(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2024-0159950
(43) 공개일자 2024년11월07일

(51) 국제특허분류(Int. Cl.)
G06F 12/1027 (2016.01) *G06F 12/1009* (2016.01)
(52) CPC특허분류
G06F 12/1027 (2013.01)
G06F 12/1009 (2013.01)
(21) 출원번호 10-2024-7033873
(22) 출원일자(국제) 2022년12월20일
심사청구일자 없음
(85) 번역문제출일자 2024년10월10일
(86) 국제출원번호 PCT/GB2022/053304
(87) 국제공개번호 WO 2023/175289
국제공개일자 2023년09월21일
(30) 우선권주장
2203646.1 2022년03월16일 영국(GB)

(71) 출원인
에이알엠 리미티드
영국 캠브리지 씨비1 9엔제이 체리힌톤 풀번로드 110
(72) 발명자
반스, 그레임 피터
영국 씨비1 9엔제이 캠브리지셔 캠브리지 풀번 로드 110 씨/오 에이알엠 리미티드
크라스케, 사이먼 존
영국 씨비1 9엔제이 캠브리지셔 캠브리지 풀번 로드 110 씨/오 에이알엠 리미티드
(74) 대리인
이화익

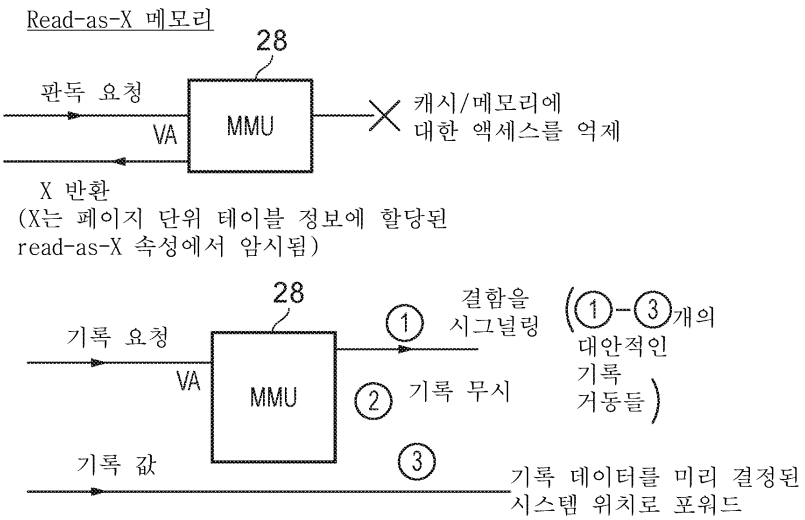
전체 청구항 수 : 총 22 항

(54) 발명의 명칭 **메모리 어드레스 공간의 페이지에 대한 read-as-X 속성**

(57) 요 약

프로세싱 회로(4)는 명령들에 응답하여 데이터 프로세싱을 수행한다. 메모리 관리 회로(28)는 메모리 어드레스 공간의 주어진 페이지 내의 어드레스에 대한 판독들이 특정된 값 X를 반환하는 것으로 취급되어야 하는 것을 나타내는 read-as-X 속성과 메모리 어드레스 공간의 주어진 페이지를 연관시킬 수 있는 페이지 테이블 정보에 기반하여 메모리에 대한 액세스를 제어한다. 메모리 어드레스 공간의 판독 타겟 블록에 대한 판독 타겟 값을 판독하기 위해 발행된 판독 요청에 대해, 판독 타겟 블록의 적어도 일부가 read-as-X 속성과 연관된 페이지에 대응한다는 결정에 응답하여, 메모리 관리 회로(28)는 특정된 값 X가 판독 타겟 값의 적어도 일부로서 프로세싱 회로(4)에 반환되게 제어한다. 이는 메모리 어드레스 공간의 큰 영역들이 이들 영역들에 대한 물리적 메모리를 커밋(commit)할 필요없이 특정된 값을 저장하는 것으로 취급될 수 있게 한다.

대 표 도 - 도3



명 세 서

청구범위

청구항 1

장치로서,

명령들에 응답하여 데이터 프로세싱을 수행하기 위한 프로세싱 회로; 및

메모리 어드레스 공간의 주어진 페이지 내의 어드레스에 대한 판독들이 특정된 값 X를 반환하는 것으로 취급되어야 하는 것을 나타내는 read-as-X 속성(read-as-X property)과 메모리 어드레스 공간의 주어진 페이지를 연관시킬 수 있는 페이지 테이블 정보에 기반하여 메모리에 대한 액세스를 제어하기 위한 메모리 관리 회로를 포함하고; 여기서

상기 메모리 어드레스 공간의 판독 타겟 블록에 대한 판독 타겟 값을 판독하기 위해 상기 프로세싱 회로에 의해 발행된 판독 요청에 대해, 상기 판독 타겟 블록의 적어도 일부가 상기 read-as-X 속성과 연관된 페이지에 대응한다는 결정에 응답하여, 상기 메모리 관리 회로는 상기 특정된 값 X가 상기 판독 타겟 값의 적어도 일부로서 상기 프로세싱 회로에 반환되게 제어하도록 구성되는, 장치.

청구항 2

제1항에 있어서,

상기 read-as-X 속성의 적어도 하나의 변형에 대해, 상기 판독 타겟 블록 모두가 상기 read-as-X 속성과 연관된 페이지에 대응한다는 결정에 응답하여, 상기 메모리 관리 회로는 캐시 또는 상기 메모리로부터 상기 판독 타겟 값을 판독하기 위한 캐시 액세스 요청 또는 메모리 액세스 요청의 발행을 억제하도록 구성되는, 장치.

청구항 3

제1항 또는 제2항에 있어서,

메모리 액세스 요청의 타겟 어드레스와 연관된 어드레스 태그가 상기 타겟 어드레스에 대응하는 메모리 시스템 위치와 연관된 배정 태그(allocation tag)에 대응하는지 여부를 결정하고, 상기 어드레스 태그와 상기 배정 태그 사이의 미스매칭을 검출하는 태그 검사 동작에 응답하여 태그 에러 핸들링 응답(tag error handling response)을 트리거하도록 상기 태그 검사 동작을 수행하기 위한 태그 검사 회로를 포함하는, 장치.

청구항 4

제3항에 있어서,

상기 메모리 관리 회로가, 상기 태그 검사 동작에서 검사될 주어진 배정 태그를 획득하기 위한 태그 판독 요청에 대해, 상기 주어진 배정 태그에 대응하는 페이지가 상기 read-as-X 속성과 연관된다고 결정하는 것에 응답하여:

상기 메모리 관리 회로는 상기 특정된 값 X가 상기 주어진 배정 태그로서 반환되게 제어하도록 구성되고;

상기 태그 검사 회로는 상기 주어진 배정 태그에 대해 상기 특정된 값 X를 사용하여 상기 태그 검사 동작을 수행하도록 구성되는, 장치.

청구항 5

제3항에 있어서,

상기 read-as-X 속성은:

데이터를 저장하기 위해 사용되는 상기 메모리 어드레스 공간의 페이지; 및

배정 태그를 저장하기 위해 사용되는 상기 메모리 어드레스 공간의 페이지 둘 모두에 특정될 수 있는, 장치.

청구항 6

제1항 내지 제5항 중 어느 한 항에 있어서,

상기 read-as-X 속성의 적어도 하나의 변형에 대해, 판독 타겟 블록이 read-as-X 속성과 연관된 페이지에 대응한다는 결정에 응답하여, 메모리 관리 회로는 특정된 값 X가 상기 판독 타겟 값의 일부로서 상기 프로세싱 회로로 반환되게 제어하고, 캐시 액세스 요청 또는 메모리 액세스 요청이 캐시 또는 상기 메모리로부터 판독 타겟 값의 다른 부분을 판독하기 위해 발행되게 허용하도록 구성되는, 장치.

청구항 7

제1항 내지 제6항 중 어느 한 항에 있어서,

상기 read-as-X 속성의 적어도 하나의 변형에 대해, 상기 특정된 값 X는 0인, 장치.

청구항 8

제1항 내지 제7항 중 어느 한 항에 있어서,

상기 read-as-X 속성의 적어도 하나의 변형에 대해, 상기 특정된 값 X는 모든 비트들이 1로 설정된 값인, 장치.

청구항 9

제1항 내지 제8항 중 어느 한 항에 있어서,

상기 메모리 관리 회로는 상기 특정된 값 X에 대한 상이한 값들에 대응하는 상기 read-as-X 속성의 복수의 변형들을 지원하도록 구성되는, 장치.

청구항 10

제1항 내지 제9항 중 어느 한 항에 있어서,

상기 메모리 관리 회로는, 주어진 페이지가 상기 read-as-X 속성의 어드레스-의존 변형과 연관될 때, 상기 특정된 값 X가 상기 주어진 페이지의 어드레스에 의존하여, 상기 메모리 관리 회로가 상기 read-as-X 속성의 동일한 인코딩과 연관된 상기 메모리 어드레스 공간의 상이한 페이지들에 대한 상기 판독 타겟 값으로서 반환될 상기 특정된 값 X로서 상이한 값들을 선택할 수 있는 상기 read-as-X 속성의 어드레스-의존 변형을 지원하도록 구성되는, 장치.

청구항 11

제1항 내지 제10항 중 어느 한 항에 있어서,

상기 메모리 어드레스 공간의 기록 타겟 블록에 기록 값을 기록하기 위해 상기 프로세싱 회로에 의해 발행된 기록 요청에 대해, 상기 기록 타겟 블록의 적어도 일부가 상기 read-as-X 속성과 연관된 페이지에 대응한다는 결정에 응답하여, 상기 메모리 관리 회로는 상기 기록 요청을 read-as-X 기록 거동에 따라 프로세싱하도록 구성되고;

상기 read-as-X 기록 거동은:

결함의 시그널링을 트리거:

상기 기록 요청을 무시; 및

상기 기록 값의 적어도 일부를 미리 결정된 시스템 위치로 포워딩 중 하나를 포함하는, 장치.

청구항 12

제11항에 있어서,

상기 미리 결정된 시스템 위치는:

진단 추적 정보를 저장하기 위한 추적 버퍼;

상기 프로세싱 회로를 포함하는 프로세서 이외의 다른 프로세서; 및

주변장치 중 하나를 포함하는, 장치.

청구항 13

제11항 또는 제12항에 있어서,

상기 메모리 관리 회로는 상이한 read-as-X 기록 거동들에 대응하는 상기 read-as-X 속성의 복수의 변형들을 지원하도록 구성되는, 장치.

청구항 14

제1항 내지 제13항 중 어느 한 항에 있어서,

상기 메모리 관리 회로는 상기 주어진 페이지에 대응하는 페이지 테이블 엔트리의 필드가 상기 read-as-X 속성을 나타내는 인코딩을 가질 때, 상기 주어진 페이지가 상기 read-as-X 속성을 갖는 것으로 결정하도록 구성되는, 장치.

청구항 15

제1항 내지 제13항 중 어느 한 항에 있어서,

상기 주어진 페이지에 대응하는 페이지 테이블 엔트리가 간접 테이블의 선택된 엔트리를 나타내는 간접 테이블 엔트리 선택 값을 특정하고 상기 간접 테이블의 선택된 엔트리가 상기 read-as-X 속성을 식별하는 인코딩을 가질 때, 상기 메모리 관리 회로는 상기 주어진 페이지가 상기 read-as-X 속성을 갖는 것으로 결정하도록 구성되는, 장치.

청구항 16

제15항에 있어서,

상기 메모리 관리 회로는 상기 read-as-X 속성의 상이한 변형들에 대응하는 상기 간접 테이블의 선택된 엔트리에 대해 적어도 2개의 상이한 인코딩들을 지원하도록 구성되는, 장치.

청구항 17

제16항에 있어서,

상기 메모리 관리 회로는 추가 간접 테이블 내의 추가 엔트리의 인코딩에 기반하여 상기 주어진 페이지에 대한 상기 특정된 값 X를 결정하도록 구성되고, 상기 추가 엔트리는 상기 간접 테이블의 상기 선택된 엔트리의 인코딩에 의해 표시되는, 장치.

청구항 18

제1항 내지 제17항 중 어느 한 항에 있어서,

상기 메모리 어드레스 공간의 주어진 페이지에 대응하는 주어진 페이지 테이블 엔트리는 어드레스 매핑 필드를 특정하고;

상기 주어진 페이지가 상기 read-as-X 속성과 연관되지 않을 때, 상기 메모리 관리 회로는 어드레스 변환을 수행하기 위해 상기 주어진 페이지 테이블 엔트리의 상기 어드레스 매핑 필드를 사용하도록 구성되고;

상기 주어진 페이지가 상기 read-as-X 속성과 연관될 때, 상기 메모리 관리 회로는 어드레스 변환 이외의 목적을 위해 상기 주어진 페이지 테이블 엔트리의 상기 어드레스 매핑 필드를 사용하도록 구성되는, 장치.

청구항 19

제18항에 있어서,

상기 주어진 페이지가 상기 read-as-X 속성과 연관될 때, 상기 메모리 관리 회로는 상기 어드레스 매핑 필드를 사용하여 상기 특정된 값 X를 표시하도록 구성되는, 장치.

청구항 20

방법으로서,

명령들에 응답하여 데이터 프로세싱을 수행하는 단계; 및

상기 메모리 어드레스 공간의 주어진 페이지 내의 어드레스에 대한 판독들이 특정된 값 X를 반환하는 것으로 취급되어야 하는 것을 나타내는 read-as-X 속성과 메모리 어드레스 공간의 주어진 페이지를 연관시킬 수 있는 페이지 테이블 정보에 기반하여 메모리에 대한 액세스를 제어하는 단계를 포함하고; 여기서

상기 메모리 어드레스 공간의 판독 타겟 블록에 대한 판독 타겟 값을 판독하기 위해 상기 프로세싱 회로에 의해 발행된 판독 요청에 대해, 상기 판독 타겟 블록의 적어도 일부가 상기 read-as-X 속성과 연관된 페이지에 대응한다는 결정에 응답하여, 상기 특정된 값 X는 상기 판독 타겟 값의 적어도 일부로서 상기 프로세싱 회로에 반환되는, 방법.

청구항 21

타겟 코드의 실행을 위한 명령 실행 환경을 제공하기 위해 호스트 데이터 프로세싱 장치를 제어하기 위한 컴퓨터 프로그램으로서, 상기 컴퓨터 프로그램은:

상기 타겟 코드의 명령들에 응답하여 데이터 프로세싱을 수행하기 위해 상기 호스트 데이터 프로세싱 장치를 제어하도록 상기 타겟 코드의 명령들을 디코딩하기 위한 명령 디코딩 프로그램 로직; 및

시뮬레이션된 메모리 어드레스 공간의 주어진 페이지 내의 어드레스에 대한 판독들이 특정된 값 X를 반환하는 것으로 취급되어야 하는 것을 나타내는 read-as-X 속성과 상기 시뮬레이션된 메모리 어드레스 공간의 주어진 페이지를 연관시킬 수 있는 페이지 테이블 정보에 기반하여 상기 시뮬레이션된 메모리 어드레스 공간에 대한 액세스를 제어하기 위한 메모리 관리 프로그램 로직을 포함하고, 여기서

상기 시뮬레이션된 메모리 어드레스 공간의 판독 타겟 블록에 대한 판독 타겟 값을 판독하기 위한 판독 요청에 대해, 상기 판독 타겟 블록의 적어도 일부가 상기 read-as-X 속성과 연관된 페이지에 대응한다는 결정에 응답하여, 상기 메모리 관리 프로그램 로직은 상기 특정된 값 X가 상기 판독 타겟 값의 적어도 일부로서 반환되게 제어하도록 구성되는, 컴퓨터 프로그램.

청구항 22

제21항의 컴퓨터 프로그램을 저장하는 저장 매체.

발명의 설명

기 술 분 야

[0001] 본 출원은 데이터 프로세싱의 분야에 관한 것이다.

[0002] 메모리 관리 회로는 메모리 어드레스 공간의 각자의 페이지들에 대한 페이지 테이블 정보에 기반하여 메모리 어드레스 공간에 대한 액세스를 제어하기 위해 프로세싱 시스템에 제공될 수 있다. 예를 들어, 페이지 테이블 정보는 액세스 권한들, 어드레스 변환 매핑들 및 주어진 페이지와 연관된 다른 속성들을 특정할 수 있다.

[0003] 적어도 일부 예들은 장치를 제공하고, 장치는:

[0004] 명령들에 응답하여 데이터 프로세싱을 수행하기 위한 프로세싱 회로; 및

[0005] 메모리 어드레스 공간의 주어진 페이지 내의 어드레스에 대한 판독들이 특정된 값 X를 반환하는 것으로 취급되어야 하는 것을 나타내는 read-as-X 속성(read-as-X property)과 메모리 어드레스 공간의 주어진 페이지를 연관시킬 수 있는 페이지 테이블 정보에 기반하여 메모리에 대한 액세스를 제어하기 위한 메모리 관리 회로를 포함하고; 여기서:

[0006] 메모리 어드레스 공간의 판독 타겟 블록에 대한 판독 타겟 값을 판독하기 위해 프로세싱 회로에 의해 발행된 판독 요청에 대해, 판독 타겟 블록의 적어도 일부가 read-as-X 속성과 연관된 페이지에 대응한다는 결정에 응답하여, 메모리 관리 회로는 특정된 값 X가 판독 타겟 값의 적어도 일부로서 프로세싱 회로에 반환되게 제어하도록 구성된다.

[0007] 적어도 몇몇 예들은 방법을 제공하고, 그 방법은:

[0008] 명령들에 응답하여 데이터 프로세싱을 수행하는 단계; 및

[0009] 메모리 어드레스 공간의 주어진 페이지 내의 어드레스에 대한 판독들이 특정된 값 X를 반환하는 것으로 취급되어야 하는 것을 나타내는 read-as-X 속성과 메모리 어드레스 공간의 주어진 페이지를 연관시킬 수 있는 페이지 테이블 정보에 기반하여 메모리에 대한 액세스를 제어하는 단계를 포함하고; 여기서:

[0010] 메모리 어드레스 공간의 판독 타겟 블록에 대한 판독 타겟 값을 판독하기 위해 프로세싱 회로에 의해 발행된 판독 요청에 대해, 판독 타겟 블록의 적어도 일부가 read-as-X 속성과 연관된 페이지에 대응한다는 결정에 응답하여, 특정된 값 X는 판독 타겟 값의 적어도 일부로서 프로세싱 회로에 반환된다.

[0011] 적어도 일부 예들은 타겟 코드의 실행을 위한 명령 실행 환경을 제공하기 위해 호스트 데이터 프로세싱 장치를 제어하기 위한 컴퓨터 프로그램을 제공하고, 컴퓨터 프로그램은,

[0012] 타겟 코드의 명령들에 응답하여 데이터 프로세싱을 수행하기 위해 호스트 데이터 프로세싱 장치를 제어하도록 타겟 코드의 명령들을 디코딩하기 위한 명령 디코딩 프로그램 로직; 및

[0013] 시뮬레이션된 메모리 어드레스 공간의 주어진 페이지 내의 어드레스에 대한 판독들이 특정된 값 X를 반환하는 것으로 취급되어야 하는 것을 나타내는 read-as-X 속성과 시뮬레이션된 메모리 어드레스 공간의 주어진 페이지를 연관시킬 수 있는 페이지 테이블 정보에 기반하여 시뮬레이션된 메모리 어드레스 공간에 대한 액세스를 제어하기 위한 메모리 관리 프로그램 로직을 포함하고, 여기서:

[0014] 시뮬레이션된 메모리 어드레스 공간의 판독 타겟 블록에 대한 판독 타겟 값을 판독하기 위한 판독 요청에 대해, 판독 타겟 블록의 적어도 일부가 read-as-X 속성과 연관된 페이지에 대응한다는 결정에 응답하여, 메모리 관리 프로그램 로직은 특정된 값 X가 판독 타겟 값의 적어도 일부로서 반환되게 제어하도록 구성된다.

[0015] 컴퓨터 프로그램은 컴퓨터-판독가능 저장 매체 상에 저장될 수 있다. 저장 매체는 일시적 또는 비일시적일 수 있다.

도면의 간단한 설명

[0016] 본 기법의 추가적인 양태들, 특징들 및 장점들은 첨부의 도면들과 연계하여 판독될 예의 다음의 설명으로부터 명백해질 것이다.

도 1은 데이터 프로세싱 장치의 예를 예시한다.

도 2는 메모리에 대한 판독 요청들 및 기록 요청들의 핸들링을 예시한다.

도 3은 read-as-X 속성과 연관된 것으로서 페이지 테이블 정보에 의해 특정된 메모리 어드레스 공간의 영역에 대한 판독 요청들 및 기록 요청들의 핸들링을 예시한다.

도 4는 판독 요청들의 핸들링을 예시하는 흐름도이다.

도 5는 기록 요청들의 핸들링을 예시하는 흐름도이다.

도 6은 페이지 테이블 정보에서 직접 인코딩될 수 있는 것보다 더 많은 수의 권한 유형들을 지원하기 위해 권한 간접 테이블의 사용을 예시한다.

도 7은 read-as-X 속성을 갖는 메모리 어드레스 영역에 대한 판독을 위해 반환될 특정 값 X를 특정하는데 사용되는 추가 간접 테이블의 사용을 예시한다.

도 8은 read-as-X 속성의 어드레스 의존 변형을 예시한다.

도 9는 태그 검사 동작을 예시한다.

도 10은 시뮬레이터 예를 예시한다.

장치는 명령들에 응답하여 데이터 프로세싱을 수행하기 위한 프로세싱 회로, 및 메모리 어드레스 공간의 주어진 페이지 내의 어드레스에 대한 판독들이 특정된 값 X를 반환하는 것으로 취급되어야 하는 것을 나타내는 read-as-X 속성과 메모리 어드레스 공간의 주어진 페이지를 연관시킬 수 있는 인코딩을 갖는 페이지 테이블 정보에 기반하여 메모리에 대한 액세스를 제어하기 위한 메모리 관리 회로를 가진다. 메모리 어드레스 공간의 판독 타

겟 블록에 대한 판독 타겟 값을 판독하기 위해 프로세싱 회로에 의해 발행된 판독 요청에 대해, 판독 타겟 블록의 적어도 일부가 read-as-X 속성과 연관된 페이지에 대응한다는 결정에 응답하여, 메모리 관리 회로는 특정된 값 X가 판독 타겟 값의 적어도 일부로서 프로세싱 회로에 반환되게 제어한다. 이는 메모리 어드레스 공간의 비교적 큰 영역들이 메모리 어드레스 공간의 이들 영역들에 대한 물리적 메모리를 커밋(commit)할 필요없이 특정된 값 X를 저장하는 것으로 취급될 수 있게 한다.

특정된 값 X를 갖는 것으로 알려진 메모리 어드레스 공간의 영역들에 대해 사용되는 물리적 메모리의 양을 감소시키기 위한 대안적인 접근법은 특정된 값 X를 저장하기 위한 물리적 메모리의 단일 페이지를 배정하고, 메모리 어드레스 공간의 대응하는 영역들 각각의 가상 어드레스들을 동일한 물리적 페이지에 매핑하기 위해 페이지 테이블 정보를 사용하는 것일 수 있다. 이 접근법에서, 특정된 값 X로 설정된 가상 메모리의 더 큰 영역을 표현하기 위해 더 작은 양의 물리적 메모리를 사용하는 능력을 지원하기 위해 아키텍처 레벨에서 특정 액세스 권한 또는 속성 유형들이 지원될 필요가 없을 것인데, 이는 소프트웨어가 이러한 거동을 제공하기 위해 페이지 테이블 정보의 어드레스 변환 매핑 필드를 사용할 수 있기 때문이다. 그러나, 이러한 접근법은 다수의 단점들을 갖는다. 첫째, 물리적 메모리가 다수의 상이한 가상 페이지들에 매핑될 수 있더라도, 물리적 메모리의 적어도 하나의 페이지-크기 블록은 여전히 특정된 값 X를 갖는 데이터를 물리적으로 저장하기 위해 배정되어야 한다. 더 큰 페이지 크기들(예를 들어, 4kB보다 큰 페이지 크기)을 갖거나 계층의 최종 레벨에서 매핑되는 하나의 페이지에 대응하는 영역의 크기보다 더 큰 메모리 영역에 대한 동일한 속성들의 세트를 정의하는 블록 디스크립터(descriptor)를 정의하기 위해 계층 구조의 더 높은 레벨의 페이지 테이블에 대한 능력을 지원하는 구현들에서, 이 접근법은 여전히 특정된 값 X를 표현하기 위해 커밋될 비교적 큰 양의 물리적 메모리를 요구할 수 있다(예를 들어, 블록 디스크립터는 수 MB 또는 GB의 메모리를 커버할 수 있음). 특정된 값 X를 저장하기 위한 물리적 스토리지를 예비하는 것(이는 종종 소프트웨어에는 흥미 없는 0과 같은 디폴트 값일 수 있음)은 특정된 값 X를 갖는 것으로 알려지지 않은 더 유용한 데이터를 저장하기 위해 이용가능한 하드웨어에 구현된 이용가능한 물리적 메모리의 분율을 감소시킨다. 의미 있는 데이터에 이용가능한 물리적 스토리지가 적으면, 다른 데이터는 외부 스토리지로 페이지 아웃(page out)되어야 할 수 있고, 이는 해당 데이터를 필요로 하는 액세스들의 성능에 영향을 준다. 또한, 특정된 값 X를 갖는 데이터에 대한 판독들에 대해, 이들 판독들은 캐시 또는 기본 물리적 메모리 스토리지로부터 데이터에 액세스할 필요성으로 인해 여전히 비교적 긴 레이턴시를 겪을 수 있다.

대조적으로, 아래에서 논의되는 접근법에서, 메모리 어드레스 공간의 페이지는 페이지 테이블 정보에 의해 read-as-X 속성과 연관될 수 있고, 이는 해당 페이지의 어드레스에 대한 판독들이 특정된 값 X를 반환하는 것으로 취급되어야 한다는 것을 나타낸다. 이는, 판독 요청이 메모리 어드레스 공간의 판독 타겟 블록에 대한 판독 타겟 값을 판독하기 위해 프로세싱 회로에 의해 발행되고, 메모리 관리 회로가 (판독 요청의 타겟 어드레스에 기반하여) 판독 타겟 블록의 적어도 일부가 read-as-X 속성과 연관된 페이지에 대응한다고 결정할 때, 메모리 관리 회로가, 특정된 값 X가 판독 타겟 값의 적어도 일부로서 프로세싱 회로에 반환되도록 제어할 수 있다는 것을 의미한다. 메모리 관리 회로가 페이지 테이블 정보로부터 단순히 특정된 값을 식별할 수 있기 때문에, 이는 특정된 값 X가 할당된 메모리 어드레스 공간의 영역들에 대해 임의의 물리적 메모리를 커밋할 필요성을 회피한다. 따라서, 이는 시스템에서 구현되는 나머지 물리적 메모리 스토리지의 더 나은 활용을 가능하게 하고, 그 중 더 많은 분율은 특정된 값 X를 갖지 않는 다른 데이터에 사용될 수 있고, 그렇지 않으면 외부 스토리지로 전송되어야 하는 다른 데이터에 대한 성능을 개선한다. 또한, 이 접근법은, 메모리 관리 회로가 통상적으로 물리적 메모리에 저장된 데이터를 판독하기 위한 캐시 또는 메모리에 대한 액세스보다 더 일찍 페이지 테이블 정보를 조회하여, 특정된 값 X 자체의 판독들에 대한 성능을 개선할 수 있고, 따라서 프로세싱의 더 이른 스테이지에서 값 X 를 반환할 수 있다.

메모리 관리 회로에 의해 지원되는 read-as-X 속성의 적어도 하나의 변형에 대해, 판독 타겟 블록 모두가 read-as-X 속성과 연관된 페이지에 대응한다는 결정에 응답하여, 메모리 관리 회로는 캐시 또는 메모리로부터 판독 타겟 값을 판독하기 위한 캐시 액세스 요청 또는 메모리 액세스 요청의 발행을 억제하도록 구성된다. 따라서, read-as-X 속성의 사용은 값 X가 캐시 또는 메모리로부터 전혀 판독될 필요성을 회피할 수 있다(대신, 값 X는 페이지 테이블 정보에 표현된 검출된 read-as-X 속성에 기반하여 메모리 관리 회로 내에서 생성될 수 있음). 캐시 액세스의 레이턴시를 제거함으로써, 이것은 판독 요청 프로세싱을 가속화하고 또한 다른 데이터에 대한 캐시 용량을 확보하게 한다.

이러한 변형으로, 판독 요청에 의해 특정된 판독 타겟 블록이 read-as-X 속성을 갖는 하나의 페이지와 read-as-X 속성을 갖지 않는 다른 페이지 사이의 페이지 경계를 넘는 것이 가능하다. 이 경우, 프로세싱 회로로 반환되는 판독 타겟 값의 일부만이 특정된 값 X를 특정할 수 있고, 다른 부분은 정상 판독 액세스 요청에 대한 것으로

서 캐시 또는 메모리로부터 판독될 수 있다. 실제로, 페이지 경계를 넘는 액세스들은 캐시 라인 경계를 또한 넘을 수 있기 때문에 일반적으로 2개의 별개의 캐시 조회들을 요구할 것이고, 따라서 액세스는 2개의 별개의 액세스들로서 효과적으로 취급될 수 있는데, 하나는 제1 페이지 내의 부분에 대한 것이고 다른 하나는 제2 페이지 내의 부분에 대한 것이다. 따라서, 메모리 어드레스 공간의 비-read-as-X 페이지에 대응하는 이들 액세스들 중 하나는 캐시 조회를 트리거하여 메모리를 정상적으로 액세스할 수 있는 반면, read-as-X 페이지에 대응하는 액세스들 중 다른 하나는 캐시 조회를 억제하고 특정된 값 X를 판독 타겟 블록의 해당 부분에 대해 반환된 값으로서 반환할 수 있다. 실제로, 판독 액세스가 read-as-X에 부분적으로만 대응하는 시나리오들은 액세스가 페이지 경계를 넘을 가능성을 감소시키기 위해 메모리 액세스들의 어드레스 정렬을 사용하는 것이 일반적이라는 것을 감안할 때 비교적 드물게 있을 수 있고, 또한 페이지 크기는 주어진 액세스에 대한 페이지 경계를 넘을 가능성이 낮도록 판독 타겟 블록의 크기에 비해 상대적으로 클 수 있다.

read-as-X 속성은 메모리 어드레스 공간의 비교적 큰 블록들이 특정 값을 저장하는 것으로 설정될 필요가 있는 광범위한 시나리오들에 유용할 수 있다. 예를 들어, 가상 어드레스 공간의 영역들을 먼저 초기화할 때, 이들 영역들에 대한 임의의 기록이 수행되기 전에, 임의의 개입하는 판독들이 이들 가상 어드레스들에 대해 이전에 어떠한 데이터가 저장되었든 무관하게 알려진 값(예를 들어, 0)을 반환하는 것을 보장하기 위해 이들 영역들의 이전 콘텐츠들을 클리어(clear)하는 것이 바람직할 수 있다. 따라서, 가상 어드레스 공간의 새롭게 초기화된 영역들은 이들 영역들에 대한 제1 기록 동작이 발생할 때까지 read-as-X 속성이 할당될 수 있다. 예를 들어, 기록 시 복사 접근법(copy on write approach)은 초기 배정 시에, 새로 초기화된 페이지에 대한 속성들이 read-as-X를 나타내도록 설정될 수 있는 소프트웨어에 의해 사용될 수 있어서, 해당 페이지에 대한 제1 기록은 예외 핸들러(exception handler)가 물리적 메모리의 새로운 블록을 배정하게 할 수 있는 결함을 트리거하고, 페이지 테이블 정보에서 어드레스 변환 매핑을 사용하여 해당 물리적 메모리를 가상 페이지에 매핑하고, 이어서 후속하여 가상 페이지를 정상 메모리로서 취급하기 위해 read-as-X 속성을 클리어할 것이다. read-as-X 속성을 지원하는 아키텍처를 제공함으로써, 소프트웨어에 의해 원해지는 이 기록 시 복사 접근법은 상술된 바와 같이 특정된 값 X를 저장하기 위해 커밋된 물리적 메모리의 단일 페이지에 매핑하기 위해 어드레스 변환 매핑을 사용함으로써 기록 시 복사를 제어하는 대안과 비교하여, 물리적 스토리지의 더 높은 활용이 지원될 수 있고 성능을 개선할 수 있다.

물론, 이 기록 시 복사 접근법은 단지 일 예일 뿐이며, 소프트웨어가, 물리적 메모리 스토리지를 커밋하지 않고, 가상 어드레스 공간의 영역이 특정된 값 X를 갖는 것으로 취급된다는 것을 나타낼 수 있는 것이 유용하다고 발견될 수 있는 많은 다른 경우들이 있을 수 있다.

read-as-X 속성이 유용할 수 있는 다른 예는 메모리 액세스 요청의 타겟 어드레스와 연관된 어드레스 태그가 타겟 어드레스에 대응하는 메모리 시스템 위치와 연관된 배정 태그에 대응하는지 여부를 결정하기 위해 태그 검사 동작을 수행하기 위한 태그 검사 회로를 지원하는 시스템에서이다.

. 태그 검사 회로는 어드레스 태그와 배정 태그 사이의 미스매칭을 검출하는 태그 검사 동작에 응답하여 태그 에러 핸들링 응답을 트리거한다. 아래에서 더 상세히 논의되는 바와 같이, 이러한 태그 검사 동작은 소프트웨어가 액세스할 수 없어야 하는 메모리 영역에 액세스하게 하기를 원하는 공격자에 의해 이용될 수 있는 보안 취약성들을 야기할 수 있는 소정 메모리 사용 에러들을 검출하기 위한 아키텍처 지원을 제공하는데 유용할 수 있다. 메모리 어드레스 공간의 영역이 처음에 할당될 때, 대응하는 배정 태그가 할당될 수 있고, 메모리 어드레스 공간의 해당 영역을 참조하는데 사용되도록 의도된 포인터는 대응하는 어드레스 태그가 할당될 수 있다. 포인터가 메모리에 액세스하기 위한 어드레스를 생성하기 위해 후속적으로 사용될 때, 태그 검사 회로는 (적어도 태그 검사 모드에 있을 때) 메모리 액세스 요청의 타겟 어드레스의 어드레스 태그를 어드레싱된 메모리 시스템 위치와 연관된 배정 태그에 비교하고, 이어서 임의의 검출 미스매칭에 응답하여 태그 에러 핸들링 응답을 트리거할 수 있다. 태그 에러 프로세싱 응답은 결함을 트리거하거나, 에러 로그 또는 다른 저장 위치에 에러를 로깅하는 것 등 같은 동작들의 범위일 수 있다.

태그 검사 동작들을 지원하는 시스템들에서, 메모리 블록에 대한 배정 태그는 판독 액세스에 의해 그 자체가 메모리 어드레스 공간의 특정 블록에 액세스될 수 있고, 따라서 배정 태그 자체들은 물리적 스토리지를 소비할 수 있다. 소정 태그 값은 너무 일반적이어서, 1 페이지보다 큰 크기의 메모리 어드레스 공간의 비교적 큰 영역들이 동일한 태그 값을 갖는 것으로 소프트웨어에 의해 설정되는 것이 일반적일 수 있다. 예를 들어, 소프트웨어는 더 큰 보안 요건이 있는 메모리 어드레스 공간의 영역들에 대한 배정 태그들의 특정 값들을 설정하도록 선택할 수 있고, 이어서 메모리의 임의의 다른 영역에 대한 배정 태그들을 소정 디폴트 태그 값(예를 들어, 0)을 갖는 것으로 남겨둘 수 있다. 더 큰 보안 요건을 갖는 영역들의 배정 태그들에 대한 비-디폴트 태그 값들의 사용

을 우선순위화함으로써, 이는 이러한 영역들 중 하나에 액세스하도록 의도되지 않은 메모리 액세스가 우연에 의해, 어드레싱된 메모리 시스템 위치의 배정 태그와 매칭되는 어드레스 태그를 가질 기회를 감소시킨다. 이는 소프트웨어가 모두 동일한 태그 값(예를 들어, 0)을 갖는 비교적 큰 메모리 영역들에 대한 배정 태그들을 구성하기를 원할 수 있다는 것이 비교적 일반적임을 의미한다. 따라서, read-as-X 속성에 대한 지원이 없는 경우, 비교적 상당한 양의 물리적 메모리 스토리지는 특정 보안 요건이 없는 메모리의 영역들에 대한 디폴트 배정 태그 값을 저장하기 위해서만 사용될 수 있다. read-as-X 속성을, 특정된 값 X를 갖는 배정 태그 값들을 보유하도록 의도된 메모리 어드레스 공간의 페이지들과 연관시킴으로써, 이것은 메모리 태깅을 지원하는 시스템에서 이용가능한 물리적 메모리 스토리지의 활용 효율성을 크게 개선시킬 수 있다.

따라서, 태그 검사 동작에서 검사될 주어진 배정 태크를 획득하기 위한 태그 판독 요청에 대해, 주어진 배정 태크에 대응하는 페이지가 read-as-X 속성과 연관된다고 메모리 관리 회로가 결정하는 것에 응답하여: 메모리 관리 회로는 특정된 값 X가 주어진 배정 태크로서 반환되게 제어하도록 구성되고; 태그 검사 회로는 주어진 배정 태크에 대해 특정된 값 X를 사용하여 태그 검사 동작을 수행하도록 구성된다. 따라서, 태그 검사 동작은, 이용가능한 물리적 메모리의 성능 및 활용을 개선할 수 있는, 특정된 값 X를 획득하기 위해 캐시 또는 물리적 메모리 스토리지에 액세스할 필요없이 메모리 관리 회로에 의해 제공되는 특정된 값 X를 사용하여 수행될 수 있다.

메모리 태깅을 지원하는 시스템에서, read-as-X 속성은: 데이터를 저장하기 위해 사용되는 메모리 어드레스 공간의 페이지; 및 배정 태그들을 저장하기 위해 사용되는 메모리 어드레스 공간의 페이지 둘 모두에 대해 특정될 수 있다. 따라서, read-as-X 메커니즘은 데이터-특정 또는 태그-특정 속성일 필요는 없지만, 메모리 어드레스 공간의 페이지와 연관되는 임의의 정보에 대해 이용가능한 일반적인 아키텍처 메커니즘일 수 있다.

일부 예들에서, read-as-X 속성의 적어도 하나의 변형에 대해, 판독 타겟 블록이 read-as-X 속성과 연관된 페이지에 대응한다는 결정에 응답하여, 메모리 관리 회로는 특정된 값 X가 판독 타겟 값의 일부로서 프로세싱 회로로 반환되고, 캐시 액세스 요청 또는 메모리 액세스 요청이 캐시 또는 메모리로부터 판독 타겟 값의 다른 부분을 판독하기 위해 발행되도록 허용하게 제어한다. 이 경우, 특정된 값 X는 판독 타겟 값의 일부에 대해서만 반환되고, 다른 일부는 정상 메모리 액세스에 대해 캐시 또는 메모리로부터 반환된다. 이는 실제 물리적 메모리에 저장된 데이터가 물리적 메모리에 저장될 필요가 없는 read-as-X 속성에 기반하여 암시된 추가 비트들과 함께 더 넓은 형태로 확장되게 하는 데 유용할 수 있다. 예를 들어, 메모리 태깅을 지원하는 시스템의 경우, 판독 값의 일부에 대해 반환된 특정된 값 X는 물리적 메모리에 저장된 대응하는 데이터와 연관하여 암시적으로 정의된 배정 태그를 나타낼 수 있어서, 디폴트 배정 태그 값이 할당될 메모리 영역들에 대한 배정 태그들은 물리적 메모리가 그러한 태그들을 저장하기 위해 커밋될 것을 요구하지 않고 read-as-X 속성에 기반하여 암시적으로 결정될 수 있다.

임의의 데이터 값이나 비트들의 패턴은 read-as-X 속성을 가진 페이지들에 대해 특정된 값 X로 할당될 수 있다.

예를 들어, read-as-X 속성의 하나 이상의 변형에 대해, 특정된 값 X는 0이다. read-as-X 속성을 갖는 페이지를 0을 포함하는 것으로 취급하는 것은 특히 유용할 수 있는데, 왜냐하면 메모리의 큰 영역들이 0이 아닌 다른 값에 대한 경우보다 0을 저장하는 것으로 취급되어야 하는 더 많은 수의 소프트웨어 사용 경우들이 있을 수 있기 때문이다. 예를 들어, 위에서 설명된 기록 시 복사 및 메모리 태깅 사용 경우들은 0들을 0들이 아닌 다른 값들보다 read-as-X 페이지들에 할당할 가능성이 더 클 수 있다.

그러나, 또한 특정된 값 X가 0이 아닌 값을 갖는 X로 read-as-X 속성의 적어도 하나의 변형을 구현할 수 있다. 예를 들어, read-as-X 속성의 적어도 하나의 변형에 대해, 특정된 값 X는 모든 비트들이 1로 설정된 값이다. 예를 들어, 메모리 태깅 예에서, 모든 비트들이 1로 설정된 배정 태그 값은 특별한 목적들을 위해 예비될 수 있다(예를 들어, 이는 태그가 임의의 어드레스 태그 값과 매칭하는 것으로 취급될 것임을 나타내는 "모두 매칭" 태그 값을 나타낼 수 있음). 따라서, X=1111111...는 또한, 페이지에 대한 판독이 1과 동일한 모든 비트들을 반환하는 것으로 취급되는 것을 나타내는 페이지 테이블 정보의 인코딩에 대한 아키텍처 지원을 정당화하기 위해 메모리 어드레스 공간의 큰 영역들에 할당하기에 좋은 후보일 수 있다.

특정된 값 X가 0과 1의 비트 값들의 혼합을 갖는 값인 것이 또한 가능할 것이다. 특정된 값 X는 전체 페이지에 걸친 단일 값이라고 간주될 수 있거나, 또는 대안적으로 X는 read-as-X 속성을 갖는 페이지에 대응하는 메모리 범위를 채우기 위해 다수 번 반복되는 더 작은 수의 비트들을 갖는 것으로 간주될 수 있다.

특정된 값 X를 정의하기 위해 아키텍처에서 지원되는 상이한 옵션들의 수와 페이지 테이블 정보 또는 다른 구성 정보에서 이러한 상이한 옵션들을 나타내는 데 필요한 인코딩 공간의 양 사이에 트레이드-오프(trade-off)가 존

재한다는 것이 이해될 것이다. 일부 구현들은 특정된 값 X(예를 들어, X=0)의 단일 변형에 대응하는 read-as-X 인코딩들을 지원할 수 있다. 다른 예들에서, 메모리 관리 회로는 물리적 메모리를 소비하지 않고 가상 메모리의 영역들에 할당될 다양한 상이한 값들에 대한 read-as-X 거동의 사용을 허용하기 위해, 특정된 값 X에 대한 상이한 값들에 대응하는 read-as-X 속성의 2개 이상의 변형들을 지원하도록 구성된다.

일 예에서, 메모리 관리 회로는 read-as-X 속성의 어드레스-의존 변형을 지원할 수 있고, 이에 대해 주어진 페이지가 read-as-X 속성의 어드레스-의존 변형과 연관될 때, 특정된 값 X는 주어진 페이지의 어드레스에 의존하여, 메모리 관리 회로는 read-as-X 속성의 동일한 인코딩과 연관된 메모리 어드레스 공간의 상이한 페이지들에 대한 판독 타겟 값으로서 반환될 특정된 값 X로서 상이한 값들을 선택할 수 있다. 예를 들어, 특정된 값 X는 어드레스의 적어도 하나의 비트에 따라 선택될 수 있다. 예를 들어, 일부 아키텍처들은 사용자 어드레스 공간 또는 커널 어드레스 공간으로부터 액세스가 이루어지고 있는지 여부를 나타내기 위해 어드레스의 하나 이상의 최상위 비트들(메모리 위치들을 선택하는데 사용되지 않는 임의의 상위 어드레스 비트들을 제외)을 사용할 수 있고, 사용자 또는 커널 액세스가 요청되고 있는지 여부에 따라 read-as-X 속성에 대해 특정된 값 X로서 상이한 값들을 반환하는 것이 유용할 수 있다. 이 접근법은, 단일 유형의 read-as-X 속성이 둘 모두의 목적들을 수행할 수 있기 때문에, 이 거동을 지원하기 위해 페이지 테이블 정보의 2개의 상이한 인코딩들을 소비할 필요성을 회피하고, 특정된 값 X로서 어떤 값을 반환할지를 선택하기 위한 추가 정보는 페이지 테이블 정보가 아닌 어드레스 자체에서 온다.

read-as-X 속성을 갖는 것으로 지정된 페이지들에 대한 기록들의 경우, 다양한 상이한 거동들이 구현될 수 있다. 하나의 옵션은 read-as-X 페이지에 대한 기록이 결함 시그널링을 트리거하는 것이고, 이는 위에서 논의된 기록 시 복사 접근법을 지원하는데 유용할 수 있고, 여기서 결함은 해당 페이지에 대한 read-as-X 속성을 제거하기 전에, 예외 핸들러가 기록될 페이지에 대한 일부 물리적 메모리를 실행 및 배정하게 할 수 있다.

대안적으로, read-as-X에 대한 기록 요청은, 결함을 발생시킴으로써 프로세싱 회로의 프로세싱을 중단하지 않고, 간단히 무시될 수 있다. 이 경우, 기록 시 복사 거동을 구현하기 위해, 소프트웨어는 또한 기록 액세스들이 허용되지 않는다는 것을 나타내기 위해 read-as-X 페이지에 대한 액세스 권한을 설정할 수 있어서, 결함은, 일단 특정된 값 이외의 일부 데이터를 저장하기 위해 실제로 요구되면, 물리적 메모리의 동적 배정을 허용하기 위해 페이지에 대한 제1 기록에서 생성될 수 있다.

다른 옵션은, read-as-X로 지정된 페이지에 대한 기록이, 기록 값(어드레스 공간의 기록 타겟 블록에 기록될 것으로 기록 요청에 의해 특정된 값)의 적어도 일부가 미리 결정된 시스템 위치로 포워딩되는 것을 요청하는 것으로 해석될 수 있다는 것이다. 예를 들어, 미리 결정된 시스템 위치는 랜덤 액세스 메모리 스토리지 이외의 위치일 수 있다. 예를 들어, 미리 결정된 시스템 위치는: 진단 추적 정보를 저장하기 위한 추적 버퍼; 기록 요청을 발행한 프로세싱 회로를 포함하는 프로세서 이외의 다른 프로세서; 및 주변장치(입력/출력 디바이스) 중 하나일 수 있다. 어드레스 공간의 가상으로 어드레싱된 페이지에 대한 read-as-X 속성의 사용은, 가상 어드레스를 물리적 어드레스로 변환하고, 해당 물리적 어드레스를 특정하는 규칙적인 기록 요청을 메모리 시스템에 발행하고, 이어서 어드레스가 미리 결정된 시스템 위치에 대응한다고 결정하기 위해 메모리 시스템 내의 물리적 어드레스 매핑 컴포넌트를 사용하여 기록 데이터를 해당 위치에 포워딩하는 것과 비교하여, 기록 데이터와 함께 포워딩될 타겟 위치를 식별하는 더 빠른 방식일 수 있다. 따라서, 미리 결정된 시스템 위치로부터 데이터를 판독할 어떠한 필요도 없지만, 미리 결정된 시스템 위치에 데이터를 기록하는 가속화된 수단이 요구되는 시스템 위치들에 대해, read-as-X 속성은 유용할 수 있다. 사용 경우들의 예는 다음을 포함할 수 있다:

- 특정 소프트웨어 프로그램이 프로세서에서 실행될 때 발생하는 거동 및 성능 문제들을 조사하기 위해 후속 분석을 위해 진단 정보를 전송할 수 있는 추적 버퍼들. 계측된 소프트웨어 코드는 추적 버퍼에 기록될 추적 정보를 특정하는 코드의 소정 포인트에서의 기록 메모리 액세스를 포함할 수 있고, 이 메모리 액세스의 어드레스가 read-as-X 페이지에 매핑될 때, 이는 추적 정보가 정규 액세스 메커니즘들을 사용하는 것보다 더 빨리 추적 버퍼로 포워딩되게 할 수 있다.

- 네트워크 또는 다른 통신 인터페이스의 입력/출력 버퍼에 대한 가속화된 액세스.

- 표준 메모리 시스템 버스 인터페이스보다 더 많은 정보를 요구할 수 있는 공유된 컴포넌트들에 대한 액세스들은, 예를 들어, 현재 실행 컨텍스트에 관한 정보를 전달하는 것이 유리할 수 있는 DMA, 또는 생산자-소비자 모드에서 동작하는 임베디드 밀접-결합된 멀티-CPU 시스템들에 대한 액세스들을 제공할 수 있고, 여기서 페이지

테이블 속성은 특정 하드웨어 FIFO(선입, 선출 버퍼)가 예를 들어 사용됨을 나타내기 위해 사용될 수 있다.

프로세싱 시스템 설계자들은 메모리 어드레스 공간의 read-as-X 메모리 페이지들에 대한 이러한 기록 거동의 다른 사용들을 발견할 수 있다는 것이 이해될 것이다.

따라서, 메모리 어드레스 공간의 기록 타겟 블록에 기록 값을 기록하기 위해 프로세싱 회로에 의해 발행된 기록 요청에 대해, 기록 타겟 블록의 적어도 일부가 read-as-X 속성과 연관된 페이지에 대응한다는 결정에 응답하여, 메모리 관리 회로는 기록 요청을 read-as-X 기록 거동에 따라 프로세싱하도록 구성된다. read-as-X 기록 거동은: 결함의 시그널링을 트리거하는 것; 기록 요청을 무시하는 것; 및 기록 값의 적어도 일부를 미리 결정된 시스템 위치(예를 들어, 추적 버퍼, 다른 프로세서 또는 주변장치)로 포워딩하는 것 중 하나를 포함한다.

명령 세트 아키텍처의 일부 구현들에 대해, 단일 유형의 read-as-X 기록 거동(예를 들어, read-as-X 메모리에 대한 기록들의 폴팅(faulting))이 구현될 수 있고, 소프트웨어가 상이한 read-as-X 기록 거동을 선택하는 것이 가능하지 않을 수 있다.

다른 예들에서, 메모리 관리 회로는 상이한 read-as-X 기록 거동들에 대응하는 read-as-X 속성의 2개 이상의 변형들을 지원할 수 있다. 이 경우, 특정 기록 요청을 위해 사용될 특정 read-as-X 기록 거동은 소프트웨어-구성가능 제어 정보에 의존할 수 있고, 이는 페이지 테이블 정보 그 자체 또는 제어 레지스터 또는 다른 구성가능한 저장 요소에서 개별적으로 특정될 수 있다. 예를 들어, read-as-X 기록 거동은 페이지 테이블 정보 내에서 또는 특정 페이지에 대한 페이지 테이블 정보로부터 참조되는 다른 정보 내에서 페이지 단위로 특정될 수 있거나, 대안적으로, 예를 들어, read-as-X 기록 거동이 결정되게 하는 정보(예를 들어, 값을 특정하는 read-as-X 기록 모드)를 나타내는 제어 레지스터를 사용하여, 전체의 장치에 대해 설정된 글로벌 설정일 수 있다.

어드레스 공간의 주어진 페이지의 read-as-X 속성은 다른 방식들로 인코딩될 수 있다. 일 예에서, 메모리 관리 회로는 주어진 페이지에 대응하는 페이지 테이블 엔트리의 필드가 read-as-X 속성을 나타내는 인코딩을 가질 때, 주어진 페이지가 read-as-X 속성을 갖는 것으로 결정한다. 따라서, 이 경우, 페이지 테이블 엔트리 인코딩은 주어진 페이지가 read-as-X 속성을 갖는지 여부를 직접 특정할 수 있다.

다른 접근법은, 주어진 페이지에 대응하는 페이지 테이블 엔트리가 간접 테이블의 선택된 엔트리를 나타내는 간접 테이블 엔트리 선택 값을 특정하고 간접 테이블의 선택된 엔트리가 read-as-X 속성을 식별하는 인코딩을 가질 때, 메모리 관리 회로가 주어진 페이지가 read-as-X 속성을 갖는 것으로 결정하는 것일 수 있다. 이것은, 프로그래머에 의한 선택에 이용가능한 모든 유형들의 메모리 속성들/권한들을 인코딩할 수 있는 페이지 테이블 엔트리 형식 내에 여분의 인코딩 공간이 부족할 경우 유용할 수 있다. 간접 테이블이 페이지 당 엔트리를 포함할 필요가 없기 때문에, 페이지 테이블 엔트리 자체에서 실현될 것보다 주어진 페이지에 대해 선택될 속성들에 대한 더 넓은 옵션들의 세트들을 표현하기 위한 더 많은 공간이 있을 수 있어서(간접 테이블 엔트리들의 총 수는 단일 간접 테이블 엔트리에 대해 선택가능한 속성 유형들의 총 수보다 작을 수 있음), 페이지 테이블 엔트리에서 간접 테이블 엔트리 선택 값은 하나의 간접 테이블 엔트리에서 비트들의 수보다 더 적은 비트들을 갖는다. 따라서, 일부 예들에서, read-as-X 속성은 페이지 테이블 정보에서 직접적으로 인코딩되기보다는 간접 테이블의 참조된 엔트리에서 인코딩될 수 있다.

간접 테이블은 메모리에 저장된 구조일 수 있거나 또는 예를 들어, 시스템 레지스터 명령들에 의해 액세스되는 제어 레지스터를 사용하여 구현되는 하드웨어 구조일 수 있다.

간접 테이블을 지원하는 접근법에서, 메모리 관리 회로는 read-as-X 속성의 상이한 변형들에 대응하는 간접 테이블의 선택된 엔트리에 대해 적어도 2개의 상이한 인코딩들을 지원할 수 있다. 예를 들어, read-as-X 속성의 상이한 변형들은 특정된 값 X에 대한 상이한 값, 상이한 read-as-X 기록 거동들에 대응할 수 있거나, read-as-X 속성과 조합하여 표시될 다른 액세스 권한들 또는 속성들에 의해 구별될 수 있다.

일 접근법에서, 메모리 관리 회로는 추가 간접 테이블 내의 추가 엔트리의 인코딩에 기반하여 주어진 페이지에 대한 특정된 값 X를 결정할 수 있고, 추가 엔트리는 간접 테이블의 선택된 엔트리의 인코딩에 의해 표시된다. 이는 소프트웨어가 특정된 값 X에 대해 특정하고자 할 수 있는 상이한 값들의 수가 read-as-X 속성에 대해 정의될 상이한 속성 변형들 또는 거동들(특정된 값 X의 값 이외의 특징들에 의해 구별됨)의 수보다 훨씬 더 클 수 있고, 따라서 read-as-X에 대한 속성/거동 변형들을 정의하기 위한 간접 테이블(특정된 값 X 이외의) 및 특정된 값 X를 정의하기 위한 추가 간접 테이블을 정의하는 것이 read-as-X 속성에 대한 속성/거동/특정된 값의 각각의 원하는 조합을 나타내기 위한 인코딩을 갖는 단일 테이블보다 더 효율적일 수 있음을 인식한다.

read-as-X 메모리로 지정되지 않은 페이지들에 대한 정상 메모리 액세스들을 지원하기 위해, 메모리 어드레스

공간의 주어진 페이지에 대응하는 주어진 페이지 테이블 엔트리는 어드레스 매핑 필드를 특정할 수 있고, 이는 주어진 페이지가 read-as-X 속성과 연관되지 않을 때 어드레스 변환을 수행하는데 사용된다. 어드레스 매핑 필드는 종종 페이지 테이블 엔트리의 전체 비트들의 수의 상당한 분율을 차지하는 페이지 테이블 엔트리의 가장 큰 필드일 수 있다.

그러나, 페이지가 read-as-X 속성과 연관된 경우, 해당 페이지에 물리적 메모리가 배정되지 않기 때문에, 어드레스 변환이 필요하지 않고, 따라서 어드레스 매핑 필드가 사용되지 않을 수 있다.

일부 예들에서, 다른 정보를 인코딩하기 위해 추가 인코딩 공간을 사용할 필요가 없는 것을 피하기 위해, read-as-X 속성과 연관된 페이지에 대해, 메모리 관리 회로는 어드레스 변환 이외의 목적을 위해 주어진 페이지 테이블 엔트리의 어드레스 매핑 필드를 사용할 수 있다. 따라서, read-as-X 페이지들의 경우, 정상 메모리에 대해 그렇지 않으면 가능할 것보다 페이지 테이블 엔트리에서 이용가능한 인코딩 공간의 더 많은 양이 존재할 수 있다. 예를 들어, 특정된 값 X는 어드레스 매핑 필드를 사용하여 표시될 수 있다. 또한, 어드레스 매핑 필드를 사용하여 read-as-X 속성의 다른 변형들(예를 들어, 상이한 read-as-X 기록 거동들 또는 read-as-X 속성과 조합될 다른 속성들)을 표시하는 것이 가능하다. 이러한 정보를 인코딩하기 위해 어드레스 매핑 필드를 재사용함으로써, 페이지 테이블 엔트리 포맷을 확장하거나 또는 이 정보를 인코딩하기 위해 간접 테이블과 같은 다른 구조들을 사용할 필요가 적다.

다른 구현들은 어드레스 매핑 필드를 사용하여, 메모리 관리 회로에 의해 이해되는 read-as-X 속성에 대한 정보를 미리 규정된 아키텍처적으로 정의된 의미를 갖도록 인코딩하지 않는 것을 선호할 수 있다. 대신, read-as-X 속성과 연관된 페이지 테이블 엔트리의 어드레스 매핑 필드는 소프트웨어가 프로세싱 회로에 의해 지원되는 명령 세트 아키텍처에 의해 규정되지 않는 임의의 소프트웨어 정의 목적들을 위해 사용되도록 자유로울 수 있다.

이제 특정 예들이 도면들을 참조하여 설명될 것이다. 청구범위들이 이러한 정확한 예들로 제한되지 않음이 이해될 것이다.

도 1은 데이터 프로세싱 장치(2)의 예를 개략적으로 예시한다. 데이터 프로세싱 장치는 프로세싱 파이프라인(4)(예를 들어, CPU(중앙 프로세싱 유닛)의 일부를 형성할 수 있는 프로세싱 회로의 예)을 갖는다. 프로세싱 회로(4)는 명령들에 의해 표현되는 데이터 프로세싱 동작들을 수행하기 위해 명령 세트 아키텍처(ISA)에 정의된 명령들을 실행하기 위한 것이다. 프로세싱 파이프라인(4)은 다수의 파이프라인 스테이지들을 포함한다. 이러한 예에서, 파이프라인 스테이지들은 명령 캐시(8)로부터 명령들을 페치하기 위한 페치 스테이지(6); 파이프라인의 나머지 스테이지들에 의해 프로세싱될 마이크로 동작들(디코딩된 명령들)을 생성하기 위해 페치된 프로그램 명령들을 디코딩하기 위한 디코드 스테이지(10); 마이크로 동작들을 위해 요구되는 피연산자들이 레지스터 파일(14)에서 이용가능한지 여부를 검사하고, 일단 주어진 마이크로 동작을 위해 요구되는 피연산자들이 이용가능하면 실행을 위한 마이크로 동작들을 발행하기 위한 발행 스테이지(12); 결과 값들을 생성하기 위해 레지스터 파일(14)로부터 판독된 피연산자들을 프로세싱함으로써, 마이크로 동작들에 대응하는 데이터 프로세싱 동작들을 실행하기 위한 실행 스테이지(16); 및 프로세싱의 결과들을 레지스터 파일(14)에 다시 기록하기 위한 재기록 스테이지(18)를 포함한다. 이것은 가능한 파이프라인 아키텍처의 단지 하나의 예에 불과하며, 다른 시스템들이 추가적인 스테이지들 또는 상이한 구성의 스테이지들을 가질 수 있다는 것이 인식될 것이다. 예를 들면, 비순차적 프로세서에서, 레지스터 리네이밍 스테이지(register renaming stage)는 프로그램 명령들 또는 마이크로 동작들에 의해 특정되는 아키텍처 레지스터들을 레지스터 파일(14)의 물리적 레지스터들을 식별하는 물리적 레지스터 특정자들에 매핑하기 위해 포함될 수 있다. 일부 예들에서, 디코드 스테이지(10)에 의해 디코딩되는 ISA에서 정의된 프로그램 명령들과 실행 스테이지에 의해 프로세싱되는 대응하는 마이크로 동작들 사이에는 일 대 일 관계가 있을 수 있다. 프로그램 명령들과 마이크로 동작들 사이에는 일 대 다 또는 다 대 일 관계가 있는 것이 또한 가능하고, 그 결과, 예를 들면, 단일의 프로그램 명령은 2개 이상의 마이크로 동작들로 분할될 수 있거나, 또는 2개 이상의 프로그램 명령들은 융합되어 단일의 마이크로 동작으로서 프로세싱될 수 있다.

실행 스테이지(16)는 상이한 클래스들의 프로세싱 동작을 실행하기 위한 다수의 프로세싱 유닛들을 포함한다. 예를 들어, 실행 유닛들은 레지스터들(14)로부터 판독된 스칼라 피연산자들에 대해 산술 또는 논리 연산들을 수행하기 위한 스칼라 산술/로직 유닛(ALU)(20); 부동 소수점 값들에 대해 연산들을 수행하기 위한 부동 소수점 유닛(22); 분기 동작들의 결과를 평가하고 그에 따라 현재의 실행 지점을 나타내는 프로그램 카운터를 조정하기 위한 분기 유닛(24); 및 메모리 시스템(8, 30, 32, 34)에서 데이터에 액세스하기 위한 로드/저장 동작들을 수행하기 위한 로드/저장 유닛(26)을 포함할 수 있다.

메모리 관리 유닛(MMU)(28)은 메모리 액세스 권한 검사들을 제어하고 데이터 액세스 명령들의 피연산자들에 기

반하여 로드/저장 유닛(26)에 의해 특정된 가상 어드레스들과 메모리 시스템 내의 데이터의 저장 위치들을 식별하는 물리적 어드레스들 사이에서 어드레스 변환들을 수행하기 위해 제공된다. MMU는 메모리 시스템에 저장된 페이지 테이블들로부터의 어드레스 변환 데이터를 캐싱하기 위한 변환 색인 버퍼(translation lookaside buffer, TLB)(29)를 갖는데, 여기서 페이지 테이블들의 페이지 테이블 엔트리들은 어드레스 변환 매핑들을 정의하고, 또한 파이프라인 상에서 실행되는 주어진 프로세스가 주어진 메모리 영역으로부터의 명령들을 판독, 기록 또는 실행하도록 허용되는지 여부를 관리하는 액세스 권한들을 특정할 수 있다. MMU(28)가 로드/저장 유닛(26)과 연관된 것으로 도시되어 있지만, MMU(28)는 또한 페치 스테이지(6)에 의해 트리거되는 명령 페치들에서 조회될 수 있다(또는 별개의 명령-측 MMU는 데이터 액세스들을 위해 로드/저장 유닛(26)에 의해 사용되는 데이터-측 MMU와 별개로, 명령 페치들을 핸들링하도록 구현될 수 있다 - 이 경우, MMU들 둘 모두는 페이지 테이블들의 공유된 세트로부터 TLB들(29) 내에 정보를 캐싱할 수 있음).

이 예에서, 메모리 시스템은 레벨 1 데이터 캐시(30), 레벨 1 명령 캐시(8), 공유 레벨 2 캐시(32) 및 메인 시스템 메모리(34)를 포함한다. 이것은 가능한 메모리 계층 구조의 하나의 예에 불과하며 다른 배열들의 캐시들이 제공될 수 있다는 것이 인식될 것이다. 실행 스테이지(16)에서 도시되는 특정한 유형의 프로세싱 유닛(20 내지 26)은 단지 하나의 예에 불과하며, 다른 구현들은, 동일한 유형의 다수의 마이크로 동작들이 병렬로 핸들링될 수 있도록, 프로세싱 유닛들의 상이한 세트를 가질 수 있거나 또는 동일한 유형의 프로세싱 유닛의 다수의 인스턴스들을 포함할 수 있다.

장치(2)는 또한, 이들 시스템 컴포넌트들에 매핑된 어드레스들을 특정하는 메모리 액세스 요청들을 발행함으로써 프로세싱 회로(4)에 액세스가능할 수 있는 다수의 시스템 컴포넌트들을 포함할 수 있다. 예를 들어, 시스템 컴포넌트들은 I/O(입력/출력) 포트(44)를 통해 외부 디바이스에 액세스가능할 수 있는 진단 정보를 레코드하는 데 사용되는 추적 버퍼(42)를 포함할 수 있다. 외부 디바이스는, 예를 들어, 소프트웨어로 성능 문제들을 식별하기 위한 목적으로, 프로세싱 회로(4) 상에서 실행되는 소프트웨어의 진단 분석을 수행하기 위해 진단(추적) 정보를 사용할 수 있다. 진단 정보는, 예를 들어, 소프트웨어에서 실행되는 명령들의 시퀀스의 프로그램 흐름, 소프트웨어에 의해 이루어진 데이터 액세스들, 및 소프트웨어를 실행할 때 프로세서의 내부 상태에 관한 정보를 나타낼 수 있다. 추적 하드웨어는 소프트웨어 실행 및 프로세서 상태의 모니터링에 응답하여 추적 정보를 자동으로 레코드하도록 제공될 수 있다. 대안적으로, 소프트웨어가 추적 생성 명령들로 계측된 경우, 소프트웨어 자체는 소프트웨어 정의 추적 데이터가 추적 버퍼(42)에 기록되게 하기 위한 명령(예를 들어, 추적 버퍼(42)에 매핑된 어드레스를 특정하는 기록 액세스 요청)을 실행할 수 있다.

시스템 컴포넌트의 다른 예는, 도 1에 명시적으로 표시되지 않았지만, 도 1의 제1 프로세서에 대해 도시된 것과 유사한 컴포넌트들(4, 8, 14, 28, 30, 32)을 포함할 수 있는 추가 프로세서(46)(예를 들어, 다른 CPU)일 수 있다. 프로세서간 통신은 MMU(28)에 의해 관리되는 판독/기록 메모리 액세스 요청들을 사용하여 수행될 수 있다. 메모리 시스템은 임의의 알려진 코히어런시 프로토콜을 사용하여 각자의 프로세서들의 캐시들 사이의 코히어런시를 유지하기 위한 코히어런트 인터커넥트를 포함할 수 있다.

또한, 시스템은 다수의 주변(입력/출력) 디바이스들(48)을 포함한다. 예를 들어, 주변장치들(48)은 특정 프로세싱 기능을 가속화하도록 설계된 하드웨어 가속기, 네트워크 제어기, 디스플레이 제어기, 사용자 입력 디바이스, 또는 입력/출력 포트를 통해 시스템(2)과 통신하는 임의의 외부(오프-칩) 디바이스를 포함할 수 있다. 다시, 주변장치들 및 I/O 디바이스들과의 상호작용은 이들 컴포넌트들에 매핑된 어드레스들을 특정하는 판독/기록 메모리 액세스들을 발행함으로써 수행된다.

장치(2)는 또한 도 9와 관련하여 아래에서 더 상세히 설명되는 태그 검사 회로(250)를 포함할 수 있다.

도 1은 단지 가능한 프로세서 파이프라인 구현의 일부 컴포넌트들의 단순화된 표현에 불과하며, 프로세서는 간결성을 위해 예시되지 않은 많은 다른 요소들을 포함할 수 있다는 것이 인식될 것이다.

도 2는 "정상" 메모리로 취급되는 어드레스 공간의 페이지들에 대한 판독 및 기록 요청들의 프로세싱을 예시한다. "정상" 메모리 유형은 정규 랜덤 액세스 데이터 또는 명령 저장을 위해 사용되도록 의도된 페이지에 대해 사용될 수 있다. 판독 요청이 수신되면, MMU(28)는 그 TLB(29)에서 판독 요청에 의해 특정된 가상 어드레스를 조회한다. TLB(29)가 가상 어드레스에 대한 페이지 테이블 정보를 이미 포함하는 경우, 페이지 테이블 정보에 의해 제공되는 속성들 및 어드레스 변환 매핑 정보는 가상 어드레스의 물리적 어드레스로의 변환을 제어하고, 판독 요청이 허용될 수 있는지 여부를 결정하기 위해 액세스 권한들을 검사하고(판독 요청이 허용되지 않는 경우, 결함이 시그널링됨), 속성들에 기반하여 서비스의 판독 요청 방식을 제어하는데 사용된다(예를 들어, 속성들은 페이지에 대한 데이터가 캐싱가능한지를 특정할 수 있음). TLB(29)가 가상 어드레스에 대한 요구되는 페

이지 테이블 정보를 아직 포함하지 않는 경우, 액세스 권한들을 검사하고 어드레스 변환을 진행하기 전에, 메모리 시스템에 저장된 페이지 테이블 구조들로부터 요구되는 페이지 테이블 정보를 획득하기 위해 페이지 테이블 워크 동작이 수행된다. 이어서, 따라서, 페이지 테이블 정보 내의 속성들이 "정상" 메모리 유형을 특정하는 판독 액세스에 대해, 액세스 권한들이 충족되면, 판독 액세스의 가상 어드레스는 물리적 어드레스로 변환되고, 이어서 해당 물리적 어드레스를 특정하는 판독 메모리 액세스 요청은 메모리 시스템으로 전송된다. 메모리 액세스 요청이 캐시들(30, 32) 중 하나에서 히트(hit)되면, 특정된 물리적 어드레스에 대한 데이터는 캐시로부터 판독되어 프로세싱 회로(4)로 반환될 수 있다. 요청이 캐시들(30, 32)에서 누락되면, 메인 메모리(34)로의 액세스가 수행되고 판독 타겟 값은 예를 들어 레지스터(14)에 값을 기록함으로써 프로세싱 회로(4)로 반환된다.

기록 요청의 경우, TLB(29)의 조회 및 (필요한 경우 페이지 테이블 워크)는 판독 요청의 경우와 동일하다. 기록 요청의 가상 어드레스에 대한 페이지 테이블 정보에 정의된 속성들에 기반하여, MMU(28)는 기록 요청이 허용되는지 여부를 결정하고, 허용되면, 타겟 가상 어드레스 및 페이지 테이블 정보에 특정된 어드레스 매핑 정보에 기반하여 획득된 변환된 물리적 어드레스를 특정하는 기록 메모리 액세스 요청을 생성한다. 기록 요청은 (예를 들어, 레지스터(14)로부터 판독된 피연산자에 기반하여) 프로세싱 회로(4)에 의해 제공된 기록 데이터 값이 캐시(30, 32) 및/또는 메인 메모리(34) 내의 대응하는 위치에 기록되도록 요청한다.

다양한 이유로, 소프트웨어가 때때로 0으로 채워지는 것과 같이 미리 결정된 값을 할당받도록 가상 어드레스 공간의 비교적 큰 영역들을 제어할 수 있는 것이 유용할 수 있다. 많은 운영 체제들은 기록 시 복사 0-채움 메모리를 제공하기 위한 메커니즘(소프트웨어)을 구현한다. 일반적으로, 이는 0이 저장되는 물리적 메모리의 단일 범위에 모든 이들 영역들을 매핑하기 위해 가상 어드레스들의 0-채움 영역들에 대한 어드레스 변환 매핑을 설정함으로써 수행된다. 따라서, 커밋될 필요가 있는 물리적 메모리의 양은 페이지 테이블 정보의 단일 페이지 테이블 엔트리에 의해 커버되는 영역(예를 들어, 페이지 또는 블록)으로 감소될 수 있다. 그러나, 4kB보다 큰 페이지 크기들이 지원되거나 메모리의 더 큰 영역들(예를 들어, MB 또는 GB 정도 크기들)이 블록 디스크립터를 사용하여 페이지 테이블 계층의 더 높은 레벨에서 단일 페이지 테이블 엔트리에서 커버될 때, 단순히 0들을 저장하기 위해 이러한 양의 물리적 메모리를 예비하는 것은 비용이 많이 든다.

도 3은 "read-as-X" 메모리라고 하는 MMU(28)에 의해 지원될 수 있는 새로운 메모리 유형을 예시한다. MMU(28)에 의해 사용되는 페이지 테이블 정보는 가상 어드레스 공간의 주어진 페이지를 read-as-X 속성과 연관시킬 수 있고, 이는 해당 페이지에 대한 판독이 특정된 값 X를 반환해야 함을 나타낸다. X의 값은 read-as-X 속성 자체 또는 페이지 테이블 정보에 의해 정의된 다른 정보로부터 암시될 수 있고, 따라서 X의 수치 값을 식별하기 위해 캐시 또는 메모리 액세스를 필요로 하지 않는다. 따라서, 판독 요청에서, MMU(28)가 판독 타겟 가상 어드레스에 대한 대응하는 페이지 테이블 정보로부터 판독 요청에 의해 판독되기를 원하는 데이터의 판독 타겟 블록이 read-as-X 속성과 연관된 페이지 내에 속한다는 것을 식별할 때, MMU(28)는 가상 어드레스를 물리적 어드레스로 변환할 필요가 없고, 그렇지 않으면 판독 데이터의 값을 획득하기 위해 발행될 캐시(30, 32) 또는 메모리(34)에 대한 임의의 액세스를 억제할 수 있다. X의 데이터 값을 획득하기 위해 캐시/메모리 액세스가 억제될 수 있더라도, 페이지 테이블 워크가 필요한 경우, 판독 타겟 가상 어드레스와 연관된 페이지 테이블 정보를 획득하기 위해 캐시/메모리 액세스가 여전히 필요할 수 있다는 점에 유의한다. MMU(28)는 MMU(28)의 내부 회로 로직을 사용하여 특정된 값 X를 생성할 수 있고, 판독 동작의 결과로서 이 값을 프로세싱 회로(4)에 반환할 수 있다. 특정된 값 X는, 예를 들어, 0일 수 있거나, 모든 비트들이 1로 설정된 값, 또는 페이지 테이블 정보 또는 페이지 테이블 정보에 의해 참조되는 다른 정보에서 전용 인코딩을 정당화하기에 충분히 자주 사용될 것으로 예상되는 다른 이진 값과 같은 0이 아닌 값일 수 있다. 특정된 값 X가 데이터의 판독 블록의 크기보다 더 적은 비트들을 갖는 경우, 반환되는 데이터의 판독 블록을 채우는 것이 반복될 수 있다.

따라서, (판독/기록 액세스 권한들을 사용하여 어드레스 변환 및 액세스 제어를 지원하기 위해 기존의 아키텍처 메커니즘들을 사용하여 구현될 수 있는) 위에서 설명된 소프트웨어-구현된 기록 시 복사 메커니즘과는 달리, read-as-X 속성은 MMU(28)에 의해 지원되는 페이지 테이블 정보에서 정의된 새로운 아키텍처 속성이다. 이는 페이지 테이블 정보에서 약간의 추가 인코딩 공간을 필요로 할 수 있지만, 이는 미리 결정된 값으로 메모리의 큰 영역들을 채우기 위한 소프트웨어-구현 기록 시 복사 메커니즘에 비해 몇 가지 이점들을 갖는다. 첫째, 특정된 값 X를 획득하기 위한 캐시 또는 물리적 메모리에 대한 액세스가 없기 때문에, 이는 특정된 값 X가 할당된 영역들에 대해 물리적 메모리가 전혀 커밋될 필요가 없다는 것을 의미하므로, 비교적 큰 페이지/블록 크기가 사용되더라도, 특정된 값 X를 실제로 저장하기 위해 물리적 메모리의 큰 블록을 소비할 필요가 없다. 이것은, 특정된 값 X 이외의 더 의미 있는 값들을 갖는 다른 데이터를 저장하기 위해 더 많은 스토리지가 이용가능하기 때문에, 프로세싱 시스템에서 구현되는 이용가능한 물리적 스토리지가 더 효율적으로 활용될 수 있다는 것을 의미

한다. 이것은 외부(오프 칩) 스토리지로 내보내야 하는 의미 있는 데이터의 양을 감소시켜, 다른 데이터가 필요한 코드의 성능을 개선한다. 또한, 특정된 값 X를 획득하기 위해 캐시 또는 메모리 액세스가 필요하지 않기 때문에, read-as-X 속성을 갖는 페이지를 타겟으로 하는 판독 요청들이 더 빠르게 프로세싱될 수 있고, 따라서 성능이 개선된다.

페이지 테이블 정보가 read-as-X 속성과 연관된 어드레스를 특정하는 기록 요청이 발행되면, 여러 가지 상이한 read-as-X 기록 거동들이 가능하다. 도 3은 몇 가지 옵션들을 예시한다. 제1 옵션은 read-as-X 속성을 갖는 페이지를 타겟으로 하는 기록 요청이 발행될 때 결함을 시그널링하는 것이다. 이는 0-채움 가상 페이지에 대한 제1 기록에서 기록 시 복사를 지원하는데 유용할 수 있으며, 결함은 운영 체제가 해당 가상 페이지에 대한 일부 물리적 메모리를 배정하게 하고, 이어서 가상 페이지가 해당 물리적 메모리에 매핑된 후에 기록을 프로세싱하도록 진행할 수 있다. 다른 옵션은 단순히 기록 동작을 무시하고 데이터를 임의의 물리적 메모리에 기록하지 않는 것이다. 이는 read-as-X 속성이, 결함이 필요한 것은 아니지만 read-as-X 메모리에 대한 기록들을 처리하기 위해 아키텍처적으로 규정된 기법이 있어야 하는 경우 기록 시 복사 이외의 목적에 사용되는 경우 결함들의 가능성을 줄이기 위해 선택될 수 있다(이는 소프트웨어가 메모리를 read-as-X로 지정했다면, 그러한 기록들이 드물게 될 것으로 예상될 수 있음). 제3 옵션은 기록 요청의 기록 데이터가 추적 버퍼(42), 다른 프로세서(46) 또는 주변장치(48)와 같은 미리 결정된 시스템 위치로 포워딩될 수 있다는 것이다. 이러한 시스템 위치들로의 데이터의 포워딩은 그렇지 않으면 가상 어드레스들(read-as-X 속성을 갖지 않음)에 대한 판독 또는 기록 메모리 액세스들을 사용하여 수행될 수 있으며, 이 가상 어드레스들은 해당 시스템 위치에 대응하는 것과 같은 메모리 버스 인프라구조에 의해 식별된 물리적 어드레스로 변환된다. 그러나, 이러한 정기적인 액세스들로, 이러한 포워딩은 대응하는 물리적 어드레스로의 어드레스 변환 및 이어서 메모리 시스템으로 전송될 물리적 어드레스를 특정하는 메모리 액세스 요청들을 요구할 것이고, 따라서 메모리 시스템 내에서 구현되는 매핑 컴포넌트는 타겟 물리적 어드레스에 기반하여, 기록 데이터의 목적지인 시스템 위치를 식별할 수 있다. 대조적으로, read-as-X 속성을 사용함으로써, 기록 데이터가 미리 결정된 시스템 위치로 포워딩된다는 사실이 페이지 테이블 정보로부터 (직접 또는 페이지 테이블 정보에 의해 참조되는 간접 테이블로) MMU(28)에 의해 식별될 수 있고, 따라서 MMU(28)가 그 매핑 컴포넌트를 호출하지 않고 목적지를 직접 특정할 수 있기 때문에, 메모리 시스템에 메모리 액세스 요청을 포워딩하고 데이터의 타겟 위치를 식별하기 위해 매핑 컴포넌트를 대기하는 지연이 감소될 수 있다. 따라서, 데이터를 미리 결정된 시스템 위치로 포워딩하는 read-as-X 거동은 추적 버퍼(42)(프로세싱 시스템 상의 소프트웨어의 실행 시 진단 정보를 레코드하는데 사용됨), 다른 프로세서(46), 또는 주변장치(이를테면 DMA 제어기, 네트워크 제어기, 또는 하드웨어 가속기)와 같은 시스템 컴포넌트들에 대한 가속화된 액세스를 제공할 수 있다.

도 4는 판독 요청들의 핸들링을 도시하는 흐름도이다. 단계(100)에서, 판독 요청은 프로세싱 회로(4)에 의해 발행되어 판독 타겟 값이 메모리 어드레스 공간의 판독 타겟 블록과 연관된 메모리 시스템 위치로부터 판독될 것을 요청한다. 판독 요청은 판독 타겟 블록을 식별하는 판독 타겟 어드레스, 및 선택적으로 크기 파라미터를 특정할 수 있다.

단계(102)에서, MMU(28)는 TLB(29) 내의 판독 타겟 어드레스의 조회를 수행하여, 판독 타겟 블록에 대응하는 타겟 페이지에 대한 페이지 테이블 정보를 찾는다. TLB(29)가 필요한 페이지 테이블 정보를 아직 저장하지 않은 경우, 페이지 테이블 워크 동작은 필요한 페이지 테이블 정보를 획득하기 위해 메모리에 저장된 페이지 테이블 구조들을 횡단하기 위해 수행될 수 있다. 필요한 타겟 페이지의 페이지 테이블 구조에 유효한 페이지 테이블 정보가 정의되어 있지 않으면, 결함이 신호할 수 있고 도 4의 방법이 중단된다.

단계(104)에서, MMU(28)는 획득된 페이지 테이블 정보에 기반하여, 판독 타겟 블록이 read-as-X 속성과 연관된 페이지에 대응하는지 여부를 결정한다. 대응하지 않으면, 단계(106)에서, MMU(28)는 타겟 페이지에 대한 페이지 테이블 정보에서 특정된 임의의 액세스 권한들이 만족되는지 여부를 결정하고, 액세스 권한들의 임의의 위반이 존재하면 단계(108)에서 결함이 시그널링된다. 페이지 테이블 정보에서 특정된 액세스 권한들이 만족되면, 단계(110)에서 MMU(28)는 페이지 테이블 정보에서 특정된 어드레스 매핑 정보에 기반하여 어드레스 변환을 수행하여, 판독 타겟 어드레스로서 특정된 가상 어드레스에 대응하는 타겟 물리적 어드레스를 획득한다. 프로세싱 회로(4)의 일부 상태들에서, 가상 어드레스로부터 물리적 어드레스로의 어드레스 매핑은 페이지 테이블 구조들의 2개의 스테이지들에 의존할 수 있다: 프로세싱 회로(4) 상에서 실행되는 운영 체제의 제어 하에서 설정된 매핑 정보에 기반하여 가상 어드레스로부터 중간 어드레스로의 매핑들을 정의하는 스테이지-1 페이지 테이블 구조, 및 프로세싱 회로(4) 상에서 실행되는 하이퍼바이저의 제어 하에서 설정된 매핑 정보에 기반하여 중간 어드레스로부터 물리적 어드레스로의 매핑들을 정의하는 스테이지-2 페이지 테이블 구조. 프로세싱 회로(4)의 다

른 동작 상태들에서, 변환은 가상 어드레스로부터 물리적 어드레스로 직접 단일 스테이지에서 있을 수 있다. 어드레스 변환을 수행하기 위해 임의의 알려진 어드레스 변환 기법이 사용될 수 있다.

단계(112)에서, MMU(28)는 변환된 물리적 어드레스를 프로세싱 회로(4)의 로드/저장 유닛(26)으로 반환하고, 로드/저장 유닛(26)은 판독 타겟 값의 적어도 일부가 레벨 1 캐시(30)로부터 반환되는 것을 요청하기 위해 캐시 액세스 요청을 발행한다. 요청이 레벨 1 캐시에서 누락되면, 추가 요청들이 레벨 2 캐시(32) 또는 메인 메모리(34)로 전송될 수 있다. 일부 경우들에서, 캐시 액세스는, 예를 들어 페이지 테이블 정보가 캐시 가능하지 않은 페이지를 표시하는 경우, 바이패스될 수 있고, 이 경우 요청은 메모리(34)를 직접 타겟팅할 수 있다. 판독 타겟 값은 메모리 시스템의 저장 구조들(30, 32, 34) 중 하나로부터 획득되고, 프로세싱 회로(4)로 반환되어, 예를 들어, 레지스터들(14)에 값을 저장하고, 그로부터 값은 프로세싱 회로(4)에 의해 프로세싱되는 후속 명령에 대한 피연산자로서 사용될 수 있다.

다른 한편, 단계(104)에서, 판독 타겟 블록이 read-as-X 속성과 연관된 페이지에 대응하는 것으로 결정된다면, 단계(114)에서 MMU(28)는 다시 페이지 테이블 정보에 특정된 임의의 액세스 권한들이 판독 요청에 의해 만족되는지 여부를 결정하고, 그렇지 않으면 단계(116)에서 결함이 시그널링된다. 액세스 권한들이 만족되는 것으로 결정되면, 단계(118)에서 캐시 또는 메모리 액세스 요청(이는 통상적으로 판독 타겟 데이터가 프로세싱 회로(4)로 반환되는 것을 요청하기 위해 발행될 수 있음)이 억제되고, 대신 단계(120)에서, 특정된 값 X는 판독 타겟 블록의 적어도 일부에 대응하는 값으로서 프로세싱 회로(4)로 반환된다. 다시, 반환된 데이터 값은 프로세싱 회로의 레지스터(14)에 저장될 수 있다. 특정된 값 X는 페이지 테이블 정보 또는 페이지 테이블 정보에 의해 참조되는 다른 정보로부터 추론되므로, 특정된 값 X를 결정하기 위해 어떠한 캐시 또는 메모리 액세스도 필요로 하지 않는다.

드물게, 판독 타겟 블록은 read-as-X 속성을 갖는 것으로 표시된 하나의 페이지와 read-as-X 속성을 갖지 않는 다른 페이지 사이의 페이지 경계를 넘을 수 있다. 이 경우, read-as-X 페이지 및 비-read-as-X 페이지에 대한 판독 액세스의 부분들은 단계(102)에서 페이지 테이블 정보의 별개의 조회들을 필요로 하고 비-read-as-X 페이지에 대한 각자의 단계들(106 내지 112) 및 read-as-X 페이지에 대한 단계들(114 내지 120)의 별개의 프로세싱을 트리거할 2개의 별개의 액세스들로서 효과적으로 취급될 수 있다. 따라서, 이 경우, 판독 요청에 의해 요청된 판독 타겟 값의 일부만이 MMU(28)에 의해 특정된 값 X로서 생성될 것이며, 단계(112)에서 비-read-as-X 페이지에 대응하는 다른 부분은 메모리 시스템으로부터 획득될 것이다. 따라서, 판독 요청이 read-as-X 페이지를 부분적으로 타겟팅할 수 있기 때문에 판독 요청의 전체 판독 타겟 값이 특정 값 X로서 반환되는 것이 필수적인 것은 아니다.

도 5는 기록 요청의 프로세싱을 예시하는 흐름도이다. 단계(140)에서, 기록 요청은 메모리 어드레스 공간의 기록 타겟 블록과 연관된 메모리 시스템 위치들에 기록 값의 기록을 요청하는 프로세싱 회로(4)에 의해 발행된다. 판독 요청에 대해, 단계(142)에서, MMU(28)는 TLB(29)에서의 기록 요청에 의해 특정된 기록 타겟 어드레스의 조회를 수행하여, 어드레스들의 기록 타겟 블록에 대응하는 타겟 페이지에 대한 페이지 테이블 정보를 찾는다. 다시, TLB(29)가 필요한 정보를 아직 갖고 있지 않으면, 메모리 내의 페이지 테이블 구조들에 액세스하기 위해 페이지 테이블 워크가 수행될 수 있다. 단계(144)에서, MMU(28)는, 획득된 페이지 테이블 정보에 기반하여, 어드레스들의 기록 타겟 블록이 read-as-X 속성과 연관된 페이지에 대응하는지 여부를 결정한다.

기록 타겟 블록과 연관된 타겟 페이지가 read-as-X 페이지가 아닌 경우, 단계(146)에서, 대응하는 페이지 테이블 정보에 의해 특정된 임의의 액세스 권한들이 검사되고, 액세스 위반이 식별되면, 단계(148)에서 결함이 시그널링된다. 기록 요청에 의해 타겟팅된 페이지에 대해 특정된 액세스 권한들이 만족되면, 단계(150)에서, MMU(28)는 기록 액세스의 타겟 물리적 어드레스를 획득하기 위해, 도 4의 단계(110)와 동일한 방식으로 어드레스 변환을 수행한다. 단계(152)에서, 캐시 또는 메모리 액세스 요청은 타겟 물리적 어드레스와 연관된 (캐시(30, 32) 또는 메모리(34) 내의) 메모리 시스템 위치에 기록 요청에 의해 특정된 기록 값의 기록을 요청하기 위해 로드/저장 유닛(26)에 의해 발행된다.

단계(144)에서, 기록 타겟 블록이 read-as-X 속성과 연관된 페이지에 대응하는 것으로 결정된다면, 단계(154)에서, MMU(28)는 타겟 페이지에 대한 액세스들을 위해 만족되는데 필요한 임의의 액세스 권한들을 다시 검사하고, 임의의 액세스 권한들이 만족되지 않으면 단계(156)에서 결함이 시그널링된다.

read-as-X 페이지에 대한 기록 액세스에 대해, 기록들이 허용되는 데 필요한 임의의 다른 액세스 권한들이 만족된다고 결정되면, 어드레스 공간의 read-as-X 페이지에 대한 기록 요청에 응답하는 여러 옵션들이 있다. 제1 옵션에서, 단계(158)에서, 결함이 시그널링된다. 제2 옵션에서, 160에서, 기록 요청은 캐시 또는 메모리에 임

의의 기록 요청을 발행하지 않고 단순히 무시되지만, 어떠한 결함도 시그널링되지 않으며, 따라서 프로세싱은 프로세싱 회로(4)에 의해 중단되지 않고 계속되도록 허용된다. 제3 옵션에서, 단계(162)에서, 기록 요청에 의해 특정된 기록 값은 추적 버퍼(42), 다른 프로세서(46) 또는 주변장치(48)와 같은 미리 결정된 시스템 위치로 포워딩된다. 일부 구현들은 단계들(158, 160 및 162)이 나타내는 옵션들 중 하나만을 지원할 수 있다. 다른 구현들은 이러한 옵션들 중 2개 이상을 지원할 수 있고, 어떤 옵션이 특정 기록 요청에 취해지는지는 소프트웨어에 의해 프로그래밍가능한 제어 정보에 의존할 수 있다(예를 들어, 이러한 제어 정보를 프로그램 하는 능력은 운영 체제 또는 하이퍼바이저 소프트웨어와 같은 소정 특권 레벨 이상의 소프트웨어로 제한될 수 있음). 예를 들어, 제어 정보는, 어느 read-as-X 기록 거동이 read-as-X 페이지로의 기록 요청에 응답하여 취해지는지를 직접적으로 특정할 수 있는 페이지 테이블 정보 자체일 수 있거나, 또는 제어 정보는 read-as-X 페이지로의 기록 요청들을 핸들링하는 방법에 대한 더 많은 정보를 제공하는 페이지 테이블 정보에 의해 참조되는 간접 테이블일 수 있다. 대안적으로, 어떤 read-as-X 기록 거동을 사용할지를 제어하는 제어 정보는 페이지 테이블 정보와 독립적일 수 있고 페이지 테이블 정보에 의해 직접 또는 간접적으로 참조되지 않을 수 있다(예를 들어, 제어 정보는 개별 페이지에 특정한 페이지-특정 정보가 아니라, 모든 페이지들에 대한 read-as-X 기록 거동을 제어하는데 사용되는 글로벌 정보일 수 있음). 예를 들어, 제어 레지스터는 현재의 동작 모드에서 사용될 read-as-X 기록 거동을 나타내는 모드 상태 값을 나타낼 수 있고, 이는 read-as-X 속성과 연관된 어드레스 공간의 임의의 페이지들에 대해 전역적으로 적용될 수 있다.

페이지 테이블 정보는 다수의 다양한 방식으로 read-as-X 속성을 인코딩할 수 있다. 일부 경우들에서, 어드레스 공간의 주어진 페이지에 대한 페이지 테이블 엔트리는 페이지가 read-as-X 속성을 갖는 것을 식별하는 메모리 속성 또는 권한 유형 값을 직접 특정할 수 있다. 예를 들어, 페이지 테이블 엔트리의 권한들 또는 속성 필드의 하나 이상의 비트필드 인코딩들은 read-as-X 속성을 나타내도록 할당될 수 있고, 그 필드의 다른 인코딩들은 권한들 또는 속성들의 다른 유형들을 식별한다.

대안적으로, 도 6에 도시된 바와 같이, read-as-X 속성에 대한 정보를 정의하기 위해 간접 테이블이 사용될 수 있다. 이는 페이지 테이블 형식이 read-as-X 속성의 원하는 수의 변형들에 대한 지원을 추가하기에 충분한 여분의 인코딩 공간을 갖지 않는 경우에 유용한다. 이는, 위에서 논의된 바와 같이, 특정된 값 X의 상이한 값들에 대응하는 read-as-X의 변형들 또는 상이한 read-as-X 기록 거동들과 같은, 지원되기를 원하는 read-as-X 속성의 하나 초과의 변형들이 존재할 경우에 특히 유용할 수 있다.

도 6에 도시된 바와 같이, 페이지 테이블 구조(200)는 가상 어드레스를 변환된 어드레스(이 예에서는 물리적 어드레스이지만, 2-스테이지 변환 프로세스의 스테이지-1 변환에서는 변환된 어드레스가 대신 중간 어드레스일 수 있음), 및 하나 이상의 액세스 권한 또는 속성 필드들과 연관시킬 수 있다. 설명의 편의를 위해 가상 어드레스가 도 6의 테이블 구조에 명시적으로 도시되어 있지만, 실제로 가상 어드레스는 구조를 인덱싱하는데 사용될 수 있으므로 테이블 구조 자체에 명시적으로 저장될 필요가 없다는 것이 이해될 것이다.

이 예에서, 권한 필드는 MMU(28)의 제어 레지스터들로서 구현되는 권한 간접 레지스터들(202)의 세트를 참조한다. 권한 간접 레지스터들은 소정 권한 레벨 이상에서 실행되는 소프트웨어 명령들에 의해 프로그래밍 가능하고, 지원되는 다양한 사용 권한 유형들 중에서 선택할 수 있는 간접 테이블을 나타낸다. 권한 간접 레지스터들을 사용함으로써, 페이지 테이블 엔트리들 자체의 권한 필드에서 제한된 비트들의 수로 표시될 수 있는 것보다 더 많은 수의 권한 유형들이 지원될 수 있다. 예를 들어, 비트 권한 필드는 권한 간접 레지스터들(202)의 4개의 권한 간접 엔트리들 중 하나로부터 선택할 수 있지만, 이어서 권한 간접 레지스터들(202)의 각각의 엔트리는 대응하는 권한 유형에 대해 4개 초과의 상이한 옵션들 중에서 선택할 수 있다. 물론, 권한 간접 엔트리들의 수는 4개보다 많거나 적을 수 있다.

이어서, 이는 주어진 페이지에 대해 다음과 같은 다양한 권한 유형들을 지원할 수 있게 한다:

- 판독 전용: 판독 액세스들은 허용되고, 기록 또는 명령 페치 액세스들은 금지되고;
- 액세스들 안됨: 모든 페이지 엑세스는 금지되고;
- 판독/실행 액세스: 판독 및 명령 페치 액세스들은 허용되고 기록 액세스들은 금지되고;
- 0으로 판독: 특정된 값이 0인 read-as-X 속성의 변형(예를 들어, 이 변형에서 기록-결함 기록-거동은 도 5의 단계(158)에 도시된 바와 같이 사용될 수 있음);

- 판독/기록: 판독 및 기록 액세스들은 허용되고 명령 페치 액세스들은 금지되고;

- Read-as-0 write-ignore: read-as-X 페이지에 기록이 요청될 때 도 5의 단계(160)의 기록-무시 동작이 사용되는 Read-as-zero의 변형;

- Read-as-0 write-to-trace-buffer: 사용할 read-as-X 기록 거동이 도 5의 옵션 3인 Read-as-zero의 변형으로서, 미리 결정된 시스템 위치는 추적 버퍼(42)임;

- 어드레스에 따른 read-as-X: X 값이 아래 추가로 도 8에 설명된 바와 같이 판독 요청의 어드레스에 따라 달라지는 read-as-X 속성의 변형.

이들은 가능할 수 있는 단지 일부 예시적인 인코딩일뿐이고, 많은 다른 권한 유형들이 또한 사용될 수 있다는 것이 인식될 것이다. 일반적으로, 소프트웨어는 페이지 테이블 엔트리들 그 자체에서 요구되는 비트들의 수를 지원되는 권한 유형들의 전체 범위로부터 선택을 인코딩하는데 요구될 비트들의 수 미만으로 감소시키기 위해, 페이지 테이블 엔트리들에 의한 선택을 위해 ISA에서 지원되는 이용가능한 권한 유형들의 특정 서브세트를 선택하기 위해 권한 간접 레지스터들에 기록할 값들을 선택할 수 있다. 도 6은 페이지 테이블 엔트리들의 권한 필드에 대한 선택에 이용가능한 옵션들을 특정하는 것으로서 간접 레지스터들(202)을 도시하지만, 다른 구현들은 속성 필드에 간접 테이블 접근법을 적용할 수 있다. read-as-X 속성은 권한이 아닌 메모리 속성으로 간주될 수 있으므로, 도 6의 예와 같이 권한 필드 대신 속성 필드 내에서 직접 또는 간접적으로 인코딩될 수 있다.

도 7은 메모리의 read-as-X 페이지에 대한 판독을 위해 반환될 특정 값 X를 특정하기 위해 추가 간접의 사용을 예시한다. 다시, 페이지 테이블 구조(200)는 MMU(28)의 제어 레지스터들에 특정된 간접 테이블(202)을 사용하여, 주어진 페이지에 대해 어떤 권한 유형이 선택되는지를 정의하고, 페이지 테이블 엔트리는 그것의 권한 필드들에 간접 테이블(202)의 엔트리들 중 하나를 특정하는 엔트리 식별자를 특정한다. 간접 테이블 엔트리들의 적어도 일부 인코딩들에 대해, 간접 테이블 엔트리의 값은 제어 레지스터들에 저장된 추가적인 간접 테이블(204)의 엔트리에 대한 추가적인 참조로서 해석될 수 있다. 이는 read-as-X 페이지에 대한 특정된 값 X에 대해 더 넓은 범위의 가능한 값들을 지원하는데 유용할 수 있고, 여기서 초기 간접 테이블은 페이지가 read-as-X 페이지인지 비-read-as-X 페이지인지를 특정할 수 있고, read-as-X 기록 거동 또는 read-as-X 페이지에 대한 기록 데이터와 함께 포워딩될 시스템 컴포넌트의 유형과 같은 read-as-X 속성의 다른 속성들을 특정할 수 있지만, 적어도 일부 인코딩들에 대해서는 메모리의 read-as-X 페이지를 표시하는 것으로서 지정되고, 인코딩은 추가의 간접 테이블(204)에 대한 참조로서 해석될 수 있으며, 여기서 참조된 엔트리는 4개의 판독들이 메모리의 read-as-X 페이지로 반환될 특정 값을 특정한다. 이 접근법은, 메인 페이지 테이블들의 권한 필드를 충분한 비트들의 수로 확장하여 특정된 값을 표시할 필요 없이, 광범위한 수치 값들이 read-as-X 액세스들에 대해 특정될 수 있게 할 것이다. 이는, 간접 테이블(202) 및 추가 간접 테이블(204)이 모든 페이지들에 대해 전역적으로 한번 특정될 수 있는 반면, 페이지 테이블 엔트리들(200)이 페이지-별로 제공될 것이며, 따라서 임의의 데이터 값들을 read-as-X 페이지에 대해 반환될 특정된 값 X로서 특정하기 위해 페이지 테이블 구조의 권한 또는 속성 필드들의 크기를 확장하는 것이 페이지 테이블 구조들의 크기를 크게 증가시킬 것이기 때문에 유익하다.

도 7이 특정된 값 X에 대한 다수의 상이한 값들로부터의 선택을 특정하기 위해 추가 간접 테이블(204)을 사용하는 접근법을 도시하지만, 다른 접근법들에서, X의 특정한 값의 선택은 간접 테이블(202)의 다른 정보와 결합될 수 있어서, 추가 간접 테이블(204)이 필요하지 않다.

도 8은 도 6에 표시된 read-as-X 속성의 "어드레스에 따라 값 X로 판독" 변형에 대한 예시적인 거동을 더 상세히 예시한다. 이 경우, read-as-X 속성의 인코딩은 동일하지만 어드레스 공간의 상이한 범위들에 있는 2개의 페이지들의 경우, 해당 페이지들에 대한 판독 액세스들을 위해 특정된 값 X의 상이한 값들이 반환될 수 있다. 예를 들어, 판독 타겟 어드레스의 하나 이상의 비트들(예를 들어, 메모리 내의 메모리 액세스 위치들을 선택하기 위해 사용되지 않는 임의의 상위 비트들을 제외한 어드레스의 최상위 부분)은 어떤 값이 특정된 값 X로서 반환되어야 하는지를 선택하기 위해 사용될 수 있다. 이는 일부 명령 세트 아키텍처들이 사용자 데이터 및 사용자 프로그램 코드에 대해 어드레스 공간의 하위 부분(상위 비트들이 0과 동일함)의 어드레스를 사용하고 커널(운영 체제) 데이터 및 커널 프로그램 코드에 대해 어드레스 공간의 상위 부분(상위 비트들이 1과 동일함)의 어드레스를 사용하는 것과 같이, 상이한 목적들을 위해 어드레스 공간의 상이한 범위들을 사용하기 때문에 유용할 수 있다. 일부 사용 경우들에 대해, read-as-X 메모리에 대한 액세스들에 대해 특정된 값 X로서 반환될 디폴트 값은 사용자 레벨 액세스들 및 커널 레벨 액세스들에 대해 상이해야 하는 것이 바람직할 수 있다. 예를 들어,

read-as-X 접근법이 아래의 도 9와 관련하여 추가로 논의되는 바와 같이 메모리 태깅을 지원하는 시스템에서 배정 태그를 핸들링하기 위해 사용될 때, 사용자 액세스들 및 커널 액세스들에 의해 각각 사용되는 read-as-X 페이지에 상이한 태그 값들을 할당하여, 커널 데이터에 대한 사용자 액세스가 미스매칭 태그들을 초래하는 것은 유용할 수 있다. read-as-X 속성의 어드레스-의존 변형을 제공함으로써, 이는 어드레스 공간의 상이한 범위들에서 사용될 X의 상이한 값들에 대해 2개의 별개의 권한/속성 유형 인코딩들이 배정될 필요성을 회피하고, 페이지 테이블 정보 또는 간접 테이블에서 인코딩 공간을 절약하고, 이는 다른 목적들을 위해 페이지 테이블 엔트리 또는 간접 테이블 엔트리의 적어도 하나의 다른 인코딩을 자유롭게 할 수 있다. 도 8은 어드레스 공간이 2개로 나누어지고, 따라서 2개의 상이한 범위들의 어드레스들이 특정된 값 X의 상이한 값들에 대응하는 예를 도시하지만, 다른 접근법들이 특정된 값 X에 대해 어떤 값을 사용할지를 선택하기 위해 타겟 어드레스의 2 비트 초과를 사용할 수 있고, 이 경우 어드레스 공간이 read-as-X 페이지들에 대해 상이한 X 값들에 대응하는 2개 초과의 영역들로 효과적으로 나누어질 수 있다는 것이 인식될 것이다.

또한, X의 값을 선택하기 위해 어드레스의 최상위 비트를 사용하는 것이 반드시 필요한 것은 아니다. X의 값을 선택하기 위해 어드레스의 덜 중요한 비트 포지션에서 하나 이상의 비트들을 사용하는 예에서, 이는 X의 값에서 교번하는 어드레스 공간의 다수의 교번 영역들의 효과를 가질 것이고, 이는 예를 들어, 주어진 메모리 범위에 대한 배정 태그 값들을 블록 단위로 교번 값들로 설정하는데 필요한 물리적 저장 및 명시적 태그 기록 동작들의 양을 감소시키기 위해 배정 태그 값들을 순환시키는 데 유용할 수 있다.

read-as-X 속성은 이러한 태그들을 저장하기 위해 물리적 메모리를 커밋할 필요 없이 소정 태그 값을 갖는 배정 태그들이 표현될 수 있게 하기 위해 메모리 태깅을 지원하는 시스템들에 특히 유용할 수 있다. 도 9는 C 및 C++와 같은 일부 상위 레벨 프로그래밍 언어들이 메모리 액세스들과 관련된 소정 에러들을 검사하기 위한 런타임 에러 검출 검사들을 포함하지 않는다는 점에서 메모리에 안전하지 않은 것으로 간주되는 문제를 다루는 데 사용될 수 있는 메모리 태깅의 개념을 예시한다. 메모리에 안전하지 않은 언어들의 지속적인 사용은 주어진 명령 세트 아키텍처에 따른 컴파일된 코드에서, 공격자 또는 기타 악의적인 당사자에 의한 악용에 취약할 수 있는 메모리 관련 에러들이 많이 있을 수 있음을 의미한다. 이러한 에러들은 다음을 포함할 수 있다:

- 코드에 의해 공급되는 어레이 인덱스가 어레이의 올바른 경계들을 벗어나는, 경계 위반들(Bounds violations);

- 메모리 위치가 이미 배정해제(deallocate)되거나 해제(free)된 후 메모리 위치에 대한 액세스가 이루어지는, 해제 후 사용 에러들(Use-after-free errors);

- 함수 내에서 사용되는 변수(예를 들어 스택 상의 값)와 연관된 어드레스에 대한 메모리 액세스가 함수로부터 이미 반환된 후에 이루어지는, 반환 후 사용(Use-after-return);

- 변수들이 선언된 범위 밖에서 변수에 액세스되는, 범위 외 사용 에러들(Use-out-of-scope errors);

- 변수가 초기화되기 전에 변수와 연관된 메모리 어드레스에 액세스하는, 초기화 전 사용 에러들(Use-before-initialisation errors).

이는 예측할 수 없는 거동을 초래할 수 있고 잠재적으로 공격자들이 악용할 수 있는 접근 수단을 제공할 수 있는 메모리 관련 에러들의 몇 가지 예들이다. 따라서, 메모리 에러들의 소정 클래스들의 런타임 검출을 보조하기 위해, 주어진 프로세싱 장치에 의해 지원되는 명령 세트 아키텍처 내에서 아키텍처 지원을 제공하는 것이 바람직할 수 있다.

위에서 논의된 유형의 소정 메모리 사용 에러들에 대해 보호하기 위한 하나의 접근법은 하나 이상의 메모리 위치들의 블록들과 연관하여 메모리 시스템에 저장되는 배정 태그들을 제공하는 것일 수 있다. 메모리 시스템 내의 특정 어드레싱된 위치를 식별하는 타겟 어드레스에 기반하여 태그 보호 메모리 액세스 동작이 요청될 때, 태그 검사 회로는 타겟 어드레스에 의해 식별된 어드레스된 위치를 포함하는 하나 이상의 메모리 위치들의 블록과 연관하여 메모리 시스템에 저장된 배정 태그와 타겟 어드레스와 연관된 어드레스 태그를 비교할 수 있다. 태그 검사 회로는 배정 태그와 어드레스 태그 사이에서 매칭이 검출되는지 여부의 표시를 생성할 수 있다. 이 표시는 메모리 액세스가 성공하도록 허용되는지 또는 후속 동작들이 성공할 수 있는지, 또는 메모리 액세스들이 정상적으로 계속되도록 허용하는 동안 단순히 보고될 수 있는지 여부를 제어하는 데 사용될 수 있다.

이것은 예를 들어 C 또는 C++와 같은 메모리에 안전하지 않은 언어에 기반하여 코드를 컴파일하는 컴파일러가

메모리의 영역들이 초기화될 때, 코드가 액세스할 것으로 예상되는 메모리의 블록들에 대한 배정 태그들이 특정 값들로 설정되게 하고, 매칭되는 어드레스 태그 값들을 이들 블록들을 포인팅하는 타겟 어드레스들을 제공하는 데 사용되는 포인터들과 연관시키게 하는 명령들을 생성할 수 있기 때문에 유용할 수 있다. 메모리 사용 에러가 발생하고, 예를 들어 어드레스 포인터가 범위 밖에서 사용되거나 초기화된 유효 범위의 경계들 밖으로 연장되는 경우, 어드레싱된 위치와 연관된 배정 태그가 타겟 어드레스와 연관된 어드레스 태그와 매칭하지 않을 수 있고, 이어서 매칭이 검출되는지 여부의 표시가 일부 에러 핸들링 응답 또는 에러 보고 메커니즘을 트리거링하는 데 사용될 수 있다. 취해진 특정 응답은 실행되는 소프트웨어의 특정 요구들 또는 아키텍처의 특정 마이크로-아키텍처 구현에 의존할 수 있다. 따라서, 고급 레벨 언어가 메모리 액세스 에러들에 대해 보호되기 위해 런타임 에러 검사들을 수행하기 위한 수단을 갖지 않더라도, 컴파일된 코드에 대해 사용되는 ISA는 이러한 검사들을 수행하기 위한 아키텍처 특징들을 포함할 수 있다.

따라서, 도 9에 도시된 바와 같이, 태그 검사 모드가 지원될 때, 메모리 시스템 내의 메모리 위치들을 참조하기 위해 사용되는 물리적 어드레스 공간(244)은 각각 소정 수의 어드레싱가능한 위치들(246)을 포함하는 다수의 블록들(245)로 논리적으로 파티셔닝될 수 있다. 간결함을 위해, 도 9의 예에서, 각각의 블록(245)은 4개의 메모리 위치들(246)을 포함하지만, 다른 블록 크기들이 또한 사용될 수 있다. 각각의 블록(245)은 대응하는 배정 태그(240)와 연관된다. 소정 수의 블록들(245)과 연관된 배정 태그들은 함께 수집되고, 물리적 어드레스 공간 내의 상이한 아키텍처적으로 액세스가능한 메모리 위치 내에, 또는 아키텍처적으로 액세스가능하지 않은(동일한 물리적 어드레스 공간에 매핑되지 않은) 메모리 시스템에 제공된 추가적인 저장 위치들 내에 저장될 수 있다. 어느 태그 저장 위치들이 각각의 블록(245)에 대응하는지의 특정 매핑은 하드와이어링될 수 있거나 또는 프로그래밍가능할 수 있다.

따라서, 태그 보호 메모리 액세스가 요구될 때(예를 들어, 소정 메모리 액세스 명령 연산코드들이 태그 보호 메모리 액세스들을 나타내기 위해 할당될 수 있거나, 또는 대안적으로 태그 검사 모드에서 동작할 때 발행된 임의의 메모리 액세스가 태그 보호되는 것으로 간주될 수 있음), (액세스될 어드레싱된 위치를 식별하는 타겟 어드레스(247)와 연관된) 어드레스 태그(246)는 어드레싱된 위치를 포함하는 메모리 위치들의 블록(245)과 연관된 배정 태그(240)와 비교된다. 예를 들어, 도 9에서 타겟 어드레스(247) 포인트는 메모리의 소정 위치(B1)를 가리킨다. 따라서, 위치 B1을 포함하는 위치들 B의 블록과 연관된 배정 태그 B는 태그 검사 회로(250)에 의해 타겟 어드레스(247)와 연관된 어드레스 태그(246)에 대해 비교된다. 도 1에 도시된 바와 같이, 태그 검사 회로(250)는 프로세싱 회로(4)의 일부로 간주될 수 있거나, 대안적으로(도 1에 도시되지 않음) 태그 검사 회로(250)는 메모리 시스템의 일부로서 구현될 수 있다. 도 9의 상단에 도시된 바와 같이, 어드레스 태그(246)는 타겟 어드레스(247) 자체의 선택된 비트들의 함수로서 결정될 수 있다. 특히, 어드레스 태그는 어드레싱된 위치(44)로서 선택될 특정 메모리 위치를 표시하기 위해 사용되지 않는 타겟 어드레스의 일부 내의 비트들로부터 결정될 수 있다(타겟 위치(B1)의 선택은 이러한 비트들에 의존하지 않음). 예를 들어, 일부 아키텍처들에서, 타겟 어드레스의 비트들의 상단 부분은 항상 부호 확장(모두 0들 또는 모두 1들)과 같은 소정 고정 값을 가질 수 있고, 따라서 의미있지 않다 - 이러한 상위 비트들은 액세스할 메모리 위치를 선택하는데 사용되지 않는다. 따라서, 어드레스는 임의의 태그 값으로 이들 미사용 비트들을 덮어씀으로써 어드레스 태그(246)로 태깅될 수 있다. 특정 어드레스 태그 값은 예를 들어 프로그래머 또는 컴파일러에 의해 선택될 수 있다. 어드레스 태그(246) 및 배정 태그(240)는 비교적 적은 수의 비트들, 예를 들어 4 비트들일 수 있고, 따라서 메모리 스토리지 내에서 그리고 타겟 어드레스 내에서 많은 공간을 점유할 필요가 없다. 4 비트의 태그 공간, 즉 16개의 가능한 태그 값을 제공하는 것은 종종 많은 일반적인 유형의 메모리 액세스 에러를 검출하기에 충분할 수 있다.

따라서, 태그 보호 메모리 액세스가 수행될 때, 태그 검사 회로(250)는 타겟 어드레스와 연관된 어드레스 태그(246)를 어드레싱된 위치(B1)를 포함하는 메모리 위치들의 블록(245)과 연관된 배정 태그(240)와 비교하고, 이들이 매칭되는지를 결정한다. 태그 검사 회로(250)는 어드레스 태그(246)와 배정 태그(240)가 매칭되는지 여부를 표시하는 매칭 표시를 생성한다. 예를 들어, 이러한 일치 표시는 어드레스 태그(246)와 배정 태그(250) 사이에 불일치가 존재하는 경우 생성되어, 예외 핸들러가 결함의 원인을 다루도록 트리거링하는 결함 신호일 수 있다. 또한, 매치 표시는 매칭이 있었는지 여부를 표시하는 상태 레지스터에 배치된 표시, 또는 에러가 검출된 어드레스 및/또는 에러를 트리거링한 명령 어드레스의 명령을 표시하기 위해 에러 보고에 추가된 엔트리일 수 있다. 일부 구현예들은 태그 검사 에러를 야기하는 메모리 액세스가 성공하는 것을 방지할 수 있는 반면, 다른 접근법들은 태그 검사 에러에도 불구하고 메모리 액세스가 진행되게 할 수 있지만, 소프트웨어의 후속 버전들에서의 에러들을 제거하는 것을 공급자가 보조하기 위해 코드의 소프트웨어 공급자에게 이용가능하게 될 수 있는 에러 로그에 에러를 기록할 수 있다. 따라서, 태그 매칭/불일치 표시가 태그 검사 회로(250)에 의해 생성될 수

있는 방식들의 범위가 존재한다는 것이 이해될 것이다.

어드레스 태그(246)를 결정하기 위해 타겟 어드레스(247)의 일부를 사용하는 실시예에서, 이 어드레스 태그는 타겟 어드레스에 의해 식별된 어드레싱된 위치로부터의 정보가 캐시 내에 저장되는지 여부를 결정하기 위해 캐시(30, 32)에 의해 사용될 수 있는 타겟 어드레스의 태그 일부와 상이하다는 점에 유의한다. 많은 캐싱 스킴(scheme)들은 데이터의 캐싱된 피스(piece)의 어드레스의 태그 부분을 그 데이터와 함께 캐시 내에 저장하여, 주어진 어드레스에 대한 캐시를 검색할 시, 캐싱된 데이터가 실제로 요구되는 어드레스에 대응하는지의 여부를 결정하기 위해, 어드레스의 일부가 캐싱된 데이터와 함께 저장된 태그들과 비교될 수 있다. 그러나, 이 경우, 캐시 내의 태그와 비교되는 어드레스의 태그 부분은 데이터가 요구되는 특정 어드레싱된 위치를 실제로 식별하는 어드레스의 일부의 부분일 것이고, 즉, 어드레스의 캐시 태그 부분을 변경하는 것은 정의에 의해 어드레스가 메모리 시스템 내의 상이한 어드레싱된 위치를 가리키게 할 것이다. 대조적으로, 태그 보호 메모리 동작에 사용되는 어드레스 태그와 함께, 메모리 액세스 회로는 어드레스 태그와 독립적으로 데이터 액세스가 요구되는 어드레싱된 위치를 선택할 수 있다. 즉, 어드레스 태그(246)가 상이한 값들로 설정되더라도, 타겟 어드레스에 의해 참조되는 어드레싱된 위치(B1)는, 어드레싱된 위치(B1)의 선택이 타겟 어드레스(247)의 다른 일부에만 의존하고 어드레스 태그(246)에 사용되는 부분과 무관하기 때문에 여전히 동일할 수 있다. 이는 컴파일러들이 메모리 시스템의 관련 데이터 블록들에 배정된 대응하는 배정 태그 값들과 매칭되도록 특정 어드레스와 연관된 어드레스 태그를 임의의 값으로 자유롭게 설정할 수 있게 제공한다.

태그 검사 회로(250)는 시스템(2)의 상이한 부분들에 위치될 수 있다. 일부 경우들에서, 태그 검사 회로는 도1에 도시된 바와 같이 프로세싱 요소(예를 들어, CPU) 내에 위치될 수 있다. 다른 예들은 태그 검사 회로를 액세스된 데이터와 태그를 저장하는 메모리 스토리지(16)에 더 가깝게 위치시킬 수 있다.

메모리 태깅을 지원하는 시스템에서, 메모리 태깅을 사용하여 보호되기를 원하는 어드레스 공간의 범위들에 대해 배정 태그들(240)을 정의하기 위해, 이것은 (read-as-X 속성에 대한 지원이 없는 경우) 배정 태그들의 저장을 위해 커밋될 비교적 큰 양의 물리적 메모리 스토리지를 요구할 수 있다. 그러나, 실제로 배정 태그들의 합리적인 분율이 0과 같은 특정된 값으로 설정될 수 있고, 따라서 이것은 이용가능한 물리적 저장의 비교적 상당한 부분을 낭비할 수 있다는 것이 인식된다. 특정된 값 X를 갖는 배정 태그들을 저장하기 위해 할당된 페이지들에 대해 read-as-X 속성을 사용함으로써, 이는 태그 값들을 저장하기 위해 커밋되어야 하는 물리적 메모리의 양을 감소시키고, 따라서 이용가능한 물리적 스토리지를 더 효율적으로 사용하게 한다. 따라서, 태그 보호된 액세스가 요청될 때, 대응하는 배정 태그의 어드레스는 MMU(28)에 의해 구현되는 매핑 방식(예를 들어, 태그 위치 어드레스들에 대한 데이터 위치 어드레스들의 하드와이어드 매핑, 또는 소프트웨어에서 프로그래밍가능한 태그 매핑 테이블에 기반한 프로그래밍가능한 매핑)에 따라 판독/기록되는 데이터의 타겟 어드레스로부터 MMU(28)에 의해 도출될 수 있고, 태그 액세스에 필요한 어드레스가 페이지 테이블 정보에 기반하여 read-as-X 페이지로의 매핑으로서 결정되는 경우, MMU(28)는 캐시(30, 32) 또는 메모리(34)에 액세스하지 않고 특정된 값 X를 생성할 수 있고, 요청된 배정 태그의 특정된 값 X를 태그 검사 회로(250)에 반환할 수 있으며, 이는 이어서 특정된 값 X를 대응하는 어드레스의 어드레스 태그(246)와 비교하여, 임의의 태그 검사 에러가 발생한 것인지를 결정할 수 있다. read-as-X 속성은 또한 데이터 액세스에 대해 발생할 수 있으므로, 태그 액세스들에 사용하는 것만으로 제한되지 않는다.

메모리 태깅을 지원하기 위해, 판독 액세스가 캐시(30, 32) 또는 메모리(34)로부터 판독된 값을 부분적으로 반환해야 하고, 다른 부분에 대해 0과 같은 read-as-X 속성으로부터 암시된 암시적으로 정의된 특정 값 X를 반환해야 한다는 것을 나타내는 read-as-X 속성의 변형을 제공하는 것이 유용할 수 있다. 이는 read-as-X 속성과 연관된 값 X를 가질 때 배정 태그 자체가 물리적 메모리에 저장될 필요가 없이, 메모리 영역에 대한 배정 태그가 판독 액세스에서 대응하는 데이터와 함께 반환되는 것을 허용하는 데 유용할 수 있다. 따라서, read-as-X는 X가 단일(가능하게는 캐시 라인 크기) 판독 액세스에서 일부 비트들을 대체하는 것을 허용할 수 있다. 예를 들어, 캐시 라인은 (명목상으로) 다수의 데이터 바이트들 및 단일 MTE-형 "태그" 바이트를 저장하는 것으로 간주될 수 있지만, 태그 바이트는 read-as-X 메모리 속성을 사용하여 암시될 수 있는 경우 물리적으로 저장될 필요가 없을 수 있다. 예를 들어, read-as-X 속성을 갖는 4 KB 페이지에 대응하는 물리적 메모리는 128-비트 데이터 값들의 256 세트들을 물리적으로 저장할 수 있지만, 대응하는 가상 페이지는 실제로 132-비트 값들의 256 세트들에 매핑되는 것으로 고려될 수 있으며, 각각의 132-비트 값은 정상적으로 액세스된 데이터의 128 비트들 및 값 X가 read-as-X 속성에 기반하여 암시적으로 결정될 때 반환되는 4-비트 태그를 포함한다. 따라서, 해당 페이지 내의 어드레스로 판독되는 캐시-라인 크기는 물리적으로 저장된 데이터의 128 비트들만이 아니라 태그를 포함하는 132-비트 값을 반환할 것이다. 이는 가상 메모리의 영역에 대한 태그들이 물리적 스토리지를 커밋하

지 않고 정의될 수 있게 한다.

read-as-X 특성의 다수의 변형들이 위에 설명되었으며, 임의의 특정 시스템 구현이 이들 변형들 중 임의의 하나 이상을 지원할 수 있음이 이해될 것이다.

도 10은 사용될 수 있는 시뮬레이터 구현을 예시한다. 앞서 설명된 실시예들이 관련되는 기법들을 지원하는 특정한 프로세싱 하드웨어를 동작시키기 위한 장치들 및 방법들의 관점에서 본 발명을 구현하지만, 컴퓨터 프로그램의 사용을 통해 구현되는 본원에서 설명되는 실시예들에 따른 명령 실행 환경을 제공하는 것이 또한 가능하다. 이러한 컴퓨터 프로그램들은, 하드웨어 아키텍처의 소프트웨어 기반 구현을 제공하는 한, 시뮬레이터들로서 종종 지칭된다. 다양한 시뮬레이터 컴퓨터 프로그램들은 에뮬레이터들, 가상 기계들, 모델들, 및 동적 바이너리 변환기들을 비롯한, 바이너리 변환기들을 포함한다. 전형적으로, 시뮬레이터 구현은 호스트 프로세서(330) 상에서 실행될 수 있으며, 선택적으로 호스트 운영 체제(320)를 실행하여, 시뮬레이터 프로그램(310)을 지원할 수 있다. 일부 배열들에서, 하드웨어와 제공된 명령 실행 환경, 및/또는 동일한 호스트 프로세서 상에서 제공되는 다수의 별개의 명령 실행 환경들 사이에서 다수의 시뮬레이션 층들이 있을 수 있다. 역사적으로, 합리적인 속도로 실행되는 시뮬레이터 구현들을 제공하기 위해 강력한 프로세서들이 필요하였지만, 그러한 접근법은 소정 상황들에서, 이를테면 호환성 또는 재사용 이유 때문에 다른 프로세서에 기본적인 코드를 실행하려는 요구가 있을 때, 정당화될 수 있다. 예를 들어, 시뮬레이터 구현은 호스트 프로세서 하드웨어에 의해 지원되지 않는 추가적인 기능을 갖는 명령 실행 환경을 제공하거나, 상이한 하드웨어 아키텍처와 전형적으로 연관되는 명령 실행 환경을 제공할 수 있다. 시뮬레이션의 개요는 ["Some Efficient Architecture Simulation Techniques", Robert Bedichek, Winter 1990 USENIX Conference, Pages 53 - 63]에서 주어진다.

실시예들이 특정한 하드웨어 구성들 또는 특징들을 참조하여 앞서 설명된 정도까지, 시뮬레이션된 실시예에서, 동등한 기능성은 적절한 소프트웨어 구성들 또는 특징들에 의해 제공될 수 있다. 예를 들어, 특정한 회로는 시뮬레이션된 실시예에서 컴퓨터 프로그램 로직으로서 구현될 수 있다. 유사하게, 레지스터 또는 캐시와 같은 메모리 하드웨어는 호스트 프로세서(330)의 호스트 스토리지(예를 들어, 메모리 또는 레지스터들)에 저장된 소프트웨어 데이터 구조로서 시뮬레이션된 실시예에서 구현될 수 있다. 앞서 설명된 실시예들에서 참조되는 하드웨어 요소들 중 하나 이상이 호스트 하드웨어(예를 들면, 호스트 프로세서(330))에서 존재하는 배열들에서, 일부 시뮬레이션된 실시예들은, 적절한 경우, 호스트 하드웨어를 이용할 수 있다.

시뮬레이터 프로그램(310)은 컴퓨터 판독가능 저장 매체(이것은 비일시적 매체일 수 있음) 상에 저장될 수 있고, 타겟 코드(300)(이것은 애플리케이션들, 운영 체제들 및 하이퍼바이저를 포함할 수 있음)에 프로그램 인터페이스(명령 실행 환경)를 제공할 수 있는데, 이것은 시뮬레이터 프로그램(310)에 의해 모델링되는 하드웨어 아키텍처의 인터페이스와 동일하다. 따라서, 타겟 코드(300)의 프로그램 명령들은 시뮬레이터 프로그램(310)을 사용하는 명령 실행 환경 내로부터 실행될 수 있고, 따라서 위에서 논의된 장치(2)(예를 들어 위에서 논의된 read-as-X 속성의 사용을 지원하는 메모리 관리 회로(28))의 하드웨어 특징들을 실제로 갖지 않는 호스트 컴퓨터(330)는 그러한 특징을 에뮬레이팅할 수 있다.

따라서, 시뮬레이터 프로그램(310)은 타겟 코드(300)의 명령들을 디코딩하고, 그들을 호스트 장치(330)의 고유한 명령 세트 내의 명령들의 대응하는 세트들에 매핑하는 명령 디코딩 프로그램 로직(312)을 가질 수 있다. 레지스터 에뮬레이팅 프로그램 로직(314)은, 이를테면 호스트 장치(330)의 레지스터들 또는 메모리의 데이터에 액세스하는 것에 의해, 타겟 코드에 의해 요청되는 레지스터 액세스들을 호스트 장치(330)의 호스트 하드웨어 상에서 유지되는 대응하는 데이터 구조에 액세스하여 매핑한다. 메모리 관리 프로그램 로직(316)은 상기의 하드웨어 구현 실시예에 설명된 바와 같이 MMU(28)에 대응하는 방식으로, 타겟 코드(300)에 의한 시뮬레이션된 어드레스 공간에 대한 액세스를 시뮬레이션하기 위해 어드레스 변환, 페이지 테이블 워크들 및 액세스 허가 검사를 구현한다. 메모리 어드레스 공간 시뮬레이팅 프로그램 로직(318)은 타겟 프로그램 코드(300)의 소프트웨어에 의해 유지되는 페이지 테이블 정보를 사용하는 어드레스 변환에 기반하여 메모리 관리 프로그램 로직(316)에 의해 획득된 시뮬레이션된 물리적 어드레스들을 호스트 프로세서(330)의 호스트 메모리를 액세스하는데 사용되는 호스트 가상 어드레스들에 매핑하도록 제공된다. 이들 호스트 가상 어드레스들은 그들 자체가 호스트에 의해 지원되는 표준 어드레스 변환 메커니즘을 사용하여 호스트 물리적 어드레스들로 변환될 수 있다(호스트 물리적 어드레스들로의 호스트 가상 어드레스의 변환은 시뮬레이터 프로그램(310)에 의해 제어되는 것의 범위 밖에 있음).

메모리 관리 프로그램 로직(316)이, 시뮬레이션된 어드레스 공간의 주어진 판독 타겟 블록에 대한 타겟 프로그램 코드(300)에 의한 판독 액세스의 경우, 주어진 판독 타겟 블록이 read-as-X 속성을 갖는 것으로서 타겟 프로

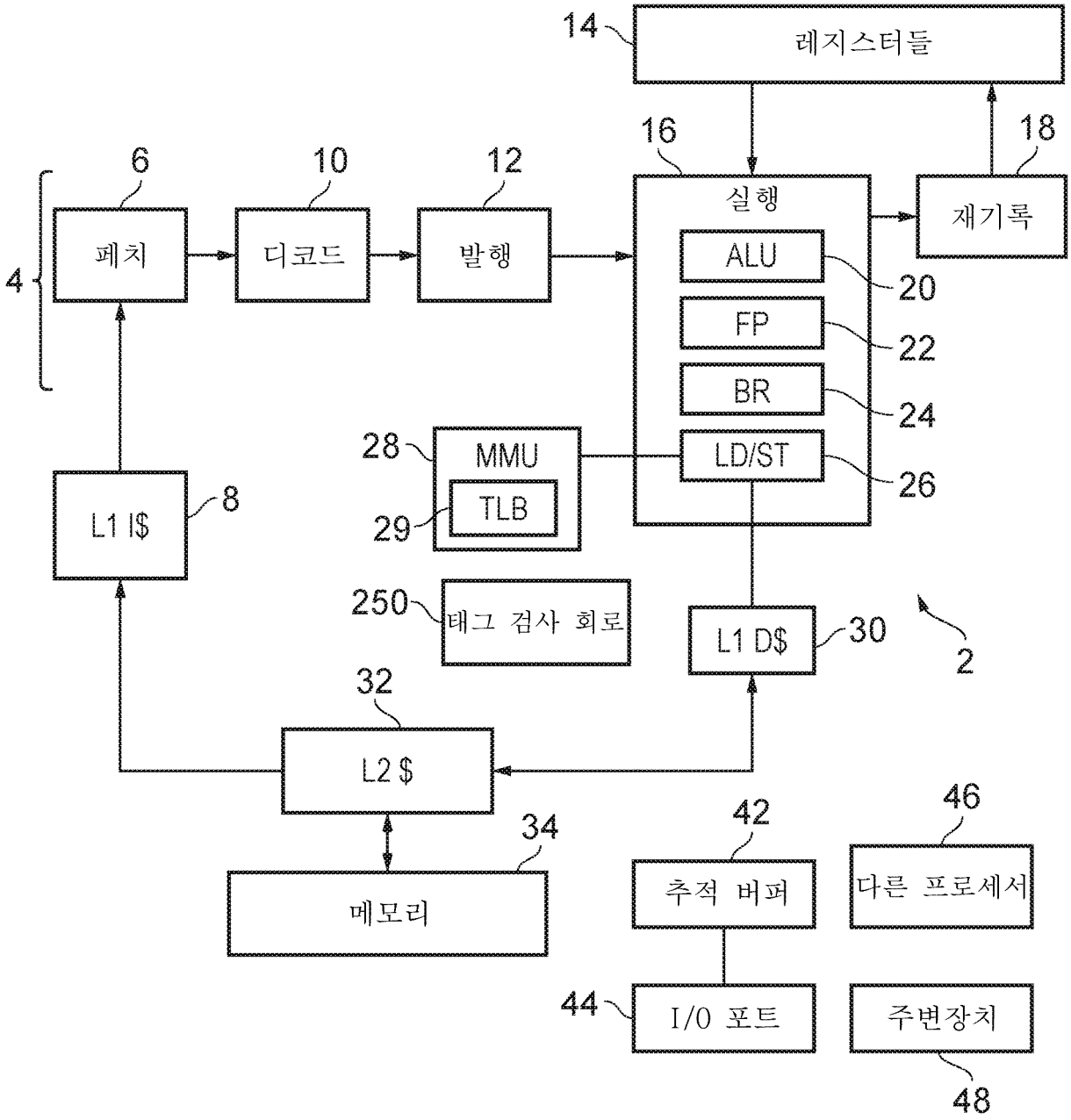
그램 코드(300)에 의해 설정된 페이지 테이블 정보(호스트 어드레스 변환을 위해 호스트에 의해 사용되는 페이지 테이블 정보가 아님)를 사용하여 식별된 페이지에 대응한다는 것을 식별할 때, 메모리 관리 프로그램 로직(316)은, 판독 타겟 블록의 시뮬레이션된 가상 어드레스를 시뮬레이션된 물리적 어드레스로 변환하거나 또는 시뮬레이션된 물리적 어드레스를 호스트 프로세서(330)의 호스트 스토리지 상에 매핑하기 위해 메모리 어드레스 공간 시뮬레이션 로직(318)을 호출할 필요없이, 특정된 값 X가 타겟 프로그램 코드(300)에 반환되도록 제어한다. 대신, 값 X는 read-as-X 속성의 식별에 기반하여 시뮬레이터 프로그램(310) 자체의 프로그램 코드에 의해 생성될 수 있고, 타겟 프로그램 코드(300)에 의한 후속 프로세싱에 대해 이용가능하게 될 수 있다. 이는 타겟 프로그램 코드(300)가 도 1의 하드웨어 장치(2) 상에서 실행되었다면 타겟 프로그램 코드(300)가 관찰할 것으로 예상되 것이었을 거동을 에뮬레이션한다.

본 출원에서 "...하도록 구성된"이라는 단어는 장치의 요소가 정의된 동작을 수행할 수 있는 구성을 갖는다는 것을 의미하는 데 사용된다. 이러한 맥락에서, "구성"은 하드웨어 또는 소프트웨어의 배열 또는 상호연결 방식을 의미한다. 예를 들어, 장치는 정의된 동작을 제공하는 전용 하드웨어를 가질 수 있거나, 프로세서 또는 다른 프로세싱 디바이스는 기능을 수행하도록 프로그래밍될 수 있다. "구성된"은 정의된 동작을 제공하기 위해 장치 요소를 어떤 방식으로든 변경해야 한다는 것을 의미하지 않는다.

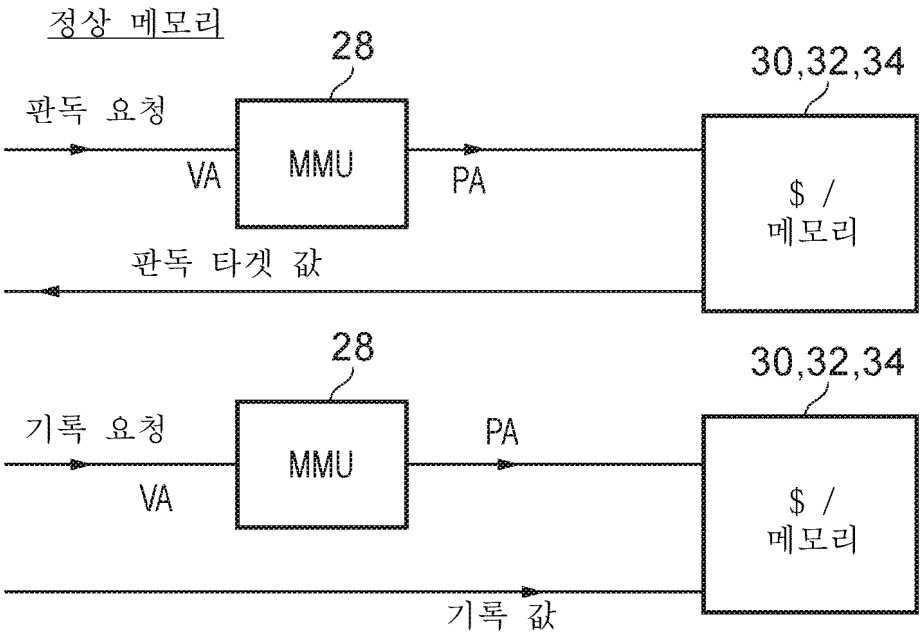
본 발명의 예시적인 실시예들이 첨부의 도면들을 참조하여 본원에서 상세히 설명되었지만, 본 발명은 그들 정확한 실시예들로 제한되지 않는다는 것, 및 첨부된 청구범위에 의해 정의되는 바와 같은 본 발명의 범위로부터 벗어나지 않으면서 기술 분야에서 숙련된 자에 의해 다양한 변경들 및 수정들이 그 안에서 실행될 수 있다는 것이 이해되어야 한다.

도면

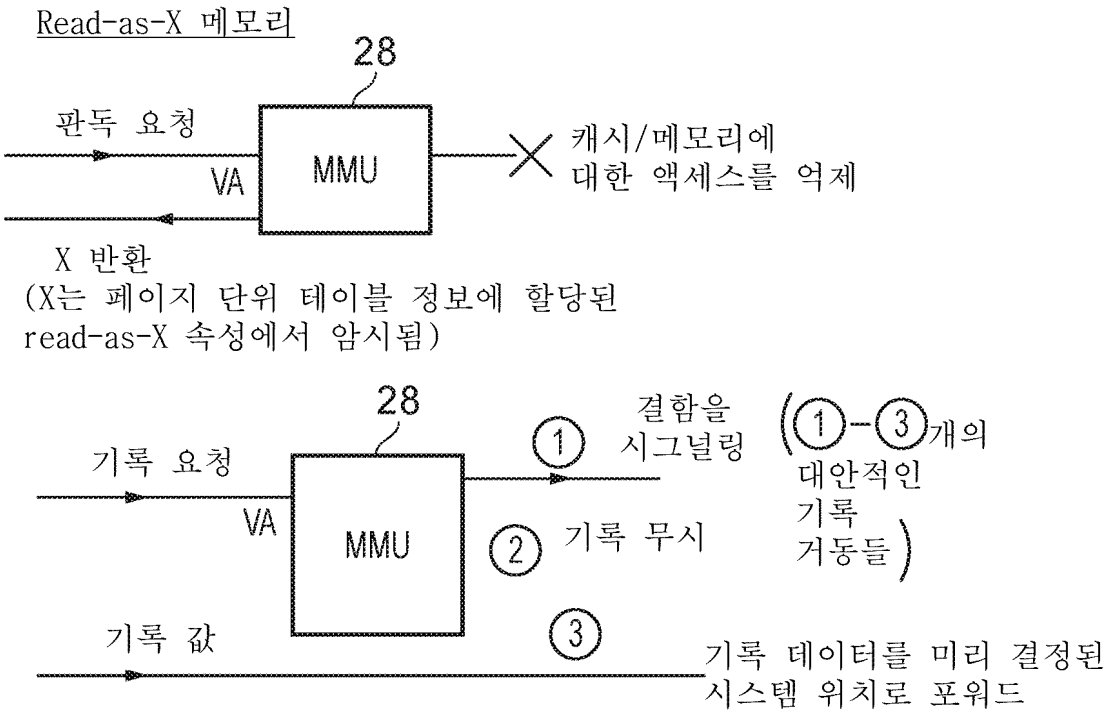
도면1


14
레지스터들
16
18
6
10
12
실행
재기록
4
페치
디코드
발행
ALU
20
FP
22
BR
24
28
MMU
LD/ST
26
29
TLB
L1 I$
8
250
태그 검사 회로
L1 D$
30
2
32
L2 $
34
42
46
메모리
추적 버퍼
다른 프로세서
44
I/O 포트
주변장치
48

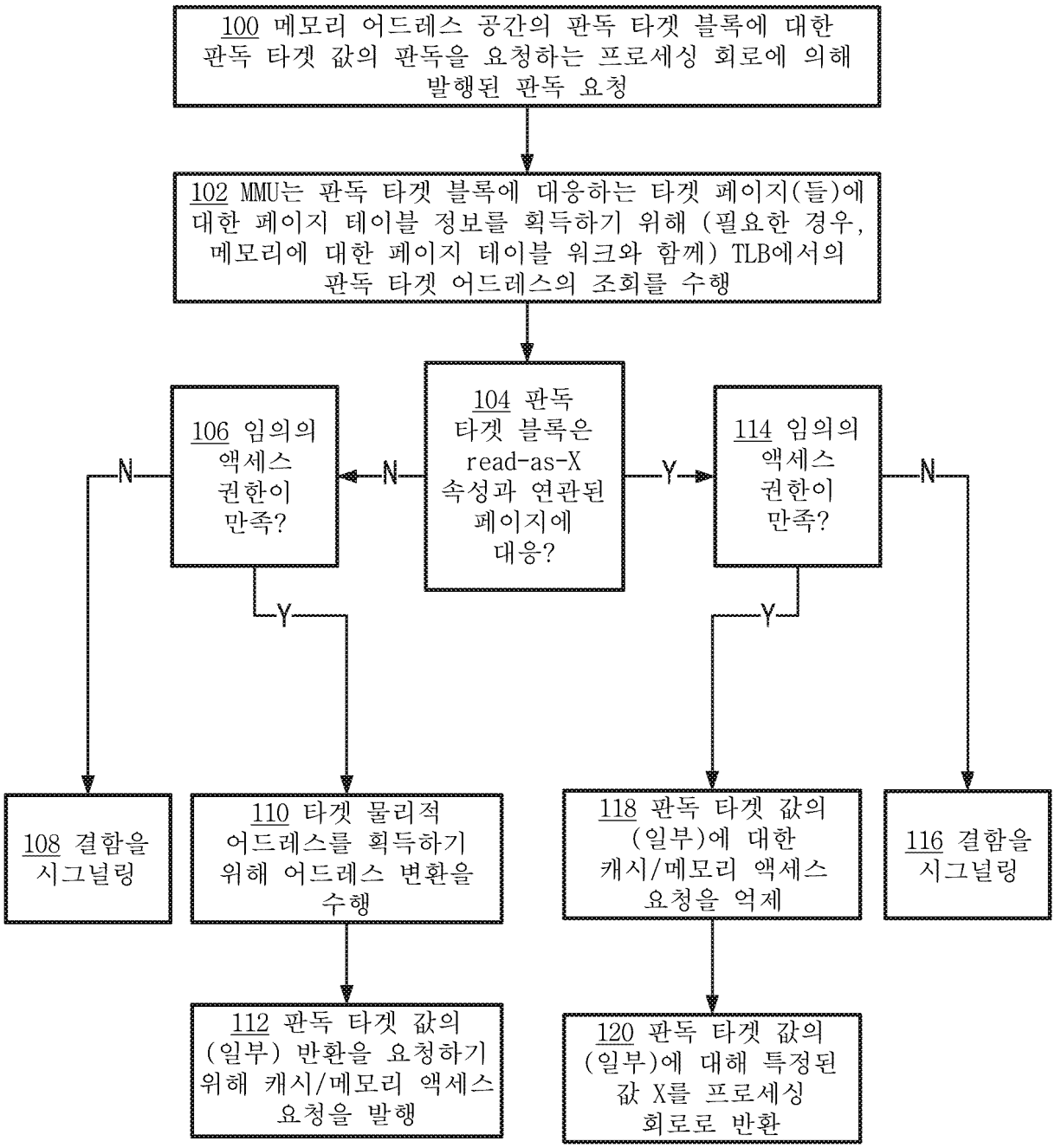
도면2


정상 메모리
28
30,32,34
판독 요청
VA
MMU
PA
$ /
메모리
판독 타겟 값
28
30,32,34
기록 요청
VA
MMU
PA
$ /
메모리
기록 값

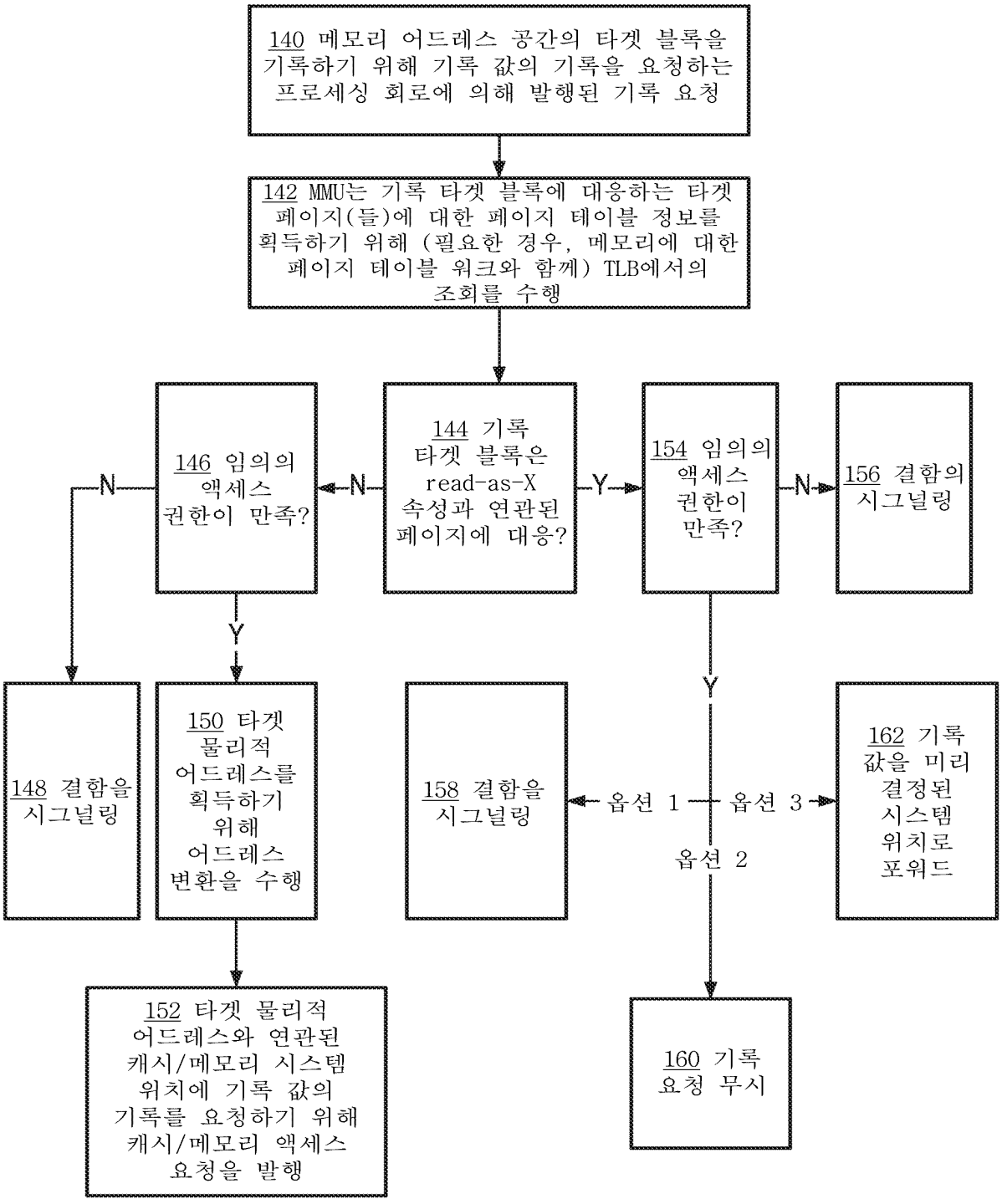
도면3


Read-as-X 메모리
28
판독 요청
VA
MMU
캐시/메모리에
대한 액세스를 억제
X 반환
(X는 페이지 단위 테이블 정보에 할당된
read-as-X 속성에서 암시됨)
28
결함을
시그널링
①-③개의
대안적인
기록
거동들
기록 요청
VA
MMU
② 기록 무시
③
기록 값
기록 데이터를 미리 결정된
시스템 위치로 포워드

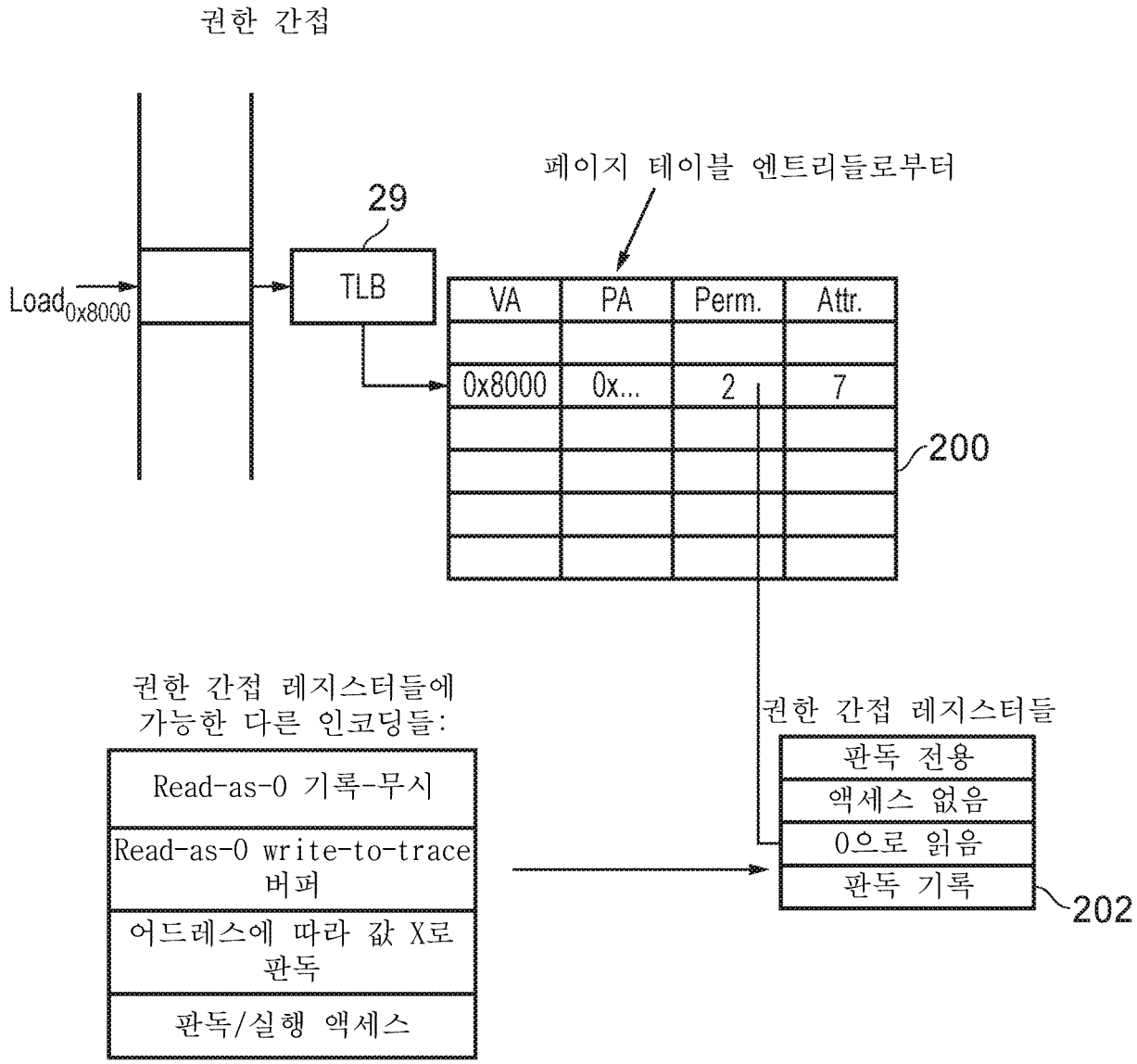
도면4


판독 요청
100 메모리 어드레스 공간의 판독 타겟 블록에 대한 판독 타겟 값의 판독을 요청하는 프로세싱 회로에 의해 발행된 판독 요청
102 MMU는 판독 타겟 블록에 대응하는 타겟 페이지(들)에 대한 페이지 테이블 정보를 획득하기 위해 (필요한 경우, 메모리에 대한 페이지 테이블 워크와 함께) TLB에서의 판독 타겟 어드레스의 조회를 수행
104 판독 타겟 블록은 read-as-X 속성과 연관된 페이지에 대응?
106 임의의 액세스 권한이 만족?
114 임의의 액세스 권한이 만족?
N
N
Y
N
Y
Y
108 결함을 시그널링
110 타겟 물리적 어드레스를 획득하기 위해 어드레스 변환을 수행
112 판독 타겟 값의 (일부) 반환을 요청하기 위해 캐시/메모리 액세스 요청을 발행
118 판독 타겟 값의 (일부)에 대한 캐시/메모리 액세스 요청을 억제
120 판독 타겟 값의 (일부)에 대해 특정된 값 X를 프로세싱 회로로 반환
116 결함을 시그널링

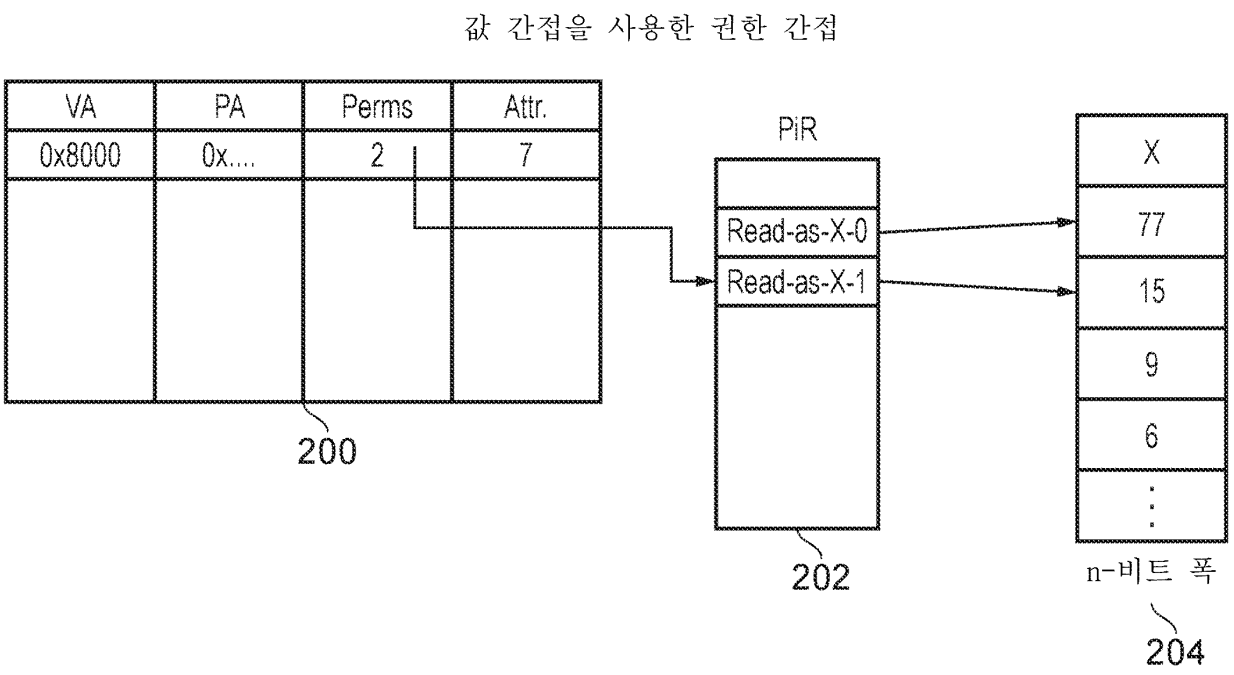
도면5


기록 요청
140 메모리 어드레스 공간의 타겟 블록을 기록하기 위해 기록 값의 기록을 요청하는 프로세싱 회로에 의해 발행된 기록 요청
142 MMU는 기록 타겟 블록에 대응하는 타겟 페이지(들)에 대한 페이지 테이블 정보를 획득하기 위해 (필요한 경우, 메모리에 대한 페이지 테이블 워크와 함께) TLB에서의 조회를 수행
144 기록 타겟 블록은 read-as-X 속성과 연관된 페이지에 대응?
146 임의의 액세스 권한이 만족?
154 임의의 액세스 권한이 만족?
156 결함의 시그널링
148 결함을 시그널링
150 타겟 물리적 어드레스를 획득하기 위해 어드레스 변환을 수행
158 결함을 시그널링
162 기록 값을 미리 결정된 시스템 위치로 포워드
152 타겟 물리적 어드레스와 연관된 캐시/메모리 시스템 위치에 기록 값의 기록를 요청하기 위해 캐시/메모리 액세스 요청을 발행
160 기록 요청 무시
N
N
Y
N
Y
Y
옵션 1
옵션 3
옵션 2

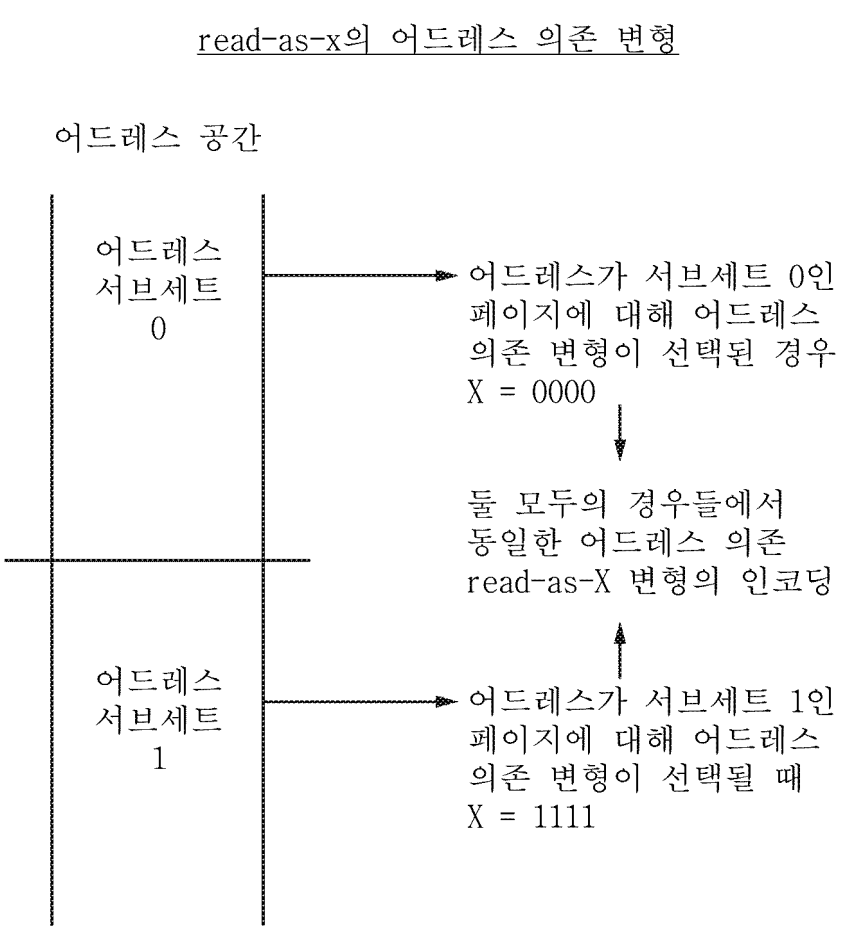
도면6


권한 간접
Load0x8000
29
TLB
페이지 테이블 엔트리들로부터
VA
PA
Perm.
Attr.
0x8000
0x...
2
7
200
권한 간접 레지스터들에 가능한 다른 인코딩들:
Read-as-0 기록-무시
Read-as-0 write-to-trace 버퍼
어드레스에 따라 값 X로 판독
판독/실행 액세스
권한 간접 레지스터들
판독 전용
액세스 없음
0으로 읽음
판독 기록
202

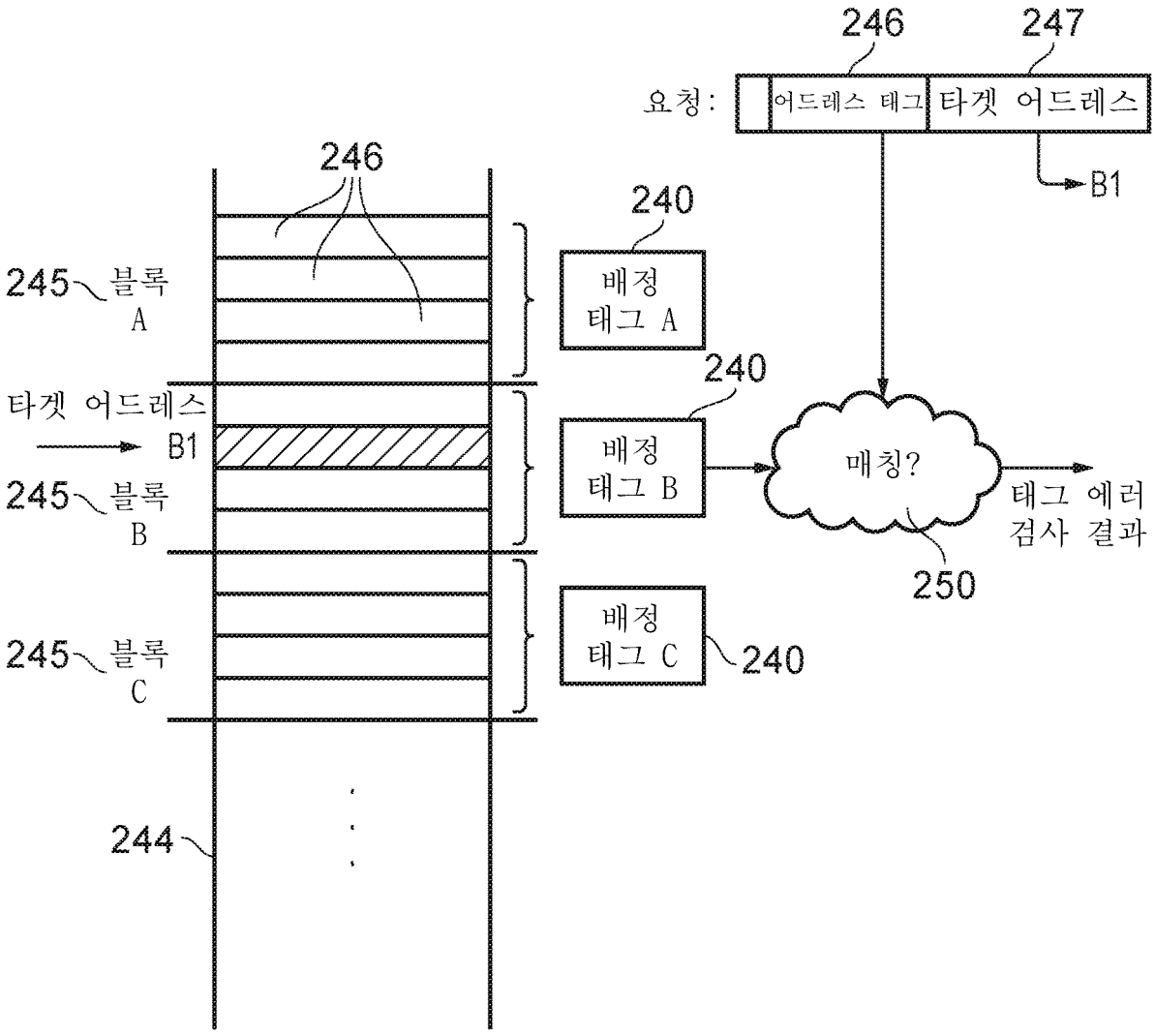
도면7


값 간접을 사용한 권한 간접
VA
PA
Perms
Attr.
0x8000
0x....
2
7
200
PiR
Read-as-X-0
Read-as-X-1
202
X
77
15
9
6
n-비트 폭
204

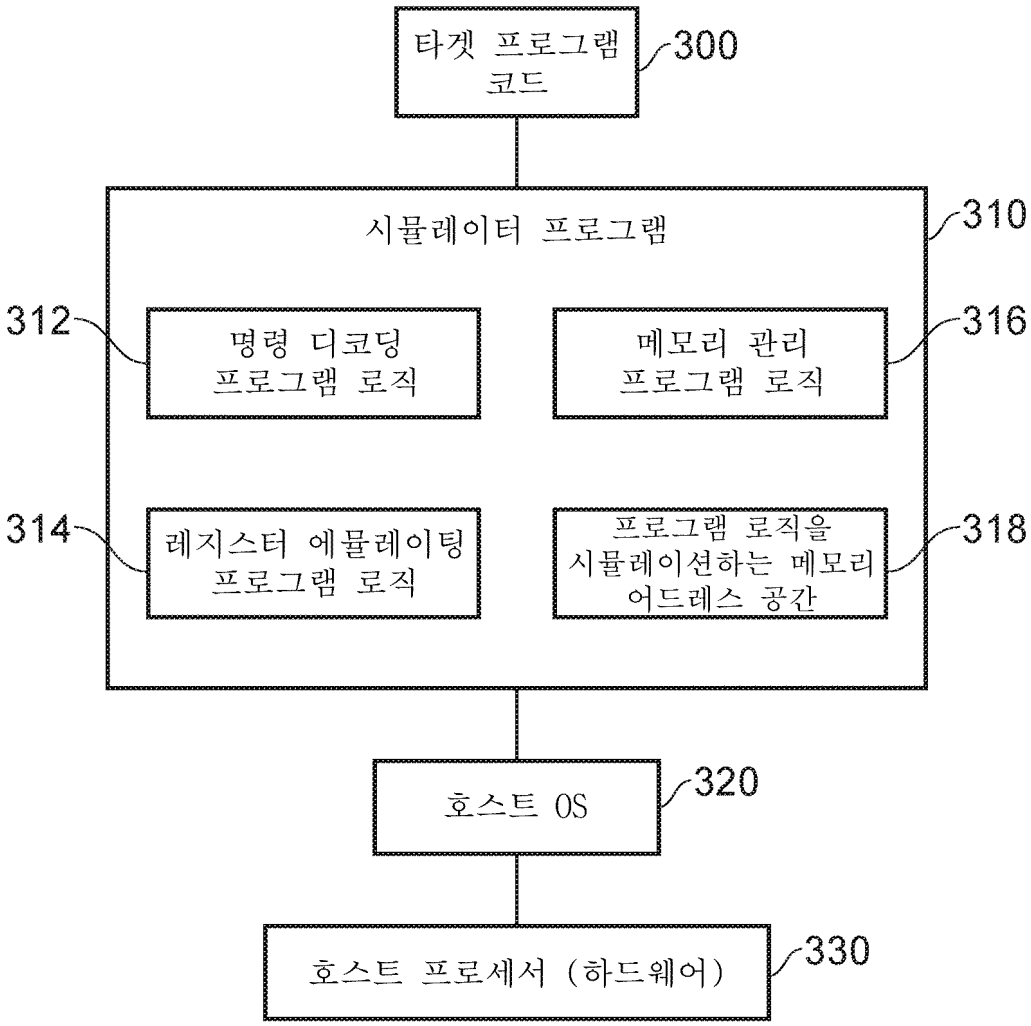
도면8


read-as-x의 어드레스 의존 변형
어드레스 공간
어드레스 서브세트 0
어드레스가 서브세트 0인 페이지에 대해 어드레스 의존 변형이 선택된 경우 X = 0000
둘 모두의 경우들에서 동일한 어드레스 의존 read-as-X 변형의 인코딩
어드레스 서브세트 1
어드레스가 서브세트 1인 페이지에 대해 어드레스 의존 변형이 선택될 때 X = 1111

도면9


246
247
요청:
어드레스 태그
타겟 어드레스
B1
246
240
245
블록
A
배정
태그 A
타겟 어드레스
B1
240
245
블록
B
배정
태그 B
매칭?
태그 에러
검사 결과
250
245
블록
C
배정
태그 C
240
244

도면10


타겟 프로그램 코드
300
시뮬레이터 프로그램
310
312
명령 디코딩 프로그램 로직
316
메모리 관리 프로그램 로직
314
레지스터 에뮬레이팅 프로그램 로직
318
프로그램 로직을 시뮬레이션하는 메모리 어드레스 공간
호스트 OS
320
호스트 프로세서 (하드웨어)
330