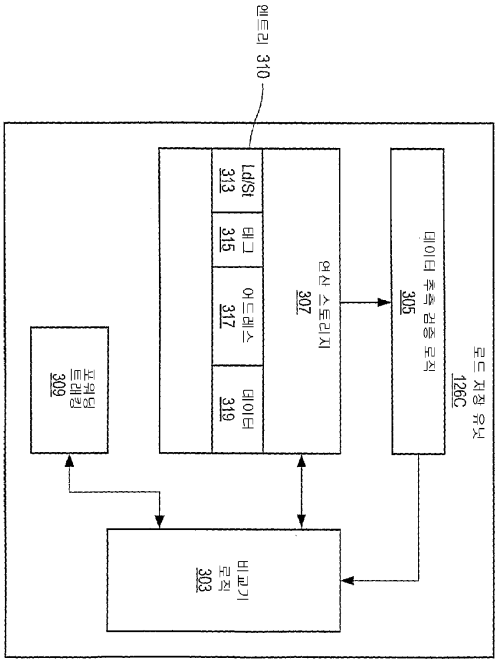
[0016] 본 발명이 다양한 수정 및 대안적인 형태로 될 수 있으나, 이 중에서 특정한 실시예들만이 도면에 예시로서 나타나 있고, 상세히 설명되어 있다. 그러나 도면 및 상세한 설명이 본 발명을 개시된 특정 형태로만 제한하는 것은 아니라, 이와는 반대로 본 발명은 청구범위에 기재된 사항에 의해서 파악된 발명의 사상 및 범위 내에 있는 모든 수정, 균등물, 그리고 대안을 포함한다. 제목은 단지 체계성을 위해서 사용된 것이며, 설명 또는 청구범위를 제한하거나 해석하는데 사용되어야 한다는 것을 의미하는 것은 아니다. 더욱이, "~일 수"라는 단어는 가능함(즉, 잠재성을 가진 것)으로 이 출원 전체에서 사용되었고, 없어서는 안 될 사항(즉, 필연사항)을 의미하는 것으로 사용된 것이 아니다. "포함된"이라는 용어 및 그의 파생어는 "포함하지만, 그에 한정되지 않음"을 의미한다. "접속된"이라는 용어는 "직접 또는 간접으로 접속됨"을 의미하고, "연결됨"이라는 용어는 "직접 또는 간접으로 연결됨"을 의미한다.

도면

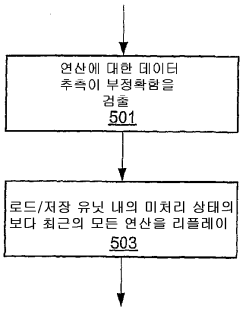
도면1


마이크로 프로세서 100
레지스터 파일 116
트랩 큐 102
결과 버스 130
기능 유닛 126A
기능 유닛 126B
로드/저장 유닛 126C
실행 코어(들) 124
스케줄러(들) 118
디스패치 유닛 104
리플레이
데이터 캐시 128
프리페치 유닛 108
명령어 캐시 106
메모리 제어기 160
시스템 메모리 200

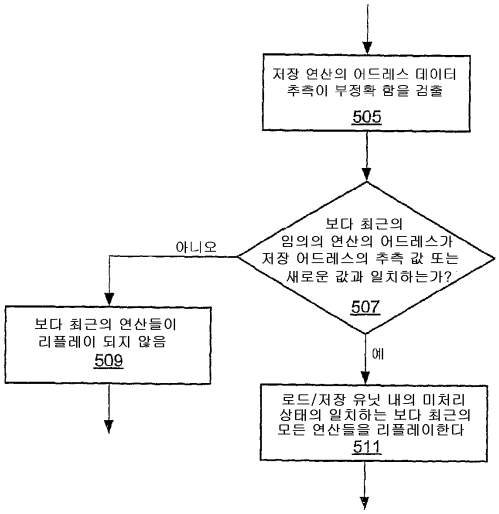
도면2


엔트리 310
Ld/St 313
태그 315
어드레스 317
데이터 319
완성 스토리지 307
데이터 추측 검증 로직 305
로드 저장 유닛 126C
포워딩 트랙킹 309
비교기 로직 303

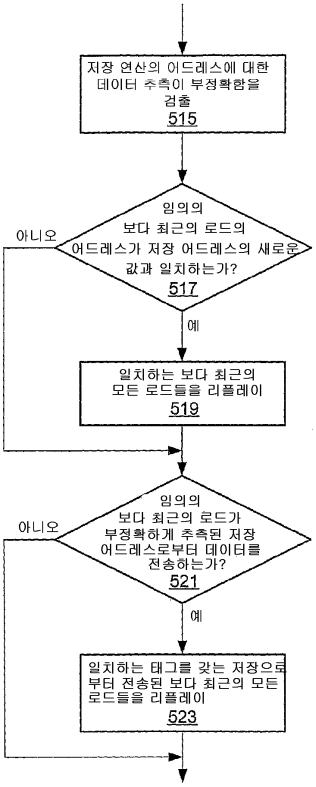
도면3A


연산에 대한 데이터
추측이 부정확함을
검출
501
로드/저장 유닛 내의 미처리 상태의
보다 최근의 모든 연산을 리플레이
503

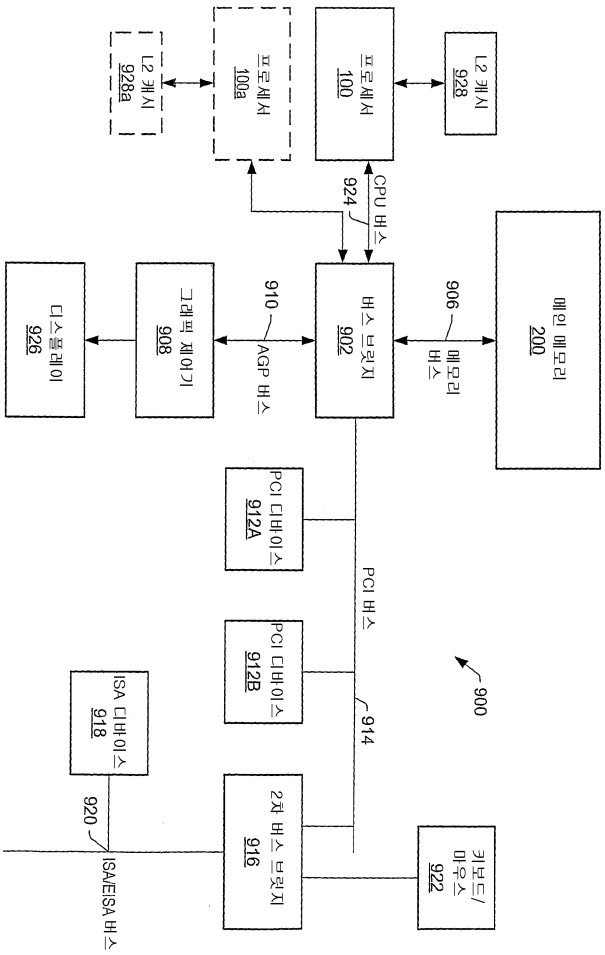
도면3B


저장 연산의 어드레스 데이터
추측이 부정확 함을 검출
505
보다 최근의
임의의 연산의 어드레스가
저장 어드레스의 추측 값 또는
새로운 값과 일치하는가?
507
아니오
예
보다 최근의 연산들이
리플레이 되지 않음
509
로드/저장 유닛 내의 미처리
상태의 일치하는 보다 최근의
모든 연산들을 리플레이한다
511

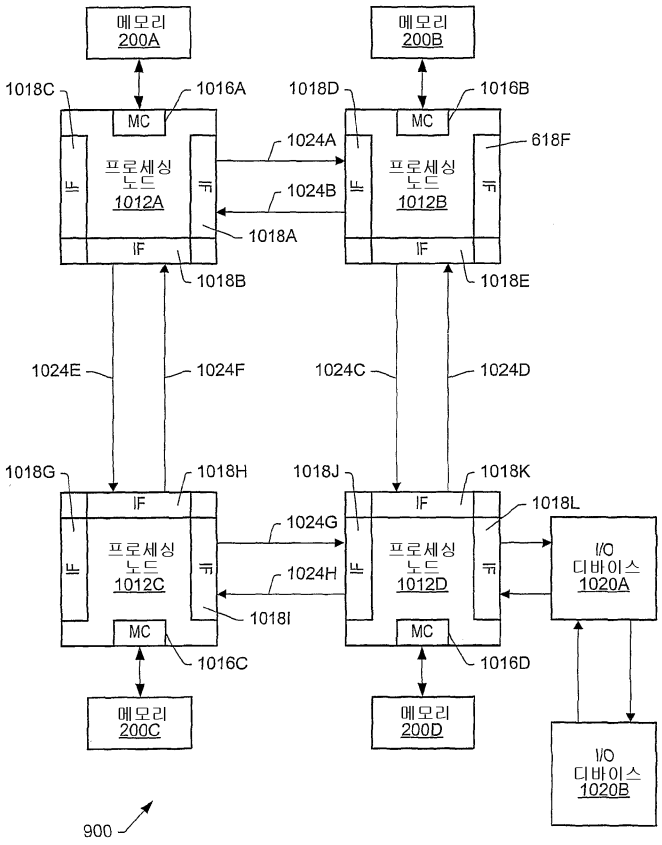
도면3C


저장 연산의 어드레스에 대한 데이터 추측이 부정확함을 검출
515
임의의 보다 최근의 로드의 어드레스가 저장 어드레스의 새로운 값과 일치하는가?
517
아니오
예
일치하는 보다 최근의 모든 로드들을 리플레이
519
임의의 보다 최근의 로드가 부정확하게 추측된 저장 어드레스로부터 데이터를 전송하는가?
521
아니오
예
일치하는 태그를 갖는 저장으로부터 전송된 보다 최근의 모든 로드들을 리플레이
523

도면4


프로세서 100
L2 캐시 928
프로세서 100a
L2 캐시 928a
CPU 버스 924
버스 브릿지 902
메인 메모리 200
메모리 버스 906
그래픽 제어기 908
AGP 버스 910
디스플레이 926
PCI 디바이스 912A
PCI 디바이스 912B
PCI 버스 914
900
2차 버스 브릿지 916
키보드/마우스 922
ISA 디바이스 918
920
ISA/EISA 버스

도면5


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