(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2024년11월07일
(11) 등록번호 10-2727263
(24) 등록일자 2024년11월04일

(51) 국제특허분류(Int. Cl.)
G06F 9/38 (2024.01) *G06F 12/0875* (2016.01)
G06F 12/14 (2006.01) *G06F 21/71* (2013.01)
G06F 9/30 (2018.01)
(52) CPC특허분류
G06F 9/3834 (2023.08)
G06F 12/0875 (2013.01)
(21) 출원번호 10-2020-7020303
(22) 출원일자(국제) 2018년12월14일
심사청구일자 2021년11월30일
(85) 번역문제출일자 2020년07월14일
(65) 공개번호 10-2020-0101943
(43) 공개일자 2020년08월28일
(86) 국제출원번호 PCT/GB2018/053636
(87) 국제공개번호 WO 2019/135063
국제공개일자 2019년07월11일
(30) 우선권주장
62/613,280 2018년01월03일 미국(US)
16/208,701 2018년12월04일 미국(US)
(56) 선행기술조사문헌
인용참증 1 1부.*
KR1020140011940 A
KR1020150138306 A
KR1020140113444 A
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
에이알엠 리미티드
영국 캠브리지 씨비1 9엔제이 체리힌톤 풀번로드 110
(72) 발명자
그리센드와이트 리차드 로이
영국 캠브리지 씨비1 9엔제이 체리힌톤 풀번로드 110 에이알엠 리미티드
가브리엘리 지아코모
영국 캠브리지 씨비1 9엔제이 체리힌톤 풀번로드 110 에이알엠 리미티드
호스넬 매튜 제임스
영국 캠브리지 씨비1 9엔제이 체리힌톤 풀번로드 110 에이알엠 리미티드
(74) 대리인
이화익

전체 청구항 수 : 총 7 항 심사관 : 김평수

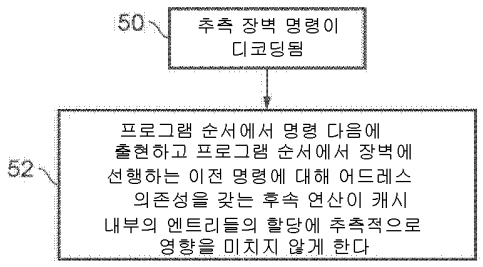
(54) 발명의 명칭 **추측 장벽 명령**

(57) 요 약

장치는 데이터 처리를 행하는 처리회로와 명령들을 디코드하고 처리회로를 제어하여 데이터 처리를 행하게 하는 명령 디코딩 회로를 구비한다. 명령 디코딩 회로는, 추측 장벽 명령에 응답하여 처리회로를 제어함으로써, 프로그램 순서에서 추측 장벽 명령 이후에 출현하고 프로그램 순서에서 추측 장벽 명령에 선행하는 이전 명

(뒷면에 계속)

대 표 도 - 도3



령에 대한 어드레스 의존성을 갖는 후속 연산이 캐시 내부의 엔트리들의 할당에 추측적으로 영향을 미치지 않게 한다. 이것은 추측적 캐시 타이밍 부채널 공격에 대한 보호를 제공한다.

(52) CPC특허분류
G06F 12/1491 (2013.01)
G06F 21/71 (2013.01)
G06F 9/30058 (2023.08)
G06F 9/30087 (2013.01)
G06F 9/3842 (2013.01)
G06F 2212/1052 (2013.01)

명 세 서

청구범위

청구항 1

데이터 처리를 행하는 처리회로와,

명령들을 디코드하고 상기 처리회로를 제어하여 데이터 처리를 행하게 하는 명령 디코딩 회로를 구비하고,

상기 명령 디코딩 회로는, 추측 장벽(speculation barrier) 명령에 응답하여 상기 처리회로를 제어함으로써, 프로그램 순서에서 상기 추측 장벽 명령 이후에 출현하고 상기 프로그램 순서에서 상기 추측 장벽 명령에 선행하는 이전 명령에 대한 어드레스 의존성을 갖는 후속 연산이 캐시 내부의 엔트리들의 할당에 추측적으로 영향을 미치지 않게 하고,

상기 추측 장벽 명령이 완료할 때까지,

로드, 스토어, 데이터 또는 명령 프리로드 명령 RW2가 프로그램 순서에서 상기 추측 장벽 명령 이후에 출현하고 조건부 선택 명령의 결과에 대한 어드레스 의존성을 가질 때,

상기 조건부 선택 명령이 그것의 입력 레지스터들 중에서 한 개에 대해 추측적으로 실행된 로드 명령 R1에 대한 레지스터 데이터 의존성을 갖고,

상기 조건부 선택 명령이 그것의 나머지 입력 레지스터에 대해 R1에 대한 레지스터 데이터 의존성을 갖지 않고,

상기 조건부 선택 명령에 대한 조건이, R1이 아키텍처 상 실행되지 않으면 R1에 의존하지 않는 입력 레지스터가 선택되도록 하는 것인 경우,

상기 처리회로는, 상기 캐시 내부의 어떤 엔트리들이 할당되었는지 또는 퇴출되었는지의 평가에 의해 R1으로부터 추측적으로 로드된 값의 일부를 결정하기 위해 사용될 수 있도록 RW2가 상기 캐시 내부의 엔트리들의 할당에 영향을 미치지 않게 하도록 구성된 장치.

청구항 2

삭제

청구항 3

삭제

청구항 4

삭제

청구항 5

데이터 처리를 행하는 처리회로와,

명령들을 디코드하고 상기 처리회로를 제어하여 데이터 처리를 행하게 하는 명령 디코딩 회로를 구비하고,

상기 명령 디코딩 회로는, 추측 장벽(speculation barrier) 명령에 응답하여 상기 처리회로를 제어함으로써, 프로그램 순서에서 상기 추측 장벽 명령 이후에 출현하고 상기 프로그램 순서에서 상기 추측 장벽 명령에 선행하는 이전 명령에 대한 어드레스 의존성을 갖는 후속 연산이 캐시 내부의 엔트리들의 할당에 추측적으로 영향을 미치지 않게 하고,

상기 추측 장벽 명령이 완료할 때까지,

로드, 스토어, 데이터 또는 명령 프리로드 명령 RW2가 프로그램 순서에서 상기 추측 장벽 명령 이후에 출현하고 조건부 이동 명령의 결과에 대한 어드레스 의존성을 가질 때,

상기 조건부 이동 명령이 그것의 입력 레지스터들 중에서 한 개에 대해 추측적으로 실행된 로드 명령 R1에 대한 레지스터 의존성을 갖지 않고,

상기 조건부 이동 명령에 대한 조건이, R1이 아키텍처 상 실행되지 않으면 조건을 통과하도록 하는 것인 경우,

상기 처리회로는, 상기 캐시 내부의 어떤 엔트리들이 할당되었는지 또는 퇴출되었는지의 평가에 의해 R1으로부터 추측적으로 로드된 값의 일부를 결정하기 위해 사용될 수 있도록 RW2가 상기 캐시 내부의 엔트리들의 할당에 영향을 미치지 않게 하도록 구성된 장치.

청구항 6

제 1항 또는 제 5항에 있어서,

상기 캐시는,

데이터 캐시,

명령 캐시, 및

분기 예측 캐시 중에서 한 개인 장치.

청구항 7

호스트 처리장치를 제어하여 타겟 프로그램 코드의 명령들을 실행하기 위한 명령 실행 환경을 제공하기 위한, 컴퓨터 판독 가능한 기억 매체에 기억된, 컴퓨터 프로그램으로서,

상기 타겟 프로그램 코드의 명령들을 디코드하여 처리 프로그램 로직을 제어함으로써 데이터 처리를 행하게 하는 명령 디코딩 프로그램 로직을 포함하고,

상기 명령 디코딩 프로그램 로직은 추측 장벽 명령에 응답하여, 프로그램 순서에서 상기 추측 장벽 명령 이후에 출현하고 상기 프로그램 순서에서 상기 추측 장벽 명령에 선행하는 이전 명령에 대한 어드레스 의존성을 갖는 후속 연산이 캐시 내부의 엔트리들의 할당에 추측적으로 영향을 미치지 않게 하고,

상기 추측 장벽 명령이 완료할 때까지,

로드, 스토어, 데이터 또는 명령 프리로드 명령 RW2가 프로그램 순서에서 상기 추측 장벽 명령 이후에 출현하고 조건부 선택 명령의 결과에 대한 어드레스 의존성을 가질 때,

상기 조건부 선택 명령이 그것의 입력 레지스터들 중에서 한 개에 대해 추측적으로 실행된 로드 명령 R1에 대한 레지스터 데이터 의존성을 갖고,

상기 조건부 선택 명령이 그것의 나머지 입력 레지스터에 대해 R1에 대한 레지스터 데이터 의존성을 갖지 않고,

상기 조건부 선택 명령에 대한 조건이, R1이 아키텍처 상 실행되지 않으면 R1에 의존하지 않는 입력 레지스터가 선택되도록 하는 것인 경우,

상기 명령 디코딩 프로그램 로직은, 상기 캐시 내부의 어떤 엔트리들이 할당되었는지 또는 퇴출되었는지의 평가에 의해 R1으로부터 추측적으로 로드된 값의 일부를 결정하기 위해 사용될 수 있도록 RW2가 상기 캐시 내부의 엔트리들의 할당에 영향을 미치지 않게 하도록 구성된 컴퓨터 프로그램.

청구항 8

데이터 처리방법으로서,

추측 장벽 명령을 디코드하는 단계와,

상기 추측 장벽 명령의 디코딩에 응답하여, 처리회로를 제어함으로써, 프로그램 순서에서 상기 추측 장벽 명령 이후에 출현하고 상기 프로그램 순서에서 상기 추측 장벽 명령에 선행하는 이전 명령에 대한 어드레스 의존성을

갖는 후속 연산이 캐시 내부의 엔트리들의 할당에 추측적으로 영향을 미치지 않게 하는 단계를 포함하고,

상기 추측 장벽 명령이 완료할 때까지,

로드, 스토어, 데이터 또는 명령 프리로드 명령 RW2가 프로그램 순서에서 상기 추측 장벽 명령 이후에 출현하고 조건부 선택 명령의 결과에 대한 어드레스 의존성을 가질 때,

상기 조건부 선택 명령이 그것의 입력 레지스터들 중에서 한 개에 대해 추측적으로 실행된 로드 명령 R1에 대한 레지스터 데이터 의존성을 갖고,

상기 조건부 선택 명령이 그것의 나머지 입력 레지스터에 대해 R1에 대한 레지스터 데이터 의존성을 갖지 않고,

상기 조건부 선택 명령에 대한 조건이, R1이 아키텍처 상 실행되지 않으면 R1에 의존하지 않는 입력 레지스터가 선택되도록 하는 것인 경우,

상기 데이터 처리방법은, 상기 캐시 내부의 어떤 엔트리들이 할당되었는지 또는 퇴출되었는지의 평가에 의해 R1으로부터 추측적으로 로드된 값의 일부를 결정하기 위해 사용될 수 있도록 RW2가 상기 캐시 내부의 엔트리들의 할당에 영향을 미치지 않게 하는 단계를 더 포함하는 데이터 처리방법.

청구항 9

삭제

청구항 10

삭제

청구항 11

삭제

청구항 12

삭제

청구항 13

삭제

청구항 14

삭제

청구항 15

삭제

청구항 16

삭제

청구항 17

삭제

청구항 18

삭제

청구항 19

삭제

청구항 20

호스트 처리장치를 제어하여 타겟 프로그램 코드의 명령들을 실행하기 위한 명령 실행 환경을 제공하기 위한, 컴퓨터 판독 가능한 기억 매체에 기억된, 컴퓨터 프로그램으로서,

상기 타겟 프로그램 코드의 명령들을 디코드하여 처리 프로그램 로직을 제어함으로써 데이터 처리를 행하게 하는 명령 디코딩 프로그램 로직을 포함하고,

상기 명령 디코딩 프로그램 로직은 추측 장벽 명령에 응답하여, 프로그램 순서에서 상기 추측 장벽 명령 이후에 출현하고 상기 프로그램 순서에서 상기 추측 장벽 명령에 선행하는 이전 명령에 대한 어드레스 의존성을 갖는 후속 연산이 캐시 내부의 엔트리들의 할당에 추측적으로 영향을 미치지 않게 하고,

상기 추측 장벽 명령이 완료할 때까지,

로드, 스토어, 데이터 또는 명령 프리로드 명령 RW2가 프로그램 순서에서 상기 추측 장벽 명령 이후에 출현하고 조건부 이동 명령의 결과에 대한 어드레스 의존성을 가질 때,

상기 조건부 이동 명령이 그것의 입력 레지스터들 중에서 한 개에 대해 추측적으로 실행된 로드 명령 R1에 대한 레지스터 의존성을 갖지 않고,

상기 조건부 이동 명령에 대한 조건이, R1이 아키텍처 상 실행되지 않으면 조건을 통과하도록 하는 것인 경우,

상기 명령 디코딩 프로그램 로직은, 상기 캐시 내부의 어떤 엔트리들이 할당되었는지 또는 퇴출되었는지의 평가에 의해 R1으로부터 추측적으로 로드된 값의 일부를 결정하기 위해 사용될 수 있도록 RW2가 상기 캐시 내부의 엔트리들의 할당에 영향을 미치지 않게 하도록 구성된 컴퓨터 프로그램.

청구항 21

데이터 처리방법으로서,

추측 장벽 명령을 디코드하는 단계와,

상기 추측 장벽 명령의 디코딩에 응답하여, 처리회로를 제어함으로써, 프로그램 순서에서 상기 추측 장벽 명령 이후에 출현하고 상기 프로그램 순서에서 상기 추측 장벽 명령에 선행하는 이전 명령에 대한 어드레스 의존성을 갖는 후속 연산이 캐시 내부의 엔트리들의 할당에 추측적으로 영향을 미치지 않게 하는 단계를 포함하고,

상기 추측 장벽 명령이 완료할 때까지,

로드, 스토어, 데이터 또는 명령 프리로드 명령 RW2가 프로그램 순서에서 상기 추측 장벽 명령 이후에 출현하고 조건부 이동 명령의 결과에 대한 어드레스 의존성을 가질 때,

상기 조건부 이동 명령이 그것의 입력 레지스터들 중에서 한 개에 대해 추측적으로 실행된 로드 명령 R1에 대한 레지스터 의존성을 갖지 않고,

상기 조건부 이동 명령에 대한 조건이, R1이 아키텍처 상 실행되지 않으면 조건을 통과하도록 하는 것인 경우,

상기 데이터 처리방법은, 상기 캐시 내부의 어떤 엔트리들이 할당되었는지 또는 퇴출되었는지의 평가에 의해 R1으로부터 추측적으로 로드된 값의 일부를 결정하기 위해 사용될 수 있도록 RW2가 상기 캐시 내부의 엔트리들의 할당에 영향을 미치지 않게 하는 단계를 더 포함하는 데이터 처리방법.

발명의 설명

기 술 분 야

[0001] 본 발명은 데이터 처리 분야에 관한 것이다.

배 경 기 술

[0002] 데이터 처리장치는, 명령에 대한 입력 피연산자들이 올바른지 아닌지 또는 명령이 실행될 필요가 있는지 아닌지가 알려지기 전에 명령들이 실행되는 명령들의 추측 실행을 지원한다. 예를 들어, 처리장치는, 분기의

실제 결과가 무엇인지 알려지기 전에 후속 명령들이 페치, 디코드 및 추측적으로 실행될 수 있도록, 분기 명령들의 결과를 예측하는 분기 예측기를 갖는다. 또한, 일부 시스템은, 메모리로부터 실제 값이 실제로 반환되기 전에 메모리에서 로드된 값이 예측됨으로써, 후속 명령들이 더 빠르게 처리될 수 있도록 하는 로드 추측을 지원한다. 다른 형태의 추측도 가능하다.

[0003] 적어도 일부 실시예는, 데이터 처리를 행하는 처리회로와, 명령들을 디코드하고 상기 처리회로를 제어하여 데이터 처리를 행하게 하는 명령 디코딩 회로를 구비하고, 상기 명령 디코딩 회로는, 추측 장벽(speculation barrier) 명령에 응답하여 상기 처리회로를 제어함으로써, 프로그램 순서에서 상기 추측 장벽 명령 이후에 출현하고 상기 프로그램 순서에서 상기 추측 장벽 명령에 선행하는 이전 명령에 대한 어드레스 의존성을 갖는 후속 연산이 캐시 내부의 엔트리들의 할당에 추측적으로 영향을 미치지 않게 하고, 상기 추측 장벽 명령이 완료할 때까지, 로드, 스토어, 데이터 또는 명령 프리로드 명령 RW2가 프로그램 순서에서 상기 추측 장벽 명령 이후에 출현하고 조건부 선택 명령의 결과에 대한 어드레스 의존성을 가질 때, 상기 조건부 선택 명령이 그것의 입력 레지스터들 중에서 한 개에 대해 추측적으로 실행된 로드 명령 R1에 대한 레지스터 데이터 의존성을 갖고, 상기 조건부 선택 명령이 그것의 나머지 입력 레지스터에 대해 R1에 대한 레지스터 데이터 의존성을 갖지 않고, 상기 조건부 선택 명령에 대한 조건이, R1이 아키텍처 상 실행되지 않으면 R1에 의존하지 않는 입력 레지스터가 선택되도록 하는 것인 경우, 상기 처리회로는, 상기 캐시 내부의 어떤 엔트리들이 할당되었는지 또는 퇴출되었는지의 평가에 의해 R1으로부터 추측적으로 로드된 값의 일부를 결정하기 위해 사용될 수 있도록 RW2가 상기 캐시 내부의 엔트리들의 할당에 영향을 미치지 않게 하도록 구성된 장치를 제공한다.

[0004] 적어도 일부 실시예는, 호스트 처리장치를 제어하여 타겟 프로그램 코드의 명령들을 실행하기 위한 명령 실행 환경을 제공하기 위한 컴퓨터 프로그램으로서, 상기 타겟 프로그램 코드의 명령들을 디코드하여 처리 프로그램 로직을 제어함으로써 데이터 처리를 행하게 하는 명령 디코딩 프로그램 로직을 포함하고, 상기 명령 디코딩 프로그램 로직은 추측 장벽 명령에 응답하여, 프로그램 순서에서 상기 추측 장벽 명령 이후에 출현하고 상기 프로그램 순서에서 상기 추측 장벽 명령에 선행하는 이전 명령에 대한 어드레스 의존성을 갖는 후속 연산이 캐시 내부의 엔트리들의 할당에 추측적으로 영향을 미치지 않게 하고, 상기 추측 장벽 명령이 완료할 때까지, 로드, 스토어, 데이터 또는 명령 프리로드 명령 RW2가 프로그램 순서에서 상기 추측 장벽 명령 이후에 출현하고 조건부 선택 명령의 결과에 대한 어드레스 의존성을 가질 때, 상기 조건부 선택 명령이 그것의 입력 레지스터들 중에서 한 개에 대해 추측적으로 실행된 로드 명령 R1에 대한 레지스터 데이터 의존성을 갖고, 상기 조건부 선택 명령이 그것의 나머지 입력 레지스터에 대해 R1에 대한 레지스터 데이터 의존성을 갖지 않고, 상기 조건부 선택 명령에 대한 조건이, R1이 아키텍처 상 실행되지 않으면 R1에 의존하지 않는 입력 레지스터가 선택되도록 하는 것인 경우, 상기 명령 디코딩 프로그램 로직은, 상기 캐시 내부의 어떤 엔트리들이 할당되었는지 또는 퇴출되었는지의 평가에 의해 R1으로부터 추측적으로 로드된 값의 일부를 결정하기 위해 사용될 수 있도록 RW2가 상기 캐시 내부의 엔트리들의 할당에 영향을 미치지 않게 하도록 구성된 컴퓨터 프로그램을 제공한다.

[0005] 적어도 일부 실시예는, 데이터 처리방법으로서, 추측 장벽 명령을 디코드하는 단계와, 상기 추측 장벽 명령의 디코딩에 응답하여, 처리회로를 제어함으로써, 프로그램 순서에서 상기 추측 장벽 명령 이후에 출현하고 상기 프로그램 순서에서 상기 추측 장벽 명령에 선행하는 이전 명령에 대한 어드레스 의존성을 갖는 후속 연산이 캐시 내부의 엔트리들의 할당에 추측적으로 영향을 미치지 않게 하는 단계를 포함하고, 상기 추측 장벽 명령이 완료할 때까지, 로드, 스토어, 데이터 또는 명령 프리로드 명령 RW2가 프로그램 순서에서 상기 추측 장벽 명령 이후에 출현하고 조건부 선택 명령의 결과에 대한 어드레스 의존성을 가질 때, 상기 조건부 선택 명령이 그것의 입력 레지스터들 중에서 한 개에 대해 추측적으로 실행된 로드 명령 R1에 대한 레지스터 데이터 의존성을 갖고, 상기 조건부 선택 명령이 그것의 나머지 입력 레지스터에 대해 R1에 대한 레지스터 데이터 의존성을 갖지 않고, 상기 조건부 선택 명령에 대한 조건이, R1이 아키텍처 상 실행되지 않으면 R1에 의존하지 않는 입력 레지스터가 선택되도록 하는 것인 경우, 상기 데이터 처리방법은, 상기 캐시 내부의 어떤 엔트리들이 할당되었는지 또는 퇴출되었는지의 평가에 의해 R1으로부터 추측적으로 로드된 값의 일부를 결정하기 위해 사용될 수 있도록 RW2가 상기 캐시 내부의 엔트리들의 할당에 영향을 미치지 않게 하는 단계를 더 포함하는 데이터 처리방법을 제공한다.

[0006] 적어도 일부 실시예는, 데이터 처리를 행하는 처리회로와, 명령들을 디코드하고 상기 처리회로를 제어하여 데이터 처리를 행하고, 조건부 분기 명령에 응답하여, 상기 처리회로를 제어하여, 상기 조건부 분기 명령 이후의 다음 명령이 제1 명령인 제1 결과와, 상기 다음 명령이 제2 명령인 제2 결과 중에서 한 개를 선택하게 하도록 구성된 명령 디코딩 회로와, 상기 조건부 분기 명령에 대해 상기 제1 결과가 선택되어야 하는지 상기 제2 결과가 선택되어야 하는지 예측하는 분기 예측회로를 구비하고, 상기 조건부 분기 명령의 일방향 추측 변종

(one-way-speculation variant)에 응답하여, 상기 분기 예측회로 및 상기 명령 디코딩 회로 중에서 적어도 한 개는, 상기 처리회로를 제어하여, 상기 분기 예측회로가 상기 조건부 분기 명령의 상기 일방향 추측 변종에 대해 상기 제1 결과를 예측할 때 상기 제1 명령에 대한 추측 실행에 대해 제한을 적용하고, 상기 분기 예측회로가 상기 조건부 분기 명령의 상기 일방향 추측 변종에 대해 상기 제2 결과를 예측할 때 상기 제2 명령에 대한 추측 실행에 대해 상기 제한의 적용을 생략하도록 구성된 장치를 제공한다.

[0007] 적어도 일부 실시예는, 조건부 분기 명령에 응답하여, 상기 조건부 분기 명령 이후의 다음 명령이 제1 명령인 제1 결과와, 상기 다음 명령이 제2 명령인 제2 결과 중에서 한 개를 선택하는 단계와, 상기 조건부 분기 명령에 대해 상기 제1 결과가 선택되어야 하는지 상기 제2 결과가 선택되어야 하는지 예측하는 단계와, 상기 조건부 분기 명령이 상기 조건부 분기 명령의 일방향 추측 변종인 경우, 상기 조건부 분기 명령의 상기 일방향 추측 변종에 대해 상기 제1 결과가 예측될 때 상기 제1 명령에 대한 추측 실행에 대해 제한을 적용하고, 상기 조건부 분기 명령의 상기 일방향 추측 변종에 대해 상기 제2 결과가 예측될 때 상기 제2 명령에 대한 추측 실행에 대해 상기 제한의 적용을 생략하는 단계를 포함하는 데이터 처리방법을 제공한다.

[0008] 본 발명의 또 다른 발명내용, 특징 및 이점은 다음의 첨부도면을 참조하여 주어지는 이하의 실시예의 상세한 설명으로부터 명백해질 것이다.

도면의 간단한 설명

[0009] 도 1은 추측 실행 명령을 지원하는 데이터 처리장치의 일례를 개략적으로 나타낸 것이다.

도 2는 추측 기반의 캐시 기반의 타이밍 부채널에 근거한 잠재적인 공격의 일례를 개략적으로 나타낸 것이다.

도 3은 추측 장벽 명령의 처리방법을 나타낸 것이다.

도 4는 스토어 추측 장벽 명령의 처리방법을 나타낸 것이다.

도 5는 조건부 분기 명령의 일방향 추측 변종의 처리방법을 나타낸 것이다.

도 6은 사용될 수 있는 시뮬레이터 예를 나타낸 것이다.

발명을 실시하기 위한 구체적인 내용

[0010] 데이터 처리장치는, 메모리에 있는 일부 데이터가 처리회로 상에서 실행되고 있는 특정한 프로세스에 의해 액세스될 수 없도록 보장하기 위한 메카니즘을 갖는다. 예를 들어, 메모리의 특정 영역에 대한 액세스를 제어하기 위해 특권 기반의 메카니즘 및/또는 메모리 보호 속성이 사용된다. 최근에, 추측 실행 및 캐싱을 이용하는 시스템에서, 추측 오류(misspeculation) 이후에 추측적으로 실행된 명령들의 아키텍처 효과가 역전된 후에도 데이터 캐시 내부에 추측적으로 실행된 명령들의 효과가 지속된다는 속성을 이용함으로써, 악의적인 사람이 액세스하지 않는 메모리의 영역으로부터 정보를 얻을 가능성이 있다는 것이 인식되었다. 이와 같은 공격은 분기 예측기 또는 기타 추측 메카니즘을 학습시켜, 더 큰 특권을 갖는 코드를 속여 특권을 갖는 코드가 민감한 정보에 따라 메모리 어드레스들의 패턴을 액세스하게 하도록 설계된 명령들의 시퀀스를 추측적으로 실행하게 함으로써, 이 민감한 정보에 대한 액세스를 갖지 않는 더 적은 특권을 갖는 코드가 캐시 타이밍 부채널을 이용하여, 더 큰 특권을 갖는 코드에 의해 캐시에 어떤 어드레스가 할당되었는지 또는 이 캐시로부터 어떤 어드레스가 퇴출되었는지 탐지함으로써, 민감한 정보가 추론될 수 있도록 할 수도 있는 일부 정보를 제공한다. 이와 같은 공격은 추측적인 부채널 공격으로 부를 수 있다.

[0011] 이하에서 더욱 상세히 설명하는 것과 같이, 데이터 처리장치의 명령 디코딩 회로는 추측 장벽 명령을 지원한다. 추측 장벽 명령에 응답하여, 명령 디코딩 회로는 처리회로를 제어하여, 프로그램 순서에서 추측 장벽 명령 이후에 출현하고 프로그램 순서에서 추측 장벽 명령에 선행하는 이전 명령에 대한 어드레스 의존성을 갖는 후속 연산이 캐시 내부의 엔트리들의 할당에 추측적으로 영향을 미치지 않게 한다. 예를 들어, 민감한 정보를 액세스하는 것이 허용되는 컴파일러 생성 코드의 프로그래머는, 신뢰되지 않은 코드에 의해 제공된 값으로부터 유도되는 어드레스를 갖는 이전 로드 명령과 이전 로드 명령에 의해 로드된 값에 근거하여 계산된 어드레스를 갖는 나중 로드 명령 사이에 추측 장벽을 포함할 수도 있다. 장벽의 포함한, 캐시 타이밍 채널을 사용하여 신뢰되지 않은 코드가 액세스가 불가능한 민감한 정보에 대한 정보를 결정할 수도 있는 방식으로 나중의 로드 명령이 캐시 내부의 엔트리들의 할당에 추측적으로 영향을 미치지 않게 한다. 이 때문에, 추측 장벽 명령에 대한 지원을 제공함으로써, 보안이 증진될 수 있다.

[0012] 캐시 내부의 엔트리들의 할당의 추측적 영향을 금지하는 추측 장벽 명령에 뒤따르는 후속 연산은 로드 명령 및 스토어 명령 중에서 한 개이다. (로드 또는 스토어 명령이 어드레스 의존성을 갖는) 이전 명령은 로드 명령이다.

[0013] 일 실시예에서는, 추측 장벽 명령이 완료할 때까지,

[0014] · 로드, 스토어, 데이터 또는 명령 프리로드 RW2가 프로그램 순서에서 추측 장벽 명령 이후에 출현하고 조건부 선택 명령의 결과에 대한 어드레스 의존성을 가질 때,

[0015] ◦ 조건부 선택 명령이 그것의 입력 레지스터들 중에서 한 개에 대해 추측적으로 실행된 로드 R1에 대해 레지스터 데이터 의존성을 갖고,

[0016] ◦ 조건부 선택 명령이 그것의 나머지 입력 레지스터에 대해 R1에 대한 레지스터 의존성을 갖지 않고,

[0017] ◦ 조건부 선택 명령에 대한 조건이, R1이 아키텍처 상 실행되지 않는 경우 R1에 의존하지 않는 입력이 선택되도록 하는 것인 경우,

[0018] 처리회로는, 캐시 내부의 어떤 엔트리들이 할당되거나 퇴출되었는지의 평가에 의해 R1으로부터 추측적으로 로드된 값의 일부를 결정하기 위해 사용될 수 있도록, RW2가 캐시 내부의 엔트리들의 할당에 영향을 미치지 않게 하도록 구성된다. 이것은 후속 연산 RW2의 캐시에 대한 영향을 분석함으로써 로드 R1에 의해 추측적으로 로드된 값에 대한 정보를 공격자가 유도할 수 없게 한다.

[0019] 또 다른 실시예에서는, 추측 장벽 명령이 완료할 때까지,

[0020] · 로드, 스토어, 데이터 또는 명령 프리로드 RW2가 프로그램 순서에서 추측 장벽 명령 이후에 출현하고 조건부 선택 명령의 결과에 대한 어드레스 의존성을 가질 때,

[0021] ◦ 조건부 선택 명령이 그것의 입력 레지스터에 대해 추측적으로 실행된 로드 R1에 대해 레지스터 데이터 의존성을 갖지 않고,

[0022] ◦ 조건부 이동 명령에 대한 조건이 R1이 아키텍처 상 실행되지 않는 경우 조건을 통과하는 것인 경우,

[0023] 처리회로는, 캐시 내부의 어떤 엔트리들이 할당되거나 퇴출되었는지의 평가에 의해 R1으로부터 추측적으로 로드된 값의 일부를 결정하기 위해 사용될 수 있도록, RW2가 캐시 내부의 엔트리들의 할당에 영향을 미치지 않게 하도록 구성된다. 마찬가지로, 이것은 공격자가 로드 R1의 올바르지 않은 추측을 이용하여 후속 연산 RW2의 캐시 할당에 대한 영향의 분석에 근거하여 R1에 의해 로드된 값에 대한 정보를 얻을 수 없게 한다.

[0024] 추측 장벽 명령에 뒤따르는 후속 연산에 의한 엔트리들의 할당의 추측적인 영향이 금지되는 캐기는 데이터 캐시, 명령 캐시 및 분기 예측 캐시 중 어느 한 개일 수 있다. 캐시의 다른 예로는, 수치 예측기 캐시, 로드/스토어 에일리어싱 예측기 캐시 및 기타 예측 구조일 수도 있다.

[0025] 처리회로가 추측 장벽 명령에 뒤따르는 후속 연산이 캐시 내부의 엔트리들의 할당에 추측적으로 영향을 미치지 않게 하는 다양한 방법이 존재한다. 일 실시예에서, 이것은, 적어도 추측 장벽 명령이 완료할 때까지, 후속 연산의 추측적 실행을 금지함으로써 금지될 수 있다. 프로그램 순서에서 선행하는 명령이 해결되면 추측 장벽 명령이 완료된 것으로 생각될 수도 있다. 일부 실시예에서는, 추측 장벽 명령이 완료되면, (추측이 추측 장벽 명령에 뒤따르는 연산과 관련된 예측에 근거한다면) 후속 연산의 추측적 실행이 허용될 수도 있다. 다른 실시예에서는, 처리회로는, 후속 연산의 추측적 실행을 허가하면서, 후속 연산이 캐시 내부의 엔트리들의 할당에 추측적으로 영향을 미치는 것을 금지한다. 캐시 내부의 엔트리들의 할당에 대한 영향은, 캐시를 할당하는 엔트리를 이전에 갖지 않은 지정된 어드레스에 대한 새로운 엔트리의 할당이거나, 지정된 어드레스가 더 이상 캐시되지 않도록 지정된 어드레스와 관련된 이전에 캐시된 엔트리의 퇴출일 수도 있다.

[0026] 예를 들어, 추측적 실행 그 자체를 금지하는 것이 아니라, 캐시에 대한 추측적 실행의 영향을 금지할 수도 있다. 예를 들어, 일부 구현예는, 추측 장벽 명령에 선행하는 명령의 결과의 추측적 예측에 근거하여 실행되고 추측 장벽 명령이 아직 완료되지 않은 경우에, 장벽 명령이 뒤따르는 추측적으로 실행된 연산이 데이터를 캐시하지 못하게 하도록 선택할 수도 있다.

[0027] 이와 달리, 추측적 장벽 명령이 완료될 때까지 후속 연산에 의해 추측적으로 로드된 데이터가 추측 장벽에 유지될 수도 있으며, 추측이 해결되고 추측 장벽 명령이 완료될 때까지, 주 캐시 내부에 할당되지 않는다. 예를 들어, 추측 버퍼는 주 캐시의 캐시 웨이(cache way)와 함께 추가적인 캐시 웨이로부터 참조되는 버퍼이지

만, 추측 버퍼의 콘텐츠는 특정한 특권 레벨의 천이(예를 들어, 더 적은 특권을 갖는 상태로의 천이)시에 폐기됨으로써, 추측적으로 로드된 데이터가 아직 올바른 것으로 해결되지 않았을 때 더 적은 특권을 갖는 코드가 이 추측적으로 로드된 데이터를 액세스할 수 없다. 이와 같은 접근방법은, 주 캐시 영역이 추측적으로 로드된 값들로 오염되는 것을 방지함으로써(또한 이들 추측적으로 로드된 값들의 할당으로 인한 주 캐시로부터 다른 값들의 퇴출을 방지함으로써), 그후에 더 적은 특권을 갖는 상태로의 천이가 존재하는 경우에, 더 적은 특권을 갖는 코드가 캐시 타이밍 부채널을 이용하여 캐시된 값에 대한 정보를 추론할 수 없게 된다.

[0028] 일 실시예에서는, 추측 장벽 명령에 응답하여, 명령 디코딩 회로는 처리회로를 제어하여 후속 명령이, 이전 명령에 대한 데이터 값 예측과, 이전 명령이 조건부 분기 명령 이외의 명령일 때 이전 명령에 대한 조건 코드 예측과, 이전 명령이 벡터값의 어떤 데이터 성분들이 처리할 활성 성분들인지를 나타내는 프레디케이트(predicate) 정보에 의존하는 벡터 명령인 경우에 이 프레디케이트 정보의 예측 중에서 적어도 한 개를 이용하여 추측적으로 실행되지 않게 한다. 후속 명령들이 추측 장벽 명령에 선행하는 명령에 대해 행해진 데이터 값 예측, 조건 코드 예측 또는 프레디케이트 예측의 결과를 추측적으로 이용할 수 없게 함으로써, 공격자가 이용가능한 방법을 줄일 수 있다.

[0029] 일 실시예에서, 처리회로는 추측 장벽 명령에 선행하는 아키텍처 상 해결되지 않은 이전 명령과 관련된 예측에 근거하여 추측 장벽 명령에 뒤따르는 명령들의 제어 흐름의 추측 제어를 허가한다. 이 때문에, 추측 장벽 명령 전후의 흐름 제어 추측이 여전히 허가될 수도 있다. 장벽에 뒤따르는 추측적으로 실행된 명령에 응답하는 캐싱의 효과의 억제가 전술한 공격을 방해하는데 충분하기 때문에 추측적인 제어 흐름을 허가하는 것이 가능하다. 장벽 전후의 제어 흐름의 추측적 제어를 가능하게 함으로써, 성능을 향상시킬 수 있다. 마찬가지로, 일부 실시예에서는, 데이터 값 예측이나, 이전 명령이 아직 아키텍처 상 해결되지 않았을 때 프로그램 순서에서 추측 장벽 명령에 선행하는 이전 명령의 조건부 코드 예측의 결과를 사용하지 않는다면, 추측 장벽 명령에 뒤따르는 조건부 명령의 추측적 실행이 허가된다.

[0030] 일부 데이터 처리장치에서 다른 형태의 추측은 로드 또는 스토어 명령의 어드레스의 추측이다. 로드 또는 스토어 명령의 어드레스는 이전 명령에 의해 계산된 레지스터로부터의 값에 근거한 어드레스 계산에 의존한다. 스토어 명령 뒤에 로드의 어드레스가 아직 알려지지 않은 로드 명령이 뒤따를 때, 로드의 어드레스가 이전 스토어의 어드레스와 다른 경우보다는 가능성이 더 작기는 하지만, 로드 명령의 어드레스가 실제로 이전 스토어 명령의 어드레스와 같은 것으로 판명될 수 있는 위험이 존재한다. 스토어 명령의 실행은 스토어 갑이 의존하는 이전 명령들의 결과의 대기로 인해 지연될 수도 있다. 로드가 스토어와 실제로 동일한 어드레스를 액세스하는지 여부는 로드 명령이 실행될 때까지 알려지지 않는다. 처리회로가 실행을 위한 로드 명령을 발행하기 전에 스토어 명령이 완료되는 것을 대기한다면, 로드 어드레스가 실제로 스토어 어드레스와 다른 것으로 판명되는 경우 이것이 불필요한 지연을 일으킬 수도 있을 것이다.

[0031] 이 때문에, (이전 스토어를 우회하면서) 로드 명령을 일찍 추측적으로 발행함으로써, 로드 명령의 어드레스가 실제로 스토어 명령의 어드레스와 다른 것으로 판명되는 경우, 로드가 이전 스토어 명령으로부터의 값을 대기하면서 불필요하게 지연되지 않으므로, 성능을 향상시킨다. 그러나, 로드 어드레스가 실제로 이전 스토어 어드레스와 같은 것으로 판명되는 경우, 로드 명령이 스토어 명령의 실행으로부터 발생되는 값이 아니라 스토어 명령이 실행되기 전에 타겟 어드레스에 기억된 데이터 값을 로드하므로, 이와 같은 추측적 실행이 잘못된 값이 로드되게 할 수도 있다.

[0032] 일부 시나리오에서는, 공격자가 민감한 정보에 액세스하기 위한 시도로 이와 같은 로드에 의한 이전 스토어의 추측적 우회를 이용하려도 시도할 수도 있다. 이와 같은 공격을 방지하기 위해 스토어-추측 장벽 명령이 제공된다. 스토어-추측 장벽 명령에 응답하여, 명령 디코딩 회로는 프로그램 순서에서 스토어-추측 장벽 명령에 선행하는 이전의 스토어 명령이 프로그램 순서에서 스토어-추측 장벽 명령에 뒤따르고 이전의 스토어 명령과 동일한 어드레스를 지정하는 후속하는 로드 명령의 추측적 실행에 의해 우회되지 않게 한다. 스토어-추측 장벽 명령은 전술한 추측 장벽 명령과 다른 명령 인코딩을 가질 수 있다. 이 때문에, 스토어-추측 장벽 명령의 지원은 증가된 보안을 제공한다.

[0033] 스토어-추측 장벽 명령의 다양한 변종이 제공된다. 제1 변종의 스토어-추측 장벽 명령에 대해, 후속 로드 명령은 이전 스토어 명령과 동일한 가상 어드레스를 지정하는 명령이다. 제2 변종에 대해, 후속 로드 명령은 이전 스토어 명령과 동일한 물리 어드레스를 지정하는 것이다. 예를 들어, 운영체계에 진입하거나 벗어날 때 추측적 로드가 캐시 타이밍 부채널 메카니즘을 사용하여 이동되지 않도록 하는데 제2 변종의 제공이 유용할 수 있다.

[0034] 스토어-추측 장벽 명령에 응답하여, 명령 디코딩 회로는, 프로그램 순서에서 스토어-추측 장벽 명령에 선행하는 이전 로드 명령이, 프로그램 순서에서 스토어-추측 장벽 명령에 뒤따르고 이전 로드 명령과 동일한 어드레스를 지정하는 후속하는 스토어 명령으로부터 발생되는 데이터를 추측적으로 로드하지 못하게 할 수도 있다. 이것은, 스토어가 다른 어드레스를 갖지만, 나중에 이전 로드와 동일한 어드레스에 실제로 스토어하는 것으로 판명된다는 가정하에서, 추측 장벽 명령에 뒤따르는 스토어가 이전 로드에 앞서서 실행되는 반대의 시나리오를 이용하려고 시도하는 공격자로부터 보호하는데, 이 경우 스토어의 추측적 실행이 잘못된 값이 이전 로드에 의해 로드되게 할 수도 있다. 스토어-추측 장벽 명령은, 공격자가 캐시 타이밍 부채널을 이용하여 이와 같은 잘못된 추측에 기인한 캐시 할당의 변경을 탐지할 수 있는 것을 방지하여, 보안을 향상시킬 수 있다.

[0035] 추측적 실행을 일으키는 한가지 종류의 명령은 조건부 분기 명령이며, 이것에 응답하여 명령 디코더는 처리회로를 제어하여, 제1 명령이 조건부 분기 명령 이후의 다음 명령으로서 선택되는 제1 결과, 또는 제2 명령이 조건부 분기 명령 이후의 다음 명령으로서 선택되는 제2 결과를 선택하게 한다. 예를 들어, 제1 및 제2 결과들 중에서 한 개는 분기를 타는(taken) 결과이고, 나머지는 조건부 분기 명령의 분기를 타지 않는(not-taken) 결과이다. 실제 결과가 알려지기 전에, 조건부 분기 명령에 대해 제1 결과가 선택되어야 하는지 또는 제2 결과가 선택되어야 하는지 예측하기 위해 분기 예측회로가 설치된다. 그후, 후속 명령들이 분기 예측에 근거하여 추측적으로 실행될 수 있다.

[0036] 일 실시예에서는, 조건부 분기 명령의 일방향-추측 변종이 제공되며, 이에 응답하여 분기 예측회로 및 명령 디코딩 회로 중 적어도 한 개가 제어회로를 제어하여, 분기 예측회로가 조건부 분기 명령의 일방향-추측 변종에 대해 제1 결과를 예측할 때 제1 명령에 대한 추측적 실행에 대해 제한을 적용하게 하는 반면에, 분기 예측회로가 조건부 분기 명령의 일방향-추측 변동에 대해 제2 결과를 예측하는 경우에 제2 명령에 대해 추측적 실행에 대한 제한을 생략한다.

[0037] 이 때문에, 분기 예측기가 조건부 분기에 대한 한 개의 결과를 예측할 때보다 분기 예측기가 나머지 결과를 예측할 때 추측적 실행이 더 제한된다. 이것은 전술한 종류의 추측 기반의 캐시 타이밍 부채널 공격으로부터 보호하는데 유용할 수 있다. 예를 들어, 조건부 분기가 잠재적으로 신뢰되지 않은 코드로부터 그것에 전달된 값이 허용가능한 범위 내에 있는지 여부를 검사하거나, 또는 잠재적으로 신뢰되지 않은 코드에 의해 제공된 패스워드가 올바른지 여부를 검사하는 경우, 분기를 타는 결과 및 분기를 타지 않는 결과들 중에서 한 개가 잠재적으로 민감한 정보의 처리를 계속하도록 설계되는 반면에, 나머지 결과는 이 민감한 정보를 액세스할 위험을 갖고 있지 않을 수도 있다. 이 때문에, 일반적으로 조건부 분기의 결과들 중에서 어느 것이 나머지보다 공격 위험이 더 높은지 프로그래머 또는 컴파일러에 의해 알려진다. 조건부 분기 명령의 일방향-추측 변종을 제공하여, 어떤 추측적 실행이 한 개의 예측된 분기 결과보다 나머지 예측된 분기 결과에 대해 더 제한되는지 뒤따름으로써, 제2 결과가 예측될 때에는 추측적 실행을 허가함으로써 프로세서가 성능을 향상시킬 수 있게 할 수 있으면서, 제1 결과가 예측될 때에는 추측적 실행을 제한함으로써 보안을 증가시킬 수 있다.

[0038] 조건부 분기에 대해 제1 결과가 예측되는 경우에 추측 실행을 제한하는 다양한 방법이 존재한다. 일 실시예에서, (적어도 예측이 올바른 것으로 해결될 때까지) 조건부 분기를 뒤따르는 지정된 명령의 추측 실행이 금지될 수도 있다. 이와 달리, (마찬가지로, 적어도 예측이 올바른 것으로 해결될 때까지) 추측의 제한이 여전히 후속 명령들이 추측적으로 실행될 수 있도록 하지만, 분기를 뒤따르는 후속 명령에 적용된 캐싱시의 제한을 포함할 수도 있다. 예를 들어, 캐싱은 조건부 분기에 대한 제1 결과의 예측 후에 추측적으로 실행된 명령에 응답하여 금지되거나, 또는 전술한 실시예에서와 마찬가지로 주 캐시로부터 분리된 별개의 추측 버퍼 내에 가능하게 될 수 있다.

[0039] 일반적으로, 조건부 분기 명령의 일방향 추측 변종은 (분기 예측기에 의해 제1 결과가 예측되는지 또는 제2 결과가 예측되는지에 무관하게 추측 실행이 제한되지 않는) 조건부 분기 명령의 종래의 양방향(two-way) 추측 변종과 다양하게 구별된다. 일 실시예에서, 일방향 추측 변종은 다른 명령 인코딩(예를 들어, 다른 명령 옵코드, 또는 서로 다른 변종들을 구별하는 명령 인코딩 내부의 필드)을 갖는다. 이와 달리, 조건부 분기 명령의 일방향 및 양방향 추측 변종들은 동일한 인코딩을 가질 수도 있지만, 데이터 처리장치의 제어 레지스터에 기억된 제어 파라미터가 이와 같은 조건부 분기 명령과 마주칠 때 어떤 변종을 사용할 것인지를 지정한다.

[0040] 또한, 일방향 추측 변종 그 자체의 다양한 대안적인 변종들이 존재한다. 예를 들어, 제1 변종에 대해, 제1 결과는 분기를 타는 결과를 포함하고 제2 결과는 분기를 타지 않는 결과를 포함한다. 제2 변종에 대해, 제1 및 제2 결과가 반대이므로, 제1 결과가 분기를 타지 않는 결과를 포함하는 반면에, 제2 결과가 제2 결과를 포함한다. 또한, 이들 변종은 서로 다른 명령 인코딩에 의해 또는 제어 레지스터 내부의 파라미터에 의해 구별될 수

도 있다. 이 때문에, 제1 변종의 경우에 분기를 타는 결과에 대해 또는 제2 변종의 경우에 분기를 타지 않는 결과에 대해 추측 실행에 대한 제한이 적용되는 다양한 변종들을 제공함으로써, 프로그래머 또는 컴파일러가, 분기를 타는 결과 또는 분기를 타지 않는 결과의 어느 것이 추측적 캐시 타이밍 부채널에 의한 공격에 더 취약한지에 따라 어떤 변종을 사용할 것인지를 선택할 수 있는 유연성을 제공한다.

[0041] 조건부 분기 명령의 일방향 변종에 대해 제1 결과가 예측될 때 추측 실행의 제한을 적용하지만 제2 결과가 예측될 때에는 적용하지 않도록 처리장치의 마이크로 아키텍처가 보장하는 다양한 방법이 존재한다.

[0042] 일 실시예에서, 분기 예측회로는, 이전의 조건부 분기 명령들의 결과에 근거하여 학습된 분기 예측 상태 정보에 근거하여 예측을 하고, 분기 예측 상태 정보의 학습으로부터 조건부 분기 명령의 일방향 추측 변종을 제외한다. 분기 예측 상태의 학습으로부터 일방향 변종을 제외함으로써, 분기 예측기가 일방향 변종의 실제 결과를 학습하지 못하게 하여, 예를 들어, 일방향 변종에 대한 기억된 분기 예측 상태를 포함시키지 않거나, 또는 실제 분기 결과에 근거하여 학습되지 않고 디폴트로 일방향 분기에 대해 제2 결과를 예측하는 분기 예측 상태의 엔트리를 포함시킴으로써, 추측이 제한되는 제2 결과를 향해 일방향 변종에 대한 예측이 치우칠 수 있도록 한다.

[0043] 예를 들어, 일부 분기 예측기에서는, 기억된 분기 예측 상태가 존재하지 않는 명령에 대해 예측이 디폴트로 분기를 타지 않는 결과가 된다. 이 경우, (제1 결과가 분기를 타는 것이고 제2 결과가 분기를 타지 않는) 일방향 추측 조건부 분기 명령의 제1 변종에 대해, 분기 예측기가 기억된 분기 예측 상태가 존재하지 않는 명령에 대해 디폴트로 제2 결과(분기를 타지 않는 것)를 예측하므로, 일방향 분기에 대해 분기 예측 상태 정보를 할당하지 않는 것이 가능하여, 제1 결과에 근거하여 명령을 추측적으로 실행하는 것이 가능하지 않도록 보장한다. 이와 달리, (예측이 제한되어야 하는 제1 결과가 분기를 취하지 않는 결과인) 제2 변종에 대해, 제2(분기를 타는 것) 결과를 디폴트 예측으로 지정하는 분기 예측회로에 엔트리가 할당될 수도 있으며, 그후 (일방향 분기의 실제 결과에 근거한 학습이 존재하지 않으므로) 일방향 분기에 대한 실제 분기 결과가 제1 결과이더라도 계속해서 제2 결과를 예측하여, 제1 결과에 근거한 추측이 존재할 수 없도록 보장한다.

[0044] 마이크로 아키텍처에서 조건부 분기 명령의 일방향 추측 변종의 제1 결과에 대한 추측을 제한하는 또 다른 방법은, 일방향 추측 제어가 존재하지 않는 조건부 분기와 마찬가지로, 분기 예측기가 조건부 분기 명령의 일방향 추측 변종의 실제 결과에 근거하여 그것의 분기 예측 상태 정보를 여전히 학습시키지만, 분기 예측기에 의해 어떤 예측이 출력되는지에 근거하여, 처리 파이프라인이 추측이 허용되는지 여부, 또는 추측이 허용되는 경우에, 캐싱에 대한 제한이 존재해야 하는지 여부를 제어한다. 이 때문에, 이와 같은 마이크로 아키텍처 접근방법에 따르면, 분기 예측기가 나머지 양방향 추측된 조건부 분기와 유사한 일방향 추측 변종에 근거하여 그것의 분기 예측 상태를 학습시키지만, 추측을 할 것인지 아닌지의 필터링이 일방향 추측 변종에 대해 어떤 결과가 예측되는지에 근거하여 적용될 수 있다.

[0045] 이때, 이들 마이크로 아키텍처 접근방법들 모두는, 제1 결과가 예측될 때에는 추측시의 제한이 적용되지만 제2 결과가 예측될 때에는 제한이 생략되는 조건부 분기 명령의 일방향 추측 변종의 아키텍처 상 정의 내에 포함된다.

[0046] 더구나, 일부 실시예에서, 명령 디코더는, 분기 예측회로에 의해 제1 결과가 선택되는지 또는 제2 결과가 선택되는지에 무관하게 분기에 뒤따르는 다음 명령에 대해 추측 실행에 대한 제한이 적용되는 조건부 분기 명령의 추측 제한된 변종을 더 지원한다. 이것은 프로그램 흐름이 양쪽 분기들이 잠재적인 추측 기반의 캐시 타이밍 부채널 공격의 위험에 있는 것으로 생각되는 경우에 유용할 수 있으므로, 이와 같은 특정한 분기에 대해, 파이프라인이 이 명령에 대해 행해진 분기에 대해 추측적으로 작용하거나, 또는 이와 달리, 추측적 실행이 허용되는 경우에는, 추측이 제1 결과 또는 제2 결과에 근거하였는지 여부에 무관하게, 조건부 분기 명령과 관련된 조건이 해결될 때까지 이와 같은 상태 실행된 명령들의 캐싱의 효과가 제한될 수도 있다.

[0047] 이 때문에, 전술한 기술은 잠재적인 부채널 공격을 방지하는 강건성을 제공하는데 도움이 된다.

[0048] 도 1은 다수의 파이프라인 스테이지들을 포함하는 처리 파이프라인을 갖는 데이터 처리장치(2)의 일례를 개략적으로 나타낸 것이다. 파이프라인은 분기 명령의 결과를 예측하는 분기 예측기(4)를 구비한다. 페치 스테이지(6)는 분기 예측기(4)에 의해 행해진 예측에 근거하여 일련의 페치 어드레스들을 발생한다. 페치 스테이지(6)는 명령 캐시(8)로부터 페치 어드레스들에 의해 식별된 명령들을 페치한다. 디코드 스테이지(10)는 페치된 명령들을 디코드하여 파이프라인의 후속 스테이지들을 제어하기 위한 제어 정보를 발생한다. 재명명(rename) 스테이지(12)는 레지스터 재명명을 행하여 명령들에 의해 식별된 아키텍처 레지스터 지정자들을 하드웨어로 설치

된 레지스터들(14)을 식별하는 물리 레지스터 지정자들에 매핑한다. 레지스터 재명명은, 동일한 아키텍처 레지스터를 지정하는 명령들을 하드웨어 레지스터 파일 내부의 서로 다른 물리 레지스터들에 매핑하여 이들 명령들 사이의 해저드(hazard)가 제거될 수 있도록 할 수 있으므로, 명령들이 캐시(8)로부터 페치되었던 프로그램 순서와는 다른 순서로 명령들이 실행될 수 있는 가능성을 증가시킴으로써, 이전 명령이 피연산자가 이용가능해질 때까지 대기하고 있는 동안 다음 명령이 실행될 수 있도록 하여 성능을 향상시킬 수 있기 때문에, 비순차 실행을 지원하는데 유용할 수 있다. 아키텍처 레지스터들을 다양한 물리 레지스터들에 매핑할 수 있는 능력은 분기 예측 실패의 경우에 아키텍처 상태의 롤백(rolling back)을 용이하게 할 수도 있다. 발행 스테이지916)는, 이들 명령을 처리하기 위해 필요한 피연산자들이 레지스터들(14)에서 이용가능해질 때까지 실행을 기다리는 명령들을 대기열로 만든다. 실행 스테이지(18)는 명령들을 실행하여 대응하는 처리 연산을 수행한다. 후기록(writeback) 스테이지920)는 실행된 명령들의 결과를 다시 레지스터들(14)에 기록한다.

[0049] 실행 스테이지(18)는, 분기 명령이 올바르게 예측되었는지 평가하는 분기 유닛(21), 산술 또는 논리 연산을 행하는 ALU(arithmetic logic unit)(22), 부동소수점 피연산자들을 사용하여 연산을 행하는 부동소수점 유닛(24), 단일 명령에 응답하여 복수의 독립적인 데이터 성분들이 처리되는 벡터 연산을 처리하는 벡터 처리 유닛(25)과, 메모리 시스템으로부터 레지스터들(14)에 데이터를 로드하는 로드 연산이나 레지스터들(14)로부터 메모리 시스템에 데이터를 기억하는 스토어 연산을 행하는 로드/스토어 유닛(26) 등의 다수의 실행 유닛들을 구비한다. 본 실시예에서, 메모리 시스템은, 레벨 1 명령 캐시(8), 레벨 1 데이터 캐시(30), 데이터와 명령들 사이에서 공유되는 레벨 2 캐시(32)와 메인 메모리(24)를 포함하지만, 이것은 가능한 메모리 계층구조의 일례에 지나지 않으며 다른 구현예는 또 다른 레벨의 캐시 또는 이와 다른 배치를 가질 수 있다는 것은 자명하다. 메모리에 대한 액세스는 어드레스 변환 및/또는 메모리 보호를 제어하는 메모리 관리 유닛(mmu)(35)을 사용하여 제어된다. 로드/스토어 유닛(26)은 MMU(35)의 변환 색인 버퍼(TLB)(36)를 이용하여 파이프라인에 의해 발생된 가상 어드레스를 메모리 시스템 내부의 위치를 식별하는 물리 어드레스에 매핑한다. 이I, 도 1에 도시된 파이프라인은 일례에 지나지 않으며, 다른 실시예는 이와 다른 세트의 파이프라인 스테이지들 또는 실행 유닛들을 가질 수 있다는 것은 자명하다. 예를 들어, 순차 프로세서는 재명명 스테이지(12)를 갖지 않아도 된다.

[0050] 분기 예측기(4)는, 조건부 분기 명령에 대한 분기 결과의 예측 및/또는 간접 분기 명령에 대한 타겟 어드레스의 예측에 근거하여, 데이터 처리 연산이 실제로 필요한지 여부가 알려지기 전에 데이터 처리 연산을 추측적으로 행하기 위해 데이터 처리장치에 의해 사용되는 추측 메카니즘의 일례이다. 또한, 실행 스테이지를 제어하여 명령들과 관련된 정보의 (분기 예측 이외의) 예측에 근거하여 이들 명령을 추측적으로 실행하게 하기 위한 실행 유닛(180과 관련된 추측 제어회로(40)가 존재한다.

[0051] 예를 들어, 조건부 명령은 실행 스테이지(18)를 제어하여, 레지스터들(14)에 기억되는 조건 상태 코드들(42)의 값을 조건으로 하여, 조건부 처리 연산을 행하게 한다. 일부 조건 설정 명령은 명령의 결과에 근거하여 조건 상태 코드들(42)이 갱신되게 한다. 예를 들어, ALU(22)에 의해 처리된 산술 명령은, 산술 연산의 결과가 제로값이었는지 여부, 결과가 음이 없는지 여부, 또는 연산이 부호를 갖는 오버플로우를 발생하였는지 또는 부호를 갖지 않는 오버플로우를 발생하였는지 등과 같이, 결과의 속성을 표시하도록 조건 코드들(42)이 갱신되게 할 수도 있다. 그후, 후속하는 조건부 명령들은 조건 상태 코드들(42)의 현재값이 특정한 테스트 조건을 만족하는지 여부를 테스트한다. 아키텍처 관점에서, 코드가 테스트 조건을 만족하면, (산술 또는 논리 연산 등의) 관련된 처리 연산이 행해지는 반면에, 조건 상태 코드들(42)이 테스트 조건을 만족하지 않으면, 조건부 연산이 행해지지 않고, 그 대신에 명령이 아키텍처 효과를 갖지 않는 비연산 명령으로서 취급된다. 그러나, 마이크로아키텍처에서, 실제 조건 코드들이 알려지기 전에, 추측 제어회로(40)가 조건 상태 코드들(42)의 예측에 근거하여 조건부 명령과 관련된 처리 연산을 추측적으로 실행하여, 조건 코드들을 변경하는 이전 명령들이 완료되는 것을 대기하지 않게 한다. 예측이 올바르지 않은 것으로 판명되는 경우, 추측적으로 실행된 명령들의 결과가 폐기될 수 있으며, 프로그램 흐름이 최종의 올바른 실행 지점으로 되돌아갈 수 있다.

[0052] 추측 제어회로(40)에 의해 행해질 수 있는 또 다른 형태의 추측은, 벡터 처리 유닛(25)에 의해 실행된 벡터 명령과 관련된 프레디케이트 값(44)의 예측일 수 있다. SIMD(single instruction multiple data) 명령으로도 알려진 벡터 명령은 동일한 레지스터 내부에 기억된 복수의 데이터 성분들에 대해 작용한다. 예를 들어, 벡터 가산 명령은 벡터 처리 유닛을 기동하여 복수의 가산 연산을 행하며, 이때 이들 가산 연산들 각각은 2개의 벡터 레지스터들의 대응하는 위치들에 있는 각각의 쌍의 데이터 성분들을 가산하여, 결과 벡터 레지스터에 기록되는 대응하는 결과 성분을 생성한다. 이것은 한 개의 명령에 응답하여 다수의 독립적인 가산이 행해질 수 있도록 할 수 있다. 벡터 명령들은, 벡터 명령들을 포함하는 벡터화된 명령들의 루프의 한번의 반복에 응답하여 처리할 스칼라 루프의 복수의 반복이 벡터 처리 유닛(250에 의해 실행될 수 있도록 함으로써 처리 명령들의 스칼

라 루프가 더 신속하게 처리될 수 있도록 하는데 유용할 수 있다.

[0053] 벡터화된 시퀀스의 명령들 내에, 조건부 기능을 포함시킴으로써, 벡터의 한 개의 성분이 특정한 조건을 만족하지 않으면, 그렇지 않은 경우에 이 성분에 대해 행해질 후속 연산들이 실행되지 않는 반면에, 동일한 벡터 내의 나머지 성분들이 필요한 조건을 만족하는 경우 이들 성분들이 여전히 처리되도록 하는 것이 바람직할 수도 있다. 또한, 스칼라 루프들을 벡터화할 때, 스칼라 루프의 반복 횟수가 벡터 내부에 설치된 성분들의 수의 정확한 배수에 매핑되며, 이 경우 최종 벡터 루프 반복에서 벡터를 완전히 채우는데 충분한 스칼라 반복이 존재하지 않으므로, 벡터의 일부 성분들이 처리될 필요가 없는 루프 테일(loop tail) 반복이 존재할 수도 있다. 이 때문에, 벡터의 어떤 성분들이 활성 성분들인지 지정하는 프레디케이트 값(44)을 정의하는 것이 유용할 수 있다, 결과 벡터의 비활성 성분들은, 제로값으로 클리어되거나, 또는 명령을 실행하기 전에 목적지 레지스터의 이들 부분에 기억되었던 이전 값을 유지할 수도 있다.

[0054] 따라서, 대응하는 벡터 명령의 결과를 결정할 수 있기 전에, 프레디케이트 값(44)을 알 필요가 있다. 프레디케이트 값(44)은, 이전 명령들, 예를 들어, 다른 명령들의 결과에 대해 대기하고 있는 조건부 명령들에 의해 설정될 수도 있다. 프레디케이트가 실제로 계산되는 것을 대기하는 것은 벡터 명령을 지연시킨다. (예를 들어, 동일한 명령을 실행하는 이전 인스턴스, 또는 모든 성분들이 활성이라는 디폴트의 가정에 근거하여) 프레디케이트의 값에 대해 행해질 수 있는 예측이 존재하는 경우, 예측이 올바른 경우에, 벡터 명령이 추측적으로 실행되어 성능을 향상시킬 수 있다. 프레디케이트의 예측이 나중에 올바르지 않은 것으로 판명되는 경우, 처리가 이전의 실행 지점으로 되돌아가 올바르지 않게 추측된 명령들의 결과를 폐기할 수 있다. 이 때문에, 또 다른 형태의 추측 제어는 프레디케이트 값(44)의 예측에 근거하여 벡터 명령들을 추측적으로 실행하는 것이다.

[0055] 또 다른 형태의 추측은 로드/스토어 유닛에 의해 실행된 로드 또는 스토어 명령들의 어드레스에 대한 것일 수 있다. 예를 들어, 로드 명령이 이전의 스토어 명령을 뒤따르거나 스토어 명령이 이전의 로드 명령을 뒤따르는 경우, 이들이 실제로 서로 다른 데이터 값을 액세스하므로 독립적이라는 가정에서, 제1 명령에 앞서서 제2 명령이 추측적으로 실행되어, 어드레스들이 서로 다른 것으로 판명된 경우에 성능을 향상시킬 수 있다. 그러나, 추측이 올바르지 않은 것으로 판명되고 한쌍의 명령들 중에서 제2 명령이 실제로 결국 제1 명령과 동일한 어드레스를 액세스하는 경우, 추측이 올바르지 않아, 명령들 중에서 한 개가 올바르지 않은 결과를 제공하게 할 수도 있다. 예측 실패가 검출되는 경우, 처리가 이전의 실행 시점으로 되돌아갈 수 있다.

[0056] 이와 같은 추측 메카니즘이 공격자에 의해 잠재적으로 이용되어, 공격자가 액세스하는 것이 허용되지 않는 민감한 정보를 액세스할 수도 있다. 처리장치는 특권 기반의 메카니즘을 사용하여 동작하는데, 이 경우, MMU(35)는 주어진 특권 레벨 이상에서 실행된 코드에 대한 메모리 어드레스 공간의 특정한 영역들에 대한 액세스를 제한하는 액세스 허가를 규정한다. 특권을 갖지 않는 코드를 관리하고 있는 공격자는, 캐시 타이밍 부채널을 인용하여, 공격자가 액세스하지 못하는 메모리의 특권 영역에 있는 민감한 정보에 액세스하려고 시도할 수도 있다.

[0057] 캐시 타이밍 부채널을 뒷받침하는 기본적인 원리는, 캐시 내부의 할당의 패턴, 특히 할당에 대해 어떤 캐시 세트들이 사용되었는지가, 캐시 내부에 이전에 있었던 엔트리들을 액세스하는데 걸린 시간을 측정하거나, 할당된 엔트리들을 액세스하기 위한 시간을 측정함으로써 결정될 수 있다. 그후, 이것을 이용하여 어떤 어드레스들이 캐시 내부에 할당되었는지 판정할 수 있다.

[0058] 추측 기반의 캐시 타이밍 부채널의 신규함은 추측 메모리 판독의 이용이다. 추측 메모리 판독은 고급 마이크로프로세서와 매우 고성능을 가능하게 하는 전체적인 성능의 일부를 대표한다. 아키텍처 상 해결되지 않은 분기(또는 다른 프로그램 흐름의 변경)를 넘는 캐시에 기억가능한 위치에 대한 추측적 메모리 판독을 행함으로써, 이들 판독의 결과 그 자체를 사용하여 또 다른 추측적 메모리 판독의 어드레스들을 형성한다. 추측 판독은, 제1 추측 판독의 값을 나타내는 어드레스들을 갖는 캐시 내부에의 엔트리들의 할당을 일으킨다. 이것은 신뢰하지 않은 코드가 이 신뢰받지 않은 코드에서 그렇지 않은 경우 액세스가 불가능한 위치의 제1 추측 판독을 일으키도록 신뢰받지 않은 코드가 추측을 제어할 수 있는 경우 이용가능한 부채널이 된다. 그러나, 캐시 내부의 제2 추측 할당의 효과는 이 신뢰받지 않은 코드에 의해 측정될 수 있다.

[0059] 모든 형태의 관리 소프트웨어에 대해, 신뢰받지 않는 소프트웨어가 오프셋으로 사용할 데이터 값을 신뢰된 소프트웨어에 의해 액세스되는 어레이 또는 이와 유사한 구조 내부에 전달하는 것이 일반적이다. 예를 들어, (신뢰받지 않는) 어플리케이션이 파일 기술자 ID에 근거하여 개방된 파일에 대한 정보를 요청한다. 물론, 관리 소프트웨어는, 오프셋의 사용 전에 오프셋이 적절한 범위 내에 있는지 검사하므로, 이와 같은 패러다임에 대한 소프트웨어는 다음과 같은 형태로 작성될 수도 있다:

```
1   struct array {
2      unsigned long length;
3      unsigned char data[];
4    };
5    struct array *arr = ...;
6    unsigned long untrusted_offset_from_user = ...;
7    if (untrusted_offset_from_user < arr->length) {
8      unsigned char value;
9      value =arr->data[untrusted_offset_from_user];
10       ...
11    }
```

[0060]

[0061] 최근의 마이크로프로세서에서는, 프로세서 구현이 보통 (상기한 코드의 라인 9가 암시하는) 추측적으로 데이터 액세스를 행하여 (라인 7이 암시하는) untrusted_offset_from_user 범위 검사와 관련된 분기를 실행하기 전에 값을 확립할 수도 있다. (OS 커널 또는 하이퍼바이저 등의) 관리자 레벨에서 이 코드를 실행하는 프로세서는 신뢰받지 않는 소프트웨어에 의해 전달된 untrusted_offset_from_user에 대한 범위 밖의 값에 의해 결정된 이 관리자 레벨이 액세스가능한 통산 메모리의 어딘가에서 추측적으로 로드할 수 있다. 추측이 올바르지 않으면, 로드된 값이 하드웨어에 의해 폐기되므로, 이것은 아키텍처 상의 문제가 아니다.

[0062] 그러나, 최신 프로세서는, 추가적인 추측을 위해 추측적으로 로드된 값을 이용할 수 있다. 추측 기반의 캐시 타이밍 부채널에 의해 이용되는 것이 이와 같은 추가적인 추측이다. 예를 들어, 이전의 예는 다음과 같은 형태를 갖도록 확장될 수도 있다.

```
1 struct array {
2   unsigned long length;
3   unsigned char data[];
4 };
5 struct array *arr1 = ...; /* small array */
6 struct array *arr2 = ...; /*array of size 0x400 */
7 unsigned long untrusted_offset_from_user = ...;
8   if (untrusted_offset_from_user < arr1->length) {
9     unsigned char value;
10    value =arr1->data[untrusted_offset_from_user];
11    unsigned long index2 =((value&1)*0x100)+0x200;
12    if (index2 < arr2->length) {
13      unsigned char value2 = arr2->data[index2];
14    }
15 }
```

[0063]

[0064] 이 예에서, untrusted_offset_from_user(라인 10)와 결합된 arr1->data로부터 계산된 어드레스를 사용하여 메모리에서 로드되는 value가 추가적인 메모리 액세스의 기초로서 사용된다(라인 13). 따라서, value2의 추측적 로드는 value에 대해 추측적으로 로드된 데이터로부터 유도되는 어드레스로부터 발생된다. 프로세서에 의한 value2의 추측적 로드가 캐시 내부에의 할당을 일으키는 경우, 이 로드의 어드레스의 일부를 표준의 캐시 타이밍 부채널을 이용하여 추론할 수 있다. 이 어드레스는 value에 있는 데이터에 의존하기 때문에, value의 데이터의 일부를 부채널을 이용하여 추론할 수 있다. 이와 같은 접근방법을 value의 다른 비트들에 적용함으로써, (다수의 추측 실행에서) value의 데이터 전체가 결정될 수 있다.

[0065] 도 2는 이와 같은 종류의 공격을 모식적으로 나타낸 도면이다. 도 2의 예에서, 변수 x는 더 낮은 특권 레벨 EL0에서 동작하는 신뢰되지 않은 코드에서 얻어지는 전술한 untrusted_offset_from_user에 대응한다. 변수 y는, EL0가 액세스 불가능하지만 공격자가 액세스하기 원하는 비밀 정보를 액세스하는 것이 허용되는 더 높은 특권 레벨 EL1에서 동작하는 코드에 의해 로드되는 상기한 예의 value에 대응한다. 변수 x는 array 1의 사이즈를 나타내는 사이즈 파라미터와 비교하고, 신뢰되지 않은 파라미터가 어레이 사이즈보다 큰 경우에는, (상기한 예에서 각각 라인 10 및 13에 있는 로드에 대응하는) 후속하는 로드 명령 LD를 지나 조건부 분기가 분기된다.

나중에 신뢰되지 않은 변수 x가 범위를 벗어난 것으로 판정되더라도, 조건부 분기가 분기를 타지 않는 결과를 결정하든 것으로 가정하여 이들 로드가 추측적으로 실행된다. 이것은 공격자가 비밀의 어드레스 상에 #a+x 매핑을 하도록 "x"를 선택한 경우에, 경계를 벗어난(out-of-bounds) 어드레스 #a+x에 대한 로드가 공격자가 액세스하지 않아야 하는 비밀 정보를 로드할 수 있게 한다. 그후, 두 번째 로드는 비밀의 로드에 근거하여 선택된 어드레스에 있는 두 번째 어레이 array2로부터 값을 로드한다. 이와 같은 두 번째 로드는 캐시 할당의 변화를 일으키고, 그후 이것은 더 낮은 특권 레벨(EL0)에서 동작하는 더 적은 특권을 갖는 코드에 의해 이용될 수 있으며, 이 코드는 캐시 타이밍 분석을 이용하여 두 번째 어레이의 어떤 특정한 어드레스가 캐시되었는지 탐색하여 비밀 정보를 추론한다.

[0066] 이 때문에, 신뢰되지 않은 소프트웨어는, untrusted_offset_from_user(x)에 대한 범위를 벗어난 양을 제공함으로써, 관리자 소프트웨어가 액세스가능한 모든 위치를 액세스할 수 있으므로, 이와 같은 접근방법을 신뢰되지 않은 소프트웨어가 사용하여 관리자 소프트웨어가 액세스가능한 메모리의 값을 복원할 수 있다.

[0067] 최근의 프로세서는 명령 캐시, 데이터 캐시 및 분기 예측 캐시를 포함하는 복수의 다른 종류의 캐시를 갖는다. 이들 캐시 내부의 엔트리들의 할당이 신뢰되지 않은 입력에 근거하여 로드된 일부 데이터의 어떤 부분의 값에 의해 결정되는 경우, 원리상 이와 같은 부채널이 활성화될 수도 있다.

[0068] 이와 같은 메카니즘의 일반화로서, 기반을 이루는 하드웨어 기술은, 분기를 지나는 코드가 추측적으로 실행될 수도 있으므로, 분기 후의 메모리를 액세스하는 시퀀스가 추측적으로 실행된다는 것을 의미한다는 것을 알 수 있다. 이와 같은 추측에서, 추측적으로 로드된 한 개의 값이 추측적으로 행해질 수 있는 두 번째 로드 또는 간접적인 분기에 대한 어드레스를 구성하는데 사용되는 경우, 이 두 번째 로드 또는 간접 분기는, 그렇지 않았다던 첫 번째 추측 로드에 의해 로드된 값을 판독할 수 없게 되는 코드에 의한 캐시의 타이밍 분석을 이용하여 판독될 수 있도록 이 값의 암시를 남길 수 있다. 이와 같은 일반화는, 널리 작성되는 다수의 코드 시퀀스가 다른 더 적은 특권을 갖는 소프트웨어에 의해 판독될 수도 있는 캐시 할당들의 패턴 내부에 정보를 누설하게 된다는 것을 의미한다. 이와 같은 문제의 가장 심각한 형태는, 더 적은 특권을 갖는 소프트웨어가 이와 같은 식으로 어떤 값들이 누설되는지 선택할 수 있는, 이와 같은 섹션에서 전술한 것이다.

[0069] 이와 같은 부채널을 Linux 커널에 포함된 eBPF 바이트코드 인터프리터 또는 JIT 엔진을 이용하여 커널 공간에서 실행되는 코드를 사용하여 다수의 프로세서에 대해 증명하였다. 이와 같은 방식으로 실행된 코드는 추측적으로 로드된 데이터의 필요한 시프트와 역참조를 행하는 루틴을 유지한다. 이와 같은 메카니즘의 사용은 직접적으로 이용될 수 있는 커널 공간 내의 적절한 루틴을 탐색할 필요성을 없앴다.

[0070] 이때, 이것은 추측을 이용하는 한가지 예시적인 방법인 것을 알 수 있다. 코드의 분석은, 이와 같은 메카니즘을 사용하여 의미있는 양의 정보가 검색될 수 있을 정도로 신뢰되지 않은 오프셋을 이용하여 로드된 값 그 자체를 사용하여 어드레스를 형성하는 적은 수의 위치가 존재한다는 것을 나타내었다.

[0071] 프로세서가 해결되지 않은 분기를 지나 추측하는 것은 일반적이므로, 이와 같은 거동은 비순차적으로 실행을 행하는 캐시된 프로세서 상에서 관찰되지 쉽다. 순차적으로 실행을 행하는 일부 프로세서에 대해서는, 불충분한 추측 실행이 존재하여 이와 같은 접근방법을 사용하여 캐시 내부에의 필요한 할당을 일으킬 수 있다.

[0072] 누설되고 있는 값이 더 적은 특권을 갖는 소프트웨어에 의해 결정되는 시나리오에 대한 실제적인 소프트웨어 경감은, 비밀을 유도한 액세스가 비추측적으로 실행되는 것일 때에만 비밀로부터 유도된 어드레스(이것은 상기한 예에서 value2를 로드하는데 사용되는 어드레스이다)가 비밀(value 내부의 데이터)을 표시하도록 보장하는 것이다.

[0073] 이것은, 분기의 결과를 결정하기 위해(즉, 이전의 예에서는, untrusted_offset_from_user를 삭제하기 위해) 사용되는 조건에 근거하여 조건부 선택 또는 조건부 이동 명령을 이용하여 일부 구현예에서 달성될 수 있다. 이것이 작용하지 않는 구현예에서는, (후술하는) 새로운 장벽을 이용할 수 있다(이와 같은 명령은 조건부 선택/조건부 이동이 사용될 수 있는 구현예에서는 NOP이다). 따라서, 조건부 선택/조건부 이동과 새로운 장벽의 조합은 이와 같은 문제를 해소하는데 충분하다. 새로운 장벽의 상세에 대해서는 이하에서 설명한다.

[0074] 일반적으로, 이와 같은 부채널의 이용을 허용하는 시퀀스가 특권을 갖는 코드에 존재하는 것은 드물다. 그러나, 더 낮은 레벨의 특권에 의해 주어진 바이트 코드의 컴파일은 특권을 갖는 소프트웨어 내부에 이와 같은 시퀀스를 주입하기 위한 방안이다. 이와 같은 바이트 코드를 컴파일하는 적시(just-in-time) 컴파일러가 이들 메카니즘을 컴파일된 시퀀스들의 일부로서 사용하는 것은 매우 중요하다. 이와 같은 종류의 코드 주입 메카니즘(예를 들어, eBPF)의 제공은 이것이 실용적인 시스템에서는 디스에이블된다.

[0075] 이와 같은 문제를 겪을 수 있는 또 다른 영역은, Javascript 인터프리터 또는 Java 런타임에서 발생되는 것과 같이, 한 개의 예외 레벨 내에 소프트웨어 시행된 특권 경계들이 존재하는 경우이다. 예를 들어, 인터프리터에서는, 특권의 소프트웨어 시행의 중요한 성분이 이 예에서 나타낸 신뢰되지 않은 값들의 삭제를 포함하여, 잠재적으로 이 메카니즘의 예를 제공한다. 마찬가지로, Java 바이트 코드의 런타임 컴파일에 의해 생성된 시퀀스들이 그들의 발생된 시퀀스 내부에 제2의 해결책을 포함시킬 필요가 있다.

[0076] 이 장벽을 삽입하는 것이 비현실적인 경우에, DSB SYS 및 ISB의 조합을 삽입하는 또 다른 접근방법이 삽입되어 추측을 금지할 수 있지만, 이것은 종래의 비밀/조건부 이동 및 CSDB 장벽을 사용하는 것보다 훨씬 큰 성능을 가질 가능성이 있다.

[0077] 새로운 장벽의 첫 번째 예:

[0078] CSDB는 새로운 조건부 추측 장벽이다.

[0079] 장벽이 완료할 때까지:

[0080] 1) 프로그램 순서에서 장벽 이후에 출현하고 조건부 선택 명령의 결과에 대한 어드레스 의존성을 갖는 로드, 스토어, 데이터 또는 명령 프리로드 RW2에 대해,

[0081] i. 조건부 선택 명령이 입력 레지스터들 중에서 한 개에 대해 추측적으로 실행된 로드 R1에 대한 레지스터 데이터 의존성을 갖고,

[0082] ii. 조건부 선택 명령이 그것의 나머지 입력 레지스터에 대해 R1에 대한 레지스터 의존성을 갖지 않고,

[0083] iii. 조건부 선택 명령에 대한 조건이, R1이 아키텍처 상 실행되지 않으면 R1에 의존하지 않는 입력이 선택되도록 하는 것인 경우,

[0084] RW2의 추측 실행이, 캐시 내부의 어느 엔트리들이 할당되었는지 또는 퇴출되었는지의 평가에 의해 R1으로부터 추측적으로 로드된 데이터 값의 일부를 결정하기 위해 사용될 수 있도록 캐시 내부의 엔트리들의 할당에 영향을 미치지 않는다.

[0085] 2) 프로그램 순서에서 장벽 이후에 출현하며 조건부 선택 명령의 결과에 대한 레지스터 의존성을 갖는 타겟 어드레스를 갖는 간접 분기(B2)에 대해,

[0086] i. 조건부 선택 명령이 입력 레지스터들 중에서 한 개에 대해 추측적으로 실행된 로드 R1에 대한 레지스터 데이터 의존성을 갖고,

[0087] ii. 조건부 선택 명령이 그것의 나머지 입력 레지스터에 대해 R1에 대한 레지스터 의존성을 갖지 않고,

[0088] iii. 조건부 선택 명령에 대한 조건이, R1이 아키텍처 상 실행되지 않으면 R1에 의존하지 않는 입력이 선택되도록 하는 것인 경우,

[0089] B2의 추측 실행이, 캐시 내부의 어느 엔트리들이 할당되었는지 또는 퇴출되었는지의 평가에 의해 R1으로부터 추측적으로 로드된 데이터 값의 일부를 결정하기 위해 사용될 수 있도록 캐시 내부의 엔트리들의 할당에 영향을 미치지 않는다.

[0090] 장벽은 추측적으로 실행될 수 없지만, 추측이 아닌 것이 알려지면 완료될 수 있다.

[0091] 새로운 장벽의 두 번째 예:

[0092] CSDB는 새로운 조건부 추측 장벽이다.

[0093] 장벽이 완료할 때까지:

[0094] 1) 프로그램 순서에서 장벽 이후에 출현하고 조건부 이동 명령의 결과에 대한 어드레스 의존성을 갖는 로드, 스토어, 데이터 또는 명령 프리로드 RW2에 대해,

[0095] i. 조건부 이동 명령이 그것의 입력 레지스터에 추측적으로 실행된 로드 R1에 대한 레지스터 데이터 의존성을 갖지 않고,

[0096] ii. 조건부 이동 명령에 대한 조건이, R1이 아키텍처 상 실행되지 않으면 조건이 통과하도록 하는 것인 경우,

[0097] RW2의 추측 실행이, 캐시 내부의 어느 엔트리들이 할당되었는지 또는 퇴출되었는지의 평가에 의해 R1으로부터

추측적으로 로드된 데이터 값의 일부를 결정하기 위해 사용될 수 있도록 캐시 내부의 엔트리들의 할당에 영향을 미치지 않는다.

[0098] 2) 프로그램 순서에서 장벽 이후에 출현하며 조건부 이동 명령의 결과에 대한 레지스터 의존성을 갖는 타겟 어드레스를 갖는 간접 분기(B2)에 대해,

[0099] i. 조건부 이동 명령이 그것의 입력 레지스터에 추측적으로 실행된 로드 R1에 대한 레지스터 데이터 의존성을 갖지 않고,

[0100] ii. 조건부 이동 명령에 대한 조건이, R1이 아키텍처 상 실행되지 않으면 R1에 의존하지 않는 입력이 선택되도록 하는 것인 경우,

[0101] B2의 추측 실행이, 캐시 내부의 어느 엔트리들이 할당되었는지 또는 퇴출되었는지의 평가에 의해 R1으로부터 추측적으로 로드된 데이터 값의 일부를 결정하기 위해 사용될 수 있도록 캐시 내부의 엔트리들의 할당에 영향을 미치지 않는다.

[0102] 장벽은 추측적으로 실행될 수 없지만, 추측이 아닌 것이 알려지면 완료될 수 있다.

[0103] 장벽의 이용

[0104] 이들 예는 프로세서 상에서 실행된 어셈블리 코드 내에서 장벽이 사용되는 방법을 나타내고 있다.

[0105] 이전에 나타낸 예를 들면:

```
struct array {
  unsigned long length;
  unsigned char data[];
};
struct array *arr1 = ...; /* small array */
struct array *arr2 = ...; /* array of size 0x400 */
unsigned long untrusted_offset_from_user = ...;
  if (untrusted_offset_from_user < arr1->length) {
    unsigned char value;
    value = arr1->data[untrusted_offset_from_user];
    unsigned long index2 = ((value&1)*0x100)+0x200;
    if (index2 < arr2->length) {
      unsigned char value2 = arr2->data[index2];
    }
}
```

[0106]

[0107] 첫 번째 예에서, 이것은 다음 형태의 어셈블리로 컴파일된다:

```
  LDR X1, [X2] ; X2 is a pointer to arr1->length
  CMP X0, X1 ; X0 holds untrusted_offset_from_user
  BGE out_of_range
  LDRB W4, [X5,X1] ; X5 holds arr1->data base
  AND X4, X4, #1
  LSL X4, X4, #8
  ADD X4, X4, #0x200
  CMP X4, X6 ; X6 holds arr2->length
  BGE out_of_range
  LDRB X7, [X8, X4] ; X8 holds arr2->data base
out_of_range
```

[0108]

[0109] 이것을 다음과 같이 변경함으로써 이 경우에 부채널이 경감될 수 있다:

```
  LDR X1, [X2] ; X2 is a pointer to arr1->length
  CMP X0, X1 ; X0 holds untrusted_offset_from_user
  BGE out_of_range
  LDRB W4, [X5,X1] ; X5 holds arr1->data base
  CSEL X4, XZR, X4, GE
  CSDB ; this is the new barrier
  AND X4, X4, #1
  LSL X4, X4, #8
  ADD X4, X4, #0x200
  CMP X4, X6  ; X6 holds arr2->length
  BGE out_of_range
  LDRB X7, [X8, X4] ; X8 holds arr2->data base
out_of_range
```

[0110]

[0111] 두 번째 예에서, 동등한 코드가 다음과 같이 된다:

[0112] 원본 코드:

```
  LDR R1, [R2] ; R2 is a pointer to arr1->length
  CMP R0, R1 ; R0 holds untrusted_offset_from_user
  BGE out_of_range
  LDRB R4, [R5,R1] ; R5 holds arr1->data base
  AND R4, R4, #1
  LSL R4, R4, #8
  ADD R4, R4, #0x200
  CMP R4, R6  ; R6 holds arr2->length
  BGE out_of_range
  LDRB R7, [R8, R4]; R8 holds arr2->data base
out_of_range
```

[0113]

[0114] 경감이 가해진 코드:

```
  LDR R1, [R2] ; R2 is a pointer to arr1->length
  CMP R0, R1 ; R0 holds untrusted_offset_from_user
  BGE out_of_range
  LDRB R4, [R5,R1] ; R5 holds arr1->data base
  MOVGE R4, #0
  CSDB
  AND R4, R4, #1
  LSL R4, R4, #8
  ADD R4, R4, #0x200
  CMP R4, R6  ; R6 holds arr2->length
  BGE out_of_range
  LDRB R7, [R8, R4]; R8 holds arr2->data base
out_of_range
  out_or_range
```

[0115]

[0116] 이와 같은 부채널이 데이터 캐시, 명령 캐시 또는 분기 예측 캐시에 발생되는 것을 방지하기 위해,

[0117] · 데이터 어드레스가 신뢰되지 않은 오프셋으로부터 판독된 값으로부터 결정될 때,

[0118] · 간접 분기 목적지가 신뢰되지 않은 오프셋으로부터 판독된 값으로부터 결정될 때,

[0119] · 분기 판정이 신뢰되지 않은 오프셋으로부터 판독된 값으로부터 결정될 때,

[0120] 이와 같은 접근방식이 사용될 수 있다.

[0121] 신뢰되지 않은 값의 사용을 포함하는 특정한 코드 시퀀스에 적용될 때, 이와 같은 경감은 이 코드 시퀀스가 이 부채널을 이용하여 데이터를 액세스할 수 없도록 방지한다.

[0122] 일부, 그러나 전부는 아닌, 구현예에 대해, 디바이스 메모리에 있는 암호화 키 등의 특히 중요한 비밀을 매핑하는 것은 캐시 내부에 할당되지 않게 한다. 운영체계 하에서 실현가능한 경우, 이와 같은 방식으로 이 데이터를 매핑하는 것은, 비록 상당히 증가한 성능 비용이 들더라고, 이들 구현예에 대한 추가적인 보호장치로서 사용될 수도 있다.

[0123] 이 때문에, 도 3에 나타낸 것과 같이, 추측 장벽 명령의 처리방법이 제공된다. 스텝 50에서는, 추측 장벽 명령이 디코딩된다. 이에 응답하여, 스텝 52에서, 명령 디코더(10)는 파이프라인의 나머지 스테이지들을 제어하여, 프로그램 순서에서 조건부 장벽 명령 CSDB 다음에 출현하고 프로그램 순서에서 장벽에 선행하는 이전 명령에 대해 어드레스 의존성을 갖는 후속 연산이 캐시(이 캐시는 데이터 캐시930), 명령 캐시(8), 분기 예측기(4) 내부의 분기 예측 캐시, 또는 예를 들어 수치 예측기 캐시 또는 로드/스토어 에일리어싱 예측기 등의 다른 캐시일 수 있다) 내부의 엔트리들의 할당에 추측적으로 영향을 미치지 않게 한다. 캐시 할당에 대한 추측적 영향의 금지는, 후속 명령의 추측 실행을 금지하거나, 추측 실행은 허용하지만, 이것이 캐시의 갱신을 금지하거나 예를 들어 특권 레벨에 축소가 있는 경우 삭제될 수 있는 캐시의 별개의 추측 영역 내부에의 캐시된 엔트리들의 할당을 금지함으로써(추측이 올바른 것으로 해결될 때 데이터가 추측 영역으로부터 주 영역 내부로 전달된다), 행해질 수 있다. 장벽이 완료할 때까지, 즉 장벽에 선행하는 이전 명령들과 관련된 추측이 해결될 때까지, 장벽에 뒤따르는 후속 명령들에 대해 캐시에 대한 추측적 영향의 제약이 지속된다.

[0124] 추측 장벽 명령 CSDB의 또 다른 예는 이하에서 설명한다. 장벽의 의미론은, 프로그램 순서에서 CSDB 이후에 출현하는 분기 명령 이외의 명령의 다음 중에서 어느 결과를 이용하여 추측적으로 실행될 수 없다는 것이다:

[0125] 명령의 데이터 값 예측, 또는

[0126] 조건부 분기 명령 이외의 명령에 대한 조건 코드들(42)의 예측, 또는

[0127] 프로그램 순서에서 아키텍처 상 해결되지 않은 CSDB 이전에 출현하는 벡터 명령에 대한 벡터 프레디케이션 상태(44)의 예측.

[0128] CSDB의 정의를 위해, 조건 코드(42) 및 벡터 프레디케이트 값(44)은 데이터 값인 것으로 생각하지 않는다.

[0129] 이와 같은 정의는, 프로그램 순서에서 아키텍처 상 해결되지 않은 CSDB 이전에 출현하는 명령들의 데이터 값 또는 조건 코드의 결과를 사용하지 않는 경우, CSDB 이전 및 이후의 제어 흐름 추측 및 CSDB 이후의 조건부 데이터 처리 명령의 추측 실행을 허가한다. 전술한 코드 예는 장벽 명령의 예에서도 사용될 수 있다.

[0130] 추측의 또 다른 예는 (프로그램 순서 내에서 어느 한쪽의 순서로 발생하는) 로드 명령 및 스토어 명령이 동일한 어드레스를 액세스하는지 여부에 대한 추측이다. 다수의 최근의 고성능 프로세서에서는, 성능 최적화가 행해짐으로써, 어드레스에 대한 로드가 타겟 어드레스가 하드웨어에 의해 아직 알려지지 않지만 실제로는 로드의 어드레스와 동일한 이전의 스토어를 추측적으로 우회하게 된다. 이것이 발생하면, 로드가 스토어에 의해 기록된 값보다 이 어드레스에 있는 데이터의 이전의 값을 추측적으로 판독하게 된다. 그후, 이 추측적으로 로드된 값이 캐시 내부에의 할당을 일으키는 후속하는 추측적인 메모리 액세스를 위해 사용될 수 있으며, 이들 할당의 타이밍이 어드레스로서 선택된 데이터 값들에 대한 관측 부채널로서 사용될 수 있다. 원리상, 향상된 비순차 프로세서에서는, 다음과 같은 형태를 갖는 코드 시퀀스에 있어서,

```
STR X1, [X2]
...
LDR X3, [X4] ; X4 contains the same address as X2
<arbitrary data processing of X3>
LDR X5, [X6, X3]
```

[0131]

[0132] 이 시퀀스의 두 번째 로드가 첫 번째 로드로부터 X3에서 추측적으로 반환된 값으로부터 유도되었던 X3에 대한 값을 이용하여 추측적으로 행해질 수도 있다. 이 추측적으로 로드된 값은 이 값을 오버라이트한 STR보다 프로그램 실행에 있어서 더 일찍부터인 첫 번째 어드레스에 유지된 값으로부터 취해질 수도 있다. 두 번째 로드의 추측 실행에 의해 발생된 캐시 할당은 X3 내부에 추측적으로 로드된 이와 같은 이전 데이터에 대한 정보를 누설하게 된다. 이것을 공격자가 이용하여 이 값의 발견을 방지하기 위해 일부의 이전 데이터를 스토어가 오버라이트하고 있는 상황을 우회할 수도 있다. 이와 같은 추측적인 우회 접근방법은 추측적인 로드의 체인을 통해 확장되므로, 이 경우에

```
STR X1, [X2]
...
LDR X3, [X4] ; X4 contains the same address as X2
<arbitrary data processing of X3>
LDR X5, [X6, X3]
<arbitrary data processing of X5>
LDR X7, [X8, X5]
```

[0133]

[0134] 이 시퀀스의 두 번째 및 세 번째 로드가 이 값을 오버라이트한 STR보다 실행시에 더 일찍부터인 첫 번째 어드레스에 유지된 값으로부터 취해진 X3에 대한 값을 이용하여 추측적으로 행해질 수도 있다. 세 번째 로드의 추측 실행에 의해 발생된 캐시 할당은 X5에 있는 데이터에 대한 일부 정보를 누설하게 된다. 이 경우, 공격자가 X2 및 X4를 가리키는 어드레스에 유지된 이전 값을 통제하면, 이것이 후속 추측에 영향을 미쳐, 두 번째 추측적인 로드에 의한 데이터의 선택과, 세 번째 로드에 의해 기인한 캐시 할당의 검사에 의한 선택된 데이터의 누설을 허용할 수 있다.

[0135] 스토어 및 첫 번째 로드가 동일한 가상 및 물리 어드레스에 대한 것인 경우, 이와 같은 종류의 추측적 재배열은 한 개의 예외 레벨 내에서만 발생할 수 있다.

[0136] 스토어 및 첫 번째 로드가 서로 다른 가상 어드레스들이면서 동일한 물리 어드레스에 대한 것인 경우, 추측적 재배열이 서로 다른 예외 레벨들에 있는 코드 사이에서 발생할 수 있으므로, 이 경우에는:

```
STR X1, [X2]
...
ERET ; exception return to a lower level
...
LDR X3, [X4] ; X4 contains a different virtual address as X2, but the same
physical address
<arbitrary data processing of X3>
LDR X5, [X6, X3]
```

[0137]

[0138] 오프셋으로서 X3를 사용하여 캐시 내부에 추측적으로 로드된 위치는 X2 및 X4가 가리키는 물리 어드레스에 있었던 이전의 데이터 값을 표시할 수 있다.

[0139] 최근의 고성능 프로세서에서는, 로드의 이용가능성에 대해, 예를 들어, 스토어 어드레스의 발생시의 캐시 부적중의 결과로써, 스토어의 어드레스가 그것의 이용가능성에서 지연되는 경우, 동일한 어드레스에 대한 스토어 및 그후의 로드의 재배열과, 이와 같은 로드에 의한 오래된 데이터의 추측 판독을 나타내는 것이 비교적 수월하다.

[0140] 스토어 및 로드가 동일한 레지스터들을 이용하여 어드레스를 전달하는 경우, 프로세서는 보통 동일한 어드레스에 대한 이전의 스토어에 앞서서 로드를 추측적으로 실행하지 않는다. 그러나, 일부의 마이크로 아키텍처 상 특정한 경우에, 그것이 일부 구현예에서는 원리상 가능하다. 이와 같은 재배열에 대한 정확한 조건은 일

반적으로 프로세서에 의해 처리되고 있는 이전의 메모리 액세스의 지연의 복합 함수이다.

[0141] 이와 같은 메카니즘의 특정한 문제는, 코드에서 비교적 일반적인 패턴이기 때문에, 스토어 및 첫 번째 로드가 (동일한 어드레스를 갖는 스택 포인터 또는 다른 레지스터들을 사용하여) 스택 상에 액세스하는 경우일 것이다. 원칙적으로, 이것은 스택 상에 있었지만 오버라이트되었던 이전 값이 프로세서의 후속 추측을 제어하게 하는 메카니즘을 제공할 수도 있다. 특권을 갖는 스택에 대해, 스택 상에 있었던 이전 값은 실제로 더 적은 특권을 갖는 실행의 제어하에 있을 수도 있다.

[0142] 다음 시퀀스에서:

[0143]
```
STR X1, [SP]
...
LDR X3, [SP]
<arbitrary data processing of X3>
LDR X5, [X6, X3]
<arbitrary data processing of X5>
LDR X7, [X8, X5]
```

[0144] 이것은 두 번째 로드를 사용하여 프로세서의 더 큰 특권을 갖는 어드레스 공간 어드레스들의 어딘가에 있는 데이터의 추측 로드를 향하게 하기 위해 (아마도 일부 데이터의 처리를 요구하는 시스템 호출의 결과로써) 스토어 이전의 스택에 유지되었던 값을 결정한 더 적은 특권을 갖는 코드에 대해 제어 채널을 제공할 수도 있다. 이 두 번째 로드의 결과는, 그것이 캐시 할당을 일으키는 세 번째 로드의 어드레스를 형성하는데 사용된다는 사실에 의해 관찰가능해진다. 이 캐시 할당의 존재는 이들 부채널 모두에 적용되는 것과 동일한 방식으로 전통적인 캐시 타이밍 분석에 의해 검출될 수 있다. 원칙적으로, 이것은 타이밍 부채널을 이용하여 더 적은 특권을 갖는 로드에 의한 임의의 특권을 갖는 데이터의 판독을 허용할 수도 있다.

[0145] 마찬가지로, 이와 같은 예에서 나타내는 것과 같이, 스택이 함수 포인터와 함께 재사용되어 임의의 코드의 선택이 더 큰 특권을 갖는 어드레스 공간에서 추측적으로 실행될 수 있게 할 수도 있다:

[0146]
```
STR X1, [SP]
...
LDR X3, [SP]
...
BLR X3.
```

[0147] 원칙적으로, 이것은 추측 도구의 선택이 흥미로운 데이터를 누설하게 할 것이다.

[0148] 적어도 일부 구현 상에서 나타날 수도 있는 이와 같은 거동의 또 다른 형태는, 이 시퀀스에서 볼 수 있는 것과 같이, 비순차 프로세서가 로드를 명령 스트림의 나중의 스토어로부터 데이터를 추측적으로 반환하게 할 수 있는 것이다:

[0149]
```
...
LDR X3, [X4]
<arbitrary data processing of X3>
LDR X5, [X6, X3]
...
STR X1, [X2] ; X2 contains the same address as X4
```

[0150] 이것이 발생하는 경우, 두 번째 로드에 의한 캐시 내부의 할당이 캐시 타이밍 부채널의 이용에 의해 나중에 기억된 값의 관찰을 일으킬 수도 있다.

[0151] 단순한 개념 증명이 일부 구현예에 대해 입증되었는데, 이때 스토어가 동일한 어드레스에 대한 나중의 로드에 대해 그것의 어드레스가 지연되어, 전술한 종류의 나중의 추측적 메모리 액세스를 발생한다. 이들 추측적인 메모리 액세스는, 타이밍 부채널을 이용하여, 기억되거나 로드되는 메모리 위치에 유지된 이전 값의 결정에 의해 선택된 데이터의 값을 누설할 수 있는 캐시 내부의 할당을 일으킨다. 이것은 개념을 증명하기 위해 맞

춤형 코드를 이용하여 입증하였다.

[0152] 스택에 액세스할 때 보통 발생하는 것과 같이, 특히 스토어 어드레스가 로드 어드레스 이전에 또는 로드 어드레스와 동일한 시간에 이용가능한 이와 같은 형태의 우회의 더욱 일반적인 경우를 설명하지 않았으며, 유저 코드가 이전의 메모리 액세스를 지연시켜 필요한 재배열에 대해 프로세서가 이와 같은 데이터를 누설하게 하기 위한 필요한 복잡한 조건을 보장하는 것은 매우 어려울 것이다. 그러나, 이와 같은 메카니즘이 더 큰 특권을 갖는 메모리로부터 데이터를 판독하기 위한 저대역 채널로서 이용가능할 수도 있다는 것을 배제하는 것은 불가능하다.

[0153] 로드에 의한 나중의 스토어를 관찰하는 메카니즘을 설명하지 않았지만, 적어도 일부 구현상에서는 가능할 것으로 생각된다.

[0154] 이와 같은 공격을 방해하기 위해 2개의 스토어 추측 장벽 명령인 SSBB 및 PSSBB가 제공될 수 있다. SSBB의 사용은, 가상 어드레스를 사용하는 SSBB 이전의 모든 스토어가 동일한 가상 어드레스에 대한 SSBB 이후의 로드의 추측 실행에 의해 우회되지 않도록 보장한다. 또한, SSBB 장벽은, 특정한 가상 어드레스에 대한 SSBB 이전의 모든 로드들이 SSBB 이후의 스토어로부터 추측적으로 로드하지 않도록 보장한다. 이와 같은 장벽은 예외 레벨 내부의 소프트웨어 관리된 특권의 경우에 추측 로드가 이 메카니즘을 사용하여 이용되지 않게 하는데 사용될 수 있다. PSSBB 장벽의 이용은, 특정한 물리 어드레스를 사용하는 PSSBB 이전의 모든 스토어들이 동일한 물리 어드레스에 대한 PSSBB 이후의 로드의 추측 실행에 의해 우회되지 않도록 보장한다. 또한, PSSBB 장벽은, 특정한 물리 어드레스에 대한 PSSBB 이전의 모든 로드가 PSSBB 이후의 스토어로부터 추측적으로 로드하지 않도록 보장한다.

[0155] 도 4는 전술한 SSBB 또는 PSSBB 명령일 수 있는 스토어 추측 장벽 명령의 처리방법을 나타낸 것이다. 스텝 60에서, 이와 같은 스토어 추측 장벽 명령이 디코드된다. 스텝 62에서, 명령 디코더910)는 처리 파이프라인을 제어하여, 프로그램 순서에서 스토어 추측 장벽 명령에 선행하는 이전 스토어 명령이, 프로그램 순서에서 스토어 장벽 명령에 뒤따르고 이전 스토어와 동일한 어드레스를 지정하는 후속하는 로드 명령의 추측 실행에 의해 우회되지 않도록 한다. SSBB 명령에 대해, "동일한 어드레스"는 동일한 가상 어드레스를 가리키는 한편, PSSBB 명령에 대해, "동일한 어드레스"는 동일한 물리 어드레스를 가리킨다.

[0156] 또한, 스텝 64에서, 스토어 추측 장벽 명령에 응답하여, 명령 디코더는 파이프라인의 후속 스테이지들을 제어하여, 프로그램 순서에서 스토어 추측 장벽 명령에 선행하는 이전의 로드 명령이, 프로그램 순서에서 스토어 추측 장벽 명령에 뒤따르고 동일한 어드레스(마찬가지로, "동일한 어드레스"는 SSBB 또는 PSSBB 변종이 디코드되는지 여부에 따라 가상 어드레스 또는 물리 어드레스일 수 있다)를 지정하는 후속하는 스토어 명령으로부터 발생된 데이터를 로드하지 못하게 한다. 이 때문에, 이와 같은 장벽은 컴파일러 또는 프로그래머가 추측 부채널 공격에 대한 보호를 제공할 수 있게 한다.

[0157] 또 다른 예의 추측은, 출력(분기를 타거나 분기를 타지 않는 것)의 예측이 분기 예측기(4)에 의한 조건부 분기 명령인 것으로, 그후 이 분기 예측기는 예측에 따라 후속하는 명령들의 추측 실행을 발생한다. 분기들은 보통 고성능 CPU 마이크로아키텍처 설계에서 추측된다. 높은 정밀도로 추측하기 위해, 이 분기의 이전의 실행에 근거하여 분기를 탈 것인지 타지 않을 것인지 예측하는 예측 테이블이 생성된다.

[0158] 일방향 분기, 즉 예측이 일방향으로만, 즉 분기를 타거나 타지 않도록 발생하지만 이들 두가지가 발생하지 않는 분기를 제공하는 것이 제안되어 있다. 이와 같은 일방향 분기는 예측이 양쪽 방향(분기를 타는 것과 타지 않는 것)으로 허용되는 양방향 분기 이외에 명령 세트 아키텍처에서 지원될 수도 있다.

[0159] 고급 코드의 다음과 같은 예를 고려하자:

[0160]
```
<pseudo-code>
if a < b {
// sensitive code, do not speculate into here.
}
// non-sensitive code
</pseudo-code>
```

[0161] 이것은 보통 다음과 같이 처리 파이프라인 상에서 실행되는 명령들의 시퀀스로 변환될 것이다:

[0162]
```
<psuedo-asm>
cmp a, b
b.ge nonsensitive_code
sensitive_code:
; something sensitive here.
nonsensitive_code:
</pseudo-asm>
```

[0163] 보통 (a<b)가 참이면, 결국 예측기가 a<b인 것을 높은 신뢰도로 예측하게 되고 민감한 코드가 실행되어야 한다고 예측한다. 이것은 전술한 것과 같은 부채널 타이밍 공격을 일으킬 수 있다.

[0164] 그러나, "bpt"가 "branch-only-predict-taken"에 대한 어셈블리 연상기호일 때 다음과 같은 코드를 제공하면:

[0165]
```
<psuedo-asm>
cmp a, b
bpt.ge nonsensitive_code
sensitive_code:
; something sensitive here.
nonsensitive_code:
</pseudo-asm>
```

[0166] 마이크로 아키텍처가 분기 예측기를 사용하여, 그것이 분기를 타고 있다는 높은 신뢰도를 가질 때를 추측할 수 있다. 분기를 타고 있다는 예측에서 불충분한 신뢰도를 갖거나 분기를 타지 않는다는 예측에서 높은 신뢰도를 갖는 경우에는, 분기가 해결될 때까지 실행을 정지할 수 있다. 이것은 민감한 코드 영역의 추측을 방지할 것이다. bpn(branch-only-predict-not-taken)의 직교적인(orthogonal) 접근방법은 나머지 방향으로의 예측을 보호할 수도 있다.

[0167] 마이크로 아키텍처는 이들 분기를 처리하는 다수의 선택을 갖는다. 이들 분기가 정규의 분기 예측기를 사용하고 이력을 계속 구축하지만 허용된 예측 방향에서 신뢰도를 갖는 경우에만 이 이력을 이용할 수 있거나, 또는 이들 분기가 모두 예측으로부터 배제될 수 있다. 결코 예측되지 않은 세 번째 형태의 분기 bnv, (branch-never-predict)도 지원될 수 있다.

[0168] 이 때문에, 도 5는 조건부 분기 명령의 이와 같은 일방향 추측 변종을 처리하는 방법을 나타낸 것이다. 스텝 100에서, 처리 파이프라인은 조건부 분기 명령의 일방향 추측 변종과 마주친다. 일방향 추측 변종은 다양하게 식별될 수 있다. 어떤 경우에, 페치 스테이지(6)는, 예를 들어, 명령들을 부분적으로 디코드하거나, 프리디코더에 의해 명령에 추가된 프리디코드 정보를 액세스하고 명령들이 명령 캐시(8) 내부에 할당될 때 프리디코더가 명령들을 해석함으로써, 직접 조건부 분기 명령의 일방향 추측 변종을 식별할 수 있다. 또 다른 예에서는, 분기 명령들이 명령 디코더910)에 의해 디코드될 때, 분기가 조건부 분기 명령의 일방향 추측 변종인 것으로 식별되는 경우, 이것이 분기 예측기(4)로 다시 신호로 보내져, 그후 분기 예측기가 주어진 명령 어드레스에 있는 명령이 일방향 추측 변종의 조건부 분기 명령인 것으로 검출된 분기 예측 상태의 대응하는 엔트리를 갱신한다. 이것은 동일한 명령의 후속 페칭시에 페치 스테이지(6)가 기억된 분기 예측 상태로부터 그 자체가 명령을 디코드할 수 있는 능력을 갖지 않더라도 이 명령이 일방향 추측 변종이라는 것을 식별할 수 있게 할 수도 있다.

[0169] 일방향 추측 변종에 응답하여, 스텝 102에서, 분기 예측기(4)는 어떤 분기 결과가 예측되는지 판정한다. 결과가 제1 결과(이것은 분기를 타거나 타지 않는 결과일 수 있다)이면, 스텝 104에서 분기 예측기(4)는 분기 이후에 실행할 다음 명령이 제1 명령이라고 예측한다. 제1 결과가 분기를 타는 결과이면, 제1 명령은 분기 명령의 분기 타겟 어드레스에 있는 명령인 반면에, 제1 결과가 분기를 타지 않는 결과이면, 제1 명령은 조건부 분기로부터 순차적으로 뒤따르는 다음 명령이다. 스텝 106에서, 제1 명령에 대해 제한적인 추측 실행이 적용된다. 예를 들어, 제1 명령이 추측적으로 실행되는 것이 금지되거나, 또는 추측 실행에 근거하여 캐시 구조의 갱신에 대한 제약을 갖고 추측적으로 실행되는 것이 허용됨으로써, 공격자가 캐시 타이밍 부채널 공격으로부

터 정보를 유도할 수 없게 할 수도 있다.

[0170] 한편, 스텝 102에서, 제2 결과(제2 결과는 제1 결과와 반대의 결과이다)가 예측되는 경우, 스텝 108에서, 다음 명령이 제2 명령(마찬가지로 제2 결과가 분기를 타지 않는 것이면 다음의 순차적인 명령 또는 제2 결과가 분기를 타는 것이면 분기 타겟 어드레스에 있는 명령)이라고 예측된다. 스텝 110에서, (제1 결과에 대해 스텝 106에서 적용되는) 제한 추측이 제2 명령에 대해서는 생략되므로, 제2 명령의 추측 실행이 제한되지 않고 진행된다.

[0171] 이 때문에, 전술한 일방향 분기에 따르면, 제2 결과가 예측될 때에는 추측이 계속하게 할 수 있어 성능을 향상시킬 수 있지만, 제1 결과가 예측될 때에는 추측을 제한하여 전술한 형태의 공격에 대해 보호할 수 있다. 일부 구현예는 일방향 분기의 한 개의 변종만, 예를 들어, 전술한 bpt 또는 bpn만을 지원한다(bpt에 대해, 도 5의 제1 결과는 분기를 타지 않는 결과이고 제2 결과는 분기를 타는 결과이며, bpn에 대해 제1 결과는 분기를 타는 결과이고 제2 결과는 분기를 타지 않는 결과이다). 다른 구현예는 bpt 및 bpn 모두를 지원한다. 이것은, 분기를 타거나 타지 않은 결과에 뒤따라 실행할 코드가 민감한 정보의 손실에 가장 취약한지에 따라 프로그래머 또는 컴파일러가 적절한 변종을 선택하게 할 수 있다. 또한, 일부 시스템은, 예측된 결과가 제1 결과인지 또는 제2 결과인지에 무관하게 추측에 대한 제한이 항상 적용되는 비추측 변종을 지원해도 된다.

[0172] 또 다른 실시예는 이하의 절에 기재되어 있다:

[0173] 1. 데이터 처리를 행하는 처리회로와, 명령을 디코드하고 처리회로를 제어하여 데이터 처리를 행하게 하는 명령 디코딩 회로를 구비하고, 상기 명령 디코딩 회로는 조건부 추측 장벽 명령에 응답하는 장치.

[0174] 2. 호스트 처리장치를 제어하여 타겟 프로그램 코드의 명령들을 실행하기 위한 명령 실행 환경을 제공하는 컴퓨터 프로그램으로서, 타겟 프로그램 코드의 명령들을 디코드하고 처리 프로그램 로직을 제어하여 데이터 처리를 행하게 하는 명령 디코딩 프로그램 로직을 포함하고, 상기 명령 디코딩 프로그램 로직은 조건부 추측 장벽 명령에 응답하는 컴퓨터 프로그램.

[0175] 3. 조건부 추측 장벽 명령을 디코드하는 단계와, 조건부 추측 장벽 명령의 디코딩에 응답하여, 처리회로를 제어하는 단계를 포함하는 데이터 처리방법.

[0176] 도 6은 사용될 수 있는 시뮬레이터 구현예를 나타낸 것이다. 전술한 실시예는 해당 기술을 지원하는 특정한 처리 하드웨어를 작동하기 위한 장치 및 방법에 관해 본 발명을 구현하지만, 컴퓨터 프로그램의 사용을 통해 구현되는 본 발명에서 설명한 실시예에 따라 명령 실행 환경을 제공하는 것도 가능하다. 이와 같은 컴퓨터 프로그램은, 하드웨어 아키텍처의 소프트웨어 기반의 구현을 제공하는 한, 시뮬레이터로 부르는 경우가 많다. 다양한 시뮬레이터 컴퓨터 프로그램은 에뮬레이터, 가상머신, 모델, 및 동적이진 변환기를 포함하는 이진 변환기를 포함한다. 보통, 시뮬레이터 구현은, 옵션으로 시뮬레이터 프로그램(220)을 지원하는 호스트 운영체계(210)를 실행하는 호스트 프로세서(200) 상에서 실행된다. 일부 구성에서는, 하드웨어와 제공된 명령 실행 환경 사이에 복수 층의 시뮬레이션이 존재하고, 및/또는 동일한 호스트 프로세서 상에서 복수의 별개의 명령 실행 환경이 제공된다. 역사적으로, 합당한 속도에서 실행되는 시뮬레이터 구현을 제공하기 위해 강력한 프로세서들이 요구되었지만, 이와 같은 접근방법은, 호환성이나 재사용 이유로 인해 다른 프로세서에 대해 네이티브한 코드를 실행하려는 요구가 있을 때 등과 같은, 특정한 상황에서 정당화된다. 예를 들어, 시뮬레이터 구현은, 호스트 프로세서 하드웨어에 의해 지원되지 않는 추가적인 기능을 갖는 명령 실행 환경을 제공하거나, 보통 다양한 하드웨어 아키텍처와 관련된 명령 실행 환경을 제공한다. 시뮬레이터의 개관에 대해서는 "Some Efficient Architecture Simulation Techniques", Robert Bedichek, Winter 1990 USENIX Conference, Pages 53-63에 기재되어 있다.

[0177] 본 실시예를 특정한 하드웨어 구성 또는 특징을 참조하여 설명하였지만, 시뮬레이션된 실시예에서는, 적절한 소프트웨어 구성 또는 특징에 의해 동등한 기능이 제공된다. 예를 들어, 특정한 회로가 시뮬레이션된 실시예에서는 컴퓨터 프로그램 논리로 구현된다. 마찬가지로, 레지스터 또는 캐시 등의 메모리 하드웨어도 시뮬레이션된 실시예에서는 소프트웨어 데이터 구조로 구현된다. 전술한 실시예에서 참조한 한 개 이상의 하드웨어 구성요소들이 호스트 하드웨어(예를 들어, 호스트 프로세서(200) 상에 존재하는 구성에서는, 적절한 경우에, 일부 시뮬레이션된 실시예가 호스트 하드웨어를 이용한다.

[0178] 시뮬레이터 프로그램(220)은, 컴퓨터 판독가능한 기억매체(이것은 비일시적인 매체일 수도 있다)에 기억되고, 시뮬레이터 프로그램(220)에 의해 모델링되고 있는 하드웨어 아키텍처의 응용 프로그램 인터페이스와 동일한 타겟 코드(230)(이것은 어플리케이션, 운영체계 및 하이퍼바이저를 포함한다)에 대한 프로그램 인터페이

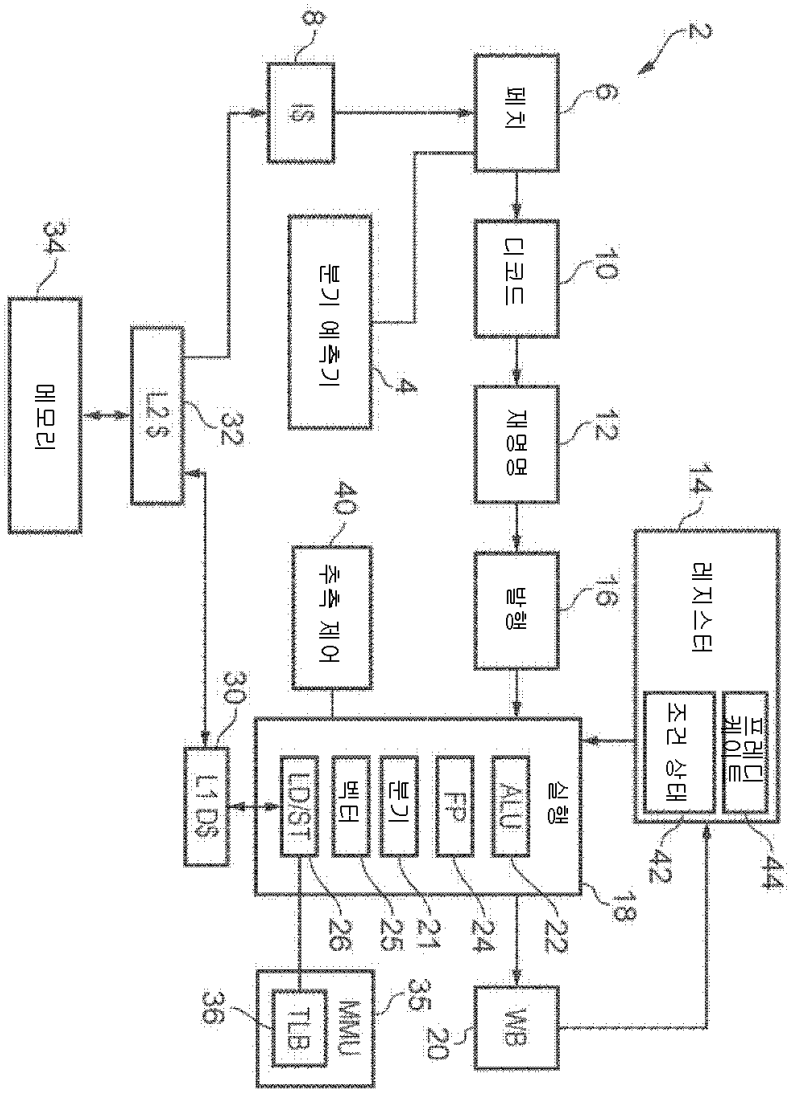
스(명령 실행 환경)를 제공한다. 따라서, 전술한 추측 장벽 명령, 스토어 추측 장벽 명령 및 조건부 분기 명령의 일방향 추측 형태를 포함하는 타겟 코드(230)의 프로그램 명령들은 시뮬레이터 프로그램(220)을 사용하여 명령 실행 환경 내에서 실행됨으로써, 전술한 장치(2)의 하드웨어 특징을 실제로 갖지 않는 호스트 컴퓨터(200)가 이들 특징을 에뮬레이트할 수 있다. 시뮬레이터 프로그램(220)은 타겟 코드(230)의 명령들을 디코드하고 이들 명령들을 호스트 하드웨어(200)에 의해 지원되는 네이티브 명령 세트로부터의 한 개 이상의 명령을 사용하여 실행된 대응하는 기능에 매핑하는 명령 디코딩 프로그램 로직(225)를 구비한다.

[0179] 본 발명에서, 단어 "하도록 구성된"은 장치의 구성요소가 정의된 동작을 행할 수 있는 구성을 갖는다는 것을 의미하기 위해 사용된다. 이와 관련하여, "구성"은 하드웨어 또는 소프트웨어의 배치 또는 상호접속 방식을 의미한다. 예를 들어, 장치는 정의된 동작을 제공하는 전용 하드웨어를 갖거나, 프로세서 또는 기타의 처리 장치가 기능을 행하도록 프로그래밍되어도 된다. "하도록 구성된"은 이 장치의 구성요소가 정의된 동작을 제공하기 위해 어떤 식으로 변경될 필요가 있는 것을 시사하는 것은 아니다.

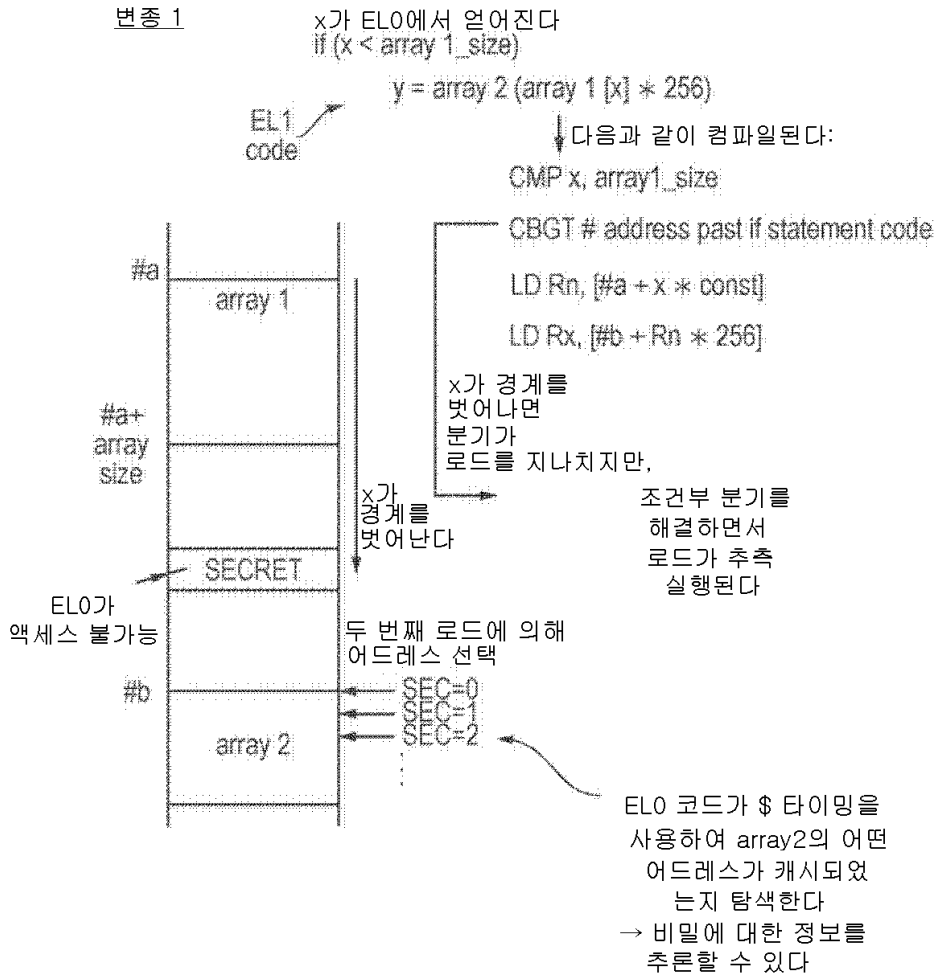
[0180] 첨부도면을 참조하여 본 발명의 예시적인 실시예들을 상세히 설명하였지만, 본 발명은 이들 실시예에 한정되지 않으며, 첨부된 청구범위의 보호범위 및 사상을 벗어나지 않으면서 본 발명이 속한 기술분야의 당업자에 의해 다양한 변경, 부가 및 변화가 행해질 수 있다는 것은 자명하다. 예를 들면, 종속항들의 특징들의 다양한 조합이 독립항들의 특징과 행해질 수도 있다.

도면

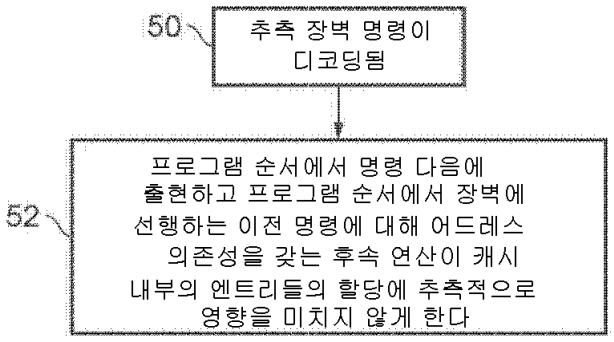
도면1



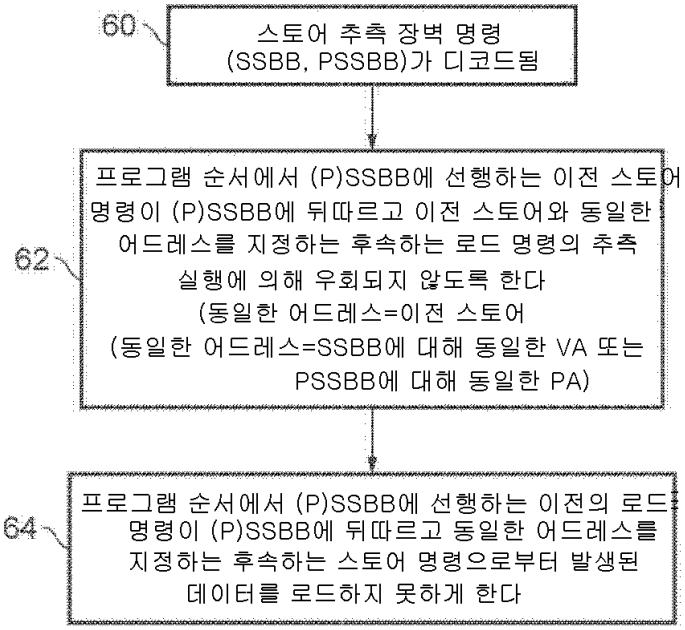
도면2


변종 1
x가 EL0에서 얻어진다
if (x < array 1_size)
y = array 2 (array 1 [x] * 256)
EL1 code
다음과 같이 컴파일된다:
CMP x, array1_size
CBGT # address past if statement code
LD Rn, [#a + x * const]
LD Rx, [#b + Rn * 256]
#a
array 1
#a+ array size
SECRET
EL0가 액세스 불가능
#b
array 2
x가 경계를 벗어난다
x가 경계를 벗어나면 분기가 로드를 지나치지만,
조건부 분기를 해결하면서 로드가 추측 실행된다
두 번째 로드에 의해 어드레스 선택
SEC=0
SEC=1
SEC=2
EL0 코드가 $ 타이밍을 사용하여 array2의 어떤 어드레스가 캐시되었는지 탐색한다
→ 비밀에 대한 정보를 추론할 수 있다

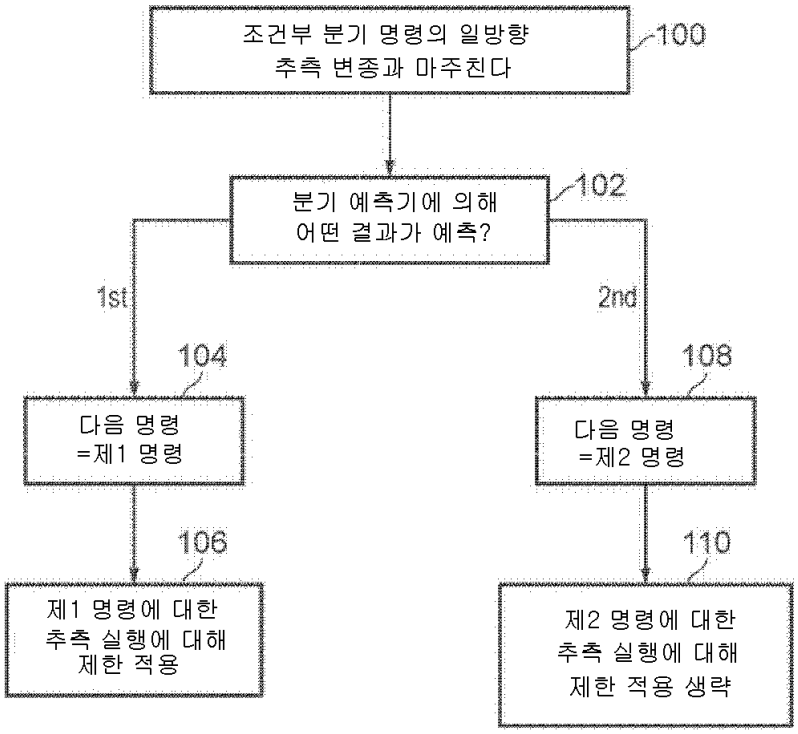
도면3


50
추측 장벽 명령이 디코딩됨
52
프로그램 순서에서 명령 다음에 출현하고 프로그램 순서에서 장벽에 선행하는 이전 명령에 대해 어드레스 의존성을 갖는 후속 연산이 캐시 내부의 엔트리들의 할당에 추측적으로 영향을 미치지 않게 한다

도면4


60
스토어 추측 장벽 명령
(SSBB, PSSBB)가 디코드됨
62
프로그램 순서에서 (P)SSBB에 선행하는 이전 스토어 명령이 (P)SSBB에 뒤따르고 이전 스토어와 동일한 어드레스를 지정하는 후속하는 로드 명령의 추측 실행에 의해 우회되지 않도록 한다
(동일한 어드레스=이전 스토어
(동일한 어드레스=SSBB에 대해 동일한 VA 또는 PSSBB에 대해 동일한 PA)
64
프로그램 순서에서 (P)SSBB에 선행하는 이전의 로드 명령이 (P)SSBB에 뒤따르고 동일한 어드레스를 지정하는 후속하는 스토어 명령으로부터 발생된 데이터를 로드하지 못하게 한다

도면5


조건부 분기 명령의 일방향 추측 변종과 마주친다
100
분기 예측기에 의해 어떤 결과가 예측?
102
1st
2nd
104
다음 명령 =제1 명령
106
제1 명령에 대한 추측 실행에 대해 제한 적용
108
다음 명령 =제2 명령
110
제2 명령에 대한 추측 실행에 대해 제한 적용 생략

도면6