(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2017년10월18일
(11) 등록번호 10-1787819
(24) 등록일자 2017년10월12일

(51) 국제특허분류(Int. Cl.)
G06F 9/30 (2017.01)
(52) CPC특허분류
G06F 9/30036 (2013.01)
G06F 9/30105 (2013.01)
(21) 출원번호 10-2015-0028036
(22) 출원일자 2015년02월27일
심사청구일자 2015년02월27일
(65) 공개번호 10-2015-0112781
(43) 공개일자 2015년10월07일
(30) 우선권주장
14/229,811 2014년03월28일 미국(US)
(56) 선행기술조사문헌
KR1020110101647 A*
KR1020130137697 A*
US20130212354 A1*
US05907842 A*
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
인텔 코포레이션
미합중국 캘리포니아 95054 산타클라라 미션 칼리지 블러바드 2200
(72) 발명자
구에론, 샤이
이스라엘 에이치에이 32714 하이파 아담 하코엔 에스티 18에이
크라스노브, 블라드
이스라엘 제트 36680 네세르 아브라함 페레즈 2
(74) 대리인
양영준, 백만기

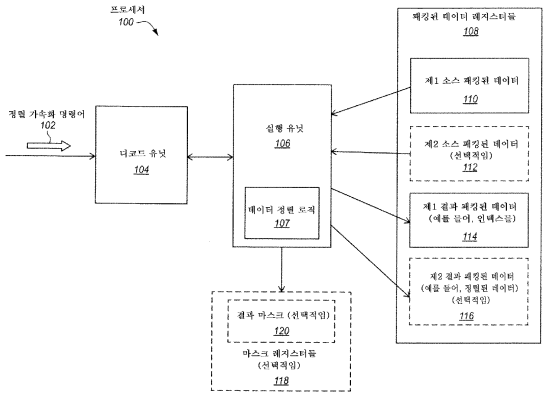
전체 청구항 수 : 총 21 항 심사관 : 김경완

(54) 발명의 명칭 **정렬 가속화 프로세서들, 방법들, 시스템들 및 명령어들**

(57) 요 약

일 양상의 프로세서는 패킹된 데이터 레지스터들, 및 명령어를 디코드하는 디코드 유닛을 포함한다. 명령어는, 적어도 4개의 데이터 엘리먼트들을 포함하는 제1 소스 패킹된 데이터를 나타내고, 적어도 4개의 데이터 엘리먼트들을 포함하는 제2 소스 패킹된 데이터를 나타내며, 목적지 스토리지 위치를 나타낼 수 있다. 실행 유닛은 패킹된 데이터 레지스터들 및 디코드 유닛과 연결된다. 실행 유닛은, 명령어에 응답하여, 결과 패킹된 데이터를 목적지 스토리지 위치에 저장한다. 결과 패킹된 데이터는 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서의 대응 데이터 엘리먼트 위치들을 식별할 수 있는 적어도 4개의 인덱스들을 포함할 수 있다. 인덱스들은 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서의 대응 데이터 엘리먼트들의 정렬된 순서를 나타내는 결과 패킹된 데이터에서의 위치들에 저장될 수 있다.

대 표 도



(52) CPC특허분류
G06F 9/30196 (2013.01)

명 세 서

청구범위

청구항 1

삭제

청구항 2

프로세서로서,

복수의 패킹된 데이터 레지스터들;

명령어를 디코드하는 디코드 유닛- 상기 명령어는 적어도 4개의 데이터 엘리먼트들의 제1 세트를 포함하는 제1 소스 패킹된 데이터를 나타내고, 적어도 4개의 데이터 엘리먼트들의 제2 세트를 포함하는 제2 소스 패킹된 데이터를 나타내며, 목적지 스토리지 위치를 나타냄 -; 및

상기 패킹된 데이터 레지스터들 및 상기 디코드 유닛과 연결되고, 상기 명령어에 응답하여, 상기 목적지 스토리지 위치에 결과 패킹된 데이터를 저장하는 실행 유닛

을 포함하고,

상기 결과 패킹된 데이터는 적어도 4개의 인덱스들을 포함하고,

상기 인덱스들은 상기 제1 소스 패킹된 데이터 및 상기 제2 소스 패킹된 데이터에서의 대응하는 데이터 엘리먼트 위치들을 식별하며,

상기 인덱스들은 상기 제1 소스 패킹된 데이터 및 상기 제2 소스 패킹된 데이터에서의 대응 데이터 엘리먼트들의 정렬된 순서를 나타내는 상기 결과 패킹된 데이터에서의 위치들에 저장되고,

상기 실행 유닛은, 상기 인덱스들 각각이 상기 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터 각각에서의 대응 데이터 엘리먼트 위치를 식별하는 상기 결과 패킹된 데이터를 저장하고, 상기 실행 유닛은, 상기 명령어에 응답하여, 적어도 4개의 마스크 엘리먼트들을 갖는 결과 마스크를 저장하고, 각각의 마스크 엘리먼트는 상기 인덱스들 중 상이한 것에 대응하며, 각각의 마스크 엘리먼트는, 대응 인덱스에 대한 데이터 엘리먼트 위치가 상기 제1 소스 패킹된 데이터 또는 상기 제2 소스 패킹된 데이터에 존재하는지를 나타내는, 프로세서.

청구항 3

제2항에 있어서,

상기 결과 마스크를 저장하는 마스크 레지스터를 더 포함하고, 상기 명령어는, 패킹된 데이터 연산을 서술(predicate)하는 서술 피연산자로서 상기 결과 마스크를 나타낼 수 있는 제2 명령어를 포함하는 명령어 세트에 포함되는 프로세서.

청구항 4

삭제

청구항 5

제2항에 있어서,

상기 실행 유닛은, 상기 명령어에 응답하여, 상기 명령어에 의해 나타나는 제2 목적지 스토리지 위치에 제2 결과 패킹된 데이터를 저장하고, 상기 제2 결과 패킹된 데이터는, 상기 정렬된 순서를 반영하는 상기 제2 결과 패킹된 데이터의 위치들에 저장된 상기 인덱스들에 대응하는 상기 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터로부터의 상기 데이터 엘리먼트들을 포함하는, 프로세서.

청구항 6

프로세서로서,

복수의 패킹된 데이터 레지스터들;

명령어를 디코드하는 디코드 유닛 - 상기 명령어는 적어도 4개의 데이터 엘리먼트들의 제1 세트를 포함하는 제1 소스 패킹된 데이터를 나타내고, 적어도 4개의 데이터 엘리먼트들의 제2 세트를 포함하는 제2 소스 패킹된 데이터를 나타내며, 목적지 스토리지 위치를 나타냄 -; 및

상기 패킹된 데이터 레지스터들 및 상기 디코드 유닛과 연결되고, 상기 명령어에 응답하여, 상기 목적지 스토리지 위치에 결과 패킹된 데이터를 저장하는 실행 유닛

을 포함하고,

상기 결과 패킹된 데이터는 적어도 4개의 인덱스들을 포함하고,

상기 인덱스들은 상기 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서의 대응하는 데이터 엘리먼트 위치들을 식별하며,

상기 인덱스들은 상기 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서의 대응 데이터 엘리먼트들의 정렬된 순서를 나타내는 상기 결과 패킹된 데이터에서의 위치들에 저장되고,

상기 디코드 유닛은, 상기 명령어에 대해 정렬된 순서로 존재할 것으로 추정되는 상기 적어도 4개의 데이터 엘리먼트들을 갖는 상기 제1 소스 패킹된 데이터를 나타내고, 상기 명령어에 대해 정렬된 순서로 존재할 것으로 추정되는 상기 적어도 4개의 데이터 엘리먼트들을 갖는 상기 제2 소스 패킹된 데이터를 나타내는 상기 명령어를 디코드하는, 프로세서.

청구항 7

프로세서로서,

복수의 패킹된 데이터 레지스터들;

명령어를 디코드하는 디코드 유닛 - 상기 명령어는 적어도 4개의 데이터 엘리먼트들의 제1 세트를 포함하는 제1 소스 패킹된 데이터를 나타내고, 적어도 4개의 데이터 엘리먼트들의 제2 세트를 포함하는 제2 소스 패킹된 데이터를 나타내며, 목적지 스토리지 위치를 나타냄 -; 및

상기 패킹된 데이터 레지스터들 및 상기 디코드 유닛과 연결되고, 상기 명령어에 응답하여, 상기 목적지 스토리지 위치에 결과 패킹된 데이터를 저장하는 실행 유닛

을 포함하고,

상기 결과 패킹된 데이터는 적어도 4개의 인덱스들을 포함하고,

상기 인덱스들은 상기 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서의 대응하는 데이터 엘리먼트 위치들을 식별하며,

상기 인덱스들은 상기 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서의 대응 데이터 엘리먼트들의 정렬된 순서를 나타내는 상기 결과 패킹된 데이터에서의 위치들에 저장되고,

상기 디코드 유닛은, 상기 명령어에 대해 정렬된 순서로 존재할 것으로 추정되지 않는 상기 적어도 4개의 데이터 엘리먼트들을 갖는 상기 제1 소스 패킹된 데이터를 나타내고, 상기 명령어에 대해 정렬된 순서로 존재할 것으로 추정되지 않는 상기 적어도 4개의 데이터 엘리먼트들을 갖는 상기 제2 소스 패킹된 데이터를 나타내는 상기 명령어를 디코드하는, 프로세서.

청구항 8

프로세서로서,

복수의 패킹된 데이터 레지스터들;

명령어를 디코드하는 디코드 유닛 - 상기 명령어는 적어도 4개의 데이터 엘리먼트들의 제1 세트를 포함하는 제1 소스 패킹된 데이터를 나타내고, 적어도 4개의 데이터 엘리먼트들의 제2 세트를 포함하는 제2 소스 패킹된 데이터를 나타내며, 목적지 스토리지 위치를 나타냄 -; 및

상기 패킹된 데이터 레지스터들 및 상기 디코드 유닛과 연결되고, 상기 명령어에 응답하여, 상기 목적지 스토리지 위치에 결과 패킹된 데이터를 저장하는 실행 유닛

을 포함하고,

상기 결과 패킹된 데이터는 적어도 4개의 인덱스들을 포함하고,

상기 인덱스들은 상기 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서의 대응하는 데이터 엘리먼트 위치들을 식별하며,

상기 인덱스들은 상기 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서의 대응 데이터 엘리먼트들의 정렬된 순서를 나타내는 상기 결과 패킹된 데이터에서의 위치들에 저장되고,

상기 실행 유닛은, 상기 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터의 모든 데이터 엘리먼트들 중 최소 1/2을 포함하는 대응 데이터 엘리먼트들의 정렬된 순서를 나타내는 상기 위치들에 상기 인덱스들이 저장되는 상기 결과 패킹된 데이터를 저장하는, 프로세서.

청구항 9

프로세서로서,

복수의 패킹된 데이터 레지스터들;

명령어를 디코드하는 디코드 유닛 - 상기 명령어는 적어도 4개의 데이터 엘리먼트들의 제1 세트를 포함하는 제1 소스 패킹된 데이터를 나타내고, 적어도 4개의 데이터 엘리먼트들의 제2 세트를 포함하는 제2 소스 패킹된 데이터를 나타내며, 목적지 스토리지 위치를 나타냄 -; 및

상기 패킹된 데이터 레지스터들 및 상기 디코드 유닛과 연결되고, 상기 명령어에 응답하여, 상기 목적지 스토리지 위치에 결과 패킹된 데이터를 저장하는 실행 유닛

을 포함하고,

상기 결과 패킹된 데이터는 적어도 4개의 인덱스들을 포함하고,

상기 인덱스들은 상기 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서의 대응하는 데이터 엘리먼트 위치들을 식별하며,

상기 인덱스들은 상기 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서의 대응 데이터 엘리먼트들의 정렬된 순서를 나타내는 상기 결과 패킹된 데이터에서의 위치들에 저장되고,

상기 실행 유닛은, 상기 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터의 모든 데이터 엘리먼트들 중 최대 1/2을 포함하는 대응 데이터 엘리먼트들의 정렬된 순서를 나타내는 상기 위치들에 상기 인덱스들이 저장되는 상기 결과 패킹된 데이터를 저장하는 프로세서.

청구항 10

프로세서로서,

복수의 패킹된 데이터 레지스터들;

명령어를 디코드하는 디코드 유닛 - 상기 명령어는 적어도 4개의 데이터 엘리먼트들의 제1 세트를 포함하는 제1 소스 패킹된 데이터를 나타내고, 적어도 4개의 데이터 엘리먼트들의 제2 세트를 포함하는 제2 소스 패킹된 데이터를 나타내며, 목적지 스토리지 위치를 나타냄 -; 및

상기 패킹된 데이터 레지스터들 및 상기 디코드 유닛과 연결되고, 상기 명령어에 응답하여, 상기 목적지 스토리지 위치에 결과 패킹된 데이터를 저장하는 실행 유닛

을 포함하고,

상기 결과 패킹된 데이터는 적어도 4개의 인덱스들을 포함하고,

상기 인덱스들은 상기 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서의 대응하는 데이터 엘리먼트 위치들을 식별하며,

상기 인덱스들은 상기 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서의 대응 데이터 엘리먼트들의 정렬된 순서를 나타내는 상기 결과 패킹된 데이터에서의 위치들에 저장되고,

상기 디코드 유닛은, 32 비트 및 64 비트 중 하나를 각각 갖는 적어도 8개의 데이터 엘리먼트들을 포함하는 상기 제1 소스 패킹된 데이터를 나타내는 상기 명령어를 디코드하는, 프로세서.

청구항 11

삭제

청구항 12

프로세서에서의 방법으로서,

명령어를 수신하는 단계- 상기 명령어는 적어도 4개의 데이터 엘리먼트들의 제1 세트를 포함하는 제1 소스 패킹된 데이터를 나타내고, 적어도 4개의 데이터 엘리먼트들의 제2 세트를 포함하는 제2 소스 패킹된 데이터를 나타내고, 목적지 스토리지 위치를 나타냄 -; 및

상기 명령어에 응답하여, 상기 목적지 스토리지 위치에 결과 패킹된 데이터를 저장하는 단계

를 포함하고,

상기 결과 패킹된 데이터는 적어도 4개의 인덱스들을 포함하고,

상기 인덱스들은 상기 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서의 대응 데이터 엘리먼트 위치들을 식별하며,

상기 인덱스들은 상기 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서의 대응 데이터 엘리먼트들의 정렬된 순서를 나타내는 상기 결과 패킹된 데이터에서의 위치들에 저장되고,

상기 수신하는 단계는, 정렬된 순서로 상기 적어도 4개의 데이터 엘리먼트들을 갖는 상기 제1 소스 패킹된 데이터를 나타내는 명령어를 수신하는 단계를 포함하는 방법.

청구항 13

프로세서로서,

복수의 패킹된 데이터 레지스터들;

명령어를 디코드하는 디코드 유닛- 상기 명령어는 정렬된 순서로 존재하지 않는 적어도 4개의 데이터 엘리먼트들을 포함하는 소스 패킹된 데이터를 나타내고, 목적지 스토리지 위치를 나타냄 -; 및

상기 패킹된 데이터 레지스터들 및 상기 디코드 유닛과 연결되고, 상기 명령어에 응답하여, 상기 목적지 스토리지 위치에 결과 패킹된 데이터를 저장하는 실행 유닛

을 포함하고,

상기 결과 패킹된 데이터는 적어도 4개의 인덱스들을 포함하고,

상기 인덱스들은 상기 소스 패킹된 데이터에서의 대응 데이터 엘리먼트들을 식별하며,

상기 인덱스들은 상기 소스 패킹된 데이터에서의 대응 데이터 엘리먼트들의 정렬된 순서를 나타내는 상기 결과 패킹된 데이터에서의 위치들에 저장되는 프로세서.

청구항 14

제13항에 있어서,

상기 실행 유닛은, 상기 명령어에 응답하여, 상기 명령어에 의해 나타나는 제2 목적지 스토리지 위치에 제2 결과 패킹된 데이터를 저장하고, 상기 제2 결과 패킹된 데이터는, 상기 정렬된 순서를 반영하는 상기 제2 결과 패킹된 데이터의 위치들에 저장된 대응 데이터 엘리먼트들을 포함하는 프로세서.

청구항 15

제13항 또는 제14항에 있어서,

상기 결과 패킹된 데이터는 상기 소스 패킹된 데이터에서의 모든 데이터 엘리먼트들에 대응하는 인덱스들을 포함하는 프로세서.

청구항 16

제13항 또는 제14항에 있어서,

상기 디코드 유닛은, 32 비트 및 64 비트 중 하나를 각각 갖는 적어도 8개의 데이터 엘리먼트들을 포함하는 상기 소스 패킹된 데이터를 나타내는 상기 명령어를 디코드하는 프로세서.

청구항 17

프로세서에서의 방법으로서,

명령어를 수신하는 단계- 상기 명령어는 정렬된 순서로 존재하지 않는 적어도 4개의 데이터 엘리먼트들을 포함하는 소스 패킹된 데이터를 나타내고, 목적지 스토리지 위치를 나타냄 -; 및

상기 명령어에 응답하여, 상기 목적지 스토리지 위치에 결과 패킹된 데이터를 저장하는 단계를 포함하고,

상기 결과 패킹된 데이터는 적어도 4개의 인덱스들을 포함하고,

상기 인덱스들은 상기 소스 패킹된 데이터에서의 대응 데이터 엘리먼트들을 식별하며,

상기 인덱스들은 상기 소스 패킹된 데이터에서의 대응 데이터 엘리먼트들의 정렬된 순서를 나타내는 상기 결과 패킹된 데이터에서의 위치들에 저장되는 방법.

청구항 18

제17항에 있어서,

상기 명령어에 의해 나타나는 제2 목적지 스토리지 위치에 제2 결과 패킹된 데이터를 저장하는 단계를 더 포함하고, 상기 제2 결과 패킹된 데이터는, 상기 정렬된 순서를 반영하는 위치들에 저장되는 대응 데이터 엘리먼트들을 포함하는 방법.

청구항 19

명령어들을 처리하는 시스템으로서,

상호접속(interconnect);

상기 상호접속과 연결되고, 명령어를 수신하는 프로세서- 상기 명령어는 적어도 4개의 데이터 엘리먼트들의 제1 세트를 포함하는 제1 소스 패킹된 데이터를 나타내고, 적어도 4개의 데이터 엘리먼트들의 제2 세트를 포함하는 제2 소스 패킹된 데이터를 나타내며, 목적지 레지스터를 나타내고, 상기 프로세서는 상기 명령어에 응답하여 상기 목적지 레지스터에 결과 패킹된 데이터를 저장하고, 상기 결과 패킹된 데이터는 적어도 4개의 인덱스들을 포함하고, 상기 인덱스들은 상기 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서의 대응 데이터 엘리먼트 위치들을 식별하고, 상기 인덱스들은 상기 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서의 대응 데이터 엘리먼트들의 정렬된 순서를 나타내는 상기 결과 패킹된 데이터에서의 위치들에 저장됨 -; 및

상기 상호접속과 연결되고, 상기 결과 패킹된 데이터의 인덱스들을 사용하여 데이터를 정렬하는 알고리즘을 저장하는 DRAM(Dynamic Random Access Memory)

을 포함하는 시스템.

청구항 20

제19항에 있어서,

상기 프로세서는, 상기 인덱스들 각각이 상기 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터 중 하나에서의 대응 싱글 데이터 엘리먼트를 식별하는 상기 결과 패킹된 데이터를 저장하는 시스템.

청구항 21

삭제

청구항 22

프로세서로서,

명령어를 수신하는 수단 - 상기 명령어는 적어도 4개의 데이터 엘리먼트들의 제1 세트를 포함하는 제1 소스 패킹된 데이터를 나타내고, 적어도 4개의 데이터 엘리먼트들의 제2 세트를 포함하는 제2 소스 패킹된 데이터를 나타내고, 목적지 스토리지 위치를 나타냄 -; 및

상기 명령어에 응답하여, 상기 목적지 스토리지 위치에 결과 패킹된 데이터를 저장하는 수단

을 포함하고,

상기 결과 패킹된 데이터는 적어도 4개의 인덱스들을 포함하고,

상기 인덱스들은 상기 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서의 대응 데이터 엘리먼트 위치들을 식별하며,

상기 인덱스들은 상기 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서의 대응 데이터 엘리먼트들의 정렬된 순서를 나타내는 상기 결과 패킹된 데이터에서의 위치들에 저장되고,

상기 명령어는, 정렬된 순서로 상기 적어도 4개의 데이터 엘리먼트들을 갖는 상기 제1 소스 패킹된 데이터를 나타내는 프로세서.

청구항 23

머신에 의해 실행되는 경우, 상기 머신으로 하여금 제12항의 방법을 수행하게 하도록 동작될 수 있는 명령어를 제공하는 비-일시적 머신-판독가능 스토리지 매체.

청구항 24

머신에 의해 실행되는 경우, 머신으로 하여금 제17항 또는 제18항의 방법을 수행하게 하도록 동작될 수 있는 명령어를 제공하는 비-일시적 머신-판독가능 스토리지 매체.

청구항 25

제17항 또는 제18항의 방법을 수행하는 수단을 포함하는 프로세서.

발명의 설명

기 술 분 야

[0001] 본 명세서에 개시되는 실시예들은 일반적으로 프로세서들에 관한 것이다. 특히, 본 명세서에 개시되는 실시예들은 일반적으로 프로세서들에서 데이터를 정렬하는 것에 관한 것이다.

배 경 기 술

[0002] 데이터 정렬 연산들은 컴퓨터들, 서버들, 데이터센터들, 정렬 네트워크들 등에서 널리 사용된다. 예를 들어, 정렬 연산들은, 몇 가지 예를 들어 보면, 스프레드시트들, 데이터베이스들, SQL(Sturctured Query Language) 데이터베이스들 또는 서버들, 데이터센터들, HPC(High Performance Computing), Apache Hadoop 소프트웨어 프레임워크, 이미지 프로세싱(예를 들어, 미디언 필터, 제스처 인식 등을 위함) 및 신경 네트워크들에 흔히 사용된다. 특히 큰 데이터 세트들을 정렬할 때, 정렬 연산들은 계산상 집약적일 수 있어 전체 성능에 영향을 주는 경향이 있을 수 있다.

발명의 내용

도면의 간단한 설명

[0003] 본 발명은 실시예들을 설명하는데 사용되는 이하의 설명 및 첨부 도면들을 참조하여 최상으로 이해될 수 있다. 도면들에서:

도 1은 정렬 가속화 명령어의 일 실시예를 수행하도록 동작될 수 있는 프로세서의 일 실시예의 블럭도이다.

도 2는 싱글 소스 정렬 인덱스들 명령어의 일 실시예를 수행하는 방법의 일 실시예의 블럭 흐름도이다.

도 3은 싱글 소스 정렬 인덱스들 연산의 일 실시예의 블럭도이다.

도 4는 싱글 소스 정렬 인덱스들 및 데이터 엘리먼트들 연산의 일 실시예의 블럭도이다.

도 5는 2개 소스 정렬 인덱스들 명령어의 일 실시예를 수행하는 방법의 일 실시예의 블럭 흐름도이다.

도 6은 데이터 엘리먼트들 중 최소 1/2에 대한 2개 소스 정렬 인덱스들 연산의 일 실시예의 블럭도이다.

도 7은 데이터 엘리먼트들 중 최대 1/2에 대한 2개 소스 정렬 인덱스들 연산의 일 실시예의 블럭도이다.

도 8은 데이터 엘리먼트들 중 최소 1/2에 대한 2개 정렬되지 않은 소스 정렬 인덱스들 연산의 일 실시예의 블럭도이다.

도 9는 데이터 엘리먼트들 중 최소 1/2에 대한 2개 소스 정렬 인덱스들 및 데이터 연산의 일 실시예의 블럭도이다.

도 10은 데이터 엘리먼트들 중 최소 1/2에 대한 마스크를 갖는 2개 소스 정렬 인덱스들 연산의 일 실시예의 블럭도이다.

도 11은 데이터 엘리먼트들 중 최대 1/2에 대한 마스크를 갖는 2개 소스 정렬 인덱스들 연산의 일 실시예의 블럭도이다.

도 12는 데이터 엘리먼트들 중 최소 1/2에 대한 마스크를 갖는 2개 정렬되지 않은 소스 정렬 인덱스들 연산의 일 실시예의 블럭도이다.

도 13은 데이터 엘리먼트들 중 최소 1/2에 대한 마스크를 갖는 2개 소스 정렬 인덱스들 및 정렬 데이터 연산의 일 실시예의 블럭도이다.

도 14는 패킹된 데이터 레지스터들의 적합한 세트의 일 실시예의 블럭도이다.

도 15a-15b는, 본 발명의 실시예들에 따라, 일반 벡터 친화형 명령어 포맷 및 그 명령어 템플릿들을 도시하는 블럭도들이다.

도 16a는, 본 발명의 실시예들에 따라, 예시적인 특정 벡터 친화형 명령어 포맷을 도시하는 블럭도이다.

도 16b는, 본 발명의 일 실시예에 따라, 전체 오피코드 필드(full opcode field)를 구성하는 특정 벡터 친화형 명령어 포맷의 필드들을 도시하는 블럭도이다.

도 16c는, 본 발명의 일 실시예에 따라, 레지스터 인덱스 필드를 구성하는 특정 벡터 친화형 명령어 포맷의 필드들을 도시하는 블럭도이다.

도 16d는, 본 발명의 일 실시예에 따라, 증대(augmentation) 연산 필드를 구성하는 특정 벡터 친화형 명령어 포맷의 필드들을 도시하는 블럭도이다.

도 17은 레지스터 아키텍처의 일 실시예의 블럭도이다.

도 18a는 순차적(in-order) 파이프라인의 일 실시예 및 레지스터 리네이밍 비순차적(out-of-order) 발행/실행 파이프라인의 일 실시예를 도시하는 블럭도이다.

도 18b는 실행 엔진 유닛에 연결되는 프론트 엔드 유닛을 포함하는 프로세서 코어의 일 실시예의 블럭도로, 이들 양자 모두는 메모리 유닛에 연결된다.

도 19a는 싱글 프로세서 코어의 일 실시예의 블럭도로, 온-다이(on-die) 상호접속 네트워크에 대한 접속, 및 L2(Level 2) 로컬 서브세트를 함께 보여준다.

도 19b는 도 19a의 프로세서 코어의 일부의 확대도의 실시예의 블럭도이다.

도 20은, 하나 보다 많은 코어를 가질 수 있고, 통합 메모리 컨트롤러를 가질 수 있으며, 통합 그래픽들을 가질 수 있는 프로세서의 일 실시예의 블럭도이다.

도 21은 컴퓨터 아키텍처의 제1 실시예의 블럭도이다.

도 22는 컴퓨터 아키텍처의 제2 실시예의 블럭도이다.

도 23은 컴퓨터 아키텍처의 제3 실시예의 블럭도이다.

도 24는 컴퓨터 아키텍처의 제4 실시예의 블럭도이다.

도 25는, 본 발명의 실시예들에 따라, 소프트웨어 명령어 변환기를 사용하여 소스 명령어 세트에서의 바이너리 명령어들을 타겟 명령어 세트에서의 바이너리 명령어들로 변환하는 블럭도이다.

발명을 실시하기 위한 구체적인 내용

[0004] 본 명세서에는 정렬 가속화 명령어들, 이러한 명령어들을 실행하는 프로세서들, 이러한 명령어들을 처리하거나 또는 실행할 때 프로세서들에 의해 수행되는 방법들 및 이러한 명령어들을 처리하거나 또는 실행하는 하나 이상의 프로세서들을 통합하는 시스템들이 개시된다. 이하의 설명에서는, 많은 특정 상세들이 제시된다(예를 들어, 특정 명령어 연산들, 패킹된 데이터 포맷들, 프로세서 구성들, 마이크로아키텍처의 상세들, 연산들의 시퀀스들 등). 그러나, 실시예들은 이러한 특정 상세들 없이 실시될 수 있다. 다른 경우들에서, 잘 알려진 회로들, 구조들 및 기술들은 본 설명의 이해를 모호하게 하는 것을 회피하기 위해 상세하게 나타내지 않았다.

[0005] 도 1은 정렬 가속화 명령어(102)의 일 실시예를 수행하도록 동작될 수 있는 프로세서(100)의 일 실시예의 블럭도이다. 일부 실시예들에서, 프로세서는 (예를 들어, 데스크톱, 랩톱 또는 다른 컴퓨터들에 자주 사용되는 타입의) 범용 프로세서일 수 있다. 대안적으로, 프로세서는 특수 목적 프로세서일 수 있다. 적절한 특수 목적 프로세서들의 예들은, 네트워크 프로세서들, 통신 프로세서들, 그래픽 프로세서들, 암호화 프로세서들, 코-프로세서들, 임베디드 프로세서들, 디지털 신호 프로세서들(DSP) 및 컨트롤러들(예를 들어, 마이크로컨트롤러들)을 포함하지만, 이에 제한되는 것은 아니다. 프로세서는 다양한 CISC(Complex Instruction Set Computing) 프로세서들, RISC(Reduced Instruction Set Computing) 프로세서들, VLIW(Very Long Instruction Word) 프로세서들, 이들의 하이브리드들, 또는 다른 타입들의 프로세서들 중 임의의 것을 포함할 수 있거나, 또는 (예를 들어, 상이한 코어들에서) 이러한 상이한 프로세서들의 조합을 가질 수 있다.

[0006] 연산 동안, 프로세서(100)는 정렬 가속화 명령어(102)의 실시예를 수신할 수 있다. 예를 들어, 이러한 명령어는 명령어 페치 유닛, 명령어 큐 등으로부터 수신될 수 있다. 정렬 가속화 명령어는 매크로명령어, 어셈블리 언어 명령어, 머신 코드 명령어, 또는 프로세서의 명령어 세트의 명령어 또는 제어 신호를 나타낼 수 있다. 일부 실시예들에서, 정렬 가속화 명령어는, 소스 패킹된 데이터(110)를, (예를 들어, 하나 이상의 필드들 또는 비트들의 세트를 통해) 명시적으로 특정할 수 있거나 또는 다른 방식으로 나타낼 수 있고(예를 들어, 암시적으로 나타냄), 결과 패킹된 데이터(114)가 저장될 목적지(예를 들어, 목적지 스토리지 위치)를 특정할 수 있거나 또는 다른 방식으로 나타낼 수 있다. 일부 실시예들에서는, 명령어가 제2 소스 패킹된 데이터(112)를 선택적으로 특정하거나 또는 다른 방식으로 나타낼 수 있고, 일부 실시예들에서는, 제2 결과 패킹된 데이터(116)가 저장될 제2 목적지(예를 들어, 목적지 스토리지 위치)를 선택적으로특정하거나 또는 다른 방식으로 나타낼 수 있다.

[0007] 다시 도 1을 참조하면, 프로세서는 디코드 유닛 또는 디코더(104)를 포함한다. 디코드 유닛은 정렬 가속화 명령어를 디코드할 수 있다. 디코드 유닛은, 소스 가속화 명령어를 반영하고, 나타내고, 및/또는 이들로부터 유도되는, 하나 이상의 마이크로 명령어들, 마이크로-연산들, 마이크로-코드 엔트리 포인트들, 디코드된 명령어들 또는 제어 신호들, 또는 다른 비교적 하위 레벨의 명령어들 또는 제어 신호들을 출력할 수 있다. 하나 이상의 하위 레벨의 명령어들 또는 제어 신호들은 하나 이상의 하위 레벨(예를 들어, 회로 레벨 또는 하드웨어 레벨)의 연산들을 통해 상위 레벨의 명령어를 구현할 수 있다. 일부 실시예들에서, 디코드 유닛은, 명령어를 수신하는 하나 이상의 입력 구조들(예를 들어, 포트(들), 상호접속(들), 인터페이스), 입력 구조들과 연결되어 명령어를 인식하고 디코드하는 명령어 인식 및 디코드 로직, 및 명령어 인식 및 디코드 로직과 연결되어 하나 이상의 대응하는 하위 레벨의 명령어들 또는 제어 신호들을 출력하는 하나 이상의 출력 구조들(예를 들어, 포트(들), 상호접속(들), 인터페이스)을 포함할 수 있다. 디코드 유닛은, 본 분야에 공지된 디코드 유닛들을 구현하는데 사용되는 마이크로코드 ROM들(Read Only Memories), 룩-업 테이블들, 하드웨어 구현들, PLA들(Programmable Logic Arrays) 및 기타 메커니즘들을 포함하지만, 이에 제한되는 것은 아니다. 여러가지 상이한 메커니즘들을 사용하여 구현될 수 있다.

[0008] 일부 실시예들에서는, 정렬 가속화 명령어가 디코드 유닛에 직접 제공되는 대신에, 명령어 에뮬레이터, 번역기, 모퍼(morpher), 해석기 또는 다른 명령어 변환 모듈이 선택적으로 사용될 수 있다. 다양한 타입들의 명령어 변환 모듈들이 본 분야에 공지되어 있으며, 소프트웨어, 하드웨어, 펌웨어 또는 이들의 조합으로 구현될 수 있다. 일부 실시예들에서는, 명령어 변환 모듈이, 예를 들어, (예를 들어, 스태틱, 다이나믹 또는 런타임 에뮬레이션

모듈로서) 별도의 다이 상에 및/또는 메모리 내에 등과 같이 프로세서 외부에 위치될 수 있다. 예를 들어, 명령어 변환 모듈은, 제1 명령어 세트의 것일 수 있는 정렬 가속화 명령어를 수신할 수 있고, 정렬 가속화 명령어를, 제2 상이한 명령어 세트의 것일 수 있는 하나 이상의 대응하는 또는 유도되는 중간 명령어들 또는 제어 신호들로, 에뮬레이트하거나, 번역하거나, 모프(morph)하거나, 해석하거나 또는 다른 방식으로 변환할 수 있다. 제2 명령어 세트의 하나 이상의 중간 명령어들 또는 제어 신호들은 디코드 유닛에 제공될 수 있고, 디코드 유닛은 이들을 프로세서의 네이티브 하드웨어(예를 들어 하나 이상의 실행 유닛들)에 의해 실행될 수 있는 하나 이상의 하위 레벨의 명령어들 또는 제어 신호들로 디코드할 수 있다.

[0009] 프로세서(100)는 또한 패킹된 데이터 레지스터들(108)의 세트를 포함한다. 패킹된 데이터 레지스터들 각각은 패킹된 데이터, 벡터 데이터 또는 SIMD 데이터를 저장하도록 동작될 수 있는 온-다이 스토리지 위치를 나타낼 수 있다. 패킹된 데이터 레지스터들은 아키텍처상-가시적인 레지스터들(예를 들어, 아키텍처의 레지스터 파일)을 나타낼 수 있다. 아키텍처상-가시적인 또는 아키텍처의 레지스터들은 소프트웨어 및/또는 프로그래머에게 보일 수 있고, 및/또는 피연산자들을 식별하기 위해 프로세서의 명령어 세트의 명령어들에 의해 나타나는 레지스터들이다. 이러한 아키텍처의 레지스터들은 주어진 마이크로아키텍처에서 다른 비-아키텍처의 또는 비-아키텍처상 가시적인 레지스터들(예를 들어, 임시 레지스터들, 재정리 버퍼들, 회수 레지스터들 등)과 대조적이다. 패킹된 데이터 레지스터들은 공지된 기술들을 사용하여 상이한 마이크로아키텍처들에서 상이한 방식으로 구현될 수 있고 임의의 특정 타입의 회로에 제한되는 것은 아니다. 적합한 타입들의 레지스터들의 예들은, 전용 물리적 레지스터들, 레지스터 리네이밍을 사용하는 다이나믹하게 할당되는 물리적 레지스터들 및 이들의 조합들을 포함하지만, 이에 제한되는 것은 아니다.

[0010] 도시된 바와 같이, 일부 실시예들에서는, 제1 소스 패킹된 데이터(110), 선택적 제2 소스 패킹된 데이터(112), 제1 결과 패킹된 데이터(114) 및 선택적 제2 결과 패킹된 데이터(116)가 패킹된 데이터 레지스터들에 각각 선택적으로 저장될 수 있다. 대안적으로, 메모리 위치들 또는 기타 스토리지 위치들이 하나 이상의 이러한 피연산자들에 대해 사용될 수 있다. 또한, 도면에서는 이들이 별도의 것으로서 도시되지만, 일부 실시예들에서는, 소스 패킹된 데이터에 대해 사용되는 패킹된 데이터 레지스터가 목적지 스토리지 위치로서 재사용될 수 있다(예를 들어, 결과 패킹된 데이터가 소스 패킹된 데이터 위에 기입될 수 있다).

[0011] 다시 도 1을 참조하면, 실행 유닛(106)은 디코드 유닛(104) 및 패킹된 데이터 레지스터들(108)에 연결된다. 실행 유닛은 정렬 가속화 명령어를 나타내는 및/또는 정렬 가속화 명령어로부터 유도되는 하나 이상의 디코드된 또는 다른 방식으로 변환된 명령어들 또는 제어 신호들을 수신할 수 있다. 실행 유닛은 또한 제1 소스 패킹된 데이터(110)를 수신할 수 있고, 일부 실시예들에서는 제2 소스 패킹된 데이터(112)를 수신할 수 있다. 실행 유닛은 정렬 가속화 명령어에 응답하여 및/또는 정렬 가속화 명령어의 결과로서 (예를 들어, 명령어로부터 디코드된 하나 이상의 명령어 또는 제어 신호에 응답하여) 동작될 수 있거나 또는 구성되어, 명령어에 의해 나타나는 제1 목적지에 제1 결과 패킹된 데이터(114)를 저장하고, 일부 실시예들에서는 명령어에 의해 나타나는 제2 목적지에 제2 결과 패킹된 데이터(116)를 저장한다. 일부 실시예들에서, 실행 유닛은 또한 선택적으로 (예를 들어, 마스크 레지스터들(118)의 선택적 세트에) 결과 마스크(120)를 저장할 수 있다. 다양한 실시예들에서, 결과 패킹된 데이터 및/또는 결과 마스크는 도 3-4 또는 6-13 중 임의의 것일 수 있다.

[0012] 실행 유닛 및/또는 프로세서는 정렬 가속화 명령어를 수행하고 및/또는 명령어의 결과에 응답하여 및/또는 명령어의 결과로서 결과를 저장하도록 동작될 수 있는 구체적인 또는 특정한 로직(예를 들어, 트랜지스터들, 집적 회로, 또는 펌웨어(예를 들어, 불휘발성 메모리에 저장된 명령어들) 및/또는 소프트웨어와 잠재적으로 조합되는 기타 하드웨어)을 포함할 수 있다. 예를 들어, 실행 유닛은 산술 논리 유닛, 논리 유닛, 데이터 재배치 유닛 등을 포함할 수 있다. 일부 실시예들에서, 실행 유닛은, 소스 피연산자들을 수신하는 하나 이상의 입력 구조들(예를 들어, 포트(들), 상호접속(들), 인터페이스), 입력 구조(들)와 연결되고 소스 패킹된 데이터를 수신 및 처리하여 결과 패킹된 데이터를 생성하는 정렬 회로 또는 로직(107), 정렬 회로 또는 로직(107)과 연결되고 결과 패킹된 데이터를 출력하는 하나 이상의 출력 구조들(예를 들어, 포트(들), 상호접속(들), 인터페이스)을 포함할 수 있다. 예를 들어, 정렬 회로 또는 로직은 비교 및 스왑 체인, 값 기반 선택 또는 라우팅 계층구조, 또는 기타 정렬 회로 또는 로직을 포함할 수 있다.

[0013] 도 2는 싱글 소스 정렬 인덱스들 명령어의 일 실시예를 수행하는 방법(225)의 일 실시예의 블럭 흐름도이다. 다양한 실시예들에서, 본 방법은 프로세서(예를 들어, 도 1의 프로세서), 명령어 처리 장치 또는 집적 회로에 의해 수행될 수 있다. 도 1의 프로세서에 대해 본 명세서에 개시되는 컴포넌트들, 특징들 및 특정 선택 상세들이 또한 도 2의 연산들 및/또는 방법에 선택적으로 적용된다.

[0014] 본 방법은, 블럭 226에서, 싱글 소스 정렬 인덱스들 명령어를 수신하는 것을 포함한다. 이러한 명령어는 프로세서 또는 그 일부(예를 들어, 명령어 페치 유닛, 디코드 유닛, 버스 인터페이스 유닛)에서 수신될 수 있다. 다양한 양상들에서, 명령어는 오프-다이 소스로부터(예를 들어, 메모리, 상호접속 등으로부터) 또는 온-다이 소스로부터(예를 들어, 명령어 캐시, 명령어 큐로부터) 수신될 수 있다. 명령어는 정렬된 순서로 있지 않는(예를 들어, 패킹된 데이터에 걸쳐 증가하는 또는 감소하는 크기로 정렬되지 않은) 적어도 4개의 데이터 엘리먼트들 포함하는 소스 패킹된 데이터를 특정할 수 있거나 또는 다른 방식으로 나타낼 수 있다. 명령어는 또한 목적지 스토리지 위치를 특정할 수 있거나 또는 다른 방식으로 나타낼 수 있다.

[0015] 블럭 227에서는, 결과 패킹된 데이터가 명령어에 응답하여 및/또는 명령어의 결과로서 목적지 스토리지 위치에 저장된다. 대표적으로, 실행 유닛 또는 프로세서가 명령어를 수행하고 그 결과를 저장할 수 있다. 결과 패킹된 데이터는 적어도 4개의 인덱스들을 포함할 수 있다. 일부 실시예들에서, 이러한 인덱스들은 소스 패킹된 데이터에서 대응 데이터 엘리먼트들을 식별할 수 있다. 일부 실시예들에서, 인덱스들은 소스 패킹된 데이터의 대응 데이터 엘리먼트들의 정렬된 순서를 나타내는 결과 패킹된 데이터에서의 위치들에 저장될 수 있다. 즉, 인덱스들은 그들이 인덱스들 자체의 값들에 기초하여서가 아니라 소스 패킹된 데이터에서 그들의 대응 데이터 엘리먼트들의 값들에 기초하여서 정렬되는 정렬된 순서로 저장될 수 있다. 일부 실시예들에서, 결과 패킹된 데이터는 소스 패킹된 데이터에서의 모든 데이터 엘리먼트들에 대응하는 인덱스들을 포함할 수 있다.

[0016] 도 3은 싱글 소스 정렬 인덱스들 명령어의 일 실시예에 응답하여 수행될 수 있는 싱글 소스 정렬 인덱스들 연산(330)의 일 실시예를 도시하는 블럭도이다. 싱글 소스 정렬 인덱스들 명령어는 적어도 4개의 데이터 엘리먼트들을 갖는 소스 패킹된 데이터(310)를 특정할 수 있거나 또는 다른 방식으로 나타낼 수 있다. 소스 패킹된 데이터는 패킹된 데이터 레지스터, 메모리 위치 또는 기타 스토리지 위치에 저장될 수 있다. 보통, 소스 패킹된 데이터에서 데이터 엘리먼트들의 수는 각 데이터 엘리먼트의 비트 사이즈 또는 폭으로 분할된 소스 패킹된 데이터의 비트 사이즈 또는 폭과 등가일 수 있다. 다양한 실시예들에서, 소스 패킹된 데이터의 폭은 64 비트, 128 비트, 256 비트, 512 비트 또는 1024 비트일 수 있다. 다양한 실시예들에서, 각 데이터 엘리먼트의 폭은 8 비트, 16 비트, 32 비트 또는 64 비트일 수 있다. 하나의 특정 비제한적 예에서, 소스 패킹된 데이터는 512 비트 폭이고 16개의 32 비트 데이터 엘리먼트들 또는 8개의 64 비트 데이터 엘리먼트들을 가질 수 있다. 데이터 엘리먼트들은 정수, 고정 소수점 또는 부동 소수점 포맷을 가질 수 있다.

[0017] 특정 도시된 실시예에서, 소스 패킹된 데이터는 8개의 데이터 엘리먼트들을 갖는다. 8개의 데이터 엘리먼트들은, 우측의 최하위 또는 최저 순위 비트 위치로부터 좌측의 최상위 또는 최고 순위 비트 위치까지, -4, 1, 8, 12, 43, 55, 1 및 12의 값들을 갖는다. 이러한 값들은 정렬된 순서로 있지 않는다(예를 들어, 패킹된 데이터에 걸쳐 증가하거나 또는 감소하는 순서로 정렬되지 않는다).

[0018] 결과 패킹된 데이터(314)는 싱글 소스 정렬 인덱스들 명령어에 응답하여 (예를 들어, 실행 유닛(106)에 의해) 생성되어 목적지 스토리지 위치에 저장될 수 있다. 목적지 스토리지 위치는 명령어에 의해 특정될 수 있거나 다른 방식으로 나타날 수 있다. 다양한 실시예들에서, 목적지 스토리지 위치는 패킹된 데이터 레지스터, 메모리 위치 또는 기타 스토리지 위치일 수 있다. 목적지 스토리지 위치는 소스 패킹된 데이터에 대해 사용된 스토리지 위치와 동일할 수 있거나, 또는 다른 스토리지 위치일 수 있다.

[0019] 일부 실시예들에서, 결과 패킹된 데이터는 적어도 4개의 인덱스들을 포함할 수 있다. 특정 도시된 예에서, 결과 패킹된 데이터는 8개의 인덱스들을 갖는다. 각각의 인덱스는 소스 패킹된 데이터에서 싱글 대응 데이터 엘리먼트를 가리키거나, 식별하거나 또는 이와 관련되는 것일 수 있다. 데이터 엘리먼트들 각각은 대응 인덱스에 의해 나타나는 소스 패킹된 데이터 내의 위치를 가질 수 있다. 인덱스는 피연산자 내의 데이터 엘리먼트의 오프셋 또는 상대 위치를 나타낼 수 있다. 예를 들어, 도면에 사용되는 하나의 가능한 규약에 따르면, 0 내지 7의 인덱스 값들은 패킹된 데이터에 걸쳐 최하위 비트 위치로부터 최상위 비트 위치로(바라볼 때 우측에서 좌측으로) 이동하는 제1 내지 제8 위치들에서의 8개 데이터 엘리먼트를 나타낼 수 있다. 더욱 설명하자면, 값이 -4인 데이터 엘리먼트는 0의 인덱스를 갖고, 값이 8인 데이터 엘리먼트는 2의 인덱스를 갖고, 값이 43인 데이터 엘리먼트는 4의 인덱스를 갖는 등등이다. 대안적으로, 다양한 기타 인덱싱 규약들이 선택적으로 사용될 수 있다(예를 들어, 0 대신 1로 시작하는 것, 7로부터 0으로 역으로 인덱싱하는 것, 임의의 맵핑 규약 등). 인덱스들은 인덱스되는 위치들을 나타내기에 충분한 수의 비트들을 가질 수 있다(예를 들어, 도시된 예에서는 3 비트).

[0020] 일부 실시예들에서, 인덱스들은 소스 패킹된 데이터에서 대응 데이터 엘리먼트들의 정렬된 순서를 나타낼 결과 패킹된 데이터에서의 위치들에 저장될 수 있다. 도시된 예에서는, 정렬된 순서가 최하위 비트 위치로부터 최상

위 비트 위치로 증가하는 증가 순서이지만, 감소하는 순서가 그 대신 선택적으로 사용될 수 있다. 도시된 예에서의 값들에 대해, 증가하는 정렬된 순서는 -4, 1, 1, 8, 12, 12, 43 및 55일 것이다. 결과적으로, 값이 -4인 데이터 엘리먼트에 대응하는 0의 인덱스 값이 결과 패킹된 데이터에서 제1 또는 최하위 위치에 저장되고, 값이 1인 최우측 데이터 엘리먼트에 대응하는 1의 인덱스 값이 제2 또는 다음 최하위 위치에 저장되는 등등이다. 도면에서, 화살표는 인덱스들과 데이터 엘리먼트들 사이의 대응을 보여주는데 사용된다.

[0021] 결과 패킹된 데이터는 정렬된 소스 데이터 엘리먼트들이 아니라 정렬된 인덱스들을 저장한다는 점에 주의하자. 일부 실시예들에서는, 정렬된 인덱스들이 상이한 명령어에 의해 후속하여 소스 데이터 엘리먼트들을 정렬하는데 선택적으로 사용될 수 있다. 예를 들어, 일부 실시예들에서는, 치환 명령어, 셔플(shuffle) 명령어 등이 인덱스들을 사용하여 소스 데이터 엘리먼트들을 정렬할 수 있다. 예를 들어, 치환 또는 셔플 명령어는 인덱스를 제1 소스 피연산자로 하고 소스 패킹된 데이터(310)를 제2 소스 피연산자로 하여 결과 패킹된 데이터(314)를 나타낼 수 있고, 인덱스들의 제어에 기초하여 정렬된 데이터 엘리먼트들을 갖는 결과 패킹된 데이터를 저장할 수 있다. 적합한 명령어의 하나의 특정한 예는 VPERMD 명령어이고, 이는 캘리포니아주 산타 클라라의 Intel Corporation으로부터 입수할 수 있고 2013년 12월에 공개된 "Intel® Architecture Instruction Set Extensions Programming Reference, 319433-017"에 개시된다.

[0022] 그러나, 일부 구현들에서는, 정렬된 데이터 엘리먼트들 대신에 인덱스들을 저장하는 것이 유리하다. 예를 들어, 정렬된 인덱스들은, 소스 패킹된 데이터(310)의 소스 데이터 엘리먼트들에 추가하여, 또는 그 대신에 다른 데이터를 정렬하는데 사용될 수 있다. 예를 들어, 이는 소스 데이터 엘리먼트들이 정렬 키들로서 사용되고 각각 다수의 데이터 엘리먼트들과 관련되는 여러가지 상이한 데이터 구조들에서의 경우일 수 있다. 더 설명하자면, 열들과 행들로 배열되는 (예를 들어, 스프레드시트에서의) 테이블의 간단한 예를 고려하자. 하나의 열은 발명자 이름들을 열거하고, 다른 열은 발명자 식별 번호들을 열거한다. 이름들 및 식별 번호들 양자 모두가 상호 함께 일관되게 정렬될 필요가 있을 수 있다. 일 예로서, 정렬은 정렬 키로서 이름에 기초할 수 있고, 정렬된 인덱스들이 결과로서 저장될 수 있다. 그러면, 정렬된 인덱스들이 이름들 및 식별 번호들 양자 모두를 정렬하는데 사용될 수 있다. 따라서, 일부 구현들에서, 정렬된 인덱스들을 저장하는 것은, 정렬된 데이터 엘리먼트들을 저장하는 것에 비해서, 다른 관련된 데이터를 정렬하는 능력 및 융통성 등의 이점들을 제공할 수 있다.

[0023] 일부 실시예들에서, 이러한 정렬은 안정된 순서(예를 들어, 오름차순)로 행해질 수 있다. 안정된이란, 2개 엘리먼트들이 등가의 탐색 키들을 갖는다면, 이들 사이의 상대적인 순서는 변경되지 않는다는 것을 의미한다. 예를 들어, 당신이 발명자 이름은 동일하지만 식별 번호들이 상이한 2개의 행들을 갖는다면, 당신은 이 테이블을 이름에 따라 정렬하고, 식별 번호들의 순서는 변경되지 않을 수 있다. 이는, 상대적인 순서가 보장되지 않고 키들이 등가인 엘리먼트들이 셔플될 수 있는 불안정한 순서와 대조된다.

[0024] 도 4는 싱글 소스 정렬 인덱스들 및 데이터 엘리먼트들 명령어의 일 실시예에 응답하여 수행될 수 있는 싱글 소스 정렬 인덱스들 및 데이터 엘리먼트들 연산(432)의 일 실시예를 도시하는 블럭도이다. 도 4의 연산은 도 3의 연산과 어느 정도 유사성들을 갖는다. 설명을 모호하게 하는 것을 회피하기 위해, 도 3의 연산에 비해 선택적으로 유사하거나 공통인 특징들 및 상세들 모두를 반복하지는 않고, 도 4의 연산에 대해 상이한 및/또는 추가적인 특징들이 주로 개시될 것이다. 그러나, 도 3의 연산의 이미 개시된 특징들 및 상세들 또한, 달리 또는 다른 방식으로 명명 백백히 다르게 언급되지 않는 한, 도 4의 연산에 선택적으로 적용될 수 있다.

[0025] 싱글 소스 정렬 인덱스들 및 데이터 엘리먼트들 명령어는 적어도 4개의 데이터 엘리먼트들을 갖는 소스 패킹된 데이터(410)를 특정하거나 또는 다른 방식으로 나타낼 수 있다. 소스 패킹된 데이터 및 데이터 엘리먼트들은 본 명세서의 다른 곳에 개시된 것(예를 들어, 도 3의 것)과 같을 수 있다.

[0026] 제1 결과 패킹된 데이터(414)가 명령어에 응답하여 (예를 들어, 실행 유닛(106)에 의해) 생성되어 제1 목적지 스토리지 위치에 저장될 수 있다. 제1 결과 패킹된 데이터(414)는 적어도 4개의 인덱스들을 갖는다. 제1 결과 패킹된 데이터는 본 명세서의 다른 곳에 개시된 것(예를 들어, 도 3의 것)과 같을 수 있다.

[0027] 본 실시예에서, 제2 결과 패킹된 데이터(416) 또한 명령어에 응답하여 생성되어 제2 목적지 스토리지 위치에 저장될 수 있다. 제2 목적지 스토리지 위치는 명령어에 의해 특정되거나 또는 다른 방식으로 나타날 수 있고, 패킹된 데이터 레지스터, 메모리 위치 또는 기타 스토리지 위치일 수 있다. 제2 결과 패킹된 데이터는, 제1 결과 패킹된 데이터(414)에 저장된 인덱스들에 대응하며, 정렬된 순서를 반영하는 제2 결과 패킹된 데이터(416)의 위치들에 저장되는, 대응 데이터 엘리먼트들을 포함할 수 있다. 예를 들어, 증가하는 정렬된 순서에서 제2 결과 패킹된 데이터는, 도면에서 우측으로부터 좌측으로, -4, 1, 1, 8, 12, 12, 43 및 55의 값들을 저장할 수 있다. 다른 실시예에서는, 인덱스들 및 데이터 엘리먼트들에 대해 감소하는 순서가 대신 사용될 수 있다.

[0028] 도 3에서 결과로서 정렬된 인덱스들이 저장된다. 도 4에서 결과들로서 정렬된 인덱스들 및 정렬된 데이터 엘리먼트들 양자 모두가 저장된다. 다른 실시예에서는, 인덱스들이 아니라 정렬 데이터 엘리먼트들이 명령어에 응답하여 선택적으로 저장될 수 있다.

[0029] 도 5는 2개 소스 정렬 인덱스들 명령어의 일 실시예를 수행하는 방법(535)의 일 실시예의 블럭 흐름도이다. 다양한 실시예들에서, 본 방법은 프로세서(예를 들어, 도 1의 프로세서), 명령어 처리 장치, 디지털 논리 디바이스 또는 집적 회로에 의해 수행될 수 있다. 도 1의 프로세서에 대해 본 명세서에 개시되는 컴포넌트들, 특징들 및 특정 선택적 상세들 또한 도 5의 연산들 및/또는 방법에 선택적으로 적용된다.

[0030] 본 방법은, 블럭 536에서, 2개 소스 정렬 인덱스들 명령어 수신하는 단계를 포함한다. 다양한 양상들에서, 명령어는 프로세서 또는 그 일부(예를 들어, 명령어 페치 유닛, 디코드 유닛, 버스 인터페이스 유닛)에서 수신될 수 있다. 다양한 양상들에서, 명령어는 오프-다이 소스로부터(예를 들어, 메모리, 상호접속 등으로부터) 또는 온-다이 소스로부터(예를 들어, 명령어 캐시, 명령어 큐로부터) 수신될 수 있다. 2개의 소스 정렬 인덱스들 명령어는, 적어도 4개의 데이터 엘리먼트들의 제1 세트를 포함하는 제1 소스 패킹된 데이터를 특정할 수 있거나 또는 다른 방식으로 나타낼 수 있고, 적어도 4개의 데이터 엘리먼트들의 제2 세트를 포함하는 제2 소스 패킹된 데이터를 특정할 수 있거나 또는 다른 방식으로 나타낼 수 있다. 명령어는 또한 목적지 스토리지 위치를 특정할 수 있거나 또는 다른 방식으로 나타낼 수 있다.

[0031] 도 5를 다시 참조하면, 블럭 537에서는, 결과 패킹된 데이터가 2개 소스 정렬 인덱스들 명령어에 응답하여 및/또는 2개 소스 정렬 인덱스들 명령어의 결과로서 목적지 스토리지 위치에 저장된다. 대표적으로, 실행 유닛 또는 프로세서가 명령어를 수행하고 그 결과를 저장할 수 있다. 결과 패킹된 데이터는 적어도 4개의 인덱스들을 포함할 수 있다. 일부 실시예들에서, 이러한 인덱스들은 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서 대응 데이터 엘리먼트 위치들을 식별할 수 있다. 일부 실시예들에서는, 인덱스들이 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터 중 하나에서의 싱글 대응 데이터 엘리먼트들을 실제 식별할 수 있다(예를 들어, 도 6-9 참조). 대안적으로, 인덱스들은 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터 각각에서 대응 데이터 엘리먼트 위치들만을 식별할 수 있고, 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터 중 하나를 식별하는 데 다른 비트가 사용되어, 싱글 대응 데이터 엘리먼트를 나타낼 수 있다(예를 들어, 도 10-13 참조). 인덱스들은 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터의 대응 데이터 엘리먼트들의 정렬된 순서를 나타내는 결과 패킹된 데이터에서의 위치들에 저장될 수 있다. 그 결과는 2개 소스들 중 어느 하나에서의 데이터 엘리먼트들에 대응하는 정렬된 인덱스들을 저장하는 병합 정렬을 나타낼 수 있다.

[0032] 도 6은 데이터 엘리먼트들 중 최소 1/2에 대한 2개 소스 정렬 인덱스들 명령어의 일 실시예에 응답하여 수행될 수 있는 데이터 엘리먼트들 중 최소 1/2에 대한 2개 소스 정렬 인덱스들 연산(640)의 일 실시예의 블럭도이다. 명령어는, 적어도 4개의 데이터 엘리먼트들의 제1 세트를 갖는 제1 소스 패킹된 데이터(610)를 특정할 수 있거나 또는 다른 방식으로 나타낼 수 있고, 적어도 4개의 데이터 엘리먼트들의 제2 세트를 갖는 제2 소스 패킹된 데이터(612)를 특정할 수 있거나 또는 다른 방식으로 나타낼 수 있다. 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터는 각각, 다른 것과 독립적으로, 패킹된 데이터 레지스터, 메모리 위치 또는 기타 스토리지 위치에 저장될 수 있다. 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터는 본 명세서의 다른 곳에 개시되는 사이즈들, 데이터 엘리먼트들의 개수, 데이터 엘리먼트들의 사이즈들 및 데이터 엘리먼트들의 타입들을 가질 수 있다.

[0033] 특정 도시된 예에서는, 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터가 각각 8개의 데이터의 엘리먼트들을 갖는다. 제1 소스 패킹된 데이터는, 우측의 최하위 위치로부터 좌측의 최상위 위치로, -4, 1, 1, 8, 12, 12, 43 및 55의 값을 갖는다. 제2 소스 패킹된 데이터는, 우측의 최하위 위치로부터 좌측의 최상위 위치로, -14, -12, 0, 10, 16, 18, 24 및 60의 값을 갖는다. 본 실시예에서, 제1 소스 패킹된 데이터에서의 값들 및 제2 소스 패킹된 데이터에서의 값들은 각각 최하위 비트 위치로부터 최상위 비트 위치로 증가하는 순서로 정렬된다. 일부 실시예들에서는, 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터 각각 내의 데이터 엘리먼트들이 (예를 들어, 명령어가 정확하게 동작하도록) 개별 패킹된 데이터 피연산자 내에서 정렬된 순서로 존재할 것이 추정될 수 있거나(예를 들어, 명령어에 대해 암시적으로 추정됨) 또는 요구될 수 있다. 다른 실시예들에서는, 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터 각각 내의 데이터 엘리먼트들이 정렬된 순서로 존재할 것이 추정되지 않을 수 있거나 또는 요구되지 않을 수 있다(예를 들어, 엘리먼트들이 정렬되지 않은 순서로 존재할 때 명령어가 정확하게 동작할 수 있음).

[0034] 결과 패킹된 데이터(614)는 명령어에 응답하여 (예를 들어, 실행 유닛(106)에 의해) 생성되어 목적지 스토리지

위치에 저장될 수 있다. 다양한 실시예들에서, 목적지 스토리지 위치는 패킹된 데이터 레지스터, 메모리 위치 또는 기타 스토리지 위치일 수 있다. 목적지 스토리지 위치는 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터 중 하나에 대해 사용되는 것과 동일한 위치일 수 있거나 또는 상이한 스토리지 위치일 수 있다.

[0035] 일부 실시예들에서, 결과 패킹된 데이터는 적어도 4개의 인덱스들을 포함할 수 있다. 일부 실시예들에서, 결과 패킹된 데이터는 소스 패킹된 데이터 중 하나에서의 데이터 엘리먼트들의 수와 동일한 수의 인덱스들을 포함할 수 있지만, 이것이 요구되는 것은 아니다. 특정 도시된 예에서, 결과 패킹된 데이터는 8개의 인덱스들을 갖는다. 각각의 인덱스는 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서 대응 데이터 엘리먼트 위치들을 가리키거나, 식별하거나 또는 이와 관련될 수 있다. 일부 실시예들에서, 인덱스들은 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터 중 하나에서 싱글 대응 데이터 엘리먼트들을 실제로 식별할 수 있다. 데이터 엘리먼트들 각각은 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터 내에 인덱스된 위치를 가질 수 있다. 인덱스는 제1 및 제2 패킹된 데이터 내의 데이터 엘리먼트의 오프셋 또는 상대 위치를 나타낼 수 있다. 예를 들어, 도시된 예에 사용되는 하나의 가능한 규약에 따르면, 0 내지 7의 인덱스 값들은 제1 패킹된 데이터(610)에 걸쳐 최하위 비트 위치로부터 최상위 비트 위치로(바라볼 때 우측에서 좌측으로) 이동하는 8개 데이터 엘리먼트를 인덱스할 수 있고, 8 내지 15의 인덱스 값들은 제2 소스 패킹된 데이터(612)에 걸쳐 최하위 비트 위치로부터 최상위 비트 위치로(바라볼 때 우측에서 좌측으로) 이동하는 8개 데이터 엘리먼트를 인덱스할 수 있다. 도시된 예에서, 화살표는 인덱스들과 데이터 엘리먼트들 사이의 대응관계를 보여주는데 사용된다. 도시된 바와 같이, 값이 -14인 데이터 엘리먼트는 8의 인덱스를 갖고, 값이 -12인 데이터 엘리먼트는 9의 인덱스를 갖는 등등이다. 대안적으로, 특정 구현에 요구되는 다양한 기타 인덱싱 규약들이 선택적으로 사용될 수 있다(예를 들어, 0 대신 1의 인덱스로 시작하는 것, 역으로 인덱싱하는 것, 임의의 맵핑 인덱싱 규약 등).

[0036] 인덱스들은 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서 대응 데이터 엘리먼트들의 정렬된 순서를 나타낼 결과 패킹된 데이터에서의 위치들에 저장될 수 있다. 일부 실시예들에서, 인덱스들은 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서의 모든 데이터 엘리먼트들 중 순서화된 서브세트에 대해서만 저장될 수 있다. 일부 실시예들에서, 순서화된 서브세트는 모든 데이터 엘리먼트들 중 순서화된 최소 서브세트(예를 들어, 최소 1/2)일 수 있다. 도시된 예에서, 8개의 최소 값들은 -14, -12, -4, 0, 1, 1, 8 및 10이다. 도시된 실시예에서, 8개의 인덱스들은 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서 대응 8개 최소 데이터 엘리먼트들의 (최하위 비트 위치로부터 최상위 비트 위치로 증가하는) 정렬된 증가하는 순서를 나타내는 결과 패킹된 데이터에서의 위치들에 저장된다. 도시된 바와 같이, 값이 -14인 데이터 엘리먼트에 대응하는 인덱스 8이 결과 패킹된 데이터에서 최하위 위치에 저장되고, 값이 -12인 데이터 엘리먼트에 대응하는 인덱스 9가 다음 최하위 위치에 저장되고, 값이 -4인 데이터 엘리먼트에 대응하는 인덱스 0이 3번째 하위 위치에 저장되는 등등이다. 다른 실시예들에서는, 감소하는 순서 또는 역 순서가 선택적으로 사용될 수 있다. 정렬 연산은 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터 양자 모두에 대해 정렬된 인덱스들을 병합한다.

[0037] 도 7은 데이터 엘리먼트들 중 최대 1/2에 대한 2개 소스 정렬 인덱스들 명령어의 일 실시예에 응답하여 수행될 수 있는 데이터 엘리먼트들 중 최대 1/2에 대한 2개 소스 정렬 인덱스들 연산(742)의 일 실시예를 도시하는 블럭도이다. 명령어는, 적어도 4개의 데이터 엘리먼트의 제1 세트를 갖는 제1 소스 패킹된 데이터(710)를 특정할 수 있거나 또는 다른 방식으로 나타낼 수 있고, 적어도 4개의 데이터 엘리먼트들의 제2 세트를 갖는 제2 소스 패킹된 데이터(712)를 특정할 수 있거나 또는 다른 방식으로 나타낼 수 있다. 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터는 본 명세서의 다른 곳에 개시되는 특징들 및 변경사항들을 가질 수 있다. 도시된 실시예에서, 제1 소스 패킹된 데이터에서의 값들 및 제2 소스 패킹된 데이터에서의 값들은 증가하는 순서로 정렬되지만, 이것이 다른 실시예들에서 또는 다른 명령어들에 대해 요구되는 것은 아니다.

[0038] 결과 패킹된 데이터(714)는 명령어에 응답하여 (예를 들어, 실행 유닛(106)에 의해) 생성되어 목적지 스토리지 위치에 저장될 수 있다. 일부 실시예들에서, 결과 패킹된 데이터는 적어도 4개의 인덱스들을 포함할 수 있다. 일부 실시예들에서, 인덱스들은 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터 중 하나에서 싱글 대응 데이터 엘리먼트들을 식별할 수 있다. 일부 실시예들에서, 인덱스들은 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서 대응 데이터 엘리먼트들의 정렬된 순서를 나타낼 결과 패킹된 데이터에서의 위치들에 저장될 수 있다.

[0039] 도 7의 실시예에서, 순서화되는 서브세트가 최소 서브세트인 대신에, 순서화되는 서브세트는 예를 들어 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서 모든 데이터 엘리먼트들 중 최대 1/2인 순서화된 최대 서브세트일 수 있다. 예를 들어, 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서 대응 8개의 최대 데이터 엘리먼트의 정렬된 순서를 나타낼 결과 패킹된 데이터에서의 위치들에 8개의 인덱스들이 저장될 수 있다. 도면에

도시된 데이터 엘리먼트들의 예시적인 값들을 고려하면, 8개의 최대 데이터 엘리먼트들은 값이 12, 12, 16, 18, 24, 43, 55 및 60인 것들이다. 도시된 실시예에서, 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서 대응 8개의 최대 데이터 엘리먼트들의 (최하위 비트 위치로부터 최상위 비트 위치로 증가하는) 정렬된 증가하는 순서를 나타낼 결과 패킹된 데이터에서의 위치들에 8개의 인덱스들이 저장된다. 도시된 바와 같이, 값이 12인 최우측 데이터 엘리먼트에 대응하는 인덱스 4가 최하위 위치에 저장되고, 값이 12인 최좌측 데이터 엘리먼트에 대응하는 인덱스 5가 다음 최하위 위치에 저장되고, 값이 16인 데이터 엘리먼트에 대응하는 인덱스 12가 제3 최하위 위치에 저장되는 등등이다. 다른 실시예들에서는, 그 대신 감소하는 순서가 선택적으로 사용될 수 있다.

[0040] 도 6-7은 오직 최소 또는 최대 1/2에 대한 인덱스들이 저장되는 실시예들을 도시한다. 다른 실시예들에서는, 최소 및 최대 1/2에 대한 인덱스들이 소스 피연산자들과 동일 사이즈의 결과에 저장될 수 있다. 예를 들어, 인덱스들이 대응 데이터 엘리먼트들 보다 충분히 작으면(예를 들어, 1/2 사이즈 이하) 이러한 것이 가능할 수 있다. 이러한 접근방식은 일부 실시예들에서 이점을 제공할 수 있다. 대안적으로, 최소 및 최대 1/2에 대한 인덱스들이 별도로 유지되어 대응 셔플 명령어, 블렌드(blend) 명령어 등의 활용을 촉진할 수 있다.

[0041] 도 8은 데이터 엘리먼트들 중 최소 1/2에 대한 2개 정렬되지 않은 소스 정렬 인덱스들 명령어의 일 실시예에 응답하여 수행될 수 있는 데이터 엘리먼트들 중 최소 1/2에 대한 2개 정렬되지 않은 소스 정렬 인덱스들 연산(844)의 일 실시예를 도시하는 블럭도이다. 명령어는, 적어도 4개의 데이터 엘리먼트의 제1 세트를 갖는 제1 소스 패킹된 데이터(810)를 특정할 수 있거나 또는 다른 방식으로 나타낼 수 있고, 적어도 4개의 데이터 엘리먼트들의 제2 세트를 갖는 제2 소스 패킹된 데이터(812)를 특정할 수 있거나 또는 다른 방식으로 나타낼 수 있다. 특정 도시된 예에서, 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터는 각각 8개의 데이터를 갖는다. 제1 소스 패킹된 데이터는, 우측의 최하위 위치로부터 좌측의 최상위 위치로, -4, 1, 8, 12, 43, 55, 1 및 12의 값들을 갖는다. 이러한 실시예에서, 제1 소스 패킹된 데이터에서의 값들은 정렬되지 않은 순서로 되어 있다. 유사하게, 제2 소스 패킹된 데이터에서의 값들은 정렬되지 않은 순서로 되어 있다. 이러한 실시예에서, 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터 각각 내의 데이터 엘리먼트들은 정렬된 순서로 있을 것이 추정되지 않을 수 있거나 또는 요구되지 않을 수 있다(예를 들어, 명령어는 엘리먼트들이 정렬되지 않은 순서로 있을 때 정확하게 동작할 수 있음).

[0042] 결과 패킹된 데이터(814)는 명령어에 응답하여 (예를 들어, 실행 유닛(106)에 의해) 생성되어 목적지 스토리지 위치에 저장될 수 있다. 일부 실시예들에서, 결과 패킹된 데이터는 적어도 4개의 인덱스들을 포함할 수 있다. 일부 실시예들에서, 인덱스들은 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터 중 하나에서 싱글 대응 데이터 엘리먼트들을 식별할 수 있다. 일부 실시예들에서, 인덱스들은 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서 대응 데이터 엘리먼트들의 정렬된 순서를 나타낼 결과 패킹된 데이터에서의 위치들에 저장될 수 있다. 도시된 실시예에서, 결과 패킹된 데이터는, 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서 모든 데이터 엘리먼트들 중 최소 서브세트- 본 경우에는 최소 1/2 -에 대응하는 인덱스들을 갖는다. 다른 실시예들에서, 결과 패킹된 데이터는, 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서 모든 데이터 엘리먼트들 중 최대 서브세트- 예를 들어, 최대 1/2 -에 대응하는 인덱스들을 가질 수 있다.

[0043] 도 9는 데이터 엘리먼트들 중 최소 1/2에 대한 2개 소스 정렬 인덱스들 및 데이터 명령어의 실시예에 응답하여 수행될 수 있는 데이터 엘리먼트들 중 최소 1/2에 대한 2개 소스 정렬 인덱스들 및 데이터 연산(946)의 일 실시예를 도시하는 블럭도이다. 명령어는, 적어도 4개의 데이터 엘리먼트들의 제1 세트를 갖는 제1 소스 패킹된 데이터(910)를 특정할 수 있거나 또는 다른 방식으로 나타낼 수 있고, 적어도 4개의 데이터 엘리먼트들의 제2 세트를 갖는 제2 소스 패킹된 데이터(912)를 특정할 수 있거나 또는 다른 방식으로 나타낼 수 있다. 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터는 본 명세서에서 다른 곳에 개시되는 사이즈들, 데이터 엘리먼트들의 수, 데이터 엘리먼트 사이즈들 및 데이터 엘리먼트 타입들을 가질 수 있다. 도시된 실시예에서, 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터 각각 내에서의 데이터 엘리먼트들은 개별 패킹된 데이터 피연산자 내에서 정렬된 순서로 될 것이 추정될 수 있거나(예를 들어, 명령어에 대해 암시적으로 추정됨) 또는 요구될 수 있다. 다른 실시예들에서는, 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터 각각 내의 데이터 엘리먼트들이 개별 패킹된 데이터 피연산자 내에서 정렬된 순서로 존재할 것이 추정되지 않을 수 있거나 또는 요구되지 않을 수 있다.

[0044] 제1 결과 패킹된 데이터(914)가 명령어에 응답하여 (예를 들어, 실행 유닛(106)에 의해) 생성되어 제1 목적지 스토리지 위치에 저장될 수 있다. 제1 결과 패킹된 데이터(914)는 인덱스들을 가질 수 있다. 제1 결과 패킹된 데이터 및 인덱스들은 도 6에 대해 이전에 개시된 것과 유사하거나 또는 공통인 특징들 및 상세들을 가질 수 있다. 도시된 실시예에서, 결과 패킹된 데이터는 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서 모든

데이터 엘리먼트들 중 최소 서브세트- 본 경우 최소 1/2 -에 대응하는 인덱스들을 갖는다. 다른 실시예에서, 결과 패킹된 데이터는 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서 모든 데이터 엘리먼트들 중 최대 서브세트- 예를 들어, 최대 1/2 -에 대응하는 인덱스들을 가질 수 있다.

[0045] 본 실시예에서, 제2 결과 패킹된 데이터(916) 또한 명령어에 응답하여 생성되어 제2 목적지 스토리지 위치에 저장될 수 있다. 제2 목적지 스토리지 위치는 명령어에 의해 특정될 수 있거나 또는 다른 방식으로 나타날 수 있고, 패킹된 데이터 레지스터, 메모리 위치 또는 기타 스토리지 위치일 수 있다. 제2 목적지 스토리지 위치는 제1 또는 제2 소스 패킹된 데이터 중 하나에 대해 사용된 것과 동일한 스토리지 위치일 수 있거나, 또는 상이한 스토리지 위치일 수 있다. 제2 결과 패킹된 데이터는, 제1 결과 패킹된 데이터에 저장된 인덱스들에 대응하며, 정렬된 순서를 반영하는 제2 결과 패킹된 데이터의 위치들에 저장되는, 대응 데이터 엘리먼트들을 포함할 수 있다. 도시된 실시예에서, 제2 결과 패킹된 데이터는 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서 모든 데이터 엘리먼트들 중 정렬된 최소 서브세트- 본 경우 정렬된 최소 1/2 -를 갖는다. 구체적으로, 제2 결과 패킹된 데이터는, 우측으로부터 좌측으로, -14, -12, -4, 0, 1, 1, 8 및 10인 데이터 엘리먼트들을 저장한다. 다른 실시예에서, 제2 결과 패킹된 데이터는, 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서 모든 데이터 엘리먼트들 중 정렬된 최대 서브세트- 예를 들어 정렬된 최대 1/2 -를 그 대신 가질 수 있다.

[0046] 도 6에서 결과로서 정렬된 인덱스들이 저장된다. 도 9에서 결과들로서 정렬된 인덱스들 및 정렬된 데이터 엘리먼트들 양자 모두가 저장된다. 다른 실시예에서는, 인덱스들이 아니라 정렬 데이터 엘리먼트들이 명령어에 응답하여 선택적으로 저장될 수 있다.

[0047] 도 7-9는 도 6의 연산과 어느 정도 유사성들을 갖는 연산들을 도시한다. 설명을 모호하게 하는 것을 회피하기 위해서, 도 6의 연산에 대해 선택적으로 유사하거나 또는 공통인 특징들 및 상세들 모두를 반복하지는 않고, 도 7-9의 연산들에 대한 상이한 및/또는 추가적인 특징들이 주로 개시되었다. 그러나, 도 6의 연산의 이미 개시된 특징들 및 상세들 또한, 달리 또는 다른 방식으로 명명 백백히 다르게 언급되지 않는 한, 도 7-9의 연산들 중 임의의 것에 선택적으로 적용될 수 있다는 점이 이해되어야 한다.

[0048] 도 10은 데이터 엘리먼트들 중 최소 1/2에 대해 마스크를 갖는 2개 소스 정렬 인덱스들 명령어의 일 실시예에 응답하여 수행될 수 있는 데이터 엘리먼트들 중 최소 1/2에 대한 마스크를 갖는 2개 소스 정렬 인덱스들 연산(1048)의 일 실시예를 도시하는 블럭도이다. 명령어는, 적어도 4개의 데이터 엘리먼트의 제1 세트를 갖는 제1 소스 패킹된 데이터(1010)를 특정할 수 있거나 또는 다른 방식으로 나타낼 수 있고, 적어도 4개의 데이터 엘리먼트들의 제2 세트를 갖는 제2 소스 패킹된 데이터(1012)를 특정할 수 있거나 또는 다른 방식으로 나타낼 수 있다. 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터는 본 명세서에서 다른 곳에 개시되는 사이즈들, 데이터 엘리먼트들의 수, 데이터 엘리먼트의 사이즈들 및 데이터 엘리먼트들의 타입들을 가질 수 있다.

[0049] 도시된 실시예에서, 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터 각각에서의 값들은 순서대로 정렬되지만, 이것이 요구되는 것은 아니다. 일부 실시예들에서는, 소스 데이터 엘리먼트들이 개별 패킹된 데이터 내에서 정렬된 순서로 될 것이 추정될 수 있거나(예를 들어, 명령어에 대해 암시적으로 추정됨) 또는 요구될 수 있다. 도시된 예에서는, 피연산자들에서 비트 중요도(bit significance)가 증가할수록 증가하는 순서가 사용되지만, 다른 실시예에서는 감소하는 순서가 선택적으로 사용될 수 있다. 다른 실시예들에서는, 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터 각각 내의 데이터 엘리먼트들이 개별 패킹된 데이터 내에서 정렬된 순서로 존재할 것이 추정되지 않을 수 있거나 또는 요구되지 않을 수 있다(예를 들어, 엘리먼트들이 정렬되지 않은 순서로 되어 있을 때 명령어가 정확하게 동작할 수 있음).

[0050] 결과 패킹된 데이터(1014)는 명령어에 응답하여 (예를 들어, 실행 유닛(106)에 의해) 생성되어 목적지 스토리지 위치에 저장될 수 있다. 일부 실시예들에서 결과 패킹된 데이터는 적어도 4개의 인덱스들을 포함할 수 있다. 일부 실시예들에서, 각각의 인덱스는 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서 한 쌍의 대응 데이터 엘리먼트 위치들을 가리키거나, 식별하거나 또는 이와 관련될 수 있다. 인덱스들 자체는, 싱글 대응 데이터 엘리먼트를 실제로 식별하지 않을 수 있고, 오히려 실제 대응 데이터 엘리먼트가 제1 소스 패킹된 데이터에 위치되는지 아니면 제2 소스 패킹된 데이터에 위치되는지를 나타내지 않고서 제1 및 제2 패킹된 데이터에서 한 쌍의 대응 데이터 엘리먼트 위치들을 식별할 수 있다. 예를 들어, 도시된 예에서 사용되는 하나의 가능한 규약에 따르면, 0 내지 7의 인덱스 값들은 제1 소스 패킹된 데이터에 걸쳐 최하위 비트 위치로부터 최상위 비트 위치로(바라볼 때 우측에서 좌측으로) 이동할 때 8개 데이터 엘리먼트 위치들을 나타낼 수 있고, 0 내지 7의 동일한 인덱스 값들은 제2 소스 패킹된 데이터에 걸쳐 최하위 비트 위치로부터 최상위 비트 위치로 이동할 때 8개 데이터 엘리먼트 위치들을 나타낼 수 있다. 예를 들어, 값이 43인 데이터 엘리먼트 및 값이 24인 데이터 엘리

먼트 양자 모두가 동일한 인덱스 6을 갖는다. 이미 언급된 바와 같이, 인덱스들에 대한 다양한 기타 규약들이 그 대신 선택적으로 사용될 수 있다(예를 들어, 0 대신 1의 인덱스로 시작하는 것, 역으로의 또는 감소하는 규약, 임의의 맵핑 규약 등). 이러한 실시예들에서, 인덱스 혼자서는 싱글 대응 데이터 엘리먼트를 식별하기에 불충분할 수 있다.

[0051] 일부 실시예들에서, 인덱스들은 대응 데이터 엘리먼트들의 정렬된 순서를 나타낼 결과 패킹된 데이터에서의 위치들에 저장될 수 있다. 도시된 실시예에서, 인덱스들은 대응 데이터 엘리먼트들의 정렬된 증가하는 순서를 나타낼 결과 패킹된 데이터에서의 위치들에 저장된다. 다른 실시예들에서는, 감소하는 순서가 선택적으로 사용될 수 있다. 또한, 도시된 실시예에서, 인덱스들은 모든 데이터 엘리먼트들 중 순서화된 최소 서브세트- 예를 들어, 최소 1/2 -에 대해서만 저장된다. 대안적으로, 최대 서브세트(예를 들어, 최대 1/2) 또는 중간 서브세트 등의 다른 서브세트가 사용될 수 있다. 대안적으로, 소스 패킹된 데이터에서 모든 데이터 엘리먼트들에 대한 인덱스들이 선택적으로 저장될 수 있다.

[0052] 도 10을 다시 참조하면, 명령어에 응답하여 결과 마스크(1020)가 생성되어 제2 목적지 스토리지 위치에 저장될 수 있다. 목적지 스토리지 위치는 명령어에 의해 특정될 수 있거나 또는 다른 방식으로 나타날 수 있다. 일부 실시예들에서, 제2 목적지 스토리지 위치는 마스크 레지스터(예를 들어, 마스크 레지스터들(118) 중 하나)일 수 있다. 대안적으로, 결과 마스크는 범용 레지스터, 메모리 위치 또는 기타 스토리지 위치에 저장될 수 있다.

[0053] 일부 실시예들에서, 결과 마스크는 적어도 4개의 마스크 엘리먼트들을 포함한다. 일부 실시예들에서, 결과 마스크는 결과 패킹된 데이터에서의 인덱스들의 수와 동일한 수의 마스크 엘리먼트들을 포함할 수 있다. 각각의 마스크 엘리먼트는 결과 패킹된 데이터에서의 인덱스들 중 상이한 대응 인덱스에 대응할 수 있다. 각각의 마스크 엘리먼트는, 대응 인덱스에 의해 나타나는 데이터 엘리먼트 위치에서의 싱글 대응 데이터 엘리먼트가 제1 소스 패킹된 데이터에 위치되는지 아니면 그 대신 제2 소스 패킹된 데이터에 위치되는지를 나타낼 수 있다. 즉, 각각의 마스크 엘리먼트는 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터 중 하나를 식별할 수 있거나 또는 선택할 수 있고, 이에 의해 식별되거나 또는 선택된 소스 패킹된 데이터에서의 인덱스된 위치에서 대응 싱글 데이터 엘리먼트를 식별할 수 있다.

[0054] 일부 실시예들에서, 각각의 마스크 엘리먼트는 싱글 비트일 수 있지만, 2 이상의 비트들이 대안적으로 선택적으로 사용될 수 있다(예를 들어, 다수-비트 데이터 엘리먼트의 최상위 또는 최하위 비트). 도시된 예에 사용되는 하나의 가능한 규약에 따르면, 2진값 일(즉, 1)로 설정되는 싱글 비트는 해당 데이터 엘리먼트가 제1 소스 패킹된 데이터(1010)에 위치된다는 것을 나타내는 한편, 2진값 제로(즉, 0)로 클리어되는 비트는 해당 데이터 엘리먼트가 제2 소스 패킹된 데이터(1012)에 위치된다는 것을 나타낸다. 대안적으로, 그 반대의 규약이 선택적으로 사용될 수 있다. 더 설명하자면, 값이 -14인 데이터 엘리먼트는, 결과 패킹된 데이터의 최우측 위치에서 0의 대응 인덱스 및 결과 마스크에서의 최우측 위치에서 0의 마스크 엘리먼트 값에 의해 식별된다(제2 소스 패킹된 데이터를 나타냄). 유사하게, 값이 -4인 데이터 엘리먼트는, 결과 패킹된 데이터의 우측으로부터 3번째 위치에서 0의 대응 인덱스 및 결과 마스크의 우측으로부터 3번째 위치에서 1의 마스크 엘리먼트 값에 의해 식별된다(제1 소스 패킹된 데이터를 나타냄).

[0055] 일부 실시예들에서는, 상이한 소스 패킹된 데이터에서의 2개의 피연산자간(inter-operand) 데이터 엘리먼트들이 등가의 값들을 가지면, 소스 패킹된 데이터 중 하나로부터의 그 엘리먼트들이 보다 작은 값을 가졌던 것처럼 해석될 수 있는 규약이 채택될 수 있지만, 이것이 요구되는 것은 아니다. 일부 실시예들에서는, 동일한 소스 패킹된 데이터에서의 2개의 피연산자간 데이터 엘리먼트들이 등가의 값들을 가지면, 최하위 데이터 엘리먼트가 보다 작은 값을 가졌던 것처럼 해석될 수 있는 규약이 채택될 수 있지만, 이것이 요구되는 것은 아니다.

[0056] 추가의 패킹된 데이터 피연산자 선택 비트들(예를 들어, 마스크 엘리먼트들)을 (도 6-9에 도시된 접근방식에서와 같이) 인덱스들에 통합하는 대신에 결과 마스크에 포함시키는 것은, 특정 실시예들에서 이점들을 제공할 수 있다. 예를 들어, 프로세서가 결과 마스크를 프레디케이트(predicate) 피연산자로서 사용하여 패킹된 데이터 연산을 마스크할 수 있거나 또는 서술할 수 있는 때가 그러한 경우일 것이다. 일부 실시예들에서, 마스킹 또는 서술은 데이터 엘리먼트 그래뉼러티 마다(per-data element granularity) 있을 수 있어, 대응 데이터 엘리먼트들의 상이한 쌍들에 대한 연산들이 다른 것들에 개별적으로 및/또는 독립적으로 서술될 수 있거나 또는 조건부 제어될 수 있다. 결과 마스크에서의 마스크 엘리먼트들은 서술 엘리먼트들 또는 조건부 제어 엘리먼트들을 나타낼 수 있다. 일 양상에서, 마스크 엘리먼트들은 대응 소스 데이터 엘리먼트들 및/또는 대응 결과 엘리먼트들과 일-대-일 대응관계로 포함될 수 있다. 예를 들어, 각각의 마스크 엘리먼트 또는 비트의 값은 대응 연산이 수행될지 여부 및/또는 대응 결과 데이터 엘리먼트가 저장될지 여부를 제어할 수 있다. 각각의 마스크 엘리먼

트 또는 비트는, 이러한 연산이 소스 데이터 엘리먼트들의 대응 쌍에 대해 수행되는 것을 허용하고 대응 결과 데이터 엘리먼트가 목적지에 저장되는 것을 허용하도록 제1 값을 가질 수 있거나, 또는 이러한 연산이 소스 데이터 엘리먼트들의 대응 쌍에 대해 수행되는 것을 허용하지 않고 및/또는 대응 결과 데이터 엘리먼트가 목적지에 저장되는 것을 허용하지 않도록 제2의 상이한 값을 가질 수 있다. 하나의 가능한 규약에 따르면, 2진 제로(즉, 0)로 클리어된 마스크 비트는 마스크된 연산을 나타낼 수 있는 한편, 2진 일(즉, 1)로 설정된 마스크 비트는 마스크되지 않은 연산을 나타낼 수 있다.

[0057] 일부 실시예들에서는, 결과 마스크를 생성하는 명령어에 추가하여, 명령어 세트는, 또한, 대응 연산들이 수행될지 및/또는 대응 결과들이 저장될지 여부를 서술하거나, 조건부 제어하거나 또는 마스크하는데 사용되는 소스 서술 피연산자 또는 조건부 제어 피연산자로서 결과 마스크를 나타낼 수 있거나 또는 액세스할 수 있는 제2 명령어를 포함할 수 있다. 특정 구현들에서 소스 서술 피연산자로서 결과 마스크를 나타낼 수 있는 명령어의 하나의 특정 예는 VMOVDQA32 명령어이고, 이는 2013년 12월에 공개된 "Intel® Architecture Instruction Set Extensions Programming Reference, 319433-017"에 개시된다. VMOVDQA32 명령어는, 정렬된 패킹된 더블워드 정수 값들을, 서술용 소스 기입 마스크(writemask)를 사용하여, 소스 패킹된 데이터 피연산자로부터 결과 패킹된 데이터 피연산자로 이동시킬 수 있다. 결과 마스크는 VMOVDQA32 명령어에 의해 소스 기입 마스크로서 나타날 수 있다. 일부 실시예들에서 및/또는 일부 알고리즘들에 대해서, 결과 마스크는 이와 같이 서술된 명령어들에 의해 사용되어 전체 알고리즘의 관점으로부터 특정 성능 및/또는 효율 이점들을 제공할 수 있다. 일부 실시예들에서는, 결과 패킹된 데이터(1014)가 그 대신 추가 비트를 인덱스들에 통합할 수 있기에 충분한 비트들을 갖더라도, 결과 마스크가 사용될 수 있다.

[0058] 도 11은 데이터 엘리먼트들 중 최대 1/2에 대한 마스크를 갖는 2개 소스 정렬 인덱스들 명령어의 실시예에 응답하여 수행될 수 있는 데이터 엘리먼트들 중 최대 1/2에 대한 마스크를 갖는 2개 소스 정렬 인덱스들 연산(1150)의 일 실시예를 도시하는 블럭도이다. 명령어는, 적어도 4개의 데이터 엘리먼트의 제1 세트를 갖는 제1 소스 패킹된 데이터(1110)를 특정할 수 있거나 또는 다른 방식으로 나타낼 수 있고, 적어도 4개의 데이터 엘리먼트들의 제2 세트를 갖는 제2 소스 패킹된 데이터(1112)를 특정할 수 있거나 또는 다른 방식으로 나타낼 수 있다. 도시된 실시예에서, 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터 각각에서의 값들은 순서대로 정렬되지만, 이것이 요구되는 것은 아니다. 일부 실시예들에서는, 소스 데이터 엘리먼트들이 개별 패킹된 데이터 내에서 정렬된 순서로 될 것이 추정될 수 있거나(예를 들어, 명령어에 대해 암시적으로 추정됨) 또는 요구될 수 있다. 도시된 예에서는, 피연산자들에서 비트 중요성이 증가할수록 증가하는 순서가 사용되지만, 다른 실시예에서는 감소하는 순서가 선택적으로 사용될 수 있다. 다른 실시예들에서는, 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터 각각 내의 데이터 엘리먼트들이 개별 패킹된 데이터 내에서 정렬된 순서로 존재할 것이 추정되지 않을 수 있거나 또는 요구되지 않을 수 있다(예를 들어, 엘리먼트들이 정렬되지 않은 순서로 되어 있을 때 명령어가 정확하게 동작할 수 있음).

[0059] 결과 패킹된 데이터(1114)는 명령어에 응답하여 (예를 들어, 실행 유닛(106)에 의해) 생성되어 목적지 스토리지 위치에 저장될 수 있다. 일부 실시예들에서 결과 패킹된 데이터는 적어도 4개의 인덱스들을 포함할 수 있다. 일부 실시예들에서, 각각의 인덱스는 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서 한 쌍의 대응 데이터 엘리먼트 위치들을 가리키거나, 식별하거나 또는 이와 관련될 수 있다. 마찬가지로, 결과 마스크(1120)는 명령어에 응답하여 저장될 수 있다. 결과 마스크는 인덱스들에 대응하는 마스크 엘리먼트들을 가질 수 있다.

[0060] 도 11의 실시예에서, 인덱스들은 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서의 모든 데이터 엘리먼트들 중 순서화된 최대 서브세트- 예를 들어, 도시된 실시예에서는 최대 1/2 -에 대응할 수 있다. 예를 들어, 8개의 인덱스들은 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서 대응 8개의 최대 데이터 엘리먼트들(예를 들어, 12, 12, 16, 18, 24, 43, 55 및 60)의 정렬된 순서를 나타낼 결과 패킹된 데이터에서의 위치들에 저장될 수 있다. 마찬가지로, 결과 마스크의 마스크 엘리먼트들은 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서의 모든 엘리먼트들 중 순서화된 최대 서브세트- 예를 들어, 도시된 실시예에서는 최대 1/2 -에 대응할 수 있다.

[0061] 도시된 실시예에서, 8개의 인덱스들은 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서 대응 8개의 최대 데이터 엘리먼트들의 정렬된 증가하는 순서를 나타낼 결과 패킹된 데이터에서의 위치들에 저장된다. 다른 실시예들에서는, 감소하는 순서 또는 역인 순서가 선택적으로 사용될 수 있다.

[0062] 도 12는 데이터 엘리먼트들 중 최소 1/2에 대해 마스크를 갖는 2개 정렬되지 않은 소스 정렬 인덱스들 명령어의 일 실시예에 응답하여 수행될 수 있는 데이터 엘리먼트들 중 최소 1/2에 대한 마스크를 갖는 2개 정렬되지 않은

소스 정렬 인덱스들 연산(1252)의 일 실시예를 도시하는 블럭도이다. 명령어는, 적어도 4개의 데이터 엘리먼트의 제1 세트를 갖는 제1 소스 패킹된 데이터(1210)를 특정할 수 있거나 또는 다른 방식으로 나타낼 수 있고, 적어도 4개의 데이터 엘리먼트들의 제2 세트를 갖는 제2 소스 패킹된 데이터(1212)를 특정할 수 있거나 또는 다른 방식으로 나타낼 수 있다. 도 12의 연산에서, 제1 소스 패킹된 데이터 피연산자에서의 데이터 엘리먼트들의 값들은 정렬된 순서로 존재하지 않는다. 유사하게, 제2 소스 패킹된 데이터 피연산자에서의 데이터 엘리먼트들의 값들은 정렬된 순서로 존재하지 않는다. 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터 각각 내의 데이터 엘리먼트들이 개별 패킹된 데이터 피연산자 내에서 정렬된 순서로 존재할 것이 추정되지 않을 수 있거나 또는 요구되지 않을 수 있다(예를 들어, 엘리먼트들이 소스 패킹된 데이터 피연사자들에서 정렬되지 않은 순서로 되어 있을 때 명령어가 정확하게 동작할 수 있음).

[0063] 결과 패킹된 데이터(1214)는 명령어에 응답하여 (예를 들어, 실행 유닛(106)에 의해) 생성되어 목적지 스토리지 위치에 저장될 수 있다. 일부 실시예들에서 결과 패킹된 데이터는 적어도 4개의 인덱스들을 포함할 수 있다. 일부 실시예들에서, 각각의 인덱스는 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서 한 쌍의 대응 데이터 엘리먼트 위치들을 가리키거나, 식별하거나 또는 이와 관련될 수 있다. 마찬가지로, 결과 마스크(1220)는 명령어에 응답하여 저장될 수 있다. 결과 마스크는 인덱스들에 대응하는 마스크 엘리먼트들을 가질 수 있다.

[0064] 도시된 실시예에서, 인덱스들 및 마스크 엘리먼트들은 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서의 모든 데이터 엘리먼트들 중 최소 서브세트- 본 경우 최소 1/2 -에 대응한다. 다른 실시예에서, 인덱스들 및 마스크 엘리먼트들은 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서의 모든 데이터 엘리먼트들 중 최대 서브세트- 예를 들어, 최대 1/2 -에 대응할 수 있다.

[0065] 도시된 실시예에서, 8개의 인덱스들은 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서 대응 8개의 최대 데이터 엘리먼트들의 정렬된 증가하는 순서를 나타낼 결과 패킹된 데이터에서의 위치들에 저장된다. 다른 실시예들에서는, 감소하는 순서 또는 역인 순서가 선택적으로 사용될 수 있다.

[0066] 도 13은 데이터 엘리먼트들 중 최소 1/2에 대한 마스크를 갖는 2개 소스 정렬 인덱스들 및 정렬 데이터 명령어의 실시예에 응답하여 수행될 수 있는 데이터 엘리먼트들 중 최소 1/2에 대한 마스크를 갖는 2개 소스 정렬 인덱스들 및 정렬 데이터 연산(1354)의 일 실시예를 도시하는 블럭도이다. 명령어는, 적어도 4개의 데이터 엘리먼트들의 제1 세트를 갖는 제1 소스 패킹된 데이터(1310)를 특정할 수 있거나 또는 다른 방식으로 나타낼 수 있고, 적어도 4개의 데이터 엘리먼트들의 제2 세트를 갖는 제2 소스 패킹된 데이터(1312)를 특정할 수 있거나 또는 다른 방식으로 나타낼 수 있다. 도시된 실시예에서, 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터 각각 내에서의 값들은 순서대로 정렬되지만, 이것이 요구되는 것은 아니다. 일부 실시예들에서는, 소스 데이터 엘리먼트들이 개별 패킹된 데이터 내에서 정렬된 순서로 존재하는 것이 추정될 수 있거나(예를 들어, 명령어에 대해 암시적으로 추정됨) 또는 요구될 수 있다. 도시된 예에서는, 피연산자들에서 비트 중요성이 증가할수록 증가하는 순서가 사용되지만, 다른 실시예에서는 감소하는 순서가 선택적으로 사용될 수 있다. 다른 실시예들에서는, 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터 각각 내의 데이터 엘리먼트들이 개별 패킹된 데이터 내에서 정렬된 순서로 존재할 것이 추정되지 않을 수 있거나 또는 요구되지 않을 수 있다(예를 들어, 엘리먼트들이 정렬되지 않은 순서로 되어 있을 때 명령어가 정확하게 동작할 수 있음).

[0067] 제1 결과 패킹된 데이터(1314)가 명령어에 응답하여 (예를 들어, 실행 유닛(106)에 의해) 생성되어 목적지 스토리지 위치에 저장될 수 있다. 일부 실시예들에서 결과 패킹된 데이터는 적어도 4개의 인덱스들을 포함할 수 있다. 일부 실시예들에서, 각각의 인덱스는 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서 한 쌍의 대응 데이터 엘리먼트 위치들을 가리키거나, 식별하거나 또는 이와 관련될 수 있다. 마찬가지로, 결과 마스크(1320)는 명령어에 응답하여 저장될 수 있다. 결과 마스크는 인덱스들에 대응하는 마스크 엘리먼트들을 가질 수 있다.

[0068] 본 실시예에서, 제2 결과 패킹된 데이터(1316) 또한 명령어에 응답하여 생성되어 제2 목적지 스토리지 위치에 저장될 수 있다. 제2 목적지 스토리지 위치는 명령어에 의해 특정될 수 있거나 또는 다른 방식으로 나타날 수 있고, 패킹된 데이터 레지스터, 메모리 위치 또는 기타 스토리지 위치일 수 있다. 제2 목적지 스토리지 위치는 제1 또는 제2 소스 패킹된 데이터 중 하나에 대해 사용된 것과 동일한 스토리지 위치일 수 있거나, 또는 상이한 스토리지 위치일 수 있다. 제2 결과 패킹된 데이터는, 저장된 인덱스들 및 마스크 엘리먼트들에 대응하며, 정렬된 순서를 반영하는 제2 결과 패킹된 데이터의 위치들에 저장되는, 대응 데이터 엘리먼트들을 포함할 수 있다.

[0069] 도시된 실시예에서, 정렬된 데이터 엘리먼트들, 인덱스들 및 마스크 엘리먼트들은 제1 소스 패킹된 데이터 및

제2 소스 패킹된 데이터에서의 모든 데이터 엘리먼트들 중 최소 서브세트- 본 경우 최소 1/2 -에 대응한다. 다른 실시예에서, 정렬된 데이터 엘리먼트들, 인덱스들 및 마스크 엘리먼트들은 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서의 모든 데이터 엘리먼트들 중 최대 서브세트- 예를 들어, 최대 1/2 -에 대응할 수 있다.

[0070] 도시된 실시예에서, 정렬된 데이터 엘리먼트들, 인덱스들 및 마스크 엘리먼트들은 증가하는 순서를 나타내는 위치들에 저장된다. 다른 실시예들에서는, 감소하는 순서 또는 역인 순서가 선택적으로 사용될 수 있다.

[0071] 도 10에서 결과로서 정렬된 인덱스들이 저장된다. 도 13에서 결과로서 정렬된 인덱스들 및 정렬된 데이터 엘리먼트들 양자 모두가 저장된다. 다른 실시예에서는, 인덱스들이 아니라 정렬 데이터 엘리먼트들이 명령어에 응답하여 선택적으로 저장될 수 있다.

[0072] 도 11-13의 연산들은 도 10의 연산과 어느 정도 유사성들을 갖는다. 설명을 모호하게 하는 것을 회피하기 위해서, 도 10의 연산에 대해 선택적으로 유사하거나 또는 공통인 특징들 및 상세들 모두를 반복하지는 않고, 도 11-13의 연산들에 대한 상이한 및/또는 추가적인 특징들이 주로 개시되었다. 그러나, 도 10의 연산의 이미 개시된 특징들 및 상세들 또한, 달리 또는 다른 방식으로 명명 백백히 다르게 언급되지 않는 한, 도 11-13의 연산들 중 임의의 것에 선택적으로 적용될 수 있다는 점이 이해되어야 한다.

[0073] 명령어 세트는 본 명세서에 개시되는 명령어들 중 하나 이상을 포함할 수 있다. 예를 들어, 일부 실시예들에서, 명령어 세트는 정렬된 결과를 생성할 수 있는 (예를 들어, 도 3-4 중 하나에 대해 도시되거나 또는 개시된 바와 같은) 제1 명령어 및 (예를 들어, 도 6, 7, 9, 10, 11 및 13 중 하나에 대해 도시되거나 개시된 바와 같이) 정렬된 소스 패킹된 데이터를 추정하거나 또는 요구하는 제2 명령어를 선택적으로 포함할 수 있다. 다른 예로서, 일부 실시예들에서, 명령어 세트는 (예를 들어, 도 6 및 10 중 하나에 대해 도시되거나 개시된 바와 같이) 모든 소스 데이터 엘리먼트들 중 최소 1/2에 대한 정렬을 위한 제1 명령어 및 (예를 들어, 도 7 및 11 중 하나에 도시되거나 개시된 바와 같이) 모든 데이터 엘리먼트들 중 최대 1/2에 대한 정렬을 위한 제2 명령어를 선택적으로 포함할 수 있다. 대안적으로, 명령어 세트는 본 명세서에 도시되고 개시된 바와 같은 명령어들 중 하나만을 포함할 수 있다.

[0074] 이하의 코드 토막(snippet)은 본 명세서에 개시되는 바와 같은 명령어들의 예들을 사용하여 32개 정수들을 정렬하는 알고리즘의 예시적인 실시예를 나타낸다. sortassistd 명령어는 도 3에 도시된 것과 유사한 연산으로 더블워드 엘리먼트들을 오름차순으로 정렬한다. sortedmergedassistl 및 sortedmergedassisth 명령어는 도 10-11의 연산들과 유사하게 이미 정렬된 엘리먼트들에 대한 정렬된 인덱스들을 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에 저장하는 2개 소스 명령어들이다.

[0075] sort32:

[0076] vmovdqu32 (%rdi), %zmm0

[0077] vmovdqu32 64(%rdi), %zmm1

[0078] sortassistd %zmm0, %zmm2

[0079] sortassistd %zmm1, %zmm3

[0080] vpermd %zmm0, %zmm2, %zmm0

[0081] vpermd %zmm1, %zmm3, %zmm1

[0082] sortedmergedassistl %zmm1, %zmm0, %zmm2, %k1

[0083] sortedmergedassisth %zmm1, %zmm0, %zmm3, %k2

[0084] vpermd %zmm0, %zmm2, %zmm4

[0085] vpermd %zmm1, %zmm2, %zmm5

[0086] vpermd %zmm0, %zmm3, %zmm6

[0087] vpermd %zmm1, %zmm3, %zmm7

[0088] vmovdqu32 %zmm5, %zmm4, {%k1}

[0089] vmovdqu32 %zmm7, %zmm6, {%k2}

[0090] vmovdqu32 %zmm4, {%rdi}

[0091] vmovdqu32 %zmm6, 64{%rdi}

[0092] ret

[0093] 일부 실시예들에서, 명령어 포맷은 연산 코드 또는 오피코드를 포함할 수 있다. 오피코드는 수행될 명령어 및/또는 연산(예를 들어, 정렬 인덱스 연산)을 식별하도록 동작될 수 있는 복수의 비트들 또는 하나 이상의 필드들을 나타낼 수 있다. 특정 명령어에 따라서, 명령어 포맷은 또한 하나 이상의 소스 및/또는 목적지 특정자들을 선택적으로 포함할 수 있다. 예를 들어, 이러한 특정자들 각각은 레지스터의 어드레스, 메모리 위치 또는 기타 스토리지 위치를 특정하는 비트들 또는 하나 이상의 필드들을 포함할 수 있다. 대안적으로, 이러한 명시적인 특정자 대신, 하나 이상의 소스들 및/또는 목적지들은 명백히 특정되는 대신 명령어에 선택적으로 내재될 수 있다. 또한, 그것은 소스가 목적지로서 재사용되는 것을 선택적으로 암시할 수 있다. 또한, 명령어 포맷은, 선택적으로 추가적 필드들을 추가할 수 있고, 특정 필드들을 중복할 수 있는 등이다. 필드들은 연속적인 시퀀스들의 비트들을 포함할 필요는 없지만 오히려 비연속적이거나 또는 분리된 피트들로 구성될 수 있다. 일부 실시예들에서, 명령어 포맷은 VEX 또는 EVEX 인코딩 또는 명령어 포맷을 따를 수 있지만, 본 발명의 범위가 이에 제한되는 것은 아니다.

[0094] 도 14는 패킹된 데이터 레지스터들(1408)의 적합한 세트의 예시적인 실시예의 블럭도이다. 패킹된 데이터 레지스터들은 ZMM0 내지 ZMM31로 레이블되는 32개의 512 비트 패킹된 데이터 레지스터들을 포함한다. 도시된 실시예에서, 하위 16개 레지스터들, 즉 ZMM0-ZMM15의 하위 256 비트들은 YMM0-YMM15로 레이블되는 개별 256 비트 패킹된 데이터 레지스터들에 대해 에일리어싱되거나(aliased) 또는 오버레이되지만(overlaid), 이러한 것이 요구되는 것은 아니다. 마찬가지로, 도시된 실시예에서, 레지스터들 YMM0-YMM15의 하위 128 비트는 XMM0-XMM15로 레이블된 개별 128 비트 패킹된 데이터 레지스터들에 대해 에일리어싱되거나 또는 중첩되지만, 이것 또한 요구되는 것은 아니다. 512 비트 레지스터들 ZMM0 내지 ZMM31은 512 비트 패킹된 데이터, 256 비트 패킹된 데이터 또는 128 비트 패킹된 데이터를 유지하도록 동작될 수 있다. 256 비트 레지스터들 YMM0-YMM15는 256 비트 패킹된 데이터 또는 128 비트 패킹된 데이터를 유지하도록 동작될 수 있다. 128 비트 레지스터들 XMM0-XMM15는 128 비트 패킹된 데이터를 유지하도록 동작될 수 있다. 일부 실시예들에서, 레지스터들 각각은 부동 소수점 데이터 또는 패킹된 정수 데이터를 저장하는데 사용될 수 있다. 적어도 8 비트 바이트 데이터, 16 비트 워드 데이터, 32 비트 더블워드, 32 비트 단일 정밀도 부동 소수점 데이터, 64 비트 쿼드워드 및 64 비트 2배 정밀도 부동 소수점 데이터를 포함하는 상이한 데이터 엘리먼트 사이즈들이 지원된다. 대안적인 실시예들에서는, 상이한 개수들의 레지스터들 및/또는 상이한 사이즈들의 레지스터들이 사용될 수 있다. 또 다른 실시예들에서, 레지스터들은 보다 작은 레지스터들에 대한 보다 큰 레지스터들의 에일리어싱을 사용하거나 사용하지 않을 수 있고 및/또는 부동 소수점 데이터를 저장하는데 사용되거나 또는 사용되지 않을 수 있다.

[0095] 명령어 세트는 하나 이상의 명령어 포맷들을 포함한다. 주어진 명령어 포맷은, 다양한 필드들(비트들의 수, 비트들의 위치)을 정의하여, 다른 것들 중에서, 수행될 연산(오피코드) 및 그 연산이 수행될 피연산자(들)를 특정한다. 일부 명령어 포맷들은 명령어 템플릿들(또는 서브포맷들)의 정의를 통해 더욱 분할된다. 예를 들어, 주어진 명령어 포맷의 명령어 템플릿들은, 명령어 포맷의 필드들의 상이한 서브세트들을 갖는 것으로 정의될 수 있고(포함되는 필드들이 통상적으로는 동일한 순서로 존재하지만, 포함되는 필드들이 보다 적기 때문에 적어도 일부는 상이한 비트 위치들을 갖는다) 및/또는 상이하게 해석되는 주어진 필드를 갖는 것으로 정의될 수 있다. 따라서, ISA의 각 명령어는, 주어진 명령어 포맷(그리고, 정의되면, 그 명령어 포맷의 명령어 템플릿들 중 주어진 하나에 존재함)을 사용하여 표현되고, 연산 및 피연산자들을 특정하는 필드들을 포함한다. 예를 들어, 예시적인 ADD 명령어는, 특정 오피코드, 및 그 오피코드를 특정하는 오피코드 필드 및 피연산자들(source1/destination 및 source2)을 선택하는 피연산자 필드들을 포함하는 명령어 포맷을 갖고; 명령어 스트림에서 이러한 ADD 명령어의 출현(occurrence)은 특정 피연산자들을 선택하는 피연산자 필드들에 특정 콘텐츠를 가질 것이다. AVX(Advanced Vector Extensions)(AVX1 및 AVX2)라 하고 VEX(Vector Extensions) 코딩 스킴을 사용하는 SIMD 확장들의 세트가 발표 및/또는 공개되었다(예를 들어, "Intel® 64 and IA-32 Architectures Software Developers Manual, October 2011" 참조; 및 "Intel® Advanced Vector Extensions Programming Reference, June 2011" 참조).

[0096] **예시적인 명령어 포맷들**

[0097] 본 명세서에 개시되는 명령어(들)의 실시예들은 상이한 포맷들로 구현될 수 있다. 또한, 예시적인 시스템들, 아키텍처들, 및 파이프라인들이 이하 상세히 설명된다. 명령어(들)의 실시예들은 이러한 시스템들, 아키텍처들 및 파이프라인들에서 실행될 수 있지만, 상세히 설명되는 것들에 제한되는 것은 아니다.

[0098] **일반 벡터 친화형 명령어 포맷**

[0099] 벡터 친화형 명령어 포맷은 벡터 명령어들에 대해 적절한 명령어 포맷이다(예를 들어, 벡터 연산들에 특정한 특정 필드들이 존재한다). 벡터 및 스칼라 연산들 양자 모두가 벡터 친화형 명령어 포맷을 통해 지원되는 실시예들이 설명되지만, 대안적인 실시예들은 벡터 친화형 명령어 포맷을 통한 벡터 연산들만을 이용한다.

[0100] 도 15a-15b는 본 발명의 실시예들에 따른, 일반 벡터 친화형 명령어 포맷 및 그의 명령어 템플릿들을 도시하는 블럭도들이다. 도 15a는 본 발명의 실시예들에 따른, 일반 벡터 친화형 명령어 포맷 및 그의 클래스 A 명령어 템플릿들을 도시하는 블럭도인 반면; 도 15b는 본 발명의 실시예에 따른, 일반 벡터 친화형 명령어 포맷 및 그의 클래스 B 명령어 템플릿들을 도시하는 블럭도이다. 구체적으로, 일반 벡터 친화형 명령어 포맷(1500)에 대해 클래스 A 및 클래스 B 명령어 템플릿들이 정의되고, 양자 모두 메모리 액세스 없음(1505) 명령어 템플릿들 및 메모리 액세스(1520) 명령어 템플릿들을 포함한다. 벡터 친화형 명령어 포맷의 맥락에서 일반적이라는 용어는 임의의 특정 명령어 세트에 얽매이지 않는 명령어 포맷을 말한다.

[0101] 벡터 친화형 명령어 포맷이 이하의 것들을 지원하는 본 발명의 실시예들이 개시될 것이지만: 데이터 엘리먼트 폭들(또는 사이즈들)이 32 비트(4 바이트) 또는 64 비트(8 바이트)인 64 바이트 벡터 피연산자 길이(또는 사이즈)(그에 따라, 64 바이트 벡터는 16개의 더블워드-사이즈 엘리먼트들 또는 대안적으로 8개의 쿼드워드-사이즈 엘리먼트들로 구성됨); 데이터 엘리먼트 폭들(또는 사이즈들)이 16 비트(2 바이트) 또는 8 비트(1 바이트)인 64 바이트 벡터 피연산자 길이(또는 사이즈); 데이터 엘리먼트 폭들(또는 사이즈들)이 32 비트(4 바이트), 64 비트(8 바이트), 16 비트(2 바이트), 또는 8 비트(1 바이트)인 32 바이트 벡터 피연산자 길이(또는 사이즈); 및 데이터 엘리먼트 폭들(또는 사이즈들)이 32 비트(4 바이트), 64 비트(8 바이트), 16 비트(2 바이트) 또는 8 비트(1 바이트)인 16 바이트 벡터 피연산자 길이(또는 사이즈), 대안적인 실시예들은, 더 크거나, 더 적거나 또는 상이한 데이터 엘리먼트 폭들(예를 들어, 128 비트 (16 바이트)을 갖는 더 크거나, 더 적거나, 및/또는 상이한 벡터 피연산자 사이즈들(예를 들어, 256 바이트 벡터 피연산자들)을 지원할 수 있다.

[0102] 도 15a의 클래스 A 명령어 템플릿들은 이하를 포함한다: 1) 메모리 액세스 없음(1505) 명령어 템플릿들 내에, 메모리 액세스 없음, 전체 라운드(full round) 제어 타입 연산(1510) 명령어 템플릿, 및 메모리 액세스 없음, 데이터 변환 타입 연산(1515) 명령어 템플릿이 도시되고; 및 2) 메모리 액세스(1520) 명령어 템플릿들 내에, 메모리 액세스, 임시(1525) 명령어 템플릿, 및 메모리 액세스, 비-임시(1530) 명령어 템플릿이 도시된다. 도 15b의 클래스 B 명령어 템플릿들은 이하를 포함한다: 1) 메모리 액세스 없음(1505) 명령어 템플릿들 내에, 메모리 액세스 없음, 기입 마스크 제어, 부분 라운드(partial round) 제어 타입 연산(1512) 명령어 템플릿, 및 메모리 액세스 없음, 기입 마스크 제어, vsize 타입 연산(1517) 명령어 템플릿이 도시되고; 및 2) 메모리 액세스(1520) 명령어 템플릿들 내에, 메모리 액세스, 기입 마스크 제어(1527) 명령어 템플릿이 도시된다.

[0103] 일반 벡터 친화형 명령어 포맷(1500)은 도 15a-15b에 도시되는 순서대로 아래에 열거되는 다음과 같은 필드들을 포함한다.

[0104] 포맷 필드(1540) - 이 필드 내의 특정 값(명령어 포맷 식별자 값)은, 벡터 친화형 명령어 포맷, 및 이에 따른 명령어 스트림들 내에서의 벡터 친화형 명령어 포맷의 명령어들의 출현을 고유하게 식별한다. 이와 같이, 이 필드는 오직 일반 벡터 친화형 명령어 포맷을 갖는 명령어 세트에 대해 필요하지 않다는 점에서 선택적이다.

[0105] 베이스 연산 필드(1542) - 그 내용은 상이한 베이스 연산들을 구분한다.

[0106] 레지스터 인덱스 필드(1544) - 그 내용은, 직접 또는 어드레스 생성을 통해, 레지스터들 내에 있는지 또는 메모리 내에 있는지, 소스 및 목적지 피연산자들의 위치들을 특정한다. 이들은 PxQ(예를 들어, 32x512, 16x128, 32x1024, 64x1024) 레지스터 파일로부터 N개의 레지스터들을 선택하기에 충분한 수의 비트들을 포함한다. 일 실시예에서, N은 최대 3개의 소스들 및 1개의 목적지 레지스터일 수 있지만, 대안적인 실시예들은 더 많거나 더 적은 소스들 및 목적지 레지스터들을 지원할 수 있다(예를 들어, 소스들 중 하나가 목적지로도 작용하는 최대 2개의 소스를 지원할 수 있고, 소스들 중 하나가 목적지로도 작용하는 최대 3개의 소스를 지원할 수 있고, 최대 2개의 소스 및 1개의 목적지를 지원할 수 있다).

[0107] 수정자 필드(1546) - 그 내용은 메모리 액세스를 특정하는 일반 벡터 명령어 포맷의 명령어들의 출현을 그렇지

않는 명령어들과 구분한다; 즉, 메모리 액세스 없음(1505) 명령어 템플릿들과 메모리 액세스(1520) 명령어 템플릿들 사이를 구분한다. 메모리 액세스 연산들은 (일부 경우에 레지스터들 내의 값들을 이용하여 소스 및/또는 목적지 어드레스들을 특정하는) 메모리 계층구조에 대해 판독 및/또는 기입하고 한편, 메모리 액세스 없음 연산들은 하지 않는다(예를 들어, 소스 및 목적지들은 레지스터들이다). 일 실시예에서, 이 필드는 또한 메모리 어드레스 계산들을 수행하는 3개의 상이한 방식들 사이에서 선택하지만, 대안적인 실시예들은 메모리 어드레스 계산들을 수행하는 더 많거나, 더 적거나, 상이한 방식을 지원할 수 있다.

[0108] 증대 연산 필드(1550) - 그 내용은 각종의 상이한 연산들 중 어느 것이 베이스 연산에 부가하여 수행되어야 하는지를 구분해준다. 이 필드는 맥락 특정(context specific)이다. 본 발명의 일 실시예에서, 이 필드는 클래스 필드(1568), 알파 필드(1552), 및 베타 필드(1554)로 분할된다. 증대 연산 필드(1550)는 연산들의 공통 그룹들이 2, 3, 또는 4개의 명령어들 보다는 싱글 명령어에서 수행되는 것을 허용한다.

[0109] 스케일 필드(1560) - 그 내용은 메모리 어드레스 생성을 위한(예를 들어, 2^{scale} * index + base를 이용하는 어드레스 생성을 위한) 인덱스 필드의 내용의 스케일링(scaling)을 허용한다.

[0110] 변위 필드(1562A) - 그 내용은 메모리 어드레스 생성의 부분으로서 (예를 들어, 2^{scale} * index + base + displacement)를 이용하는 어드레스 생성을 위해) 사용된다.

[0111] 변위 인자 필드(Displacement Factor Field)(1562B)(변위 인자 필드(1562B) 바로 위의 변위 필드(1562A)의 병치(juxtaposition)는 하나 또는 다른 것이 이용됨을 나타낸다는 것에 주목하자) - 그 내용은 어드레스 생성의 부분으로서 이용되고; 이는 메모리 액세스의 사이즈(N)에 의해 스케일링될 변위 인자를 특정한다- 여기서 N은 (예를 들어, 2^{scale} * index + base + scaled displacement를 이용하는 어드레스 생성을 위한) 메모리 액세스에서의 바이트들의 수이다. 잉여 하위 비트들(Redundant low-order bits)은 무시되고, 따라서, 변위 인자 필드의 내용은 유효 어드레스를 계산하는 데 이용될 최종 변위를 생성하기 위하여 메모리 피연산자 총 사이즈(N)로 곱해진다. N의 값은 전체 오피코드 필드(1574)(본 명세서에서 나중에 설명됨) 및 데이터 조작 필드(1554C)에 기초하여 런타임에 프로세서 하드웨어에 의해 결정된다. 변위 필드(1562A) 및 변위 인자 필드(1562B)는, 메모리 액세스 없음 (1505) 명령어 템플릿들에 대해 사용되지 않고 및/또는 상이한 실시예들이 둘 중 하나만을 구현하거나 어느 것도 구현하지 않을 수 있다는 점에서, 선택적이다.

[0112] 데이터 엘리먼트 폭 필드(1564) - 그 내용은 (일부 실시예들에서는, 모든 명령어들에 대해; 다른 실시예들에서는, 명령어들 중 일부에 대해서만) 다수의 데이터 엘리먼트 폭 중 어느 것이 사용되어야 하는지를 구분해준다. 이 필드는 오직 하나의 데이터 엘리먼트 폭이 지원되고 및/또는 데이터 엘리먼트 폭들이 오피코드들의 일부 양상을 이용하여 지원되는 경우에 필요하지 않다는 점에서 선택적이다.

[0113] 기입 마스크 필드(1570) - 그 내용은, 데이터 엘리먼트 위치별 기반으로, 목적지 벡터 피연산자 내의 그 데이터 엘리먼트 위치가 베이스 연산과 증대 연산의 결과를 반영하는지를 제어한다. 클래스 A 명령어 템플릿들은 병합-기입마스킹(merging-writemasking)을 지원하는 한편, 클래스 B 명령어 템플릿들은 병합- 및 제로화-기입마스킹 양자 모두를 지원한다. 병합할 때, 벡터 마스크들은 목적지의 임의의 세트의 엘리먼트들이 (베이스 연산 및 증대 연산에 의해 특정된) 임의의 연산의 실행 중에 업데이트들로부터 보호될 수 있게 하고; 다른 일 실시예에서, 대응하는 마스크 비트가 0을 갖는 목적지의 각 요소의 이전의 값을 보존할 수 있게 한다. 대조적으로, 제로화할 때, 벡터 마스크들은 목적지의 임의의 세트의 엘리먼트들이 (베이스 연산 및 증대 연산에 의해 특정된) 임의의 연산의 실행 중에 제로화될 수 있게 하고; 일 실시예에서, 목적지의 엘리먼트는 대응하는 마스크 비트가 0 값을 가질 때 0으로 설정된다. 이러한 기능성의 서브세트는 수행되는 연산의 벡터 길이를 제어하는 능력이지만 (즉, 요소들의 스팬(span)은 첫번째 것에서 마지막 것까지 수정된다), 수정되는 엘리먼트들이 연속적인 것은 필요하지 않다. 그러므로, 기입 마스크 필드(1570)는, 로드들, 저장들, 산술, 논리 등을 포함하는 부분 벡터 연산들을 허용한다. 기입 마스크 필드(1570)의 내용이 사용될 기입 마스크를 포함하는 다수의 기입 마스크 레지스터들 중 하나를 선택하는 본 발명의 실시예들이 개시되지만(따라서, 기입 마스크 필드(1570)의 내용은 수행될 마스킹을 간접적으로 식별함), 대안적인 실시예들은 그 대신에 또는 추가적으로 마스크 기입 필드(1570) 내용이 수행될 마스킹을 직접 특정하는 것을 허용한다.

[0114] 즉치(Immediate) 필드(1572) - 그 내용은 즉치의 지정을 가능하게 해준다. 이 필드는 즉치를 지원하지 않는 일반 벡터 친화형 포맷의 구현에 존재하지 않고, 즉치를 사용하지 않는 명령어들에 존재하지 않는다는 점에서 선택적이다.

[0115] 클래스 필드(1568) - 그 내용은 명령어들의 상이한 클래스들 사이를 구별한다. 도 15a-b를 참조하면, 이 필드의 내용들은 클래스 A 및 클래스 B 명령어들 사이를 선택한다. 도 15a-b에서는, 필드에 특정 값이 존재한다는 것을 나타내기 위해 둥근 코너 사각형이 이용된다(예를 들어, 도 15a-b에서 각각 클래스 필드(1568)에 대한 클래스 A(1568A) 및 클래스 B(1568B)).

[0116] **클래스 A의 명령어 템플릿들**

[0117] 클래스 A의 메모리 액세스 없음(1505) 명령어 템플릿들의 경우, 알파 필드(1552)는 RS 필드(1552A)로서 해석되고, 그 내용은 상이한 증대 연산 타입들 중 어느 것이 수행될 것인지를 구분하는(예를 들어, 라운드(1552A.1) 및 데이터 변환(1552A.2)은 각각 메모리 액세스 없음, 라운드 타입 연산(1510) 및 메모리 액세스 없음, 데이터 변환 타입 연산(1515) 명령어 템플릿들에 대해 특정된다) 반면, 베타 필드(1554)는 특정된 타입의 연산들 중 어느 것이 수행될 것인지를 구분한다. 메모리 액세스 없음(1505) 명령어 템플릿들에서, 스케일 필드(1560), 변위 필드(1562A) 및 변위 스케일 필드(1562B)는 존재하지 않는다.

[0118] 메모리 액세스 없음 명령어 템플릿들-전체 라운드 제어 타입 연산

[0119] 메모리 액세스 없음 전체 라운드 제어 타입 연산(1510) 명령어 템플릿에서, 베타 필드(1554)는 라운드 제어 필드(1554A)로서 해석되고, 그 내용(들)은 스태틱 라운딩을 제공한다. 본 발명의 개시된 실시예에서, 라운드 제어 필드(1554A)는 모든 부동 소수점 예외 억제(SAE; suppress all floating point exceptions) 필드(1556) 및 라운드 연산 제어 필드(1558)를 포함하지만, 대안적인 실시예들은 이들 개념들 양자 모두를 동일한 필드로 인코딩하거나 이들 개념들/필드들 중 하나 또는 다른 하나만을 가질 수 있다(예를 들어, 라운드 연산 제어 필드(1558)만을 가질 수 있다).

[0120] SAE 필드(1556) - 그 내용은 예외 이벤트 보고를 디스에이블할 것인지 여부를 구별하고; SAE 필드(1556)의 내용이 억제가 인에이블됨을 나타낼 때, 주어진 명령어는 어떠한 종류의 부동 소수점 예외 플래그도 보고하지 않고, 어떠한 부동 소수점 예외 핸들러도 발생시키지 않는다.

[0121] 라운드 연산 제어 필드(1558) - 그 내용은 한 그룹의 라운딩 연산들 중 어느 것을 수행할 것인지를 구분한다(예를 들어, 라운드-업, 라운드-다운, 제로를 향한 라운드 및 최근접치로의 라운드). 따라서, 라운드 연산 제어 필드(1558)는 명령어당 기초로 라운딩 모드의 변경을 허용한다. 프로세서가 라운딩 모드들을 특정하기 위한 제어 레지스터를 포함하는 본 발명의 일 실시예에서, 라운드 연산 제어 필드(1558)의 내용은 그 레지스터 값을 무효로 한다.

[0122] 메모리 액세스 없음 명령어 템플릿들-데이터 변환 타입 연산

[0123] 메모리 액세스 없음 데이터 변환 타입 연산(1515) 명령어 템플릿에서, 베타 필드(1554)는 데이터 변환 필드(1554B)로서 해석되고, 그 내용은 다수의 데이터 변환들 중 어느 것이 수행될 것인지(예를 들어, 데이터 변환 없음, 스위즐(swizzle), 브로드캐스트)를 구별한다.

[0124] 클래스 A의 메모리 액세스(1520) 명령어 템플릿의 경우, 알파 필드(1552)는 되찾기 힌트 필드(eviction hint field, 1552B)로서 해석되고, 그 내용은 되찾기 힌트들 중 어느 것이 이용될 것인지를 구분하는(도 15a에서, 메모리 액세스, 임시(1525) 명령어 템플릿과, 메모리 액세스, 비-임시(1530) 명령어 템플릿에 대해 임시(1552B.1) 및 비-임시(1552B.2)가 각각 특정된다) 반면, 베타 필드(1554)는 데이터 조작 필드(1554C)로서 해석되고, 그 내용은 (프리미티브라고도 알려진) 다수의 데이터 조작 연산들 중 어느 것이 수행될 것인지를 구분한다(예를 들어, 조작 없음; 브로드캐스트; 소스의 업 컨버전; 및 목적지의 다운 컨버전). 메모리 액세스(1520) 명령어 템플릿들은, 스케일 필드(1560), 및 선택사항으로서의 변위 필드(1562A) 또는 변위 스케일 필드(1562B)를 포함한다.

[0125] 벡터 메모리 명령어들은 메모리로부터의 벡터 로드들 및 메모리로의 벡터 저장들을 수행하고, 변환이 지원된다. 정규 벡터 명령어들에서와 같이, 벡터 메모리 명령어들은 데이터 엘리먼트와 관련한 방식으로 메모리로부터/메모리로 데이터를 전송하고, 실제로 전송되는 엘리먼트들은 기입 마스크로서 선택되는 벡터 마스크의 내용들에 의해 지시된다.

[0126] 메모리 액세스 명령어 템플릿들-임시

[0127] 일시적 데이터는 캐싱으로부터 이득을 얻기에 충분히 빨리 재이용될 가능성이 있는 데이터이다. 그러나, 즉, 힌트, 및 상이한 프로세서들이 힌트 전체를 무시하는 것을 포함하여, 상이한 방식들로 그것을 구현할 수 있다.

[0128] 메모리 액세스 명령어 템플릿들 -비-임시

[0129] 비-일시적 데이터는 제1 레벨 캐시에서 캐싱으로부터 이득을 얻기에 충분히 빨리 재이용될 가능성이 없는 데이터이고, 되찾기를 위한 우선순위가 주어져야 한다. 그러나, 즉, 힌트, 및 상이한 프로세서들은 힌트 전체를 무시하는 것을 포함하여, 상이한 방식들로 그것을 구현할 수 있다.

[0130] **클래스 B의 명령어 템플릿들**

[0131] 클래스 B의 명령어 템플릿들의 경우에, 알파 필드(1552)는 기입 마스크 제어(Z) 필드(1552C)로서 해석되고, 그 내용은 기입 마스크 필드(1570)에 의해 제어된 기입 마스킹이 병합이어야 하는지 또는 제로화이어야 하는지를 구별한다.

[0132] 클래스 B의 메모리 액세스 없음(1505) 명령어 템플릿들의 경우, 베타 필드(1554)의 일부는 RL 필드(1557A)로서 해석되고, 그 내용은 상이한 증대 연산 타입들 중 어느 것이 수행될 것인지를 구분하는(예를 들어, 메모리 액세스 없음, 기입 마스크 제어, 부분 라운드 제어 타입 연산(1512) 명령어 템플릿과, 메모리 액세스 없음, 기입 마스크 제어, VSIZE 타입 연산(1517) 명령어 템플릿에 대해 각각 라운드(1557A.1) 및 벡터 길이(VSIZE)(1557A.2)가 특정된다) 반면, 베타 필드(1554)의 나머지는 지정된 타입의 연산들 중 어느 것이 수행될 것인지를 구분한다. 메모리 액세스 없음(1505) 명령어 템플릿들에서, 스케일 필드(1560), 변위 필드(1562A), 및 변위 스케일 필드(1562B)는 존재하지 않는다.

[0133] 메모리 액세스 없음, 기입 마스크 제어, 부분 라운드 제어 타입 연산(1510) 명령어 템플릿에서, 베타 필드(1554)의 나머지는 라운드 연산 필드(1559A)로서 해석되고 예외 이벤트 보고는 디스에이블된다(주어진 명령어는 어떠한 종류의 부동 소수점 예외 플래그도 보고하지 않고 어떠한 부동 소수점 예외 핸들러도 발생시키지 않는다).

[0134] 라운드 연산 제어 필드(1559A) - 라운드 연산 제어 필드(1558)와 같이, 그 내용은 한 그룹의 라운드 연산 중 어느 것을 수행할지를 구분해준다(예컨대, Round-up, Round-down, Round-towards-zero 및 Round-to-nearest). 따라서, 라운드 연산 제어 필드(1559A)는 명령어 당 기초로 라운딩 모드의 변경을 허용한다. 프로세서가 라운딩 모드들을 특정하기 위한 제어 레지스터를 포함하는 본 발명의 일 실시예에서, 라운드 연산 제어 필드(1550)의 내용은 그 레지스터 값을 무효로 한다.

[0135] 메모리 액세스 없음, 가입 마스크 제어, VSIZE 타입 연산(1517) 명령어 템플릿에서, 베타 필드(1554)의 나머지는 벡터 길이 필드(1559B)로서 해석되고, 그 내용은 다수의 데이터 벡터 길이들 중 어느 것이 수행될 것인지(예를 들어, 128, 256, 또는 512 바이트)를 구별한다.

[0136] 클래스 B의 메모리 액세스(1520) 명령어 템플릿의 경우에, 베타 필드(1554)의 부분은 브로드캐스트 필드(1557B)로서 해석되고, 그 내용은 브로드캐스트 타입 데이터 조작 연산이 수행될 것인지 여부를 구별하지만, 베타 필드(1554)의 나머지는 벡터 길이 필드(1559B)로서 해석된다. 메모리 액세스(1520) 명령어 템플릿들은, 스케일 필드(1560), 및 선택사항으로서의 변위 필드(1562A) 또는 변위 스케일 필드(1562B)를 포함한다.

[0137] 일반 벡터 친화형 명령어 포맷(1500)에 관하여, 포맷 필드(1540), 베이스 연산 필드(1542), 및 데이터 엘리먼트 폭 필드(1564)를 포함하는 전체 오피코드 필드(1574)가 도시되어 있다. 전체 오피코드 필드(1574)가 이들 필드들 모두를 포함하는 일 실시예가 도시되지만, 이들 모두를 지원하지는 않는 실시예들에서, 전체 오피코드 필드(1574)는 이들 필드들 모두보다 적은 필드들을 포함한다. 전체 오피코드 필드(1574)는 연산 코드(오피코드)를 제공한다.

[0138] 증대 연산 필드(1550), 데이터 엘리먼트 폭 필드(1564), 및 기입 마스크 필드(1570)는, 이들 특징들이 명령어별 기반으로 일반 벡터 친화형 명령어 포맷으로 명시되는 것을 허용한다.

[0139] 기입 마스크 필드와 데이터 엘리먼트 폭 필드의 조합은 그것들이 마스크가 상이한 데이터 엘리먼트 폭들에 기초하여 적용될 수 있게 한다는 점에서 타입 명령어들(typed instructions)을 생성한다.

[0140] 클래스 A 및 클래스 B 내에서 발견되는 다양한 명령어 템플릿들은 상이한 상황들에서 유익하다. 본 발명의 일부 실시예들에서, 상이한 프로세서들 또는 프로세서 내의 상이한 코어들은 오직 클래스 A, 오직 클래스 B, 또는 양자의 클래스들을 지원할 수 있다. 예를 들어, 범용 컴퓨팅을 위해 의도된 고성능 범용 비순차적 코어는 오직 클래스 B를 지원할 수 있고, 그래픽 및/또는 과학(쓰루풋) 컴퓨팅을 주로 대상으로 하는 코어는 오직 클래스 A를 지원할 수 있고, 양자 모두를 대상으로 하는 코어는 양자 모두를 지원할 수 있다(물론, 양자의 클래스들로부터의 템플릿들 및 명령어들의 일부 혼합을 갖지만 양자의 클래스들로부터의 템플릿들 및 명령어들 전부를 갖지

는 않는 코어는 본 발명의 관점 내에 있다). 또한, 싱글 프로세서가 다수의 코어들을 포함할 수 있고, 여기서, 코어들 전부가 동일한 클래스를 지원하거나 상이한 코어들이 상이한 클래스를 지원한다. 예를 들어, 별개의 그래픽 및 범용 코어들을 갖는 프로세서에서, 그래픽 및/또는 과학 컴퓨팅을 주로 대상으로 하는 그래픽 코어들 중 하나가 오직 클래스 A를 지원할 수 있지만, 범용 코어들 중 하나 이상이 오직 클래스 B를 지원하는 범용 컴퓨팅을 대상으로 하는 비순차적 실행 및 레지스터 리네이밍을 갖는 고성능 범용 코어들일 수 있다. 별도의 그래픽 코어를 갖지 않는 다른 프로세서는 클래스 A 및 클래스 B 양자를 지원하는 하나 이상의 범용 순차적 또는 비순차적 코어들을 포함할 수 있다. 물론, 한 클래스로부터의 특징들은 또한 본 발명의 상이한 실시예들에서 다른 클래스에 구현될 수 있다. 하이 레벨 언어로 작성되는 프로그램들이 다음을 포함하여, 여러가지 상이한 실행가능 형태로 놓여질 것이다(예를 들어 단지 시간적으로 컴파일되거나 또는 정적으로 컴파일됨): 1) 실행을 위해 타겟 프로세서에 의해 지원되는 클래스(들)의 명령어들만을 갖는 형태; 또는 2) 모든 클래스들의 명령어들의 상이한 조합들을 사용하여 기입되는 대안적인 루틴들을 갖고, 현재 코드를 실행하고 있는 프로세서에 의해 지원되는 명령어들에 기초하여 실행할 루틴들을 선택하는 제어 흐름 코드를 갖는 형태.

[0141] **예시적인 특정 벡터 친화형 명령어 포맷**

[0142] 도 16은 본 발명의 실시예들에 따른 예시적 특정 벡터 친화형 명령어 포맷을 도시하는 블럭도이다. 도 16은, 위치, 사이즈, 해석, 및 필드들의 순서 뿐만 아니라 이들 필드들의 일부에 대한 값들을 특정한다는 점에서 특정적인, 특정 벡터 친화형 명령어 포맷(1600)을 도시한다. 특정 벡터 친화형 명령어 포맷(1600)은 x86 명령어 세트를 확장하는데 이용될 수 있으므로, 필드들 중 일부는 기존 x86 명령어 세트 및 그 확장판(예를 들어, AVX)에서 이용되는 것들과 유사하거나 동일하다. 이 포맷은 확장들을 갖는 기존의 x86 명령어 세트의 프리픽스 인코딩 필드, 실제 오피코드 바이트 필드, MOD R/M 필드, SIB 필드, 변위 필드, 및 즉치 필드들과의 일관성을 유지한다. 도 15의 필드들에 맵핑되는 도 16으로부터의 필드들이 도시된다.

[0143] 본 발명의 실시예들은 예시적인 목적으로 일반 벡터 친화형 명령어 포맷(1500)의 맥락에서 특정 벡터 친화형 명령어 포맷(1600)을 참조하여 설명되지만, 본 발명은 청구되는 경우를 제외하고 특정 벡터 친화형 명령어 포맷(1600)에 제한되는 것이 아님을 이해하여야 한다. 예를 들어, 일반 벡터 친화형 명령어 포맷(1500)은 다양한 필드들에 대한 다양한 가능한 사이즈를 고려하는 반면, 특정 벡터 친화형 명령어 포맷(1600)은 특정 크기의 필드들을 갖는 것으로 도시되어 있다. 구체적인 예로서, 데이터 엘리먼트 폭 필드(1564)는 특정 벡터 친화형 명령어 포맷(1600)에서는 1 비트 필드로서 예시되어 있지만, 본 발명이 이에 제한되는 것은 아니다(즉, 일반 벡터 친화형 명령어 포맷(1500)은 데이터 엘리먼트 폭 필드(1564)의 다른 사이즈를 고려한다).

[0144] 일반 벡터 친화형 명령어 포맷(1500)은 도 16a에 나타낸 순서대로 아래에 열거되는 이하의 필드들을 포함한다.

[0145] EVEX 프리픽스(바이트들 0-3)(1602) - 4-바이트 형태로 인코딩된다.

[0146] 포맷 필드(1540)(EVEX 바이트 0, 비트 [7:0]) - 제1 바이트(EVEX 바이트 0)는 포맷 필드(1540)이고, 0x62(본 발명의 일 실시예에서 벡터 친화형 명령어 포맷을 구분해주는 데 사용되는 고유값)를 포함하고 있다.

[0147] 제2-제4 바이트들(EVEX 바이트들 1-3)은 특정 능력을 제공하는 다수의 비트 필드들을 포함한다.

[0148] REX 필드(1605)(EVEX 바이트 1, 비트들 [7-5]) - EVEX.R 비트 필드(EVEX 바이트 1, 비트 [7] - R), EVEX. X 비트 필드(EVEX 바이트 1, 비트 [6] - X) 및 1557BEX 바이트 1, 비트[5] - B)로 구성된다. EVEX.R, EVEX.X와 EVEX.B 비트 필드들은 대응 VEX 비트 필드들과 동일한 기능성을 제공하고, 1s 보수 형태을 사용하여 인코딩된다, 즉 ZMM0는 1111B로 인코딩되고, ZMM15는 0000B로 인코딩된다. 명령어들의 다른 필드들은 이 기술분야에 알려진 바와 같이 레지스터 인덱스들의 하위 3 비트를 인코딩하여서(rrr, xxx, 및 bbb), EVEX.R, EVEX.X, 및 EVEX.B를 추가함으로써 Rrrr, Xxxx, 및 Bbbb가 형성될 수 있다.

[0149] REX' 필드(1510) - 이것은 REX' 필드(1510)의 제1 부분이고, 확장된 32개의 레지스터 세트의 상위 16 또는 하위 16을 인코딩하는데 사용되는 EVEX.R' 비트 필드(EVEX 바이트 1, 비트 [4] - R')이다. 본 발명의 일 실시예에서, 이 비트는 아래 표시된 바와 같은 다른 것들과 함께, (공지된 x86 32-비트 모드에서) BOUND 명령어로부터 구별하기 위해 비트 반전된 포맷으로 저장되고, 그의 실제 오피코드 바이트가 62이지만, (아래 설명된) MOD R/M 필드에서 MOD 필드의 11의 값을 수락하지 않으며; 본 발명의 대안적인 실시예들은 반전된 포맷으로 이것 및 아래 다른 표시된 비트들을 저장하지 않는다. 1의 값을 이용하여 하위 16개의 레지스터를 인코딩한다. 달리 말하면, R'Rrrr은 다른 필드들로부터의 EVEX.R', EVEX.R 및 다른 RRR를 조합함으로써 형성된다.

[0150] 오피코드 맵 필드(1615)(EVEX 바이트 1, 비트 [3:0] - mmmm) - 그 내용은 암시적 리딩(implied leading) 오피

코드 바이트(0F, 0F 38 또는 0F 3)를 인코딩한다.

[0151] 데이터 엘리먼트 폭 필드(1564)(EVEX 바이트 2, 비트 [7] - W) - 표기 EVEX.W로 나타난다. EVEX.W는 데이터타입의 그래뉼러티(사이즈)를 정의하는데 사용된다(32 비트 데이터 엘리먼트들 또는 64 비트 데이터 엘리먼트들).

[0152] EVEX.vvvv(1620)(EVEX 바이트 2, 비트들 [6:3]-vvvv) - EVEX.vvvv의 역할은 다음을 포함할 수 있다: 1) EVEX.vvvv는 반전된 (1의 보수) 형태로 특정되는 제1 소스 레지스터 피연산자를 인코딩하고 2개 이상의 소스 피연산자를 갖는 명령어에 대해 유효하다; 2) EVEX.vvvv는 특정 벡터 시프트에 대해 1의 보수 형태로 특정되는 목적지 레지스터 피연산자를 인코딩한다; 또는 3) EVEX.vvvv는 어떠한 피연산자도 인코딩하지 않으며, 이 필드는 예약되거나 1111b를 포함해야 한다. 따라서, EVEX.vvvv 필드(1620)는 반전된 (1의 보수) 형태로 저장되는 제1 소스 레지스터 지정자의 하위 4 비트를 인코딩한다. 명령어에 따라, 추가의 상이한 EVEX 비트 필드는 특정자 사이즈를 32개의 레지스터로 확장하는데 사용된다.

[0153] EVEX.U(1568) 클래스 필드(EVEX 바이트 2, 비트 [2]-U) - EVEX.U = 0이면, 클래스 A 또는 EVEX.U0을 나타내고; EVEX.U = 1이면, 클래스 B 또는 EVEX.U1을 나타낸다.

[0154] 프리픽스 인코딩 필드(1625)(EVEX 바이트 2, 비트 [1:0]-pp) - 베이스 연산 필드에 대한 추가적인 비트들을 제공한다. EVEX 프리픽스 포맷의 레거시 SSE 명령어들에 대한 지원을 제공하는 것 이외에, 이것은 또한 SIMD 프리픽스를 콤팩트화하는 이득을 갖는다(SIMD 프리픽스를 표현하기 위한 바이트를 요구하기보다는, EVEX 프리픽스는 2비트만을 요구한다). 일 실시예에서, 레거시 포맷과 EVEX 프리픽스 포맷 양자의 SIMD 프리픽스(66H, F2H, F3H)를 사용하는 레거시 SSE 명령어들을 지원하기 위하여, 이들 레거시 SIMD 프리픽스들은 SIMD 프리픽스 인코딩 필드로 인코딩되고; 런타임에 디코더의 PLA에 제공되기 전에 레거시 SIMD 프리픽스로 확장된다(그래서 PLA는 수정 없이 이들 레거시 명령어들의 레거시 및 EVEX 포맷 양자를 실행할 수 있다). 더 새로운 명령어들은 오피코드 확장으로서 직접 EVEX 프리픽스 인코딩 필드의 내용을 사용할 수 있더라도, 특정 실시예들은 일관성을 위해 유사한 방식으로 확장하지만, 이들 레거시 SIMD 프리픽스들에 의해 특정되는 상이한 의미들을 허용한다. 대안적인 실시예는 2 비트 SIMD 프리픽스 인코딩들을 지원하도록 PLA를 재설계할 수 있고, 따라서 확장을 요구하지 않는다.

[0155] 알파 필드(1552)(EVEX 바이트 3, 비트 [7] - EH; 또한 EVEX.EH, EVEX.rs, EVEX.RL, EVEX.기입 마스크 제어, 및 EVEX.N으로도 알려지고; 또한 α로 도시됨) - 이미 개시된 바와 같이, 이 필드는 맥락 특정적이다.

[0156] 베타 필드(1554)(EVEX 바이트 3, 비트들 [6:4]-SSS, 또한 EVEX.s_{2-0}, EVEX.r_{2-0}, EVEX.rr1, EVEX.LL0, EVEX.LLB로 알려지고; 또한 βββ로 도시됨) - 이미 개시된 바와 같이, 이 필드는 맥락 특정적이다.

[0157] REX' 필드(1510) - 이것은 REX' 필드의 나머지이고, 확장된 32 레지스터 세트의 상위 16 또는 하위 16 중 어느 하나를 인코딩하는 데 사용될 수 있는 EVEX.V' 비트 필드(EVEX 바이트 3, 비트 [3] - V')이다. 이 비트는 비트 반전된 포맷으로 저장된다. 1의 값을 이용하여 하위 16개의 레지스터를 인코딩한다. 다시 말하면, V'VVVV는 EVEX.V', EVEX.vvvv를 조합하여 형성된다.

[0158] 기입 마스크 필드(1570)(EVEX 바이트 3, 비트 [2:0]-kkk) - 그 내용은 앞서 설명된 바와 같이 기입 마스크 레지스터들에서의 레지스터의 인덱스를 특정한다. 본 발명의 일 실시예에서, 특정 값 EVEX.kkk=000은 특정 명령어에 대해 어떠한 기입 마스크도 이용되지 않음을 암시하는 특수 거동을 갖는다(이것은 모두 1로 하드와이어드된(hardwired) 기입 마스크 또는 마스킹 하드웨어를 바이패스하는 하드웨어의 사용을 포함하는 다양한 방식으로 구현될 수 있다).

[0159] 실제 오피코드 필드(1630)(바이트 4)는 또한 오피코드 바이트로 알려진다. 오피코드의 부분은 이 필드에서 특정된다.

[0160] MOD R/M 필드(1640)(바이트 5)는 MOD 필드(1642), Reg 필드(1644) 및 R/M 필드(1646)를 포함한다. 이미 개시된 바와 같이, MOD 필드(1642)의 내용은 메모리 액세스 연산과 메모리 액세스 없음 연산 사이를 구별한다. Reg 필드(1644)의 역할은 2가지 상황들로 요약될 수 있다: 목적지 레지스터 피연산자 또는 소스 레지스터 피연산자를 인코딩하는 상황, 또는 오피코드 확장으로서 취급되고 어떠한 명령어 피연산자를 인코딩하는데 사용되지 않는 상황. R/M 필드(1646)의 역할은 다음을 포함할 수 있다: 메모리 어드레스를 참조하는 명령어 피연산자를 인코딩하는 것, 또는 목적지 레지스터 피연산자 또는 소스 레지스터 피연산자를 암호화하는 것.

[0161] SIB(Scale, Index, Base) 바이트(바이트 6) - 이미 개시된 바와 같이, 스케일 필드(1550)의 내용은 메모리 어드레스 생성을 위해 사용된다. SIB.XXX(1654) 및 SIB.bbb(1656) - 이러한 필드들의 내용들은 이미 레지스터 인

덱스들 Xxxx 및 Bbbb에 관해서 언급되었다.

[0162] 변위 필드(1562A)(바이트 7-10) - MOD 필드(1642)가 10을 포함할 때, 바이트 7-10은 변위 필드(1562A)이고, 이는 레거시 32-비트 변위(disp32)와 동일하게 동작하고 바이트 그래뉼러티(byte granularity)로 동작한다.

[0163] 변위 인자 필드(1562B)(바이트 7) - MOD 필드(1642)가 01을 포함할 때, 바이트 7은 변위 인자 필드(1562B)이다. 이 필드의 위치는 바이트 그래뉼러티에서 동작하는 레거시 x86 명령어 세트 8-비트 변위(disp8)의 위치와 동일하다. disp8은 부호 확장되기 때문에, 오직 -128 내지 127 바이트들 오프셋들 사이를 어드레스할 수 있고; 64 바이트 캐시 라인들에 대하여, disp8은 오직 4개의 실제 유용한 값들 -128, -64, 0, 및 64로 설정될 수 있는 8 비트들을 사용하며; 더 큰 범위가 종종 필요하기 때문에, disp32가 사용된다; 그러나, disp32는 4 바이트를 요구한다. disp8 및 disp32와 달리, 변위 인자 필드(1562B)는 disp8의 재해석이고; 변위 인자 필드(1562B)를 사용할 때, 변위 인자 필드의 내용과 메모리 피연산자 액세스의 사이즈(N)를 곱한 것에 의해 실제 변위가 결정된다. 이러한 타입의 변위를 disp8*N이라고 한다. 이것은 평균 명령어 길이를 감소시킨다(훨씬 더 큰 범위를 갖는 변위에 대해 사용되는 단일 바이트). 그러한 압축된 변위는, 유효 변위가 메모리 액세스의 그래뉼러티의 배수이고, 따라서 어드레스 오프셋의 잉여 하위 비트들이 인코딩될 필요가 없다는 추정에 기초한다. 달리 말하면, 변위 인자 필드(1562B)는 레거시 x86 명령어 세트 8-비트 변위를 대체한다. 따라서, 변위 인자 필드(1562B)는 x86 명령어 세트 8-비트 변위와 동일한 방식으로 인코딩되고(따라서 ModRM/SIB 인코딩 규칙의 변화가 없음), 유일한 예외는 disp8이 disp8*N으로 오버로드(overload)된다는 것이다. 다시 말해, 인코딩 규칙들 또는 인코딩 길이들에 있어서 어떠한 변경도 존재하지 않지만 오직 하드웨어에 의한 변위 값의 해석에 있어서 변경이 존재한다(이것은 바이트-와이즈 어드레스 오프셋(byte-wise address offset)을 획득하기 위해 메모리 피연산자의 사이즈에 의해 변위를 스케일링할 필요가 있다).

[0164] 즉치 필드(1572)는 앞서 설명된 바와 같이 동작한다.

[0165] **전체 오피코드 필드**

[0166] 도 16b는 본 발명의 일 실시예에 따라 전체 오피코드 필드(1574)를 구성하는 특정 벡터 친화형 명령어 포맷(1600)의 필드들을 도시하는 블럭도이다. 구체적으로, 전체 오피코드 필드(1574)는 포맷 필드(1540), 베이스 연산 필드(1542), 및 데이터 엘리먼트 폭(W) 필드(1564)를 포함한다. 베이스 연산 필드(1542)는 프리픽스 인코딩 필드(1625), 오피코드 맵 필드(1615), 및 실제 오피코드 필드(1630)를 포함한다.

[0167] **레지스터 인덱스 필드**

[0168] 도 16c는 본 발명의 일 실시예에 따라 레지스터 인덱스 필드(1544)를 구성하는 특정 벡터 친화형 명령어 포맷(1600)의 필드들을 도시하는 블럭도이다. 구체적으로, 레지스터 인덱스 필드(1544)는 REX 필드(1605), REX' 필드(1610), MODR/M.reg 필드(1644), MODR/M.r/m 필드(1646), VVVV 필드(1620), xxx 필드(1654) 및 bbb 필드(1656)를 포함한다.

[0169] **증대 연산 필드**

[0170] 도 16d는 본 발명의 일 실시예에 따라 증대 연산 필드(1550)를 구성하는 특정 벡터 친화형 명령어 포맷(1600)의 필드들을 도시하는 블럭도이다. 클래스 (U) 필드(1568)가 0을 포함하면, EVEX.U0(클래스 A(1568A))을 의미하고; 1을 포함하면, EVEX.U1(클래스 B(1568B))을 의미한다. U=0이고 MOD 필드(1642)가 11을 포함하면(메모리 액세스 없음 연산을 의미함), 알파 필드(1552)(EVEX 바이트 3, 비트 [7] - EH)는 rs 필드(1552A)로서 해석된다. rs 필드(1552A)가 1을 포함할 때(라운드(1552A.1)), 베타 필드(1554)(EVEX 바이트 3, 비트 [6:4] - SSS)는 라운드 제어 필드(1554A)로서 해석된다. 라운드 제어 필드(1554A)는 1 비트 SAE 필드(1556) 및 2 비트 라운드 연산 필드(1558)를 포함한다. rs 필드(1552A)가 0을 포함할 때(데이터 변환(1552A.2)), 베타 필드(1554)(EVEX 바이트 3, 비트들 [6:4] - SSS)는 3 비트 데이터 변환 필드(1554B)로서 해석된다. U=0이고 MOD 필드(1642)가 00, 01 또는 10을 포함하면(메모리 액세스 연산을 의미함), 알파 필드(1552)(EVEX 바이트 3, 비트 [7] - EH)는 EH(Eviction Hint) 필드(1552B)로서 해석되고, 베타 필드(1554)(EVEX 바이트 3, [6:4] - SSS)는 3 비트 데이터 조작 필드(1554C)로서 해석된다.

[0171] U=1일 때, 알파 필드(1552)(EVEX 바이트 3, 비트 [7] - EH)는 기입 마스크 제어(Z) 필드(1552C)로서 해석된다. U=1이고 MOD 필드(1642)가 11을 포함하면(메모리 액세스 없음 연산을 의미함), 베타 필드(1554)의 부분(EVEX 바이트 3, 비트 [4] - S_0)은 RL 필드(1557A)로서 해석되고; 1을 포함하면(라운드(1557A.1)) 베타 필드(1554)의 나머지(EVEX 바이트 3, 비트 [6-5]-S_{2-1})는 라운드 연산 필드(1559A)로서 해석되는 한편, RL 필드(1557A)가

0(VSIZE(1557.(A2))을 포함하면 베타 필드(1554)의 나머지(EVEX 바이트 3, 비트 [6-5]-S_{2-1})는 벡터 길이 필드(1559B)(EVEX 바이트 3, 비트 [6-5]-L_{1-0})로서 해석된다. U=1이고 MOD 필드(1642)가 00, 01, 또는 10을 포함하면(메모리 액세스 연산을 의미함), 베타 필드(1554)(EVEX 바이트 3, 비트 [6:4]-SSS)는 벡터 길이 필드(1559B)(EVEX 바이트 3, 비트 [6-5]-L_{1-0}) 및 브로드캐스트 필드(1557B)(EVEX 바이트 3, 비트 [4]-B)로서 해석된다.

[0172] **예시적인 레지스터 아키텍처**

[0173] 도 17은 본 발명의 일 실시예에 따른 레지스터 아키텍처(1700)의 블럭도이다. 도시된 실시예에서는, 폭이 512 비트인 32개의 벡터 레지스터(1710)가 존재하고; 이들 레지스터들은 zmm0 내지 zmm31로서 참조된다. 하위 16개 zmm 레지스터들의 하위 256 비트들은 레지스터들 ymm0-16에 중첩된다. 하위 16개 zmm 레지스터들의 하위 128 비트들(ymm 레지스터들의 하위 128 비트들)은 레지스터들 xmm0-15에 중첩된다. 특정 벡터 친화형 명령어 포맷(1600)은 아래 표에 예시된 바와 같이 이들 중첩된 레지스터 파일에 대해 동작한다.

표 1

[0174]

조정가능한 벡터 길이	클래스	연산들	레지스터들
벡터 길이 필드(1559B)를 포함하지 않는 명령어 템플릿들	A (도 15a; U=0)	1510, 1515, 1525, 1530	zmm 레지스터들(벡터 길이가 64 바이트임)
	B (도 15b; U=1)	1512	zmm 레지스터들(벡터 길이가 64 바이트임)
벡터 길이 필드(1559B)를 포함하는 명령어 템플릿들	B (도 15b; U=1)	1517, 1527	벡터 길이 필드(1559B)에 따라 zmm, ymm 또는 xmm 레지스터들(벡터 길이가 64 바이트, 32 바이트 또는 16 바이트임)

[0175] 환언하면, 벡터 길이 필드(1559B)는 최대 길이와 하나 이상의 다른 보다 짧은 길이 중에서 선택을 하고, 여기서 각각의 이러한 보다 짧은 길이는 선행 길이의 1/2 길이이며; 벡터 길이 필드(1559B)를 갖지 않는 명령어 템플릿들은 최대 벡터 길이에 대해 작용한다. 또한, 일 실시예에서, 특정 벡터 친화형 명령어 포맷(1600)의 클래스 B 명령어 템플릿들은 패킹된 또는 스칼라의 싱글/더블-정밀도 부동 소수점 데이터 및 패킹된 또는 스칼라 정수 데이터에 작용한다. 스칼라 연산들은 zmm/ymm/xmm 레지스터에서 최하위 데이터 엘리먼트 위치에서 수행되는 연산들이고; 상위 데이터 엘리먼트 위치들은 실시예에 따라 명령어 이전과 동일하게 두거나 또는 제로화된다.

[0176] 기입 마스크 레지스터들(1715) - 도시된 실시예에서는, 각각이 64 비트 사이즈인 8개의 기입 마스크 레지스터(k0 내지 k7)가 있다. 대안적인 실시예에서, 기입 마스크 레지스터들(1715)은 16 비트 사이즈이다. 이미 개시된 바와 같이, 본 발명의 일 실시예에서, 벡터 마스크 레지스터 k0는 기입 마스크로서 사용될 수 없고; 정상적으로 k0을 표시하는 인코딩이 기입 마스크에 사용될 때, 이는 0xFFFF의 하드와이어드 기입 마스크를 선택하여, 그 명령어에 대한 기입 마스킹을 효과적으로 디스에이블한다.

[0177] 범용 레지스터들(1725) - 도시된 실시예에서는, 메모리 피연산자를 어드레싱하는데 기존의 x86 어드레싱 모드와 함께 사용되는 16개의 64-비트 범용 레지스터가 있다. 이러한 레지스터들은 RAX, RBX, RCX, RDX, RBP, RSI, RDI, RSP, 및 R8 내지 R15라는 이름으로 참조된다.

[0178] MMX 패킹된 정수 플랫 레지스터 파일(1750)로 에일리어싱(aliasing)된 스칼라 부동 소수점 스택 레지스터 파일(x87 스택)(1745) - 도시된 실시예에서, x87 스택은 x87 명령어 세트 확장을 사용하여 32/64/80 비트 부동 소수점 데이터에 대해 스칼라 부동 소수점 연산들을 수행하는데 사용되는 8-엘리먼트 스택인 반면; MMX 레지스터들은 MMX 레지스터와 XMM 레지스터 사이에 수행되는 일부 연산을 위한 피연산자를 유지할 뿐 아니라 64 비트 패킹된 정수 데이터에 대해 연산들을 수행하는데 사용된다.

[0179] 본 발명의 대안적인 실시예들은 더 넓거나 더 좁은 레지스터들을 이용할 수 있다. 부가적으로, 본 발명의 대안적인 실시예들은 더 많거나, 더 적거나, 상이한 레지스터 파일들 및 레지스터들을 이용할 수 있다.

[0180] **예시적인 코어 아키텍처들, 프로세서들 및 컴퓨터 아키텍처들**

[0181] 프로세서 코어들은 상이한 방식으로, 상이한 목적들을 위해, 상이한 프로세서들에서 구현될 수 있다. 예를 들

어, 이러한 코어들의 구현들은: 1) 범용 컴퓨팅을 대상으로 하는 범용 순차적 코어; 2) 범용 컴퓨팅을 대상으로 하는 고 성능 범용 비순차적 코어; 3) 그래픽 및/또는 과학적 (쓰루풋) 컴퓨팅을 주로 대상으로 하는 특수 목적 코어를 포함할 수 있다. 상이한 프로세서들의 구현들은: 1) 범용 컴퓨팅을 대상으로 하는 하나 이상의 범용 순차적 코어들 및/또는 범용 컴퓨팅을 대상으로 하는 하나 이상의 범용 비순차적 코어들을 포함하는 CPU; 및 2) 그래픽 및/또는 과학적 (쓰루풋) 컴퓨팅을 주로 대상으로 하는 하나 이상의 특수 목적 코어들을 포함하는 코프로세서를 포함할 수 있다. 이러한 상이한 프로세서들은 상이한 컴퓨터 시스템 아키텍처들을 초래하며, 이는: 1) CPU와는 별개인 칩 상의 코프로세서; 2) CPU와 동일한 패키지 내의 별개의 다이 상의 코프로세서; 3) CPU와 동일한 다이 상의 코프로세서(이 경우에, 이러한 코프로세서를 때때로 통합 그래픽 및/또는 과학적 (쓰루풋) 로직 등의 특수 목적 로직이라고 하거나, 또는 특수 목적 코어들이라고 함); 및 4) 설명된 CPU(때때로 애플리케이션 코어(들) 또는 애플리케이션 프로세서(들)라고 함), 위에 개시된 코프로세서, 및 부가적인 기능성을 동일한 다이 상에 포함할 수 있는 시스템 온 칩(system on a chip)을 포함할 수 있다. 예시적인 코어 아키텍처들이 다음에 개시되고, 예시적인 프로세서들 및 컴퓨터 아키텍처들의 개시들이 후속된다.

[0182] **예시적인 코어 아키텍처들**

[0183] **순차적 및 비순차적 코어 블럭도**

[0184] 도 18a는 본 발명의 실시예들에 따른 예시적인 순차적 파이프라인 및 예시적인 레지스터 리네이밍, 비순차적 발행/실행 파이프라인 양자 모두를 도시하는 블럭도이다. 도 18b는 본 발명의 실시예들에 따른 프로세서에 포함될 순차적 아키텍처 코어 및 예시적인 레지스터 리네이밍, 비순차적 발행/실행 아키텍처 코어 양자 모두의 예시적인 실시예를 도시하는 블럭도이다. 도 18a-b에서 실선 박스들은 순차적 파이프라인 및 순차적 코어를 도시하는 한편, 점선 박스들의 선택적 추가는 레지스터 리네이밍, 비순차적 발행/실행 파이프라인 및 코어를 도시한다. 순차적 양상이 비순차적 양상의 서브세트라는 점을 고려하여, 비순차적 양상이 설명될 것이다.

[0185] 도 18a에서, 프로세서 파이프라인(1800)은 페치 스테이지(1802), 길이 디코드 스테이지(1804), 디코드 스테이지(1806), 할당 스테이지(1808), 리네이밍 스테이지(1810), (디스패치 또는 발행으로도 알려진) 스케줄링 스테이지(1812), 레지스터 판독/메모리 판독 스테이지(1814), 실행 스테이지(1816), 라이트 백(write back)/메모리 기입 스테이지(1818), 예외 핸들링 스테이지(1822) 및 커미트(commit) 스테이지(1824)를 포함한다.

[0186] 도 18b는 실행 엔진 유닛(1850)에 연결되는 프론트 엔드 유닛(1830)을 포함하는 프로세서 코어(1890)를 도시하며, 이들 양자 모두는 메모리 유닛(1870)에 연결된다. 코어(1890)는 RISC(Reduced Instruction Set Computing) 코어, CISC(Complex Instruction Set Computing) 코어, VLIW(Very Long Instruction Word) 코어, 또는 하이브리드 또는 대안적인 코어 타입일 수 있다. 또 다른 옵션으로서, 코어(1890)는, 예를 들어 네트워크 또는 통신 코어, 압축 엔진, 코프로세서 코어, GPGPU(General Purpose computing Graphics Processing Unit) 코어, 그래픽 코어 또는 이와 유사한 것 등의 특수 목적 코어일 수 있다.

[0187] 프론트 엔드 유닛(1830)은 명령어 캐시 유닛(1834)에 연결되는 분기 예측 유닛(1832)을 포함하고, 명령어 캐시 유닛(1834)은 명령어 TLB(Translation Lookaside Buffer)(1836)에 연결되고, 명령어 TLB(1836)는 명령어 페치 유닛(1838)에 연결되고, 명령어 페치 유닛(1838)은 디코드 유닛(1840)에 연결된다. 디코드 유닛(1840)(또는 디코더)은 명령어들을 디코딩할 수 있으며, 오리지널 명령어들로부터 디코딩되거나, 또는 다른 방식으로 이들을 반영하거나, 또는 이들로부터 유도되는, 하나 이상의 마이크로-연산들, 마이크로-코드 엔트리 포인트들, 마이크로명령어들, 다른 명령어들 또는 다른 제어 신호들을 출력으로서 생성할 수 있다. 디코드 유닛(1840)은 여러가지 상이한 메커니즘들을 사용하여 구현될 수 있다. 적합한 메커니즘들의 예들은, 룩-업 테이블들, 하드웨어 구현들, PLA들(Programmable Logic Arrays), 마이크로코드 ROM(Read Only Memory)들 등을 포함하지만, 이에 제한되는 것은 아니다. 일 실시예에서 코어(1890)는 (예를 들어, 디코드 유닛(1840)에 또는 다른 방식으로 프론트 엔드 유닛(1830) 내에) 특정 매크로 명령어들에 대한 마이크로코드를 저장하는 마이크로코드 ROM 또는 다른 매체를 포함한다. 디코드 유닛(1840)은 실행 엔진 유닛(1850)에서의 리네임/할당자 유닛(1852)에 연결된다.

[0188] 실행 엔진 유닛(1850)은 하나 이상의 스케줄러 유닛(들)(1856)의 세트 및 회수 유닛(1854)에 연결되는 리네임/할당자 유닛(1852)을 포함한다. 스케줄러 유닛(들)(1856)은 예약 스테이션들, 중앙 명령어 윈도우 등을 포함하는 임의 수의 상이한 스케줄러들을 나타낸다. 스케줄러 유닛(들)(1856)은 물리적 레지스터 파일(들) 유닛(들)(1858)에 연결된다. 물리적 레지스터 파일(들) 유닛들(1858) 각각은 하나 이상의 물리적 레지스터 파일들을 나타내고, 이들 중 상이한 것들은 스칼라 정수, 스칼라 부동 소수점, 패킹된 정수, 패킹된 부동 소수점, 벡터 정수, 벡터 부동 소수점, 상태(예를 들어, 실행될 다음 명령어의 어드레스인 명령어 포인터) 등의 하나 이상의 상이한 데이터 타입들을 저장한다. 일 실시예에서, 물리적 레지스터 파일(들) 유닛(1858)은 벡터 레지스터

유닛, 기입 마스크 레지스터 유닛 및 스칼라 레지스터 유닛을 포함한다. 이러한 레지스터 유닛들은 아키텍처의 벡터 레지스터들, 벡터 마스크 레지스터들 및 범용 레지스터들을 제공할 수 있다. 물리적 레지스터 파일(들) 유닛(들)(1858)은, 레지스터 리네이밍 및 비순차적 실행이 (예를 들어, 재정리 버퍼(들) 및 회수 레지스터 파일(들)을 사용하여; 미래 파일(들), 이력 버퍼(들) 및 회수 레지스터 파일(들)을 사용하여; 레지스터 맵들 및 레지스터들의 풀(pool)을 사용하여 등) 구현될 수 있는 다양한 방식들을 도시하도록 회수 유닛(1854)에 의해 오버랩된다. 회수 유닛(1854) 및 물리적 레지스터 파일(들) 유닛(들)(1858)은 실행 클러스터(들)(1860)에 연결된다. 실행 클러스터(들)(1860)는 하나 이상의 실행 유닛들(1862)의 세트 및 하나 이상의 메모리 액세스 유닛들(1864)의 세트를 포함한다. 실행 유닛들(1862)은 다양한 타입의 데이터(예를 들어, 스칼라 부동 소수점, 패킹된 정수, 패킹된 부동 소수점, 벡터 정수, 벡터 부동 소수점)에 대해 다양한 연산들(예로서, 시프트, 가산, 감산, 승산)을 수행할 수 있다. 일부 실시예들은 특정 펑션들이나 펑션들의 세트들에 전용의 다수의 실행 유닛들을 포함할 수 있지만, 다른 실시예들은 단 하나의 실행 유닛, 또는 모두가 모든 펑션들을 수행하는 다수의 실행 유닛을 포함할 수 있다. 스케줄러 유닛(들)(1856), 물리적 레지스터 파일(들) 유닛(들)(1858) 및 실행 클러스터(들)(1860)는 복수 개일 수 있는 것으로 도시되는데, 그 이유는 특정 실시예들이 특정 타입들의 데이터/연산들에 대해 개별 파이프라인들(예를 들어, 자신들의 스케줄러 유닛, 물리적 레지스터 파일(들) 유닛 및/또는 실행 클러스터를 각각 갖는 스칼라 정수 파이프라인, 스칼라 부동 소수점/패킹된 정수/패킹된 부동 소수점/벡터 정수/벡터 부동 소수점 파이프라인 및/또는 메모리 액세스 파이프라인 - 그리고 개별 메모리 액세스 파이프라인의 경우, 이러한 파이프라인의 실행 클러스터만이 메모리 액세스 유닛(들)(1864)을 갖는 특정 실시예들이 구현됨)을 생성하기 때문이다. 개별 파이프라인들이 사용되는 경우, 이들 파이프라인들 중 하나 이상은 비순차적 발행/실행일 수 있고 나머지는 순차적일 수 있다는 점도 이해되어야 한다.

[0189] 메모리 액세스 유닛들(1864)의 세트는, 레벨 2(L2) 캐시 유닛(1876)에 연결되는 데이터 캐시 유닛(1874)에 연결되는 데이터 TLB 유닛(1872)을 포함하는 메모리 유닛(1870)에 연결된다. 예시적인 일 실시예에서, 메모리 액세스 유닛들(1864)은 로드 유닛, 저장 어드레스 유닛 및 저장 데이터 유닛을 포함할 수 있으며, 이들 각각은 메모리 유닛(1870) 내의 데이터 TLB 유닛(1872)에 연결된다. 명령어 캐시 유닛(1834)은 메모리 유닛(1870) 내의 레벨 2(L2) 캐시 유닛(1876)에 더 연결된다. L2 캐시 유닛(1876)은 하나 이상의 다른 레벨들의 캐시에 그리고 궁극적으로 메인 메모리에 연결된다.

[0190] 예를 들어, 예시적인 레지스터 리네이밍, 비순차적 발행/실행 코어 아키텍처는 다음과 같이 파이프라인(1800)을 구현할 수 있다: 1) 명령어 페치(1838)는 페치 및 길이 디코딩 스테이지들(1802, 1804)을 수행하고; 2) 디코드 유닛(1840)은 디코드 스테이지(1806)를 수행하고; 3) 리네임/할당자 유닛(1852)은 할당 스테이지(1808) 및 리네이밍 스테이지(1810)를 수행하고; 4) 스케줄러 유닛(들)(1856)은 스케줄 스테이지(1812)를 수행하고; 5) 물리적 레지스터 파일(들) 유닛(들)(1858) 및 메모리 유닛(1870)은 레지스터 판독/메모리 판독 스테이지(1814)를 수행하고; 실행 클러스터(1860)는 실행 스테이지(1816)를 수행하고; 6) 메모리 유닛(1870) 및 물리적 레지스터 파일(들) 유닛(들)(1858)은 라이트 백/메모리 기입 스테이지(1818)를 수행하고; 7) 다양한 유닛들이 예외 처리 스테이지(1822)에 관련될 수 있고; 8) 회수 유닛(1854) 및 물리적 레지스터 파일(들) 유닛(들)(1858)은 커미트 스테이지(1824)를 수행한다.

[0191] 코어(1890)는, 본 명세서에서 개시되는 명령어(들)를 포함하는, 하나 이상의 명령어 세트들(예를 들어, (보다 새로운 버전들과 함께 추가된 일부 확장들을 갖는) x86 명령어 세트; 캘리포니아주 서니베일의 MIPS 테크놀로지스의 MIPS 명령어 세트; 캘리포니아주 서니베일의 ARM 홀딩스의 (NEON 등의 선택적 추가 확장들을 갖는) ARM 명령어 세트)을 지원할 수 있다. 일 실시예에서, 코어(1890)는 패킹된 데이터 명령어 세트 확장(예를 들어, AVX1, AVX2)을 지원하는 로직을 포함하며, 따라서 많은 멀티미디어 애플리케이션들에 의해 사용되는 연산들이 패킹된 데이터를 사용하여 수행되는 것을 허용한다.

[0192] 코어는 (2 이상의 병렬 세트들의 연산이나 쓰레드들을 실행하는) 멀티쓰레딩을 지원할 수 있고, 시분할 멀티쓰레딩(time sliced multithreading), (단일의 물리적 코어가, 물리적 코어가 동시에 멀티쓰레딩하는 쓰레드들 각각에 대해 논리적 코어를 제공하는) 동시 멀티쓰레딩, 또는 이들의 조합(예를 들어, Intel® Hyperthreading 기술에서와 같은 시분할 페칭 및 디코딩과 그 이후의 동시 멀티쓰레딩)을 포함하는 다양한 방식으로 멀티쓰레딩을 지원할 수 있다는 점이 이해되어야 한다.

[0193] 레지스터 리네이밍이 비순차적 실행의 맥락에서 설명되었지만, 레지스터 리네이밍은 순차적 아키텍처에서 사용될 수도 있다는 점이 이해되어야 한다. 도시된 프로세서의 실시예는 또한 개별 명령어 및 데이터 캐시 유닛들(1834/1874) 및 공유 L2 캐시 유닛(1876)을 포함하지만, 대안적인 실시예들은, 예를 들어, 레벨 1(L1) 내부 캐시 또는 다수 레벨들의 내부 캐시와 같이 명령어들 및 데이터 양자 모두에 대해 단일 내부 캐시를 가질 수

있다. 일부 실시예들에서, 시스템은 내부 캐시와, 코어 및/또는 프로세서에 대해 외부에 있는 외부 캐시의 조합을 포함할 수 있다. 대안적으로, 모든 캐시는 코어 및/또는 프로세서에 대해 외부에 있을 수 있다.

[0194] **특정 예시적인 순차적 코어 아키텍처**

[0195] 도 19a-b는, 코어가 칩 내의 (동일 타입 및/또는 상이한 타입들의 다른 코어들을 포함하는) 여러 논리 블럭 중 하나인, 보다 구체적인 예시적인 순차적 코어 아키텍처의 블럭도를 도시한다. 논리 블럭들은 애플리케이션에 따라 일부 고정된 펑션 로직, 메모리 I/O 인터페이스들 및 다른 필요한 I/O 로직과 고-대역폭 상호접속 네트워크(예를 들어, 링 네트워크)를 통해 통신한다.

[0196] 도 19a는, 본 발명의 실시예들에 따른, 싱글 프로세서 코어의 블럭도로, 온-다이(on-die) 상호접속 네트워크(1902)에 대한 접속, 및 레벨 2(L2) 캐시의 로컬 서브세트(1904)와 함께 보여준다. 일 실시예에서, 명령어 디코더(1900)는 패킹된 데이터 명령어 세트 확장을 갖는 x86 명령어 세트를 지원한다. L1 캐시(1906)는 스칼라 및 벡터 유닛들로의 캐시 메모리에 대한 저-지연(low-latency) 액세스들을 허용한다. 일 실시예에서는 (설계를 단순화하기 위해) 스칼라 유닛(1908) 및 벡터 유닛(1910)이 별개의 레지스터 세트들(각각, 스칼라 레지스터들(1912) 및 벡터 레지스터들(1914))을 사용하고 이들 간에 이동되는 데이터는 메모리에 기입된 다음 레벨 1(L1) 캐시(1906)로부터 다시 판독되지만, 본 발명의 대안적 실시예들은 상이한 접근방식을 사용할 수 있다(예를 들어, 단일 레지스터 세트를 사용하거나, 또는 기입 및 다시 판독되지 않고 2개의 레지스터 파일들 사이에서 데이터가 이동되는 것을 허용하는 통신 경로를 포함함).

[0197] L2 캐시의 로컬 서브세트(1904)는, 프로세서 코어 당 하나씩인 개별 로컬 서브세트들로 분할되는 글로벌 L2 캐시의 일부이다. 각각의 프로세서 코어는 L2 캐시의 자신의 로컬 서브세트(1904)에 대한 직접 액세스 경로를 갖는다. 프로세서 코어에 의해 판독된 데이터는 자신의 L2 캐시 서브세트(1904)에 저장되며, 다른 프로세서 코어들이 그들 자신의 로컬 L2 캐시 서브세트들에 액세스하는 것과 병렬로 빠르게 액세스될 수 있다. 프로세서 코어에 의해 기입된 데이터는 자신의 L2 캐시 서브세트(1904)에 저장되며, 필요한 경우에는, 다른 서브세트들로부터 플래시된다. 링 네트워크는 공유 데이터에 대한 코히어런시(coherency)를 보장한다. 링 네트워크는 양-방향성이어서, 프로세서 코어들, L2 캐시들 및 다른 논리 블럭들 등의 에이전트들이 칩 내에서 상호 통신하는 것을 허용한다. 각각의 링 데이터-경로는 방향 당 1012 비트 폭이다.

[0198] 도 19b는 본 발명의 실시예들에 따른 도 19a에서의 프로세서 코어의 부분 확대도이다. 도 19b는, L1 캐시(1904)의 L1 데이터 캐시(1906A) 부분은 물론, 벡터 유닛(1910) 및 벡터 레지스터들(1914)에 관한 보다 많은 상세를 포함한다. 구체적으로, 벡터 유닛(1910)은 정수, 단일 정밀도 부동 및 이중 정밀도 부동 명령어들 중 하나 이상을 실행하는 16-폭 VPU(Vector Processing Unit)(16-폭 ALU(1928) 참조)이다. VPU는, 스위즐(swizzle) 유닛(1920)에 의한 레지스터 입력들의 스위즐링, 수치 변환 유닛들(1922A-B)에 의한 수치 변환 및 메모리 입력에 대한 복제 유닛(1924)에 의한 복제를 지원한다. 기입 마스크 레지스터들(1926)은 결과적인 벡터 기입들을 서술하는 것(predicating)을 허용한다.

[0199] **통합 메모리 컨트롤러 및 그래픽들을 갖는 프로세서**

[0200] 도 20은, 본 발명의 실시예들에 따라, 하나 보다 많은 코어를 가질 수 있고, 통합 메모리 컨트롤러를 가질 수 있고, 및 통합 그래픽을 가질 수 있는 프로세서(2000)의 블럭도이다. 도 20의 실선 박스들은 싱글 코어(2002A), 시스템 에이전트(2010), 하나 이상의 버스 컨트롤러 유닛들(2016)의 세트를 갖는 프로세서(2000)를 도시하는 한편, 옵션인 점선 박스들의 추가는 다수의 코어들(2002A-N), 시스템 에이전트 유닛(2010) 내의 하나 이상의 통합 메모리 컨트롤러 유닛(들)(2014)의 세트, 및 특수 목적 로직(2008)을 갖는 대안적인 프로세서(2000)를 도시한다.

[0201] 따라서, 프로세서(2000)의 상이한 구현들은: 1) 통합 그래픽 및/또는 과학적 (쓰루풋) 로직(하나 이상의 코어들을 포함할 수 있음)인 특수 목적 로직(2008) 및 하나 이상의 범용 코어들(예를 들어, 범용 순차적 코어들, 범용 비순차적 코어들, 이 두 가지의 조합)인 코어들(2002A-N)을 갖는 CPU; 2) 그래픽 및/또는 과학적 (쓰루풋) 컴퓨팅을 주로 대상으로 하는 다수의 특수 목적 코어들인 코어들(2002A-N)을 갖는 코프로세서; 및 3) 다수의 범용 순차적 코어들인 코어들(2002A-N)을 갖는 코프로세서를 포함할 수 있다. 따라서, 프로세서(2000)는 범용 프로세서, 코프로세서 또는 특수 목적 프로세서, 예를 들어 네트워크 또는 통신 프로세서, 압축 엔진, 그래픽 프로세서, GPGPU(General Purpose Graphics Processing Unit), 하이-쓰루풋 MIC(Many Integrated Core) 코프로세서(30개 이상의 코어를 포함함), 임베디드 프로세서, 또는 이와 유사한 것 등일 수 있다. 프로세서는 하나 이상의 칩들 상에 구현될 수 있다. 프로세서(2000)는, 예를 들어, BiCMOS, CMOS, 또는 NMOS 등의 다수의 프로세

스 기술들 중 임의의 것을 사용하여 하나 이상의 기판들의 일부가 될 수 있고 및/또는 이들 기판 상에 구현될 수 있다.

[0202] 메모리 계층구조는 코어들 내의 하나 이상의 레벨들의 캐시, 하나 이상의 공유 캐시 유닛들(2006)의 세트, 및 통합 메모리 컨트롤러 유닛들(2014)의 세트에 연결되는 외부 메모리(도시되지 않음)를 포함한다. 공유 캐시 유닛들(2006)의 세트는, 레벨 2(L2), 레벨 3(L3), 레벨 4(L4) 또는 다른 레벨의 캐시 등의 하나 이상의 중간 레벨 캐시들, 최종 레벨 캐시(LLC) 및/또는 이들의 조합들을 포함할 수 있다. 일 실시예에서는 링 기반 상호접속 유닛(2012)이 통합 그래픽 로직(2008), 공유 캐시 유닛들(2006)의 세트 및 시스템 에이전트 유닛(2010)/통합 메모리 컨트롤러 유닛(들)(2014)을 상호접속하지만, 대안 실시예들은 이러한 유닛들을 상호접속하는 임의 수의 공지된 기술들을 이용할 수 있다. 일 실시예에서, 하나 이상의 캐시 유닛들(2006)과 코어들(2002A-N) 사이에는 코히어런시가 유지된다.

[0203] 일부 실시예들에서, 코어들(2002A-N) 중 하나 이상은 멀티-쓰레딩이 가능하다. 시스템 에이전트(2010)는 코어들(2002A-N)을 조정 및 조작하는 컴포넌트들을 포함한다. 시스템 에이전트 유닛(2010)은 예를 들어 PCU(Power Control Unit) 및 디스플레이 유닛을 포함할 수 있다. PCU는 코어들(2002A-N) 및 통합 그래픽 로직(2008)의 전력 상태를 조절하는 데 필요한 로직 및 컴포넌트들이거나 이들을 포함할 수 있다. 디스플레이 유닛은 하나 이상의 외부 접속되는 디스플레이들을 구동하기 위한 것이다.

[0204] 코어들(2002A-N)은 아키텍처 명령어 세트와 관련하여 동종 또는 이종일 수 있다; 즉, 코어들(2002A-N) 중 둘 이상은 동일 명령어 세트를 실행할 수 있는 반면, 다른 코어들은 그 명령어 세트의 서브세트 또는 상이한 명령어 세트만을 실행할 수 있다.

[0205] **예시적인 컴퓨터 아키텍처**

[0206] 도 21-24는 예시적인 컴퓨터 아키텍처들의 블럭도들이다. 랩톱들, 데스크톱들, 핸드헬드 PC들, 퍼스널 디지털 어시스턴트들, 엔지니어링 워크스테이션들, 서버들, 네트워크 디바이스들, 네트워크 허브들, 스위치들, 임베디드 프로세서들, 디지털 신호 프로세서들(DSPs), 그래픽 디바이스들, 비디오 게임 디바이스들, 셋-톱 박스들, 마이크로 컨트롤러들, 셀 폰들, 휴대용 미디어 플레이어들, 핸드헬드 디바이스들 및 다양한 다른 전자 디바이스들에 대한 기술분야에 알려진 다른 시스템 설계들 및 구성들도 적합하다. 일반적으로, 본 명세서에 개시된 바와 같은 프로세서 및/또는 다른 실행 로직을 통합할 수 있는 매우 다양한 시스템들 또는 전자 디바이스들이 일반적으로 적합하다.

[0207] 이제 도 21을 참조하면, 본 발명의 일 실시예에 따른 시스템(2100)의 블럭도가 도시된다. 시스템(2100)은 하나 이상의 프로세서들(2110, 2115)을 포함할 수 있고, 이들은 컨트롤러 허브(2120)에 연결된다. 일 실시예에서, 컨트롤러 허브(2120)는 GMCH(Graphics Memory Controller Hub)(2190) 및 IOH(Input/Output Hub)(2150)(개별 칩들 상에 존재할 수 있음)를 포함하고; GMCH(2190)는, 메모리(2140) 및 코프로세서(2145)에 연결되는 메모리 및 그래픽 컨트롤러들을 포함하고; IOH(2150)는 I/O(Input/Output) 디바이스들(2160)을 GMCH(2190)에 연결한다. 대안적으로, 메모리 및 그래픽 컨트롤러들 중 하나 또는 양자 모두는 (본 명세서에서 개시되는 바와 같이) 프로세서 내에 통합되고, 메모리(2140) 및 코프로세서(2145)는 프로세서(2110), 및 IOH(2150)와 단일 칩에 있는 컨트롤러 허브(2120)에 직접 연결된다.

[0208] 추가적인 프로세서들(2115)의 옵션적 속성이 도 21에 파선들로 표시된다. 각각의 프로세서(2110, 2115)는 본 명세서에 개시되는 처리 코어들 중 하나 이상을 포함할 수 있고, 프로세서(2000)의 일부 버전일 수 있다.

[0209] 메모리(2140)는, 예를 들어, DRAM(Dynamic Random Access Memory), PCM(Phase Change Memory), 또는 이 둘의 조합일 수 있다. 적어도 하나의 실시예에 대해, 컨트롤러 허브(2120)는 FSB(Front Side Bus) 등의 멀티-드롭 버스, QPI(QuickPath Interconnect) 등의 지점-대-지점 인터페이스, 또는 유사한 접속(2195)을 통해 프로세서(들)(2110, 2115)와 통신한다.

[0210] 일 실시예에서, 코프로세서(2145)는 예를 들어 하이-쓰루풋 MIC 프로세서, 네트워크 또는 통신 프로세서, 압축 엔진, 그래픽 프로세서, GPGPU, 임베디드 프로세서 등과 같은 특수 목적 프로세서이다. 일 실시예에서, 컨트롤러 허브(2120)는 통합 그래픽 가속기를 포함할 수 있다.

[0211] 아키텍처, 마이크로아키텍처, 열, 전력 소비 특성들 등을 포함하는 장점의 다양한 메트릭들과 관련하여 물리적 리소스들(2110, 2115) 사이에는 다양한 차이점들이 존재할 수 있다.

[0212] 일 실시예에서, 프로세서(2110)는 일반적인 타입의 데이터 처리 연산들을 제어하는 명령어들을 실행한다. 명령

어들 내에는 코프로세서 명령어들이 내장될 수 있다. 프로세서(2110)는 이러한 코프로세서 명령어들을 부속된 코프로세서(2145)에 의해 실행되어야 하는 타입의 것으로 인식한다. 따라서, 프로세서(2110)는 이러한 코프로세서 명령어들(또는 코프로세서 명령어들을 나타내는 제어 신호들)을 코프로세서 버스 또는 다른 상호접속를 통해 코프로세서(2145)에 발행한다. 코프로세서(들)(2145)는 수신된 코프로세서 명령어들을 수락 및 실행한다.

[0213] 이제, 도 22을 참조하면, 본 발명의 일 실시예에 따른 제1의 보다 구체적인 예시적인 시스템(2200)의 블럭도가 도시된다. 도 22에 도시된 바와 같이, 멀티프로세서 시스템(2200)은 지점-대-지점 상호접속 시스템이며, 지점-대-지점 상호접속(2250)을 통해 연결되는 제1 프로세서(2270) 및 제2 프로세서(2280)를 포함한다. 프로세서들(2270, 2280) 각각은 프로세서(2000)의 일부 버전일 수 있다. 본 발명의 일 실시예에서, 프로세서들(2270, 2280)은 각각 프로세서들(2110, 2115)인 한편, 코프로세서(2238)는 코프로세서(2145)이다. 다른 실시예에서는, 프로세서들(2270, 2280)이 각각 프로세서(2110) 및 코프로세서(2145)이다.

[0214] 프로세서들(2270, 2280)은 각각 IMC(Integrated Memory Controller) 유닛들(2272, 2282)을 포함하는 것으로 도시된다. 프로세서(2270)는 또한 그의 버스 컨트롤러 유닛들의 일부로서 P-P(Point-to-Point) 인터페이스들(2276, 2278)을 포함한다; 유사하게 제2 프로세서(2280)는 P-P 인터페이스들(2286, 2288)을 포함한다. 프로세서들(2270, 2280)은 P-P 인터페이스(Point-to-Point) 회로들(2278, 2288)을 이용하여 P-P 인터페이스(2250)를 통해 정보를 교환할 수 있다. 도 22에 도시된 바와 같이, IMC들(2272 및 2282)은 프로세서들을 각자의 메모리, 즉 메모리(2232) 및 메모리(2234)에 연결하며, 이들 메모리는 각 프로세서에 국부적으로 부속되는 메인 메모리의 일부일 수 있다.

[0215] 프로세서들(2270, 2280)은 지점 대 지점 인터페이스 회로들(2276, 2294, 2286, 2298)을 사용하여 개별 P-P 인터페이스들(2252, 2254)을 통해 칩셋(2290)과 정보를 각각 교환할 수 있다. 칩셋(2290)은 고-성능 인터페이스(2239)를 통해 코프로세서(2238)와 정보를 선택적으로 교환할 수 있다. 일 실시예에서, 코프로세서(2238)는 예를 들어 하이-쓰루풋 MIC 프로세서, 네트워크 또는 통신 프로세서, 압축 엔진, 그래픽 프로세서, GPGPU, 임베디드 프로세서 등과 같은 특수 목적 프로세서이다.

[0216] 공유된 캐시(도시되지 않음)는 어느 한 프로세서에 포함되거나, 양자 모두의 프로세서의 외부이지만 여전히 P-P 상호접속을 통해 프로세서들과 접속될 수 있어서, 프로세서가 저 전력 모드에 놓이는 경우 어느 한쪽 또는 양자 모두의 프로세서의 로컬 캐시 정보가 공유된 캐시에 저장될 수 있다.

[0217] 칩셋(2290)은 인터페이스(2296)를 통해 제1 버스(2216)에 연결될 수 있다. 일 실시예에서, 제1 버스(2216)는 PCI(Peripheral Component Interconnect) 버스일 수 있거나, 또는 PCI 익스프레스 버스 또는 다른 3세대 I/O 상호접속 버스 등의 버스일 수 있지만, 본 발명의 범위가 이에 제한되는 것은 아니다.

[0218] 도 22에 도시된 바와 같이, 다양한 I/O 디바이스들(2214)이 제1 버스(2216)에 연결될 수 있으며, 이와 함께 버스 브릿지(2218)가 제1 버스(2216)를 제2 버스(2220)에 연결한다. 일 실시예에서는, 코프로세서들, 하이-쓰루풋 MIC 프로세서들, GPGPU들, 가속기들(예를 들어, 그래픽 가속기 또는 DSP(Digital Signal Processing) 유닛 등), 필드 프로그래머블 게이트 어레이들 또는 임의의 다른 프로세서 등의 하나 이상의 추가적인 프로세서(들)(2215)가 제1 버스(2216)에 연결된다. 일 실시예에서, 제2 버스(2220)는 LPC(Low Pin Count) 버스일 수 있다. 일 실시예에서는, 예를 들어 키보드 및/또는 마우스(2222), 통신 디바이스들(2227) 및 명령어들/코드 및 데이터(2230)를 포함할 수 있는 디스크 드라이브 또는 기타 대용량 스토리지 디바이스 등의 저장 유닛(2228)을 포함하는 다양한 디바이스들이 제2 버스(2220)에 연결될 수 있다. 또한, 오디오 I/O(2224)가 제2 버스(2220)에 연결될 수 있다. 다른 아키텍처들도 가능하다는 점에 주의한다. 예를 들어, 도 22의 지점-대-지점 아키텍처 대신에, 시스템은 멀티-드롭 버스 또는 다른 그러한 아키텍처를 구현할 수 있다.

[0219] 이제, 도 23을 참조하면, 본 발명의 일 실시예에 따른 제2의 보다 구체적인 예시적인 시스템(2300)의 블럭도가 도시된다. 도 22 및 23에서 동일한 엘리먼트들은 동일한 참조 번호들을 가지며, 도 22의 특정 양상들은 도 23의 다른 양상들을 모호하게 하는 것을 회피하기 위해 도 23으로부터 생략되었다.

[0220] 도 23은 프로세서들(2270, 2280)이 각각 통합 메모리 및 I/O 제어 로직("CL")(2272, 2282)을 포함할 수 있다는 점을 도시한다. 따라서, CL(2272, 2282)은 통합 메모리 컨트롤러 유닛들을 포함하며, I/O 제어 로직을 포함한다. 도 23은 메모리들(2232, 2234)이 CL(2272, 2282)에 연결될 뿐만 아니라, I/O 디바이스들(2314) 또한 제어 로직(2272, 2282)에 연결된다는 것을 도시한다. 레거시 I/O 디바이스들(2315)은 칩셋(2290)에 연결된다.

[0221] 이제, 도 24를 참조하면, 본 발명의 일 실시예에 따른 SoC(2400)의 블럭도가 도시된다. 도 20에서의 유사한 엘리먼트들은 동일한 참조 번호를 갖는다. 또한, 점선 박스는 더욱 개선된 SoC들에 관한 선택적 특징들이다. 도

24에서, 상호접속 유닛(들)(2402)은: 하나 이상의 코어들(202A-N)의 세트 및 공유 캐시 유닛(들)(2006)을 포함하는 애플리케이션 프로세서(2410); 시스템 에이전트 유닛(2010); 버스 컨트롤러 유닛(들)(2016); 통합 메모리 컨트롤러 유닛(들)(2014); 통합 그래픽 로직, 이미지 프로세서, 오디오 프로세서 및 비디오 프로세서를 포함할 수 있는 하나 이상의 코프로세서들(2420)의 세트; SRAM(Static Random Access Memory) 유닛(2430); DMA(Direct Memory Access) 유닛(2432); 및 하나 이상의 외부 디스플레이들에 연결하기 위한 디스플레이 유닛(2440)에 연결된다. 일 실시예에서, 코프로세서(들)(2420)는, 예를 들어, 네트워크 또는 통신 프로세서, 압축 엔진, GPGPU, 하이-쓰루풋 MIC 프로세서, 임베디드 프로세서 등과 같은 특수 목적 프로세서를 포함한다.

[0222] 본 명세서에 개시되는 메커니즘들의 실시예들은 하드웨어, 소프트웨어, 펌웨어 또는 이러한 구현 접근방식들의 조합으로 구현될 수 있다. 본 발명의 실시예들은, 적어도 하나의 프로세서, 스토리지 시스템(휘발성 및 불휘발성 메모리 및/또는 스토리지 엘리먼트들을 포함함), 적어도 하나의 입력 디바이스, 및 적어도 하나의 출력 디바이스를 포함하는 프로그래머블 시스템들 상에서 실행되는 컴퓨터 프로그램들 또는 프로그램 코드로서 구현될 수 있다.

[0223] 도 22에 도시된 코드(2230) 등의 프로그램 코드는 입력 명령어들에 적용되어 본 명세서에 개시되는 평션들을 수행하고 출력 정보를 생성하는 수 있다. 출력 정보는 알려진 방식으로 하나 이상의 출력 디바이스에 적용될 수 있다. 본 출원의 목적을 위해, 처리 시스템은, 예를 들어, DSP(Digital Signal Processor), 마이크로컨트롤러, ASIC(Application Specific Integrated Circuit) 또는 마이크로프로세서 등의 프로세서를 갖는 임의의 시스템을 포함한다.

[0224] 프로그램 코드는 하이 레벨 절차적 또는 객체 지향적 프로그래밍 언어로 구현되어 처리 시스템과 통신할 수 있다. 프로그램 코드는, 또한, 요구되는 경우, 어셈블리 또는 기계 언어로 구현될 수 있다. 사실상, 본 명세서에 개시되는 메커니즘들이 임의의 특정 프로그래밍 언어로 범위가 제한되는 것은 아니다. 어느 경우에나, 언어는 컴파일되거나 또는 해석되는 언어일 수 있다.

[0225] 적어도 하나의 실시예의 하나 이상의 양상은, 머신에 의해 판독될 때 머신으로 하여금 본 명세서에서 개시되는 기술들을 수행하는 로직을 제조하게 하는, 프로세서 내의 다양한 로직을 표현하는, 머신 판독-가능 매체 상에 저장되는 대표적인 명령어들에 의해 구현될 수 있다. "IP 코어들"로서 알려진 그러한 표현들은 유형의 머신 판독가능 매체 상에 저장될 수 있으며, 다양한 고객들 또는 제조 설비에 공급되어, 로직 또는 프로세서를 실제로 제작하는 제조 머신들로 로드될 수 있다.

[0226] 이러한 머신-판독가능 스토리지 매체들은, 하드 디스크들, 플로피 디스크들, 광 디스크들, CD-ROM들(Compact Disk Read-Only Memories), CD-RW들(Compact Disk ReWritable's) 및 광자기 디스크들 포함하는 임의의 다른 타입의 디스크들, ROM들(Read-Only Memories), DRAM들(Dynamic Random Access Memories), SRAM들(Static Random Access Memories) 등의 RAM들(Random Access Memories), EPROM들(Erasable Programmable Read-Only Memories), 플래시 메모리들, EEPROM들(Electrically Erasable Programmable Read-Only Memories), PCM(Phase Change Memory) 등의 반도체 디바이스, 자기 또는 광학 카드, 또는 전자적 명령어들을 저장하기에 적합한 임의의 다른 타입의 매체와 같은 스토리지 매체를 포함하는 머신 또는 디바이스에 제조되거나 또는 형성되는 물품들의 비-일시적이고 유형인 배열들을 포함할 수 있고, 이에 제한되는 것은 아니다.

[0227] 따라서, 본 발명의 실시예들은, 또한, 명령어들을 포함하거나, 또는 본 명세서에 개시되는 구조들, 회로들, 장치들, 프로세서들 및/또는 시스템 특징들을 정의하는, HDL(Hardware Description Language) 등의 설계 데이터를 포함하는 비-일시적이고 유형인 머신 판독가능 매체를 포함한다. 이러한 실시예들은 또한 프로그램 제품들이라고 할 수 있다.

[0228] **에뮬레이션(바이너리 변환, 코드 모핑 등을 포함함)**

[0229] 일부 경우에는, 명령어 변환기가 소스 명령어 세트로부터 타겟 명령어 세트로 명령어를 변환하는데 사용될 수 있다. 예를 들어, 명령어 변환기는 코어에 의해 처리될 하나 이상의 다른 명령어들로 명령어를 (예를 들어, 정적 바이너리 해석, 동적 컴필레이션을 포함하는 동적 바이너리 해석을 이용하여) 해석하거나, 모프하거나, 에뮬레이트하거나, 또는 다른 방식으로 변환할 수 있다. 명령어 변환기는 소프트웨어, 하드웨어, 펌웨어, 또는 이들의 조합으로 구현될 수 있다. 명령어 변환기는 온 프로세서(on processor), 오프 프로세서(off processor), 또는 부분 온 및 부분 오프 프로세서(part on and part off processor)일 수 있다.

[0230] 도 25는 본 발명의 실시예들에 따라 소스 명령어 세트 내의 바이너리 명령어들을 타겟 명령어 세트 내의 바이너리 명령어들로 변환하기 위한 소프트웨어 명령어 변환기의 사용을 대조하는 블럭도이다. 도시된 실시예에서,

명령어 변환기는 소프트웨어 명령어 변환기이지만, 대안적으로 명령어 변환기가 소프트웨어, 펌웨어, 하드웨어, 또는 이들의 다양한 조합들로 구현될 수 있다. 도 25는 하이 레벨 언어(2502)의 프로그램을 x86 컴파일러(2504)를 사용하여 컴파일하여, 적어도 하나의 x86 명령어 세트 코어를 갖는 프로세서(2516)에 의해 선천적으로 실행될 수 있는 x86 바이너리 코드(2506)를 생성할 수 있다는 것을 도시한다. 적어도 하나의 x86 명령어 세트 코어를 갖는 프로세서(2516)는, 적어도 하나의 x86 명령어 세트 코어를 갖는 인텔 프로세서와 실질적으로 동일한 결과를 달성하기 위해서, (1) 인텔 x86 명령어 세트 코어의 명령어 세트의 상당 부분 또는 (2) 적어도 하나의 x86 명령어 세트 코어를 갖는 인텔 프로세서 상에서 실행되는 것을 목적으로 하는 오브젝트 코드 버전들의 애플리케이션들 또는 다른 소프트웨어를 호환가능하게 실행하거나 또는 다른 방식으로 처리함으로써, 적어도 하나의 x86 명령어 세트 코어를 갖는 인텔 프로세서와 실질적으로 동일한 평션을 수행할 수 있는 임의의 프로세서를 나타낸다. x86 컴파일러(2504)는 추가적인 링크 처리(linkage processing)를 갖거나 갖지 않고서 적어도 하나의 x86 명령어 세트 코어를 갖는 프로세서(2516) 상에서 실행될 수 있는 x86 바이너리 코드(2506)(예를 들어, 오브젝트 코드)를 생성하도록 동작될 수 있는 컴파일러를 나타낸다. 유사하게, 도 25는 하이 레벨 언어(2502)의 프로그램을 대안적인 명령어 세트 컴파일러(2508)를 사용하여 컴파일하여, 적어도 하나의 x86 명령어 세트 코어를 갖지 않는 프로세서(2514)(예를 들어, 캘리포니아주 서니베일의 MIPS 테크놀로지스의 MIPS 명령어 세트를 실행하고/실행하거나 캘리포니아주 서니베일의 ARM 홀딩스의 ARM 명령어 세트를 실행하는 코어들을 갖는 프로세서)에 의해 선천적으로 실행될 수 있는 대안적인 명령어 세트 바이너리 코드(2510)를 생성할 수 있다는 점을 도시한다. 명령어 변환기(2512)는 x86 바이너리 코드(2506)를, x86 명령어 세트 코어를 갖지 않는 프로세서(2514)에 의해 선천적으로 실행될 수 있는 코드로 변환하는데 사용된다. 이러한 변환된 코드는 대안적인 명령어 세트 바이너리 코드(2510)와 동일할 가능성이 낮은데, 그 이유는 이를 행할 수 있는 명령어 변환기를 제조하기 어렵기 때문이다; 그러나, 변환된 코드는 일반적인 연산을 달성할 것이며, 대안적인 명령어 세트로부터의 명령어들로 이루어질 것이다. 따라서, 명령어 변환기(2512)는, 에뮬레이션, 시뮬레이션 또는 임의의 다른 프로세스를 통해 x86 명령어 세트 프로세서 또는 코어를 갖지 않는 프로세서 또는 다른 전자 디바이스가 x86 바이너리 코드(2506)를 실행하는 것을 허용하는 소프트웨어, 펌웨어, 하드웨어 또는 이들의 조합을 나타낸다.

[0231] 도 3-4 중 임의의 것에 대해 개시되는 컴포넌트들, 특징들 및 상세들은 또한 도 1-2 중 임의의 것에 선택적으로 사용될 수 있다. 도 6-13 중 임의의 것에 대해 개시되는 컴포넌트들, 특징들 및 상세들은 또한 도 1 또는 5 중 임의의 것에 선택적으로 사용될 수 있다. 또한, 본 명세서에 개시되는 장치들 중 임의의 것에 대해 본 명세서에 개시되는 컴포넌트들, 특징들 및 상세들은 또한, 실시예들에서 그러한 장치에 의해 및/또는 그러한 장치를 가지고 수행될 수 있는 본 명세서에 개시되는 방법들 중 임의의 것에 선택적으로 사용될 수 있고 및/또는 이에 적용될 수 있다. 본 명세서에 개시되는 프로세서들 중 임의의 것은 본 명세서에 개시되는 컴퓨터 시스템들 또는 다른 시스템들 중 임의의 것에 포함될 수 있다. 일부 실시예들에서, 명령어들은 본 명세서에 개시되는 명령어 포맷들의 특징들 또는 상세들을 가질 수 있지만, 이것이 요구되는 것은 아니다.

[0232] 설명 및 청구범위에서, "연결되는(coupled)" 및/또는 "접속되는(connected)"이라는 용어들이, 이들의 파생어와 함께, 사용될 수 있다. 이들 용어가 상호 동의어로서 의도되는 것은 아니다. 오히려, 실시예들에서, "접속되는"이란 2 이상의 엘리먼트들이 상호 직접적인 물리적 및/또는 전기적 접촉을 이루고 있다는 점을 나타내는데 사용될 수 있다. "연결되는"이란 2 이상의 엘리먼트들이 상호 직접적인 물리적 및/또는 전기적 접촉을 이루고 있다는 점을 의미할 수 있다. 그러나, "연결되는"이란 2 이상의 엘리먼트들이 상호 직접적인 접촉을 이루고 있지는 않지만, 여전히 상호 협력하거나 상호작용하고 있다는 점을 또한 의미할 수 있다. 예를 들어, 실행 유닛은 하나 이상의 중간 컴포넌트들을 통해 레지스터 및/또는 디코드 유닛과 연결될 수 있다. 도면들에서, 화살표들은 접속들 및 연결들을 보여주는데 사용된다.

[0233] "및/또는(and/or)"이란 용어가 사용되었을 수 있다. 본 명세서에서 사용되는 바와 같이, "및/또는"이란 용어는 하나 또는 나머지 또는 양자 모두를 의미한다(예를 들어, A 및/또는 B는 A 또는 B 또는 A와 B 양자 모두를 의미한다).

[0234] 위의 설명에서는, 실시예들의 철저한 이해를 제공하기 위하여 특정 상세들이 제시되었다. 그러나, 다른 실시예들은 이들 특정 상세들 중 일부가 없어도 실시될 수 있다. 본 발명의 범위는 위에 제공되는 특정 예들에 의해서 결정되어야 하는 것이 아니라, 이하 청구범위에 의해서만 결정되어야 한다. 다른 경우들에서, 잘 알려진 회로들, 구조들, 장치들, 및 동작들은 설명의 이해를 모호하게 하는 것을 회피하기 위해 블럭도 형태로 및/또는 상세 없이 도시되었다. 적절한 것으로 간주되는 경우, 참조 번호들 또는 참조 번호들의 끝 부분들은, 달리 특정되거나 명백히 분명하지 않는 한, 선택적으로 유사하거나 동일한 특성들을 가질 수 있는, 대응하는 또는 유사한 엘리먼트들을 나타내기 위해 도면들 사이에서 반복되었다.

[0235] 특정 연산들은 하드웨어 컴포넌트들에 수행될 수 있거나, 또는 머신-실행가능 또는 회로-실행가능 명령어들로 구현될 수 있고, 이는 머신, 회로 또는 하드웨어 컴포넌트(예를 들어, 프로세서, 프로세서의 일부, 회로 등)가 연산들을 수행하는 명령어들로 프로그램되도록 하는데 및/또는 이러한 결과를 초래하는데 사용될 수 있다. 연산들은 또한 하드웨어 및 소프트웨어의 조합에 의해 선택적으로 수행될 수 있다. 프로세서, 머신 또는 하드웨어는 특수 또는 특정 회로를 포함할 수 있거나, 또는 기타 로직(예를 들어, 펌웨어 및/또는 소프트웨어와 잠재적으로 조합되는 하드웨어)이 명령어를 실행 및/또는 처리하여 명령어에 응답하여 결과를 저장하도록 동작될 수 있다.

[0236] 일부 실시예들은 머신-판독가능 매체를 포함하는 제조 물품(예를 들어, 컴퓨터 프로그램 제품)을 포함한다. 매체는 머신에 의해 판독가능한 형태로 정보를 제공하는, 예를 들어, 저장하는 메커니즘을 포함할 수 있다. 머신-판독가능 매체는 머신에 의해 실행되는 경우 및/또는 실행될 때 머신으로 하여금 본 명세서에 개시되는 하나 이상의 연산들, 방법들, 또는 기술들을 수행하게 하고 및/또는 수행하는 결과가 머신에 생기게 하도록 동작가능한 명령어들의 시퀀스를 제공하거나 저장할 수 있다. 머신-판독가능 매체는 본 명세서에 개시되는 명령어들의 실시예들 중 하나 이상을 저장하거나 또는 다른 방식으로 제공할 수 있다.

[0237] 일부 실시예들에서, 머신-판독가능 매체는 유형의 및/또는 비-일시적 머신-판독가능 저장 매체를 포함할 수 있다. 예를 들어, 유형의 및/또는 비-일시적 머신-판독가능 저장 매체는 플로피 디스켓, 광 스토리지 매체, 광 디스크, 광 데이터 스토리지 디바이스, CD-ROM, 자기 디스크, 광자기 디스크, ROM(Read Only Memory), PROM(Programmable ROM), EPROM(Erasable-and-Programmable ROM), EEPROM(Electrically-Erasable-and-Programmable ROM), RAM(Random Access Memory), SRAM(Static-RAM), DRAM(Dynamic-RAM), 플래시 메모리, 상 변화 메모리, 상 변화 데이터 스토리지 재료, 불휘발성 메모리, 불휘발성 데이터 스토리지 디바이스, 비-일시적 메모리, 비-일시적 데이터 스토리지 디바이스 등을 포함할 수 있다. 비-일시적 머신-판독가능 스토리지 매체는 일시적 전파되는 신호(transitory propagated signal)로 이루어지지 않는다.

[0238] 적절한 머신들의 예들은, 범용 프로세서, 특수 목적 프로세서, 명령어 처리 장치, 디지털 로직 회로, 집적 회로 등을 포함하지만, 이에 제한되는 것은 아니다. 적절한 머신들의 다른 예들은, 프로세서, 명령어 처리 장치, 디지털 로직 회로 또는 집적 회로를 포함하는 컴퓨팅 디바이스 또는 다른 전자 디바이스를 포함한다. 이러한 컴퓨팅 디바이스들 및 전자 디바이스들의 예들은, 데스크톱 컴퓨터들, 랩톱 컴퓨터들, 노트북 컴퓨터들, 태블릿 컴퓨터들, 넷북들, 스마트폰들, 셀룰러 폰들, 서버들, 네트워크 디바이스들(예를 들어, 라우터들 및 스위치들), MID들(Mobile Internet devices), 미디어 플레이어들, 랩톱 컴퓨터들, 노트북 컴퓨터들, 데스크톱 컴퓨터들, 스마트 텔레비전들, 넷톱들, 셋-톱 박스들 및 비디오 게임 컨트롤러들을 포함하지만, 이에 제한되는 것은 아니다.

[0239] 본 명세서 전반에 걸쳐서 예를 들어, "일 실시예", "실시예", "하나 이상의 실시예들", "일부 실시예들"에 대한 언급은, 특정 특징이 본 발명의 실시에 포함될 수 있지만 반드시 그러할 것이 요구되는 것은 아니라는 점을 나타낸다. 유사하게, 본 명세서를 간소화하고 다양한 본 발명의 양상들의 이해를 도울 목적으로, 설명에서는 다양한 특징들이 때때로 단일 실시예, 도면, 또는 그의 설명에서 함께 그룹화된다. 그러나, 이러한 개시의 방법이, 본 발명은 각 청구범위에 명백하게 기재된 것보다 많은 특징들을 요구하는 의도를 반영하는 것으로서 해석되어서는 안 된다. 오히려, 이하 청구범위들이 반영하는 바에 따라, 본 발명의 양상들은 단일 개시된 실시예의 모든 특징보다 적게 놓인다. 따라서, 상세한 설명에 후속하는 청구범위들은 이로써 본 상세한 설명에 명백하게 통합되고, 각 청구범위는 본 발명의 개별 실시예로서 자립한다.

[0240] **실시예들**

[0241] 이하 예들은 다른 실시예들과 관련된다. 예들에서 상세들은 하나 이상의 실시예들 어느 곳에서도 사용될 수 있다.

[0242] 예 1은, 복수의 패킹된 데이터 레지스터들 및 명령어를 디코드하는 디코드 유닛을 포함하는 프로세서이다. 명령어는, 적어도 4개의 데이터 엘리먼트들의 제1 세트를 포함하는 제1 소스 패킹된 데이터를 나타내고, 적어도 4개의 데이터 엘리먼트들의 제2 세트를 포함하는 제2 소스 패킹된 데이터를 나타내며, 목적지 스토리지 위치를 나타낸다. 실행 유닛은 패킹된 데이터 레지스터들 및 디코드 유닛과 연결된다. 실행 유닛은, 명령어에 응답하여, 목적지 스토리지 위치에 결과 패킹된 데이터를 저장한다. 결과 패킹된 데이터는 적어도 4개의 인덱스들을 포함한다. 인덱스들은 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서의 대응 데이터 엘리먼트 위치들을 식별한다. 인덱스들은 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서의 대응 데이터 엘리먼트들의 정렬된 순서를 나타내는 결과 패킹된 데이터에서의 위치들에 저장된다.

[0243] 예 2는 예 1의 프로세서를 포함하고, 실행 유닛은, 인덱스들 각각이 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터 각각에서의 대응 데이터 엘리먼트 위치를 식별하는 결과 패킹된 데이터를 저장한다. 또한, 여기서 실행 유닛은, 명령어에 응답하여, 적어도 4개의 마스크 엘리먼트들을 갖는 결과 마스크를 저장하고, 각각의 마스크 엘리먼트는 인덱스들 중 상이한 것에 대응한다. 또한, 여기서 각각의 마스크 엘리먼트는 대응 인덱스에 대한 데이터 엘리먼트 위치가 제1 소스 패킹된 데이터에 존재하는지 아니면 제2 소스 패킹된 데이터에 존재하는지를 나타낸다.

[0244] 예 3은 예 2의 프로세서를 포함하고, 결과 마스크를 저장하는 마스크 레지스터를 더 포함한다. 명령어는, 패킹된 데이터 연산을 서술하는 서술 피연산자로서 결과 마스크를 나타낼 수 있는 제2 명령어를 포함하는 명령어 세트에 포함된다.

[0245] 예 4는 예 1의 프로세서를 포함하고, 여기서 실행 유닛은, 인덱스들 각각이 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터 중 하나에서의 대응 싱글 데이터 엘리먼트를 식별하는 결과 패킹된 데이터를 저장한다.

[0246] 예 5는 예 1 내지 4 중 어느 하나의 프로세서를 포함하고, 여기서 실행 유닛은, 명령어에 응답하여, 명령어에 의해 나타나는 제2 목적지 스토리지 위치에 제2 결과 패킹된 데이터를 저장한다. 제2 결과 패킹된 데이터는, 정렬된 순서를 반영하는 제2 결과 패킹된 데이터의 위치들에 저장된 인덱스들에 대응하는 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터로부터의 데이터 엘리먼트들을 포함한다.

[0247] 예 6은 예 1 내지 4 중 어느 하나의 프로세서를 포함하고, 여기서 디코드 유닛은, 명령어에 대해 정렬된 순서로 존재할 것으로 추정되는 적어도 4개의 데이터 엘리먼트들을 갖는 제1 소스 패킹된 데이터를 나타내고, 명령어에 대해 정렬된 순서로 존재할 것으로 추정되는 적어도 4개의 데이터 엘리먼트들을 갖는 제2 소스 패킹된 데이터를 나타내는 명령어를 디코드한다.

[0248] 예 7은 예 1 내지 4 중 어느 하나의 프로세서를 포함하고, 여기서 디코드 유닛은, 명령어에 대해 정렬된 순서로 존재할 것으로 추정되지 않는 적어도 4개의 데이터 엘리먼트들을 갖는 제1 소스 패킹된 데이터를 나타내고, 명령어에 대해 정렬된 순서로 존재할 것으로 추정되지 않는 적어도 4개의 데이터 엘리먼트들을 갖는 제2 소스 패킹된 데이터를 나타내는 명령어를 디코드한다.

[0249] 예 8은 예 1 내지 4 중 어느 하나의 프로세서를 포함하고, 여기서 실행 유닛은, 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터의 모든 데이터 엘리먼트들 중 최소 1/2을 포함하는 대응 데이터 엘리먼트들의 정렬된 순서를 나타내는 위치들에 인덱스들이 저장되는 결과 패킹된 데이터를 저장한다.

[0250] 예 9는 예 1 내지 4 중 어느 하나의 프로세서를 포함하고, 여기서 실행 유닛은, 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터의 모든 데이터 엘리먼트들 중 최대 1/2을 포함하는 대응 데이터 엘리먼트들의 정렬된 순서를 나타내는 위치들에 인덱스들이 저장되는 결과 패킹된 데이터를 저장한다.

[0251] 예 10은 예 1 내지 4 중 어느 하나의 프로세서를 포함하고, 여기서 디코드 유닛은, 각각 32 비트 및 64 비트 중 하나를 갖는 적어도 8개의 데이터 엘리먼트들을 포함하는 제1 소스 패킹된 데이터를 나타내는 명령어를 디코드한다.

[0252] 예 11은 명령어를 수신하는 단계를 포함하는 프로세서에서의 방법이다. 명령어는, 적어도 4개의 데이터 엘리먼트들의 제1 세트를 포함하는 제1 소스 패킹된 데이터를 나타내고, 적어도 4개의 데이터 엘리먼트들의 제2 세트를 포함하는 제2 소스 패킹된 데이터를 나타내며, 목적지 스토리지 위치를 나타낸다. 명령어에 응답하여 목적지 스토리지 위치에 결과 패킹된 데이터를 저장한다. 결과 패킹된 데이터는 적어도 4개의 인덱스들을 포함한다. 인덱스들은 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서의 대응 데이터 엘리먼트 위치들을 식별한다. 인덱스들은 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서의 대응 데이터 엘리먼트들의 정렬된 순서를 나타내는 결과 패킹된 데이터에서의 위치들에 저장된다.

[0253] 예 12는 예11의 방법을 포함하고, 수신하는 단계는, 정렬된 순서로 적어도 4개의 데이터 엘리먼트들을 갖는 제1 소스 패킹된 데이터를 나타내는 명령어를 수신하는 단계를 포함한다.

[0254] 예 13은 예 11 내지 12 중 어느 하나의 방법을 포함하고, 여기서 결과 패킹된 데이터를 저장하는 단계는, 인덱스들 각각이 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터 중 하나에서의 대응 싱글 데이터 엘리먼트를 식별하는 결과 패킹된 데이터를 저장하는 단계를 포함한다.

[0255] 예 14는, 복수의 데이터 레지스터들 및 명령어를 디코드하는 디코드 유닛을 포함하는 프로세서이다. 명령어는, 정렬된 순서로 존재하지 않는 적어도 4개의 데이터 엘리먼트들을 포함하는 소스 패킹된 데이터를 나타내고, 목

적지 스토리지 위치를 나타낸다. 실행 유닛은, 패킹된 데이터 레지스터들 및 디코드 유닛과 연결된다. 실행 유닛은, 명령어에 응답하여, 목적지 스토리지 위치에 결과 패킹된 데이터를 저장한다. 결과 패킹된 데이터는 적어도 4개의 인덱스들을 포함한다. 인덱스들은 소스 패킹된 데이터에서의 대응 데이터 엘리먼트들을 식별한다. 인덱스들은 소스 패킹된 데이터에서의 대응 데이터 엘리먼트들의 정렬된 순서를 나타내는 결과 패킹된 데이터에서의 위치들에 저장된다.

[0256] 예 15는 예 14의 프로세서를 포함하고, 여기서 실행 유닛은, 명령어에 응답하여, 명령어에 의해 나타나는 제2 목적지 스토리지 위치에 제2 결과 패킹된 데이터를 저장하고, 제2 결과 패킹된 데이터는, 정렬된 순서를 반영하는 제2 결과 패킹된 데이터의 위치들에 저장된 대응 데이터 엘리먼트들을 포함한다.

[0257] 예 16은 예 14 내지 15 중 어느 하나의 프로세서를 포함하고, 여기서 결과 패킹된 데이터는 소스 패킹된 데이터에서의 모든 데이터 엘리먼트들에 대응하는 인덱스들을 포함한다.

[0258] 예 17은 예 14 내지 15 중 어느 하나의 프로세서를 포함하고, 여기서 디코드 유닛은, 각각 32 비트 및 64 비트 중 하나를 갖는 적어도 8개의 데이터 엘리먼트들을 포함하는 소스 패킹된 데이터를 나타내는 명령어를 디코드한다.

[0259] 예 18은 명령어를 수신하는 단계를 포함하는 프로세서에서의 방법이고, 명령어는, 정렬된 순서로 존재하지 않는 적어도 4개의 데이터 엘리먼트들을 포함하는 소스 패킹된 데이터를 나타내고, 목적지 스토리지 위치를 나타낸다. 명령어에 응답하여 목적지 스토리지 위치에 결과 패킹된 데이터를 저장한다. 결과 패킹된 데이터는 적어도 4개의 인덱스들을 포함한다. 인덱스들은 소스 패킹된 데이터에서의 대응 데이터 엘리먼트들을 식별한다. 인덱스들은 소스 패킹된 데이터에서의 대응 데이터 엘리먼트들의 정렬된 순서를 나타내는 결과 패킹된 데이터에서의 위치들에 저장된다.

[0260] 예 19는 예 18의 방법을 포함하고, 이 방법은, 명령어에 의해 나타나는 제2 목적지 스토리지 위치에 제2 결과 패킹된 데이터를 저장하는 단계를 더 포함하고, 제2 결과 패킹된 데이터는 정렬된 순서를 반영하는 위치들에 저장되는 대응 데이터 엘리먼트들을 포함한다.

[0261] 예 20은 예 18 내지 19 중 어느 하나의 방법을 포함하고, 여기서 수신하는 단계는, 각각 32 비트 및 64 비트 중 하나를 갖는 적어도 8개의 데이터 엘리먼트들을 갖는 소스 패킹된 데이터를 나타내는 명령어를 수신하는 단계를 포함하고, 저장하는 단계는, 소스 패킹된 데이터에서의 모든 데이터 엘리먼트들에 대응하는 인덱스들을 포함하는 결과 패킹된 데이터를 저장하는 단계를 포함한다.

[0262] 예 21은, 상호접속 및 상호접속과 연결되는 프로세서를 포함하는, 명령어들을 처리하는 시스템이다. 프로세서는, 적어도 4개의 데이터 엘리먼트들의 제1 세트를 포함하는 제1 소스 패킹된 데이터를 나타내고, 적어도 4개의 데이터 엘리먼트들의 제2 세트를 포함하는 제2 소스 패킹된 데이터를 나타내며, 목적지 레지스터를 나타내는 명령어를 수신한다. 프로세서는, 명령어에 응답하여, 목적지 레지스터에 결과 패킹된 데이터를 저장한다. 결과 패킹된 데이터는 적어도 4개의 인덱스들을 포함한다. 인덱스들은 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서의 대응 데이터 엘리먼트 위치들을 식별한다. 인덱스들은 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서의 대응 데이터 엘리먼트들의 정렬된 순서를 나타내는 결과 패킹된 데이터에서의 위치들에 저장된다. DRAM(Dynamic Random Access Memory)이 상호접속과 연결된다. DRAM은 결과 패킹된 데이터의 인덱스들을 사용하여 데이터를 정렬하는 알고리즘을 선택적으로 저장한다.

[0263] 예 22는 청구항 21의 시스템을 포함하고, 여기서 프로세서는, 인덱스들 각각이 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터 중 하나에서의 대응 싱글 데이터 엘리먼트를 식별하는 결과 패킹된 데이터를 저장한다.

[0264] 예 23은 비-일시적 머신-판독가능 스토리지 매체를 포함하는 제조 물품이며, 비-일시적 머신-판독가능 스토리지 매체는 명령어를 저장한다. 명령어는, 적어도 4개의 데이터 엘리먼트들의 제1 세트를 포함하는 제1 소스 패킹된 데이터를 나타내고, 적어도 4개의 데이터 엘리먼트들의 제2 세트를 포함하는 제2 소스 패킹된 데이터를 나타내며, 목적지 스토리지 위치를 나타낸다. 명령어는 머신에 의해 실행되는 경우, 머신으로 하여금 명령어에 응답하여 목적지 스토리지 위치에 결과 패킹된 데이터를 저장하게 한다. 결과 패킹된 데이터는 적어도 4개의 인덱스들을 포함한다. 인덱스들은 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서의 대응 데이터 엘리먼트 위치들을 식별한다. 인덱스들은 제1 소스 패킹된 데이터 및 제2 소스 패킹된 데이터에서의 대응 데이터 엘리먼트들의 정렬된 순서를 나타내는 결과 패킹된 데이터에서의 위치들에 저장된다.

[0265] 예 24는 청구항 23의 제조 물품을 포함하고, 여기서 명령어는 정렬된 순서로 적어도 4개의 데이터 엘리먼트들을

갖는 제1 소스 패킹된 데이터를 나타낸다.

[0266] 예 25는 청구항 11 내지 13 중 어느 하나의 방법을 수행하는 수단을 포함하는 프로세서 또는 기타 장치이다.

[0267] 예 26은, 청구항 11 내지 13 중 어느 하나의 방법을 수행하는, 모듈들, 유닛들, 로직, 회로, 수단 또는 이들의 임의의 조합을 포함하는 프로세서 또는 기타 장치이다.

[0268] 예 27은 선택적으로 비-일시적 머신-판독가능 스토리지 매체인 머신-판독가능 매체를 포함하는 제조 물품으로서, 프로세서, 컴퓨터 시스템 또는 기타 머신에 의해 실행되는 경우 및/또는 실행될 때, 머신으로 하여금 청구항 11 내지 13 중 어느 하나의 방법을 수행하게 하도록 동작될 수 있는 명령어를 제공한다.

[0269] 예 28은, 상호접속; 상호접속과 연결되는 프로세서; 및 상호접속과 연결되고, DRAM(Dynamic Random Access Memory), 그래픽 칩, 무선 통신 칩, 상 변화 메모리 및 비디오 카메라로부터 선택되는 적어도 하나의 컴포넌트를 포함하는 컴퓨터 시스템 또는 기타 전자 디바이스로서, 이러한 컴퓨터 시스템 또는 기타 전자 디바이스는 청구항 11 내지 13 중 어느 하나의 방법을 수행하도록 동작될 수 있다.

[0270] 예 29는 청구항 18 내지 20 중 어느 하나의 방법을 수행하는 수단을 포함하는 프로세서 또는 기타 장치이다.

[0271] 예 30은, 청구항 18 내지 20 중 어느 하나의 방법을 수행하는, 모듈들, 유닛들, 로직, 회로, 수단 또는 이들의 임의의 조합을 포함하는 프로세서 또는 기타 장치이다.

[0272] 예 31은 선택적으로 비-일시적 머신-판독가능 스토리지 매체인 머신-판독가능 매체를 포함하는 제조 물품으로서, 프로세서, 컴퓨터 시스템 또는 기타 머신에 의해 실행되는 경우 및/또는 실행될 때, 머신으로 하여금 청구항 18 내지 20 중 어느 하나의 방법을 수행하게 하도록 동작될 수 있는 명령어를 제공한다.

[0273] 예 32는, 상호접속; 상호접속과 연결되는 프로세서; 및 상호접속과 연결되고, DRAM(Dynamic Random Access Memory), 그래픽 칩, 무선 통신 칩, 상 변화 메모리 및 비디오 카메라로부터 선택되는 적어도 하나의 컴포넌트를 포함하는 컴퓨터 시스템 또는 기타 전자 디바이스로서, 이러한 컴퓨터 시스템 또는 기타 전자 디바이스는 청구항 18 내지 20 중 어느 하나의 방법을 수행하도록 동작될 수 있다.

[0274] 예 33은 본 명세서에 개시되는 바와 같은 하나 이상의 연산들 또는 임의의 방법을 실질적으로 수행하도록 동작될 수 있는 프로세서 또는 기타 장치이다.

[0275] 예 34는 본 명세서에 개시되는 바와 같은 하나 이상의 연산들 또는 임의의 방법을 실질적으로 수행하는 수단을 포함하는 프로세서 또는 기타 장치이다.

[0276] 예 35는 본 명세서에 개시되는 바와 같은 임의의 명령어들을 실질적으로 수행하는 프로세서 또는 기타 장치이다.

[0277] 예 36은 본 명세서에 개시되는 바와 같은 임의의 명령어들을 실질적으로 수행하는 수단을 포함하는 프로세서 또는 기타 장치이다.

[0278] 예 37은, 실질적으로 본 명세서에 개시되는 바와 같은 임의의 명령어들일 수 있고, 제1 명령어 세트의 것인 제1 명령어를 제2 명령어 세트의 하나 이상의 명령어들로 변환하는 단계를 포함하는 방법을 포함한다. 본 방법은 또한 프로세서 상에서 제2 명령어 세트의 하나 이상의 명령어들을 디코드하고 실행하는 단계를 포함한다. 실행하는 단계는 목적지에 결과를 저장하는 단계를 포함한다. 결과는 제1 명령어에 대해 본 명세서에 개시되는 바와 같은 임의의 결과들을 실질적으로 포함할 수 있다.

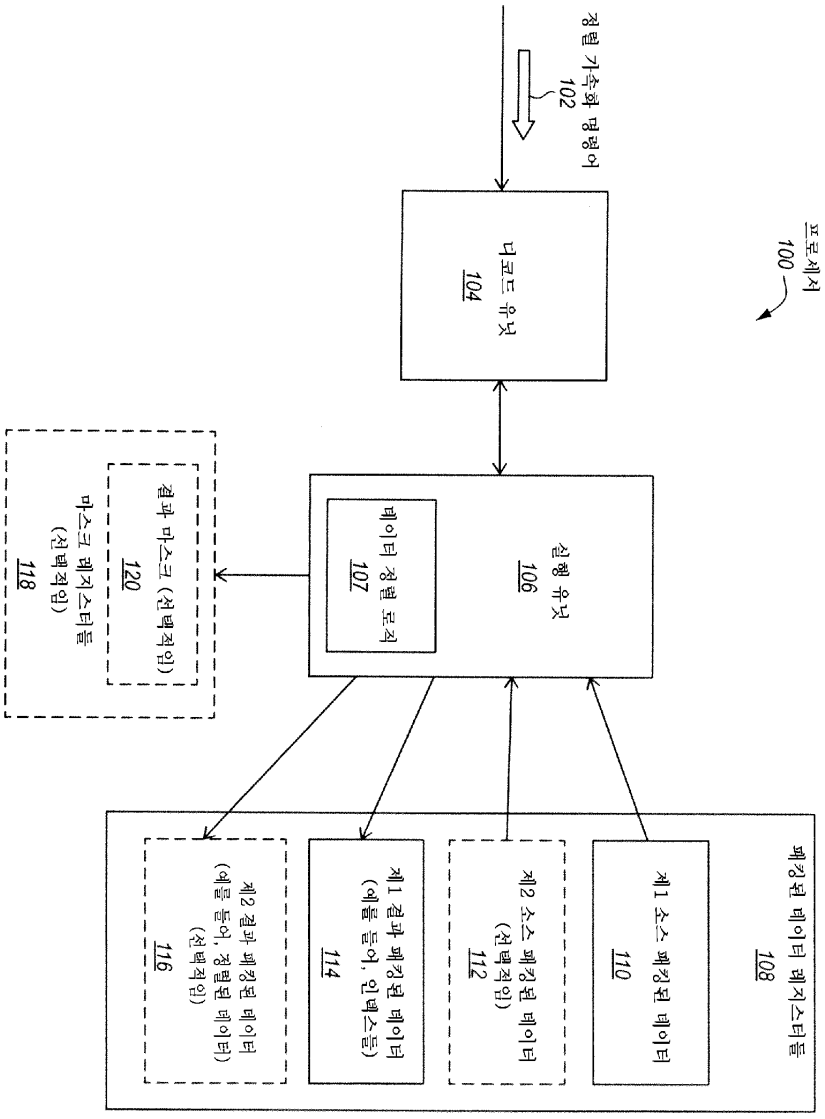
[0279] 예 38은 제1 명령어 세트의 명령어들을 디코드하도록 동작될 수 있는 디코드 유닛을 포함하는 프로세서 또는 기타 장치를 포함한다. 디코드 유닛은 제1 명령어를 에뮬레이트하는 하나 이상의 명령어들을 수신하는데, 이는 실질적으로 본 명세서에 개시된 바와 같은 임의의 명령어일 수 있고, 제2 명령어 세트의 것이다. 프로세서 또는 기타 장치는 또한 디코드 유닛과 연결되고 제1 명령어 세트의 하나 이상의 명령어들을 실행하는 하나 이상의 실행 유닛들을 포함한다. 하나 이상의 실행 유닛들은, 제1 명령어 세트의 하나 이상의 명령어들에 응답하여, 결과를 목적지에 저장하도록 동작될 수 있다. 결과는 제1 명령어에 대해 본 명세서에 개시되는 바와 같은 임의의 결과들을 실질적으로 포함할 수 있다.

[0280] 예 39는, 제1 명령어 세트의 명령어들을 디코드하도록 동작될 수 있는 디코드 유닛을 갖고 하나 이상의 실행 유닛들을 갖는 프로세서를 포함하는 컴퓨터 시스템 또는 기타 전자 디바이스를 포함한다. 컴퓨터 시스템은 또한 프로세서에 연결되는 스토리지 디바이스를 포함한다. 스토리지 디바이스는, 실질적으로 본 명세서에 개시되는

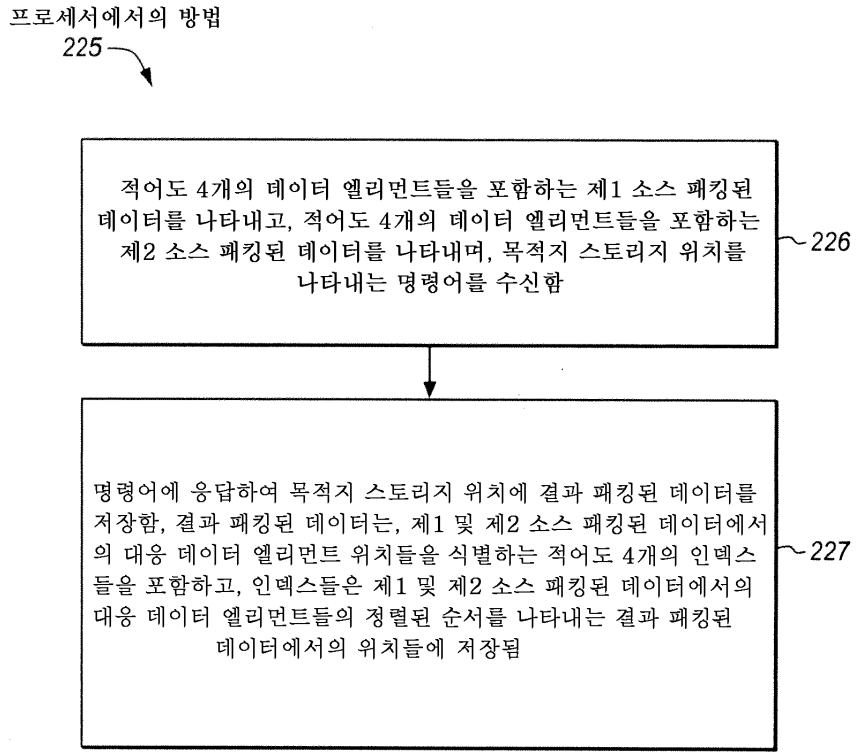
바와 같은 임의의 명령어들일 수 있고, 제2 명령어 세트의 것인 제1 명령어를 저장한다. 스토리지 디바이스는 또한 제1 명령어를 제1 명령어 세트의 하나 이상의 명령어들로 변환하는 명령어들을 저장한다. 제1 명령어 세트의 하나 이상의 명령어들은, 프로세서에 의해 실행될 때, 프로세서로 하여금 목적지에 결과를 저장하게 하도록 동작할 수 있다. 결과는 실질적으로 제1 명령어에 대해 본 명세서에 개시되는 바와 같은 임의의 결과들을 포함할 수 있다.

도면

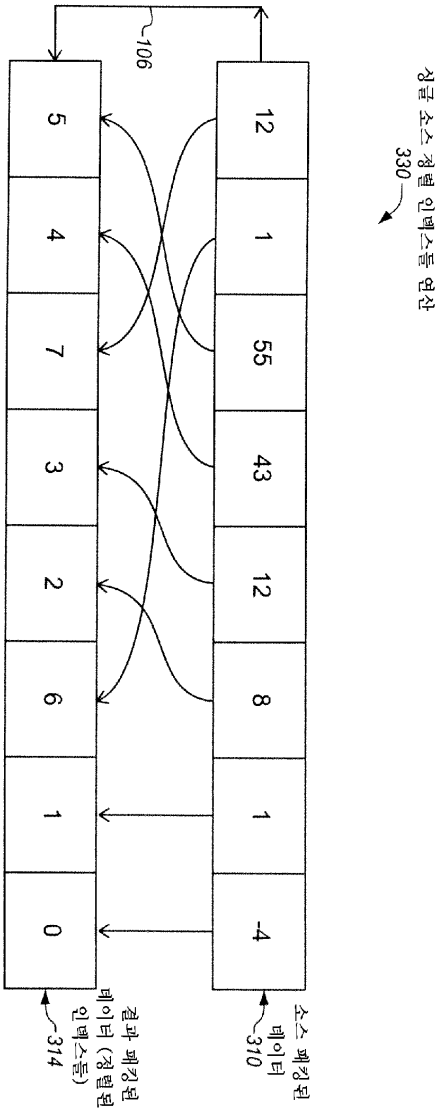
도면1



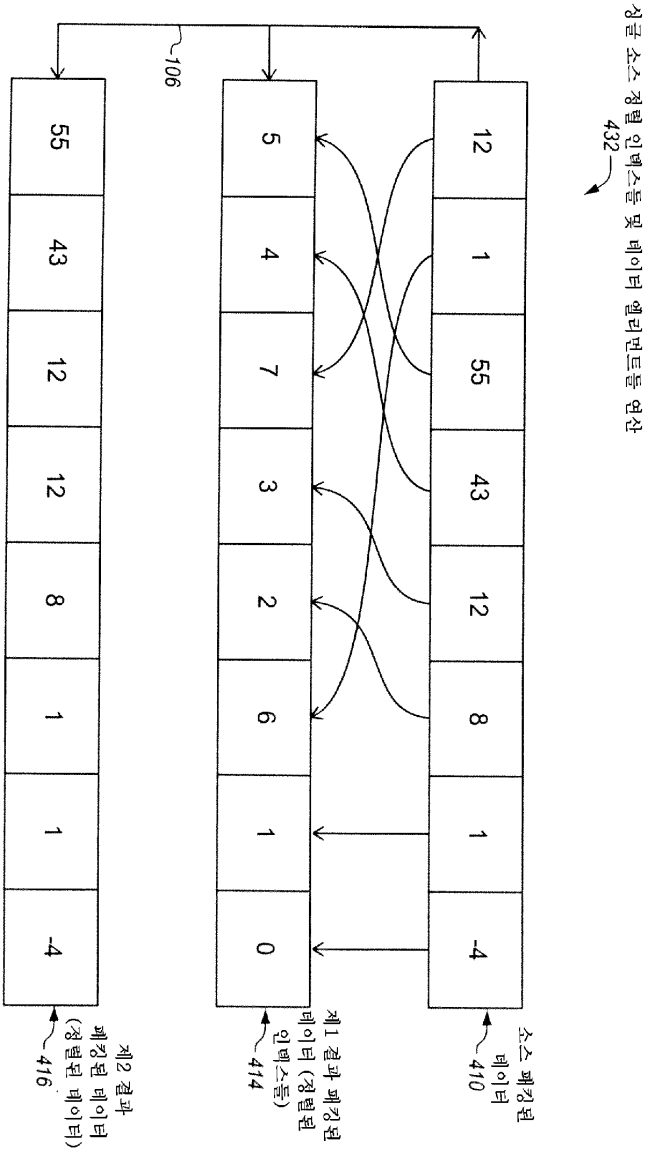
도면2


프로세서에서의 방법
225
적어도 4개의 데이터 엘리먼트들을 포함하는 제1 소스 패킹된 데이터를 나타내고, 적어도 4개의 데이터 엘리먼트들을 포함하는 제2 소스 패킹된 데이터를 나타내며, 목적지 스토리지 위치를 나타내는 명령어를 수신함
226
명령어에 응답하여 목적지 스토리지 위치에 결과 패킹된 데이터를 저장함, 결과 패킹된 데이터는, 제1 및 제2 소스 패킹된 데이터에서의 대응 데이터 엘리먼트 위치들을 식별하는 적어도 4개의 인덱스들을 포함하고, 인덱스들은 제1 및 제2 소스 패킹된 데이터에서의 대응 데이터 엘리먼트들의 정렬된 순서를 나타내는 결과 패킹된 데이터에서의 위치들에 저장됨
227

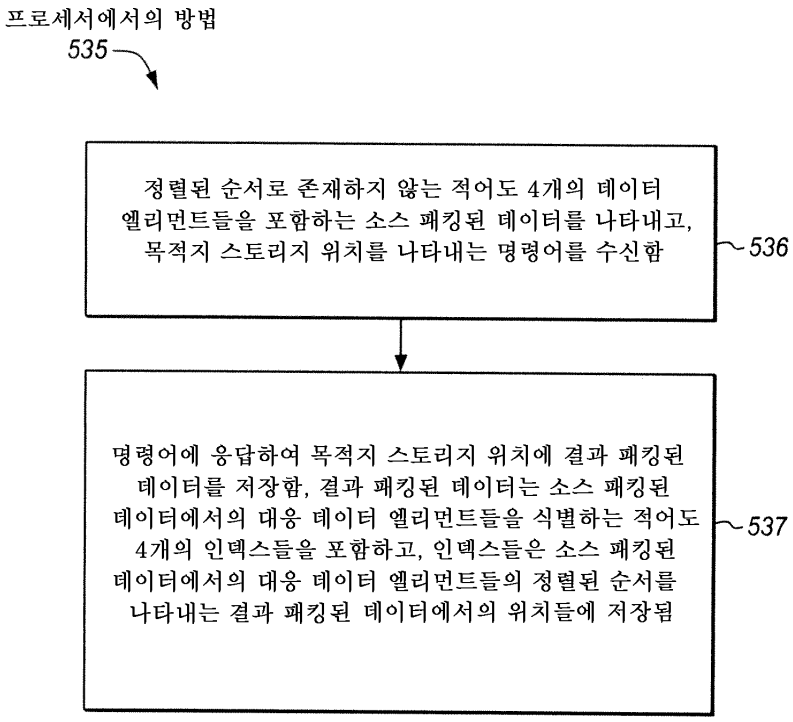
도면3


106
12
1
55
43
12
8
1
-4
5
4
7
3
2
6
1
0
싱글 소스 정렬 인덱스들 연산
330
소스 패킹된 데이터
310
결과 패킹된 데이터 (정렬된 인덱스들)
314

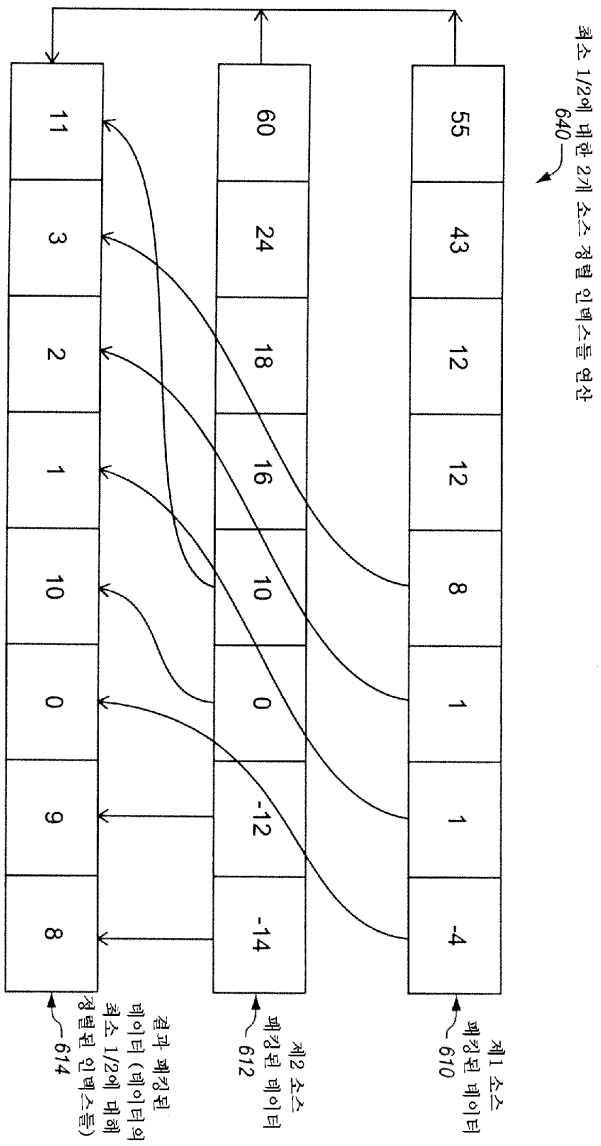
도면4


싱글 소스 정렬 인덱스들 및 데이터 엘리먼트들 연산
432
106
12
1
55
43
12
8
1
-4
소스 패킹된 데이터
410
5
4
7
3
2
6
1
0
제1 결과 패킹된 데이터 (정렬된 인덱스들)
414
55
43
12
12
8
1
1
-4
제2 결과 패킹된 데이터 (정렬된 데이터)
416

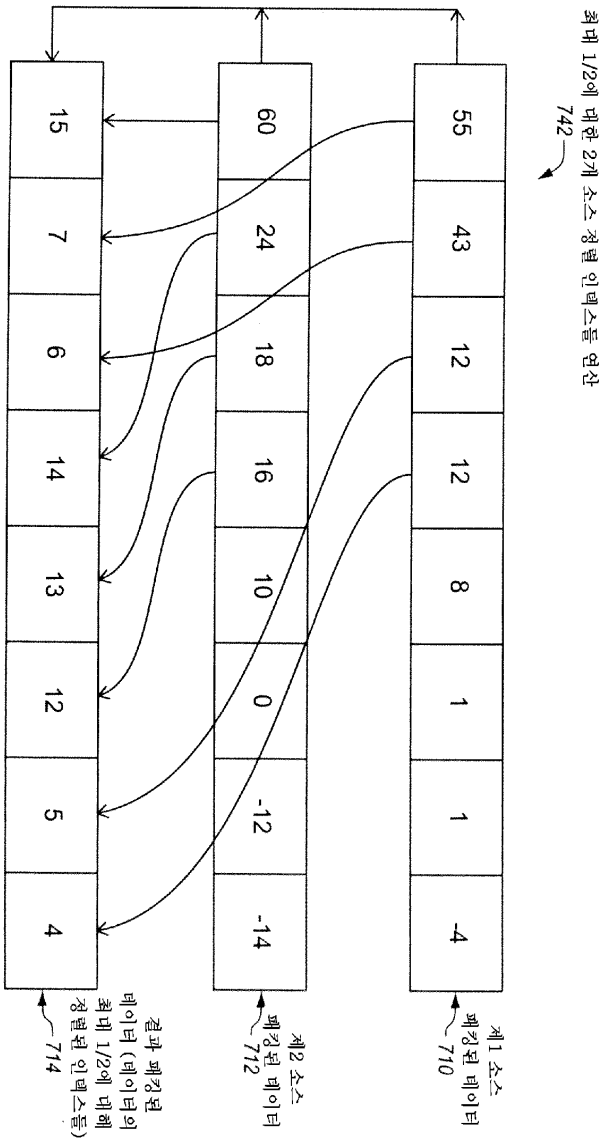
도면5


프로세서에서의 방법
535
정렬된 순서로 존재하지 않는 적어도 4개의 데이터 엘리먼트들을 포함하는 소스 패킹된 데이터를 나타내고, 목적지 스토리지 위치를 나타내는 명령어를 수신함
536
명령어에 응답하여 목적지 스토리지 위치에 결과 패킹된 데이터를 저장함, 결과 패킹된 데이터는 소스 패킹된 데이터에서의 대응 데이터 엘리먼트들을 식별하는 적어도 4개의 인덱스들을 포함하고, 인덱스들은 소스 패킹된 데이터에서의 대응 데이터 엘리먼트들의 정렬된 순서를 나타내는 결과 패킹된 데이터에서의 위치들에 저장됨
537

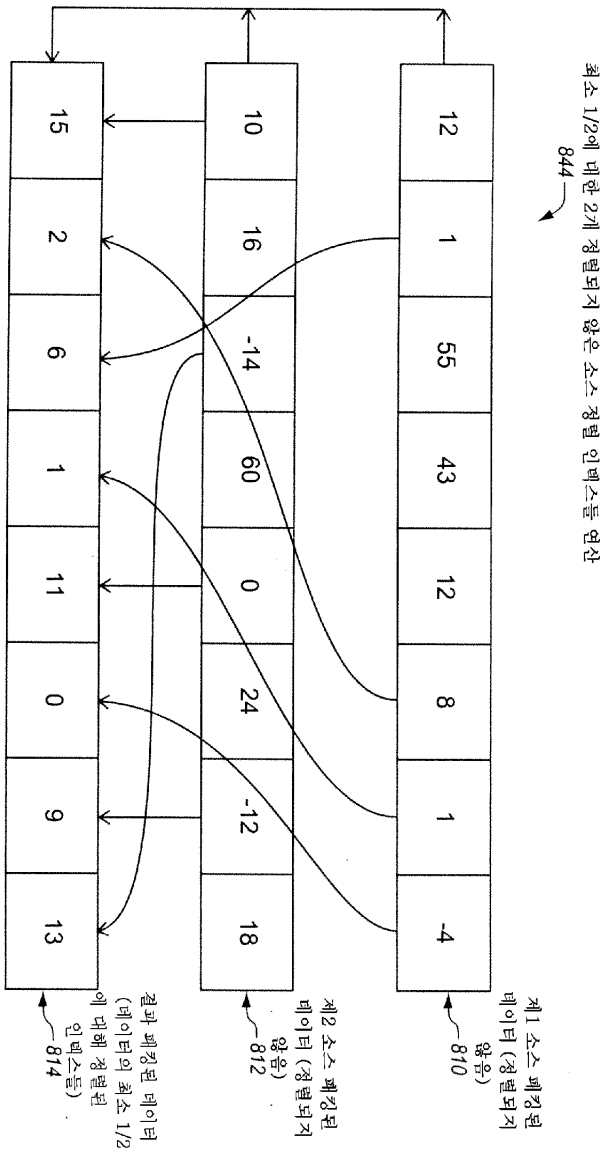
도면6


최소 1/2에 대한 2개 소스 정렬 인덱스들 연산
640
55
43
12
12
8
1
1
-4
제1 소스 패킹된 데이터
610
60
24
18
16
10
0
-12
-14
제2 소스 패킹된 데이터
612
11
3
2
1
10
0
9
8
결과 패킹된 데이터 (데이터의 최소 1/2에 대해 정렬된 인덱스들)
614

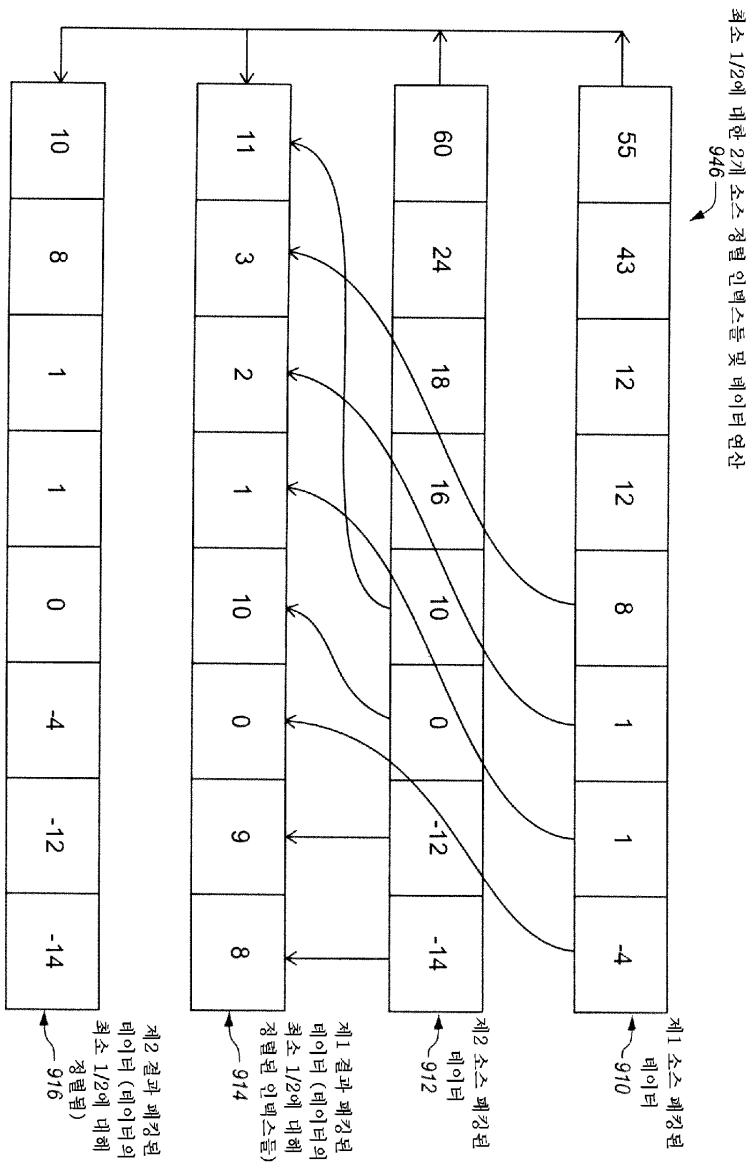
도면7


최대 1/2에 대한 2개 소스 정렬 인덱스들 연산
742
55
43
12
12
8
1
1
-4
제1 소스 패킹된 데이터
710
60
24
18
16
10
0
-12
-14
제2 소스 패킹된 데이터
712
15
7
6
14
13
12
5
4
결과 패킹된 데이터 (데이터의 최대 1/2에 대해 정렬된 인덱스들)
714

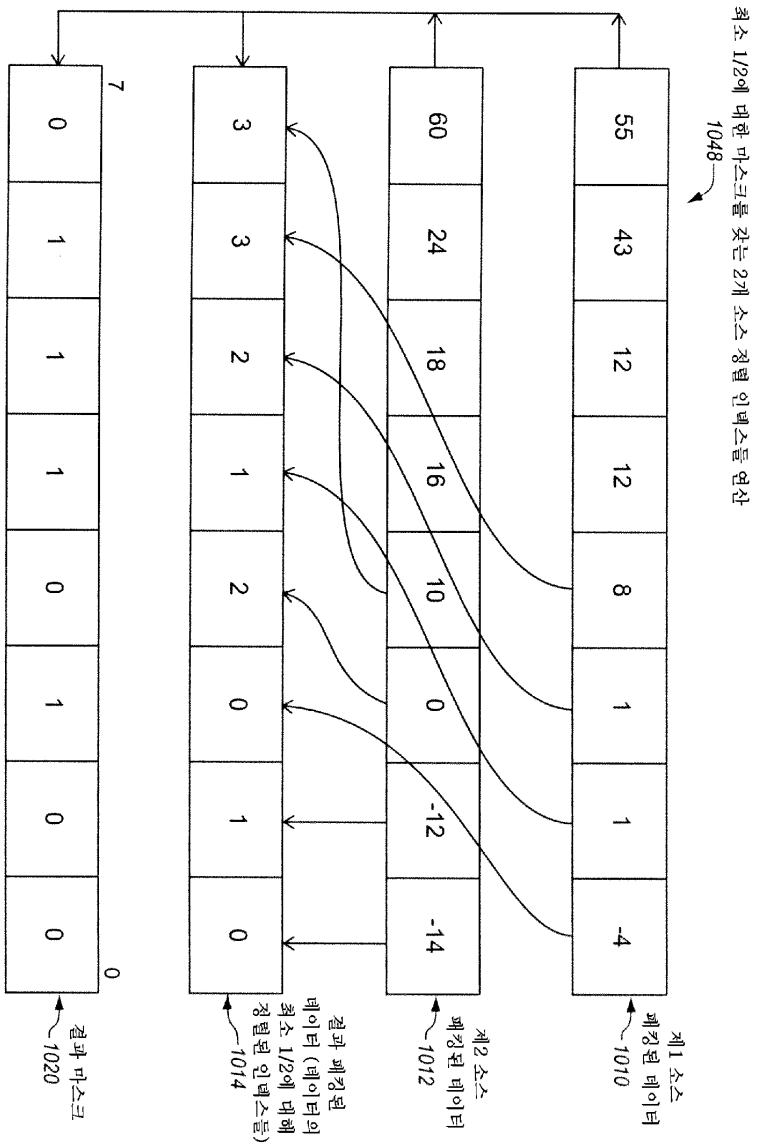
도면8


최소 1/2에 대한 2개 정렬되지 않은 소스 정렬 인덱스들 연산
844
12
1
55
43
12
8
1
-4
제1 소스 패킹된 데이터 (정렬되지 않음)
810
10
16
-14
60
0
24
-12
18
제2 소스 패킹된 데이터 (정렬되지 않음)
812
15
2
6
1
11
0
9
13
결과 패킹된 데이터 (데이터의 최소 1/2에 대해 정렬된 인덱스들)
814

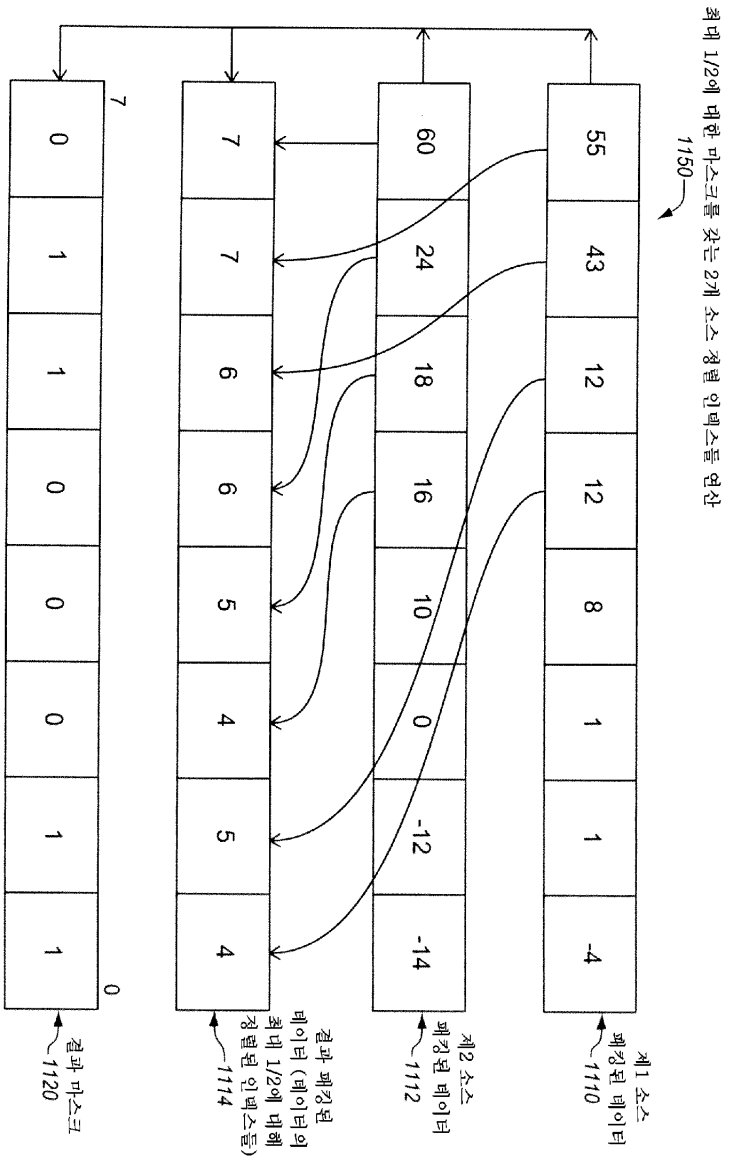
도면9


최소 1/2에 대한 2개 소스 정렬 인덱스들 및 데이터 연산
946
55
43
12
12
8
1
1
-4
60
24
18
16
10
0
-12
-14
11
3
2
1
10
0
9
8
10
8
1
1
0
-4
-12
-14
제1 소스 패킹된 데이터
910
제2 소스 패킹된 데이터
912
제1 결과 패킹된 데이터 (데이터의 최소 1/2에 대해 정렬된 인덱스들)
914
제2 결과 패킹된 데이터 (데이터의 최소 1/2에 대해 정렬됨)
916

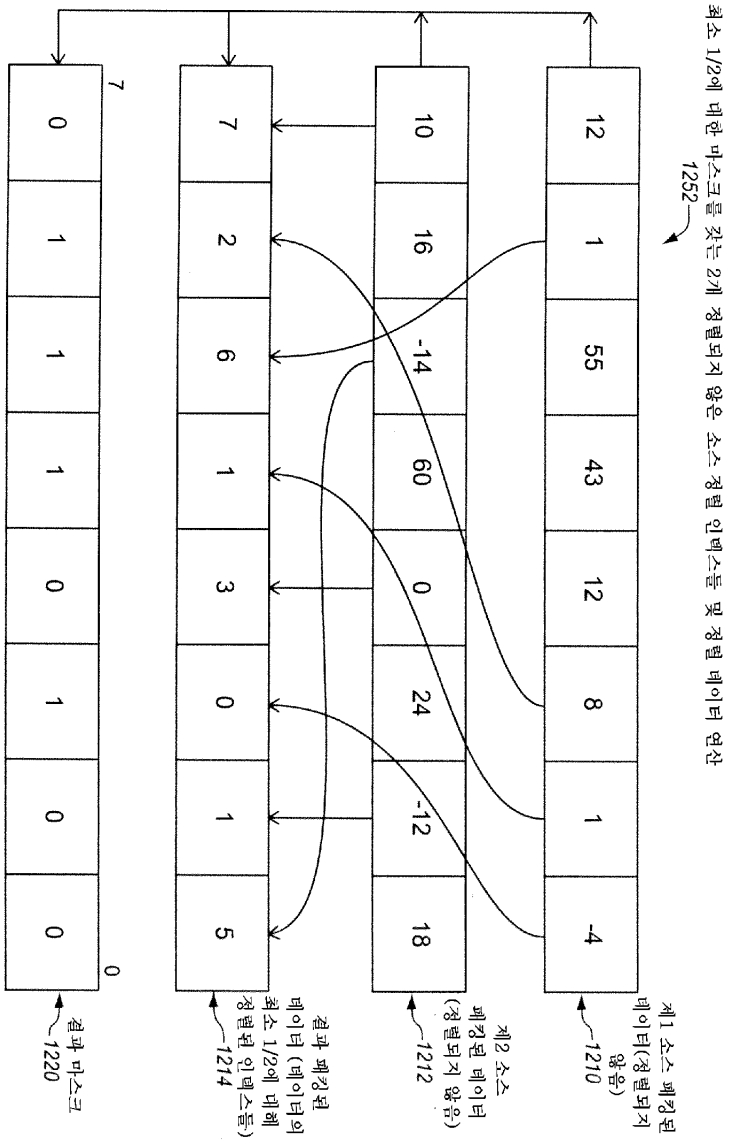
도면10


최소 1/2에 대한 마스크를 갖는 2개 소스 정렬 인덱스들 연산
1048
55
43
12
12
8
1
1
-4
제1 소스 패킹된 데이터
1010
60
24
18
16
10
0
-12
-14
제2 소스 패킹된 데이터
1012
3
3
2
1
2
0
1
0
결과 패킹된 데이터 (데이터의 최소 1/2에 대해 정렬된 인덱스들)
1014
0
1
1
1
0
1
0
0
7
0
결과 마스크
1020

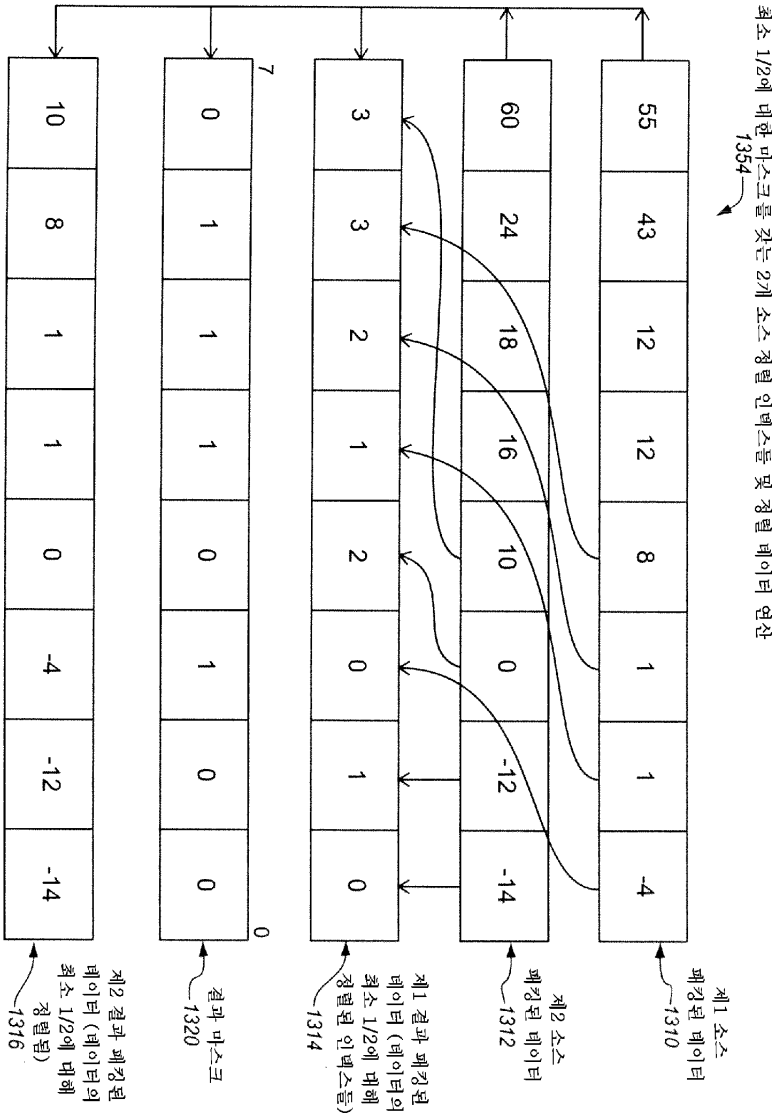
도면11


최대 1/2에 대한 마스크를 갖는 2개 소스 정렬 인덱스들 연산
1150
55
43
12
12
8
1
1
-4
제1 소스 패킹된 데이터
1110
60
24
18
16
10
0
-12
-14
제2 소스 패킹된 데이터
1112
7
7
6
6
5
4
5
4
결과 패킹된 데이터 (데이터의 최대 1/2에 대해 정렬된 인덱스들)
1114
7
0
0
1
1
0
1
1
1
0
결과 마스크
1120

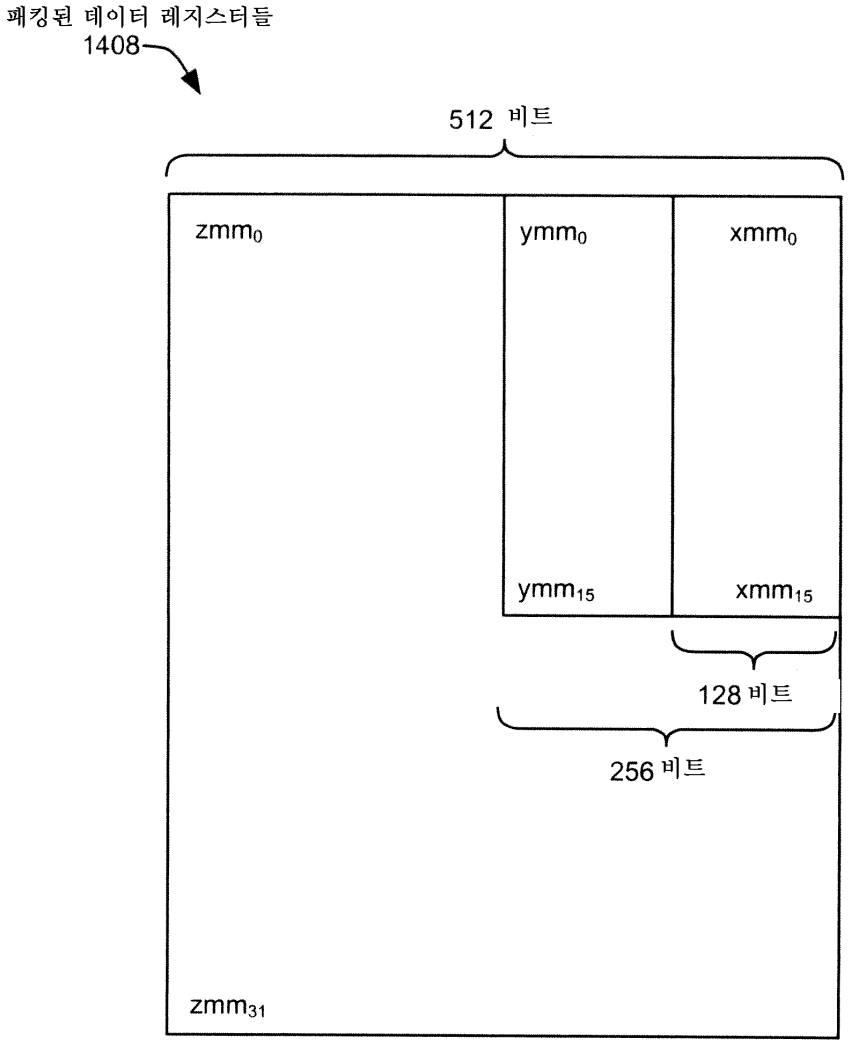
도면12


최소 1/2에 대한 마스크를 갖는 2개 정렬되지 않은 소스 정렬 인덱스들 및 정렬 데이터 연산
1252
12
1
55
43
12
8
1
-4
제1 소스 패킹된 데이터(정렬되지 않음)
1210
10
16
-14
60
0
24
-12
18
제2 소스 패킹된 데이터 (정렬되지 않음)
1212
7
2
6
1
3
0
1
5
결과 패킹된 데이터 (데이터의 최소 1/2에 대해 정렬된 인덱스들)
1214
0
1
1
1
0
1
0
0
결과 마스크
1220
7
0

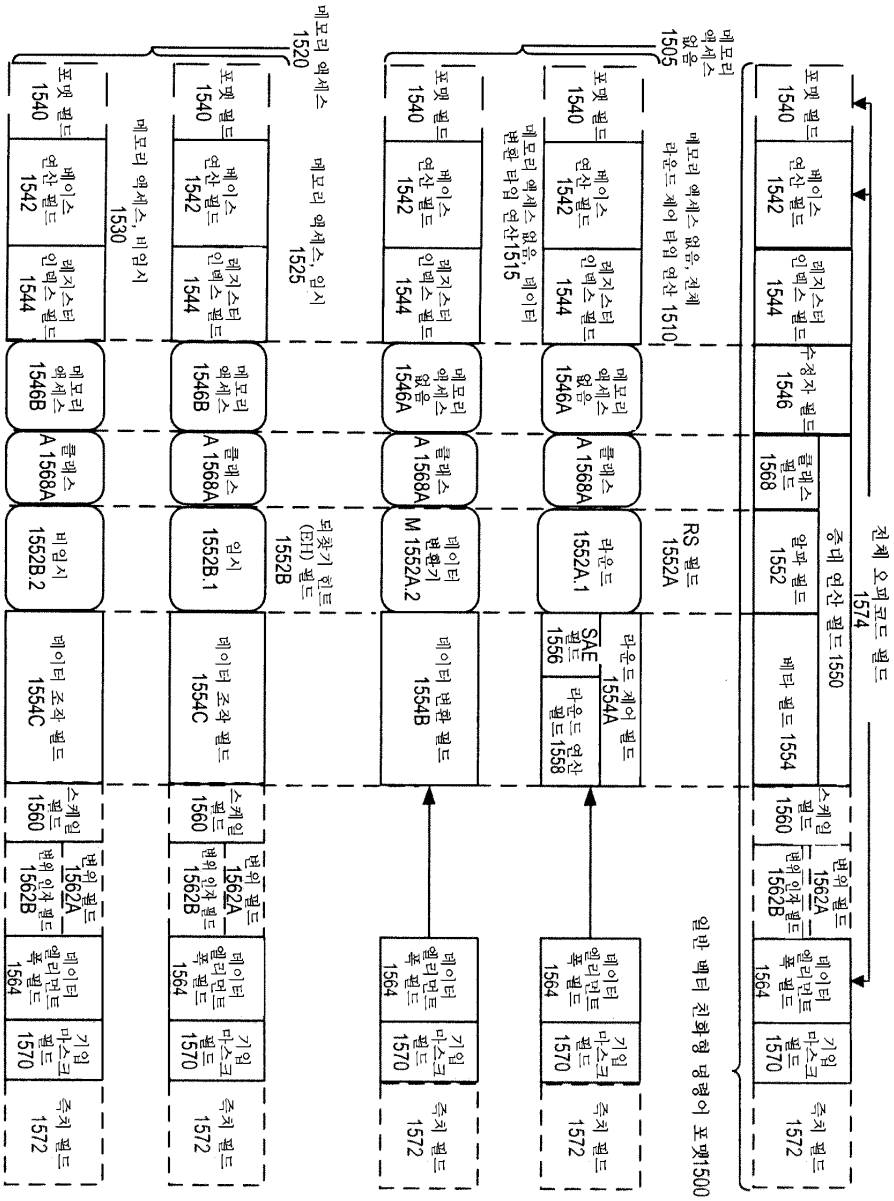
도면13


최소 1/2에 대한 마스크를 갖는 2개 소스 정렬 인덱스들 및 정렬 데이터 연산
1354
55
43
12
12
8
1
1
-4
제1 소스 패킹된 데이터
1310
60
24
18
16
10
0
-12
-14
제2 소스 패킹된 데이터
1312
3
3
2
1
2
0
1
0
제1 결과 패킹된 데이터 (데이터의 최소 1/2에 대해 정렬된 인덱스들)
1314
7
0
1
1
1
0
1
0
0
결과 마스크
1320
0
10
8
1
1
0
-4
-12
-14
제2 결과 패킹된 데이터 (데이터의 최소 1/2에 대해 정렬됨)
1316

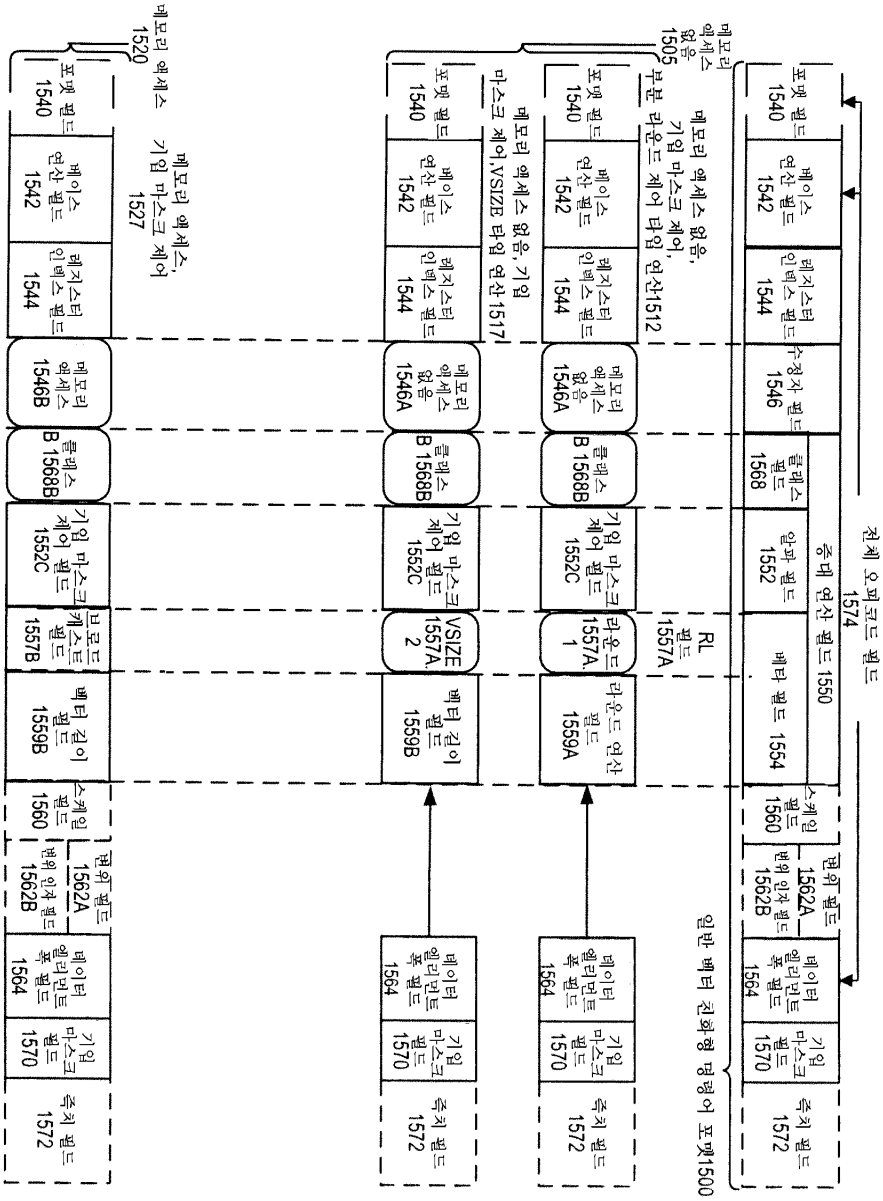
도면14


패킹된 데이터 레지스터들
1408
512 비트
zmm0
ymm0
xmm0
ymm15
xmm15
128 비트
256 비트
zmm31

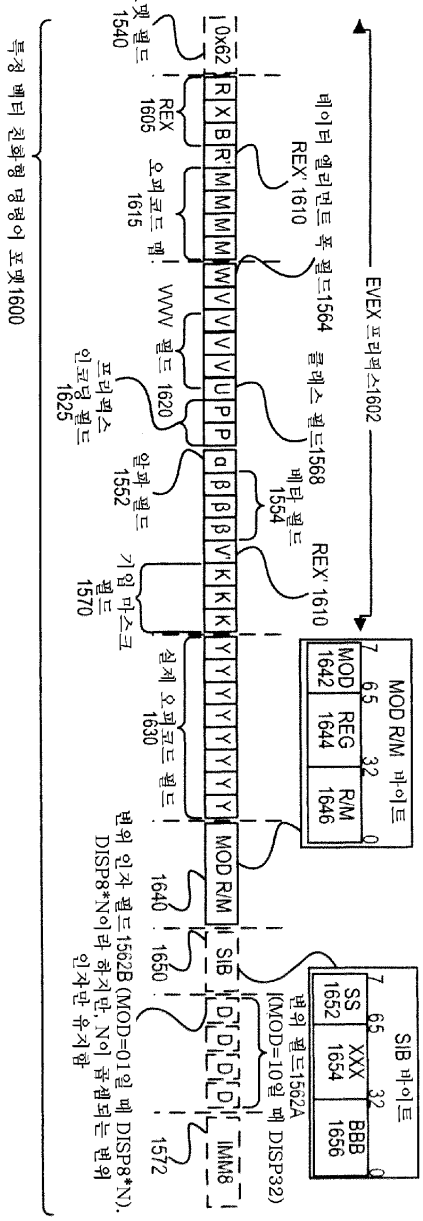
도면15a


전체 오피코드 필드
1574
포맷 필드
1540
베이스 연산 필드
1542
레지스터 인덱스 필드
1544
주정자 필드
1546
클래스 필드
1568
중미 연산 필드 1550
알파 필드
1552
베타 필드 1554
스케일 필드
1560
변위 필드
1562A
변위 인자 필드
1562B
데이터 엘리먼트 풀 필드
1564
기입 마스크 필드
1570
즉치 필드
1572
일반 백터 친화형 명령어 포맷1500
메모리 액세스 없음
1505
메모리 액세스 없음, 전체 라운드 제어 타입 연산 1510
메모리 액세스 없음, 데이터 변환 타입 연산1515
메모리 액세스
1520
메모리 액세스, 일시
1525
메모리 액세스, 비일시
1530
메모리 액세스 없음
1546A
클래스 A 1568A
RS 필드
1552A
라운드 제어 타입 연산
1554A
SAE 필드
1556
라운드 연산 필드1558
데이터 변환 필드
1554B
메모리 액세스
1546B
클래스 A 1568A
일시
1552B.1
데이터 조작 필드
1554C
비일시
1552B.2
퇴출기 힌트 (EH) 필드
1552B
M 1552A.2

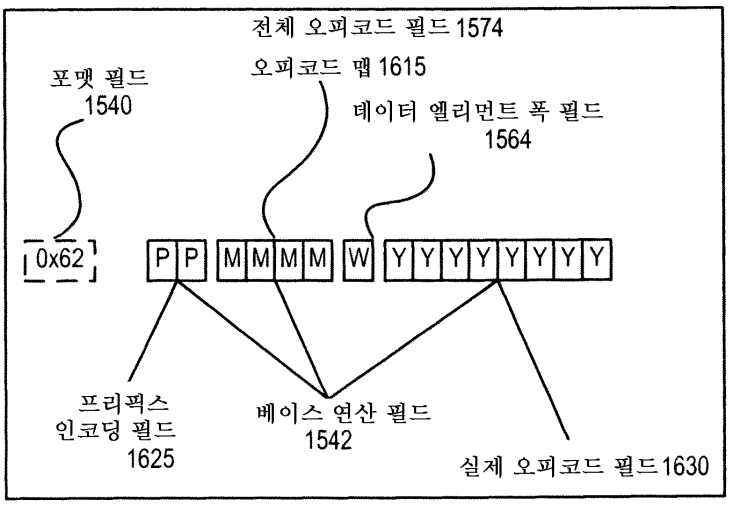
도면15b


전체 오피코드 필드
1574
일반 벡터 전화형 명령어 포맷1500
포맷 필드
1540
베이스 연산 필드
1542
데이터 엘리먼트 폭 필드
1564
기입 마스크 필드
1570
즉치 필드
1572
스케일 필드
1560
변위 인자 필드
1562B
변위 필드
1562A
레지스터 인덱스 필드
1544
수정자 필드
1546
클래스 필드
1568
알파 필드
1552
베타 필드 1554
중대 연산 필드 1550
메모리 액세스 없음
1505
부분 라운드 제어 타입 연산1512
메모리 액세스 없음, 기입 마스크 제어, VSIZE 타입 연산1517
메모리 액세스 없음, 기입 마스크 제어, RL 필드 1557A
메모리 액세스
1520
메모리 액세스, 기입 마스크 제어
1527
메모리 액세스 없음
1546A
메모리 액세스
1546B
기입 마스크 제어 필드
1552C
RL 필드 1557A
라운드 연산 필드
1559A
VSIZE 1557A2
라운드 제어 필드 1557A1
벡터 길이 필드
1559B
브로드캐스트 필드
1557B

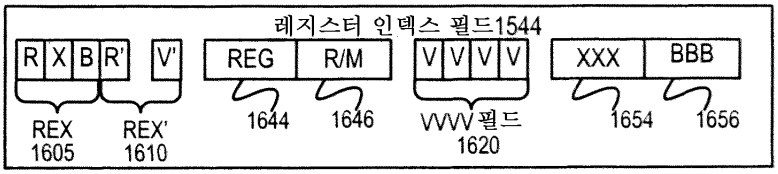
도면16a


특정 벡터 친화형 명령어 포맷1600
EVEX 프리픽스1602
0x62
포맷 필드 1540
REX 1605
오퍼코드 맵 1615
VVVV 필드 1620
프리픽스 인코딩 필드 1625
알파 필드 1552
베타 필드 1554
기입 마스크 필드 1570
실제 오퍼코드 필드 1630
MOD R/M 1640
SIB 1650
변위 필드1562A
변위 인자 필드1562B (MOD=01일 때 DISP8*N).
DISP8*N이라 하지만, N이 곱셈되는 변위 인자만 유지됨
(MOD=10일 때 DISP32)
IMM8 1572
베이터 엘리먼트 폭 필드1564
클래스 필드1568
REX' 1610
REX' 1610
MOD R/M 바이트
MOD 1642
REG 1644
R/M 1646
SIB 바이트
SS 1652
XXX 1654
BBB 1656

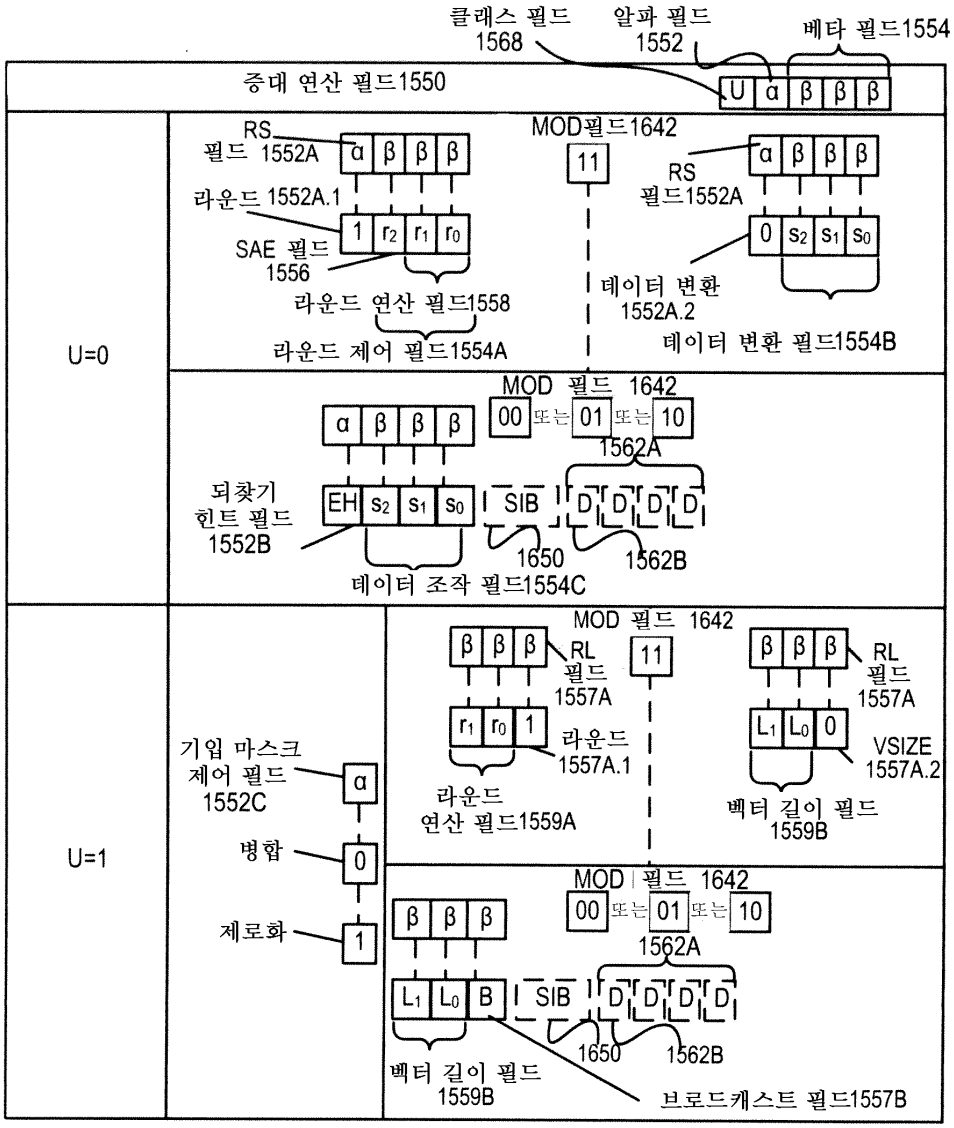
도면16b


전체 오피코드 필드 1574
포맷 필드
1540
오피코드 맵 1615
데이터 엘리먼트 폭 필드
1564
0x62
P P
M M M M
W
Y Y Y Y Y Y Y Y
프리픽스
인코딩 필드
1625
베이스 연산 필드
1542
실제 오피코드 필드 1630

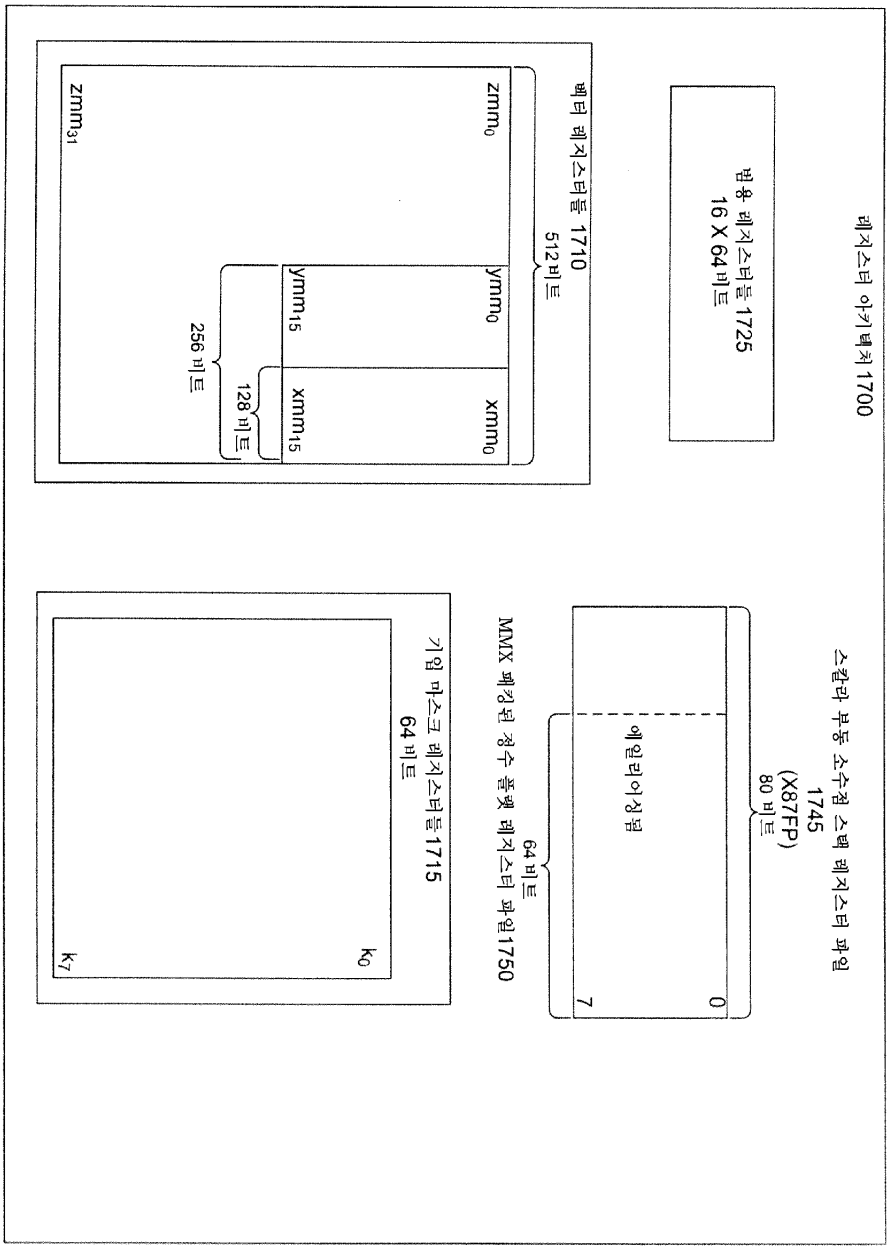
도면16c


레지스터 인덱스 필드1544
R X B R'
V'
REG
R/M
V V V V
XXX
BBB
REX
1605
REX'
1610
1644
1646
VVVV 필드
1620
1654
1656

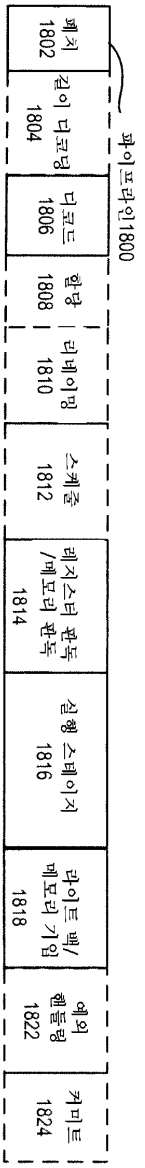
도면16d


클래스 필드 1568
알파 필드 1552
베타 필드1554
증대 연산 필드1550
U α β β β
U=0
RS 필드 1552A
α β β β
라운드 1552A.1
1 r2 r1 r0
SAE 필드 1556
라운드 연산 필드1558
라운드 제어 필드1554A
MOD필드1642
11
RS 필드1552A
α β β β
0 s2 s1 s0
데이터 변환 1552A.2
데이터 변환 필드1554B
MOD 필드 1642
00 또는 01 또는 10
1562A
α β β β
되찾기 힌트 필드 1552B
EH s2 s1 s0
SIB
D D D D
1650
1562B
데이터 조작 필드1554C
U=1
기입 마스크 제어 필드 1552C
α
병합
0
제로화
1
MOD 필드 1642
β β β
RL 필드 1557A
11
r1 r0 1
라운드 1557A.1
라운드 연산 필드1559A
β β β
RL 필드 1557A
L1 L0 0
VSIZE 1557A.2
벡터 길이 필드 1559B
MOD 필드 1642
00 또는 01 또는 10
1562A
β β β
L1 L0 B
SIB
D D D D
1650
1562B
벡터 길이 필드 1559B
브로드캐스트 필드1557B

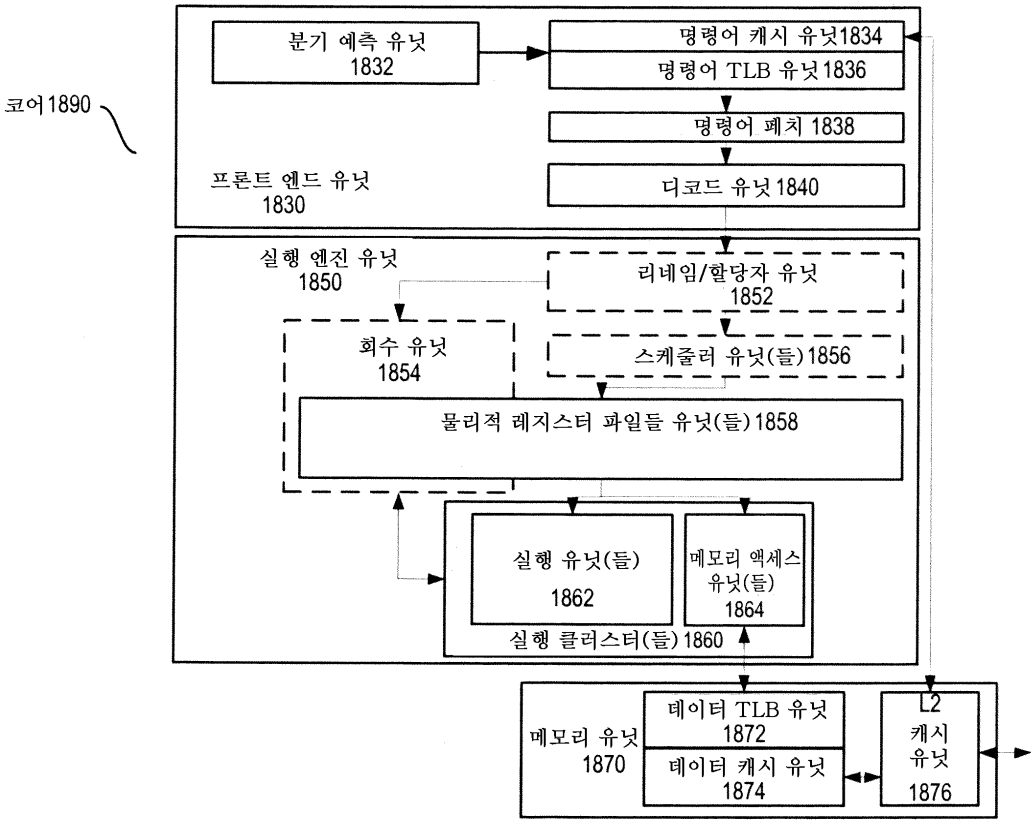
도면17


레지스터 아키텍처 1700
범용 레지스터들 1725
16 X 64 비트
스칼라 부동 소수점 스택 레지스터 파일
1745
(X87FP)
80 비트
0
에일리어싱 됨
7
64 비트
MMX 패킹된 정수 플랫 레지스터 파일1750
벡터 레지스터들 1710
512 비트
zmm0
ymm0
xmm0
ymm15
xmm15
128 비트
256 비트
zmm31
기입 마스크 레지스터들1715
64 비트
k0
k7

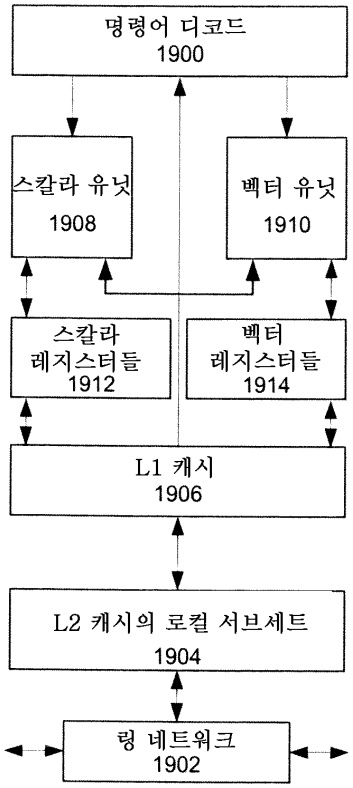
도면18a


파이프라인1800
페치
1802
길이 디코딩
1804
디코드
1806
할당
1808
리네이밍
1810
스케줄
1812
레지스터 판독/메모리 판독
1814
실행 스테이지
1816
라이트 백/메모리 기입
1818
예외 핸들링
1822
커미트
1824

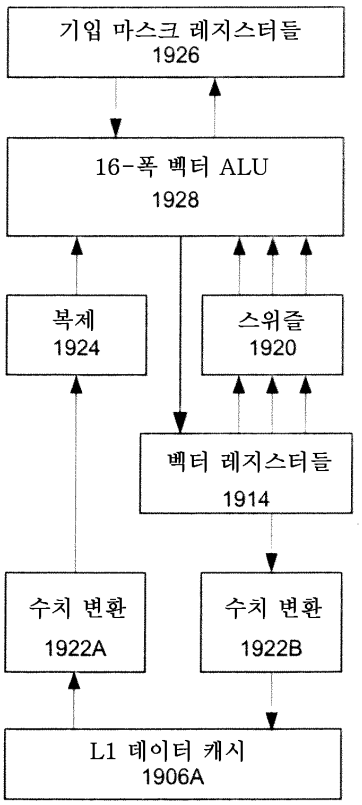
도면18b


코어1890
분기 예측 유닛
1832
명령어 캐시 유닛1834
명령어 TLB 유닛1836
명령어 페치 1838
디코드 유닛1840
프론트 엔드 유닛
1830
실행 엔진 유닛
1850
리네임/할당자 유닛
1852
회수 유닛
1854
스케줄러 유닛(들)1856
물리적 레지스터 파일들 유닛(들)1858
실행 유닛(들)
1862
메모리 액세스
유닛(들)
1864
실행 클러스터(들)1860
메모리 유닛
1870
데이터 TLB 유닛
1872
데이터 캐시 유닛
1874
L2
캐시
유닛
1876

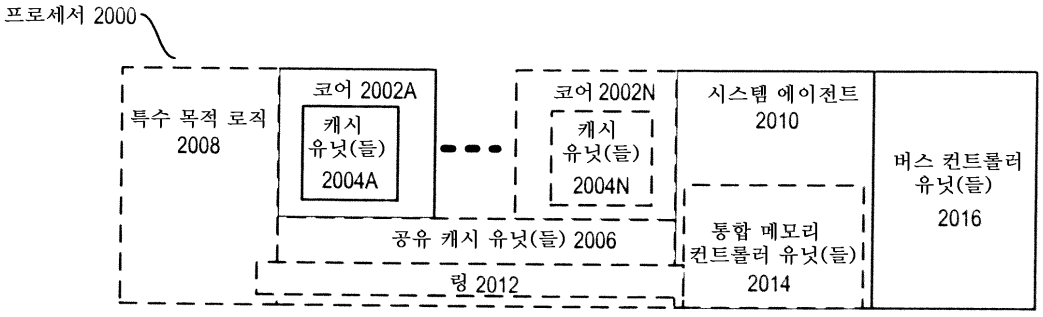
도면19a


명령어 디코드
1900
스칼라 유닛
1908
벡터 유닛
1910
스칼라
레지스터들
1912
벡터
레지스터들
1914
L1 캐시
1906
L2 캐시의 로컬 서브세트
1904
링 네트워크
1902

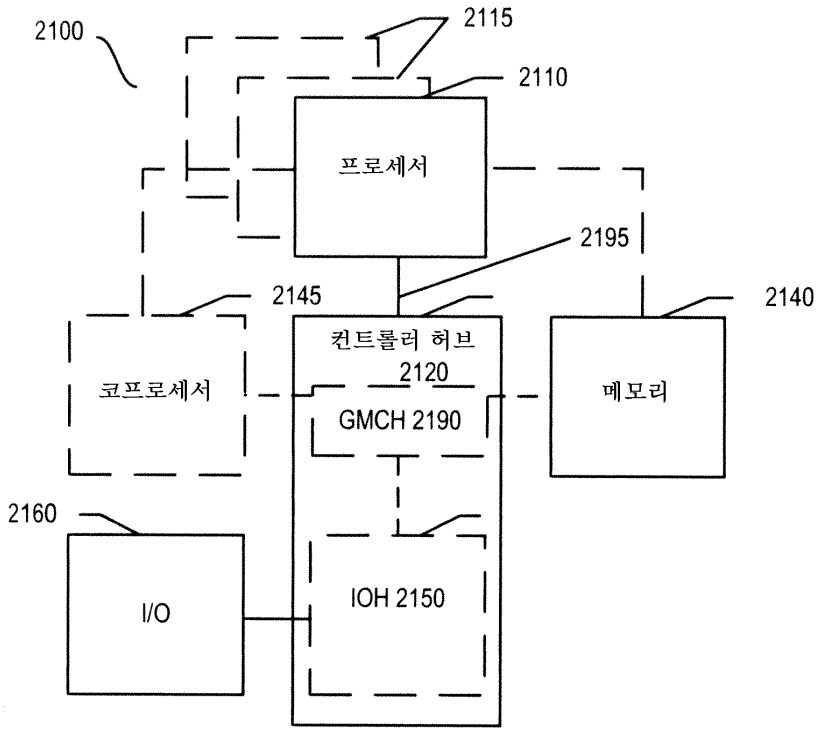
도면19b


기입 마스크 레지스터들
1926
16-폭 벡터 ALU
1928
복제
1924
스위즐
1920
벡터 레지스터들
1914
수치 변환
1922A
수치 변환
1922B
L1 데이터 캐시
1906A

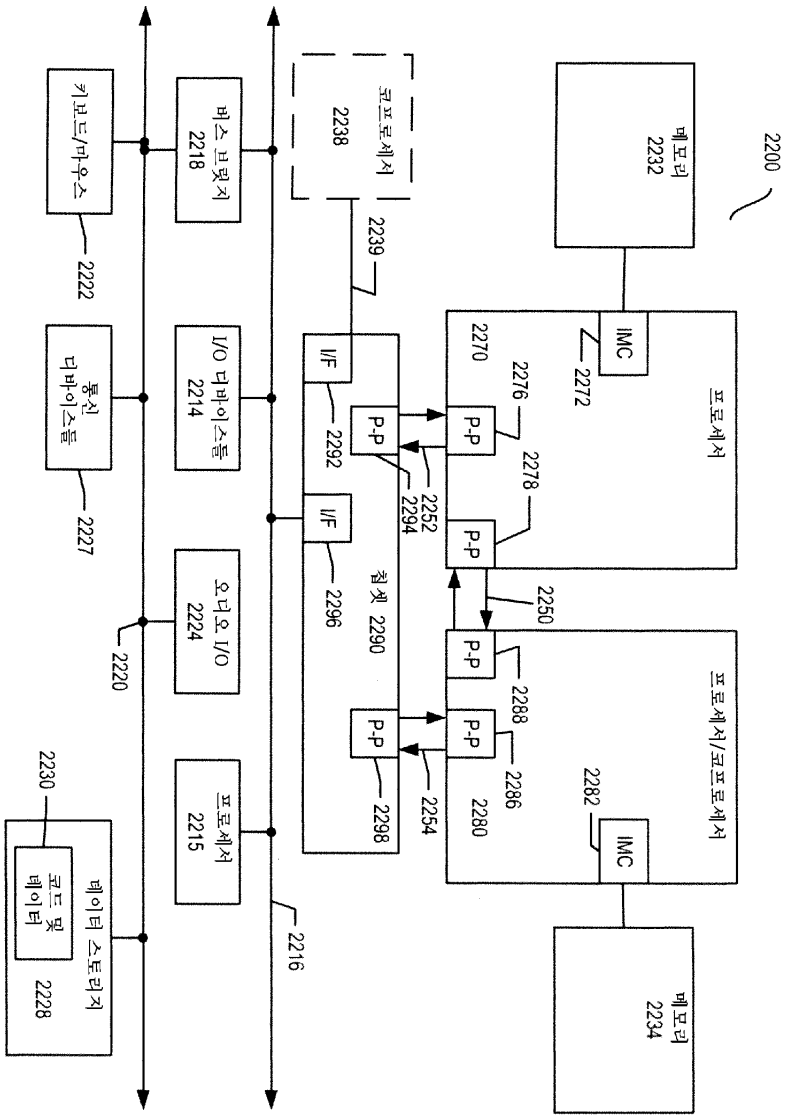
도면20


프로세서 2000
특수 목적 로직
2008
코어 2002A
캐시
유닛(들)
2004A
코어 2002N
캐시
유닛(들)
2004N
시스템 에이전트
2010
버스 컨트롤러
유닛(들)
2016
공유 캐시 유닛(들) 2006
통합 메모리
컨트롤러 유닛(들)
2014
링 2012

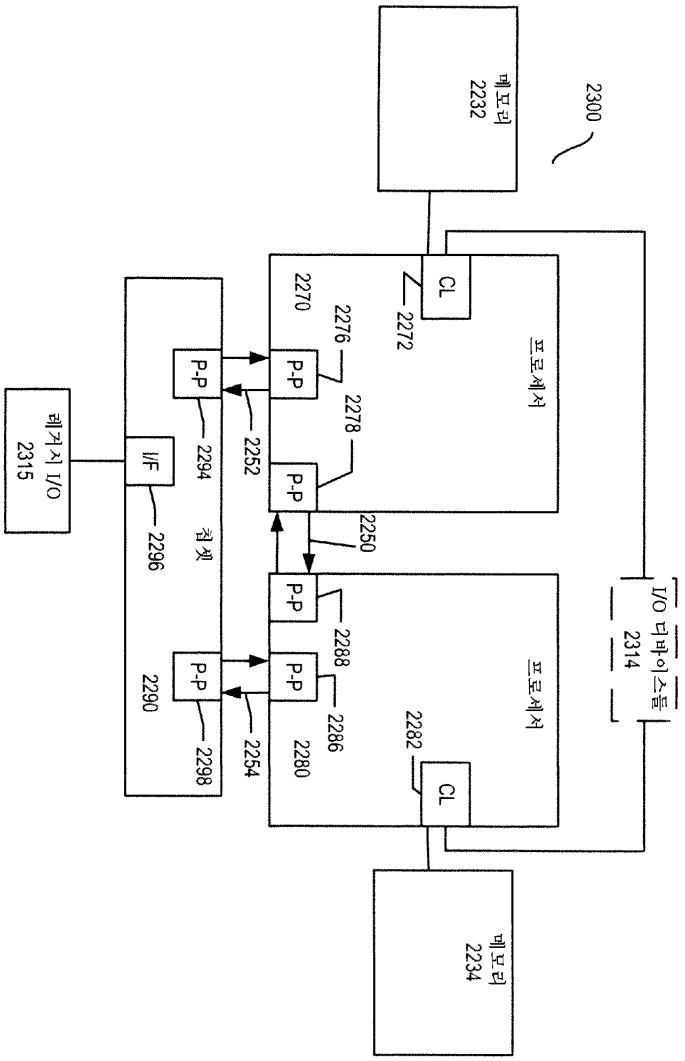
도면21


2100
2115
2110
프로세서
2195
2145
2140
컨트롤러 허브
2120
코프로세서
GMCH 2190
메모리
2160
I/O
IOH 2150

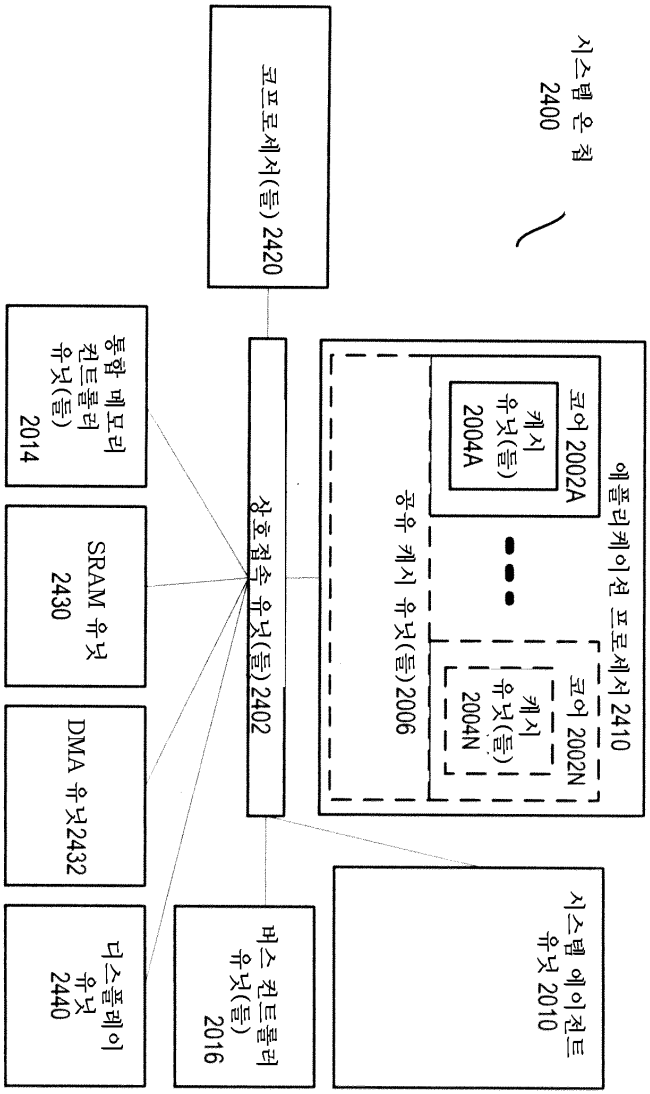
도면22


2200
메모리
2232
IMC
2272
프로세서
2270
P-P
2276
P-P
2278
P-P
2250
P-P
2288
P-P
2286
2280
2282
IMC
프로세서/코프로세서
메모리
2234
코프로세서
2238
2239
I/F
2292
P-P
2252
2294
I/F
2296
칩셋 2290
P-P
2298
2254
2216
키보드/마우스
2222
버스 브릿지
2218
I/O 디바이스들
2214
통신
디바이스들
2227
오디오 I/O
2224
2220
프로세서
2215
데이터 스토리지
2230
코드 및
데이터
2228

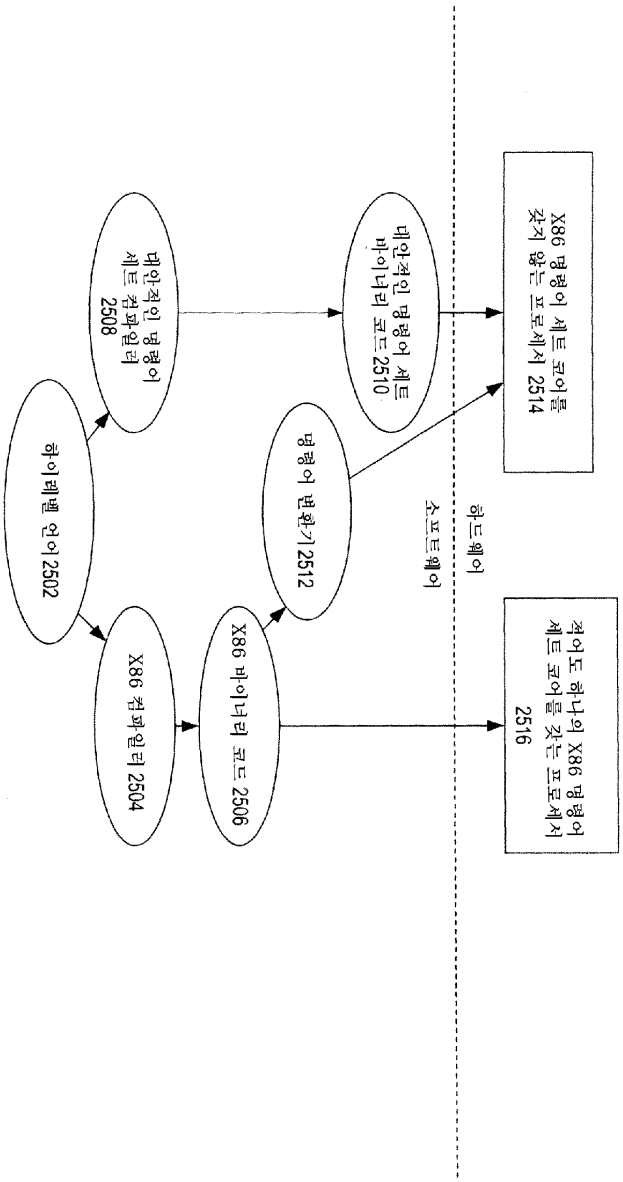
도면23


2300
메모리
2232
CL
2272
2270
2276
2278
P-P
프로세서
2250
P-P
2288
P-P
2286
2280
2282
CL
프로세서
2234
메모리
I/O 디바이스들
2314
P-P
2294
2252
I/F
2296
칩 셋
P-P
2290
2254
P-P
2298
레거시 I/O
2315

도면24


시스템 온 칩
2400
코프로세서(들) 2420
통합 메모리 컨트롤러 유닛(들) 2014
SRAM 유닛 2430
DMA 유닛2432
디스플레이 유닛 2440
상호접속 유닛(들) 2402
에플리케이션 프로세서 2410
코어 2002A
캐시 유닛(들) 2004A
코어 2002N
캐시 유닛(들) 2004N
공유 캐시 유닛(들)2006
버스 컨트롤러 유닛(들) 2016
시스템 에이전트 유닛 2010

도면25


하이레벨 언어 2502
대안적인 명령어 세트 컴파일러 2508
X86 컴파일러 2504
X86 바이너리 코드 2506
명령어 변환기 2512
대안적인 명령어 세트 바이너리 코드 2510
X86 명령어 세트 코어를 갖지 않는 프로세서 2514
적어도 하나의 X86 명령어 세트 코어를 갖는 프로세서 2516
소프트웨어
하드웨어