



(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2017-0051465
(43) 공개일자 2017년05월11일

(51) 국제특허분류(Int. Cl.)
G06F 9/52 (2006.01)
(52) CPC특허분류
G06F 9/526 (2013.01)
(21) 출원번호 10-2017-7008698
(22) 출원일자(국제) 2015년07월28일
심사청구일자 없음
(85) 번역문제출일자 2017년03월30일
(86) 국제출원번호 PCT/GB2015/052177
(87) 국제공개번호 WO 2016/038328
국제공개일자 2016년03월17일
(30) 우선권주장
1415834.9 2014년09월08일 영국(GB)

(71) 출원인
에이알엠 리미티드
영국 캠브리지 씨비1 9엔제이 체리턴 폴번로드 110
(72) 발명자
홀름, 룬
영국 캠브리지 캠브리지셔 씨비1 9엔제이 체리턴 폴번로드 110 에이알엠 리미티드
만셀, 데이비드 해나
영국 캠브리지 캠브리지셔 씨비1 9엔제이 체리턴 폴번로드 110 에이알엠 리미티드
(74) 대리인
이화익

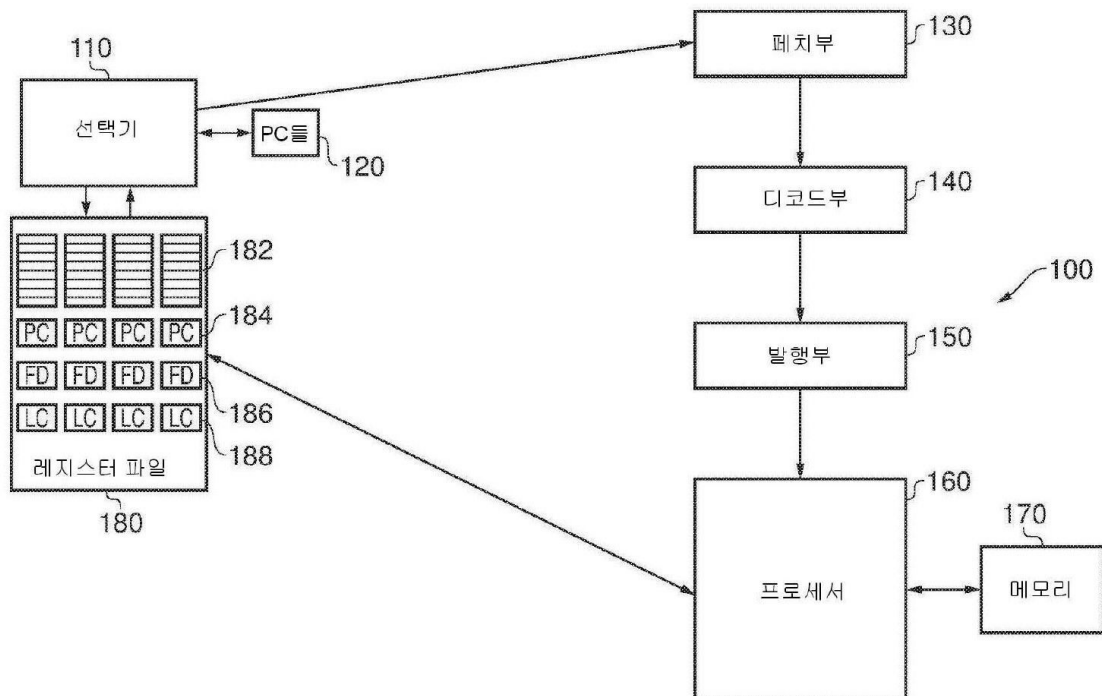
전체 청구항 수 : 총 27 항

(54) 발명의 명칭 복수의 쓰레드를 실행하기 위한 데이터 처리장치에서의 공유 자원

(57) 요약

데이터 처리장치(100)는, 쓰레드들을 실행하고, 적어도 상기 쓰레드들의 서브세트에 대해 실행되는 명령어를 식별하는 일반 프로그램 카운터(PC)(120)를 구비한다. 각 쓰레드는, 쓰레드 PC(184)를 갖는다. 상기 쓰레드들의 서브세트는, 공유 자원에 대한 배타적 액세스를 추적하는 적어도 하나의 록 파라미터(188, 500-504)를 갖는다. 프
(뒷면에 계속)

대표도



로세서(160)는, 쓰레드에 대해 실행된 제1 명령어에 응답하여, 쓰레드가 상기 공유 자원에 대한 배타적 액세스를 얻었다는 것을 나타내기 위해 상기 적어도 하나의 룩 파라미터(188, 500-504)를 변경한다. 상기 프로세서는, 쓰레드에 대해 실행된 제2 명령어에 응답하여, 쓰레드가 더 이상 배타적 액세스를 갖지 않는 것을 나타내기 위해 상기 적어도 하나의 룩 파라미터(188, 500-504)를 변경한다. 선택기(110)는, 상기 적어도 하나의 룩 파라미터(188, 500-504)에 근거하여 쓰레드들의 상기 서브세트 중 하나를 선택하고, 상기 일반 PC(120)를 상기 쓰레드 PC(184)로 선택한다.

명세서

청구범위

청구항 1

복수의 쓰레드를 실행하도록 구성된 데이터 처리장치로서,

적어도 상기 복수의 쓰레드의 서브세트에 대해 실행되는 명령어들로 이루어진 스트림에서의 하나의 명령어를 식별하는 일반 프로그램 카운터 - 상기 서브세트에서의 각 쓰레드가 상기 명령어들로 이루어진 스트림에서의 하나의 명령어를 식별하는 관련된 쓰레드 프로그램 카운터를 가짐-

상기 쓰레드들의 상기 서브세트의 선택된 쓰레드를 선택하고, 상기 일반 프로그램 카운터를 상기 선택된 쓰레드와 관련된 상기 쓰레드 프로그램 카운터로 설정하도록 구성된 선택기; 및

상기 선택된 쓰레드를 포함한 상기 쓰레드들의 상기 서브세트의 하나 이상에 대해 상기 일반 프로그램 카운터에 의해 식별된 명령어를 실행하도록 구성된 프로세서를 구비하고,

상기 쓰레드들의 상기 서브세트는 상기 쓰레드들의 상기 서브세트 중 어느 것이 공유 자원에 대한 배타적 액세스를 갖는지를 추적하는 적어도 하나의 록 파라미터와 관련되고;

상기 프로세서는, 쓰레드가 그 쓰레드에 대해 실행된 제1 명령어에 응답하여 상기 공유 자원에 대한 배타적 액세스를 얻었다는 것을 나타내기 위해 상기 적어도 하나의 록 파라미터를 변경하고, 상기 쓰레드가 그 쓰레드에 대해 실행된 제2 명령어에 응답하여 더 이상 상기 공유 자원에 대한 배타적 액세스를 갖지 않는 것을 나타내기 위해 상기 쓰레드와 관련된 상기 적어도 하나의 록 파라미터를 변경하도록 구성되고;

상기 선택기는 상기 적어도 하나의 록 파라미터에 근거하여 상기 선택된 쓰레드를 선택하도록 구성된, 데이터 처리장치.

청구항 2

제 1 항에 있어서,

상기 프로세서는, 상기 쓰레드에 대해 실행된 적어도 하나의 록킹 명령어를 포함하는 록킹 시퀀스에 응답하여 상기 공유 자원에 대한 쓰레드 배타적 액세스를 허가하도록 구성된, 데이터 처리장치.

청구항 3

제 2 항에 있어서,

상기 록킹 시퀀스는 상기 제1 명령어를 포함하지 않는, 데이터 처리장치.

청구항 4

제 1 항 내지 제 3 항 중 어느 한 항에 있어서,

상기 프로세서는, 쓰레드에 대해 실행된 적어도 하나의 언록킹 명령어를 포함하는 언록킹 시퀀스에 응답하여 상기 쓰레드를 위해 상기 공유 자원에 대한 배타적 액세스를 양보하도록 구성된, 데이터 처리장치.

청구항 5

제 4 항에 있어서,

상기 언로킹 시퀀스는 상기 제2 명령어를 포함하지 않는, 데이터 처리장치.

청구항 6

제 1 항 내지 제 5 항 중 어느 한 항에 있어서,

상기 선택기는, 상기 공유 자원에 대한 배타적 액세스를 갖지 않는 것으로서 상기 적어도 하나의 록 파라미터에 의해 나타낸 제2 쓰레드보다 우선적으로 상기 공유 자원에 대한 배타적 액세스를 갖는 것으로서 상기 적어도 하나의 록 파라미터에 의해 나타낸 제1 쓰레드를, 상기 선택된 쓰레드로서 선택하도록 구성된, 데이터 처리장치.

청구항 7

제 1 항 내지 제 6 항 중 어느 한 항에 있어서,

상기 선택기는, 상기 쓰레드들의 서브세트의 각 쓰레드와 관련된 함수 호출 깊이 파라미터에 근거하여 상기 선택된 쓰레드를 선택하도록 구성된, 데이터 처리장치.

청구항 8

제 1 항 내지 제 7 항 중 어느 한 항에 있어서,

상기 선택기는, 상기 쓰레드의 서브세트의 각 쓰레드와 관련된 상기 쓰레드 프로그램 카운터에 근거하여 상기 선택된 쓰레드를 선택하도록 구성된, 데이터 처리장치.

청구항 9

제 1 항 내지 제 8 항 중 어느 한 항에 있어서,

상기 서브세트의 각 쓰레드는 대응한 록 파라미터를 가지고, 상기 프로세서는, 쓰레드에 대해 실행된 제1 명령어에 응답하여 상기 공유 자원에 대한 배타적 액세스를 상기 쓰레드가 얻었다는 것을 나타내기 위해 상기 쓰레드에 대응한 상기 록 파라미터를 변경하고, 상기 쓰레드에 대해 실행된 제2 명령어에 응답하여 상기 공유 자원에 대한 배타적 액세스를 상기 쓰레드가 더 이상 갖지 않는다는 것을 나타내기 위해 상기 쓰레드에 대응한 상기 록 파라미터를 변경하도록 구성된, 데이터 처리장치.

청구항 10

제 9 항에 있어서,

상기 프로세서는, 제1 쓰레드가 제1 공유 자원에 대해 배타적 액세스를 갖고 제2 쓰레드가 제2 공유 자원에 대한 배타적 액세스를 갖게 허용하도록 구성되는, 데이터 처리장치.

청구항 11

제 9 항 또는 제 10 항에 있어서,

상기 선택기는, 쓰레드들의 상기 서브세트의 각 쓰레드의 대응한 록 파라미터에 근거하여 쓰레드들의 제1 세트를 선택하도록 구성되고,

상기 선택기는, 상기 쓰레드들의 제1 세트 중에서, 함수 호출 깊이 파라미터에 근거하여 쓰레드들의 제2 세트를 선택하도록 구성되고,

상기 선택기는, 상기 쓰레드들의 제2 세트 중에서 상기 선택된 쓰레드를, 상기 쓰레드들의 제2 세트에서 상기

쓰레드들과 관련된 쓰레드 프로그램 카운터들에 근거하여 선택하도록 구성되는, 데이터 처리장치.

청구항 12

제 1 항 내지 제 8 항 중 어느 한 항에 있어서,

상기 적어도 하나의 록 파라미터는, 쓰레드들의 상기 서브세트간에 공유된 공유 록 파라미터와, 쓰레드들의 상기 서브세트 중 어느 것이 공유 자원에 대한 배타적 액세스를 갖는지를 나타내는 록 소유자 파라미터를 포함하는, 데이터 처리장치.

청구항 13

제 12 항에 있어서,

상기 공유 록 파라미터는, 쓰레드들의 상기 서브세트 중 어떤 것이 공유 자원에 대한 배타적 액세스를 갖는지를 나타내는 록 플래그를 포함하는, 데이터 처리장치.

청구항 14

제 12 항 또는 제 13 항에 있어서,

상기 공유 록 파라미터는, 상기 록 소유자 파라미터에 의해 나타낸 쓰레드가 배타적 액세스를 갖는 자원의 수를 나타내는 록 카운트를 포함하는, 데이터 처리장치.

청구항 15

제 12 항 내지 제 14 항 중 어느 한 항에 있어서,

제1 쓰레드가 제1 공유 자원에 대한 배타적 액세스를 갖는 경우, 상기 프로세서는 제2 공유 자원에 대한 배타적 액세스를 갖는 제2 쓰레드를 방지하도록 구성된, 데이터 처리장치.

청구항 16

제 12 항 내지 제 15 항 중 어느 한 항에 있어서,

상기 적어도 하나의 록 파라미터는 쓰레드가 공유 자원에 대한 배타적 액세스를 갖는 것을 나타내는 경우, 상기 선택기는 상기 록 소유자 파라미터에 의해 나타낸 쓰레드를 상기 선택된 쓰레드로서 선택하도록 구성되고,

상기 적어도 하나의 록 파라미터는 쓰레드가 공유 자원에 대한 배타적 액세스를 갖지 않는 것을 나타내는 경우, 상기 선택기는, 쓰레드들의 상기 서브세트 중에서, 함수 호출 깊이 파라미터에 근거하여 쓰레드들의 또 다른 서브세트를 선택하고, 상기 쓰레드들의 상기 또 다른 서브세트에서 쓰레드들과 관련된 상기 쓰레드 프로그램 카운터들에 근거하여 쓰레드들의 상기 또 다른 서브세트 중에서 상기 선택된 쓰레드들을 선택하도록 구성된, 데이터 처리장치.

청구항 17

제 1 항 내지 제 16 항 중 어느 한 항에 있어서,

상기 프로세서는, 상기 제1 명령어에 응답하여 상기 적어도 하나의 록 파라미터를 증분하도록 구성되고,

상기 프로세서는, 상기 제2 명령어에 응답하여 상기 적어도 하나의 록 파라미터를 감분하도록 구성되는, 데이터 처리장치.

청구항 18

제 17 항에 있어서,

상기 적어도 하나의 록 파라미터는, 쓰레드가 배타적 액세스를 갖는 공유 자원들의 수를 나타내는, 데이터 처리 장치.

청구항 19

제 1 항 내지 제 18 항 중 어느 한 항에 있어서,

상기 쓰레드들의 상기 서브세트의 하나 이상은 상기 선택된 쓰레드의 대응한 쓰레드 파라미터들에 일치하는 하나 이상의 관련된 쓰레드 파라미터들을 갖는 쓰레드들을 포함하고, 상기 하나 이상의 관련된 쓰레드 파라미터들이 적어도 상기 쓰레드 프로그램 카운터를 포함하는, 데이터 처리장치.

청구항 20

제 1 항 내지 제 19 항 중 어느 한 항에 있어서,

상기 명령어는 마이크로 연산을 포함하는, 데이터 처리장치.

청구항 21

제 1 항 내지 제 20 항 중 어느 한 항에 있어서,

상기 제1 명령어와 상기 제2 명령어는, 적어도 하나의 사전 정의된 명령어 세트에서의 NOOP 명령어로서 부호화되는, 데이터 처리장치.

청구항 22

제 1 항 내지 제 21 항 중 어느 한 항에 있어서,

상기 프로세서는, 상기 제1 명령어와 상기 제2 명령어 중 적어도 하나에 응답하여 또 다른 연산을 행하도록 구성되는, 데이터 처리장치.

청구항 23

제 22 항에 있어서,

상기 또 다른 연산은, 상기 공유 자원에 대한 배타적 액세스를 얻기 위한 록킹 시퀀스의 일부인 연산과, 상기 공유 자원에 대한 배타적 액세스를 양보하기 위한 언록킹 시퀀스의 일부인 연산과, 상기 공유 자원을 액세스하기 위한 연산과, 쓰레드가 상기 공유 자원에 대한 배타적 액세스를 가졌는지를 판단하기 위한 연산과, 비교 및 교환 연산 중 하나 이상인, 데이터 처리장치.

청구항 24

제 1 항 내지 제 23 항 중 어느 한 항에 있어서,

상기 공유 자원은 메모리에서의 데이터를 포함하는, 데이터 처리장치.

청구항 25

제 24 항에 있어서,

상기 프로세서는, 상기 공유 자원에 대한 배타적 액세스를 양보하기에 앞서 메모리 배리어 연산을 메모리에 발
행하도록 구성되는, 데이터 처리장치.

청구항 26

복수의 쓰레드의 서브세트에 대해 명령어들로 이루어진 스트림을 실행하는 데이터 처리 방법으로서, 상기 서브
세트에서의 각 쓰레드가 상기 명령어들로 이루어진 스트림에서의 하나의 명령어를 식별하는 관련된 쓰레드 프로
그램 카운터를 갖고, 상기 데이터 처리 방법은,

상기 쓰레드들의 상기 서브세트의 선택된 쓰레드를 선택하고, 일반 프로그램 카운터를 상기 선택된 쓰레드와 관
련된 상기 쓰레드 프로그램 카운터로 설정하는 단계-상기 일반 프로그램 카운터가 쓰레드들의 상기 서브세트에
대해 실행되는 명령어들로 이루어진 스트림에서의 하나의 명령어를 식별함-; 및

상기 선택된 쓰레드를 포함한 상기 쓰레드들의 상기 서브세트의 하나 이상에 대해 상기 일반 프로그램 카운터에
의해 식별된 명령어를 실행하는 단계를 포함하고,

상기 쓰레드들의 상기 서브세트는 상기 쓰레드들의 상기 서브세트 중 어느 것이 공유 자원에 대한 배타적 액세
스를 갖는지를 추적하는 적어도 하나의 룩 파라미터와 관련되고;

상기 적어도 하나의 룩 파라미터는, 쓰레드가 그 쓰레드에 대해 실행된 제1 명령어에 응답하여 상기 공유 자원
에 대한 배타적 액세스를 얻었다는 것을 나타내도록 변경되고, 상기 쓰레드가 그 쓰레드에 대해 실행된 제2 명
령어에 응답하여 더 이상 상기 공유 자원에 대한 배타적 액세스를 갖지 않는 것을 나타내도록 변경되고;

상기 선택기는 상기 적어도 하나의 룩 파라미터에 근거하여 상기 선택된 쓰레드를 선택하도록 구성된, 데이터
처리 방법.

청구항 27

복수의 쓰레드를 실행하도록 구성된 데이터 처리장치로서,

적어도 복수의 쓰레드의 서브세트에 대해 실행되는 명령어들로 이루어진 스트림에서의 하나의 명령어를 식별하
는 일반 프로그램 카운터 수단 - 상기 서브세트에서의 각 쓰레드가 상기 명령어들로 이루어진 스트림에서의 하
나의 명령어를 식별하는 관련된 쓰레드 프로그램 카운터 수단을 가짐-;

상기 쓰레드들의 상기 서브세트의 선택된 쓰레드를 선택하고, 상기 일반 프로그램 카운터 수단을 상기 선택된
쓰레드와 관련된 상기 쓰레드 프로그램 카운터 수단에 설정하는 선택 수단; 및

상기 선택된 쓰레드를 포함한 상기 쓰레드들의 상기 서브세트의 하나 이상에 대해 상기 일반 프로그램 카운터
수단에 의해 식별된 명령어를 실행하는 프로세서 수단을 구비하고,

상기 쓰레드들의 상기 서브세트는 상기 쓰레드들의 상기 서브세트 중 어느 것이 공유 자원에 대한 배타적 액세
스를 갖는지를 추적하는 적어도 하나의 룩 파라미터와 관련되고;

상기 프로세서 수단은, 쓰레드가 그 쓰레드에 대해 실행된 제1 명령어에 응답하여 상기 공유 자원에 대한 배타
적 액세스를 얻었다는 것을 내기 위해 상기 적어도 하나의 룩 파라미터를 변경하고, 상기 쓰레드가 그 쓰레드
에 대해 실행된 제2 명령어에 응답하여 더 이상 상기 공유 자원에 대한 배타적 액세스를 갖지 않는 것을 나타내
기 위해 상기 적어도 하나의 룩 파라미터를 변경하기 위한 수단이고;

상기 선택 수단은 상기 적어도 하나의 룩 파라미터에 근거하여 상기 선택된 쓰레드를 선택하기 위한 수단인, 데
이터 처리장치.

발명의 설명

기술 분야

[0001] 본 기술은, 데이터 처리분야에 관한 것이다. 특히, 본 기술은, 복수의 쓰레드가 실행하는 데이터 처리 장치 및 방법을 생각한다.

배경 기술

[0002] 데이터 처리장치는 복수의 쓰레드를 실행할 수도 있다. 상기 쓰레드들은 정확히 같은 방식으로 진행되어도 된다. 특히, 각 쓰레드는 자신의 프로그램 카운터를 유지할 수도 있다. 시스템 자체는 자신의 일반 프로그램 카운터를 유지하여, 그 일반 프로그램 카운터가 언급하는 명령어는 그 일반 프로그램 카운터와 일치하는 프로그램 카운터들을 갖는 쓰레드들에 대해 실행된다. 이러한 종류의 시스템은, 예를 들면, 단일 명령어 다중 쓰레드(SIMT)로서 알려질 수도 있다. 따라서, 각 단계에서, 단일 명령어는 적어도 상기 복수의 쓰레드의 서브세트에 대해 실행된다. 상기 쓰레드들의 서브세트에 대해 명령어가 실행되었으면, 상기 일반 프로그램 카운터는 새로운 명령어라고 하도록 변화되어도 된다. 이러한 시스템에서는, 상기 쓰레드들 각각에 의해 유지된 상기 프로그램 카운터들 중 최소의 프로그램 카운터와 일치하도록 상기 일반 프로그램 카운터를 설정하는 것이 바람직할 수도 있다. 이렇게, 가장 많이 떨어진 상기 쓰레드는, 진행하여서 상위 프로그램 카운터들을 갖는 쓰레드들의 따라잡기를 가능하게 될 수 있다. 이에 따라, 쓰레드들은 수렴하게 된다, 즉 단일 명령어가 가능한 많이 쓰레드들에 대해 실행될 수도 있도록 동일한 프로그램 카운터 값을 공유하게 될 수도 있다.

[0003] 이러한 방식에 대한 변형은 각 쓰레드의 함수 호출 깊이를 추적하는 것을 수반한다. 특히, 쓰레드가 함수 호출을 만들 때마다 그 쓰레드들에 대한 함수 호출 깊이 카운터가 증분되고, 그 쓰레드가 함수 호출로부터 복귀할 때마다 그 쓰레드에 대한 함수 호출 깊이 카운터가 감분된다. 상기 일반 프로그램 카운터가 변화될 때, 그것은, 먼저 최고의 함수 호출 깊이 카운터를 갖는 쓰레드와 다음에 (이러한 쓰레드들이 다수일 경우) 최저의 프로그램 카운터 값을 갖는 세트로부터의 쓰레드나 쓰레드들에 일치하도록 설정된다. 달리 말하면, 프로그램 카운터 값은, 최고의 함수 호출 깊이를 갖는 쓰레드들 전체중에서만 생각된다. 이에 따라, 이것은, 프로그램 카운터 값만을 생각했을 경우에 일어날 수 있는 성능 문제나 데드락(deadlock) 상황을 방지하는데 도움이 된다. 상기 쓰레드들의 서브세트가 함수 호출을 만드는 경우, 이에 따라 그 쓰레드들과 관련된 프로그램 카운터가 극적으로 증가하게 될 수도 있어, 그 함수 자체로부터 다음에 실행되어야 하는 명령어들은 최저의 프로그램 카운터 값을 갖지 않는다. 상기 함수 호출 깊이에 근거하여 실행하기 위한 쓰레드들을 선택함으로써, 상기 함수는 먼저 최저의 프로그램 카운터 값을 갖지 않는 경우에도 처리될 수 있다.

[0004] 상기 2개의 메커니즘 양쪽은, 실행될 코드가 배타적 액세스를 필요로 하는 공유 자원을 포함하는 데드락을 겪기 쉽다. 예를 들면, 록(lock)은 임의의 한 순간에 단일 쓰레드에 의해 보유되어서, 그 쓰레드와 상기 공유 자원에 대한 그 쓰레드 액세스만이 허용될 수도 있다.

[0005] 상기 공유 자원에 액세스하는 쓰레드는, 그 공유 자원에 액세스할 수 없는 나머지 쓰레드들보다 높은 프로그램 카운터를 가질 수도 있다. 따라서, 상기 나머지 쓰레드들은 실행할 수 있는 한편, 상기 공유 자원에 액세스하는 쓰레드는 실행할 수 없을 것이다. 데드락은, 상기 나머지 쓰레드들이 공유 자원에 액세스할 수 없고 또 다른 쓰레드가 공유 자원에 액세스하기 때문에 일어날 것이다. 그렇지만, 상기 공유 자원에 액세스 하는 쓰레드는 결코 실행되면 안될 수도 있다. 이에 따라, 그 쓰레드들 중 아무것도 어떠한 진행도 하지 않고 상기 시스템이 정지한다.

발명의 내용

[0006] 본 기술의 일 측면에서는, 복수의 쓰레드를 실행하도록 구성된 데이터 처리장치를 제공하고, 상기 데이터 처리장치는, 적어도 복수의 쓰레드의 서브세트에 대해 실행되는 명령어들로 이루어진 스트림에서의 하나의 명령어를 식별하는 일반 프로그램 카운터 - 상기 서브세트에서의 각 쓰레드가 상기 명령어들로 이루어진 스트림에서의 하나의 명령어를 식별하는 관련된 쓰레드 프로그램 카운터를 가짐-; 상기 쓰레드들의 상기 서브세트의 선택된 쓰레드를 선택하고, 상기 일반 프로그램 카운터를 상기 선택된 쓰레드와 관련된 상기 쓰레드 프로그램 카운터로 설정하도록 구성된 선택기; 및 상기 선택된 쓰레드를 포함한 상기 쓰레드들의 상기 서브세트의 하나 이상에 대해 상기 일반 프로그램 카운터에 의해 식별된 명령어를 실행하도록 구성된 프로세서를 구비하고, 상기 쓰레드들의 상기 서브세트는 상기 쓰레드들의 상기 서브세트 중 어느 것이 공유 자원에 대한 배타적 액세스를 갖는지를 추적하는 적어도 하나의 록 파라미터와 관련되고; 상기 프로세서는, 쓰레드가 그 쓰레드에 대해 실행된 제1 명령어에 응답하여 상기 공유 자원에 대한 배타적 액세스를 얻었다는 것을 나타내기 위해 상기 적어도 하나의 록

파라미터를 변경하고, 상기 쓰레드가 그 쓰레드에 대해 실행된 제2 명령어에 응답하여 더 이상 상기 공유 자원에 대한 배타적 액세스를 갖지 않는 것을 나타내기 위해 상기 적어도 하나의 록 파라미터를 변경하도록 구성되고; 상기 선택기는 각 쓰레드와 관련된 상기 록 파라미터에 근거하여 상기 선택된 쓰레드를 선택하도록 구성된다.

- [0007] 본 기술의 다른 측면에서는, 복수의 쓰레드의 서브세트에 대해 명령어들로 이루어진 스트림을 실행하는 데이터 처리 방법을 제공하고, 상기 서브세트에서의 각 쓰레드가 상기 명령어들로 이루어진 스트림에서의 하나의 명령어를 식별하는 관련된 쓰레드 프로그램 카운터를 갖고, 상기 데이터 처리 방법은,
- [0008] 상기 쓰레드들의 상기 서브세트의 선택된 쓰레드를 선택하고, 일반 프로그램 카운터를 상기 선택된 쓰레드와 관련된 상기 쓰레드 프로그램 카운터로 설정하는 단계-상기 일반 프로그램 카운터가 쓰레드들의 상기 서브세트에 대해 실행되는 명령어들로 이루어진 스트림에서의 하나의 명령어를 식별함-; 및
- [0009] 상기 선택된 쓰레드를 포함한 상기 쓰레드들의 상기 서브세트의 하나 이상에 대해 상기 일반 프로그램 카운터에 의해 식별된 명령어를 실행하는 단계를 포함하고,
- [0010] 상기 쓰레드들의 상기 서브세트는 상기 쓰레드들의 상기 서브세트 중 어느 것이 공유 자원에 대한 배타적 액세스를 갖는지를 추적하는 적어도 하나의 록 파라미터와 관련되고;
- [0011] 상기 적어도 하나의 록 파라미터는, 쓰레드가 그 쓰레드에 대해 실행된 제1 명령어에 응답하여 상기 공유 자원에 대한 배타적 액세스를 얻었다는 것을 나타내도록 변경되고, 상기 쓰레드가 그 쓰레드에 대해 실행된 제2 명령어에 응답하여 더 이상 상기 공유 자원에 대한 배타적 액세스를 갖지 않는 것을 나타내도록 변경되고;
- [0012] 상기 선택기는 상기 적어도 하나의 록 파라미터에 근거하여 상기 선택된 쓰레드를 선택하도록 구성된다.
- [0013] 다른 측면에서는, 복수의 쓰레드를 실행하도록 구성된 데이터 처리장치를 제공하고, 상기 데이터 처리장치는,
- [0014] 적어도 복수의 쓰레드의 서브세트에 대해 실행되는 명령어들로 이루어진 스트림에서의 하나의 명령어를 식별하는 일반 프로그램 카운터 수단 - 상기 서브세트에서의 각 쓰레드가 상기 명령어들로 이루어진 스트림에서의 하나의 명령어를 식별하는 관련된 쓰레드 프로그램 카운터 수단을 가짐-;
- [0015] 상기 쓰레드들의 상기 서브세트의 선택된 쓰레드를 선택하고, 상기 일반 프로그램 카운터 수단을 상기 선택된 쓰레드와 관련된 상기 쓰레드 프로그램 카운터 수단에 설정하는 선택 수단; 및
- [0016] 상기 선택된 쓰레드를 포함한 상기 쓰레드들의 상기 서브세트의 하나 이상에 대해 상기 일반 프로그램 카운터 수단에 의해 식별된 명령어를 실행하는 프로세서 수단을 구비하고,
- [0017] 상기 쓰레드들의 상기 서브세트는 상기 쓰레드들의 상기 서브세트 중 어느 것이 공유 자원에 대한 배타적 액세스를 갖는지를 추적하는 적어도 하나의 록 파라미터와 관련되고;
- [0018] 상기 프로세서 수단은, 쓰레드가 그 쓰레드에 대해 실행된 제1 명령어에 응답하여 상기 공유 자원에 대한 배타적 액세스를 얻었다는 것을 나타내기 위해 상기 적어도 하나의 록 파라미터를 변경하고, 상기 쓰레드가 그 쓰레드에 대해 실행된 제2 명령어에 응답하여 더 이상 상기 공유 자원에 대한 배타적 액세스를 갖지 않는 것을 나타내기 위해 상기 적어도 하나의 록 파라미터를 변경하기 위한 수단이고;
- [0019] 상기 선택 수단은 상기 적어도 하나의 록 파라미터에 근거하여 상기 선택된 쓰레드를 선택하기 위한 수단이다.

도면의 간단한 설명

- [0020] 본 기술의 또 다른 측면, 특징, 이점 및 실시예를 아래의 도면을 참조하여 예시로만 설명한다:

- 도 1은 일 실시 예에 따른 데이터 처리장치를 도시한 것이고,
- 도 2는 프로그램 카운터 값이 다른 다수의 상이한 쓰레드의 실행 예를 도시한 것이고,
- 도 3은 S IMT 시스템에서 일어날 수도 있는 코드를 도시한 것이고,
- 도 4는 도 3의 데드락을 어떻게 방지하는지를 설명하는 제2 코드 예를 도시한 것이고,
- 도 5는 SIMT 시스템의 쓰레드간에 어떻게 선택이 일어나는지를 흐름도 형태로 도시한 것이고,
- 도 6은 쓰레드들의 실행 방법을 흐름도 형태로 도시한 것이고,

도 7은 쓰레드들의 서브세트에 대해 유지된 상태 데이터의 다른 예를 나타내고,

도 8은 제3 코드 예를 나타내고,

도 9는 도 7의 상태 데이터를 사용하여 쓰레드 프로그램 카운터를 선택하는 방법을 도시한 흐름도다.

발명을 실시하기 위한 구체적인 내용

- [0021] 제1 예에서, 쓰레드들의 서브세트는 상기 서브세트 중 어느 쓰레드가 공유 자원에 대한 배타적 액세스를 갖는지를 추적하는 적어도 하나의 록 파라미터와 관련되어 있다. 상기 프로세서는, 쓰레드에 의해 실행된 제1 명령어에 응답하여 상기 적어도 하나의 록 파라미터를 변경함으로써 상기 쓰레드가 그 공유 자원에 대한 배타적 액세스를 얻었다는 것을 나타내어도 된다. 마찬가지로, 상기 프로세서는, 제2 명령어에 응답하여 상기 적어도 하나의 록 파라미터를 변경함으로써 상기 쓰레드가 더 이상 그 공유 자원에 대한 배타적 액세스를 갖지 않는 것을 나타내어도 된다. 상기 일반 프로그램 카운터에 대한 값으로서 쓰레드 프로그램 카운터를 사용하는 선택된 쓰레드를 선택하고, 이렇게 함으로써 쓰레드들의 상기 서브세트에 대해 다음에 실행할 명령어인지를 판정할 때, 상기 데이터 처리장치는 상기 적어도 하나의 록 파라미터를 생각할 수도 있다. 상기 록 파라미터에 근거하여 상기 일반 프로그램 카운터의 값을 선택함으로써, 어느 쓰레드들이 임의의 특별한 순간에 실행하여서 데드락이 발생할 기회를 감소 또는 방지할 수 있는지를 제어하는 것이 가능하다.
- [0022] 예를 들면, 상기 제1 명령어는 공유 자원에 대한 배타적 액세스를 얻거나 얻었을 때 실행되어도 되고, 상기 제2 명령어는 쓰레드가 공유 자원에 대한 액세스를 양보하거나 양보하였을 때 실행되어도 된다.
- [0023] 상기 프로세서는, 상기 공유 자원에 대한 쓰레드에 대해 실행된 적어도 하나의 록킹(locking) 명령어를 포함하는 록킹 시퀀스에 응답하여 그 쓰레드에 대한 배타적 액세스를 허가하도록 구성되어도 된다. 록킹 시퀀스는 각종 비교, 로드 및 스토어를 수반하는 다수의 상이한 명령어를 포함하여도 된다. 그 시퀀스 내에는 마지막으로 공유 자원의 "소유권"을 설정하는 록킹 명령어가 있을 수도 있다. 일부의 예들에서, 상기 록킹 명령어는, 상기 공유 자원에 대한 배타적 액세스를 갖는 쓰레드의 아이덴티티를 저장 또는 기록시킬 수도 있다. 다른 예들에서, 상기 록킹 명령어는, 쓰레드가 상기 록의 소유권을 갖는지를 명시적으로 식별하지 않고, 상기 자원이 록되었다는 표시를 설정하는 것만 할 수도 있다. 상기 록킹 시퀀스는, 일부의 경우에 상기 제1 명령어를 포함하지 않을 수도 있다. 달리 말하면, 쓰레드에 의해 상기 공유 자원에 대한 배타적 액세스를 얻는 상기 록킹 시퀀스는, 상기 쓰레드가 상기 공유 자원에 대한 배타적 액세스를 얻었다는 것을 상기 프로세서에 나타내는 상기 제1 명령어에서 분리되어도 된다. 이 때문에, 코드는, 제1 명령어가 선행하거나 후속하는 상기 록킹 명령어로 이루어진 록킹 시퀀스를 포함할 수도 있다. 이러한 접근법은, 상기 제1 명령어를 지원하지 않는 레거시 시스템과의 역호환성을 보장하는데 유용할 수도 있다. 상기 제1 명령어가 지원되지 않는 경우에도, 그래도 상기 록킹 시퀀스는 통상 상기 레거시 시스템에 의해 처리되어도 된다. 일부의 예들은, 상기 제1 명령어 이전에 록킹 시퀀스를 실행하여, 쓰레드가 먼저 상기 공유 자원에 대한 배타적 액세스를 얻으려고 하고 나서, 성공한 경우, 그에 따라서 상기 록 파라미터는 상기 제1 명령어를 실행함으로써 갱신된다. 그 밖의 예들은, 상기 록킹 시퀀스 이전에 상기 제1 명령어를 실행하여, 실제로 상기 록킹 시퀀스를 사용하여 록이 구현되기 전에 상기 제1 명령어는 쓰레드에 대한 록킹 특권을 요구하는데 사용되는 것이 효과적이다. 두 번째의 경우에는, 상기 록킹 시퀀스로의 진행은, 또 다른 쓰레드가 이미 록을 가졌는지에 의존할 수도 있는, 성공적으로 실행되는 제1 명령어에 의존할 수도 있다. 그 밖의 실시예에서, 상기 록킹 시퀀스는 제1 명령어를 포함한다. 달리 말하면, 상기 록킹 시퀀스는, 공유 자원에 대한 배타적 액세스를 상기 쓰레드들 중 하나에 허가하고, 또한 그 쓰레드가 상기 공유 자원에 대한 배타적 액세스를 갖는 것을 나타내도록 그 쓰레드에 대한 록 파라미터를 설정한다. 따라서, 상기 록킹 시퀀스와 상기 제1 명령어를 구현하는 방식은 다르다.
- [0024] 마찬가지로, 상기 프로세서는, 상기 공유 자원에 대한 쓰레드에 대해 실행된 적어도 하나의 언록킹(unlocking) 명령어를 포함하는 언록킹 시퀀스에 응답하여 그 쓰레드에 대한 배타적 액세스를 양보하도록 구성되어도 된다. 상기 언록킹 시퀀스는, 특별한 쓰레드에 의해 공유 자원에 대한 배타적 액세스를 잃는 비교, 로드 또는 스토어 등으로 이루어진 다수의 상이한 명령어를 포함하여도 된다. 공유 자원에 대한 배타적 액세스는, 자원에 대한 록을 제거하여서, 또는 예를 들면 상이한 쓰레드에 대한 배타적 액세스를 통과시켜서, 양보되어도 된다. 상기 언록킹 시퀀스내에는, 어느 쓰레드가 공유 자원에 대한 배타적 액세스를 갖는지를 지우기 위해(clear), 또는 특별한 쓰레드가 더 이상 공유 자원에 대해 액세스하지 않는 것을 나타내기 위해, 사용되는 언록킹 명령어가 있을 수도 있다. 달리 말하면, 상기 언록킹 명령어는, 특별한 쓰레드가 더 이상 공유 자원에 대한 배타적 액세스를 갖지 않는 것을 나타내기 위해 변수를 설정 또는 지우는데 사용되어도 된다. 일부의 실시예에서, 상기 언록킹

시퀀스는 제2 명령어를 포함하지 않는다. 달리 말하면, 상기 언로킹 시퀀스는, 상기 쓰레드가 더 이상 상기 공유 자원에 대한 배타적 액세스를 갖지 않는 것을 나타내기 위해 쓰레드에 관련된 록 파라미터를 변경하지 않는다. 이에 따라, 상기 언로킹을 행하는 상기 코드에서의 명령어나 명령어들과, 쓰레드가 더 이상 공유 자원에 대해 액세스 하지 않는 것을 나타내도록 상기 록 파라미터를 설정하는 상기 코드에서의 명령어간에 분리되어 있다. 이것은, 상술한 것처럼 역호환성 이유 때문에 유용하다. 제2 명령어는 상기 언로킹 시퀀스 후에 실행되어도 된다. 그 밖의 실시예에서, 상기 언로킹 시퀀스는 상기 제2 명령어를 포함한다.

[0025] 상기 선택기는, 상기 공유 자원에 대한 배타적 액세스를 갖지 않는 것으로서 상기 적어도 하나의 록 파라미터에 의해 나타낸 제2 쓰레드보다 우선적으로 상기 공유 자원에 대한 배타적 액세스를 갖는 것으로서 상기 적어도 하나의 록 파라미터에 의해 나타낸 제1 쓰레드를, 상기 선택된 쓰레드로서 선택하도록 구성된다. 달리 말하면, 상기 선택기는, 상기 공유 자원에 대한 배타적 액세스를 갖는 쓰레드가 우선적으로 처리되도록 구성되어도 된다. 이 때문에, 그 쓰레드는, 상기 공유 자원을 언로킹하기 위해서 상기 공유 자원의 그 사용을 완료 가능할 수도 있다. 이것이 데드락을 방지하는데 도움이 되는 것은, 상기 공유 자원을 언로킹할 능력이 있는 상기 제1 쓰레드가 실행하여서 그 공유 자원을 언로킹하도록 허용되기 때문이다.

[0026] 상기 선택기는, 상기 쓰레드들의 서브세트에서 각 쓰레드와 관련된 함수 호출 깊이 파라미터에 근거하여 상기 선택된 쓰레드를 선택하도록 구성되어도 된다. 달리 말하면, 상기 선택기는, 상기 일반 프로그램 카운터를 설정하는데 어느 쓰레드의 쓰레드 프로그램 카운터를 사용해야 하는지를 결정할 때, 상기 함수 깊이에 호출 파라미터와 적어도 하나의 록 파라미터의 양쪽을 생각하여도 된다.

[0027] 상기 선택기는, 상기 쓰레드의 서브세트의 각 쓰레드와 관련된 상기 쓰레드 프로그램 카운터에 근거하여 상기 선택된 쓰레드를 선택하도록 구성되어도 된다. 달리 말하면, 상기 선택기는, 쓰레드 프로그램 카운터를 상기 일반 프로그램 카운터로 설정할지를 결정할 때, 상기 쓰레드 프로그램 카운터와 상기 적어도 하나의 록 파라미터 양쪽을 생각하여도 된다.

[0028] 상기 선택된 쓰레드의 선택은 다른 방식으로 실시되어도 된다. 일부의 예들에서, 쓰레드는 직접 선택되어도 되고 나서, 그 쓰레드의 상기 쓰레드 프로그램 카운터는 상기 일반 프로그램 카운터에 이용되어도 된다. 그 밖의 예들에서, 특정 쓰레드 프로그램 카운터는, 주어진 프로그램 카운터 선택 알고리즘을 사용하여 선택되어도 된다 (대응한 쓰레드를 상기 선택된 쓰레드로서 암시적으로 선택함). 따라서, 일반적으로, 본 출원에서 선택된 쓰레드를 선택하는 것에 대한 참조는, 쓰레드와 관련된 쓰레드 프로그램 카운터를 선택하는 것을 포함하기 위한 것이다.

[0029] 상기 록 파라미터는 다수의 상이한 형태를 취하여도 된다. 일례에서, 상기 서브세트내의 각 쓰레드는, 대응한 록 파라미터를 가질 수도 있다. 상기 공유 자원에 대한 배타적 액세스를 얻는 쓰레드가 상기 제1 명령어를 실행할 때, 그 쓰레드에 대한 상기 록 파라미터는, 상기 쓰레드가 상기 공유 자원에 대한 배타적 액세스를 얻었다는 것을 나타내도록 변경될 수 있다. 마찬가지로, 상기 쓰레드가 상기 쓰레드에 대한 배타적 액세스를 양보할 때, 상기 대응한 록 파라미터도, 그 쓰레드에 대해 실행된 제2 명령어에 응답하여, 상기 쓰레드가 더 이상 상기 공유 자원에 대한 배타적 액세스를 갖지 않는 것을 나타내도록 변경되어도 된다. 예를 들면, 쓰레드마다 상기 록 파라미터는, 공유 자원이 현재 얼마나 그 쓰레드에 의해 록되는가를 나타낼 수도 있다. 이 경우에, 상기 프로세서는, 다중 쓰레드가 한 번에 다른 공유 자원에 대한 배타적 액세스를 갖도록 허가하여도 된다. 상기 선택된 쓰레드를 선택할 때, 상기 선택기는, 쓰레드들의 제1 세트를 선택하기 위해서(예를 들면, 최고 수의 록킹된 자원을 나타내는 록 파라미터를 갖는 쓰레드들을 선택하는 것) 상기 쓰레드들의 록 파라미터들을 생각하고 나서, 상기 쓰레드들의 제1 세트 중에서, 함수 호출 깊이에 근거하여 쓰레드들의 제2 세트를 선택하고 나서, 상기 쓰레드들의 제2 세트에서 상기 쓰레드들과 관련된 상기 쓰레드 프로그램 카운터들에 근거하여 상기 선택된 쓰레드로서 상기 쓰레드들의 제2 세트 중 하나를 선택하여도 된다. 이 접근법이 향상된 성능을 제공하는 것은, 다중 쓰레드가 둘 다 서로 병렬로 진행할 수 있도록 다른 자원에 대한 록들을 한 번에 유지하는 것이 가능하기 때문이다.

[0030] 달리 말하면, 상기 선택기는, 먼저 상기 함수 호출 깊이가 후속하는 쓰레드들의 록 파라미터를 생각하고 끝으로 상기 프로그램 카운터를 생각하도록 구성되어도 된다. 이러한 순서로 상기 파라미터들을 생각함으로써, 가장 적절한 쓰레드로부터의 명령어를 특별한 순간에 실행시키는 것이 가능할 수도 있다. 이때, 상기 선택기가 상기 제1 세트 또는 상기 제2 세트로부터의 선택을 행하게 할 필요가 없을 수도 있다. 예를 들면, 단일의 쓰레드만이 원하는 록 파라미터를 가지면, 그 쓰레드의 쓰레드 프로그램 카운터는 상기 일반 프로그램 카운터를 설정하는데 사용될 것이다. 마찬가지로, 상기 쓰레드들의 상기 제1 세트가 2개의 쓰레드를 포함하고 그들 2개의 쓰레드 중

하나만이 원하는 함수 호출 깊이를 가지면, 특히 상기 쓰레드들의 상기 제2 세트는 형성되지 않을 수도 있고, 그 대신에 원하는 함수 호출 깊이를 갖는 상기 쓰레드들의 제1 세트에서 쓰레드의 상기 쓰레드 프로그램 카운터는, 상기 일반 프로그램 카운터로 설정되어도 된다. 달리 표현하면, 비록 상기 선택기가 이전에 언급된 3개의 선택을 행하도록 구성되어도 되지만, 항상 상기 3개의 선택이 어느 경우에도 실시될 필요가 없을 수도 있다.

[0031] 다른 예에서, 상기 적어도 하나의 록 파라미터는, 상기 쓰레드들의 상기 서브세트간에 공유된 공유 록 파라미터와, 상기 쓰레드들의 상기 서브세트 중 어느 것이 공유 자원에 대한 배타적 액세스를 갖는지를 나타내는 록 소유자 파라미터를 포함하여도 된다. 일부의 경우에, 상기 공유 록 파라미터는, 상기 록 소유자 파라미터에 의해 나타낸 록 소유 쓰레드가 얼마나 자원에 대한 배타적 액세스를 갖는지를 나타내는 록 카운트를 포함할 수도 있다. 그 밖의 경우에, 상기 공유 록 파라미터는, 현재, 쓰레드들의 상기 서브세트 중 어떤 것이 공유 자원에 대한 배타적 액세스를 갖는지를 나타내는 록 플래그를 포함하여도 된다. 이러한 예에 의해, 상기 프로세서는, 쓰레드들의 서브세트 당 하나의 쓰레드에 대하여만 한 번에 공유 자원에 대한 배타적 액세스를 갖게 하도록 보장할 수도 있다. 따라서, 하나의 쓰레드가 제1 공유 자원에 대한 배타적 액세스를 가지면, 제2 쓰레드는 (제2 공유 자원이 상기 제1 공유 자원과 다를 경우에도) 상기 제2 공유 자원에 대한 배타적 액세스를 하면 안 된다. 이 접근법은, 다른 쓰레드들이 일련의 자원을 록킹하는 것을 통해 나아가고 나서 그 록들을 차례로 해제하는 경우에 재귀적 록들이 있을 때 일어날 수 있었던 데드락들에 대해 향상된 보호를 제공한다. 실행될 코드가 상기 재귀적 록들을 포함할 수 없다는 것이 알려지는 경우, 각 쓰레드가 대응한 록 파라미터를 갖는 이전의 접근법은, 성능을 향상시키는 것이 바람직할 수도 있다(그리고, 이전의 접근법은 그런데도 불구하고 종래기술과 비교하여 데드락의 발생을 크게 감소시킨다). 그렇지만, 재귀적 록들을 원하는 경우, 록들을 하나의 쓰레드에 대해 한 번에 제한하는 것은, 데드락에 대해 향상된 보호를 제공할 수도 있다. 이 경우에, 쓰레드마다 록 파라미터를 제공하는 것이 불필요한 것은, 하나의 쓰레드만이 록을 한 번에 유지할 수 있기 때문이다. 대신에, 어느 쓰레드가 현재 록을 유지하는지를 나타내는 록 소유자 파라미터 쓰레드와, 쓰레드들의 상기 서브세트간에 공유된 록 파라미터(예를 들면, 상기 록 소유 쓰레드가 유지한 쓰레드들의 수를 나타내는 록 카운트)는, 상기 프로그램 카운터의 선택이 그 밖의 쓰레드에 대해서 상기 록을 유지하는 쓰레드를 선택하는 것을 선호하여 데드락을 피하고 순방향 진행을 보장할 수 있도록, 어느 쓰레드가 록을 유지하는지를 추적하기에 충분하다. 상기 록 카운트는, 쓰레드에 의해 설정되었던 록들 전부를 다시 양보한 후 그 밖의 쓰레드가 록을 유지하게 하도록 보장하는데 유용하다.

[0032] 제2 예에서, 상기 적어도 하나의 록 파라미터는 쓰레드가 하나 이상의 공유 자원에 대한 배타적 액세스를 갖는 것을 나타내는 경우에, 상기 선택기는 상기 록 파라미터로 나타낸 상기 록 소유 쓰레드를 상기 선택된 쓰레드로서 선택하여, 그 록 소유 쓰레드는, 결국 상기 록을 해제하고 또 다른 쓰레드가 그 록을 차례로 얻을 수 있도록 진행할 수 있다. 한편, 쓰레드가 록을 유지하지 않는 경우에는, 상기 선택된 쓰레드의 선택은, 상기 제1 예와 같이 상기 쓰레드들의 상기 함수 호출 깊이와 쓰레드 프로그램 카운터들에 근거할 수도 있다.

[0033] 상술한 양쪽의 예에서, 상기 적어도 하나의 록 파라미터는, (상기 록 파라미터가 쓰레드들의 상기 서브세트간에 공유되거나 단일의 쓰레드에 따라 다른지를) 상기 제1 명령어에 응답하여 증분되어도 되고 상기 제2 명령어에 응답하여 감분되어도 된다. 일례에서, 상기 록 파라미터는, 상기 쓰레드가 공유 자원에 대한 배타적 액세스를 갖는 것을 나타내는 제1 상태와, 상기 쓰레드가 공유 자원에 대한 배타적 액세스를 갖지 않는 것을 나타내는 제2 상태의 2개의 상태만을 가질 수도 있다. 이러한 파라미터가 표현하기 쉬운 것은, 쓰레드마다 단일 비트만을 필요로 하기 때문이다. 이에 따라, 각 쓰레드의 상태를 표현하기 위해서 매우 작은 공간과 이 때문에 매우 작은 에너지를 필요로 한다. 그 밖의 실시예에서, 상기 록 파라미터는 쓰레드가 배타적 액세스를 갖는 공유 자원의 수를 나타낼 수도 있다. 달리 말하면, 상기 록 파라미터는 카운터로서 작동하여도 되고, 이 카운터는 쓰레드가 공유 자원에 대한 배타적 액세스를 얻을 때는 증분되고, 쓰레드가 공유 자원에 대한 배타적 액세스를 잃을 때는 감분된다. 이러한 시스템은, 쓰레드마다 우선권의 가장 좋은 지시자를 제공한다. 특히, 하나의 공유 자원에 대해서만 배타적 액세스를 갖는 쓰레드에 대해서 몇몇의 공유 자원에 대한 배타적 액세스를 갖는 쓰레드를 우선적으로 처리하는 것이 바람직할 수도 있다. 그렇지만, 이러한 추가의 정보의 저장에 보다 많은 공간을 필요로 하므로, 상기 시스템에서 실행될 쓰레드마다 보다 많은 에너지가 소비되어야 한다.

[0034] 상기 프로세서에서 실행하는 쓰레드들의 상기 서브세트의 하나 이상은, 상기 선택된 쓰레드의 대응한 쓰레드 파라미터들에 일치하는 하나 이상의 쓰레드 파라미터를 갖는 쓰레드들을 포함하여도 된다. 일례에서, 상기 하나 이상의 쓰레드 파라미터는, 상기 쓰레드 프로그램 카운트만을 포함하여도 된다. 따라서, 그 실행된 하나 이상의 쓰레드는, 상기 일반 프로그램 카운터에서 식별한 동일한 명령어를 식별하는 쓰레드 프로그램 카운터를 갖는 쓰레드들이어도 된다. 달리 말하면, 상기 데이터 처리장치는, 상기 일반 프로그램 카운터와 같은 프로그램 카운터

를 갖는 쓰레드 전부를 실행할 수도 있다. 또는, 상기 쓰레드 파라미터들도, 상기 쓰레드 프로그램 카운터와 아울러, 함수 호출 깊이 파라미터 또는 록 파라미터 중 한쪽이나 양쪽을 포함할 수도 있어, 명령어들은 상기 선택된 쓰레드의 쓰레드 파라미터들에 일치하는 쓰레드 파라미터의 조합을 갖는 쓰레드들에 대해 실행된다. 이렇게 하여, 단일 명령어는 다중 쓰레드에 대해 동시에 실행될 수도 있다.

[0035] 상기 데이터 처리장치의 프로세서에서 실행하는 명령어들은, 마이크로 연산이어도 된다. 일부의 시스템에서는, 복합 명령어(예를 들면, 로드/스토어 다중 명령어)는, 마이크로 연산으로 분리된 후, 그들 명령어는 상기 프로세서에 도달한다. 따라서, 본 출원에서의 "명령어"에의 언급은, 명령어들, 또는 하나의 명령어의 일부에 해당하는 마이크로 연산들을 말하는 것으로서 해석되어야 한다.

[0036] 상기 제1 명령어와 상기 제2 명령어는, 적어도 하나의 사전 정의된 명령어 세트에서의 NOOP(무연산) 명령어로서 부호화되어도 된다. 이러한 경우에, 상기 데이터 처리장치에서 사용한 상기 명령어 세트는, 상기 사전 정의된 명령어 세트의 향상된 버전이다. 달리 말하면, 상기 데이터 처리장치에서 사용한 상기 명령어 세트는, 사전 정의된 명령어 세트에 규정되어 있지 않는 명령어를 규정한다. 상기 제1 명령어와 상기 제2 명령어가 적어도 하나의 사전 정의된 명령어 세트에서의 NOOP 명령어로서 부호화되기 때문에, 상기 제1 명령어와 상기 제2 명령어는 사전 정의된 명령어 세트들에서 무시되어도 된다. 그래도 데이터 처리장치상에서 동작하도록 기록된 코드는, 상기 제1 명령어와 상기 제2 명령어가 단지 제2 데이터 처리장치에 영향을 미치지 않으므로 상기 사전 정의된 명령어 세트를 실행하는 상기 제2 데이터 처리장치상에서 정확히 기능한다. 따라서, 상기 데이터 처리장치를 위해 기록된 코드는, 레거시 시스템과 역호환 가능하다고 말할 수도 있다. 이것은, 상이한 시스템을 위한 다른 버전의 코드를 기록하는 것이 필요하지 않아, 코드 개발을 보다 효율적으로 한다는 것을 의미한다.

[0037] 상기 프로세서는, 상기 제1 명령어와 상기 제2 명령어 중 적어도 하나에 응답하여 또 다른 연산을 행하도록 구성되어도 된다. 예를 들면, 상기 또 다른 연산은, 상기 공유 자원에 대한 배타적 액세스를 얻기 위한 록킹 시퀀스의 일부인 연산과, 상기 공유 자원에 대한 배타적 액세스를 양보하기 위한 언록킹 시퀀스의 일부인 연산과, 상기 공유 자원을 액세스하기 위한 연산과, 쓰레드가 상기 공유 자원에 대한 배타적 액세스를 가졌는지를 판단하기 위한 연산과, 비교 및 교환 연산 중 하나 이상이어도 된다. 이에 따라 쓰레드가 다른 명령어와 겹치는 록 파라미터를 갱신하는 기능성은 실행될 필요가 있는 명령어의 수를 감소시킬 수 있다.

[0038] 상기 공유 자원은, 메모리에서의 데이터를 포함하여도 된다. 상기 공유 자원의 그 밖의 예는, 하드웨어 디바이스, 통신 채널, 또는 코드의 임계 영역이어도 된다.

[0039] 상기 프로세서는, 상기 공유 자원에 대한 배타적 액세스를 양보하기에 앞서 메모리 배리어 연산을 메모리에 발행하도록 구성되어도 된다. 상기 메모리 시스템은, 상기 프로세서로부터 트랜잭션들을 수신하는 순서까지 다른 순서로 트랜잭션들을 처리하여도 된다. 메모리 배리어 연산은, 상기 메모리가 트랜잭션들을 재정렬할 수 있는 정도를 상기 프로세서에 의해 제어하도록 발행되어도 된다. 상기 메모리 시스템에 의한 메모리 연산의 재정렬은, 메모리 배리어들에 걸쳐 금지된다. 즉, 상기 메모리가 연속적인 메모리 배리어 연산들간의 연산들을 임의의 순서로 마음대로 처리해도 좋지만, 메모리 배리어 연산이 수신될 경우, 상기 메모리는 상기 메모리 배리어 연산 후에 수신된 메모리 연산전의 상기 메모리 배리어 연산 전에 수신된 메모리 연산 전부를 처리해야 한다. 메모리 배리어가 발행되지 않으면, 자원에 대한 배타적 액세스가 양보된 후에 발행된 트랜잭션은 배타적 액세스가 여전히 유지된 동안에 상기 시스템 메모리에 의해 처리되어 데이터 값들이 부정확하게 될 수도 있을 위험이 있다. 상기 공유 자원에 대한 배타적 액세스를 양보하기 전에 메모리 배리어 연산들을 발행함으로써, 상기 시스템은, 한 번에 단일 쓰레드만으로 상기 공유 자원을 액세스하도록 보장할 수 있고, 상기 메모리에서 데이터의 일관성을 보장할 수 있다.

[0040] 도 1은 일 실시예에 따른 데이터 처리장치(100)를 도시한 것이다. 일반 프로그램 카운터(120)는, 선택기(110)에 의해 특별한 쓰레드와 관련된 상기 프로그램 카운터들(184) 중 하나로 설정된다. 그리고, 상기 일반 프로그램 카운터(120)의 값은, 상기 일반 프로그램 카운터(120)에서 참조한 명령어를 불러오는 페치(fetch)부(130)에 보내진다. 그 불러온 명령어는 그 명령어를 복호하는 디코드부(140)에 전달되고, 그 복호된 명령어는 발행부(150)에 보내진다. 상기 발행부(150)는, 상기 쓰레드들 중 하나 이상에 대한 상기 불러온 명령어를 실행하기 위해서 하나 이상의 신호를 상기 프로세서(160)에 발행한다. 상기 프로세서(160)는, 하나 이상의 쓰레드에 대한 명령어를 동시에 또는 실질적으로 동시에 실행 가능할 수도 있다. 일부의 실시예에서, 상기 프로세서(160)는, 다중 쓰레드에 대해 동일한 명령어를 병렬로 실행하도록, 각각의 쓰레드를 각각 처리하기 위한 병렬 기능부들을 가져도 된다. 그 밖의 시스템은, 단일의 기능부만을 가지므로 동일한 명령어는 다음 명령어로 움직이기 전에 쓰레드들의 상기 서브세트의 각각에 대해 순차로 처리된다. 그 밖의 시스템은 일부의 병렬부를 갖기도 하지만, 쓰레드들의

레드의 총수보다 적어서, 그 쓰레드의 총수미만의 여러 묶음으로 다수의 쓰레드에 대해 처리된다.

[0041] 또한, 상기 프로세서(160)는 메인 메모리뿐만 아니라 캐시도 구비할 수도 있는 메모리(170)에 액세스할 수도 있다. 또한, 상기 프로세서는, 데이터 값인 로딩되거나 저장되는 레지스터 파일(180)과 통신할 수도 있다. 본 실시예에서, 레지스터 파일(180)은, 쓰레드마다 다수의 상이한 하드웨어 유닛을 구비한다. 예를 들면, 도 1에 도시된 상기 레지스터 파일(180)은, 쓰레드마다, 일 세트의 레지스터(182), 프로그램 카운터(184), 함수 호출 깊이 카운터(186), 및 록 카운터(188)를 구비한다. 도 1에 도시된 것들과 상이한 하드웨어 구성이 가능할 수도 있다는 것을 알 것이다. 예를 들면, 단일 레지스터 뱅크(182)는, 하나보다 많은 쓰레드용 레지스터들로 이루어져도 된다. 게다가, 도시된 하드웨어 부품의 서브세트는 상기 레지스터 파일(180) 외부에 있어도 된다.

[0042] 본 명세서에서 설명된 실시예에서는, 단일 세트의 쓰레드들만을 생각하고, 이때 적어도 그 쓰레드들의 서브세트는 병렬로 실행된다. 그렇지만, 데이터 처리장치(100)는 이러한 구성에 한정되지 않는다는 것을 알 것이다. 예를 들면, 상기 데이터 처리장치(100)는, 쓰레드들의 다수의 그룹이나 서브세트에 관해 실행하여도 되고, 이때 쓰레드들의 각 그룹은 실질적으로 동시에 실행될 수 있다. 이에 따라, 상기 일반 프로그램 카운터(120)는, 일 세트의 일반 프로그램 카운터들을 구비하여도 되고, 그 세트에서의 각 일반 프로그램 카운터는 쓰레드들의 상이한 그룹에 관련되어 있다. 이 설명의 나머지는, 쓰레드들의 단일 그룹(서브세트)만을 생각하지만, 설명된 기술들은 다수의 쓰레드 그룹이 있을 때 각기 상이한 쓰레드 그룹에 적용될 수 있었다는 것을 알 것이다.

[0043] 도 2는 명령어들의 시퀀스를 실행하는 쓰레드들 T#0 내지 T#31의 단일 그룹을 도시한 것이다. 그 쓰레드들 각각이 그 자신의 레지스터들을 갖고 실행중 명령어들이 레지스터들을 참조하므로, 각 쓰레드는 동일한 명령어를 실행할 때 다르게 동작할 수도 있다. 명령어 0은, 레지스터 5와 레지스터 3에 보유된 데이터 값들에 의해 레지스터 0에 함께 가산 및 저장시킨다. 라인 1상의 명령어는, 이전의 가산이 0이었을 경우 그 쓰레드의 실행에 의해 라인 3에서 'label'에 점프시킨다. 이 경우에, 쓰레드 T#2(0+0)에서 행한 가산의 결과와 쓰레드 T#4(-1+1)에서 행한 가산의 결과 양쪽이 0이므로, 이들 2개의 쓰레드에 대한 제어의 흐름이 'label'에 점프한다. 그 밖의 쓰레드들은 점프하지 않지만 대신에 라인 2에서의 명령어로 계속된다.

[0044] 따라서, 쓰레드 T#2와 T#4용 프로그램 카운터는 3이고 나머지 쓰레드용 프로그램 카운터는 2이다. 그 쓰레드들의 수렴을 권장하기 위해서, 상기 데이터 처리장치(100)용 상기 일반 프로그램 카운터(120)는, 쓰레드 전체 중에서 최저의 쓰레드 프로그램 카운터(184)(즉, 2)로 설정될 것이다. 이렇게 상기 일반 프로그램 카운터(120)의 값을 선택함으로써, 덜 멀리 전진된 쓰레드들은 진행하므로 그 밖의 쓰레드들을 추월하게 할 수 있음으로써, 쓰레드들이 수렴하게 된다. 이것은, 상기 시스템의 병렬성이 향상되기 때문에, 즉 보다 많은 쓰레드들이 병렬로 실행하는 상태가 되는 것이 바람직한 상태다. 이에 따라, 상기 일반 프로그램 카운터(120)는, 임의의 쓰레드와 관련된 최저의 프로그램 카운터 값인 2값으로 설정된다. 쓰레드 T#2와 T#4의 쓰레드 프로그램 카운터가 2가 아니므로, 라인 2에서의 명령어는 쓰레드 T#2와 T#4에 대해 실행되지 않는다. 나머지 쓰레드들은, 레지스터 6에, 레지스터 0과 레지스터 4에서의 데이터 값에 의해 함께 곱하여 저장하는 라인 2에서의 명령어를 행한다. 그 곱셈을 행하였으면, 그 명령어를 실행한 각 쓰레드용 쓰레드 프로그램 카운터는, 3으로 전진된다. 이에 따라, 쓰레드 전부는 동일한 프로그램 카운터를 갖고 수렴이 이루어진다. 라인 3에서의 "명령어"는 단지 라벨일 뿐이어서 실행은 라인 4에 진행된다. 이 명령어는, 레지스터 9에 저장된 메모리 어드레스를 메인 메모리로부터 액세스시키고 그 어드레스에서의 데이터 값을 레지스터 8에 저장시킨다. 도 2에 도시된 것처럼, 각 쓰레드가 다른 메모리 어드레스 값을 레지스터 9에 저장(예를 들면, 쓰레드 T#0이 메모리 어드레스 100을 저장하는 한편, 쓰레드 T#5가 메모리 어드레스 200을 저장)하므로, 각 쓰레드는 다른 메모리 어드레스에 액세스하기 때문에, 다른 값을 그들 각각의 레지스터 8에 저장할 것이다. 동일한 캐시 라인이나 메모리의 동일한 페이지를 단일 메모리 어드레스의 대상으로 하는 상기 메모리 액세스들을 통합하여, 전력과 시간을 절약하는 것이 가능하다.

[0045] 도 3은 2개의 쓰레드에 관해 실행중인 코드를 도시한 것이고, 이때의 코드는 공유 자원을 포함하고, 그 공유 자원에 대한 액세스가 록을 거쳐 제어된다. 여기서, 상기 공유 자원은, 라인 7과 9 사이의 코드의 임계 영역에 의해 액세스된다. 상기 공유 자원은, 상기 코드의 임계 영역에서 필요로 하고 한 번에 하나의 쓰레드가 상기 자원에 대한 배타적 액세스를 필요로 하는 어떠한 것이어도 된다. 예를 들면, 상기 자원은, 공유 데이터 구조이거나 하드웨어 유닛 또는 하드웨어 디바이스의 이용이어도 된다. 또는, 상기 코드의 임계 영역 자체는 공유 자원으로서 간주되어도 된다. 본 예에서, 상기 선택기(110)는 상기 일반 프로그램 카운터(120)를, 최고의 함수 호출 깊이 카운터(186)를 갖는 쓰레드의 상기 쓰레드 프로그램 카운터(184)로 설정하도록 구성된다. 다중 쓰레드의 최고의 함수 호출 깊이 카운터(186)를 갖는 경우에, 상기 일반 프로그램 카운터(120)는 최고의 함수 호출 깊이 카운터(186)를 갖는 상기 쓰레드들 중에서 최저의 프로그램 카운터(184)로 설정된다. 이전에 설명된 것처럼, 통상의 상황하에, 쓰레드들의 수렴을 일어나게 하기 위해서 프로그램 카운터 값이 상위인 쓰레드들보다 우선적으로,

최저의 프로그램 카운터를 갖는 쓰레드들에 의해 실행하는 것이 바람직하다. 그렇지만, 도시될 것처럼, 이러한 프로그램 카운터 선택의 메커니즘은, 코드의 임계 영역이 있을 때 데드락에 의해 일어나게 한다.

[0046] 도 3의 몇몇의 명령어는 레지스터 x19에서의 어드레스를 참조한다. 이 어드레스는, 쓰레드에 의한 배타적 액세스를 위해 상기 공유 자원이 록 되었는지 아닌지를 나타내는 데이터 값에 해당할 수도 있다. 본 예에서, 레지스터 x19에서 참조한 어드레스에서의 값이 0일 때, 이것은 라인 7-9에서의 코드의 임계 영역에서 사용한 자원에 대해 록이 설정되어 있지 않았다는 것을 나타내고, 이 어드레스에서의 값이 1일 때는, 록이 설정되어 있었다. 그렇지만, 자원에 대해 배타적 액세스를 허가하였는지를 표현하는 그 밖의 방식들이 있다는 것을 알 것이다. 이 때, 상기 쓰레드들이 이 코드 예를 실행할 때, 라인 2와 5에서의 명령어들을 실행하는 상이한 쓰레드들이 상이한 메모리 어드레스를 포함하는 그들 각각의 레지스터 세트(182)에서의 다른 버전의 레지스터 x19에 액세스할 수도 있다. 따라서, 쓰레드들이 액세스하려고 하는 공유 자원은 쓰레드간에 다를 수도 있다. 한편, 레지스터 x19가 동일한 어드레스를 포함하는 다중 쓰레드가 있으면, 아래에 설명된 것처럼 충돌 가능성이 있다.

[0047] 먼저, 쓰레드 전부가 정확히 같은 방식으로 실행할 것이다. 라인 0은 라벨을 포함하므로, 그 실행에 영향을 미치지 않는다. 라인 1은 데이터 값 1을 레지스터 w1에 저장시킨다. 라인 2에서의 명령어는 로드 배타적 명령어로서 알려져 있다. 이것은 데이터 값을 액세스시키고 모니터를 설정시킨다. 이 모니터는, 상기 데이터 값이 변경되는지를 검출한다. 따라서, 본 예에서, 레지스터 x19를 참조한 메모리 어드레스에 저장된 데이터 값은, 액세스되고, 레지스터 w0에 저장되고, 그 메모리 어드레스를 위해 모니터가 설정된다. 라인 3에서, 레지스터 w0에 저장된 데이터 값이, 수 0에 비교된다. 달리 말하면, 그 모니터를 설정하는 것 외에는, 라인 2와 3은 레지스터 x19를 참조한 메모리 어드레스에 저장된 데이터 값들이 0인지 아닌지를 일괄하여 판정한다. 라인 4에서의 명령어는, 그 데이터 값들이 같지 않은 경우 라인 0에서의 라벨 `retry_lock`으로 실행을 점프시킨다. 달리 말하면, 레지스터 w0에 저장된 데이터 값이 0이 아닌 경우(즉, 또 다른 쓰레드가 이미 상기 공유 자원에 관한 록을 갖는 경우), 제어의 흐름이 라벨 `retry_lock`으로 복귀하고, 그렇지 않은 경우는, 그 제어의 흐름은 라인 5로 계속된다. 따라서, 라인 2 내지 4에서의 명령어들은, 또 다른 쓰레드가, 라인 2에서의 로드 명령어를 실행하였을 때까지는 이미 상기 록이 되었는지를 검사한다.

[0048] 한편, 라인 5 내지 6에서의 명령어들은, 라인 2에서의 상기 배타적 로드 명령어를 실행한 후이지만, 상기 코드의 임계 영역을 시작하기 전에, 또 다른 쓰레드가 일부의 포인트에서 상기 록을 얻었는지를 검사한다. 이는, 다중 쓰레드가 상기 록을 동시에 얻을 수 없도록 보장한다. 라인 5는 스토어 배타적 명령어로서, 이 명령어는, 감시중인 메모리 어드레스가 변경되지 않고 있다는 것을 이전에 설정된 모니터가 나타내는 경우에만, 데이터 값을 저장시킨다. 본 예에서, 그 명령어는, 라인 2의 상기 로드 배타적 명령어에 의해 상기 모니터를 확립한 후에 레지스터 x19에서 참조한 메모리 어드레스에 저장된 데이터 값이 변경되지 않았을 경우에만 레지스터 x19에서 참조한 메모리 위치에 레지스터 w1에 저장된 데이터 값(즉, 1)을 저장시킨다. 그 후, 상기 스토어가 성공적이거나 실패일 때의 결과는, 레지스터 w2에 저장된다. 특히, 상기 스토어가 성공적이었을 경우, 그 값 0은 w2에 저장된다. 그렇지 않은 경우는, 상기 스토어가 성공하지 못했을 경우, 논제로 값은 w2에 저장된다. 물론, 당업자라면, 성공 또는 실패를 나타내는데 사용된 특정한 수가 중요하지 않고 이들 수는 쉽게 반전될 수 있었다는 것을 알 것이다. 또 다른 쓰레드가 레지스터 x19에서 참조한 어드레스에 저장된 데이터 값을 변경하지 않고 있을 경우 상기 스토어 연산이 실시되지만 하므로, 레지스터 x19의 값이 쓰레드 전부에 대해 같은 경우, 꼭 하나의 쓰레드는 데이터 값을 변경할 것이다. 상이한 쓰레드가 상이한 록된 자원을 필요로 하면, 상이한 어드레스는 상이한 쓰레드를 위해 레지스터 x19에 배치되어, 하나보다 많은 쓰레드가 라인 5에서의 스토어 명령어를 성공적으로 실행하는 것이 가능하다. 이 때문에, 쓰레드마다 상기 스토어 배타적 명령어의 결과는, 그 쓰레드가 상기 록을 취득하였는가 아닌가를 나타낸다. 라인 6에서는, 레지스터 w2에 저장된 데이터 값이 0인가 아닌가(즉, 상기 스토어 배타적 명령어가 성공적이었는가 아닌가)를 판정하기 위해 비교한다. 제어 흐름은, w2에 저장된 데이터 값이 0이 아닐 경우 라인 0에서 라벨 `retry_lock`으로 복귀한다. 그렇지 않은 경우, 흐름은 라인 7상에서 계속된다.

[0049] 라인 7 내지 9(도 3에는 미도시됨)는 상기 임계 영역을 표현한다. 즉, 이전의 록킹 시퀀스에 의해, 그 임계 영역은, 공유 자원 당 단일의 쓰레드에 의해 한 번에 실행될 수 있을 뿐이다(다중 쓰레드는 상이한 자원을 사용하고 있는 경우 그 임계 영역을 실행하여도 된다). 쓰레드는, 동일한 자원을 필요로 하는 또 다른 쓰레드에 들어가기 전에 상기 임계 영역에서 실행중 코드를 끝내야 한다. 그 임계 영역은, 데이터 일관성을 유지하기 위해 배타적 액세스를 필요로 하는 공유 위치나 공유 데이터 구조에 액세스하는 로드/스토어 명령어와 같은 공유 자원을 이용하는 임의의 명령어들이나, 하나의 쓰레드에서만 한 번에 사용할 수 있는 하드웨어 디바이스를 이용하는 명령어들을 포함할 수도 있다.

[0050] 라인 10에서 시작하는 언록킹 시퀀스가 도시되어 있다. 이 코드는, 상기 임계 영역이 실행된 후에 실행된다. 라

인 10에서는, 데이터 값 0을 레지스터 w1에 저장한다. 라인 11에서는, 이 쓰레드가 더 이상 상기 록을 갖지 않는다는 것을 나타내기 위해, 레지스터 w1에 저장된 데이터 값(즉, 0)을 레지스터 x19에서 참조한 메모리 위치에 저장한다. 이것은, 라인 2 내지 4에서 명령어들에 도달하는 또 다른 쓰레드가 상기 록을 성공적으로 취득할 수 있다는 것을 의미한다.

[0051]

이전에 언급된 것처럼, 록을 임계 영역에 제공하기 위한 본 코드는, 다중 쓰레드가 동일한 록을 얻으려고 하고 있는 중일 경우 SIMT 데이터 처리장치에서 데드락을 일어나게 할 수도 있다(즉, 동일한 메모리 위치는 상이한 쓰레드를 위한 레지스터 뱅크들(182)의 레지스터 x19의 각각의 버전으로 나타낸다). 예를 들면, 도 3에 도시된 것처럼, 쓰레드 T#0은 상기 록에 대한 액세스를 취득하므로 상기 코드의 라인 7로 계속되는 쓰레드이어도 된다. 그렇지만, 쓰레드 T#1(및 동일한 자원을 대상으로 하는 그 밖의 쓰레드)은 상기 록을 취득하지 못한다. 이에 따라, 상기 코드의 제1 실행에 있어서, 쓰레드 T#0이외의 쓰레드들은, 상기 록을 얻는데 실패하여 그들의 제어 흐름을 라인 0으로 복귀시키기 전에 그 코드의 라인 6까지 실행할 것이다. 이후의 시도를 위해, 쓰레드 T#0이외의 쓰레드들은, 쓰레드 T#0이 레지스터 x19에서 참조한 어드레스에 값 #1을 저장하였을 것이기 때문에 라인 4까지 실행할 수도 있다. 상기 쓰레드들의 전부는, 동일한 함수 호출 깊이 카운터 값이 1이다. 이 때문에, 상기 선택기(110)는, 상기 일반 프로그램 카운터(120)를, 상기 각 쓰레드와 관련된 최저의 쓰레드 프로그램 카운터(184)로 설정한다. 쓰레드 T#0이외의 모든 쓰레드들의 프로그램 카운터 값은 0이어서, 상기 일반 프로그램 카운터 값(120)은 0으로 설정된다. 이에 따라, 쓰레드 T#0이외의 쓰레드들은 실행 가능하게 된다. 그렇지만, 그 쓰레드들은, 상기 록이 T#0에 의해 유지되기 때문에 상기 임계 영역에 대해 상기 록을 취득하지 못한다. 이 때문에, 라인 4에서는, 그 쓰레드들의 제어 흐름이 라인 0으로 지나간다. 또, 상기 선택기(110)에서 프로그램 카운터 값을 선택할 때, 쓰레드 T#0이외의 쓰레드들은, 쓰레드 T#0(7)보다 낮은 프로그램 카운터 값(0)을 갖는다. 이에 따라, 상기 일반 프로그램 카운터 값(120)은, 0으로 설정되고, 쓰레드 T#0이외의 쓰레드들은 실행 가능하게 된다. 이 처리는, 아주 오랜 시간 계속될 가능성이 있다. 쓰레드 T#0만이 상기 임계 영역에 대한 액세스를 얻을 수 있고, 쓰레드 T#0은 그 임계 영역이 실행된 후에 그 임계 영역에 대한 액세스만을 얻을 수 있다. 그렇지만, 쓰레드 T#0은, 그것의 프로그램 카운터 값이 그 밖의 쓰레드보다 높기 때문에 임계 영역을 실행할 수 없어, 상기 일반 프로그램 카운터(120)는, 결코 쓰레드 T#0과 관련된 상기 쓰레드 프로그램 카운터(184)로 설정되지 않는다. 이 때문에, 본 예에서는 데드락이 일어난다. 쓰레드를 진행할 수 없으므로 상기 시스템은 정지한다.

[0052]

도 4는 이 데드락 과제에 대한 해결책을 도기한 것이다. 본 실시예에서, 록 카운터(188)는 쓰레드마다 설치되고, 상기 선택기(110)는 상기 함수 호출 깊이 및 쓰레드 프로그램 카운터와 아울러, 상기 록 카운터에 근거하여 (쓰레드 프로그램 카운터가 상기 일반 프로그램 카운터로 설정된) 상기 선택된 쓰레드를 선택하도록 구성된다. 상기 일반 프로그램 카운터(120)는, 최고의 록 카운터(188)를 갖는 쓰레드와 관련된 상기 쓰레드 프로그램 카운터(184)로 설정된다. 다중 쓰레드의 록 카운터(188)가 최고인 경우, 최고의 함수 호출 깊이 카운터(186)를 갖는 쓰레드들은 최고의 록 카운터(188)를 갖는 쓰레드들 중에서 생각된다. 다중 쓰레드가 최고의 록 카운터(188)를 갖고 최고의 함수 호출 깊이 카운터(186)를 갖는 경우, 그 쓰레드들 중에서 최저의 프로그램 카운터 값을 갖는 쓰레드가 선택된다. 그리고, 상기 일반 프로그램 카운터는 그 선택된 쓰레드들의 쓰레드 프로그램 카운터에 근거하여 갱신되고 나서, 상기 일반 프로그램 카운터에서 나타낸 명령어는, 일치하는 쓰레드 파라미터들(적어도 일치하는 쓰레드 프로그램 카운터, 선택사항으로 일치하는 함수 호출 깊이와 록 파라미터)을 갖는 임의의 쓰레드에 대해 실행된다. 그 밖의 실시예는, 특정한 쓰레드 프로그램 카운터나, 쓰레드 프로그램 카운터, 함수 호출 파라미터 및 록 파라미터의 특정한 조합이 (쓰레드를 선택하는 대신에) 직접 선택되는 경우에 다른 메커니즘을 적용하여도 된다는 것을 알 것이다.

[0053]

도 4의 코드는, 라인 6과 12 각각에서 2개의 새로운 명령어의 가산으로 도 3의 코드와 같다. 이들 명령어는, 쓰레드와 관련된 상기 록 카운터(188)의 조건적 증분과 감분의 원인이 된다. 특히, 라인 6의 명령어 'cond_inc_lock_count w2'는, 쓰레드와 관련된 상기 록 카운터(188)를, w2의 값이 제로일 경우에 증분시킨다. 도 3을 참조하여 설명된 것처럼, w2의 값은 0 또는 1로 설정되고, 이때 (그 쓰레드가 상기 록을 취득할 때 성공한) 상기 값 0이 STXR 명령어의 성공을 표현하고 (그 쓰레드가 상기 록의 취득을 실패한) 상기 값 1이 STXR 명령어의 실패를 표현한다. 이에 따라, 라인 6의 명령어는, 그 쓰레드가 라인 5의 선행 명령어로 상기 록을 취득하도록 관리되었을 경우 특별한 쓰레드에 대한 록 카운터(188)를 증분시킨다. 라인 12에서 명령어 'dec_lock_count'는, 특별한 쓰레드에 대한 록 카운터(188)를 감분시킨다. 이 명령어는 무조건적이다. 그것은, 라인 8 내지 12에서의 코드가 한 번에 단일 쓰레드에 의해서만 실행되기 때문이다 - 이 쓰레드는 구체적으로는 현재 상기 록을 유지하므로 상기 임계 영역에 액세스한다. 그 결과, 어느 쓰레드의 록 카운터가 감분되어야 하

는지에 관하여 애매모호함이 없다.

[0054] 상기 코드가 실행할 때, 라인 6에서의 명령어 때문에, 상기 록(T#0)을 취득하는 스레드는 그것의 록 카운터를 증분시켰다. 반대로, 그 밖의 스레드는, 그들의 레지스터 w2의 값이 논제로이기(상기 록의 취득을 실패하기) 때문에 그들의 록 카운터들을 증분시키지 않는다. 라인 7에서, 스레드 T#0은 라인 8상에서 계속되고, 나머지 스레드들은 라인 0에서 라벨 'retry_lock'에 복귀할 것이다. 이전에 설명된 것처럼, 본 실시예에서, 상기 선택기(110)는, 상기 일반 프로그램 카운터(120)를, 최고의 록 카운터를 갖는 스레드의 프로그램 카운터로 설정하도록 구성된다. 이에 따라, 스레드 T#0의 프로그램 카운터가 8일 때, 그 스레드는, 비록 그것의 프로그램 카운터(182)가 그 밖의 스레드보다 높을지라도 그것의 상위 록 카운터(188)에 의해 실행하기를 계속할 수 있다. 이 때문에, 스레드 T#0은, 실행을 계속할 수 있고, 결국 상기 록을 해제하는 라인 10 내지 11의 언록킹 시퀀스의 코드를 실행한다. 그리고, 스레드 T#0은 그 록 카운터를 감분시키는 라인 12로 계속된다. 이 시점에서, 상기 각 스레드의 록 카운터는 0이고, 상기 각 스레드의 함수 호출 깊이는 1이므로, 상기 일반 프로그램 카운터(120)는 모든 스레드들 중에서 최저의 스레드 프로그램 카운터(184)로 설정된다. 이에 따라, 스레드 T#0이외의 스레드들은, 동작이 허용되고 라인 0과 7 사이의 코드를 실행한다. 이 실행 동안에, 상기 스레드들 중 하나는, 이전에 설명된 것처럼 상기 록을 취득하고 그 록 카운터(188)를 증분시킬 것이다. 이 시퀀스는, 모든 스레드들이 코드의 임계 영역을 지나갈 때까지 반복된다. 이에 따라, 코드의 임계 영역이 있음에도 불구하고 데드락 상황을 피한다.

[0055] 도 4의 본 예에서, 상기 cond_inc_lock_count 명령어와 dec_lock_count 명령어는 특별한 스레드와 관련된 록 카운터(188)에만 영향을 미친다. 그러나, 이들 명령어는 추가의 연산을 행하여도 된다는 것을 알 것이다. 예를 들면, 상기 조건적 증분 명령어는, 라인 5의 상기 배타적 스토어 명령어나 라인 7의 비교 분기 명령어와 조합될 수 있었다. 이러한 경우에, 상기 조건적 증분 명령어는, 비교 및 분기 명령어에 관한 연산 전이나 병렬로 또한, 상기 배타적 스토어 명령어에 관한 연산 후에 실시된다. 마찬가지로, 상기 감분 록 카운터는, 그 밖의 연산과 조합되어도 된다. 예를 들면, 상기 감분 록 카운터 명령어는, 수치 0을 레지스터 x19에서 가리킨 어드레스에 저장하는 명령어와 조합되어도 된다.

[0056] 또한, 상기 록킹 시퀀스를 구현하는 방식이 많이 있다. 도 4의 예에서는, 라인 2의 로드 배타적 명령어와 라인 5의 스토어 배타적 명령어의 조합을 이용한다. 상기 록킹 메커니즘을 구현하는 다른 방식은, 비교 및 교환 명령어(CMPXCHG)를 이용할 수도 있다. 상기 비교 및 교환 명령어는, 원자 프리미티브다. 달리 말하면, 그 명령어는, 실행 중간에 중단될 수 없지만 대신에 일단 시작하였으면 완료할 때까지 동작한다. 비교 및 교환 명령어는 3개의 파라미터를 취한다. 그 파라미터들 중 하나는 메모리내의 위치다. 또 다른 파라미터는 비교 값이고, 추가로 또 다른 파라미터는 스토어 값이다. 상기 비교 및 교환 명령어는, 메모리내의 위치에서의 데이터 값이 비교 값인가 아닌가를 시험하고, 비교 값일 경우, 그 스토어 값을 그 메모리 위치에 기록하고 그 연산이 성공적이었다는 것을 나타내는 결과를 복귀시킨다. 상기 메모리 위치에서의 데이터 값이 상기 비교 값이 아닌 경우, 그 메모리 위치(그 메모리 위치는 자신의 원래 값을 유지하고 있다)에 아무것도 기록되지 않고, 그 대신에 그 결과는 그 연산이 성공적이지 않았다는 것을 나타낸다. 또, 상기 비교 및 교환 명령어는, 그 밖의 연산과 조합되어도 되고, 상기 cond_inc_lock_count 명령어와 조합되어도 된다.

[0057] 메모리 배리어 연산은, 상기 dec_lock_count 명령어 전과 상기 코드에서의 임계 영역 후에 행해져도 된다. 데이터 처리장치에 있어서, 메모리 액세스 명령어들은 효율성을 위해 기록되어도 된다. 그렇지만, 이러한 재정렬은, 메모리 배리어에 걸쳐 일어나지 않을 수도 있다. 따라서, 메모리 배리어는, 코드의 임계 영역의 일부인 메모리 액세스 연산들이 상기 임계 영역을 완료한 후 발행되었던 메모리 액세스 연산들 앞에 처리되도록 보장하는데 도움이 될 수 있다.

[0058] 상기 cond_inc_lock_count 명령어와 dec_lock_count 명령어는 각각 소정의 명령어 세트에서, 상기 cond_inc_lock_count 명령어와 dec_lock_count 명령어의 인코딩이 NOOP 명령어에 해당하도록 부호화될 수도 있다. 예를 들면, 상기 데이터 처리장치(100)에서 사용한 명령어 세트는, 이들 2개의 명령어를 NOOP 명령어로서 부호화하는 향상된 버전의 명령어 세트이어도 된다. 따라서, 데이터 처리장치(100)이외의 데이터 처리장치에는, 2개의 명령어가 아무 효과도 없을 수도 있다. 이에 따라, 데이터 처리장치(100)에 사용하도록 설계된 코드는, 그 밖의 데이터 처리장치들과 역호환 가능할 수도 있고, SIMT를 이용하지 않는 데이터 처리장치와 역호환 가능할 수도 있다.

[0059] 도 5는 일반 프로그램 카운터(120)로서 사용되는 스레드 프로그램 카운터를 갖는 상기 선택된 스레드를 선택하는 방법을 도시한 것이다. 이 방법은 상기 제1 세트와 상기 제2 세트 양쪽을 빈 세트로 설정하는 단계 s200에서

시작한다. 단계 s205에서는, 모든 쓰레드들 중 다음 쓰레드를 선택한다. 단계 s210, s215, s220과 함께 이 단계는, 총괄하여, 모든 쓰레드들을 통해 반복하는 루프를 형성한다. 단계 s210에서는, 단계 s205에서 선택한 쓰레드의 록 파라미터(즉, 록 카운터 188)가 모든 쓰레드들의 최고의 록 파라미터인가 아닌가를 판정한다. 이 쓰레드의 록 파라미터가 모든 쓰레드들의 최고의 록 파라미터인 경우, 단계 s215의 흐름으로 진행되고, 그렇지 않은 경우는 단계 s220의 흐름으로 계속된다. 단계 s210에서는, 데이터 처리장치가 모든 쓰레드들의 최고의 록 파라미터를 알고 있다고 가정한다. 이 정보가 쉽게 이용 가능하지 않을 경우, 그것은, 먼저 모든 쓰레드들을 통해 반복함으로써, 또는 상기 이전의 최고의 록 파라미터보다 높은 록 파라미터에 접할 때 이전의 쓰레드들에서 본 최고의 록 파라미터의 런닝 카운트를 유지하고 상기 제1 세트에 이미 추가된 쓰레드들을 폐기함으로써, 결정될 수 있다. 단계 s215에서는, 상기 제1 세트에 상기 선택된 쓰레드를 추가하고 나서, 단계 s220의 흐름으로 진행된다. 단계 s220에서는, 반복할 쓰레드가 보다 많이 있는가 아닌가를 판정한다. 쓰레드가 보다 많이 있는 경우에는, 다음 쓰레드를 선택하는 단계 s205의 흐름으로 되돌아간다. 쓰레드가 보다 많이 없는 경우에는, 단계 s225의 흐름으로 계속된다. 따라서, 단계 s220의 끝까지, 상기 쓰레드들 중 어느 하나의 상위 록 파라미터와 같은 록 파라미터를 갖는 쓰레드들을 포함하는 쓰레드들의 제1 세트를 판정한다. 단계 s225는, 상기 제1 세트에 정확히 하나의 엔트리가 있는가 아닌가를 판정한다. 그 제1 세트에 정확히 하나의 엔트리가 있을 경우, 단계 s230의 흐름으로 계속되어 상기 제1 세트에서의 상기 엔트리를 상기 선택된 쓰레드로서 복귀시킨다. 즉, 상기 일반 프로그램 카운터(120)는, 상기 제1 세트에서 단지 그 쓰레드와 관련된 쓰레드 프로그램 카운터(184)로 설정된다. 단계 s225에서는 상기 제1 세트에 정확히 하나의 엔트리가 없는 경우, 단계 s235의 흐름으로 계속된다. 총괄하여, 단계 s205 내지 s230은 쓰레드 프로그램 카운터 선택의 결정을 쓰레드들의 전부의 록 카운터 파라미터에 근거하려고 시도한다. 그 쓰레드들 중 하나가 나머지 쓰레드의 전부보다 높은 록 카운터 파라미터를 갖는 경우, 그 쓰레드는 선택된 쓰레드이고 상기 일반 프로그램 카운터(120)는 상위 록 카운터 파라미터(188)를 갖는 쓰레드에 대응한 쓰레드 프로그램 카운터(184)로 설정된다. 그렇지 않은 경우, 상기 쓰레드들의 또 다른 내로우잉(narrowing)은 아래에 설명된 것처럼 실시된다.

[0060] 단계 s235에서는, 상기 제1 세트로부터 다음 쓰레드를 선택한다. 단계 s235 내지 s250은, 총괄하여, 상기 제1 세트에서의 모든 쓰레드들을 통해 반복하는 루프를 형성한다. 단계 s240에서는, 상기 선택된 쓰레드의 함수 호출 깊이가 상기 제1 세트에서의 모든 쓰레드들의 최고의 함수 호출 깊이와 같은지를 판정한다. 또, 상기 제1 세트에서의 모든 쓰레드들의 최고의 함수 호출 깊이를 판정하는 것이 가능하다고 가정한다. 이 정보를 판정할 수도 있는 방식의 하나는, 먼저, 상기 제1 세트에서의 모든 쓰레드들을 통해 반복하거나, 상술한 것처럼 런닝 카운트 값을 유지하여서 반복하는 방식이 있다. 단계 s240에서, 상기 선택된 쓰레드의 함수 호출 깊이가 상기 제1 세트에서의 모든 쓰레드들의 최고의 함수 호출 깊이와 같을 경우, 단계 s245의 흐름으로 계속된다. 그렇지 않은 경우, 단계 s250의 흐름으로 계속된다. 단계 s245에서는, 상기 선택된 쓰레드를 상기 제2 세트에 추가하고 단계 s250의 흐름으로 계속된다. 단계 s250에서는, 반복될 상기 제1 세트에 보다 많은 쓰레드가 있는가 아닌가를 판정한다. 상기 제1 세트에 보다 많은 쓰레드가 있을 경우, 단계 s235의 흐름으로 되돌아가 상기 루프가 계속된다. 그렇지 않은 경우, 단계 s255의 흐름으로 계속되어 상기 제2 세트에 정확히 하나의 엔트리가 있는가 아닌가를 판정한다. 단계 s255에서 상기 제2 세트에 정확히 하나의 엔트리가 있을 경우, 단계 s260의 흐름으로 계속된다. 단계 s260에서는, 상기 선택된 쓰레드로서 상기 제2 세트에서의 단일의 엔트리를 복귀시킨다. 달리 말하면, 상기 일반 프로그램 카운터(120)는, 상기 제2 세트에서의 그 단일 쓰레드와 관련된 쓰레드 프로그램 카운터(184)로 설정된다. 이러한 상황은, 예를 들면, 하나보다 많은 쓰레드가 최고의 록 카운터 파라미터(188)를 공유하는 경우이지만 그 쓰레드들 중 하나의 쓰레드만이 최고의 함수 호출 깊이 카운터(186)를 갖는 경우에, 일어날 수도 있다. 이에 따라, 이러한 쓰레드에 대하여, 상기 일반 프로그램 카운터(120)를 그 쓰레드와 관련된 쓰레드 프로그램 카운터(184)로 설정함으로써, 실행시킨다. 총괄하여, 단계 s235 내지 s260은 상기 일반 프로그램 카운터(120)를 결정하는데 사용되는 제2 기준에 관한 것이다.

[0061] 상기 제2 세트에서 정확히 하나의 엔트리가 없을 경우, 단계 s265의 흐름으로 진행되어 최저의 프로그램 카운터(184)를 갖는 상기 제2 세트에서의 엔트리가 상기 선택된 쓰레드로서 복귀된다. 달리 말하면, 상기 일반 프로그램 카운터(120)는, 상기 제2 세트에서 모든 쓰레드들 중에서 상기 쓰레드 프로그램 카운터들(184) 중 최저의 것으로 설정된다. 이 때문에, 단계 s265는 제3 기준을 제공하여 상기 일반 프로그램 카운터(120)를 결정한다.

[0062] 도 6은 각 사이클에서 실행될 쓰레드들이 어떻게 선택되는지를 도시한 것이다. 단계 s400에서는, 복수의 쓰레드 중 하나(또는 마찬가지로, 상기 쓰레드들에 대응한 쓰레드 프로그램 카운터들 중 하나는, 상기 록 파라미터들(록 카운터들)(188), 함수 호출 깊이 카운터들(186) 및 쓰레드 프로그램 카운터들(184)에 근거하여 선택된다. 본 처리의 예는 도 5에 대해 도시된 예다. 단계 s410에서는, 상기 일반 프로그램 카운터(120)를 상기 선택된 쓰레드의 쓰레드 프로그램 카운터로 설정된다. 달리 말하면, 상기 일반 프로그램 카운터(120)는, 단계 s400에서

선택된 쓰레드에 대응한 상기 쓰레드 프로그램 카운터(184)에 일치하도록 설정된다. 끝으로, 단계 s420에서는, 상기 일반 프로그램 카운터(120)에서 식별한 명령어를, 상기 선택된 쓰레드의 쓰레드 파라미터들에 일치하는 관련된 쓰레드 파라미터들을 갖는 모든 쓰레드들에 대해 실행한다(상기 쓰레드 파라미터들은, 쓰레드 프로그램 카운터(184)만을 포함하여도 되거나, 상기 록 카운터(188)와 함수 호출 깊이(186)의 한쪽 또는 양쪽과 상기 쓰레드 프로그램 카운터의 조합을 포함하여도 된다). 이 처리는, 각 쓰레드와 관련된 쓰레드 프로그램 카운터(184)를 변경시킴에 따라, 그 처리는 단계 s400에서 시작하여 상이한 일반 프로그램 카운터(120)를 결정시키는 것이 반복되어도 된다.

[0063] 도 4에 대해 설명된 예에서, 상기 프로세서는 뒤틀림에 있어서 모든 쓰레드들에 대한 록들을 긍정적으로 취득하려고 하고 나서, 어느 쓰레드들이 록들을 취득할 수 있었는지에 근거하여 실행 우선순위를 정리한다.

[0064] 제2 실시예는 도 7 내지 도 9에 대해 설명한다. 본 실시예에서, 그룹에서 하나의 쓰레드만이 한 번에 록을 유지시킨다. 또한, 반면에, 도 4에 상기 프로세서가 STXR 명령어를 포함한 록킹 시퀀스를 사용하여 모든 쓰레드들에 대한 록들을 긍정적으로 취득하려고 하고 나서, 상기 쓰레드가 록킹 특권을 얻었던 신호까지 록 카운트를 성공적으로 증분하는 경우에만, 도 7 내지 도 9에서 상기 순서는, 먼저 명령어를 실행하여 상기 록킹 특권을 요구함에 따라 그 록 파라미터를 갱신하도록 뒤집힌다. 이 시점에서, 상기 프로세서는, 하나의 쓰레드만이 록킹 특권을 요구하는 명령어를 성공적으로 실행할 수 있도록 보장한다(그 밖의 쓰레드의 경우는, 이 명령어의 실행이 실패하므로 그 쓰레드에 대한 쓰레드 프로그램 카운터가 증분되지 않는다). 상기 록킹 특권을 얻었으면, 성공적이었던 쓰레드는, 그 록킹 시퀀스를 실행하여 상기 록을 취득한다. 이렇게 하여, 상기 그룹에서 하나의 쓰레드에 대해서만 어떠한 지점에서나 제시간에 록들을 유지시킨다. 그래도, 상기 언록킹 시퀀스는, 도 4에서와 같다 - 먼저 상기 록을 해제하고 나서, 상기 록킹 특권을 해제한다. 록들은, 상기 록 특권을 유지하면서 단일의 쓰레드에서 복수회 요구받을 수 있고, 상기 쓰레드에서 유지한 록들의 수는 록 카운터를 사용하여 추적될 수 있다. 상기 제1 실시예와 같은 방식으로 쓰레드 당 록 카운터와 상위 록 카운트를 선호하는 프로그램 카운터 선택 메커니즘을 갖는 제2 실시예를 구현하는 것이 가능하지만, 한 번에 그룹당 단일의 쓰레드에 록킹 특권(실행 우선권)을 주기 때문에, 상기 그룹에서 어떠한 쓰레드가 록킹 특권, 이 쓰레드의 아이덴티티, 및 그 그룹용 단일 록 카운터를 갖는지 여부의 단일의 표시를 전체적으로 그 그룹에 대해 저장하기만 함으로써 이 방식을 단순화시킬 수 있다.

[0065] 따라서, 도 7은 본 제2 실시예에서 사용하기 위한 레지스터 파일(180)과 상태 데이터의 예를 도시한 것이다. 도 7은 쓰레드마다 록 카운터(188)가 제공되지 않는다는 점에서 도 1의 레지스터 파일과 다르다. 대신에, 쓰레드들의 그룹용 일반 프로그램 카운터(120)와 아울러, 다수의 록 파라미터는, 쓰레드들의 그룹간에 공유된다:

[0066] · 상기 그룹에서 어떠한 쓰레드가 하나 이상의 공유 자원에 대해 록 하였는지를 나타내는 록 세트 플래그(500);

[0067] · 상기 그룹의 어느 쓰레드가 하나 이상의 공유 자원을 록 하였는지를 나타내는 록 소유자 파라미터(502); 및

[0068] · 얼마나 많은 자원이 상기 록 소유자 파라미터(502)에 의해 나타낸 록 소유 쓰레드에 의해 록 되었는지를 나타내는 록 카운트 파라미터(504).

[0069] 일부의 예들에서는, 상기 록 세트 플래그(500)를 생략하여도 되고, 대신에 상기 록 소유자 파라미터(502) 또는 록 카운트 파라미터(504)를 사용하여 쓰레드가 록을 유지하는지의 여부를 신호로 보낼 수 있었다. 예를 들면, 상기 록 카운트 파라미터(504)가 제로일 경우, 이것은, 쓰레드가 록을 유지하지 않는다는 것의 신호를 보낼 수도 있고, 상기 록 카운트 파라미터(504)가 논제로 값을 갖는 경우, 이것은 쓰레드가 록을 유지한다는 것의 신호를 보낼 수도 있다. 이와는 달리, 상기 록 소유자 파라미터(502)는, 그 쓰레드가 록을 유지하는 경우 상기 쓰레드들 중 하나에 대응한 값을 가질 수도 있고, 쓰레드가 록을 유지하지 않는 경우 쓰레드들 중 임의의 것에 대응하지 않는 값을 가질 수도 있으므로, 이것은 어떠한 록이 설정되었는지를 효과적으로 나타낼 수도 있다. 그렇지만, 어떠한 록이 설정되었는지를 나타내는 별개의 록 세트 플래그(500)를 종종 제공하는 것은, 예를 들면 상기 록 세트 플래그(500)가 상기 록 소유자 파라미터(502) 또는 록 카운트 값(504)에 대응한 멀티비트 값들보다 검사할 보다 빠르고 보다 에너지 효율적인 단일의 비트를 포함할 수도 있기 때문에, 유용할 수 있다.

[0070] 도 8은 다음과 같이 상기 제2 실시예에 대한 코드 예를 도시한 것이다:

[0071] 라인 0: 상기 프로세서에서 어떠한 특정한 액션도 일으키지 않는 상기 록킹 시퀀스의 시작을 표현하는 라벨

[0072] 라인 1: inc_lock_count는 록킹 특권(쓰레드가 자원에 대한 하나 이상의 록을 설정하기 위한 권한)을 요구하는 명령어다. 이 명령어는, 상기 그룹에서 하나의 쓰레드에 대해서 성공적으로 실행될 수 있을 뿐이다. 따라서, 이

러한 명령어 후에 하나의 쓰레드는 다음 명령어로 진행될 수 있고(이때의 쓰레드의 프로그램 카운터는 다음 명령어로 증분된다), 나머지 쓰레드들은 이 명령어를 실패한다(그래서, 그들의 쓰레드 프로그램 카운터들은 그 명령어를 성공적으로 실행할 수 있을 때까지 동일한 명령어 `inc_lock_count`를 유지한다). `inc_lock_count`가 성공적으로 실행되는 경우, 쓰레드가 록킹 특권을 유지하는 것을 나타내도록 상기 록 세트 플래그(500)를 설정하고, 현재의 쓰레드의 식별자를 나타내도록 상기 록 소유자 파라미터(502)를 설정하고, 록 카운트(504)를 증분한다.

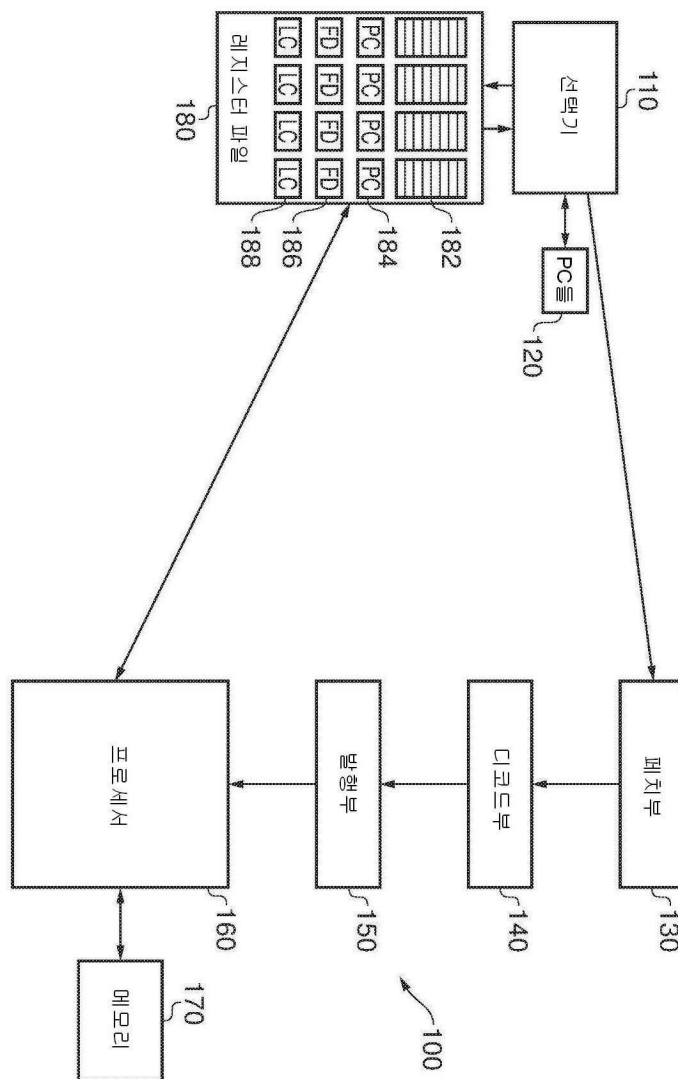
- [0073] 라인 2: 록킹 특권을 갖는 쓰레드가 록을 얻을 수 없었을 경우로 분기하는 라벨 `retry_lock`
- [0074] 라인 3-5: 다른 처리(예를 들면, 현재의 쓰레드들의 그룹 외의 쓰레드, 또는 예를 들면 멀티 프로세서 시스템에서의 다른 프로세서상에서 실행하는 처리)가 이미, 레지스터 `x19`에서의 어드레스에 의해 식별된 자원에 대한 록을 얻었는지를 검사하기 위한, 도 4의 라인 1-3와 같음.
- [0075] 라인 6: 다른 처리가 `x19`에서 식별된 어드레스에 대한 록을 유지하는 경우에 라인 10-12에서 "백 오프(back off)" 시퀀스에 분기하기 위한 조건적 분기.
- [0076] 라인 7-8: 도 4의 라인 5와 7과 같음. 레지스터 `x19`에서 상기 어드레스에 대한 록을 설정하기 위한 스토어 배타적 명령어. 상기 스토어 배타적 명령어가 실패할 경우 라인 2에 역으로 분기하기 위한 조건적 분기. 도 4의 라인 6과 달리, 도 8에서 상기 록 카운트가 이미 라인 1에서 증분되었기 때문에, 상기 록이 성공적으로 취득되는 경우 상기 록 카운트를 증분시킬 조건적 명령어가 없다.
- [0077] 라인 9: 상기 록이 성공적으로 취득되는 경우, 상기 록된 자원을 사용하여 무언가를 행하기 위해 라인 13에 분기.
- [0078] 라인 10: 레지스터 `x19`에서 상기 어드레스에 의해 식별된 자원에 대한 록이 이미 설정되었을 경우 상기 록킹 특권을 양보하기 위한 백오프 시퀀스의 시작을 표현하는 라벨 `"backoff_lock"`. 그 록킹 특권을 양보함으로써, 레지스터 `x19`의 그 버전에 있어서 상이한 어드레스를 가질 수도 있는 다른 쓰레드는, 그것의 록을 성공적으로 얻을 수 있을 수도 있어, 진행한다.
- [0079] 라인 11: 록 카운트(504)를 감분한다. 이에 따라 록 카운트(504)에서 제로가 되는 경우, 록 세트 플래그(500)를 지워, 다른 쓰레드는 록킹 특권을 얻을 수 있다. 라인 11 후에, 상기 쓰레드는 약간의 시간을 대기하여 동일한 그룹에서의 다른 쓰레드가 그 처리를 상기 록을 사용하여 끝내고 그 록을 해제할 시간을 허용할 수 있다.
- [0080] 라인 12: 상기 록을 다시 얻기를 해보기 위해 라인 0에서 록킹 시퀀스의 시작으로 역으로 분기.
- [0081] 라인 13: 상기 록이 성공적으로 취득되는 경우 실행된 상기 시퀀스의 시작을 표현하는 라벨 `"lock_acquired"`. 이후에, 임의의 수의 명령어는 상기 록된 자원을 사용하여 실행되어도 된다.
- [0082] 라인 14: 상기 록된 자원을 사용한 처리가 완료하면, 언록킹 시퀀스를 시작한다.
- [0083] 라인 15-17: 이 경우에, 상기 `dec_lock_count` 명령어가 실행되고 이에 따라 록 카운트(504)가 제로가 되는 경우, 상기 록 세트 플래그(500)는 지워지는 것을 제외하고는, 상기 록을 양보하기 위해 도 4의 라인 10-12와 같음.
- [0084] 따라서, 본 예에 의해, 쓰레드들의 동일한 그룹내에서 하나의 쓰레드만이 한 번에 록을 유지하여, 데드락에 대해 추가로 보호할 수도 있다.
- [0085] 도 9는, 제2 실시예를 이용할 때 상기 선택된 쓰레드(따라서, 그 쓰레드 프로그램 카운터를 상기 쓰레드들의 그룹용 일반 프로그램 카운터로 설정함)를 선택하는 방법을 도시한 것이다. 상기 그룹내에서 하나의 쓰레드만이 록들을 한 번에 유지할 수 있으므로, 상기 프로그램 카운터 선택 처리의 초기의 부분은 보다 단순해진다. 단계 600에서는, 상기 선택기(110)는, 상기 그룹내에서 임의의 쓰레드가 록을 유지하는지를 상기 록 세트 플래그(500)가 나타내는지를 판정한다. 그럴 경우, 단계 602에서는, 상기 선택기(110)는, 상기 록 소유자 파라미터(502)로 나타낸 쓰레드를 상기 선택된 쓰레드로서 선택한다.
- [0086] 한편, 상기 록 세트 플래그(500)는 상기 그룹에서의 쓰레드가 현재 록을 유지하지 않는 것을 나타내는 경우, 단계 640에서는, 상기 그룹(서브세트)의 모든 쓰레드들을 쓰레드들의 "제1 세트"의 일부라고 생각한다. 그리고, 그 방법은 도 5의 단계 S225로 진행되고, 그 후, 상기 방법은 함수 호출 깊이와 쓰레드 프로그램 카운터에 근거하여 선택된 쓰레드를 선택하기 위해 도 5와 마찬가지로 진행된다.
- [0087] 상기 선택된 쓰레드를 선택하였으면, 도 6의 단계 S410과 S420은 이전과 같이 적용될 수 있다.

[0088] 본 출원에서, "...하도록 구성된"의 단어는, 장치의 요소가 규정된 연산을 실시할 수 있는 구성을 갖는다는 것을 의미한다. 본 문맥에서, "구성"은, 하드웨어 또는 소프트웨어의 상호연결의 배치 또는 방식을 의미한다. 예를 들면, 상기 장치는 상기 규정된 연산을 제공하는 전용 하드웨어를 가져도 되거나, 프로세서 또는 다른 처리 디바이스는 그 기능을 행하도록 프로그래밍되어도 된다. "하도록 구성된"은, 상기 장치 요소들이 임의의 방식으로 상기 규정된 연산을 제공하기 위해서 변경될 필요가 있다는 것을 암시하지는 않는다.

[0089] 본 발명에서는 본 발명의 실시예들을 첨부도면을 참조하여 상세히 설명하였지만, 본 발명은 구체적인 실시예들에 한정되지 않고, 첨부된 청구항에서 기재된 것과 같은 본 발명의 범위 및 사상으로부터 벗어나지 않고 당업자가 변경 및 변형을 여러 가지로 실시할 수 있다.

도면

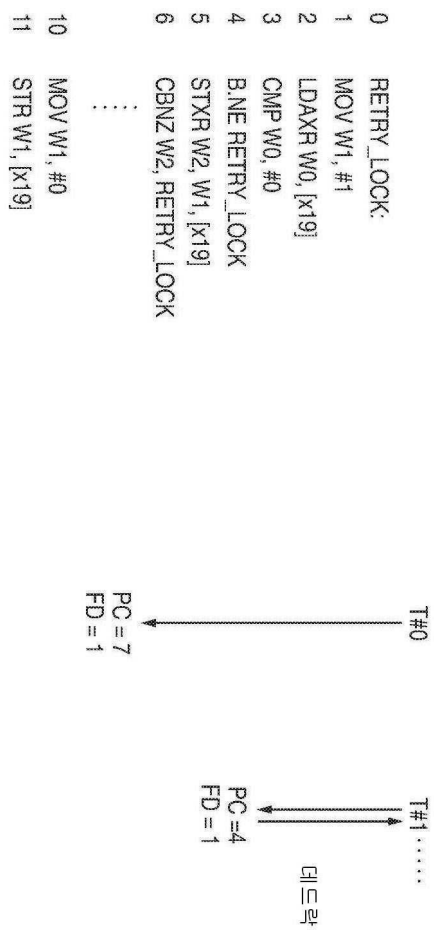
도면1



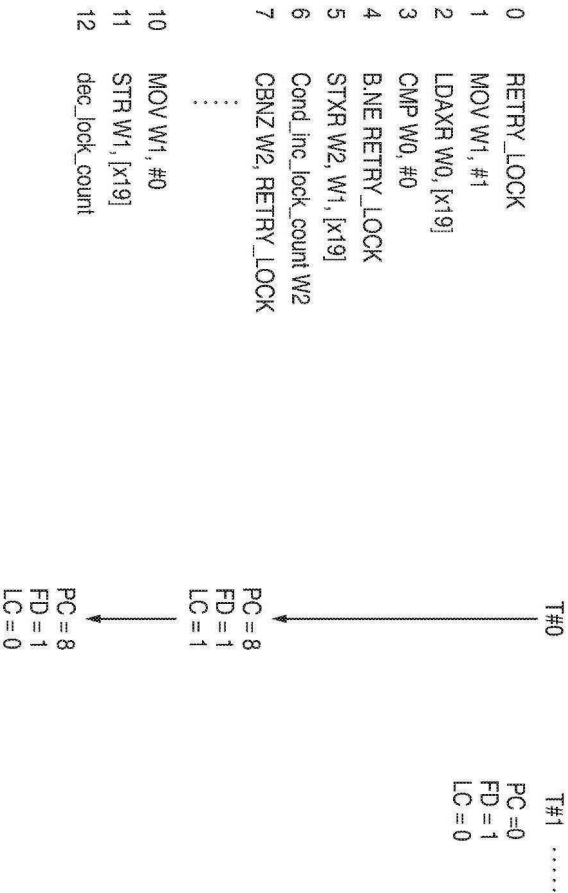
도면2

	T # 0	T # 1	T # 2	T # 3	T # 4	T # 5	T # 6	T # 31
0	adds r0, r5, r3 15 + 21 56	0 68	2 0	-1 98	5 1	997 563	32 7	6
1	beq label N	N	Y	N	Y	N	N	...
2	mul r6, r0, r4 35 * 45 235	806 2342	98 98	67 43	456 53	674 23	...	...
3	label: 100	104	108	208	20c	200	300	...
4	ldr r8, [9] 100	104	108	208	20c	200	300	...

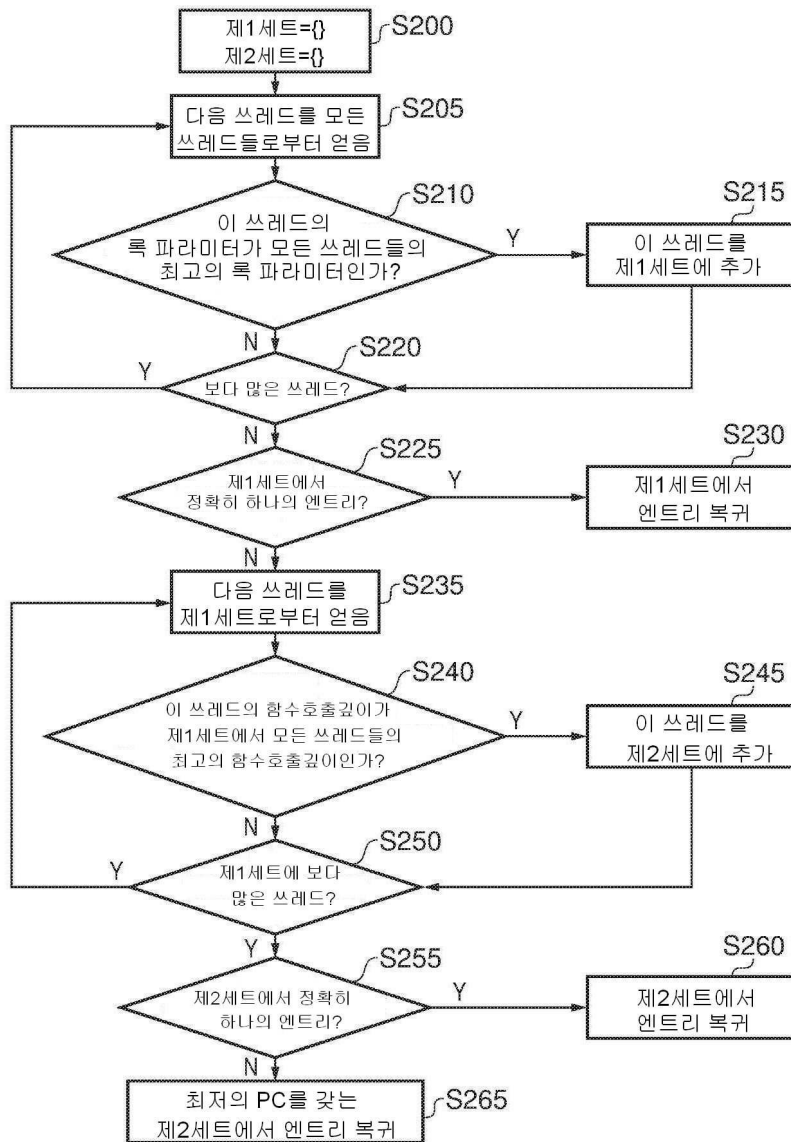
도면3



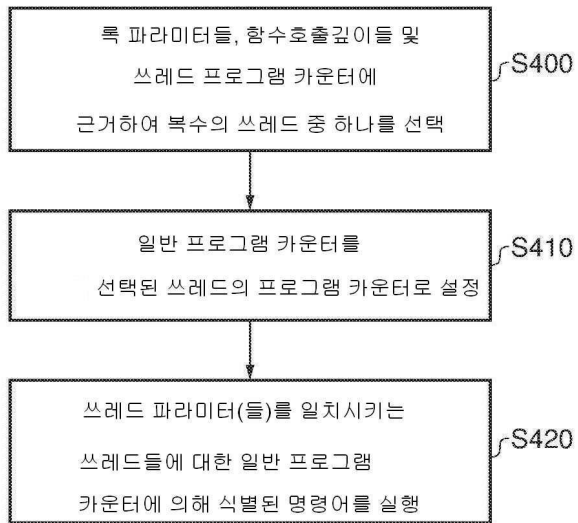
도면4



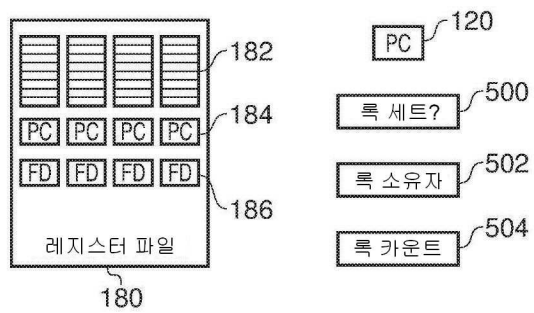
도면5



도면6



도면7



도면8

```

0   locking_sequence_start:
1   inc_lock_count
2   retry_lock:
3   mov w1, #1
4   ldaxr w0, [x19]
5   cmp w0, #0
6   bne backoff_lock
7   stxr w2, w1, [x19]
8   cbnz w2, retry_lock
9   br lock_acquired

10  backoff_lock:
11  dec_lock_count
    #decrease the lock count, in case another thread in the
    subset of threads wants a different lock and could proceed

...# wait a while, in the hope that the lock holding thread will finish
and release the lock

12  br locking_sequence_start
    #restart the full locking sequence, including increasing lock count

13  lock_acquired

...#do something with the locked data

14  release_lock:
15  mov w1, #0
16  str w1, [x19]
17  dec_lock_count

```

도면9

