(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2015-0075067
(43) 공개일자 2015년07월02일

(51) 국제특허분류(Int. Cl.)
G06F 9/06 (2006.01)
(52) CPC특허분류
G06F 9/06 (2013.01)
(21) 출원번호 10-2015-0081361(분할)
(22) 출원일자 2015년06월09일
심사청구일자 2015년06월09일
(62) 원출원 특허 10-2013-0049975
원출원일자 2013년05월03일
심사청구일자 2013년05월03일
(30) 우선권주장
13/464,647 2012년05월04일 미국(US)

(71) 출원인
애플 인크.
미합중국 95014 캘리포니아 쿠퍼티노 인피니트 루프 1
(72) 발명자
마이어, 스테판 지.
미국 95014 캘리포니아주 쿠퍼티노 엠/에스 74-3 피에이 인피니트 루프 1
밀리우스, 존 에이치.
미국 95014 캘리포니아주 쿠퍼티노 엠/에스 23-2 씨에이치피 인피니트 루프 1
(뒷면에 계속)
(74) 대리인
양영준, 백만기

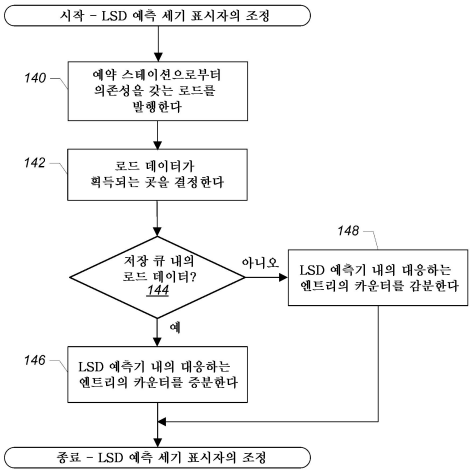
전체 청구항 수 : 총 12 항

(54) 발명의 명칭 **로드-저장 의존성 예측기 내용 관리**

(57) 요 약

비순차 프로세서에서 로드-저장 의존성들을 관리하기 위한 방법들 및 장치들이 설명된다. 로드 저장 의존성 예측기는 의존적이고 비순차적으로 실행하는 것으로 발견된 로드-저장 쌍들에 대한 엔트리들을 저장하기 위한 테이블을 포함할 수 있다. 테이블 내의 각 엔트리는 의존성 예측의 세기를 표시하기 위해 카운터를 포함한다. 카운터가 임계값 위에 있으면, 로드-저장 쌍에 대해 의존성이 시행된다. 카운터가 임계값 아래에 있으면, 로드-저장 쌍에 대해 의존성이 시행되지 않는다. 저장이 디스패치될 때, 테이블은 검색되고, 테이블 내의 임의의 매칭 엔트리들이 암된다. 로드가 디스패치되고, 암된 엔트리에 대해 매치되고, 카운터가 임계값 위에 있으면, 로드는 대응하는 저장이 발행할 때까지 발행을 기다릴 것이다.

대 표 도 - 도7



(72) 발명자

윌리엄스, 3세, 제라드 알.

미국 95014 캘리포니아주 쿠퍼티노 엠/에스 74-3피 에이 인피니트 루프 1

바츠, 수파른

미국 95014 캘리포니아주 쿠퍼티노 엠/에스 23-씨 에이치피 인피니트 루프 1

명 세 서

청구범위

청구항 1

프로세서로서,

재정렬 버퍼;

하나 이상의 예약 스테이션;

엔트리들을 갖는 테이블 - 각각의 엔트리는,

로드 연산에 대한 로드 식별자,

저장 연산에 대한 저장 식별자,

재정렬 버퍼 엔트리 번호,

예측 표시자의 세기, 및

상기 각각의 엔트리의 저장 식별자에 매칭되는 식별자를 갖는 저장 연산이 디스패치되었다는 것을 표시하기 위해 사용되는 암된 비트(armed bit)를 포함하고, 설정될 때 상기 암된 비트는, 상기 각각의 엔트리의 저장 식별자에 매칭되는 식별자를 갖는 상기 저장 연산이 발행된 이후까지 상기 각각의 엔트리의 로드 식별자에 매칭되는 식별자를 갖는 로드 연산이 발행되는 것을 방지함 -;

상기 하나 이상의 예약 스테이션에 연결되는 로드-저장 의존성 예측기

를 포함하고,

상기 로드-저장 의존성 예측기는,

로드 연산이 더 오래된 저장 연산에 의존할 때를 예측하고,

상기 테이블의 엔트리 내에 저장된 식별자에 매칭되는 식별자를 갖는 저장 연산의 디스패치를 검출하는 것에 응답하여, 상기 테이블 내의 상기 엔트리에 대한 암된 비트를 설정하고,

주어진 로드 연산이 설정된 암된 비트를 갖는 상기 테이블의 엔트리에 매칭되는 식별자를 갖는다는 것을 검출하는 것에 응답하여:

상기 주어진 로드 연산이 추론적으로(speculatively) 실행되지 않도록 상기 주어진 로드 연산에 대한 실행의 순서를 시행(enforce)하고,

상기 주어진 로드 연산에 대한 데이터가 제1 위치로부터 검색된다고 결정하는 것에 응답하여, 상기 예측 표시자의 세기를 증분(increment)하며 - 상기 제1 위치는 저장 큐(store queue)를 포함함 -,

상기 주어진 로드 연산에 대한 상기 데이터가 상기 제1 위치와 상이한 제2 위치에서 검색된다고 결정하는 것에 응답하여, 상기 예측 표시자의 세기를 감분(decrement)하도록 - 상기 제2 위치는 캐시임 -

구성되는, 프로세서.

청구항 2

제1항에 있어서,

상기 예측 표시자의 세기는 카운터를 포함하는, 프로세서.

청구항 3

제2항에 있어서,

상기 로드-저장 의존성 예측기는 또한,

주어진 저장 연산이 검출되는 것에 응답하여, 상기 주어진 저장 연산에 대한 매치들(matches)에 대해 상기 테이블을 검색하고,

매치를 찾고 대응하는 카운터가 임계값을 초과한다고 결정하는 것에 응답하여, 상기 주어진 저장 연산의 매칭 엔트리의 상기 암된 비트를 설정하고,

상기 주어진 저장 연산에 대한 재정렬 버퍼 엔트리 번호를 상기 매칭 엔트리에 저장하도록

구성되는 프로세서.

청구항 4

제3항에 있어서,

상기 주어진 로드 연산은 상기 주어진 저장 연산에 대한 상기 재정렬 버퍼 엔트리 번호의 식별을 갖는 예약 스테이션에 저장되는, 프로세서.

청구항 5

제4항에 있어서,

상기 주어진 저장 연산의 발행에 응답하여:

상기 주어진 저장 연산에 대한 상기 재정렬 버퍼 엔트리 번호가 방송(broadcast)되고,

상기 방송된 재정렬 버퍼 엔트리 번호는 상기 주어진 로드 연산을 저장하는 상기 예약 스테이션에 의해 검출되며,

상기 방송된 재정렬 버퍼 엔트리 번호를 검출하는 것에 응답하여, 상기 주어진 로드 연산이 발행되도록 하는, 프로세서.

청구항 6

제1항에 있어서,

상기 예측 표시자의 세기는 에이지-아웃 카운터(age-out counter)를 포함하고,

프로그램가능 기간(programmable period of time)의 만료에 응답하여, 상기 에이지-아웃 카운터는 감분되고, 상기 프로그램가능 기간 동안에는 상기 테이블 내의 엔트리가 액세스되지 않는, 프로세서.

청구항 7

엔트리들을 갖는 테이블을 유지하는 단계 - 각각의 엔트리는,

로드 연산에 대한 로드 식별자,

저장 연산에 대한 저장 식별자,

재정렬 버퍼 엔트리 번호,

예측 표시자의 세기, 및

상기 각각의 엔트리의 저장 식별자에 매칭되는 식별자를 갖는 저장 연산이 디스패치되었다는 것을 표시하기 위해 사용되는 암된 비트를 포함하고, 설정될 때 상기 암된 비트는, 상기 각각의 엔트리의 저장 식별자에 매칭되는 식별자를 갖는 상기 저장 연산이 발행된 이후까지 상기 각각의 엔트리의 로드 식별자에 매칭되는 식별자를 갖는 로드 연산이 발행되는 것을 방지함 -;

로드 연산이 더 오래된 저장 연산에 의존할 때를 예측하는 단계;

상기 테이블의 엔트리 내에 저장된 식별자에 매칭되는 식별자를 갖는 저장 연산의 디스패치를 검출하는 것에 응답하여, 상기 테이블 내의 상기 엔트리에 대한 암된 비트를 설정하는 단계;

주어진 로드 연산이 설정된 암된 비트를 갖는 상기 테이블의 엔트리에 매칭되는 식별자를 갖는다는 것을 검출하는 것에 응답하여, 상기 주어진 로드 연산이 추론적으로 실행되지 않도록 상기 주어진 로드 연산에 대한 실행의

순서를 시행하는 단계;

상기 주어진 로드 연산에 대한 데이터가 제1 위치로부터 검색된다고 결정하는 것에 응답하여, 상기 예측 표시자의 세기를 증분하는 단계 - 상기 제1 위치는 저장 큐를 포함함 -; 및

상기 주어진 로드 연산에 대한 상기 데이터가 상기 제1 위치와 상이한 제2 위치에서 검색된다고 결정하는 것에 응답하여, 상기 예측 표시자의 세기를 감분하는 단계 - 상기 제2 위치는 캐시임 -

를 포함하는 방법.

청구항 8

제7항에 있어서,

상기 주어진 로드 연산을 디스패치하는 단계;

상기 주어진 로드 연산에 대한 식별자에 매칭되는 로드 식별자를 포함하는 엔트리에 대해 상기 테이블을 검색하는 단계; 및

매칭 로드 식별자(matching load identifier)를 포함하는 단일 암된 엔트리(single armed entry)를 찾는 것에 응답하여, 상기 주어진 로드 연산과 주어진 저장 연산 사이의 의존성을 확립하는 단계

를 더 포함하는 방법.

청구항 9

제7항에 있어서,

상기 주어진 로드 연산을 디스패치하는 단계;

상기 주어진 로드 연산에 대한 식별자에 매칭되는 로드 식별자를 포함하는 엔트리에 대해 상기 테이블을 검색하는 단계; 및

매칭 로드 식별자를 포함하는 복수의 암된 엔트리(multiple armed entries)를 찾는 것에 응답하여, 상기 주어진 로드 연산에 대한 멀티매치 의존성(multimatch dependency)을 확립하는 단계

를 더 포함하는 방법.

청구항 10

제8항에 있어서,

상기 주어진 로드 연산을, 상기 주어진 저장 연산에 대한 재정렬 버퍼 엔트리 번호의 식별을 갖는 예약 스테이션에 저장하는 단계;

상기 주어진 저장 연산의 발행에 응답하여, 상기 주어진 저장 연산에 대한 상기 재정렬 버퍼 엔트리 번호를 방송하는 단계; 및

상기 방송된 재정렬 버퍼 엔트리 번호를 검출하는 것에 응답하여, 상기 예약 스테이션으로부터 상기 주어진 로드 연산을 발행하도록 하는 단계

를 더 포함하는 방법.

청구항 11

제9항에 있어서,

상기 주어진 로드 연산에 대한 멀티매치 의존성을 확립하는 것에 응답하여, 각각의 예약 스테이션 내의 가장 오래된 저장 연산을 식별하는 단계

를 더 포함하는 방법.

청구항 12

제7항에 있어서,

상기 예측 표시자의 세기는 에이지-아웃 카운터를 포함하고,

프로그램가능 기간의 만료에 응답하여, 상기 에이지-아웃 카운터는 감분되고, 상기 프로그램가능 기간 동안에는 상기 테이블 내의 엔트리가 액세스되지 않는, 방법.

발명의 설명

기 술 분 야

[0001] 본 발명은 일반적으로 프로세서들에 관한 것이고, 특히, 프로세서들에서 로드-저장 의존성들(load-store dependencies)을 관리하기 위한 방법들 및 메커니즘들에 관한 것이다.

배 경 기 술

[0002] 수퍼스칼라 프로세서들은 클록 사이클당 복수의 명령어들을 발행 및 실행함으로써 그리고 설계와 일관되는 최고 가능한 클록 주파수를 활용함으로써 고성능의 실현을 시도한다. 클록 사이클당 실행되는 명령어들의 수를 증가시키는 하나의 방법은 비순차적 실행(out of order execution)을 수행하는 것이다. 비순차적 실행에서, 명령어들은 프로그램 시퀀스(또는 "프로그램 순서")에 특정된 것과 상이한 순서로 실행될 수 있다.

[0003] 일부 프로세서들은 실현되는 성능 이득을 최대화하기 위한 시도로 비순차적으로 및/또는 추론적으로(speculatively) 명령어들을 스케줄링함에 있어서 가능한 한 공격적(aggressive)일 수 있다. 예를 들어, 로드 메모리 연산들이 더욱 통상적으로 의존적인 명령어들을 갖기 때문에, 더 오래된 저장 메모리 연산(older store memory operation)들 이전에 로드 메모리 연산들을 스케줄링하는 것이 바람직할 수 있다. 그러나, 일부 경우에, 로드 메모리 연산은 더 오래된 저장 메모리 연산에 의존할 수 있다(예를 들어, 저장 메모리 연산은 로드 메모리 연산에 의해 액세스되는 적어도 하나의 바이트를 갱신한다). 이러한 경우, 로드 메모리 연산은 저장 메모리 연산 이전에 실행되는 경우에 올바르지 않게 실행된다. 로드 메모리 연산이 의존적인 더 오래된 저장 메모리 연산 이전에 실행되는 경우, 프로세서는 플러싱(flush) 및 리다이렉트(redirect)될 필요가 있을 수 있고, 이것은 프로세서 성능을 저하할 것이다.

[0004] 프로그램 순서에서 연산이 다른 연산 이전에 있으면 연산은 다른 연산보다 더 오래된 것이다. 프로그램 순서에서 연산이 다른 연산에 후속한다면 연산은 다른 연산보다 더 신생의 것이다. 유사하게, 연산들은 다른 연산들 이전 또는 이후에 있는 것으로서 표시될 수 있거나, 또는 이전의 연산들, 선행 연산들, 후속 연산들 등이라고 할 수 있다. 그러한 참조들은 연산들의 프로그램 순서를 가리킬 수 있다. 또한, "로드 메모리 연산" 또는 "로드 연산"은 메모리 또는 캐시로부터 프로세서로의 데이터의 전송을 가리킬 수 있고, "저장 메모리 연산" 또는 "저장 연산"은 프로세서로부터 메모리 또는 캐시로의 데이터의 전송을 가리킬 수 있다. "로드 연산들" 및 "저장 연산들"은 더욱 간결하게 본원에서 각각 "로드들" 및 "저장들"이라고 할 수 있다.

[0005] 로드들과 저장들 사이의 의존성들은 동적이지만, 이들 이벤트들을 방지하기 위한 메커니즘들은 통상적으로 사실상 정적이다. 따라서, 로드-저장 쌍에 대한 순서를 벗어난 위반(out of order violation)을 방지하기 위한 노력으로, 프로세서는 과잉보상하고 공격적으로 비순차 스케줄링을 하지 않을 가능성이 높다. 이러한 경우, 프로세서는 명령어들을 불필요하게 순서대로 실행시킬 것이다. 의존성이 더 이상 요구되지 않지만 그럼에도 시행된다면, 메모리 레벨 병렬화는 감소하고 프로세서 효율성은 감소할 것이다.

발명의 내용

[0006] 로드-저장 의존성들을 예측하기 위한 시스템들, 장치들, 프로세서들, 및 방법들이 고려된다. 프로세서는 적어도 디스패치 유닛, 로드-저장 의존성 예측기, 및 예약 스테이션을 포함할 수 있다. 더 신생의 로드(younger load)와 의존적인 더 오래된 저장 사이의 정렬 위반(ordering violation)이 검출될 때, 이것은 로드-저장 의존성 예측기에 대한 트레이닝 이벤트(training event)를 구성한다. 로드-저장 쌍이 트레이닝된 후에, 다음에 로드가 디스패치 유닛을 통해 나올 때 예측기는 로드에 의존성을 부가할 수 있다. 이러한 부가된 의존성은, 저장이 스케줄링될 때까지 로드가 예약 스테이션으로부터 스케줄링되어서는 안 된다는 것을 나타낸다.

[0007] 일 실시예에서, 예측기 테이블은 의존적인 것으로 발견된 로드-저장 쌍들을 저장하는 데 활용될 수 있다. 더 신생의 로드가 어드레스 의존성을 공유하는 더 오래된 저장의 앞에 발행될 때, 엔트리는 예측기 테이블에 할당될 수 있고, 일 실시예에서, 엔트리는 저장에 대한 저장 프로그램 카운터(program counter; PC)의 적어도 일부 및 의존적인 로드에 대한 로드 PC 값의 적어도 일부와 연관될 수 있다. 예측기 테이블 내의 각 엔트리는 또한

카운터 필드를 포함할 수 있고, 카운터 필드는 그 특정 로드-저장 쌍에 대한 데이터 의존성 예측의 세기(strength)를 나타낼 수 있다. 카운터 필드는 예측된 의존성들이 오래되거나 더 이상 유효하지 않을 때 그것들을 턴 오프(turn off)되게 한다.

[0008] 카운터 필드의 값은 또한 예측기 테이블에 대한 대체 폴리시(replacement policy)에 영향을 미칠 수 있다. 대체 포인터(replacement pointer)는 끊임없이 예측기의 엔트리들을 스캔하고 낮은 카운터 값들을 갖는 엔트리들을 찾고 있을 수 있다. 일 실시예에서, 예측기 테이블이 액세스될 때마다, 대체 포인터는 전진(advance)할 수 있다. 대체 포인터가 제로의 카운터 값을 갖는 엔트리를 발견하면, 포인터는 이 엔트리에서 중지할 수 있다. 새로운 엔트리가 새로운 의존적인 로드-저장 쌍에 대해 할당되면, 포인터에 의해 표시된 제로와 같은 카운터를 갖는 기존의 엔트리는 새로운 엔트리를 위해 활용될 수 있다.

[0009] 이들 및 다른 특징들 및 이점들은 본원에 제시된 접근법들에 대한 다음의 상세한 설명에 비추어 이 기술분야의 통상의 기술자에게 명백해질 것이다.

도면의 간단한 설명

[0010] 방법들 및 메커니즘들의 상기 및 추가 이점들은 첨부 도면들과 함께 다음의 설명을 참조함으로써 더 잘 이해될 수 있다.

도 1은 집적 회로의 일부분의 일 실시예를 도시한다.

도 2는 프로세서 코어의 일 실시예를 도시하는 블록도이다.

도 3은 맵/디스패치 유닛 및 예약 스테이션의 일 실시예를 도시하는 블록도이다.

도 4는 로드-저장 의존성 예측기 테이블의 일 실시예를 도시한다.

도 5는 로드-저장 의존성 예측기 테이블에서 활용되는 카운터 값들의 표현의 일 실시예를 도시하는 블록도이다.

도 6은 로드 연산을 프로세싱하기 위한 방법의 일 실시예를 도시하는 일반화된 흐름도이다.

도 7은 로드-저장 의존성 예측 세기 표시자를 조정하기 위한 방법의 일 실시예를 도시하는 일반화된 흐름도이다.

도 8은 로드-저장 의존성 예측기 테이블 내의 엔트리들을 대체하기 위한 방법의 일 실시예를 도시하는 일반화된 흐름도이다.

도 9는 시스템의 일 실시예의 블록도이다.

도 10은 컴퓨터 판독 가능한 매체의 일 실시예의 블록도이다.

발명을 실시하기 위한 구체적인 내용

[0011] 다음의 설명에서, 본원에 제시된 방법들 및 메커니즘들의 철저한 이해를 제공하기 위해 다수의 특정 상세들이 제시된다. 그러나, 이 기술분야의 통상의 기술자는 이들 특정 상세들 없이 다양한 실시예들이 실시될 수 있다는 것을 인식해야 한다. 일부 사례들에서, 잘 알려진 구조들, 컴포넌트들, 신호들, 컴퓨터 프로그램 명령어들, 및 기법들은 본원에 설명된 접근법들을 불명료하게 하는 것을 피하기 위해 상세하게 나타내지 않았다. 예시의 간단함 및 명확함을 위해, 도면들에 도시된 요소들은 반드시 비례적으로 그려지지는 않았다는 것을 알 것이다. 예를 들어, 요소들 중 일부의 치수들은 다른 요소들에 비해 과장될 수 있다.

[0012] 이 명세서는 "일 실시예"에 대한 참조들을 포함한다. 상이한 문맥들에서 "일 실시예에서"라는 구절의 출현은 반드시 동일한 실시예를 가리키지는 않는다. 특정 특징들, 구조들, 또는 특성들은 본 개시와 일관되는 임의의 적절한 방식으로 결합될 수 있다. 또한, 이 출원 전체에 걸쳐서 이용되는 바와 같이, 단어 "~할 수 있다(may)"는 의무적인 의미(즉, ~해야 한다(must)는 의미)라기보다는, 허용적인 의미(즉, ~할 가능성이 있다는 의미)로 이용된다. 유사하게, 단어들 "포함한다(include)", "포함하는(including)", 및 "포함한다(includes)"는 포함하지만 그것으로 한정되지 않음을 의미한다.

[0013] 전문용어. 다음 단락들은 본 개시(첨부된 청구항들을 포함함)에서 발견되는 용어들에 대한 정의들 및/또는 문맥을 제공한다:

[0014] "포함하는(Comprising)". 이 용어는 오픈 엔드형(open-ended)이다. 첨부된 청구항들에서 이용되는 바와 같이,

이 용어는 부가적인 구조 또는 단계들을 배제하지 않는다. "로드-저장 의존성 예측기를 포함하는 프로세서..."라고 기재하는 청구항을 고려한다. 이러한 청구항은 프로세서가 부가적인 컴포넌트들(예를 들어, 캐시, 인출 유닛, 실행 유닛)을 포함하는 것을 배제하지 않는다.

[0015] "~하도록 구성되는(Configured To)". 다양한 유닛들, 회로들, 또는 다른 컴포넌트들은 태스크 또는 태스크들을 수행"하도록 구성되는" 것으로서 설명 또는 청구될 수 있다. 이러한 문맥들에서, "~하도록 구성되는"은 유닛들/회로들/컴포넌트들이 동작 동안 태스크 또는 태스크들을 수행하는 구조(예를 들어, 회로)를 포함함을 표시함으로써 구조를 함축하는 데 이용된다. 이와 같이, 유닛/회로/컴포넌트는 특정된 유닛/회로/컴포넌트가 현재 동작하지 않을 때에도(예를 들어, 온(on) 상태가 아닐 때에도) 태스크를 수행하도록 구성되어 있다고 말할 수 있다. "~하도록 구성되는" 언어와 이용되는 유닛들/회로들/컴포넌트들은 하드웨어 - 예를 들어, 회로들, 동작을 구현하기 위해 실행 가능한 프로그램 명령어들을 저장하는 메모리, 등을 포함한다. 유닛/회로/컴포넌트가 하나 이상의 태스크를 수행"하도록 구성"된다고 기재하는 것은 그 유닛/회로/컴포넌트에 대해, 35 U.S.C. §112, 6절을 적용하지 않도록 명백하게 의도된다. 부가적으로, "~하도록 구성되는"은 문제가 되고 있는 태스크(들)를 수행할 수 있는 방식으로 동작하기 위해 소프트웨어 및/또는 펌웨어(예를 들어, FPGA 또는 소프트웨어를 실행하는 범용 프로세서)에 의해 조작되는 일반적인 구조(예를 들어, 일반적인 회로)를 포함할 수 있다. "~하도록 구성되는"은 또한 하나 이상의 태스크들을 구현 또는 수행하도록 적응되는 장치들(예를 들어, 집적 회로들)을 제조하기 위해 제조 프로세스(예를 들어, 반도체 제조 설비)를 적응시키는 것을 포함할 수 있다.

[0016] "~에 기초한(Based On)". 본원에 이용되는 바와 같이, 이 용어는 결정에 영향을 미치는 하나 이상의 인자들을 설명하는 데 이용된다. 이 용어는 결정에 영향을 미칠 수 있는 부가적인 인자들을 배제하지 않는다. 즉, 결정은 단지 그의 인자들에 기초하거나 그의 인자들에 적어도 부분적으로 기초할 수 있다. "B에 기초하여 A를 결정한다"라는 구절을 고려한다. B가 A의 결정에 영향을 미치는 인자일 수 있고, 그러한 구절은 A의 결정이 C에도 기초하는 것을 배제하지 않는다. 다른 경우들에서, A는 B에만 기초하여 결정될 수 있다.

[0017] 이제 도 1을 참조하면, 직접 회로(IC)의 일부분의 일 실시예를 예시하는 블록도가 도시된다. 예시된 실시예에서, IC(10)는 프로세서 콤플렉스(processor complex)(12), 메모리 컨트롤러(22), 및 메모리 물리적 인터페이스 회로들(PHY)(24 및 26)을 포함한다. IC(10)는 또한 도 1에 도시되지 않은 다른 많은 컴포넌트를 포함할 수 있다는 것에 주목한다. 다양한 실시예들에서, IC(10)는 또한 시스템 온 칩(system on chip; SoC), ASIC(application specific integrated circuit), 또는 장치라고도 할 수 있다.

[0018] 프로세서 콤플렉스(12)는 중앙 처리 유닛들(CPU)(14 및 16), 레벨 2(L2) 캐시(18), 및 버스 인터페이스 유닛(BIU)(20)을 포함할 수 있다. 다른 실시예들에서, 프로세서 콤플렉스(12)는 다른 개수의 CPU를 포함할 수 있다. CPU들(14 및 16)은 또한 프로세서들 또는 코어들이라고도 할 수 있다. CPU들(14 및 16)은 L2 캐시(18)에 결합될 수 있다. L2 캐시(18)는 BIU(20)에 결합될 수 있고, BIU(20)는 메모리 컨트롤러(22)에 결합될 수 있다. 다른 실시예들은 캐시의 부가적인 레벨들(예를 들어, 레벨 3(L3) 캐시)을 포함할 수 있다. 프로세서 콤플렉스(12)는 도 1에 도시되지 않은 다른 컴포넌트들을 포함할 수 있다는 것에 주목한다.

[0019] CPU들(14 및 16)은 명령어 세트 아키텍처에 정의된 명령어들을 실행하기 위한 회로를 포함할 수 있다. 구체적으로, 명령어들을 포함하는 하나 이상의 프로그램들은 CPU들(14 및 16)에 의해 실행될 수 있다. 임의의 명령어 세트 아키텍처는 다양한 실시예에서 구현될 수 있다. 예를 들어, 일 실시예에서, PowerPC™ 명령어 세트 아키텍처가 구현될 수 있다. 다른 예시적인 명령어 세트 아키텍처들은 ARM™ 명령어 세트, MIPS™ 명령어 세트, SPARC™ 명령어 세트, x86 명령어 세트(IA-32라고도 함), IA-64 명령어 세트 등을 포함할 수 있다.

[0020] 다양한 실시예에서, CPU들(14 및 16)은 명령어들을 비순차적으로 실행할 수 있고, 이것은 일부 경우에 정렬 위반들을 일으킬 수 있다. 예를 들어, 로드 및 저장 명령어들의 경우에, 정렬 위반들은 더 신생의 로드들이 중첩되는 물리적 어드레스들을 갖는 더 오래된 저장들 이전에 실행될 때 일어날 수 있다. 이러한 타입의 정렬 위반의 반복을 회피 또는 방지하기 위하여, 다양한 기법을 활용해서 더 신생의 로드가 그것이 의존적인 더 오래된 저장 이전에 실행되지 못하게 할 수 있다. 일 실시예에서, CPU들(14 및 16) 각각은 의존적인 것으로 예측 또는 기대되고 또한 비순차적으로 실행하는 경향이 있을 수 있는 로드-저장 쌍들을 추적하기 위한 로드-저장 의존성 예측기를 포함할 수 있다. 일 실시예에서, 의존적인 로드-저장 쌍들은 테이블에 기록될 수 있다.

[0021] 때때로, 예측기는 로드-저장 쌍에 대해 트레이닝할 수 있지만, 의존성은 예외 경우일 수 있다. 이것은 로드 및 저장 명령어들 사이의 의존성이 어드레스에 기초할 수 있고, 로드 및 저장 명령어의 어드레스들이 시간이 흐르면 변할 수 있기 때문에 발생할 수 있다. 다시 말해, 로드-저장 의존성들은 동적일 수 있다. 테이블 내의 엔트리들 중 일부는 일정 기간 후에 정확하지 않을 수 있고, 부정확한 엔트리들에 대해 의존성들을 시행함으로써,

프로세서는 임의의 이득 없이 불필요하게 로드 연산들을 지연시킬 수 있다.

[0022] 오래된 엔트리들이 테이블에 누적되는 것을 방지하기 위하여, 그리고 오래된 엔트리들에 대응하는 로드-저장 쌍들에 대해 의존성들이 시행되는 것을 방지하기 위하여, 테이블의 각 엔트리는 또한 의존성 예측의 세기를 나타내는 표시자를 포함할 수 있다. 표시자는 주어진 로드-저장 쌍에 대해 의존성이 시행되는지를 결정할 수 있다. 표시자는 또한 테이블에 새로운 엔트리가 할당될 때 낮은 표시자 값들을 갖는 엔트리들이 대체될 수 있도록, 테이블 엔트리들에 대한 대체 폴리시에 영향을 미칠 수 있다.

[0023] CPU들(14 및 16) 각각은 또한 레벨 1(L1) 캐시(도시되지 않음)를 포함할 수 있고, 각각의 L1 캐시는 L2 캐시(18)에 결합될 수 있다. 일 실시예에서, L2 캐시(18)는 CPU들(14 및 16)에 의해 낮은 레이턴시 액세스를 위해 명령어들 및 데이터를 캐싱하도록 구성될 수 있다. L2 캐시(18)는 임의의 용량 및 구성(예를 들어, 직접 맵핑, 세트 결합)을 포함할 수 있다. L2 캐시(18)는 BIU(20)를 통해 메모리 컨트롤러(22)에 결합될 수 있다. BIU(20)는 다양한 다른 장치들 및 블록들에 CPU들(14 및 16) 및 L2 캐시(18)를 결합하기 위해 다양한 다른 논리 구조들을 포함할 수 있다.

[0024] 메모리 컨트롤러(22)는 임의의 개수의 메모리 포트들을 포함할 수 있고, 메모리에 인터페이스하도록 구성되는 회로를 포함할 수 있다. 예를 들어, 메모리 컨트롤러(22)는 SDRAM(synchronous DRAM) 등의 DRAM(dynamic random access memory), DDR(double data rate) SDRAM, DDR2 SDRAM, RDRAM(Rambus DRAM) 등에 인터페이스하도록 구성될 수 있다. 메모리 컨트롤러(22)는 또한 메모리 물리적 인터페이스 회로들(PHYs)(24 및 26)에 결합될 수 있다. 메모리 PHY들(24 및 26)은 메모리 컨트롤러(22)에 결합될 수 있는 임의의 개수의 메모리 PHY들을 대표한다. 메모리 PHY들(24 및 26)은 메모리 장치들(도시되지 않음)에 인터페이스하도록 구성될 수 있다.

[0025] 다른 실시예들은 도 1에 도시된 컴포넌트들 및/또는 다른 컴포넌트들의 서브세트들 또는 수퍼세트들을 포함하는, 컴포넌트들의 다른 결합들을 포함할 수 있다는 것에 주목한다. 주어진 컴포넌트의 하나의 사례가 도 1에 도시될 수 있지만, 다른 실시예들은 주어진 컴포넌트의 2개 이상의 사례를 포함할 수 있다. 유사하게, 본 상세한 설명 전체에 걸쳐서, 하나의 사례만이 도시되더라도 주어진 컴포넌트의 2개 이상의 사례가 포함될 수 있고, 및/또는 복수의 사례가 도시되더라도 하나의 사례만을 포함하는 실시예들이 이용될 수 있다.

[0026] 이제 도 2를 보면, 프로세서 코어의 일 실시예가 도시된다. 코어(30)는 프로세서 코어의 일 예이고, 코어(30)는 도 1의 프로세서 콤플렉스(12)와 같은 프로세서 콤플렉스 내에서 활용될 수 있다. 일 실시예에서, 도 1의 CPU들(14 및 16) 각각은 코어(30)의 컴포넌트들 및 기능을 포함할 수 있다. 코어(30)는 인출 및 디코드(fetch and decode; FED) 유닛(32), 맵 및 디스패치(map and dispatch) 유닛(36), 메모리 관리 유닛(memory management unit; MMU)(40), 코어 인터페이스 유닛(CIF)(42), 실행 유닛들(44), 및 로드-저장 유닛(load-store unit; LSU)(46)을 포함할 수 있다. 코어(30)는 도 2에 도시되지 않은 다른 컴포넌트들 및 인터페이스들을 포함할 수 있다는 것에 주목한다.

[0027] FED 유닛(32)은 메모리로부터 명령어들을 판독하여 그것들을 레벨 1(L1) 명령어 캐시(34)에 배치하도록 구성되는 회로를 포함할 수 있다. L1 명령어 캐시(34)는 코어(30)에 의해 실행될 명령어들을 저장하기 위한 캐시 메모리일 수 있다. L1 명령어 캐시(34)는 임의의 용량 및 구성(예를 들어, 직접 맵핑, 세트 결합, 완전 결합(fully associative) 등)을 가질 수 있다. 또한, L1 명령어 캐시(34)는 임의의 캐시 라인 사이즈를 가질 수 있다. FED 유닛(32)은 또한 분기 명령어들을 예측하고 예측된 경로로 인출하도록 구성되는 분기 예측 하드웨어를 포함할 수 있다. FED 유닛(32)은 또한 (예를 들어, 예측 실패(misprediction), 예외, 인터럽트, 플러시 등을 통해) 리다이렉트될 수 있다.

[0028] FED 유닛(32)은 또한 명령어들을 명령어 연산들(ops)로 디코딩하도록 구성될 수 있다. 일반적으로, 명령어 연산은 실행 유닛들(44) 및 LSU(46)에 포함된 하드웨어가 실행할 수 있는 연산일 수 있다. 각 명령어는, 실행될 때, 명령어 세트 아키텍처에 따라 그 명령어에 대해 정의된 연산들의 수행을 야기하는 하나 이상의 명령어 연산들로 변환할 수 있다. FED 유닛(32)은 복수의 명령어들을 병렬로 디코딩하도록 구성될 수 있다.

[0029] 일부 실시예들에서, 각 명령어는 단일 명령어 연산으로 디코딩할 수 있다. FED 유닛(32)은 명령어의 타입, 소스 오퍼랜드들 등을 식별하도록 구성될 수 있고, 각각의 디코딩된 명령어 연산은 디코드 정보의 일부와 함께 명령어를 포함할 수 있다. 각 명령어가 단일 op로 변환하는 다른 실시예들에서, 각각의 op는 단순히 대응하는 명령어 또는 그의 일부(예를 들어, 명령어의 오피코드 필드 또는 필드들)일 수 있다. 일부 실시예들에서, FED 유닛(32)은 명령어들에 대한 op들을 발생하기 위한 회로 및/또는 마이크로코드의 임의의 결합을 포함할 수 있다. 예를 들어, 비교적 단순한 op 발생들(예를 들어, 명령어당 1개 또는 2개의 op)은 하드웨어로 핸들링될 수 있지

만, 더욱 광범위한 op 발생들(예를 들어, 명령어에 대한 3개보다 많은 op)은 마이크로코드로 핸들링될 수 있다. 다른 실시예들에서, FED 유닛(32) 내에 포함된 기능은 인출 유닛, 디코드 유닛, 및/또는 다른 유닛들과 같은 2 이상의 별개의 유닛들로 분할될 수 있다.

[0030] 디코딩된 op들은 맵/디스패치 유닛(36)에 제공될 수 있다. 맵/디스패치 유닛(36)은 op들 및 아키텍처 레지스터들을 코어(30)의 물리적 레지스터들에 맵핑하도록 구성될 수 있다. 맵/디스패치 유닛(36)은 op들로부터의 소스 레지스터 어드레스들을 재명명된 소스 레지스터들을 식별하는 소스 오퍼랜드 번호들에 맵핑하기 위해 레지스터 재명명을 구현할 수 있다. 맵/디스패치 유닛(36)은 또한 실행 유닛들(44) 및 LSU(46) 내의 예약 스테이션들에 op들을 디스패치하도록 구성될 수 있다. 맵/디스패치 유닛(36)은 로드-저장 의존성(load-store dependency; LSD) 예측기(37) 및 재정렬 버퍼(ROB)(38)를 포함할 수 있다. 디스패치되기 전에, op들은 ROB(38)에 기록될 수 있다. ROB(38)는 op들이 순서대로 커밋(commit)될 수 있을 때까지 op들을 유지하도록 구성될 수 있다. 각각의 op는 ROB(38) 내의 특정 엔트리에 대응하는 ROB 인덱스(RNUM)를 할당받을 수 있다. RNUM들은 코어(30)에서 동작중인(in flight) 연산들을 추적하는 데 이용될 수 있다. 맵/디스패치 유닛(36)은 또한 도 2에 도시되지 않은 다른 컴포넌트들(예를 들어, 맵퍼 어레이, 디스패치 유닛, 디스패치 버퍼)을 포함할 수 있다. 또한, 다른 실시예들에서, 맵/디스패치 유닛(36)은 내에 포함된 기능은 맵 유닛, 디스패치 유닛, 및/또는 다른 유닛들과 같은 2 이상의 별개의 유닛들로 분할될 수 있다.

[0031] LSD 예측기(37)는 비순차적으로 발행될 가능성이 있는 의존적인 로드-저장 명령어 쌍들을 트레이닝 및 예측하도록 구성될 수 있다. LSD 예측기(37)는 트레이닝한 로드-저장 쌍들에 대한 엔트리들을 갖는 테이블을 포함할 수 있고, 각 엔트리는 로드 및 저장 명령어들을 식별하는 정보와 예측의 세기를 포함할 수 있다. 일 실시예에서, 트레이닝 이벤트는 중첩되는 물리적 어드레스들을 갖는 더 오래된 저장 전에 더 신생의 로드의 실행에 의해 트리거되는 정렬 위반일 수 있다. 일 실시예에서, 테이블은 256-엔트리 완전 결합 구조일 수 있다. 다른 실시예들에서, 테이블은 다른 개수의 엔트리를 가질 수 있다. 다양한 실시예에서, 테이블은 테이블의 다양한 필드에 대한 내용 지정 메모리(content-addressable memory; CAM)일 수 있다.

[0032] 의존적인 로드 및 저장 연산들 사이에 정렬 위반이 존재할 때, 코어(30)는 리다이렉트되고 재동기화될 수 있다. 리다이렉트의 결과로서 다양한 교정 액션들이 취해질 수 있다. 이러한 포인트에서, 재동기화를 야기한 특정 로드-저장 쌍에 대해 트레이닝이 수행될 수 있다. 이러한 특정 쌍에 대한 엔트리가 LSD 예측기(37)에 할당될 수 있고, 예측의 세기는 고레벨로 설정될 수 있다. 그 다음에, 코어(30)의 파이프라인을 통한 다음 단계(pass)에서, 로드-저장 쌍으로부터의 저장이 유닛(36)으로부터 디스패치될 때, LSD 예측기(37)는 저장에 대해 검색될 수 있다. 매칭 엔트리(matching entry)가 발견되고 암(arm)될 수 있다. 트레이닝된 로드-저장 쌍으로부터의 로드가 유닛(36)으로부터 디스패치될 때, LSD 예측기(37)의 검색은 로드에 대해 수행될 수 있고, 로드는 암된 엔트리(armed entry)에서 매칭할 것이다. 그 다음, 로드는 의존성을 갖고 예약 스테이션에 디스패치됨으로써, 로드가 예약 스테이션으로부터 발행되기 전에 저장을 기다리게 한다.

[0033] LSD 예측기(37)는, 엔트리를 암한 저장(a store that armed an entry)이 저장이 발행되기 전에 명령어 파이프라인으로부터 플러시되는 경우에 테이블을 클린 업(clean up)하도록 구성될 수 있다. 예를 들어, 결함이 존재할 때와 같이, LSD 예측기(37)의 암된 엔트리를 디스암(disarm)할 필요가 있을 때의 시나리오가 존재할 수 있다. 로드 연산은 의존적이어서 그 이후에 플러싱되는 저장 연산을 기다릴 수 있고, 이것은 결국 코어(30)가 교착 상태에 처하게 할 수 있다. 이 경우, 저장 연산이 코어(30)로부터 플러싱될 때, LSD 예측기(37)의 테이블은 플러싱된 저장에 대응하는 임의의 암된 엔트리들에 대해 검색될 수 있다. 플러싱된 저장에 대해 발견된 임의의 매칭 엔트리들이 디스암될 수 있다. 일 실시예에서, LSD 예측기(37)의 각 엔트리는 로드-저장 쌍의 특정 저장을 식별하기 위한 저장 RNUM을 포함할 수 있다.

[0034] 실행 유닛들(44)은 임의의 개수 및 타입의 실행 유닛들(예를 들어, 정수, 부동 소수점, 벡터)을 포함할 수 있다. 실행 유닛들(44) 각각은 또한 하나 이상의 예약 스테이션(도시되지 않음)을 포함할 수 있다. CIF(42)는 LSU(46), FED 유닛(32), MMU(40), 및 L2 캐시(도시되지 않음)에 결합될 수 있다. CIF(42)는 코어(30)와 L2 캐시 사이의 인터페이스를 관리하도록 구성될 수 있다. MMU(40)는 어드레스 변환 및 메모리 관리 기능들을 수행하도록 구성될 수 있다.

[0035] LSU(46)는 L1 데이터 캐시(48), 예약 스테이션들(50 및 52), 저장 큐(54), 및 로드 큐(56)를 포함할 수 있다. 로드 및 저장 연산들은 맵/디스패치 유닛(36)으로부터 예약 스테이션들(50 및 52)로 디스패치될 수 있다. 다른 실시예들은 다른 개수의 예약 스테이션들을 포함할 수 있다. 연산들은 예약 스테이션들(50 및 52)로부터 비순차적으로 발행할 수 있다. 저장 큐(54)는 저장 연산들에 대응하는 데이터를 저장할 수 있고, 로드 큐(56)는 로

드 연산들과 연관된 데이터를 저장할 수 있다. LSU(46)는 또한 CIF(42)를 통해 L2 캐시에 결합될 수 있다. LSU(46)는 또한 도 2에 도시되지 않은 다른 컴포넌트들(예를 들어, 레지스터 파일, 사전 인출 유닛(prefetch unit), 변환 색인 버퍼)을 포함할 수 있다.

[0036] 로드-저장 순서 위반은 더 오래된 저장이 발행되는 때에 LSU(46)에 의해 검출될 수 있다. 일 실시예에서, 더 오래된 저장의 저장 어드레스는 로드 큐(56) 내의 모든 더 신생의 로드들에 대해 비교될 수 있다. 매치가 검출되면, 로드 연산은 이미 부정확한 데이터로 완료되었을 수 있다. 이것은 로드 및 저장 연산들의 RNUM들을 이용하여 맵/디스패치 유닛(36)으로 돌아가는 리다이렉트를 시그널링함으로써 장래에 정정될 수 있다. 맵/디스패치 유닛(36)은 코어(30) 파이프라인으로부터 명령어들을 플러싱하고 로드의 명령어 어드레스로 코어(30)의 전단을 리다이렉트할 수 있고, 로드 명령어는 재인출될 수 있다. 장래의 리다이렉트들을 방지하기 위하여, 맵/디스패치 유닛(36)은 LSD 예측기(37)에서 저장들에 대한 로드들의 의존성들을 예측하고 기록하여 예약 스테이션들(50 및 52)에 예측된 의존성들을 통신할 수 있다.

[0037] 통상적인 경우에, 저장이 디스패치될 때, 저장은 LSD 예측기(37)를 검색하고 나서, 저장에 대해 매치가 발견되면, 테이블 내의 매칭 엔트리는 암(즉, 활성화)될 수 있고, 저장 RNUM은 엔트리에 기록될 수 있다. 후속하여, 로드는 디스패치될 수 있고, 테이블 내의 로드들에 걸친 검색이 수행될 수 있다. 일 실시예에서, LSD 예측기(37)를 검색하는 데 이용된 식별 값들은 로드 및 저장 PC 값들의 적어도 일부일 수 있다. 다른 실시예에서, 검색을 위해 이용되고 엔트리들에 저장되는 식별 값들은 PC 값들의 적어도 일부, 아키텍처 레지스터 값들의 적어도 일부, 및/또는 마이크로오피(micro-op) 값들의 적어도 일부로부터 도출되는 해쉬 값들(hashed values)일 수 있다. 활용될 수 있는 식별자들의 다른 가능성들이 가능하고 고려된다.

[0038] 다양한 실시예에서, 로드는 LSD 예측기(37) 내의 임의의 개수의 엔트리들에 대해 매칭할 수 있다. 일 실시예에서, 매치가 일어나기 위해서 엔트리는 암될 필요가 있다. 로드가 하나의 암된 엔트리에 대해 매칭하면, 저장 RNUM에 대한 의존성은 암된 저장 RNUM을 로드에 링크함으로써 생성될 수 있다. 로드는 그 특정 저장 RNUM이 예약 스테이션으로부터 발행하는 것을 기다리는 것으로서 마킹될 수 있다. 예약 스테이션들에서, 로드들에 대한 의존성 필드가 존재할 수 있고, 로드는 예약 스테이션들 중 하나(50 또는 52)로부터 발행할 주어진 저장에 대해 의존적인 것으로서 마킹될 수 있다. 그래서 이러한 경우에, 로드는 특정 저장 RNUM을 기다리는 것으로서 마킹될 수 있고, 로드는 특정 저장이 발행된 후 하나의 사이클 후에 발행될 수 있다.

[0039] 로드가 복수의 암된 저장 엔트리들에 대해 매칭하면, 이것은 멀티매치 경우(multimatch case)라고 할 수 있다. 이러한 경우, 로드는 발행하기 전에 모든 더 오래된 저장들이 발행될 때까지 기다릴 수 있다. 예를 들어, 일 실시예에서, 로드는 로드가 발행하기 전에 모든 더 오래된 저장들이 발행하기를 기다릴 수 있도록 비트가 설정될 수 있다. 이것은 로드의 앞에 예약 스테이션들(50 및 52)로부터 모든 더 오래된 저장들이 발행하도록 강제할 것이다. 일 실시예에서, 예약 스테이션들(50 및 52) 각각은 그것이 포함하는 가장 오래된 저장을 이용 가능하게 만들 수 있다. 일단 로드가 그의 저장들 둘다보다 더 오래되면, 로드는 발행할 수 있다.

[0040] 각각의 예약 스테이션(50 및 52)은 유효한 임의의 연산들을 발행하도록 구성되는 픽커(picker)를 포함할 수 있다. 저장이 유효해질 때, 그리고 그것이 선택되어 발행될 때, 태그가 브로드캐스트될 수 있고, 그 다음에 이 저장에 의존적인 로드가 그 태그에 대해 매칭할 것이다. 이것은 예약 스테이션으로부터 발행될 수 있는 것으로서 로드를 마킹할 것이다. 다시 말해, 저장은 로드에 의해 이용되는 태그를 생성한다. 일 실시예에서, 태그는 저장의 RNUM일 수 있다. 일 실시예에서, RNUM은 9-비트 값일 수 있고, 다른 실시예들에서는, RNUM의 사이즈는 달라질 수 있다. 의존성을 갖는 로드는 예약 스테이션에서 로드와 저장된 여분의 소스를 가질 수 있고, 그 여분의 소스는 LSD 예측기(37) 내의 동일한 엔트리로부터의 저장의 RNUM일 수 있다.

[0041] 로드가 LSD 예측기(37) 내의 엔트리에 대해 매칭되고 엔트리가 암될 때, 이것은 로드가 기다릴 필요가 있는 유효 저장이 존재함을 의미한다. 엔트리는 또한 예측의 세기에 대한 표시자를 포함할 수 있다. 일 실시예에서, 표시자는 카운터일 수 있고, 카운터의 값이 임계값 위에 있으면, 엔트리는 강하고 가능성이 큰 예측으로 고려될 수 있고, 로드-저장 의존성이 확립될 수 있다. 임계값은 실시예마다 다를 수 있다. 로드가 암된 엔트리에 대해 매칭하고 표시자가 약해서, 예측을 이용하지 않음을 표시하면, 의존성은 로드에 대해 확립되지 않을 수 있다. 로드-저장 의존성이 확립되면, 로드는 저장의 RNUM을 픽업할 수 있어서, RNUM은 엔트리로부터 판독되고 로드가 디스패치될 때 로드와 함께 예약 스테이션에 전달된다. 로드는 또한 예약 스테이션에서 의존성을 갖는 것으로서 마킹될 수 있다.

[0042] 일 실시예에서, 예약 스테이션으로부터 발행하는 저장은 저장이 유효 생성자인 것으로서 마킹되는 경우에만 태그가 브로드캐스트되게 할 것이다. 저장이 LSD 예측기(37)를 검색하고 매치가 발견되지 않으면, 저장은 유효

생성자로서 설정되지 않을 것이다. 저장이 LSD 예측기(37)에서 유효 엔트리를 발견하고, 예측 세기 표시자가 로드-저장 쌍 의존성 예측이 임계값 위에 있다(즉, 예측이 턴 온되어 있다)고 나타내면, 엔트리는 암될 수 있다. 일 실시예에서, 예측 세기 표시자가 임계값 아래에 있다면, 저장이 그 저장 엔트리와 매칭하더라도, 저장은 엔트리를 암하지 않을 것이다. 일부 실시예들에서, 엔트리는 예측 세기 표시자의 값에 상관없이 저장이 매치를 발견할 때 암될 수 있다. 저장은 복수의 엔트리들에 대해 매칭할 수 있고, 복수의 엔트리들은 단일 저장에 대해 암될 수 있다.

[0043] 로드가 LSD 예측기(37)의 암된 엔트리에 대해 매칭할 때, 로드는 의존적인 것으로서 마킹되고, 로드는 대응하는 저장이 예약 스테이션으로부터 발행할 때까지 예약 스테이션으로부터 발행하는 것을 기다릴 수 있다. 그 다음, 확립된 의존성을 갖는 로드가 발행한 후에, 로드가 그의 데이터를 수신하는 곳이 결정될 수 있다. 로드가 그의 데이터를 수신하는 곳에 따라, LSD 예측기(37)의 대응하는 엔트리 내의 예측 세기 표시자가 증가하거나, 감소하거나, 또는 동일하게 남을 수 있다.

[0044] 예를 들어, 로드 데이터가 저장 큐(54)로부터 전달되었다면, LSD 예측기(37)로부터의 예측은 양호한 것으로서 고려될 수 있다. 이 경우, 저장으로부터의 데이터는 아직 캐시(48)에 저장되지 않았고, 그래서 로드가 저장을 기다린 것이 이득이 있었다. 이러한 로드 연산에 대한 로드 데이터가 여전히 저장 큐(54)에 있다면, 이것은 로드와 저장 사이에 트루 의존성(true dependency)이 실제로 존재함을 나타낼 수 있다. 다시 말해, 데이터는 의존적인 로드에 대해 저장 큐(54)로부터 전달될 필요가 있었다.

[0045] 로드 데이터에 대한 저장 큐(54)에 미스가 존재하면, 의존성은 더 이상 유효하지 않을 수 있다. 사전 의존성이 존재하였지만, 그 다음에 로드 또는 저장의 어드레스가 변경되고, 로드 및 저장은 더 이상 충돌하지 않는 것이 가능하다. 이러한 경우, 저장 데이터가 캐시(48)로부터 검색되면, 데이터는 장시간 거기에 저장되었을 수 있다. 따라서, 저장 데이터가 저장 큐(54)로부터 또는 캐시(48)로부터 전달되었는지를 결정하는 것은, 예측이 정확하였는지 여부를 나타낼 수 있다. 또한, LSD 예측기(37)의 매칭 엔트리에 저장된 예측 세기 표시자는 이러한 결정에 기초하여 갱신될 수 있다. 예측이 정확하였고, 그래서 로드 데이터가 저장 큐(54)로부터 전달되면, 예측 세기 표시자는 증가할 수 있다. 로드 데이터가 캐시(48)로부터 나오면, 예측 세기 표시자는 감소할 수 있다. 다른 실시예들에서, 의존성 예측이 정확하였는지를 결정하기 위한 다른 기법들이 활용될 수 있다.

[0046] 도 2에 예시된 기능의 분배는 프로세서 코어를 위해 활용될 수 있는 오직 가능한 마이크로아키텍처가 아니라는 것을 이해해야 한다. 다른 프로세서 코어들은 다른 컴포넌트들을 포함할 수 있고, 도시된 컴포넌트들 중 하나 이상을 생략할 수 있고, 및/또는 컴포넌트들 사이에 기능의 상이한 배열을 포함할 수 있다.

[0047] 이제 도 3을 참조하면, 맵/디스패치 유닛 및 예약 스테이션의 일 실시예의 블록도가 도시된다. 일 실시예에서, 맵/디스패치 유닛(60)은 레지스터 맵퍼(62), 재정렬 버퍼(ROB)(64), 로드 저장 의존성(LSD) 예측기(66), 및 디스패치 유닛(68)을 포함할 수 있다. 레지스터 맵퍼(62) 및 LSD 예측기(66)는 디코드 유닛(도시되지 않음)으로부터 op들을 수신하도록 결합될 수 있다. LSD 예측기(66)는 디코드 유닛으로부터 PC들을 수신하도록 결합되고, 로드-저장 유닛(도시되지 않음)으로부터 "리다이렉트(Redirect)" 및 "카운트 갱신(Count Update)" 신호들을 수신하도록 결합된다. LSD 예측기(66)는 또한 새로운 엔트리가 할당될 때 폐기될 수 있는 엔트리들에 대해 LSD 예측기(66)를 검색하는 대체 포인터에 결합된다.

[0048] 레지스터 맵퍼(62)는 아키텍처 레지스터들을 물리적 레지스터들에 맵핑하고, 디스패치 유닛(68)에 op들 및 물리적 레지스터 어드레스들을 제공하도록 구성될 수 있다. 디스패치 유닛(68)은 op들을 예약 스테이션들(70A-N)에 디스패치하도록 구성될 수 있다. 디스패치 유닛(68)은 예약 스테이션들(70A-N)에 예약 스테이션 엔트리들의 프리 리스트(free list)를 유지하도록 구성될 수 있고, 일반적으로 예약 스테이션들(70A-N) 사이에 로드의 균형을 맞추기 위해 op들에 엔트리들을 할당할 수 있다.

[0049] LSD 예측기(66)는 op들 내의 저장들 및 로드들을 검사하도록 구성될 수 있고, 임의의 검출된 저장들 및 로드들의 PC들을 앞서 정렬 위반들을 일으키고 트레이닝 테이블에 엔트리들을 할당한 저장들 및 로드들의 PC들과 비교할 수 있다. PC가 주어진 저장에 대해 매칭하면, LSD 예측기(66)는 트레이닝 테이블 내의 대응하는 엔트리를 암하도록 구성될 수 있다. 일 실시예에서, LSD 예측기(66)는 엔트리를 암하기 전에 예측 표시자의 세기를 검사할 수 있다. 표시자가 임계값 위에 있으면, 엔트리는 암될 수 있고, 그렇지 않고, 표시자가 임계값 아래에 있으면, 엔트리는 암되지 않을 수 있다. 부가적으로, LSD 예측기(66)는 저장의 식별자로서 저장에 할당된 RNUM을 캡처(capture)하도록 구성될 수 있다.

[0050] 암된 엔트리에 매칭하는 로드가 검출되고 암된 엔트리에 대한 예측 표시자의 세기가 임계값 위에 있을 때, LSD

예측기(66)는 저장 식별자를 이용하여 저장에 대한 로드의 의존성을 발생하도록 구성될 수 있어, 저장이 발행된 후까지 예약 스테이션(70)에 의해 로드가 발행되지 못하게 한다. 일 실시예에서, LSD 예측기(66)는 로드가 의존성을 갖는다는 표시자와 함께 주어진 예약 스테이션(70)에 저장 RNUM을 전달하도록 구성될 수 있다. 부가적으로, 로드에 대한 복수의 매치가 존재하면, LSD 예측기(66)는 주어진 예약 스테이션(70)에 멀티매치 표시자를 전달할 수 있다. 다른 실시예들에서, LSD 예측기(66)는 멀티매치 경우에 복수의 저장 RNUM들을 예약 스테이션(70)에 전달하도록 구성될 수 있고, 예약 스테이션(70)은 로드당 하나보다 많은 저장 RNUM을 저장하도록 구성될 수 있다. 다른 실시예들은 다른 방식으로 저장 의존성들을 표시할 수 있다.

[0051] 예약 스테이션들(70A-N)은 로드/저장 유닛(도시되지 않음) 및/또는 실행 유닛들(도시되지 않음)의 부분으로서 활용될 수 있는 임의의 개수의 예약 스테이션들을 대표한다. 각각의 예약 스테이션(70A-N)은 연산들이 대응하는 기능 유닛에 의해 실행될 때까지 연산들을 저장하도록 구성될 수 있다. 일 실시예에 따른 예약 스테이션(70A) 내의 엔트리의 예가 도 3에 도시된다. 예약 스테이션들(70A-N) 각각은 실시예에 따라 다양한 개수의 엔트리를 포함할 수 있다. 각 엔트리는 의존성 표시자, 멀티매치 표시자, 의존성의 저장 RNUM, 연산이 로드 또는 저장인지를 나타내기 위한 로드/저장(L/S) 표시자, 및 연산의 PC를 포함할 수 있다. 다른 실시예들에서, 엔트리는 다른 필드들(예를 들어, 소스 레지스터, 목적지 레지스터, 소스 오퍼랜드들)을 포함하고 및/또는 도 3에 도시된 필드들 중 하나 이상을 생략할 수 있다. 또한, 다른 타입들의 엔트리들(예를 들어, 정수, 부동 소수점)이 상이하게 포맷팅될 수 있다.

[0052] LSD 예측기(66)는 리다이렉트 표시에 기초하여 정렬 위반들을 일으키는 로드-저장 쌍들을 식별하도록 구성될 수 있다. 리다이렉트 표시는 로드 및 저장 PC들 또는 다른 로드 및 저장 식별자들을 포함할 수 있다. 따라서, 프로세서에서 동일한 코드 시퀀스가 재인출되고 재실행될 때 장래에 그러한 이벤트들을 방지하기 위하여, LSD 예측기(66)는 정렬 위반들을 일으킨 저장들 및 로드들에 의해 트레이닝될 수 있다.

[0053] 레지스터 맵퍼(62)는 각각의 논리적 레지스터에 대한 엔트리를 갖는 메모리를 포함할 수 있다. 레지스터 맵퍼(62) 내의 각각의 논리적 레지스터에 대한 엔트리는 논리적 레지스터를 갱신하기 위해 가장 최근의 op의 RNUM을 저장할 수 있다. 부가적인 상태가 또한 재명명 맵 엔트리들에 저장될 수 있다. 예를 들어, 비트가 가장 최근의 op가 실행되었는지 여부를 나타낼 수 있다. 그러한 실시예에서, 레지스터 맵퍼(62)는 주어진 예약 스테이션(70)으로부터 발행된 op들을 식별하는 신호들을 수신할 수 있고, 이것은 레지스터 맵퍼(62)가 비트를 갱신할 수 있게 할 수 있다. 가장 최근의 op가 퇴거(retire)되었는지 여부를 나타내는 비트가 또한 포함될 수 있다.

[0054] 도 3에 도시된 유닛들에 대한 접속들 전부가 예시되지는 않고, 맵/디스패치 유닛(60)은 도시되지 않은 다른 연산들을 구현하는 부가적인 회로를 포함할 수 있다는 것에 주목한다. 예를 들어, 레지스터 맵퍼(62) 및 ROB(64)는 플러싱되는 op들을 고려하기 위해 그것들의 맵핑들을 조정하기 위해 리다이렉트 표시들을 수신할 수 있다. 부가적으로, 레지스터 맵퍼(62) 및 ROB(64)는 퇴거에 대한 그것들의 상태를 조정하기 위해 퇴거 op들의 표시를 수신할 수 있다(예를 들어, 새로운 op들에 할당을 위한 엔트리들을 비우는 것, 구조화된 재명명 상태를 갱신하는 것, 등등). 이들 연산들은 LSD 예측기(66)의 연산에 보조적이고, 따라서 본원에서는 더욱 상세하게 설명되지 않는다.

[0055] PC들 및 RNUM들은 저장들에 대한 식별자들로서 이용되고, PC들은 로드들에 대한 식별자들로서 이용되지만, 다른 실시예들은 프로세서 내에서 동작중인 명령어들을 고유하게 식별할 수 있는 임의의 식별자(예를 들어, 임의의 종류의 태그 또는 시퀀스 번호)를 이용할 수 있다는 것에 주목한다.

[0056] 이제 도 4로 가면, 로드-저장 의존성 예측기 테이블의 일 실시예가 도시된다. 테이블(90)은 실시예에 따라 다양한 개수의 엔트리들을 포함할 수 있다. 각 엔트리는 중첩하는 어드레스들을 갖고 비순차적으로 발행하는 것으로 예측된 로드-저장 쌍에 대응할 수 있다. 엔트리는 정렬 위반이 검출되는 것에 응답하여 테이블(90)에 할당될 수 있다. 정렬 위반이 발생한 이벤트에서, 저장 큐 엔트리는 인출 유닛으로 돌아가는, 위반을 일으킨 로드를 포함하여, 프로세서를 플러싱할 수 있고, 테이블(90)은 이러한 위반에 대해 트레이닝될 수 있어, 이러한 특정 로드-저장 쌍에 대한 엔트리가 테이블(90)에 부가된다. 통상적으로, 리다이렉트를 트리거하는 플러싱된 저장은 이미 발행되었을 것이고, 그래서 플러싱된 로드가 재인출되고 디코딩될 때, 테이블(90) 내의 엔트리가 암되지 않을 것이고, 로드는 정상적으로 발행할 수 있다. 그 PC에서 저장의 장래의 실행들에서, 저장은 테이블(90) 내의 대응하는 엔트리를 암할 것이고, 저장이 발행할 때까지 로드가 발행하지 못하게 할 것이다.

[0057] 테이블(90)은 복수의 op들에 의한 복수의 동시 액세스들 및 갱신들을 허용하도록 구성될 수 있다. 또한, 테이블(90)이 통합 테이블로서 예시되어 있지만, 상이한 필드들은 별개의 메모리들에 대응하는 별개의 테이블들일 수 있고, 별개의 테이블들의 엔트리들은 서로 연관된다. 예를 들어, 로드 PC들은 별개의 테이블일 수 있고, 저

장 PC들은 별개의 테이블일 수 있고, 로드 PC 엔트리는 특정 로드-저장 정렬 위반이 검출되고 트레이닝된 저장 PC 엔트리에 대응할 수 있다.

[0058] 각각의 엔트리는 유효 표시자(92)를 포함할 수 있다. 유효 표시자(92)는 엔트리가 유효 엔트리인지 및 엔트리가 엔트리에 의해 표시된 로드와 저장 사이의 의존성을 시행하기 위해 이용되어야 하는지를 표시할 수 있다. 일 실시예에서, 유효 표시자(92)는 리셋시 클리어될 수 있다. 유효 표시자(92)는 또한 대체 폴리시에 영향을 줄 수 있어, 무효 엔트리들은 새로운 엔트리들이 할당될 때 대체되는 제1 엔트리들일 수 있다. 일부 실시예들에서, 유효 표시자(92)는 테이블(90)의 엔트리들에 포함되지 않을 수 있다. 대신, 이들 실시예들에서, 카운터 필드(102)의 값은 엔트리가 유효인지를 표시하기 위해 이용될 수 있다. 다른 실시예들은 테이블 내의 카운터 필드(102)를 제외하고 오직 유효 표시자(92)를 이용할 수 있다.

[0059] 각각의 엔트리는 또한 특정 저장 연산을 식별하기 위해 저장 PC 값(94)을 포함할 수 있다. 일부 실시예들에서, 저장 PC 값은 아키텍처 레지스터들과 결합될 수 있고 및/또는 해쉬될 수 있다. 저장이 디스패치될 때, 테이블(90)의 저장 PC들은 디스패치된 저장의 PC에 대해 검색될 수 있다. 테이블(90)은 저장 PC 필드에 대한 CAM일 수 있고, 메모리 내의 각 엔트리는 비교를 행하기 위한 회로를 포함한다. 저장 PC 필드는 또한 CAM으로서 동작되는 한 세트의 레지스터들 및 비교기들일 수 있다. 디스패치된 저장이 임의의 엔트리들에 대해 매칭되면, 이들 엔트리들은 암된 비트(98) 세트를 가질 수 있다. 저장의 RNUM은 또한 엔트리의 저장 RNUM(96) 필드에 기록될 수 있다. 저장이 예약 스테이션으로부터 발행되면, 암된 비트(98)는 그 특정 저장에 의해 앞서 암된 테이블(90)의 임의의 엔트리들로부터 클리어될 수 있다.

[0060] 로드가 디스패치될 때, 테이블(90)의 각 엔트리의 로드 PC 값(100)은 디스패치된 로드의 PC에 대해 검색될 수 있다. 테이블(90)은 로드 PC 필드에 대한 CAM일 수 있다. 디스패치된 로드가 임의의 암된 엔트리들에 대해 매칭되면, 의존성은 특정 로드에 대해 확립되고 시행될 수 있다. 로드가 암되지 않은 엔트리(unarmed entry)에 대해 매칭되면, 대응하는 저장이 디스패치되지 않거나 이미 발행되었기 때문에 의존성은 확립되지 않고, 따라서 정렬 위반은 발생해서는 안 된다. 로드가 복수의 암된 엔트리들에 대해 매칭되면, 로드는 로드 자체가 발행하기 전에 모든 더 오래된 저장들이 발행될 때까지 기다릴 수 있다. 로드가 단일 암된 엔트리에 대해 매칭하면, 저장 RNUM은 로드와 예약 스테이션에 기록될 수 있다. 또한 로드가 유효 의존성을 가짐을 나타내기 위해 예약 스테이션에 로드에 대한 의존성 비트 세트가 존재할 수 있다.

[0061] 각각의 엔트리는 또한 카운터 필드(102)를 포함할 수 있다. 카운터(102)의 값은 엔트리 내의 그 특정 로드-저장 쌍에 대한 예측의 세기를 나타낼 수 있다. 일 실시예에서, 카운터(102)는 2-비트 업-다운 카운터일 수 있다. 다른 실시예에서, 카운터(102)는 다른 개수들의 비트들을 활용할 수 있다. 또한, 카운터(102)는 그의 최대 및 최소 값들에서 포화하도록 구성될 수 있다.

[0062] 저장이 엔트리에 대해 매칭할 때, 카운터 값(102)은 엔트리를 암하기 전에 검사될 수 있다. 카운터 값(102)이 임계값 아래에 있으면, 엔트리는 암되지 않을 수 있다. 카운터 값(102)이 임계값 위에 있으면, 엔트리는 암될 수 있다. 일부 실시예들에서, 엔트리는 카운터 값(102)을 검사하지 않고 암될 수 있다. 로드가 엔트리에 대해 매칭할 때, 카운터 값(102)은 또한 검사될 수 있다. 카운터 값(102)이 임계값 위에 있는 경우에만 의존성이 시행될 수 있다. 임계값은 실시예마다 다를 수 있고, 특정 동작 조건들에 따라 조정될 수 있다.

[0063] 다른 실시예에서, 테이블(90)의 엔트리들과 함께 에이지-아웃 카운터들(age-out counters)이 활용될 수 있다. 각각의 엔트리는 에이지-아웃 카운터를 포함할 수 있고, 에이지-아웃 카운터는 엔트리가 먼저 할당될 때 일부 초기 값으로 설정될 수 있다. 프로그램 가능한 간격을 카운트하기 위해 간격 카운터가 또한 활용될 수 있고, 간격 카운터가 만료될 때, 테이블(90) 내의 각각의 에이지-아웃 카운터는 감분할 수 있다. 그 다음에, 간격 카운터는 시작하고 프로그램 가능한 간격을 카운트할 수 있다. 간격이 경과할 때마다, 테이블(90) 내의 각각의 에이지-아웃 카운터는 감분할 수 있다. 엔트리가 로드-저장 쌍에 의해 액세스 또는 암되는 임의의 시간에 에이지-아웃 카운터는 고정된 양만큼 증분할 수 있다. 테이블(90) 내의 엔트리가 더 이상 이용되지 않으면, 결국 그의 에이지-아웃 카운터는 제로로 될 것이고, 이 제로 포인트에서 엔트리는 새로운 엔트리로 대체될 수 있다.

[0064] 다른 실시예들에서, 테이블(90)은 부가적인 필드들을 포함할 수 있고 및/또는 도 4에 도시된 하나 이상의 필드들을 생략할 수 있다. 또한, 테이블(90)은 다른 실시예들에서 상이하게 포맷팅될 수 있다.

[0065] 이제 도 5를 참조하면, 예측기 테이블 내의 로드-저장 쌍 엔트리들에 대응하는 카운터 값들의 표현의 일 실시예가 도시된다. 카운터 값들의 이러한 할당은 테이블(110)에서 2-비트 카운터에 대해 도시된다. 다른 실시예들에서, 다른 개수들의 비트들이 카운터에 의해 활용될 수 있다.

[0066] 일 실시예에서, "11" 또는 3의 카운터 값은 "강하게 인에이블됨(strongly enabled)"을 나타낼 수 있다. 이러한 카운터 값을 갖는 엔트리에 대하여, 로드-저장 쌍에 대한 의존성이 시행될 수 있다. "10" 또는 2의 카운터 값은 "약하게 인에이블됨(weakly enabled)"을 나타낼 수 있다. 엔트리가 "약하게 인에이블"되면, 의존성은 또한 시행될 수 있다. "01" 또는 1의 카운터 값은 "약하게 디스에이블됨(weakly disabled)"을 나타낼 수 있다. 엔트리가 "약하게 디스에이블"되면, 의존성은 대응하는 로드-저장 쌍에 대해 시행되지 않을 수 있다. "00" 또는 0의 카운터 값은 "강하게 디스에이블됨(strongly disabled)"을 나타낼 수 있다. 일부 실시예들에서, "강하게 디스에이블됨"은 또한 엔트리가 무효임을 나타낼 수 있다. 도 5에 도시된 실시예에서의 임계값은 2와 1 사이이다. 다른 실시예들에서, 임계값은 다른 값들일 수 있다.

[0067] 일 실시예에서, 엔트리가 먼저 할당될 때, 디폴트로 새로운 엔트리에 대한 카운터는 약하게 인에이블되는 것으로 설정될 수 있다. 카운터가 약하게 디스에이블되면(카운터=1), 엔트리와 매칭하는 로드-저장 쌍은 확립된 의존성을 갖지 않을 수 있다. 대신에, 로드는 의존성 없이 발행될 수 있다. 다른 실시예들에서, 카운터들의 다른 사이즈들이 활용될 수 있고, 카운터 값들은 상이한 표현들을 가질 수 있다.

[0068] 이제 도 6을 보면, 로드 연산을 프로세싱하기 위한 방법의 일 실시예가 도시된다. 논의의 목적으로, 본 실시예에서의 단계들은 순차적인 순서로 도시된다. 아래 설명된 방법의 다양한 실시예들에서, 설명된 요소들 중 하나 이상은 동시에, 도시된 것과 상이한 순서로 수행될 수 있거나, 완전히 생략될 수 있다는 것에 주목해야 한다. 다른 부가적인 요소들은 또한 필요에 따라 수행될 수 있다. 부가적으로, 흐름도의 부분들은 복수의 로드 연산들을 동시에 프로세싱하기 위해 병렬로 수행될 수 있다.

[0069] 일 실시예에서, 로드 연산은 맵/디스패치 유닛에 의해 수신될 수 있다(블록 120). 로드 연산은 프로세서 파이프라인의 사전 스테이지에서 디코딩되었을 수 있다. 그 다음, 로드-저장 의존성 예측기 테이블이 로드 연산과 동일한 PC를 갖는 엔트리들에 대해 검색될 수 있다(블록 122). 검색을 수행한 후에, 몇개의 매치들이 발견되는지가 결정될 수 있다(조건부 블록 124). 매치들이 발견되지 않으면(조건부 블록 124), 로드는 의존성 없이 예약 스테이션에 디스패치될 수 있다(블록 126). 로드는 암되지 않은 엔트리들에 대해 매칭할 수 있지만, 이들 암되지 않은 매치들은 의존성이 시행되도록 요구하지 않을 실제 매치들을 구성하지 않는다. 유사하게, 로드가 암된 엔트리에 대해 매칭하지만 예측 표시자 카운터가 임계값 아래에 있으면, 이것은 실제 매치를 구성하지 않고, 따라서, 의존성은 시행되지 않을 것이다. 일부 실시예들에서, 카운터는 저장이 이미 엔트리를 암하기 전에 카운터를 검사하였다면 로드에 대한 임계값과 비교될 필요가 없을 수 있다.

[0070] 로드가 시행될 필요가 있는 의존성을 갖지 않으면, 이것은 다양한 방식으로 표시될 수 있다. 예를 들어, 일 실시예에서, 의존성 비트는 로드가 의존성을 갖지 않음을 표시하도록 클리어될 수 있다. 블록(126) 후에, 픽커는 발행할 임의의 다른 연산들을 기다리지 않고 임의의 시간에 예약 스테이션으로부터 발행을 위한 로드를 선택할 수 있다(블록 132).

[0071] 암된 엔트리와의 단일 매치가 발견되면, 로드는 의존성을 갖고 예약 스테이션에 디스패치될 수 있다(블록 128). 대응하는 저장의 RNUM은 로드와 예약 스테이션 엔트리에 기록될 수 있다. 일 실시예에서, 엔트리가 매치로 고려되기 위해, 엔트리의 카운터 필드는 임계값 위에 있을 필요가 있을 수 있다. 예를 들어, 로드가 암된 엔트리에 대해 매칭하지만, 엔트리의 카운터 필드가 임계값 아래에 있다면(즉, 약하게 또는 강하게 디스에이블되면), 이것은 실제 매치를 구성하지 않을 수 있다. 블록(128) 후에, 로드는 의존적인 발행들인 대응하는 저장까지 발행을 기다릴 수 있다(블록 134).

[0072] 암된 엔트리들과의 복수의 매치들이 로드에 대해 발견되면(조건부 블록 124), 로드는 멀티매치 표시자 세트와 예약 스테이션에 디스패치될 수 있다(블록 130). 그 다음, 로드는 모든 더 오래된 저장들이 발행할 때까지 예약 스테이션으로부터 발행을 기다릴 수 있다(블록 136). 로드/저장 유닛은 복수의 예약 스테이션을 포함할 수 있고, 각각의 예약 스테이션은 그의 엔트리들 사이에 가장 오래된 저장을 추적하도록 구성될 수 있다. 복수의 매치들을 갖는 로드가 디스패치될 때, 각각의 예약 스테이션 내의 가장 오래된 저장이 기록될 수 있고, 각각의 예약 스테이션으로부터의 가장 오래된 저장이 발행한 후에, 로드는 그 다음에 하나의 사이클 후에 발행할 수 있다.

[0073] 이제 도 7로 가면, 로드-저장 의존성 예측 세기 표시자를 조정하기 위한 방법의 일 실시예가 도시된다. 논의의 목적으로, 본 실시예에서의 단계들은 순차적인 순서로 도시된다. 아래 설명된 방법의 다양한 실시예들에서, 설명된 요소들 중 하나 이상은 동시에, 도시된 것과 상이한 순서로 수행될 수 있거나, 완전히 생략될 수 있다는 것에 주목해야 한다. 다른 부가적인 요소들은 또한 필요에 따라 수행될 수 있다.

[0074] 의존성을 갖는 로드가 예약 스테이션으로부터 발행할 수 있다(블록 140). 로드는 로드-저장 쌍으로부터의 대응하는 저장이 발행된 후까지 발행하는 것이 지연되었을 수 있다. 대응하는 저장은 동일한 예약 스테이션으로부터 또는 상이한 예약 스테이션으로부터 발행되었을 수 있다. 로드가 예약 스테이션으로부터 발행되고 실행된 후에, 로드 데이터가 획득되는 곳이 결정될 수 있다(블록 142).

[0075] 로드 데이터가 저장 큐에 있었다면(조건부 블록 144), 이러한 특정 로드-저장 쌍에 대한 의존성 예측은 양호한 것으로 고려될 수 있고, 로드-저장 의존성 예측기에서의 대응하는 엔트리의 카운터는 증분할 수 있다(블록 146). 로드 데이터에 대한 저장 큐에 미스가 존재하면(조건부 블록 144), 저장에 대한 의존성은 로드에 대해 보장되지 않았을 수 있고(즉, 의존성 예측은 더 이상 유효하지 않고), 로드-저장 의존성 예측기 내의 대응하는 엔트리의 카운터는 감분할 수 있다(블록 148). 이러한 방법은 의존성들을 갖는 복수의 상이한 로드들에 대해 병렬로 수행될 수 있다.

[0076] 이제 도 8로 가면, 로드-저장 의존성 예측기 테이블 내의 엔트리들을 대체하기 위한 방법의 일 실시예가 도시된다. 논의의 목적으로, 본 실시예에서의 단계들은 순차적인 순서로 도시된다. 아래 설명된 방법의 다양한 실시예들에서, 설명된 요소들 중 하나 이상은 동시에, 도시된 것과 상이한 순서로 수행될 수 있거나, 완전히 생략될 수 있다는 것에 주목해야 한다. 다른 부가적인 요소들은 또한 필요에 따라 수행될 수 있다.

[0077] 포인터가 로드-저장 의존성 예측기 테이블 내의 인접하는 엔트리들의 그룹을 가리킬 수 있고, 인접하는 엔트리들의 그룹의 카운터 값들이 분석될 수 있다(블록 160). 일 실시예에서, 그룹은 4개의 엔트리를 포함할 수 있다. 다른 실시예들에서, 그룹은 다른 개수들의 엔트리들을 포함할 수 있다. 그 다음, 가장 낮은 카운터 값을 갖는 엔트리가 선택될 수 있다(블록 162). 하나보다 많은 엔트리가 가장 낮은 카운터 값을 가지면, 포인터는 이들 엔트리들 중 임의의 것을 랜덤으로 선택할 수 있거나, 포인터는 일부 다른 값 또는 메트릭을 이용하여 이들 가장 낮은 카운터 값의 엔트리들 사이를 구별할 수 있다.

[0078] 새로운 엔트리가 의존성을 갖는 새로 트레이닝된 로드-저장 쌍에 대해 이 포인트에서 할당될 필요가 있다면(조건부 블록 164), 그룹 중에서 가장 낮은 카운터 값을 갖는 선택된 엔트리가 폐기될 수 있고, 새로운 엔트리는 그 자리에 할당될 수 있다(블록 166). 새로운 로드-저장 쌍은 리다이렉트 및 플러시가 시그널링되는 것에 응답하여 할당될 수 있고, 리다이렉트는 임의의 포인트에서 일어날 수 있다는 것에 주목한다. 따라서, 조건부 블록(164)은 도 8의 흐름도 내의 다른 장소들에 배치될 수 있다. 새로운 엔트리가 할당된 후에, 포인터는 엔트리들의 다음 그룹으로 이동할 수 있다(블록 172). 새로운 엔트리가 이때에 할당될 필요가 없으면(조건부 블록 164), 가장 낮은 카운터 값은 제로인지가 결정될 수 있다(조건부 블록 168).

[0079] 가장 낮은 카운터 값은 제로이면(조건부 블록 168), 포인터는 그의 현재 위치에 머물 수 있고, 엔트리가 할당되기를 기다릴 수 있다(블록 170). 가장 낮은 카운터 값이 제로가 아니면(조건부 블록 168), 포인터는 예측기 내의 엔트리들의 다음 그룹으로 이동할 수 있다(블록 172). 일 실시예에서, 포인터는 로드 또는 저장이 로드-저장 의존성 예측기에 액세스할 때까지 엔트리들의 다음 그룹으로 이동하는 것을 기다릴 수 있다. 다른 실시예에서, 포인터는 다음 클록 사이클에서 엔트리들의 다음 그룹으로 이동할 수 있다. 블록(172) 후에, 방법은 그룹 내의 엔트리들을 분석하기 위해 블록(160)으로 돌아갈 수 있다. 도 8에 예시된 방법은 교체 폴리시의 하나의 가능한 구현이고, 다른 실시예들에서, 다른 교체 폴리시들(예를 들어, 최소 최근 사용)가 활용될 수 있다.

[0080] 다음으로 도 9를 참조하면, 시스템(180)의 일 실시예의 블록도가 도시된다. 도시된 바와 같이, 시스템(180)은 데스크톱 컴퓨터(190), 랩톱 컴퓨터(200), 태블릿 컴퓨터(210), 휴대 전화(220), 또는 다른 것들의 칩, 회로, 컴포넌트들 등을 나타낼 수 있다. 예시된 실시예에서, 시스템(180)은 외부 메모리(182)에 결합된 (도 1의) IC(10)의 적어도 하나의 사례를 포함한다.

[0081] IC(10)는 하나 이상의 주변장치들(184) 및 외부 메모리(182)에 결합된다. 메모리(182) 및/또는 주변장치들(184)에 대한 하나 이상의 공급 전압들뿐만 아니라 IC(10)에 대한 공급 전압들을 공급하는 전원 공급기(186)가 또한 제공된다. 다양한 실시예에서, 전원 공급기(186)는 배터리(예를 들어, 스마트폰, 랩톱 또는 태블릿 컴퓨터 내의 재충전가능한 배터리)를 나타낼 수 있다. 일부 실시예들에서, IC(10)의 하나보다 많은 사례가 포함될 수 있다(그리고 하나보다 많은 외부 메모리(182)가 또한 포함될 수 있다).

[0082] 메모리(182)는 DRAM(dynamic random access memory), SDRAM(synchronous DRAM), 더블 데이터 레이트(double data rate)(DDR, DDR2, DDR3, 등) SDRAM(mDDR3 등과 같은 SDRAM들의 모바일 버전들, 및/또는 LPDDR2 등과 같은 SDRAM들의 저전력 버전들을 포함함), RDRAM(RAMBUS DRAM), SRAM(static RAM) 등과 같은 임의의 타입의 메모리일 수 있다. 하나 이상의 메모리 장치들은 SIMM(single inline memory module), DIMM(dual inline memory

module) 등과 같은 메모리 모듈들을 형성하기 위해 회로 기판 상에 결합될 수 있다. 대안적으로, 장치들은 칩-온-칩(chip-on-chip) 구성, 패키지-온-패키지(package-on-package) 구성, 또는 멀티-칩 모듈(multi-chip module) 구성으로 IC(88)와 장착될 수 있다.

[0083] 주변장치들(184)은 시스템(180)의 타입에 따라, 임의의 원하는 회로를 포함할 수 있다. 예를 들어, 일 실시예에서, 주변장치들(184)은 와이파이, 블루투스, 셀룰러, 글로벌 포지셔닝 시스템(global positioning system) 등과 같은 다양한 타입의 무선 통신을 위한 장치들을 포함할 수 있다. 주변장치들(184)은 또한 RAM 저장소, 고체 상태 저장소, 또는 디스크 저장소를 포함하는 부가적인 저장소를 포함할 수 있다. 주변장치들(184)은 터치 디스플레이 스크린들 또는 멀티터치 디스플레이 스크린들을 포함하는 디스플레이 스크린, 키보드 또는 다른 입력 장치들, 마이크로폰들, 스피커들 등과 같은 사용자 인터페이스 장치들을 포함할 수 있다.

[0084] 이제 도 10을 보면, (도 1의) IC(10)에 포함된 회로를 대표하는 하나 이상의 데이터 구조들을 포함하는 컴퓨터 판독 가능한 매체(230)의 블록도의 일 실시예가 도시된다. 일반적으로 말해서, 컴퓨터 판독 가능한 매체(230)는 네트워크 및/또는 무선 링크와 같은 통신 매체를 통해 전달되는, 전기, 전자기, 또는 디지털 신호들과 같은 전송 매체 또는 신호들을 통해 액세스 가능한 매체뿐만 아니라, 자기 또는 광 매체, 예를 들어, 디스크, CD-ROM, 또는 DVD-ROM, RAM(예를 들어, SDRAM, RDRAM, SRAM 등), ROM 등과 같은 휘발성 또는 불휘발성 메모리 매체와 같은 임의의 비-일시적 저장 매체를 포함할 수 있다.

[0085] 일반적으로, 컴퓨터 판독 가능한 매체(230) 상의 회로의 데이터 구조(들)는 프로그램에 의해 판독되어 회로를 포함하는 하드웨어를 제조하기 위해 직접적으로 또는 간접적으로 이용될 수 있다. 예를 들어, 데이터 구조(들)는 Verilog 또는 VHDL과 같은 고레벨 설계 언어(HDL)로의 하드웨어 기능의 하나 이상의 거동 레벨 디스크립션(behavioral-level description) 또는 레지스터 트랜스퍼 레벨(register-transfer level; RTL) 디스크립션을 포함할 수 있다. 디스크립션(들)은 합성 라이브러리로부터의 게이트들의 리스트들을 포함하는 하나 이상의 네트리스트(netlist)를 생성하기 위해 디스크립션을 합성할 수 있는 합성 툴에 의해 판독될 수 있다. 네트리스트(들)는 회로를 포함하는 하드웨어의 기능을 또한 나타내는 게이트들의 세트를 포함한다. 그 다음에 네트리스트(들)는 마스크들에 적용될 기하학적 모양들을 설명하는 하나 이상의 데이터 세트들을 생성하도록 배치되고 라우팅될 수 있다. 마스크들은 그 다음에 회로에 대응하는 반도체 회로 또는 회로들을 생성하기 위해 다양한 반도체 제조 단계에서 이용될 수 있다. 대안적으로, 컴퓨터 판독 가능한 매체(230) 상의 데이터 구조(들)는 필요에 따라, 네트리스트(들)(합성 라이브러리를 갖거나 갖지 않음) 또는 데이터 세트(들)일 수 있다. 또 다른 대안에서, 데이터 구조들은 도식 프로그램의 출력, 또는 그로부터 도출되는 네트리스트(들) 또는 데이터 세트(들)를 포함할 수 있다.

[0086] 컴퓨터 판독 가능한 매체(230)는 IC(10)의 표현을 포함하지만, 다른 실시예들은 IC(10)의 임의의 부분 또는 부분들의 결합의 표현(예를 들어, LSD 예측기(37), LSU(46))을 포함할 수 있다.

[0087] 전술한 실시예들은 오직 구현들의 비한정적인 예들이라는 것에 주목해야 한다. 전술한 개시가 완전히 이해되면, 이 기술분야의 통상의 기술자에게 다수의 변형 및 수정이 명백해질 것이다. 다음의 청구항들은 모든 그러한 변형 및 수정을 포괄하도록 해석되는 것이 의도된다.

부호의 설명

[0088] 12: 프로세서 콤플렉스

18: L2 캐시

22: 메모리 컨트롤러

30: 코어

32: 인출 및 디코드(FED) 유닛

34: L1 명령어 캐시

36: 맵/디스패치 유닛

37: LSD 예측기

38: 재정렬 버퍼(ROB)

40: 메모리 관리 유닛(MMU)

42: 코어 인터페이스 유닛(CIF)

44: 실행 유닛들

46: 로드/저장 유닛(LSU)

48: L1 데이터 캐시

50, 52: 예약 스테이션

54: 저장 큐

56: 로드 큐

60: 맵/디스패치 유닛

62: 레지스터 맵퍼

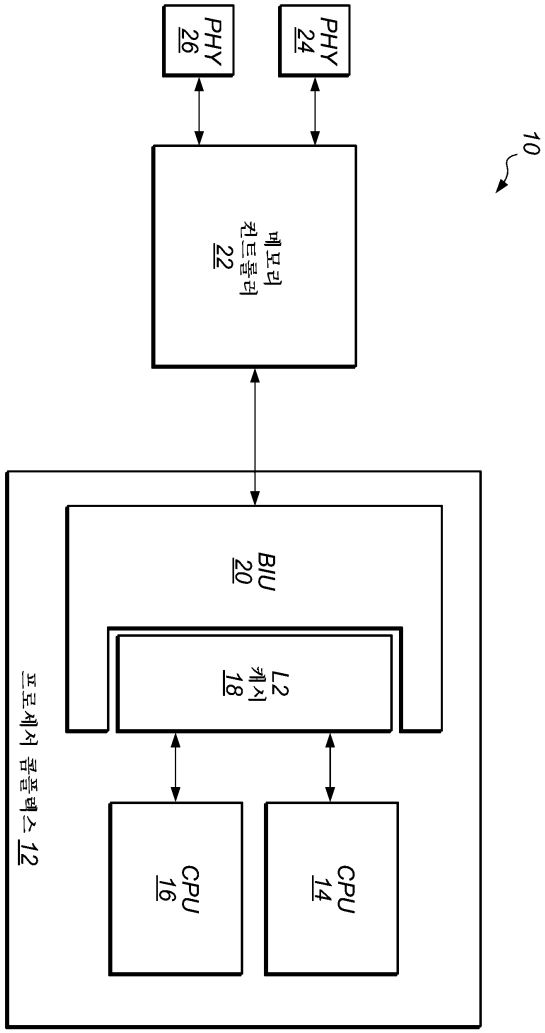
64: 재정렬 버퍼(ROB)

66: 로드-저장 의존성(LSD) 예측기

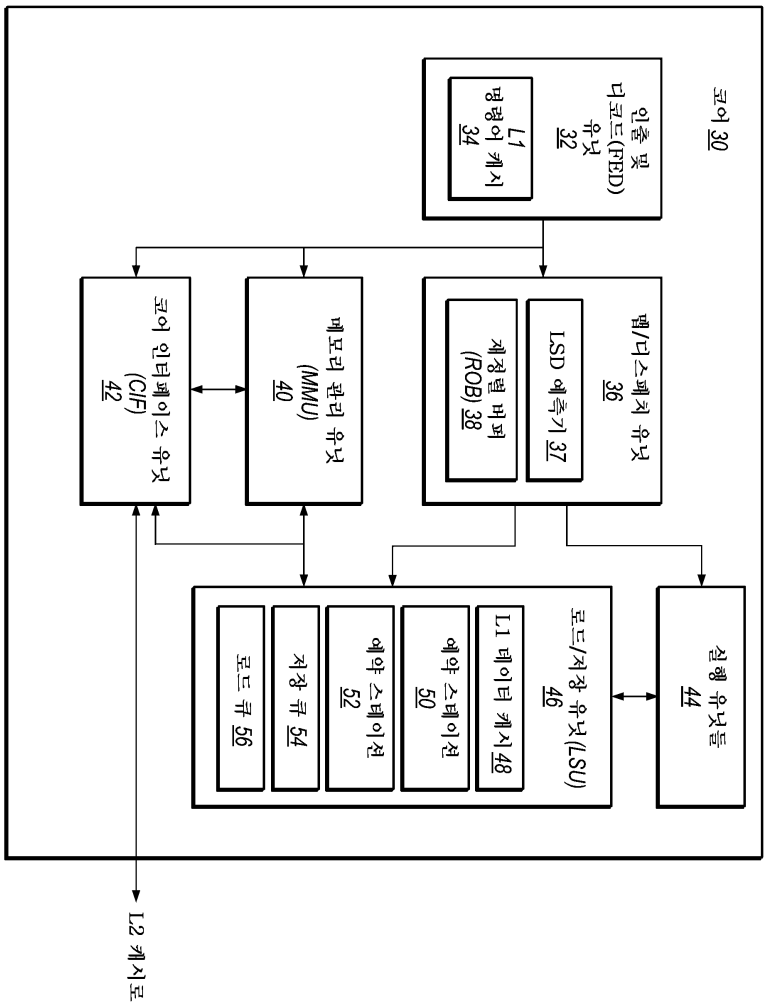
68: 디스패치 유닛

도면

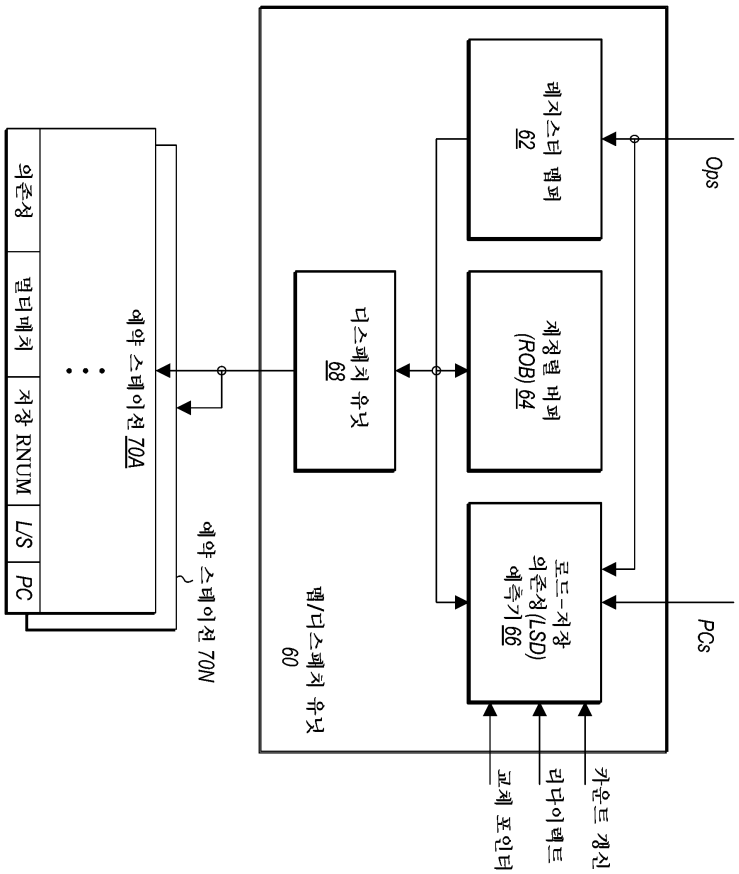
도면1


PHY 26
PHY 24
10
메모리 컨트롤러 22
BIU 20
L2 캐시 18
CPU 16
CPU 14
프로세서 콤플렉스 12

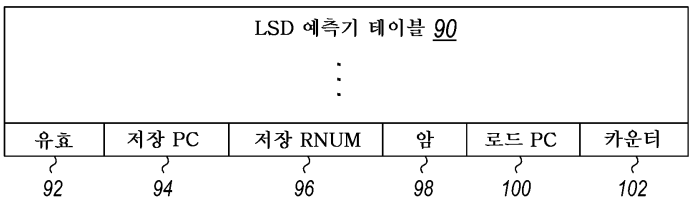
도면2


코어 30
인출 및 디코드(FED) 유닛 32
L1 명령어 캐시 34
멤/디스패치 유닛 36
LSD 예측기 37
재정렬 버퍼 (ROB) 38
메모리 관리 유닛 (MMU) 40
코어 인터페이스 유닛 (CIF) 42
실행 유닛들 44
로드/저장 유닛(LSU) 46
L1 데이터 캐시 48
예약 스테이션 50
예약 스테이션 52
저장 큐 54
로드 큐 56
L2 캐시로

도면3



도면4

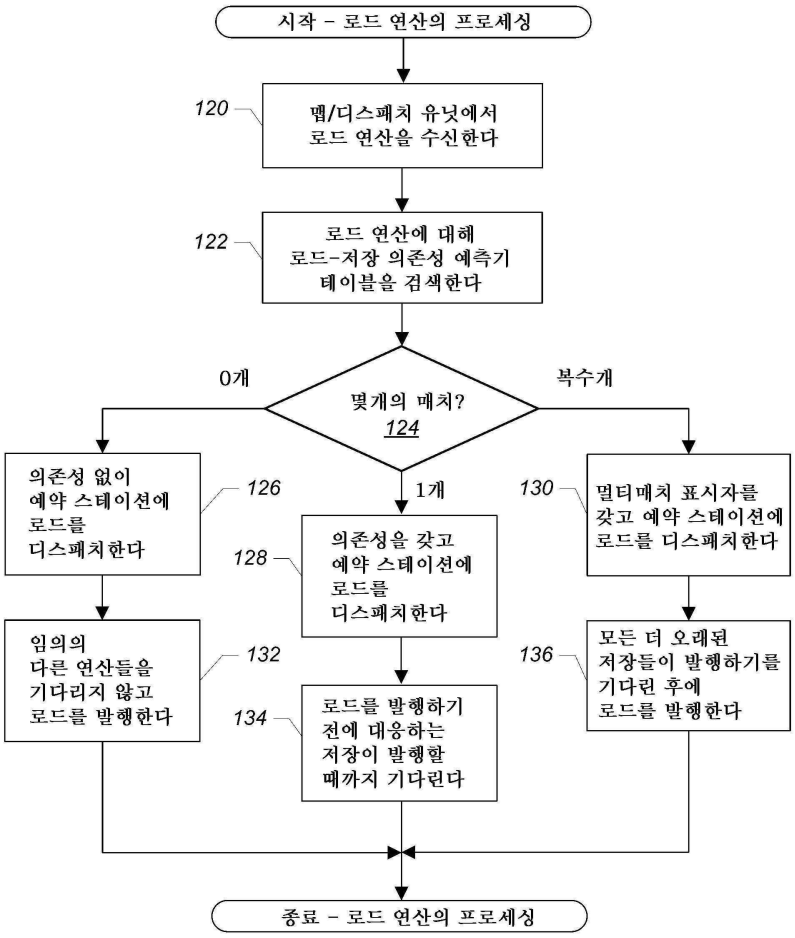


도면5

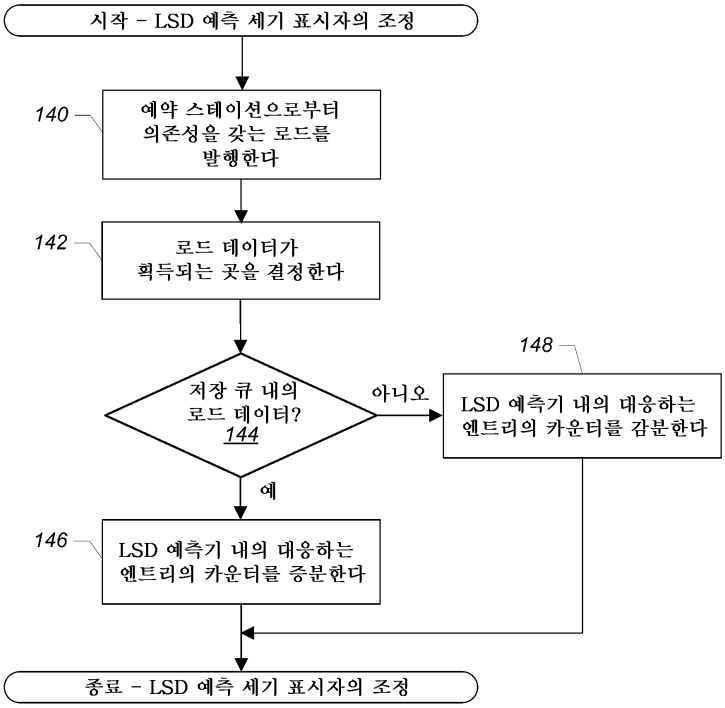
테이블 110

카운터 값	모드
11	강하게 인에이블됨
10	약하게 인에이블됨
01	약하게 디스에이블됨
00	강하게 디스에이블됨

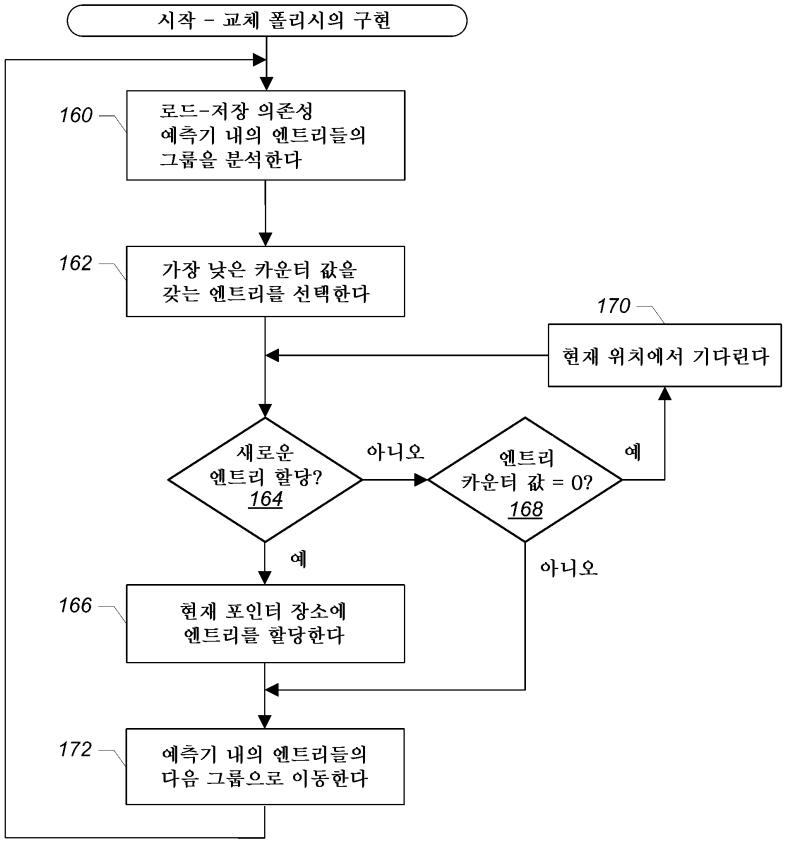
도면6


시작 - 로드 연산의 프로세싱
120
맵/디스패치 유닛에서
로드 연산을 수신한다
122
로드 연산에 대해
로드-저장 의존성 예측기
테이블을 검색한다
0개
몇개의 매치?
124
복수개
1개
의존성 없이
예약 스테이션에
로드를
디스패치한다
126
128
의존성을 갖고
예약 스테이션에
로드를
디스패치한다
130
멀티매치 표시자를
갖고 예약 스테이션에
로드를 디스패치한다
임의의
다른 연산들을
기다리지 않고
로드를 발행한다
132
134
로드를 발행하기
전에 대응하는
저장이 발행할
때까지 기다린다
136
모든 더 오래된
저장들이 발행하기를
기다린 후에
로드를 발행한다
종료 - 로드 연산의 프로세싱

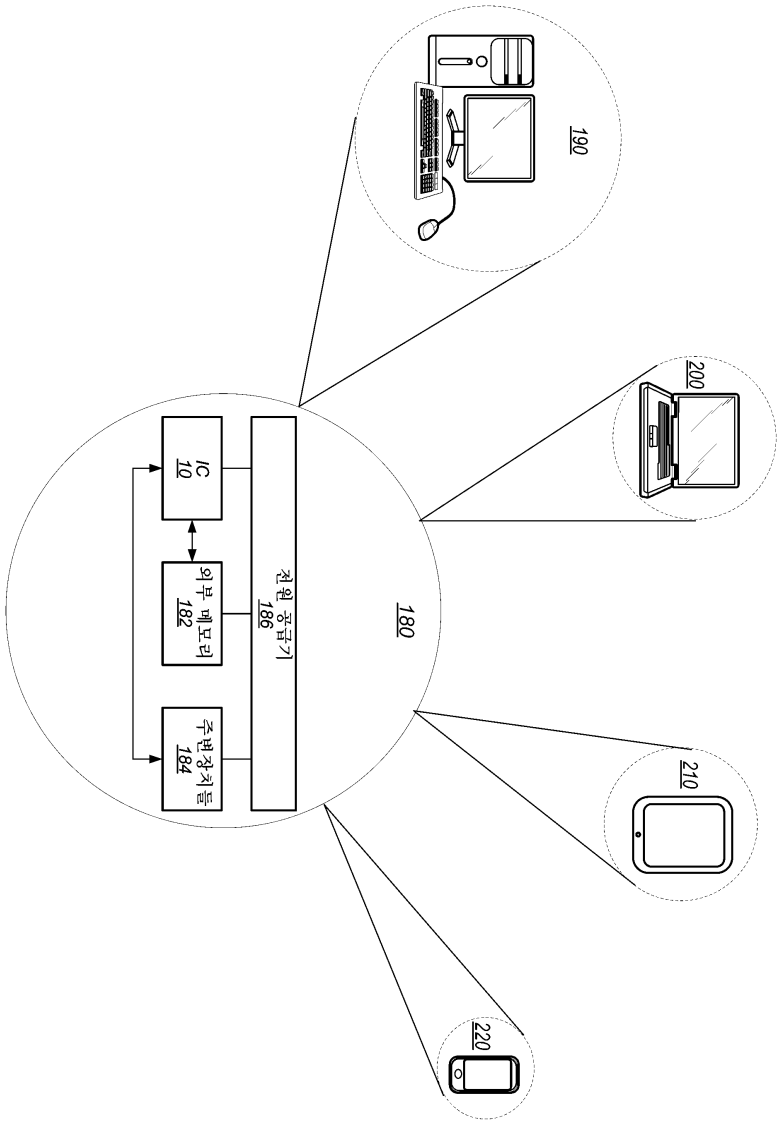
도면7


시작 - LSD 예측 세기 표시자의 조정
140
예약 스테이션으로부터 의존성을 갖는 로드를 발행한다
142
로드 데이터가 획득되는 곳을 결정한다
저장 큐 내의 로드 데이터? 144
아니오
148
LSD 예측기 내의 대응하는 엔트리의 카운터를 감분한다
예
146
LSD 예측기 내의 대응하는 엔트리의 카운터를 증분한다
종료 - LSD 예측 세기 표시자의 조정

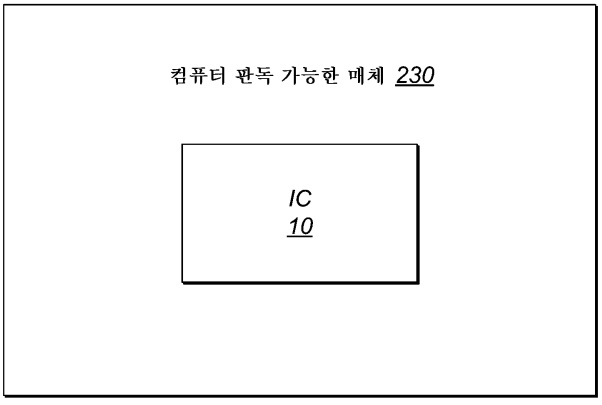
도면8


시작 - 교체 폴리시의 구현
160
로드-저장 의존성 예측기 내의 엔트리들의 그룹을 분석한다
162
가장 낮은 카운터 값을 갖는 엔트리를 선택한다
170
현재 위치에서 기다린다
새로운 엔트리 할당? 164
아니오
엔트리 카운터 값 = 0? 168
예
예
아니오
166
현재 포인터 장소에 엔트리를 할당한다
172
예측기 내의 엔트리들의 다음 그룹으로 이동한다

도면9


190
200
IC
10
외부 메모리
182
주변장치들
184
전원 공급기
186
180
210
220

도면10


컴퓨터 판독 가능한 매체 230
IC
10