(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2014년07월25일
(11) 등록번호 10-1423480
(24) 등록일자 2014년07월21일

(51) 국제특허분류(Int. Cl.)
G06F 12/00 (2006.01)
(21) 출원번호 10-2013-7005396
(22) 출원일자(국제) 2011년07월28일
심사청구일자 2013년02월28일
(85) 번역문제출일자 2013년02월28일
(65) 공개번호 10-2013-0041295
(43) 공개일자 2013년04월24일
(86) 국제출원번호 PCT/US2011/045797
(87) 국제공개번호 WO 2012/016085
국제공개일자 2012년02월02일
(30) 우선권주장
12/845,554 2010년07월28일 미국(US)
(56) 선행기술조사문헌
US06658557 B1
US20070240158 A1
전체 청구항 수 : 총 22 항

(73) 특허권자
인텔 코포레이션
미국 캘리포니아주 95054 산타클라라 미션 칼리지 불바드 2200
(72) 발명자
라즈와르 라비
미국 오레곤주 97229 포틀랜드 노스웨스트 카일라 엘엔 12792
크나우스 로라 에이
미국 오레곤주 97229 포틀랜드 노스웨스트 쉘삼 테라스 6464
(뒷면에 계속)
(74) 대리인
제일특허법인

심사관 : 이명진

(54) 발명의 명칭 **트랜잭셔널 메모리를 위한 최종 브랜치 레코드 표시자**

(57) 요 약

일 실시예에서, 프로세서는, 실행 유닛 및, 프로그램 실행 중 취해진 브랜치의 어드레스 정보를 저장하는 적어도 하나의 최종 브랜치 레코드(LBR) 레지스터를 포함한다. 이 레지스터는 트랜잭셔널 메모리(TM) 트랜잭션 중에 브랜치가 취해지는지를 나타내는 트랜잭션 표시자를 더 저장할 수 있다. 이 레지스터는 브랜치가 트랜잭션 중지에 의해 야기되는지를 나타내는 중지 표시자를 더 저장할 수 있다. 다른 실시예가 설명되고 청구된다.

대 표 도

(72) 발명자

라흐너 피터

독일 비더블유 72535 헤롤드스타트 바그너스트라세 7

라이 콘라드 케이

미국 오레곤주 98661 벤쿠버 아파트먼트 520 사우스이스트 콜롬비아 리버 드라이브 520

특허청구의 범위

청구항 1

실행 유닛, 및 프로그램 실행 중 취해진 브랜치의 소스 어드레스 정보 및 목적지 어드레스 정보 중 적어도 하나를 저장하는 적어도 하나의 최종 브랜치 레코드(last branch record;LBR) 레지스터를 포함하는 프로세서를 포함하되,

상기 적어도 하나의 LBR 레지스터는 상기 브랜치가 트랜잭셔널 메모리(a transactional memory;TM) 트랜잭션 중 취해지는지를 나타내는 트랜잭션 표시자를 더 포함하는

장치.

청구항 2

제 1 항에 있어서,

상기 적어도 하나의 LBR 레지스터는,

상기 TM 트랜잭션의 중단(an abort)이 상기 브랜치가 취해지기 이전에 발생되는지를 나타내는 중단 표시자를 더 저장하는

장치.

청구항 3

제 2 항에 있어서,

상기 소스 어드레스 정보 및 목적지 어드레스 정보 중 적어도 하나, 상기 트랜잭션 표시자 및 상기 중단 표시자를 상기 적어도 하나의 LBR 레지스터로 저장하기 위한 로직을 더 포함하는

장치.

청구항 4

제 1 항에 있어서,

상기 TM 트랜잭션이 중단되도록 하는 명령의 제 1 어드레스 정보 및 상기 중단에 응답하여 제어를 수신하는 핸들러의 제 2 어드레스 정보를 저장하는 중단 트랜잭션 레지스터를 더 포함하는

장치.

청구항 5

제 1 항에 있어서,

상기 적어도 하나의 LBR 레지스터로부터 정보를 저장하는 브랜치 트레이스 버퍼(a branch trace buffer;BTB)를 더 포함하되, 상기 적어도 하나의 LBR 레지스터는 복수의 LBR 레지스터 쌍을 포함하는

장치.

청구항 6

제 5 항에 있어서,

상기 BTB에 저장된 정보에 적어도 부분적으로 기초하여 획득된 상기 TM 트랜잭션에 대한 실행 백 트레이스(an execution back trace)를 사용하는 최적화기(an optimizer)를 더 포함하는

장치.

청구항 7

제 1 항에 있어서,

상기 적어도 하나의 LBR 레지스터에 저장된 정보에 기초한 상기 TM 트랜잭션의 실행 백 트레이스를 사용하여 상기 TM 트랜잭션 내부에 코드 플로우(code flow)의 검사를 허용하는 디버그 툴(a debug tool)을 더 포함하는

장치.

청구항 8

제 1 항에 있어서,

상기 적어도 하나의 LBR 레지스터에 저장된 정보에 기초한 상기 TM 트랜잭션의 실행 백 트레이스를 사용하여 상기 TM 트랜잭션의 코드 플로우의 사후 분석 검사(a post mortem inspection)를 가능하게 하는 분석 툴을 더 포함하는

장치.

청구항 9

브랜치가 프로그램 실행 중 취해질 때 프로세서의 최종 브랜치 레코드(LBR) 엔트리로 브랜치 어드레스 정보를 삽입하는 단계와,

상기 브랜치가 트랜잭셔널 메모리(TM) 트랜잭션의 실행 중 취해지면 상기 LBR 엔트리의 트랜잭션 표시자를 기록-그렇지 않으면 상기 트랜잭션 표시자를 기록하지 않음-하는 단계와,

상기 브랜치가 상기 TM 트랜잭션의 중단 이후에 발생하면 상기 LBR 엔트리의 중단 표시자를 기록-그렇지 않으면 상기 중단 표시자를 기록하지 않음-하는 단계를 포함하는

방법.

청구항 10

제 9 항에 있어서,

상기 브랜치가 상기 TM 트랜잭션 내에서 발생하는지를 판정하기 위하여 상기 프로세서의 트랜잭션 상태 정보를 사용하는 단계를 더 포함하는

방법.

청구항 11

제 9 항에 있어서,

상기 프로세서의 캐시 메모리의 브랜치 트레이스 버퍼로 복수의 LBR 엔트리로부터의 정보를 저장하는 단계를 더 포함하는

방법.

청구항 12

제 11 항에 있어서,

상기 브랜치 트레이스 버퍼에 저장된 하나 이상의 트랜잭션 표시자에 기초하여 상기 TM 트랜잭션 내에서 하나 이상의 브랜치를 식별하는 단계를 더 포함하는

방법.

청구항 13

제 12 항에 있어서,

상기 브랜치 트레이스 버퍼에 저장된 하나 이상의 중단 표시자에 기초하여 중단된 상기 TM 트랜잭션의 영역 중 취해지는 하나 이상의 브랜치를 식별하는 단계를 더 포함하는

방법.

청구항 14

제 13 항에 있어서,

취해진 상기 브랜치의 식별에 기초하여 상기 영역의 코드를 최적화하는 단계를 더 포함하는

방법.

청구항 15

실행 유닛, 및 각각이 프로그램 실행 중 취해진 브랜치의 소스 어드레스 정보 및 목적지 어드레스 정보를 저장하는 복수의 최종 브랜치 레코드(LBR) 레지스터 쌍을 포함하는 프로세서와,

상기 프로세서 연결된 동적 랜덤 액세스 메모리(DRAM)를 포함하되,

상기 LBR 레지스터 쌍의 각각은 상기 브랜치가 트랜잭션 메모리(TM) 트랜잭션 중 취해지는지를 나타내는 트랜잭션 표시자와 상기 TM 트랜잭션의 중단이 상기 브랜치가 취해지기 이전에 발생되는지를 나타내는 중단 표시자를 더 포함하는

시스템.

청구항 16

제 15 항에 있어서,

상기 프로세서는 상기 TM 트랜잭션에 대한 하드웨어 지원(hardware support)을 포함하는

시스템.

청구항 17

제 16 항에 있어서,

상기 하드웨어 지원은 각각 상기 TM 트랜잭션에 관한 트랜잭션 메타데이터를 저장하는 복수의 엔트리를 갖는 캐

시 메모리를 포함하는

시스템.

청구항 18

제 15 항에 있어서,

상기 프로세서는 상기 소스 어드레스 정보 및 목적지 어드레스 정보, 상기 트랜잭션 표시자 및 상기 중단 표시자를 상기 복수의 LBR 레지스터 쌍에 저장하는 로직을 더 포함하는

시스템.

청구항 19

제 15 항에 있어서,

상기 프로세서는 상기 TM 트랜잭션이 중단되도록 하는 명령의 제 1 어드레스 정보, 및 상기 중단에 응답하여 제어를 수신하는 핸들러의 제 2 어드레스 정보를 저장하는 중단 트랜잭션 레지스터를 더 포함하는

시스템.

청구항 20

제 15 항에 있어서,

상기 프로세서는 상기 복수의 LBR 레지스터 쌍으로부터의 정보의 저장을 위한 브랜치 트레이스 버퍼(BTB)를 저장하는 캐시 메모리를 더 포함하는

시스템.

청구항 21

제 20 항에 있어서,

상기 복수의 LBR 레지스터 쌍의 콘텐츠가 상기 BTB로 기록되도록 하기 위하여, 상기 복수의 LBR 레지스터 쌍으로 신호를 발행(issue)하는 로직을 더 포함하는

시스템.

청구항 22

제 15 항에 있어서,

상기 LBR 레지스터 쌍에 저장되는 정보에 적어도 부분적으로 기초하여 획득된 상기 TM 트랜잭션에 대한 실행 백 트레이스를 사용하는 최적화기를 더 포함하는

시스템.

명 세 서

기 술 분 야

[0001] 본 발명은 트랜잭셔널 메모리(transactional memory)를 위한 최종 브랜치 레코드 표시자(last branch record indicators)에 관한 것이다.

배 경 기 술

[0002] 기술이 발전함에 따라, 예를 들어, 다수의 스레드(threads)를 동시에 실행시킬 수 있는 하나 이상의 멀티코어를 통한 멀티프로세서 시스템의 형태로, 컴퓨터 시스템은 더 많은 수의 프로세서를 포함한다. 일반적으로, 스레드가 공유된 리소스에 액세스하기 이전에, 스레드는 공유된 리소스 상에서 잠금(a lock)을 획득할 수 있다. 공유된 리소스가 메모리에 저장된 데이터 구조인 상황에서, 잠금 메커니즘에 의해 제공되는 상호배타성(mutual exclusivity)의 관점에서 동일한 리소스에 액세스하도록 시도하는 모든 스레드는 이들의 연산의 실행을 직렬화할 수 있다. 이는 시스템 성능에 유해할 수 있고 예를 들어, 교착상태(deadlocks) 또는 다른 원치않는 동작에 기인하여, 프로그램 실패를 야기할 수 있다.

[0003] 시스템 내 코어 및 로지컬 프로세서의 수가 증가함에 따라 더 많은 소프트웨어 스레드가 실행가능하게 된다. 하지만, 동시에 실행될 수 있는 다수의 소프트웨어 스레드에서의 증가는 소프트웨어 스레드 사이에서 공유된 데이터를 동기화하는 것에 있어서 문제점을 발생시킨다. 다수의 코어 또는 다수의 로지컬 프로세서 시스템에서 공유된 데이터에 액세스하는 하나의 일반적인 솔루션은 공유된 데이터에 대한 다수의 액세스에 걸쳐 상호 배제를 보장하도록 잠금을 사용한다. 하지만, 다수의 소프트웨어 스레드를 실행시키도록 계속 증가하는 기능은 잠재적으로 거짓 경합(false contention) 및 실행의 직렬화를 야기한다.

[0004] 잠금 메커니즘(locking mechanism)의 활용으로부터 기인한 성능 손실을 감소시키기 위해, 일부 컴퓨터 시스템은 트랜잭셔널 메모리(TM)를 사용할 수 있다. 트랜잭셔널 메모리는 일반적으로, 잠금 메커니즘을 활용하는 것 없이 다수의 스레드가 공유된 리소스에 동시에 액세스하도록 허용하는 동기화 모델을 지칭한다.

[0005] 종종 트랜잭셔널 실행은 마이크로 연산(micro-operations), 연산, 또는 명령의 그룹을 추론적으로(speculatively) 실행하는 것을 포함할 수 있다. 현재 TM 시스템은 프로세서 하드웨어가 트랜잭션을 수행하는데 사용되는 하드웨어 TM 시스템, 트랜잭션이 소프트웨어로 구현되는 소프트웨어 TM 시스템 및 하드웨어 및 소프트웨어가 트랜잭션을 실행하는데 사용될 수 있는 하이브리드 TM 시스템을 포함한다. 통상적으로, 하드웨어 TM 시스템은 가장 효율적일 수 있지만, 트랜잭션이 너무 커진다면, 예를 들어, 하드웨어 메모리를 오버플로우하여(overflows), 트랜잭션은 일반적으로 재시작된다. 여기서, 오버플로우까지 트랜잭션을 실행시키는데 걸리는 시간은 잠재적으로 낭비된다.

[0006] 프로세서는 다양한 하드웨어를 포함하고 테스팅, 디버그 및 다른 연산을 위한 하드웨어를 또한 제공할 수 있다. 예를 들어, 프로세서 체크포인팅(processor checkpointing), 예외 레포팅(exception reporting), 브랜치 레코딩(branch recording) 등을 수행하기 위해 다양한 레지스터가 프로세서에 제공될 수 있다. 하지만, 이러한 성능 모니터링 하드웨어는 일반적으로 트랜잭셔널 메모리의 환경에서 일반적으로 사용되지 않는다.

도면의 간단한 설명

[0007] 도 1은 본 발명의 실시예에 따른 최종 브랜치 레지스터 쌍의 블록도이다.

도 2는 본 발명의 실시예에 따른 프로세서 코어의 블록도이다.

도 3은 본 발명의 다른 실시예에 따른 프로세서 코어의 블록도이다.

도 4는 본 발명의 일 실시예에 따른 브랜치 레코드를 생성하기 위한 방법의 흐름도이다.

도 5는 본 발명의 실시예에 따른 브랜치 레코드를 사용하기 위한 방법의 흐름도이다.

도 6은 본 발명의 실시예에 따른 시스템의 블록도이다.

발명을 실시하기 위한 구체적인 내용

[0008] 다양한 실시예에서 프로그램의 실행 동안 획득된 브랜치 정보가 레코딩될 수 있고 또한 프로파일링, 디버깅 및/또는 트랜잭셔널 메모리 트랜잭션의 최적화의 목적을 위해 사용될 수 있다. 이 방식으로, 개선된 트랜잭셔널 메모리 연산 및 코드가 실현될 수 있다. 다양한 실시예에서, 이러한 레지스터에 포함된 정보는 최종 브랜치 정보(last branch information)를 포함할 수 있고, 여기에 프로그램 실행 중 취해진 브랜치에 관한 정보가 레코딩된다. 이러한 브랜치의 식별 외에, 트랜잭셔널 메모리의 계류중인 트랜잭션(a peding transaction)의 실행에 관한

정보가 또한 레코딩될 수 있다. 이 정보를 사용하여, 디버깅 및 다른 연산이 트랜잭셔널 메모리 트랜잭션에 대해 실현될 수 있다.

[0009] 트랜잭셔널 실행은 보통 복수의 명령 또는 연산을 코드의 트랜잭션, 어토믹 섹션(atomic section) 또는 크리티컬 섹션(critical section)으로 그룹화하는 것을 포함한다. 일부 경우에, 트랜잭션이라는 용어는 복수의 연산으로 구성된 매크로 명령을 지칭하고, 반면 다른 경우에 명령은 더 작은 연산, 예를 들어, 마이크로 연산(micro-operation;uop)을 지칭할 수 있다. 일반적으로 트랜잭션을 확인하기 위한 두 가지 방법이 존재한다. 제 1 예는 소프트웨어에서 트랜잭션을 구획설정(demarcating)하는 것을 포함한다. 여기서, 일부 소프트웨어 구획설정은 트랜잭션을 식별하기 위해 코드에 포함된다. 이전 소프트웨어 구획설정과 함께 구현될 수 있는 다른 실시예에서, 트랜잭션은 하드웨어에 의해 그룹화되거나 트랜잭션의 시작 및 트랜잭션의 끝을 나타내는 명령에 의해 인식된다.

[0010] 프로세서에서, 트랜잭션은 추론적으로 또는 비추론적으로(non-speculatively) 실행된다. 제 2 경우로서, 명령의 그룹화는 액세스 될 메모리 위치에 대한 잠금(lock) 또는 보장된 유효 액세스(guaranteed valid access)의 어떤 형태로 실행된다. 대안으로, 트랜잭션의 추론적 실행은 더 일반적이고, 여기서 트랜잭션은 추론적으로 실행되고 트랜잭션의 종료시에 커미트(committed)된다. 본원에서 사용된 바와 같이, 트랜잭션의 계류(a pendency)는, 실행을 시작하였고 커미트 (committed)또는 중단(aborted)되지 않은, 즉, 계류중인 트랜잭션을 지칭한다.

[0011] 통상적으로, 트랜잭션의 추론적 실행 중, 트랜잭션이 커미트될 때까지 메모리에 대한 업데이트는 글로벌하게 가시적(visible)으로 이루어지지는 않는다. 트랜잭션이 여전히 계류중인 동안, 메모리로부터 로딩되고 메모리에 기록된 위치가 트랙킹(tracked)된다. 이들 메모리 위치의 성공적인 검증시에, 트랜잭션이 커미트되고 트랜잭션 동안에 이루어진 업데이트가 글로벌적하게 가시적으로 이루어진다. 하지만, 이의 계류 동안 트랜잭션이 무효화되면, 업데이트를 글로벌하게 가시적으로 수행하지 않고 트랜잭션이 재시작된다.

[0012] 다양한 실시예에서, 최종 브랜치 레코딩 설비가 프로세서에 제공될 수 있다. 이러한 설비는 레지스터 세트에 브랜치 레코드를 저장할 수 있고, 이의 일 실시예는 머신 또는 모델 특정 레지스터(model specific register;MSR)가 될 수 있다. 예를 들어, 프로세서는 가장 최근에 취해진 브랜치에 관한 정보를 저장하기 위한 MSR의 최종 브랜치 레코드(LBR) 스택을 포함할 수 있다. 브랜치 레코드는 레지스터의 쌍을 포함하고, 이 중 하나는 선형 어드레스가 될 수 있는 브랜치 프롬 명령 어드레스(a branch-from instruction address) 및 브랜치 투 명령 어드레스(a branch-to instruction address)를 저장하기 위한 것이다. 일부 구현에서, 레지스터는 자동으로 판독될 수 있고, 반면 다른 실시예에서, 제어 신호가 스택으로 하여금 브랜치 레코드를 브랜치 트레이스 메시지(branch trace message;BTM)로서 선택된 목적지 위치로 송신하게 할 수 있다.

[0013] LBR 레지스터의 동작은 사용가능(enabled) 또는 사용불가능(disabled)하게 되도록 제어될 수 있음에 유의해야한다. 예를 들어, 프로세서가 취해진 브랜치에 대해 브랜치 레코드를 자동으로 레코딩하는 것을 가능하게 하는 디버그 제어 MSR에 LBR 플래그가 존재할 수 있다. 일부 실시예에서, LBR 외에, 인터럽트(interrupts) 및 예외(exceptions)에 관한 정보가 또한 LBR 레지스터에 레코딩될 수 있지만, 다른 실시예에서는, 개별 레코딩 설비가 이러한 상황에 대해 제공될 수 있다.

[0014] 일 실시예에서, 디버거는 소스를 향한 특정 버그의 발현(manifestation)으로부터 백워드 트레이스를 가능하게 하도록 LBR 스택에서 어드레스를 사용할 수 있다.

[0015] LBR 스택의 MSR의 수는 상이한 실시예에서 달라질 수 있음에 유의해야한다. 예를 들어, 상이한 구현에서, LBR 스택의 크기는 4, 8, 또는 16이 될 수 있지만, 본 발명의 범위는 이와 관련하여 제한되지 않는다. 상술된 바와 같이, 최종 브랜치 레코딩 메커니즘은 브랜치 명령(점프, 루프 및 호출 명령과 같은)뿐만 아니라, 명령 포인터(instruction pointer)에서의 변화를 야기하는 다른 동작(외부 인터럽트, 트랩(traps) 및 폴트(faults)와 같은)을 트랙킹할 수 있다.

[0016] 이제 도 1을 참조하면, 본 발명의 실시예에 따른 LBR 레지스터 쌍의 블록도가 도시된다. 도 1에 도시된 바와 같이, 레지스터 쌍(10)은 두 MSR, 즉, 제 1 MSR(12) 및 제 2 MSR(14)을 포함할 수 있다. 보이는 바와 같이, MSR(12)는 소스 어드레스를 저장할 수 있다. 즉, 이 레지스터는 최근 브랜치의 소스 어드레스, 즉, 브랜치가 발생하는 "프롬" 어드레스를 저장할 수 있다. 다음으로, 레지스터(14)는 목적지 어드레스, 즉, 제어가 브랜치에 의해 통과되는 "투" 어드레스를 저장할 수 있다. 명령 포인터 어드레스를 저장하는 어드레스 필드(15) 외에, 추가 필드가 이들 레지스터에 존재할 수 있다. 특히, 도 1의 실시예에서, 트랙잭션 표시자(a transaction indicator)(16) 및 중단 표시자(an abort indicator)(17)는 각 레지스터와 연관될 수 있다. 더 구체적으로, 대

응하는 브랜치가 트랜잭션의 실행 중 발생할 때 트랜잭션 표시자(16)가 설정될 수 있다. 다음으로, 중단 표시자(17)는 트랜잭션이 중단된 이후에 발생하는 제 1 브랜치에 대해 설정될 수 있다. 더 정확하게, 중단은 중단 핸들러(an abort handler)로의 점프를 야기한다. 이 점프는 또한 중단 표시자 세트와 함께 LBR에 레코딩된다. 이 중단 표시자는 다른 브랜치로부터 이 브랜치를 구별하는데 사용된다. 중단 이벤트에 관한 정보의 저장에 대한 추가 상세가 이하에서 논의된다. 이 방식으로, 브랜치가 발생하는 트랜잭션의 실행에 관한 정보가 레코딩될 수 있다. 이하에서 더 논의되는 바와 같이, 이 정보를 사용하여, 잠재적으로 개선된 실행, 예를 들어, 최적화 등을 통해 트랜잭션의 실행에 대한 더 큰 이해가 획득될 수 있다.

[0017] 상술된 바와 같이, 브랜치 정보에 대한 레지스터 쌍 외에, 중단 이벤트에 관한 정보가 레코딩될 수 있다. 특히, 도 1에서 도시된 바와 같이, 추가 레지스터 쌍(20)은 계류중인 트랜잭션에 중단이 발생하게 하도록 하는 명령에 관한 정보를 포함할 수 있다. 특히, 레지스터(22)는 트랜잭션을 중단하도록 야기하는 명령에 대한 명령 포인터를 저장할 수 있다. 이 쌍 중 제 2 레지스터, 즉 레지스터(24)는 목적지 어드레스를 저장할 수 있고, 이 목적지 어드레스는 중단으로부터의 복구(recovery)를 가능하게 하는 코드, 로직 등에 대응하는 중단 핸들러의 어드레스가 될 수 있다. 도 1의 실시예에서 이들 두 레지스터 쌍만이 도시되었지만, LBR 스택은 더 많은 쌍을 포함할 수 있음을 이해해야한다. 또한, LBR에 포함된 구조, 배열 및 정보는 다른 실시예에서 상이할 수 있다. 예를 들어, 일부 실시예에서, LBR 스토리지는 원형 어레이로서 구현될 수 있다.

[0018] 추가 정보가 LBR 레지스터에 존재할 수 있음에 유의해야한다. 예를 들어, 일부 실시예에서, 취해진 브랜치가 정확하게 예측되었음을 세트가 나타낼 때, 예측 표시자가 제공될 수 있다. 그렇지 않은 경우, 표시자는 브랜치가 예측실패(mispredicted)됨을 나타내도록 클리어될 수 있다. 물론 추가적인 표시자 및 다른 정보가 이들 레지스터에 제공될 수 있다.

[0019] 일부 실시예에서 최종 브랜치 레코드 탑 오브 스택(top-of-stack;TOS)은 레코딩된 가장 최근 브랜치, 인터럽트, 또는 예외를 포함하는 LBR 스택의 MSR로의 포인터를 저장할 수 있다.

[0020] 이하에서 설명되는 바와 같이, 일부 실시예에서, 디버그 스토어(a debug store;DS) 메커니즘은 브랜치 트레이스 스토어(a branch trace store;BTS)에 BTM을 저장할 수 있고, BTM은 하드웨어 버퍼, 캐시, 또는 시스템 메모리와 같은 메모리 계층(a memory hierarchy)의 주어진 부분 중 일부가 될 수 있다. 일 실시예에서, 디버그 제어 MSR의 BTS 플래그가 설정될 때, 취해진 브랜치, 인터럽트, 또는 예외가 검출될 때마다, 브랜치 레코드가 DS 저장 영역의 BTS 버퍼에 저장된다.

[0021] 일부 실시예에서, 최종 브랜치 레코드의 필터링은, LBR에서 캡쳐되지 않을 브랜치의 서브세트의 조건을 특정하기 위한 필드를 제공할 수 있는 LBR 선택 레지스터를 통해 실현될 수 있다. 예를 들어, 이 레지스터는 사전결정된 권한 레벨 등에서 발생하는 브랜치를 필터링하기 위해 필드를 포함할 수 있다.

[0022] 따라서, 일 실시예에서, 각각의 브랜치 레코드는 두 선형 어드레스를 포함하고, 이는 브랜치, 인터럽트, 또는 예외에 대한 "프롬' 및 "투" 명령 포인터를 나타낸다. 프롬 및 투 어드레스의 콘텐츠는 브랜치의 소스에 따라, 상이할 수 있다. 레코드가 취해진 브랜치에 대한 것인 경우에, "프롬" 어드레스는 브랜치 명령의 어드레스이고 "투" 어드레스는 브랜치의 타겟 명령이다. 레코드가 인터럽트에 대한 것인 경우에, "프롬" 어드레스는 인터럽트에 대해 저장되는 리턴 명령 포인터(return instruction point;RIP)이고 "투" 어드레스는 인터럽트 핸들러 루틴의 제 1 명령의 어드레스이다. RIP는 인터럽트 핸들러로부터 리턴할 시에 실행될 다음 명령의 선형 어드레스이다. 레코드가 예외에 대한 것인 경우에, "프롬" 어드레스는 예외가 발생되도록 야기한 명령의 선형 어드레스이고 "투" 어드레스는 예외 핸들러 루틴의 제 1 명령의 어드레스이다.

[0023] 이제 도 2를 참조하면, 본 발명의 일 실시예에 따른 프로세서 코어의 블록도가 도시된다. 도 2에 도시된 바와 같이, 프로세서 코어(100)는 멀티 스테이지 파이프라인형 비순서적 프로세서(a multi-stage pipelined out-of-order processor)일 수 있다. 프로세서 코어(100)는 본 발명의 실시예에 따른 브랜치 레코드 레포팅과 함께 사용되는 다양한 피쳐를 도시하기 위해 도 2에 상대적으로 단순화된 뷰로 도시된다. 또한, 상술된 바와 같이 프로세서는 TM 트랜잭션에 대한 하드웨어 지원을 제공할 수 있거나 제공하지 않을 수 있음을 이해할 것이다. 도시를 위해 코어(100)가 이러한 하드웨어 지원을 포함함을 가정한다. 하지만, 일부 실시예에서, LBR에 저장된 트랜잭셔널 상태 정보를 사용하여, 이러한 하드웨어 지원의 부재에도, 트랜잭션 실행의 분석이 발생할 수 있다. 이 방식으로, 어떤 브랜치가 트랜잭션 동안 발생하는지, 브랜치가 트랜잭션 중단 등이 되도록 야기하는지의 이해가 달성될 수 있다. 또한 디버그, 최적화, 프로파일링 또는 다른 액티비티가 이 정보와 함께 발생할 수 있다.

[0024] 도 2에서 도시된 바와 같이, 코어(100)는 프론트 엔드 유닛(110)을 포함할 수 있고, 이는 실행될 명령을 페치

(fetch)하고 프로세서에서 나중의 사용을 위해 명령을 준비하는데 사용될 수 있다. 예를 들어, 프론트 엔드 유닛(110)은 페치 유닛(101), 명령 캐시(103), 및 명령 디코더(105)를 포함한다. 일부 구현에서, 프론트 엔드 유닛(110)은 마이크로 연산 스토리지 뿐만 아니라 마이크로코드 스토리지와 함께, 트레이스 캐시(a trace cache)를 더 포함할 수 있다. 페치 유닛(101)은 매크로 명령을 예를 들어, 메모리 또는 명령 캐시(103)로부터 페치할 수 있고, 이를 프리미티브(primitives), 즉, 프로세서에 의한 실행을 위한 마이크로연산(micro-operations)으로 디코딩하기 위해 이를 명령 디코더(105)에 피드할 수 있다.

[0025] 프론트 엔드 유닛(110)과 실행 유닛(120) 사이에 연결된 것은 명령 디스패처(an instruction dispatcher)이고, 이는 마이크로 명령을 수신하고 이들을 실행을 위해 준비하도록 비순서적 구현에서 비순서적 로직으로서 구현될 수 있다. 더 구체적으로, 명령 디스패처(115)는 실행을 위해 필요한 다양한 리소스를 할당할뿐만 아니라, 레지스터 파일(130) 및 확장된 레지스터 파일(135)과 같은 다양한 레지스터 파일 내 저장 위치 상으로 로지컬 레지스터의 재명명(renaming)을 제공하는 다양한 버퍼를 포함할 수 있다. 레지스터 파일(130)은 정수 및 부동 소수점 연산을 위한 개별 레지스터 파일을 포함할 수 있다. 확장된 레지스터 파일(135)은 벡터 사이즈 유닛(vector-sized unit)에 대한 스토리지, 예를 들어, 레지스터 당 256 또는 512 비트를 제공할 수 있다.

[0026] 도 2에서 더 보여지는 바와 같이, 프로세서(100)는 MSR(125) 세트를 포함할 수 있다. 위에서 논의된 바와 같이, 다양한 타입의 모델 특정 정보는 이러한 레지스터에 저장될 수 있다. 도 2의 실시예에서, 위에서 논의된 바와 같이 코드의 실행 중 취해진 브랜치에 관한 정보를 저장할 수 있는 LBR(128) 세트가 도시된다. 이들 또는 유사한 레지스터는 인터럽트, 트랩, 예외 등과 같은 다른 실행 발생에 관한 정보를 더 포함할 수 있다.

[0027] 예를 들어, 다양한 정수, 부동 소수점, 및 단일 명령 다중 데이터(single instruction multiple data;SIMD) 로직 유닛을 포함하는, 실행 유닛(120)에 다양한 리소스가 존재할 수 있다. 예를 들어, 이러한 실행 유닛은 하나 이상의 산술 로직 유닛(arithmetic logic unit;ALU)(122)을 포함할 수 있다. 또한, 실행 유닛은 성능 모니터링 유닛(a performance monitoring unit;PMU)(124)을 더 포함할 수 있다. 다양한 실시예에서, PMU(124)는 다양한 정보, 예를 들어, 프로파일링 카운터, MSR의 정보 등의 획득을 제어하는데 사용될 수 있다. 본원의 특정 구현에서, PMU(124) 또는 다른 이러한 로직은 LBR(128)에서 트랜잭션 실행에 관한 정보를 포함하는, 정보의 레코딩을 제어하고 추후 사용을 위한 이러한 정보를 더 획득하는데 사용될 수 있다. 결과는 퇴거 로직(retirement logic), 즉 재순서 버퍼(reorder buffer;ROB)(140)에 제공될 수 있다. 더 구체적으로, ROB(140)는 실행된 명령과 연관되는 정보를 수신하는 다양한 어레이 및 로직을 포함할 수 있다. 이 정보는 또한 명령이 유효하게 퇴거될 수 있고 결과 데이터가 프로세서의 구조적 상태에 커미트되는지 또는 명령의 적합한 퇴거를 방지하는 하나 이상의 예외가 발생되는지를 판정하도록 ROB(140)에 의해 검토된다. 물론, ROB(140)는 퇴거와 연관된 다른 연산을 처리할 수 있다.

[0028] 도 2에서 도시된 바와 같이, ROB(140)는 캐시(150)에 연결되고, 일 실시예에서, 저레벨 캐시(예를 들어, L1 캐시)일 수 있지만, 본 발명의 범위가 이에 제한되는 것은 아니다. 보이는 바와 같이, 일 실시예에서 캐시(150)는 브랜치 트레이스 버퍼(152)를 포함할 수 있고 예를 들어, LBR(128)로부터 수신되는 브랜치 정보를 저장할 수 있다. 예를 들어, PMU(124)는 LBR(128)로부터 브랜치 트레이스 버퍼(152)로의 브랜치 타겟 메시지의 생성 및 송신을 제어할 수 있다. 이로부터, 예를 들어, 본 발명의 실시예에 따른 트랜잭션 실행에 관한 정보의 분석을 가능하게 하는 프로파일러 최적화기(a profiler optimizer) 등을 통해 정보가 액세스될 수 있다. 또한, 실행 유닛(120)은 캐시(150)에 직접 연결될 수 있다. 도 2의 실시예에서 이 높은 수준이 도시되지만, 본 발명의 범위가 이에 제한되지 않음을 이해할 것이다.

[0029] 도 3은 본 발명의 실시예에 따른 프로세서 코어의 블록도이고 하드웨어에서 트랜잭셔널 메모리 액세스 요청을 실행할 수 있다. 코어(206)와 같은 코어를 사용하여, LBR에 대한 트랜잭션 상태 표시자는 코어의 하드웨어로부터 획득되는 정보를 사용하여 설정될 수 있다. 도 3에서 도시되는 바와 같이, 프로세서 코어(206)는 코어(206)에 의한 실행을 위한 명령을 페치하기 위해 페치 유닛(202)을 포함할 수 있다. 코어(206)는 또한 페치된 명령을 디코딩하는 디코드 유닛(204)을 포함할 수 있다. 예를 들어, 디코드 유닛(204)은 복수의 uops로 페치된 명령을 디코딩할 수 있다.

[0030] 추가적으로, 코어(206)는 스케쥴 유닛(207)을 포함할 수 있다. 명령이 디스패치를 위해 대기할 때까지, 예를 들어, 디코딩된 명령의 모든 소스 값이 사용가능하게 될 때까지, 스케쥴 유닛(207)은 디코딩된 명령(예를 들어, 디코드 유닛(204)으로부터 수신됨)을 저장하는 것과 연관된 다양한 동작을 수행할 수 있다. 일 실시예에서, 스케쥴 유닛(207)은 실행을 위하여 디코딩된 명령을 하나 이상의 실행 유닛(208)에 스케쥴링하고/또는 발행(또는 디스패치)할 수 있다. 실행 유닛(208)은 메모리 실행 유닛, 정수 실행 유닛, 부동 소수점 실행 유닛, 또는 다른

실행 유닛을 포함할 수 있다. 퇴거 유닛(210)은 이들이 커미트된 이후에 실행된 명령을 퇴거할 수 있다. 실시예에서, 실행된 명령의 퇴거는 명령의 실행으로부터 커미트되는 프로세서 상태, 명령에 의해 사용되는 물리적 레지스터의 할당해제(de-allocated) 등을 초래할 수 있다.

[0031] 메모리 순서 버퍼(a memory order buffer;MOB)(218)는 로딩되지 않았거나 주메모리에 다시 기록되지 않은 계류중 메모리 연산(pending memory operation)을 저장하는 로직, 로드 버퍼, 및 저장 버퍼를 포함할 수 있다. 다양한 실시예에서, 코어는 로컬 캐시, 예를 들어, 캐시(216)와 같은 전용 캐시(a private cache)를 포함할 수 있고 이는 하나 이상의 캐시 라인(224)(예를 들어, 캐시 라인 0 내지 W이고 캐시 로직(239)에 의해 관리됨)을 포함할 수 있다. 실시예에서, 캐시(216)의 각 라인은 코어(206) 상에서 실행중인 각 스레드에 대한 트랜잭션 판독 비트(226) 및/또는 트랜잭션 기록 비트(228)를 포함할 수 있다. 비트(226 및 228)는 트랜잭셔널 메모리 액세스 요청에 의해 대응하는 캐시 라인에 대한 액세스를 나타내도록(로드 및/또는 저장) 설정되거나 클리어될 수 있다. 도 3의 실시예에서, 각 캐시 라인(224)이 비트(226 및 228) 각각을 갖는 것으로서 도시되나, 다른 구성이 가능함에 유의해야한다. 예를 들어, 트랜잭션 판독 비트(226)(또는 트랜잭션 기록 비트(228))는 캐시 블록 또는 캐시(216)의 다른 부분과 같은, 캐시(216)의 선택 부분에 대응할 수 있다. 또한, 비트(226 및/또는 228)는 캐시(216) 외의 위치에 저장될 수 있다. 따라서, 이러한 하드웨어의 정보는 LBR에 대한 다양한 표시자를 설정하는데 사용될 수 있다.

[0032] TM 동작을 실행하는 것을 돕기 위해, 코어(206)는 커미트되지 않는 상태로 남아있는 트랜잭셔널 메모리 액세스 요청의 수에 대응하는 값을 저장하기 위해 트랜잭션 깊이 카운터(a transaction depth counter)(230)를 포함할 수 있다. 예를 들어, 카운터(230)에 저장된 값은 동일한 스레드에 대응하는 다수의 트랜잭셔널 메모리 액세스 요청의 네스팅 깊이(nesting depth)를 나타낼 수 있다. 일 예시에서, 다수의 트랜잭셔널 메모리 액세스 요청은 하나의 트랜잭션이 계류중인 트랜잭션(예를 들어, 라이브러리 호출 또는 다른 네스팅된 공정을 통해)내부에서 초기화될 때를 초래할 수 있다. 카운터(230)는 하드웨어 레지스터와 같은 임의의 타입의 스토리지 디바이스 또는 메모리(예를 들어, 시스템 메모리 또는 캐시(216))에 저장된 변수로서 구현될 수 있다. 코어(206)는 또한 카운터(230)에 저장된 값을 업데이트하기 위한 트랜잭션 깊이 카운터 로직(232)을 포함할 수 있다. 코어(206)는 예를 들어, 주어진 트랜잭션의 중단시에 코어(206)의 다양한 컴포넌트의 상태를 포인트 체크(또는 저장)하는 트랜잭션 체크 포인팅 로직(234) 및 코어(206)의 다양한 컴포넌트의 상태를 복구하는 트랜잭션 복구 로직(restore logic)(236)을 포함할 수 있다. 추가적으로, 코어(206)는 트랜잭셔널 상태 및 제어 레지스터(TXSR), 트랜잭션 명령 포인터(TXIP)(예를 들어, 대응하는 트랜잭션 시작(또는 즉시 진행)에서 명령에 대한 명령 포인터가 될 수 있음), 및/또는 트랜잭션 스택 포인터(TXSP)(예를 들어, 코어(206)의 하나 이상의 컴포넌트의 다양한 상태를 저장하는 스택의 헤드에 대한 스택 포인터)와 같은 다양한 트랜잭셔널 메모리 액세스 요청에 대응하는 하나 이상의 추가적인 레지스터(240)를 포함할 수 있다. 또한, 브랜치에 대한 트랜잭션 정보의 캡쳐를 가능하게 하기 위해, MSR(250)이 또한 존재할 수 있고, 이는 브랜치 정보 외에, 트랜잭션 존재 표시자 및 트랜잭션 중단 표시자와 같은 트랜잭션 상태 정보를 저장할 수 있다.

[0033] 이제 도 4를 참조하면, 본 발명의 일 실시예에 따른 방법의 흐름도가 도시된다. 도 4의 실시예에서, 방법(300)은 본 발명의 실시예에 따른 LBR 레코더 레지스터에 정보를 지정(populate)하는데 사용될 수 있다. 예를 들어, 정보(300)는 PMU 등의 로직과 같은 프로세서 로직을 사용하여 구현될 수 있다. 보이는 바와 같이, 방법(300)은 중단이 프로그램 실행중 발생하는지를 판정함으로써 시작할 수 있다(마름모꼴 310). 이 중단의 판정은 인트랜잭션 내부 상태(in-transaction internal state)의 분석에 의해 발생할 수 있고, 이는 트랜잭션이 트랜잭션 종료 표시(an end-of-transaction marker)(명령이 될 수 있음)를 통과시킬 때 또는 트랜잭션이 중단할 때 거짓(false)으로 설정한다. 중단은 상태가 아니며, 현재 실행 스트림이 있는 어느 곳에서부터 중단 핸들러 위치까지의 점프를 야기하는 이벤트임에 유의해야한다.

[0034] 중단이 마름모꼴 310에서 발생한 것으로 판정되면, 제어가 블록 320으로 전달되어, 여기서 다음 LBR 엔트리가 할당될 수 있다. 위에서 논의된 바와 같이, 상이한 타입의 브랜치 레코드 엔트리가 가능할 수 있다. 예를 들어, 단일 레지스터는 목적지 정보만을 포함할 수 있고, 레지스터 쌍은 소스 및 목적지 정보 등을 저장할 수 있다. 따라서, 일부 실시예에서, 다수의 브랜치 어드레스가 이 엔트리에 추가될 수 있다. 구체적으로, 소스 및 목적지 어드레스가 저장될 수 있다(여기서 목적지 어드레스는 중단 핸들러에 대한 것이 될 수 있음에 유의한다). 또한 블록(330)에서 도시된 바와 같이, 이 엔트리에 대한 트랜잭션 및 중단 표시자가 또한 설정될 수 있어서, 계류중인 트랜잭션의 중단이 발생함을 나타낼 수 있다.

[0035] 그 다음 제어는 블록(340)으로 전달되고, 여기서 다음 실행되는 명령이 취해질 수 있다. 보이는 바와 같이, 상술된 바와 같이, 이는 방법(330)이 중단의 종료와 관련하여 다시 시작하도록 야기한다. 대신에 이번에는, 마름

모꼴 310에서 중단이 발생하지 않음이 판정되었음을 가정하며, 브랜치가 발생하는지가 판정될 수 있는 마름모꼴 350로 제어가 대신 전달된다. 위에서 논의된 바와 같이, 이러한 브랜치는 점프, 호출 등과 같은, 주어진 명령 타입에 대해 발생할 수 있다. 브랜치가 발생하지 않으면, 방법은 이 명령과 관련하여 종료하며 제어는 다음 실행되는 명령에 대한 블록 340으로 전달된다.

[0036] 여전히 도 4를 참조하면, 위와 달리, 주어진 명령이 브랜치 명령임이 판정되면, 다양한 정보가 저장될 수 있다. 특히, 도 4에서 도시되는 바와 같이, 블록 360에서 다음 LBR 엔트리가 할당될 수 있고 브랜치 어드레스 정보가 엔트리에 저장된다.

[0037] 그 다음, 브랜치가 트랜잭션 동안 발생하는지가 판정될 수 있다(마름모꼴 370). 예를 들어, 일 실시예에서, 트랜잭션의 시작을 의미하는 명령이 실행될 때 내부 프로세서 상태 "인트랜잭션"은 참(true)으로 설정될 수 있다. 그렇지 않다면, 제어는 상술된 블록 340으로 전달된다(또한 여기서 트랜잭션 및 중단 표시자 양쪽 모두가 클리어될 수 있음을 이해해야한다).

[0038] 대신에 마름모꼴 370에서 브랜치가 트랜잭션의 실행 동안 발생한다고 판정되면, LBR 엔트리의 트랜잭션 표시자가 트랜잭션 실행 동안 브랜치가 발생됨을 나타내도록 설정될 수 있는 블록 380으로 제어가 전달된다. 또한 제어가 상술된, 블록 340으로 전달된다. 도 4의 실시예에서 이 특정 구현이 도시되었지만, 본 발명의 범위가 이에 제한되는 것은 아님을 이해해야한다.

[0039] 이제 도 5를 참조하면, 본 발명의 실시예에 따른 브랜치 레코드를 사용하기 위한 방법의 흐름도가 도시된다. 도 5에서 도시된 바와 같이, 방법(400)은 디버거, 분석기, 최적화기 등과 같은 다양한 로직에 의해 구현될 수 있다. 방법(400)은 브랜치 트레이스 버퍼로 최종 브랜치 레코드를 저장함으로써 시작할 수 있다(블록 410). 예를 들어, 주어진 구현에서, 제한된 수의 MSR 쌍은 최종 브랜치 레코드를 저장하도록 존재할 수 있다. 따라서, 이들 쌍이 채워지게 될 때, 제어 레지스터 등의 제어 비트에 응답하여, PMU 또는 전용 LBR 로직과 같은 로직은 BTM를 생성하여 이 정보를 버퍼에 송신할 수 있다. 또한, 이들 레코드는 버퍼로부터 획득될 수 있다(블록 420). 예를 들어, 최적화기 또는 다른 이러한 로직은 이들 레코드를 획득할 수 있다.

[0040] 여전히 도 5를 참조하면, 레코드의 정보에 기초하여, 하나 이상의 트랜잭션의 상황 내에서 발생하는 브랜치가 트랜잭션 표시자에 기초하여 식별될 수 있다. 예를 들어, 제 1 엔트리가 클리어된 트랜잭션 표시자(a cleared transaction indicator)를 갖고 후속하는 엔트리가 설정된 트랜잭션 표시자(a set transaction indicator)를 갖는다고 가정하면, 이는 이 제 2 엔트리가 트랜잭션 영역 내부의 제 1 브랜치임을 의미한다. 대신에 제 1 엔트리가 설정된 트랜잭션 표시자를 갖고 제 2 엔트리가 클리어 트랜잭션 표시자를 갖는다고 가정하면, 이는 제 1 엔트리가 트랜잭션 영역 내부의 최종 브랜치임을 의미한다. 또한 다시 분석 하의 레코드 세트의 모든 엔트리가 클리어된 트랜잭션 표시자를 갖는다고 가정하면, 이는 트랜잭션 실행 동안 발생되는 브랜치가 존재하지 않음을 나타내고, 대신에 반면 모든 이러한 트랜잭션 표시자가 설정된다면, 이는 모든 브랜치가 TM 트랜잭션의 상황에서 발생됨을 나타낸다.

[0041] 여전히 도 5를 참조하면, 그 다음 블록 440에서, 중단된 트랜잭션 영역의 부분으로 확인되는 임의의 브랜치가 식별될 수 있다. 특히, 트랜잭션 표시자 및 중단 표시자의 조합에 기초하여, 이러한 정보가 판정될 수 있다. 특히, 제 1 엔트리가 이의 중단 표시자 세트를 갖는다는 것은 중단이 이 LBR 엔트리 이전에 발생됨을 나타낸다. 따라서, 이 정보로부터, 브랜치 백 트레이스가 생성될 수 있다. 예를 들어, 코드를 최적화하기 위해 이 정보가 사용될 수 있다(블록 450). 트랜잭션이 중단되도록 야기하는 브랜치를 코드 섹션이 포함하다고 가정하면, 최적화기는 이러한 브랜치가 취해지지 않도록 이 코드가 수정되는 것을 야기할 수 있거나 코드 세그먼트의 추후 실행에서 트랜잭션 중단을 회피하도록 브랜치를 수정하는 것을 야기할 수 있다. 물론, 디버거 툴 내부에서 트랜잭션이 중단되는지, 그리고 어느 명령에서 중단되는지를 이해하는 것과 같이 이 트랜잭션 상태 정보의 다른 사용이 가능하다.

[0042] 트랜잭션의 특성에 기인하여, 중단 시에 어떠한 구조적 가시 상태도 외부 세계에 노출되지 않는다. 이 트랜잭션 정보는 중단 위치까지의 그외의 비가시적인 미지의 프로그램 플로우의 재구조화를 허용한다. 따라서 실시예는 LBR 레지스터에 저장된 정보의 사용이 실행 백 트레이스의 생성에서 사용되는 것을 가능하게 한다. 이 실행 백 트레이스는 또한 온라인 및 오프라인에서 사용될 수 있다. 예를 들어, 실행 백 트레이스는 TM 트랜잭션 내부에서의 코드 플로우의 검사를 가능하게 하는 디버거 툴에 의해 사용될 수 있다. 또는 오프라인 분석 툴은 사후 분석 코드 플로우 검사(a post mortem code flow inspection)을 가능하게 하는 실행 백 트레이스를 사용할 수 있다. 도 5의 실시예에서 이 특정 구현이 도시되었지만, 본 발명의 범위는 이에 제한되는 것이 아님을 이해해야한다.

[0043] 실시예는 많은 상이한 시스템 타입에서 구현될 수 있다. 이제 도 6을 참조하면, 본 발명의 실시예에 따른 시스템의 블록도가 도시된다. 도 6에서 도시된 바와 같이, 멀티프로세서 시스템(500)은 포인트 투 포인트 상호접속 시스템(point-to-point interconnect system)이고, 포인트 투 포인트 상호접속부(550)를 통해 연결되는 제 1 프로세서(570)와 제 2 프로세서(580)를 포함한다. 도 6에서 도시된 바와 같이, 프로세서(570 및 580)의 각각은 멀티코어 프로세서가 될 수 있고, 제 1 및 제 2 프로세서 코어(프로세서 코어(574a 및 574b) 및 프로세서 코어(584a 및 584b))를 포함하지만, 잠재적으로 더 많은 코어가 프로세서에 존재할 수 있다. 이러한 코어는 TM 트랜잭션 내에서 발생하는 브랜치의 분석을 가능하게 하는 본 발명의 실시예에 따른 LBR 레지스터를 포함할 수 있다.

[0044] 계속해서 도 6을 참조하면, 제 1 프로세서(570)는 메모리 제어기 허브(a memory controller hub;MCH)(572) 및 포인트 투 포인트(P-P) 인터페이스(576 및 588)를 더 포함한다. 유사하게, 제 2 프로세서(580)는 MCH(582) 및 P-P 인터페이스(586 및 588)를 포함한다. 도 6에서 도시된 바와 같이, MCH(572 및 582)는 프로세서를 각각의 메모리, 즉, 메모리(532) 및 메모리(534)에 연결하고, 이는 각각의 프로세서에 국부적으로 부착된 주메모리(예를 들어, 동적 랜덤 액세스 메모리(DRAM))의 부분이 될 수 있다. 제 1 프로세서(570) 및 제 2 프로세서(580)는 P-P 상호접속부(552 및 554)를 통해 각각 칩셋(590)에 연결될 수 있다. 도 6에서 도시된 바와 같이, 칩셋(590)은 P-P 인터페이스(594 및 598)를 포함한다.

[0045] 또한, 칩셋(590)은 P-P 상호접속부(539)에 의해 고성능 그래픽 엔진(538)과 칩셋(590)을 연결하는 인터페이스(592)를 포함한다. 다음으로, 칩셋(590)은 인터페이스(596)를 통해 제 1 버스(516)에 연결될 수 있다. 도 6에서 도시된 바와 같이, 다양한 입력/출력(I/O) 디바이스(514)는 제 1 버스(516)를 제 2 버스(520)에 연결하는 버스 브릿지(518)와 함께 제 1 버스(516)에 연결될 수 있다. 일 실시예에서, 예를 들어, 키보드/마우스(522), 통신 디바이스(526), 및 코드(530)를 포함할 수 있는 디스크 드라이브 또는 다른 대용량 스토리지 디바이스와 같은 데이터 스토리지 유닛(528)을 포함하는 다양한 디바이스가 제 2 버스(520)에 연결될 수 있다. 또한, 오디오 I/O(524)는 제 2 버스(520)에 연결될 수 있다.

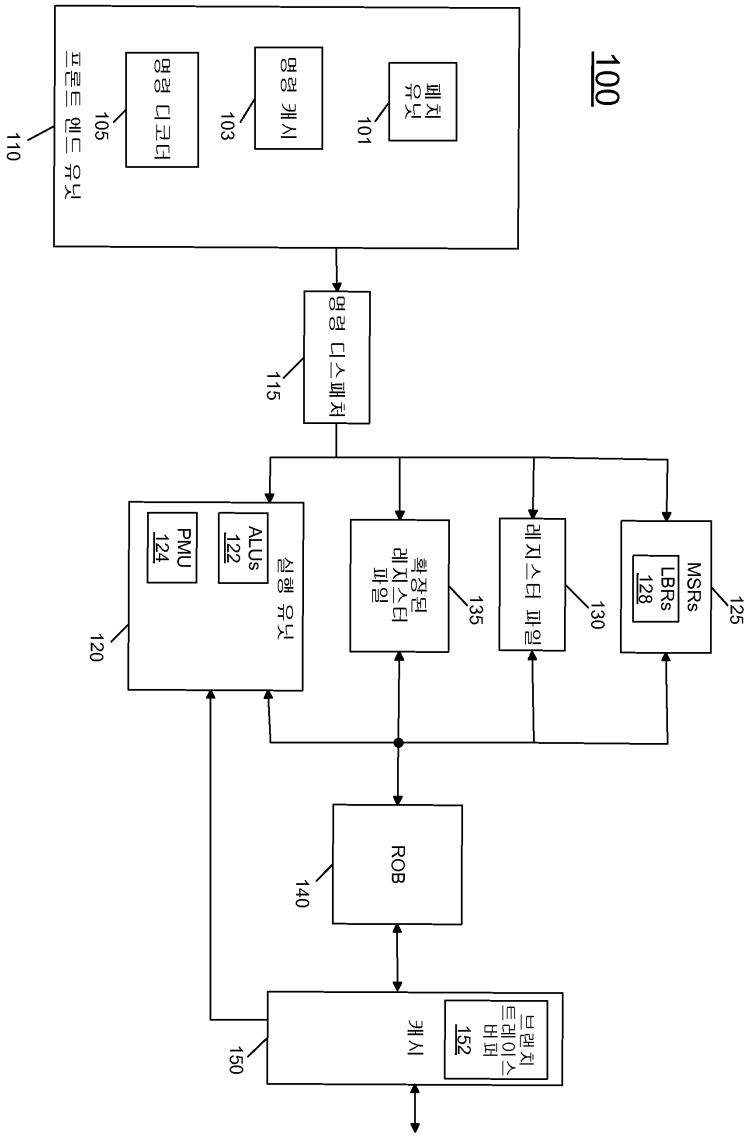
[0046] 실시예는 코드로 구현될 수 있고 저장 매체 상에 저장될 수 있으며 여기에 저장된 명령을 갖고 이는 명령을 수행하는 시스템을 프로그래밍하는데 사용될 수 있다. 저장 매체는 플로피 디스크, 광학 디스크, 솔리드 스테이트 드라이브(SDD), 컴팩트 디스크 판독전용 메모리(CD-ROM), 재기록 가능 컴팩트 디스크 (CD-RW), 및 자기광학 디스크, 판독전용 메모리(ROM)와 같은 반도체 디바이스, 동적 랜덤 액세스 메모리(DRAM), 정적 랜덤 액세스 메모리(SRAM), 삭제가능 프로그램가능 판독전용 메모리(EPROM)와 같은 랜덤 액세스 메모리(RAM), 플래시 메모리, 전기적으로 삭제가능한 프로그램가능 판독전용 메모리(EEPROM), 자기 또는 광학 카드, 또는 전자 명령을 저장하기에 적합한 임의의 다른 타입의 매체를 포함할 수 있지만 이에 제한되지 않는다.

[0047] 본 발명은 제한된 수의 실시예와 관련하여 설명되었지만, 당업자는 이로부터의 다양한 수정 및 변형을 이해할 것이다. 첨부된 청구항은 본 발명의 참 사상 및 범위 내에 속하는 것으로서 모든 이러한 수정 및 변형을 커버하도록 의도된다.

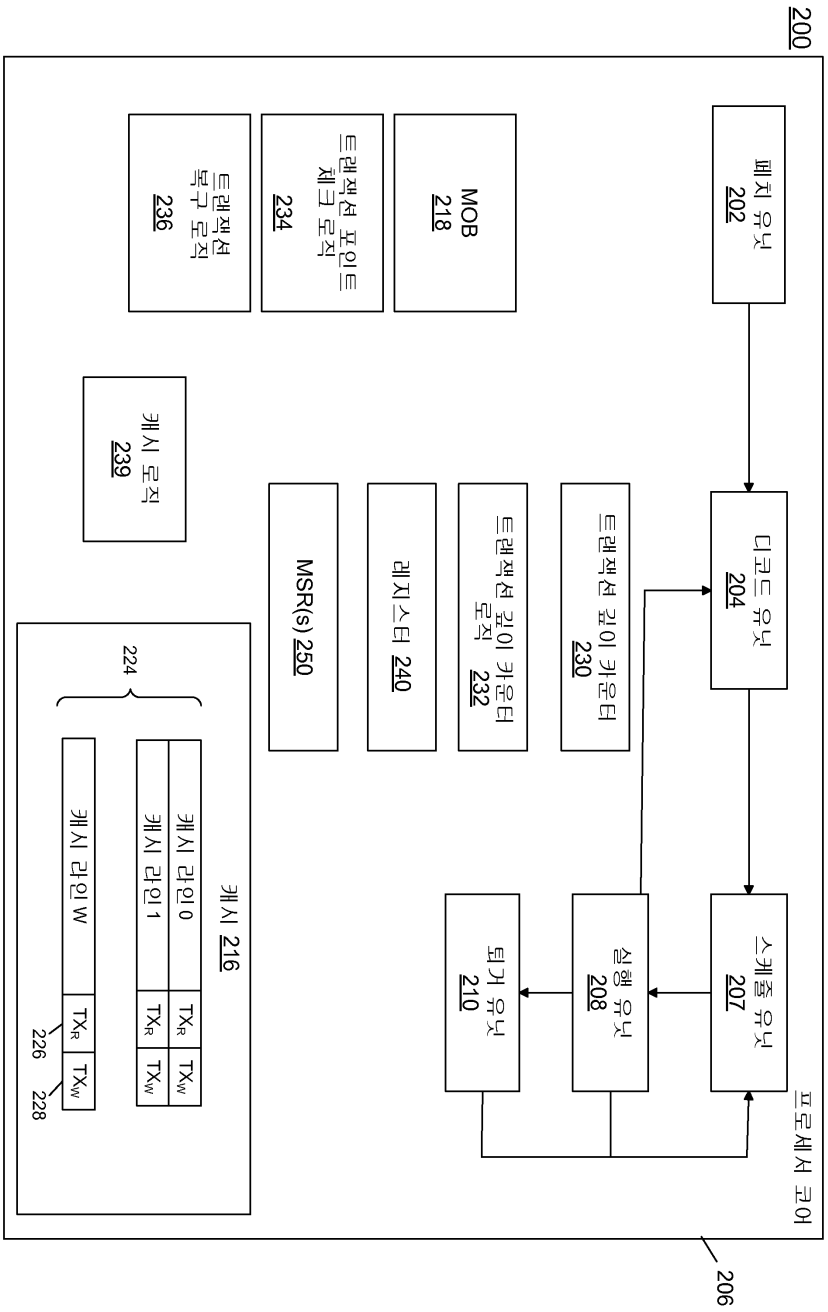
도면

도면1

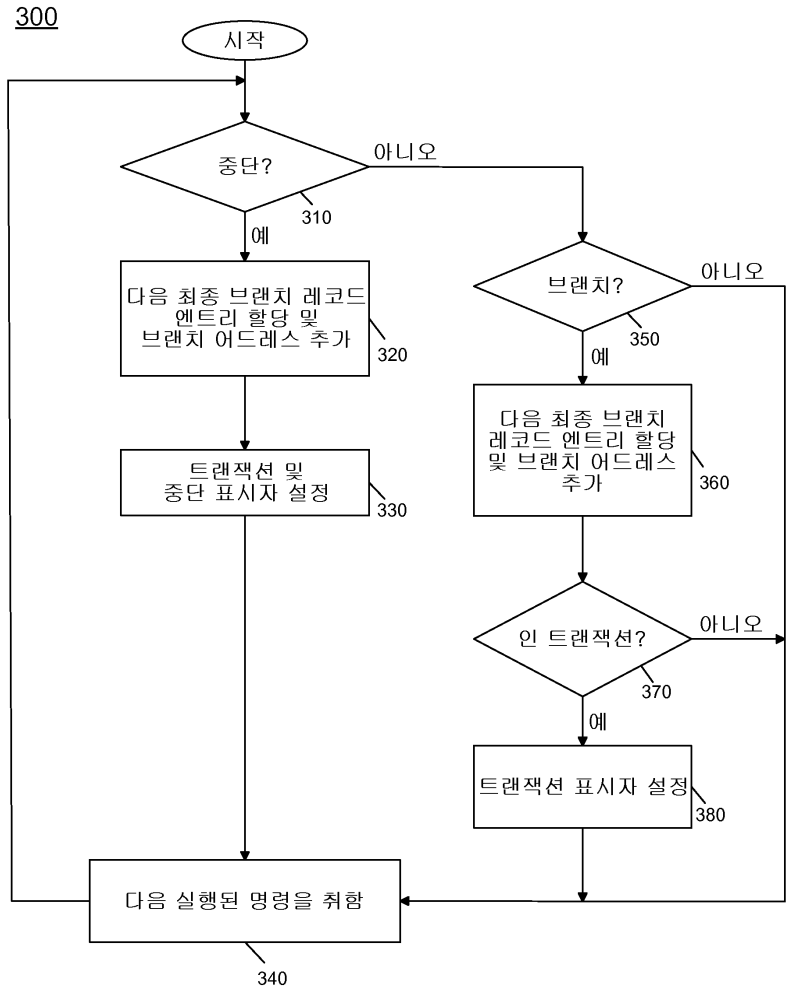
도면2


100
프론트 엔드 유닛
110
명령 디코더
105
명령 캐시
103
페치 유닛
101
명령 디스패처
115
실행 유닛
120
PMU 124
ALUs 122
확장된 레지스터 파일
135
레지스터 파일
130
MSRs
LBRs 128
125
ROB
140
캐시
150
브랜치 트레이스 버퍼 152

도면3

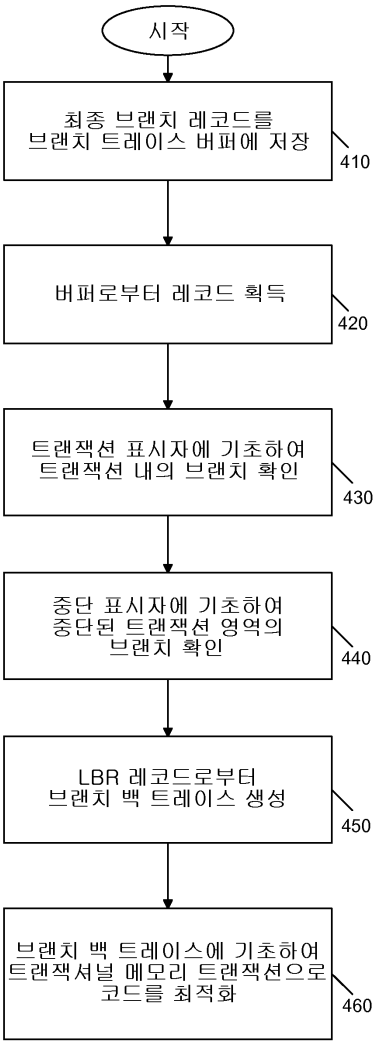

200
프로세서 코어
206
페치 유닛 202
디코드 유닛 204
스케줄 유닛 207
실행 유닛 208
퇴거 유닛 210
트랜잭션 깊이 카운터 230
트랜잭션 깊이 카운터 로직 232
레지스터 240
MSR(s) 250
캐시 로직 239
MOB 218
트랜잭션 포인트 체크 로직 234
트랜잭션 복구 로직 236
캐시 216
224
캐시 라인 0
캐시 라인 1
캐시 라인 W
TXR
TXW
226
228

도면4

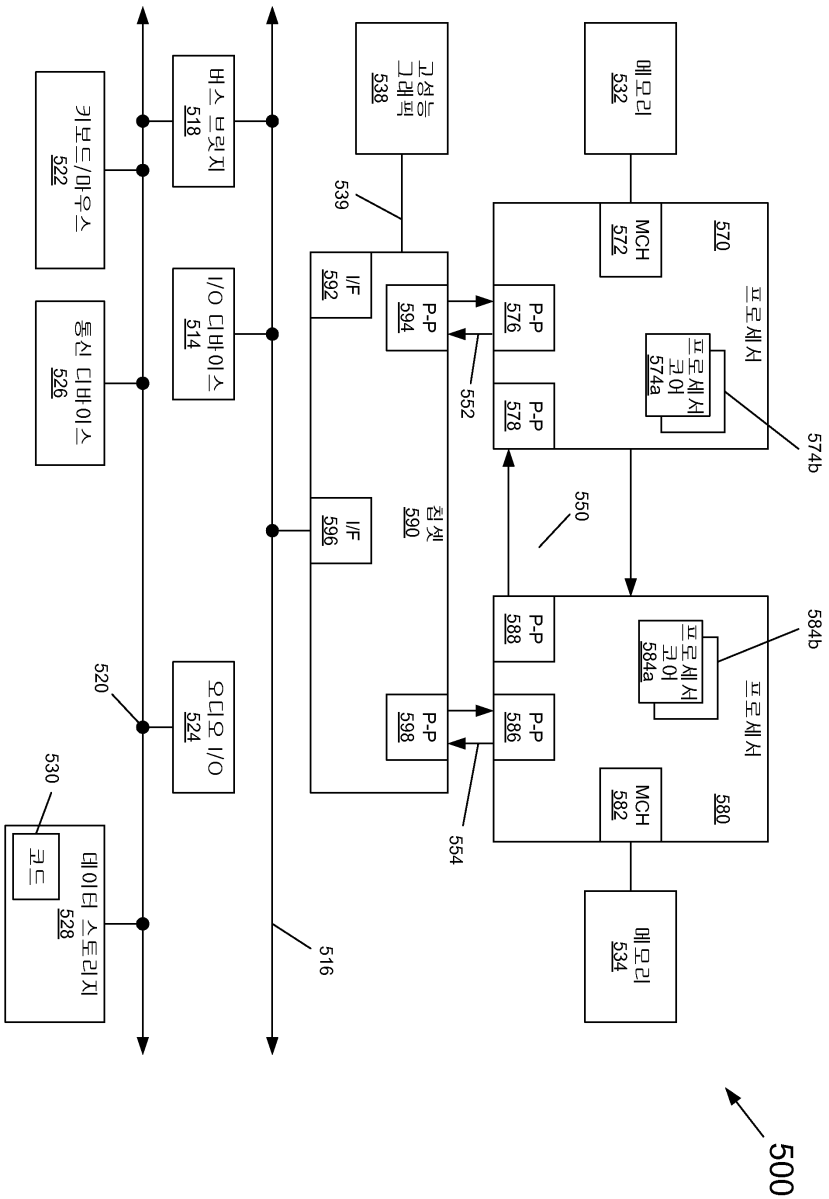

300
시작
중단?
310
아니오
예
다음 최종 브랜치 레코드 엔트리 할당 및 브랜치 어드레스 추가
320
트랜잭션 및 중단 표시자 설정
330
브랜치?
350
아니오
예
다음 최종 브랜치 레코드 엔트리 할당 및 브랜치 어드레스 추가
360
인 트랜잭션?
370
아니오
예
트랜잭션 표시자 설정
380
다음 실행된 명령을 취함
340

도면5

400


시작
최종 브랜치 레코드를 브랜치 트레이스 버퍼에 저장
410
버퍼로부터 레코드 획득
420
트랜잭션 표시자에 기초하여 트랜잭션 내의 브랜치 확인
430
중단 표시자에 기초하여 중단된 트랜잭션 영역의 브랜치 확인
440
LBR 레코드로부터 브랜치 백 트레이스 생성
450
브랜치 백 트레이스에 기초하여 트랜잭셔널 메모리 트랜잭션으로 코드를 최적화
460

도면6


500
메모리 532
MCH 572
570
프로세서
프로세서 코어 574a
574b
P-P 576
P-P 578
552
고성능 그래픽 538
539
I/F 592
P-P 594
칩셋 590
I/F 596
P-P 598
550
554
P-P 588
P-P 586
프로세서 코어 584a
584b
프로세서 580
MCH 582
메모리 534
버스 브릿지 518
I/O 디바이스 514
오디오 I/O 524
520
516
키보드/마우스 522
통신 디바이스 526
데이터 스토리지 528
코드 530