(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2016년09월27일
(11) 등록번호 10-1660659
(24) 등록일자 2016년09월21일

(51) 국제특허분류(Int. Cl.)
G06F 9/38 (2006.01) *G06F 9/32* (2006.01)
(52) CPC특허분류
G06F 9/3851 (2013.01)
G06F 9/322 (2013.01)
(21) 출원번호 10-2015-7007715
(22) 출원일자(국제) 2013년08월08일
심사청구일자 2016년02월12일
(85) 번역문제출일자 2015년03월26일
(65) 공개번호 10-2015-0054858
(43) 공개일자 2015년05월20일
(86) 국제출원번호 PCT/US2013/054148
(87) 국제공개번호 WO 2014/039206
국제공개일자 2014년03월13일
(30) 우선권주장
13/608,668 2012년09월10일 미국(US)
(56) 선행기술조사문헌
JP2004206692 A
US20100257534 A1
US20110113222 A1
Collange, Sylvain. Stack-less SIMT reconvergence at low cost. Technical Report HAL-00622654, INRIA, 2011.

(73) 특허권자
퀄컴 인코포레이티드
미국 92121-1714 캘리포니아주 샌 디에고 모어하우스 드라이브 5775
(72) 발명자
천 린
미국 92121-1714 캘리포니아주 샌디에고 모어하우스 드라이브 5775
(74) 대리인
특허법인코리아나

전체 청구항 수 : 총 40 항 심사관 : 서광훈

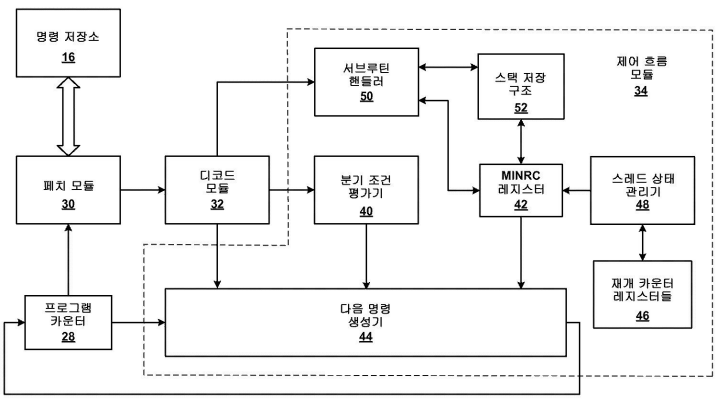
(54) 발명의 명칭 **멀티-스레딩된 프로세싱 시스템에서의 서브루틴들의 실행**

(57) 요 약

본 개시물은 발산 스레드 조건들에 종속되는 단일 명령 다중 데이터 (SIMD) 프로세싱 시스템에서 서브루틴들을 실행하기 위한 기법들에 관한 것이다. 특히, 제어 흐름 명령들의 효율적인 프로세싱을 위하여 프로그램 모듈-특정 최소 재개 카운터 (MINRC) 들을 활용하는, 발산 스레드 상태를 관리하기 위한 재개 카운터-기반 접근법이

(뒷면에 계속)

대 표 도



설명된다. 일부의 예들에서, 본 개시물의 기법들은 메인 프로그램의 실행을 제어하기 위하여 메인 프로그램 MINRC 를, 그리고 서브루틴 프로그램 모듈들의 실행을 제어하기 위하여 서브루틴-특정 MINRC 들을 이용하는 것을 포함할 수도 있다. 또한, 서브루틴 호출 및 복귀 명령들이 실행될 때, 메인 프로그램 MINRC 및 서브루틴-특정 MINRC 들을 관리하기 위한 기법들이 설명된다. 또한, 서브루틴-특정 MINRC 에 대한 업데이트된 MINRC 값이 서브루틴에 대해 할당된 프로그램 공간 내에 있는 것을 보장하기 위하여 서브루틴-특정 MINRC 를 업데이트하기 위한 기법들이 설명된다.

(52) CPC특허분류
G06F 9/3887 (2013.01)

명 세 서

청구범위

청구항 1

프로세싱 시스템에서 서브루틴들을 실행하는 방법으로서,

하나 이상의 프로세서들로, 제 1 최소 재개 카운터 (minimum resume counter; MINRC) 에 기초하여 프로그램의 실행을 제어하는 단계로서, 상기 제 1 MINRC 는 상기 프로그램을 위한 복수의 실행 스레드들과 연관된 복수의 재개 카운터 값들 중의 가장 작은 재개 카운터 값을 표시하는 값을 특정하는, 상기 프로그램의 실행을 제어하는 단계; 및

상기 하나 이상의 프로세서들로, 상기 프로그램의 서브루틴의 실행을, 상기 서브루틴과 연관된 제 2 MINRC 에 기초하여 제어하는 단계로서, 상기 제 2 MINRC 는 상기 서브루틴의 실행이 개시될 때에 활성인 상기 스레드들의 전부에 대응하는 상기 복수의 재개 카운터 값들의 서브세트로부터의 가장 작은 재개 카운터 값을 표시하는 값을 특정하는, 상기 프로그램의 상기 서브루틴의 실행을 제어하는 단계를 포함하는, 프로세싱 시스템에서 서브루틴들을 실행하는 방법.

청구항 2

제 1 항에 있어서,

서브루틴 호출 명령을 실행하는 것에 응답하여 상기 제 1 MINRC 의 상태를 저장하는 단계; 및

상기 서브루틴 호출 명령을 실행하는 것에 응답하여, 상기 서브루틴의 실행이 상기 제 2 MINRC 에 기초하여 제어되게 하는 단계를 더 포함하는, 프로세싱 시스템에서 서브루틴들을 실행하는 방법.

청구항 3

제 2 항에 있어서,

상기 제 1 MINRC 의 상태를 저장하는 단계는, 상기 제 1 MINRC 에 대해 MINRC 레지스터에 저장된 값을 스택 저장 구조 상으로 푸시하는 단계를 포함하고,

상기 서브루틴의 실행이 상기 제 2 MINRC 에 기초하여 제어되게 하는 단계는, 상기 제 2 MINRC 에 대한 초기 값으로 상기 MINRC 레지스터에 저장된 상기 값을 겹쳐쓰기하는 단계를 포함하는, 프로세싱 시스템에서 서브루틴들을 실행하는 방법.

청구항 4

제 2 항에 있어서,

서브루틴 복귀 명령을 실행하는 것에 응답하여, 상기 프로그램의 실행이 상기 제 1 MINRC 의 저장된 상기 상태에 기초하여 제어되게 하는 단계를 더 포함하는, 프로세싱 시스템에서 서브루틴들을 실행하는 방법.

청구항 5

제 4 항에 있어서,

상기 프로그램의 실행이 상기 제 1 MINRC 의 저장된 상기 상태에 기초하여 제어되게 하는 단계는 :

스택 저장 구조로부터 상기 제 1 MINRC 의 저장된 상기 상태를 팝핑 (popping) 하는 단계; 및

상기 제 1 MINRC 의 저장된 상기 상태에 대응하는 값으로 MINRC 레지스터에 저장된 값을 겹쳐쓰기하는 단계를 포함하는, 프로세싱 시스템에서 서브루틴들을 실행하는 방법.

청구항 6

제 1 항에 있어서,

상기 재개 카운터 값들의 각각은, 각각의 재개 카운터 값에 대응하는 상기 스레드들의 각각의 하나가 상기 스레드들의 상기 각각의 하나가 비활성인 경우에 활성화되도록 스케줄링되는 프로그램 카운터 값을 표시하는, 프로세싱 시스템에서 서브루틴들을 실행하는 방법.

청구항 7

제 6 항에 있어서,

상기 재개 카운터 값들의 각각은, 상기 각각의 재개 카운터 값에 대응하는 상기 스레드들의 상기 각각의 하나가 활성인 경우에 디폴트 값과 동일한, 프로세싱 시스템에서 서브루틴들을 실행하는 방법.

청구항 8

제 1 항에 있어서,

상기 복수의 재개 카운터 값들의 상기 서브세트로부터의 상기 가장 작은 재개 카운터 값을 표시하는 값으로 상기 제 2 MINRC 를 설정하는 단계를 더 포함하는, 프로세싱 시스템에서 서브루틴들을 실행하는 방법.

청구항 9

제 8 항에 있어서,

상기 제 2 MINRC 를 설정하는 단계는 :

각각의 비활성 스레드에 대하여, 상기 각각의 비활성 스레드에 대한 재개 카운터 값이 상기 제 2 MINRC 보다 더 작고 상기 각각의 비활성 스레드에 대한 상기 재개 카운터 값이 상기 서브루틴의 제 1 명령을 표시하는 값 이상인 경우에, 상기 제 2 MINRC 를 상기 각각의 비활성 스레드에 대한 상기 재개 카운터 값과 동일하게 설정하는 단계를 포함하는, 프로세싱 시스템에서 서브루틴들을 실행하는 방법.

청구항 10

제 8 항에 있어서,

상기 제 2 MINRC 를 설정하는 단계는 :

각각의 비활성 스레드에 대하여, 상기 각각의 비활성 스레드에 대한 재개 카운터 값이 제 2 MINRC 값보다 더 작고 상기 각각의 비활성 스레드와 연관된 플래그가 상기 서브루틴의 실행이 개시되었을 때에 상기 비활성 스레드가 활성이었음을 표시하는 경우에, 상기 제 2 MINRC 를 상기 각각의 비활성 스레드에 대한 상기 재개 카운터 값과 동일하게 설정하는 단계를 포함하는, 프로세싱 시스템에서 서브루틴들을 실행하는 방법.

청구항 11

제 10 항에 있어서,

상기 스레드들의 각각에 대하여, 상기 서브루틴의 실행을 개시하는 것에 응답하여, 상기 서브루틴의 실행이 개시되었을 때에 스레드가 활성인 경우에, 각각의 스레드에 대응하는 플래그를 활성 상태로 설정하는 단계; 및

상기 스레드들의 각각에 대하여, 상기 서브루틴의 실행을 개시하는 것에 응답하여, 상기 서브루틴의 실행이 개시되었을 때에 상기 스레드가 활성이 아닌 경우에, 상기 각각의 스레드에 대응하는 상기 플래그를 비활성 상태로 설정하는 단계를 더 포함하는, 프로세싱 시스템에서 서브루틴들을 실행하는 방법.

청구항 12

프로세싱 시스템에서 서브루틴들을 실행하는 디바이스로서,

제 1 최소 재개 카운터 (minimum resume counter; MINRC) 에 기초하여 프로그램의 실행을 제어하고 상기 프로그램의 서브루틴의 실행을 상기 서브루틴과 연관된 제 2 MINRC 에 기초하여 제어하도록 구성된 하나 이상의 프로세서들로서, 상기 제 1 MINRC 는 상기 프로그램을 위한 복수의 실행 스레드들과 연관된 복수의 재개 카운터 값들 중의 가장 작은 재개 카운터 값을 표시하는 값을 특정하고, 상기 제 2 MINRC 는 상기 서브루틴의 실행이 개시될 때에 활성인 상기 스레드들의 전부에 대응하는 상기 복수의 재개 카운터 값들의 서브세트로부터의 가장 작은 재개 카운터 값을 표시하는 값을 특정하는, 상기 하나 이상의 프로세서들을 포함하는, 프로세싱 시스템에서

서브루틴들을 실행하는 디바이스.

청구항 13

제 12 항에 있어서,

상기 하나 이상의 프로세서들은 또한, 서브루틴 호출 명령을 실행하는 것에 응답하여 상기 제 1 MINRC 의 상태를 저장하고, 상기 서브루틴 호출 명령을 실행하는 것에 응답하여, 상기 서브루틴의 실행이 상기 제 2 MINRC 에 기초하여 제어되게 하도록 구성되는, 프로세싱 시스템에서 서브루틴들을 실행하는 디바이스.

청구항 14

제 13 항에 있어서,

상기 하나 이상의 프로세서들은 또한, 상기 제 1 MINRC 에 대해 MINRC 레지스터에 저장된 값을 스택 저장 구조상으로 푸시하고, 상기 제 2 MINRC 에 대한 초기 값으로 상기 MINRC 레지스터에 저장된 상기 값을 겹쳐쓰기하도록 구성되는, 프로세싱 시스템에서 서브루틴들을 실행하는 디바이스.

청구항 15

제 13 항에 있어서,

상기 하나 이상의 프로세서들은 또한, 서브루틴 복귀 명령을 실행하는 것에 응답하여, 상기 프로그램의 실행이 상기 제 1 MINRC 의 저장된 상기 상태에 기초하여 제어되게 하도록 구성되는, 프로세싱 시스템에서 서브루틴들을 실행하는 디바이스.

청구항 16

제 15 항에 있어서,

상기 하나 이상의 프로세서들은 또한, 스택 저장 구조로부터 상기 제 1 MINRC 의 저장된 상기 상태를 팝핑(popping) 하고, 상기 제 1 MINRC 의 저장된 상기 상태에 대응하는 값으로 MINRC 레지스터에 저장된 값을 겹쳐쓰기하도록 구성되는, 프로세싱 시스템에서 서브루틴들을 실행하는 디바이스.

청구항 17

제 12 항에 있어서,

상기 재개 카운터 값들의 각각은, 각각의 재개 카운터 값에 대응하는 상기 스레드들의 각각의 하나가 상기 스레드들의 상기 각각의 하나가 비활성인 경우에 활성화되도록 스케줄링되는 프로그램 카운터 값을 표시하는, 프로세싱 시스템에서 서브루틴들을 실행하는 디바이스.

청구항 18

제 17 항에 있어서,

상기 재개 카운터 값들의 각각은, 상기 각각의 재개 카운터 값에 대응하는 상기 스레드들의 상기 각각의 하나가 활성인 경우에 디폴트 값과 동일한, 프로세싱 시스템에서 서브루틴들을 실행하는 디바이스.

청구항 19

제 12 항에 있어서,

상기 하나 이상의 프로세서들은 또한, 상기 복수의 재개 카운터 값들의 상기 서브세트로부터의 상기 가장 작은 재개 카운터 값을 표시하는 값으로 상기 제 2 MINRC 를 설정하도록 구성되는, 프로세싱 시스템에서 서브루틴들을 실행하는 디바이스.

청구항 20

제 19 항에 있어서,

상기 하나 이상의 프로세서들은 또한, 각각의 비활성 스레드에 대하여, 상기 각각의 비활성 스레드에 대한 재개

카운터 값이 상기 제 2 MINRC 보다 더 작고 상기 각각의 비활성 스레드에 대한 상기 재개 카운터 값이 상기 서브루틴의 제 1 명령을 표시하는 값 이상인 경우에, 상기 제 2 MINRC 를 상기 각각의 비활성 스레드에 대한 상기 재개 카운터 값과 동일하게 설정하도록 구성되는, 프로세싱 시스템에서 서브루틴들을 실행하는 디바이스.

청구항 21

제 19 항에 있어서,

상기 하나 이상의 프로세서들은 또한, 각각의 비활성 스레드에 대하여, 상기 각각의 비활성 스레드에 대한 재개 카운터 값이 제 2 MINRC 값보다 더 작고 상기 각각의 비활성 스레드와 연관된 플래그가 상기 서브루틴의 실행이 개시되었을 때에 상기 비활성 스레드가 활성이었음을 표시하는 경우에, 상기 제 2 MINRC 를 상기 각각의 비활성 스레드에 대한 상기 재개 카운터 값과 동일하게 설정하도록 구성되는, 프로세싱 시스템에서 서브루틴들을 실행하는 디바이스.

청구항 22

제 21 항에 있어서,

상기 하나 이상의 프로세서들은 또한, 상기 스레드들의 각각에 대하여, 상기 서브루틴의 실행을 개시하는 것에 응답하여, 상기 서브루틴의 실행이 개시되었을 때에 스레드가 활성인 경우에, 각각의 스레드에 대응하는 플래그를 활성 상태로 설정하고, 상기 스레드들의 각각에 대하여, 상기 서브루틴의 실행을 개시하는 것에 응답하여, 상기 서브루틴의 실행이 개시되었을 때에 상기 스레드가 활성이 아닌 경우에, 상기 각각의 스레드에 대응하는 상기 플래그를 비활성 상태로 설정하도록 구성되는, 프로세싱 시스템에서 서브루틴들을 실행하는 디바이스.

청구항 23

제 12 항에 있어서,

상기 디바이스는 무선 통신 디바이스를 포함하는, 프로세싱 시스템에서 서브루틴들을 실행하는 디바이스.

청구항 24

제 12 항에 있어서,

상기 디바이스는 이동 전화 핸드셋을 포함하는, 프로세싱 시스템에서 서브루틴들을 실행하는 디바이스.

청구항 25

프로세싱 시스템에서 서브루틴들을 실행하는 장치로서,

제 1 최소 재개 카운터 (minimum resume counter; MINRC) 에 기초하여 프로그램의 실행을 제어하기 위한 수단으로서, 상기 제 1 MINRC 는 상기 프로그램을 위한 복수의 실행 스레드들과 연관된 복수의 재개 카운터 값들 중의 가장 작은 재개 카운터 값을 표시하는 값을 특정하는, 상기 프로그램의 실행을 제어하기 위한 수단; 및

상기 프로그램의 서브루틴의 실행을 상기 서브루틴과 연관된 제 2 MINRC 에 기초하여 제어하기 위한 수단으로서, 상기 제 2 MINRC 는 상기 서브루틴의 실행이 개시될 때에 활성인 상기 스레드들의 전부에 대응하는 상기 복수의 재개 카운터 값들의 서브세트로부터의 가장 작은 재개 카운터 값을 표시하는 값을 특정하는, 상기 프로그램의 상기 서브루틴의 실행을 제어하기 위한 수단을 포함하는, 프로세싱 시스템에서 서브루틴들을 실행하는 장치.

청구항 26

제 25 항에 있어서,

서브루틴 호출 명령을 실행하는 것에 응답하여 상기 제 1 MINRC 의 상태를 저장하기 위한 수단; 및

상기 서브루틴 호출 명령을 실행하는 것에 응답하여, 상기 서브루틴의 실행이 상기 제 2 MINRC 에 기초하여 제어되게 하기 위한 수단을 더 포함하는, 프로세싱 시스템에서 서브루틴들을 실행하는 장치.

청구항 27

제 26 항에 있어서,

상기 제 1 MINRC 의 상태를 저장하기 위한 수단은, 상기 제 1 MINRC 에 대해 MINRC 레지스터에 저장된 값을 스택 저장 구조 상으로 푸시하기 위한 수단을 포함하고,

상기 서브루틴 호출 명령을 실행하는 것에 응답하여, 상기 서브루틴의 실행이 상기 제 2 MINRC 에 기초하여 제어되게 하기 위한 수단은, 상기 제 2 MINRC 에 대한 초기 값으로 상기 MINRC 레지스터에 저장된 상기 값을 겹쳐쓰기하기 위한 수단을 포함하는, 프로세싱 시스템에서 서브루틴들을 실행하는 장치.

청구항 28

제 26 항에 있어서,

서브루틴 복귀 명령을 실행하는 것에 응답하여, 상기 프로그램의 실행이 상기 제 1 MINRC 의 저장된 상기 상태에 기초하여 제어되게 하기 위한 수단을 더 포함하는, 프로세싱 시스템에서 서브루틴들을 실행하는 장치.

청구항 29

제 28 항에 있어서,

상기 서브루틴 복귀 명령을 실행하는 것에 응답하여, 상기 프로그램의 실행이 상기 제 1 MINRC 의 저장된 상기 상태에 기초하여 제어되게 하기 위한 수단은 :

스택 저장 구조로부터 상기 제 1 MINRC 의 저장된 상기 상태를 팝핑 (popping) 하기 위한 수단; 및

상기 제 1 MINRC 의 저장된 상기 상태에 대응하는 값으로 MINRC 레지스터에 저장된 값을 겹쳐쓰기하기 위한 수단을 포함하는, 프로세싱 시스템에서 서브루틴들을 실행하는 장치.

청구항 30

제 25 항에 있어서,

상기 재개 카운터 값들의 각각은, 각각의 재개 카운터 값에 대응하는 상기 스레드들의 각각의 하나가 상기 스레드들의 상기 각각의 하나가 비활성인 경우에 활성화되도록 스케줄링되는 프로그램 카운터 값을 표시하는, 프로세싱 시스템에서 서브루틴들을 실행하는 장치.

청구항 31

제 30 항에 있어서,

상기 재개 카운터 값들의 각각은, 상기 각각의 재개 카운터 값에 대응하는 상기 스레드들의 상기 각각의 하나가 활성인 경우에 디폴트 값과 동일한, 프로세싱 시스템에서 서브루틴들을 실행하는 장치.

청구항 32

제 25 항에 있어서,

상기 복수의 재개 카운터 값들의 상기 서브세트로부터의 가장 작은 재개 카운터 값을 표시하는 값으로 상기 제 2 MINRC 를 설정하기 위한 수단을 더 포함하는, 프로세싱 시스템에서 서브루틴들을 실행하는 장치.

청구항 33

프로세싱 시스템에서 서브루틴들을 실행하는 명령들을 저장하는 비일시적인 컴퓨터 판독가능한 저장 매체로서,

상기 명령들은, 실행될 경우, 하나 이상의 프로세서들로 하여금 :

제 1 최소 재개 카운터 (minimum resume counter; MINRC) 에 기초하여 프로그램의 실행을 제어하게 하는 것으로서, 상기 제 1 MINRC 는 상기 프로그램을 위한 복수의 실행 스레드들과 연관된 복수의 재개 카운터 값들 중의 가장 작은 재개 카운터 값을 표시하는 값을 특정하는, 상기 프로그램의 실행을 제어하게 하고;

상기 프로그램의 서브루틴의 실행을 상기 서브루틴과 연관된 제 2 MINRC 에 기초하여 제어하게 하는 것으로서, 상기 제 2 MINRC 는 상기 서브루틴의 실행이 개시될 때에 활성인 상기 스레드들의 전부에 대응하는 상기 복수의 재개 카운터 값들의 서브세트로부터의 가장 작은 재개 카운터 값을 표시하는 값을 특정하는, 상기 프

로그램의 상기 서브루틴의 실행을 제어하게 하는, 비일시적인 컴퓨터 판독가능한 저장 매체.

청구항 34

제 33 항에 있어서,

실행될 경우, 하나 이상의 프로세서들로 하여금 :

서브루틴 호출 명령을 실행하는 것에 응답하여 상기 제 1 MINRC 의 상태를 저장하게 하고;

상기 서브루틴 호출 명령을 실행하는 것에 응답하여, 상기 서브루틴의 실행이 상기 제 2 MINRC 에 기초하여 제어되게 하는

명령들을 더 포함하는, 비일시적인 컴퓨터 판독가능한 저장 매체.

청구항 35

제 34 항에 있어서,

상기 하나 이상의 프로세서들로 하여금, 상기 제 1 MINRC 의 상태를 저장하게 하는 명령들은, 상기 하나 이상의 프로세서들로 하여금, 상기 제 1 MINRC 에 대해 MINRC 레지스터에 저장된 값을 스택 저장 구조 상으로 푸시하게 하는 명령들을 포함하고,

상기 하나 이상의 프로세서들로 하여금, 상기 서브루틴 호출 명령을 실행하는 것에 응답하여, 상기 서브루틴의 실행이 상기 제 2 MINRC 에 기초하여 제어되게 하는 명령들은, 상기 하나 이상의 프로세서들로 하여금, 상기 제 2 MINRC 에 대한 초기 값으로 상기 MINRC 레지스터에 저장된 상기 값을 겹쳐쓰기하게 하는 명령들을 포함하는, 비일시적인 컴퓨터 판독가능한 저장 매체.

청구항 36

제 34 항에 있어서,

실행될 경우, 하나 이상의 프로세서들로 하여금 :

서브루틴 복귀 명령을 실행하는 것에 응답하여, 상기 프로그램의 실행이 상기 제 1 MINRC 의 저장된 상기 상태에 기초하여 제어되게 하는

명령들을 더 포함하는, 비일시적인 컴퓨터 판독가능한 저장 매체.

청구항 37

제 36 항에 있어서,

상기 하나 이상의 프로세서들로 하여금, 상기 서브루틴 복귀 명령을 실행하는 것에 응답하여, 상기 프로그램의 실행이 상기 제 1 MINRC 의 저장된 상기 상태에 기초하여 제어되게 하는 명령들은, 상기 하나 이상의 프로세서들로 하여금 :

스택 저장 구조로부터 상기 제 1 MINRC 의 저장된 상기 상태를 팝핑 (popping) 하게 하고;

상기 제 1 MINRC 의 저장된 상기 상태에 대응하는 값으로 MINRC 레지스터에 저장된 값을 겹쳐쓰기하게 하는

명령들을 포함하는, 비일시적인 컴퓨터 판독가능한 저장 매체.

청구항 38

제 33 항에 있어서,

상기 재개 카운터 값들의 각각은, 각각의 재개 카운터 값에 대응하는 상기 스레드들의 각각의 하나가 상기 스레드들의 상기 각각의 하나가 비활성인 경우에 활성화되도록 스케줄링되는 프로그램 카운터 값을 표시하는, 비일시적인 컴퓨터 판독가능한 저장 매체.

청구항 39

제 38 항에 있어서,

상기 재개 카운터 값들의 각각은, 상기 각각의 재개 카운터 값에 대응하는 상기 스레드들의 상기 각각의 하나가 활성인 경우에 디폴트 값과 동일한, 비일시적인 컴퓨터 판독가능한 저장 매체.

청구항 40

제 33 항에 있어서,

실행될 경우, 하나 이상의 프로세서들로 하여금 :

상기 복수의 재개 카운터 값들의 상기 서브세트로부터의 가장 작은 재개 카운터 값을 표시하는 값으로 상기 제 2 MINRC 를 설정하게 하는

명령들을 더 포함하는, 비일시적인 컴퓨터 판독가능한 저장 매체.

발명의 설명

기 술 분 야

[0001] 본 개시물은 멀티-스레딩된 (multi-threaded) 프로세싱에 관한 것으로, 더욱 상세하게는, 멀티-스레딩된 프로세싱 시스템에서 서브루틴들을 실행하기 위한 기법들에 관한 것이다.

배 경 기 술

[0002] 단일 명령 다중 데이터 (single instruction multiple data; SIMD) 프로세싱 시스템은 데이터의 다중 피스 (multiple piece) 들에 대해 동일한 명령을 실행하는 다중 프로세싱 엘리먼트들을 포함하는 일 타입의 병렬 컴퓨팅 시스템이다. SIMD 시스템은 스탠드얼론 (standalone) 컴퓨터 또는 컴퓨팅 시스템의 서브-시스템일 수도 있다. 예를 들어, 하나 이상의 SIMD 실행 유닛들은 프로그래밍가능한 쉐이딩 (programmable shading) 을 지원하는 프로그래밍가능한 쉐이딩 유닛을 구현하기 위하여 그래픽 프로세싱 유닛 (graphics processing unit; GPU) 에서 이용될 수도 있다.

[0003] SIMD 프로세싱 시스템은 프로그램을 위한 다중의 실행 스레드 (thread of execution) 들이 병렬 방식의 다중 프로세싱 엘리먼트들 상에서 동기식으로 실행하도록 함으로써, 동일한 세트의 동작들이 데이터의 다중 피스들에 대해 수행될 필요가 있는 프로그램들에 대한 스루풋 (throughput) 을 증가시킨다. 각각의 스레드는 상이한 데이터에 대해 동작하므로, 프로그램이 조건부 분기 명령 (conditional branch instruction) 들을 포함하는 경우, 분기 조건은 시스템에서 실행되는 스레드들의 일부에 대해 충족될 수도 있고 시스템에서 실행되는 다른 스레드들에 대해서는 충족되지 않을 수도 있는 것이 가능하다. 이러한 조건은 발산 분기 조건 (divergent branch condition) 으로 지칭될 수도 있고, SIMD 시스템이 다중 프로세싱 엘리먼트들 상에서 스레드들의 전부를 동기식 방식으로 실행할 수 없게 한다.

발명의 내용

[0004] 본 개시물은 발산 스레드 조건 (divergent thread condition) 들에 종속되는 단일 명령 다중 데이터 (SIMD) 프로세싱 시스템에서 서브루틴들을 실행하기 위한 기법들에 관한 것이다. 특히, 제어 흐름 명령들의 효율적인 프로세싱을 위하여 프로그램 모듈-특정 최소 재개 카운터 (minimum resume counter; MINRC) 들을 활용하는, 발산 스레드들을 관리하기 위한 재개 카운터-기반 접근법이 설명된다. 본원에서 이용된 바와 같이, 프로그램 모듈은 메인 프로그램 모듈 (예를 들어, 최상위-레벨 프로그램 모듈) 또는 서브루틴 프로그램 모듈을 지칭할 수도 있다. 이와 같이, 프로세싱 시스템에서 실행되는 각각의 서브루틴은 서브루틴에 포함된 제어 흐름 명령들의 프로세싱을 제어하기 위하여 서브루틴-특정 MINRC 를 이용할 수도 있다. 프로그램 모듈-특정 MINRC 들의 이용은 MINRC-기반 제어 흐름을 구현하는 시스템이 서브루틴 프로그램 명령들의 실행을 지원하도록 한다.

[0005] 하나의 예에서, 본 개시물은 하나 이상의 프로세서들로, 제 1 MINRC 에 기초하여 프로그램의 실행을 제어하는 단계를 포함하는 방법을 설명한다. 제 1 MINRC 는 프로그램을 위한 복수의 실행 스레드들과 연관된 복수의 재개 카운터 값들 중의 가장 작은 재개 카운터 값을 표시하는 값을 특정한다. 방법은 하나 이상의 프로세서들로, 서브루틴과 연관된 제 2 MINRC 에 기초하여 프로그램의 서브루틴의 실행을 제어하는 단계를 더 포함한다. 제 2 MINRC 는, 서브루틴의 실행이 개시될 때에 활성 (active) 인 스레드들의 전부에 대응하는 복수의 재개

카운터 값들의 서브세트 (subset) 로부터의 가장 작은 재개 카운터 값을 표시하는 값을 특정한다.

[0006] 또 다른 예에서, 본 개시물은, 제 1 MINRC 에 기초하여 프로그램의 실행을 제어하고 서브루틴과 연관된 제 2 MINRC 에 기초하여 프로그램의 서브루틴의 실행을 제어하도록 구성된 하나 이상의 프로세서들을 포함하는 시스템을 설명한다. 제 1 MINRC 는 프로그램을 위한 복수의 실행 스레드들과 연관된 복수의 재개 카운터 값들 중의 가장 작은 재개 카운터 값을 표시하는 값을 특정한다. 제 2 MINRC 는, 서브루틴의 실행이 개시될 때에 활성인 스레드들의 전부에 대응하는 복수의 재개 카운터 값들의 서브세트로부터의 가장 작은 재개 카운터 값을 표시하는 값을 특정한다.

[0007] 또 다른 예에서, 본 개시물은 제 1 MINRC 에 기초하여 프로그램의 실행을 제어하기 위한 수단을 포함하는 장치를 설명한다. 제 1 MINRC 는 프로그램을 위한 복수의 실행 스레드들과 연관된 복수의 재개 카운터 값들 중의 가장 작은 재개 카운터 값을 표시하는 값을 특정한다. 장치는 서브루틴과 연관된 제 2 MINRC 에 기초하여 프로그램의 서브루틴의 실행을 제어하기 위한 수단을 더 포함한다. 제 2 MINRC 는, 서브루틴의 실행이 개시될 때에 활성인 스레드들의 전부에 대응하는 복수의 재개 카운터 값들의 서브세트로부터의 가장 작은 재개 카운터 값을 표시하는 값을 특정한다.

[0008] 또 다른 예에서, 본 개시물은, 실행될 경우, 하나 이상의 프로세서들로 하여금, 제 1 MINRC 에 기초하여 프로그램의 실행을 제어하게 하는 명령들을 저장하는 컴퓨터 판독가능한 저장 매체를 설명한다. 제 1 MINRC 는 프로그램을 위한 복수의 실행 스레드들과 연관된 복수의 재개 카운터 값들 중의 가장 작은 재개 카운터 값을 표시하는 값을 특정한다. 명령들은 또한, 하나 이상의 프로세서들로 하여금, 서브루틴과 연관된 제 2 MINRC 에 기초하여 프로그램의 서브루틴의 실행을 제어하게 한다. 제 2 MINRC 는, 서브루틴의 실행이 개시될 때에 활성인 스레드들의 전부에 대응하는 복수의 재개 카운터 값들의 서브세트로부터의 가장 작은 재개 카운터 값을 표시하는 값을 특정한다.

[0009] 개시물의 하나 이상의 예들의 세부사항들은 첨부한 도면들 및 이하의 설명에서 기재된다. 개시물의 다른 특징들, 목적들, 및 장점들은 설명 및 도면들로부터, 그리고 청구항들로부터 명백할 것이다.

도면의 간단한 설명

[0010] 도 1 은 본 개시물의 서브루틴 실행 기법들을 구현하기 위하여 이용될 수도 있는 일 예의 프로세싱 시스템을 예시하는 블록도이다.

도 2 는 본 개시물에 따라 도 1 의 일 예의 프로세싱 시스템에서의 제어 유닛을 더욱 상세하게 예시하는 블록도이다.

도 3 은 본 개시물의 서브루틴 실행 기법들을 구현하기 위하여 이용될 수도 있는 일 예의 제어 흐름 모듈을 예시하는 블록도이다.

도 4 는 본 개시물의 서브루틴 실행 기법들을 위한 일 예의 제어 흐름을 예시하는 개념도이다.

도 5 는 본 개시물의 서브루틴 실행 기법들을 위한 또 다른 예의 제어 흐름을 예시하는 개념도이다.

도 6 은 본 개시물의 기법들에 따라 일 예의 프로그램 공간 배치들을 예시하는 개념도이다.

도 7 내지 도 18 은 본 개시물의 서브루틴 실행 기법들을 활용하는 일 예의 명령 프로세싱 기법들을 예시하는 흐름도들이다.

도 19 는 본 개시물의 서브루틴 실행 기법들을 구현하기 위하여 이용될 수도 있는 또 다른 예의 제어 흐름 모듈을 예시하는 블록도이다.

도 20 은 본 개시물에 따라 도 19 에서 예시된 제어 흐름 모듈의 예시적인 동작을 특징화하는 상태 천이도이다.

도 21 은 본 개시물에 따라 도 19 에서 예시된 제어 흐름 모듈의 예시적인 동작을 특징화하는 상태 천이표이다.

도 22 내지 도 28 은 본 개시물의 서브루틴 실행 기법들을 구현하기 위한 일 예의 의사-코드 (pseudo-code) 를 예시한다.

도 29 는 본 개시물에 따라 프로그램 모듈-특정 MINRC 들에 기초하여 프로세싱 시스템을 제어하기 위한 일 예의 기법을 예시하는 흐름도이다.

도 30 은 본 개시물에 따라 서브루틴 호출 명령을 실행하기 위한 일 예의 기법을 예시하는 흐름도이다.

도 31 은 본 개시물에 따라 서브루틴 복귀 명령을 실행하기 위한 일 예의 기법을 예시하는 흐름도이다.

발명을 실시하기 위한 구체적인 내용

[0011] 본 개시물은 발산 스레드 조건들에 종속되는 단일 명령 다중 데이터 (SIMD) 프로세싱 시스템에서 서브루틴들을 실행하기 위한 기법들에 관한 것이다. 특히, 제어 흐름 명령들의 효율적인 프로세싱을 위하여 프로그램 모듈-특정 최소 재개 카운터 (MINRC) 들을 활용하는, 발산 스레드들을 관리하기 위한 재개 카운터-기반 접근법이 설명된다. 본원에서 이용된 바와 같이, 프로그램 모듈은 메인 프로그램 모듈 (예를 들어, 최상위-레벨 프로그램 모듈) 또는 서브루틴 프로그램 모듈을 지칭할 수도 있다. 이와 같이, 프로세싱 시스템에서 실행되는 각각의 서브루틴은 서브루틴에 포함된 제어 흐름 명령들의 프로세싱을 제어하기 위하여 서브루틴-특정 MINRC 를 이용할 수도 있다. 프로그램 모듈-특정 MINRC 들의 이용은 MINRC-기반 제어 흐름을 구현하는 시스템이 서브루틴 프로그램 명령들의 실행을 지원하도록 한다.

[0012] 일부의 예들에서, 본 개시물의 기법들은 메인 프로그램의 실행을 제어하기 위하여 메인 프로그램 MINRC 를, 그리고 메인 프로그램에 의해 또는 다른 서브루틴들에 의해 호출되는 서브루틴들의 실행을 제어하기 위하여 서브루틴-특정 MINRC 들을 이용하는 것을 포함할 수도 있다. 메인 프로그램 MINRC 값은, 전형적으로 시스템에서 실행되는 스레드들의 전부인, 메인 프로그램의 실행이 개시될 때에 활성인 모든 스레드들에 대응하는 하나 이상의 재개 카운터 값들의 세트로부터의 가장 작은 재개 카운터 값을 표시할 수도 있다. 유사하게, 각각의 서브루틴-특정 MINRC 는, 각각의 서브루틴의 실행이 개시될 때에 활성인 모든 스레드들에 대응하는 하나 이상의 재개 카운터 값들의 세트로부터의 가장 작은 재개 카운터 값을 표시하는 값을 특정할 수도 있다. 각각의 재개 카운터 값은 프로세싱 시스템 상에서 실행되는 복수의 스레드들의 각각의 하나에 대응할 수도 있고, 각각의 스레드가 비활성인 경우, 재개 카운터 값은 비활성 스레드가 재활성화 (reactivate) 되도록 스케줄링되는 프로그램 카운터 값을 표시할 수도 있다.

[0013] 프로세싱 시스템의 실행을 현재 제어하고 있는 MINRC 는, 하나 이상의 스레드들이 비활성화 (deactivate) 될 때, 순방향 점프 명령 후에 또는 순방향 분기 명령 후에 어느 명령이 실행되는지를 제어하기 위하여 이용될 수도 있다. 예를 들어, 어떤 경우들에 있어서, 순방향 점프 명령 또는 순방향 분기 명령을 실행한 후에, 프로세싱 시스템은 실행되어야 할 다음 명령으로서, MINRC 에 의해 표시된 명령을 선택할 수도 있다. 메인 프로그램에 대해 그리고 서브루틴들의 각각에 대해 별도의 MINRC 들을 이용함으로써, 프로세싱 시스템은 서브루틴들이 서브루틴으로부터 메인 프로그램으로, 또는 호출 또는 복귀 명령의 실행 이외의 또 다른 서브루틴으로 분기하지 않는다는 것을 보장할 수도 있다. 이러한 방법으로, 본 개시물의 기법들은 순방향 점프 명령들 및 순방향 분기 명령들의 실행을 제어하기 위하여 MINRC 들을 이용하는 프로세싱 시스템에서 서브루틴들의 실행을 위한 적당한 제어 흐름을 보장할 수도 있다.

[0014] 본 개시물의 일부의 양태들에 따르면, 서브루틴의 진입 및 진출 시에 프로세싱 시스템의 실행을 제어하기 위하여 이용되는 MINRC 를 전환하기 위한 기법들이 설명되어 있다. 예를 들어, 호출 명령을 실행하는 것에 응답하여, 프로세싱 시스템은 호출자 프로그램 (caller program) 에 대응하는 MINRC 의 상태를 저장하고, 서브루틴 프로그램에 대응하는 새로운 MINRC 를 초기화하고, 서브루틴 프로그램에 대응하는 MINRC 에 기초하여 서브루틴의 실행을 제어하도록 구성될 수도 있다. 복귀 명령을 실행하는 것에 응답하여, 프로세싱 시스템은 호출자 프로그램에 대응하는 MINRC 의 저장된 상태를 복원하고, MINRC 의 복원된 상태에 기초하여 호출자 프로그램의 실행을 제어하는 것을 재개하도록 구성될 수도 있다.

[0015] 본 개시물의 일부의 양태들에 따르면, 프로세싱 시스템에서 하나 이상의 스레드들을 활성화 (activate) 및/또는 비활성화하는 것에 응답하여 MINRC 값을 업데이트하기 위한 기법들이 설명되어 있다. 일반적으로, MINRC 값을 업데이트하기 위하여, 프로세싱 시스템은 MINRC 값을 결정함에 있어서 이용하기 위한 재개 카운터 값들의 후보 세트를 결정할 수도 있고, 재개 카운터 값들의 후보 세트로부터의 가장 작은 재개 카운터 값을 표시하는 값으로 MINRC 를 설정할 수도 있다. 재개 카운터 값들의 후보 세트는 프로세싱 시스템에서 실행되는 모든 스레드들에 대응하는 재개 카운터 값들의 전체 세트의 서브세트일 수도 있다. 일부의 경우들에 있어서, 재개 카운터 값들의 후보 세트는, 현재 실행하는 프로그램 모듈의 실행이 개시되었을 때에 활성이 아니었던 스레드들에 대응하는 하나 이상의 재개 카운터 값들을 배제할 수도 있다. MINRC 를 업데이트하는 동안의 고려사항으로부터 이러한 재개 카운터 값들을 배제함으로써, 본 개시물의 기법들은 서브루틴-특정 MINRC 가 서브루틴과 연관된 프로그램 공간 내에 있는 값들로 업데이트되는 것을 보장할 수도 있다.

[0016] SIMD 프로세싱 시스템의 명령 세트 아키텍처 (instruction set architecture; ISA) 가 제어 흐름 명령들을 지

원하는 경우, 모든 스레드들은 단일 프로그램 카운터를 포함하는 단일 제어 흐름 유닛에 의해 제어될 수도 있다. 각각의 스레드는 상이한 데이터에 대해 동작하므로, 특별한 분기 명령에 대한 분기 조건이 시스템에서 실행되는 스레드들의 일부에 대해 충족될 수도 있고 시스템에서 실행되는 다른 스레드들에 대해서는 충족되지 않을 수도 있는 것이 가능하다. 특별한 조건부 분기 명령에 의해 특정된 조건이 시스템에서 실행되는 활성 스레드들의 전부에 대해 충족되거나 또는 충족되지 않는 경우에는, 분기 명령에 대한 분기 발산 (branching divergence) 이 균일하다고 말해진다. 이와 다르게, 조건이 활성 스레드들의 일부에 대해 충족되고 활성 스레드들의 다른 것들에 대해 충족되지 않는 경우에는, 분기 명령에 대한 분기 발산이 발산하는 것이라고 말해진다. 발산 분기가 발생하는 경우, 활성 스레드들의 일부에 대해 실행되도록 스케줄링되는 다음 명령은 활성 스레드들의 다른 것들에 대해 실행되도록 스케줄링되는 다음 명령과는 상이할 수도 있다. 이것은 SIMD 프로세싱 시스템이 스레드들의 전부를 록스텝 (lockstep) 방식으로 실행할 수 없게 할 수도 있다.

[0017] 발산 분기 명령을 처리하기 위하여, 일부의 예들에서, 본 개시물의 기법들은 동일한 다음 명령을 실행하기 위하여 나머지 활성 스레드들이 모두 동기화되도록, 분기 조건을 충족하였거나 이를 충족하지 않았던 스레드들의 하나의 서브세트를 비활성화할 수도 있다. 비활성화된 스레드들의 재활성화를 제어하기 위하여, 본 개시물의 기법들은, 프로세싱 시스템에서 실행되는 각각의 스레드에 대한 재개 카운터를 할당하는 것, 비활성화되고 있는 각각의 스레드에 대한 재개 카운터를, 각각의 스레드가 재활성화되어야 하는 프로그램 카운터 값을 표시하는 값으로 설정하는 것, 및 매 명령의 실행 이전에 재개 검사 동작을 수행하는 것을 포함하는 재개 카운터-기반 접근법을 이용할 수도 있다.

[0018] 예를 들어, 재개 카운터-기반 접근법은 발산 분기 조건에 응답하여 하나 이상의 스레드들을 비활성화할 수도 있고, 비활성화되고 있는 각각의 스레드에 대하여, 각각의 스레드에 대한 재개 카운터 (예를 들어, 레지스터) 를, 비활성화된 스레드에 의해 실행되어야 할 다음 명령에 대응하는 프로그램 카운터 값을 표시하는 값으로 설정할 수도 있다. 스레드가 활성인 경우, 스레드에 대한 재개 카운터는 디폴트 (default) 값으로 설정될 수도 있다. 일부의 예들에서, 디폴트 값은 프로그램의 어드레스 범위보다 더 큰 값 (예를 들어, 최대 레지스터 값) 에 대응하는 "무한 값 (infinite value)" 일 수도 있다. 프로그램 카운터 레지스터가 새로운 프로그램 카운터 값으로 로딩될 때마다, 재개 검사 동작이 수행될 수도 있고, 이것은 스레드에 대한 재개 카운터 값이 새로운 프로그램 카운터 값과 동일한 임의의 스레드들을 재활성화할 수도 있다.

[0019] 적당한 제어 흐름이 스레드 재활성화를 위한 상기 설명된 재개 카운터-기반 접근법을 이용하는 프로세싱 시스템에서 유지되는 것을 보장하기 위하여, 프로세싱 시스템은 "최저 값의 어드레스 우선 (least-valued address first)" 스레드 프로세싱 순서를 이용할 수도 있다. 일반적으로, "최저 값의 어드레스 우선" 스레드 프로세싱 순서는, 더 낮은 값의 어드레스들에서의 명령들을 프로세싱하도록 스케줄링되는 스레드들이 더 높은 값의 어드레스들에서의 명령들을 프로세싱하도록 스케줄링되는 스레드들 이전에 실행되는 프로세싱 순서를 지칭할 수도 있다. 이러한 프로세싱 순서는 제어 흐름이 이러한 스레드들을 먼저 재활성화하지 않고 비활성 스레드들에 대한 임의의 재개 포인트들 위로 점프하는 것을 방지할 수도 있다. 다시 말해서, 이러한 프로세싱 순서는, 모든 스레드들이 활성일 것이고 최종 프로그램 명령문 (statement) 이 실행을 종료한 시간까지 프로세싱을 완료하였을 것이라는 것을 보장할 수도 있다.

[0020] "최저 값의 어드레스 우선" 스레드 프로세싱 순서는 분기 명령의 방향 (즉, 순방향 또는 역방향) 에 기초하여 어느 스레드들이 발산 분기 명령에 응답하여 비활성화되는지를 구별할 수도 있다. 발산 역방향 분기 명령에 대하여, 본 개시물의 기법들은 분기 조건이 충족되지 않는 스레드들을 비활성화할 수도 있고, 비활성화되고 있는 각각의 스레드에 대한 재개 카운터 값을, 분기 명령 후에 발생하는 다음의 순차적인 명령과 연관된 값으로 설정할 수도 있고, 분기 명령에 의해 특정된 타겟 명령과 연관된 값으로 프로그램 카운터를 로딩할 수도 있고, 분기 조건이 충족되는 그러한 스레드들을 실행하도록 진행할 수도 있다. 발산 순방향 분기 명령에 대하여, 본 개시물의 기법들은 분기 조건이 충족되는 스레드들을 비활성화할 수도 있고, 비활성화되고 있는 각각의 스레드에 대한 재개 카운터 값을, 분기 명령에 의해 특정된 타겟 명령과 연관된 값으로 설정할 수도 있고, 분기 명령 후에 발생하는 다음의 순차적인 명령과 연관된 값으로 프로그램 카운터를 로딩할 수도 있고, 분기 조건이 충족되지 않는 그러한 스레드들을 실행하도록 진행할 수도 있다. 스레드들을 이러한 방식으로 비활성화하는 것은, 더 낮은 값의 어드레스들에서의 명령들을 프로세싱하도록 스케줄링되는 발산 스레드들이 더 높은 값의 어드레스들에서의 명령들을 프로세싱하도록 스케줄링되는 스레드들 이전에 실행되는 것을 보장한다 (즉, "최저 값의 어드레스 우선" 스레드 프로세싱 순서).

[0021] 하나 이상의 스레드들이 이미 비활성화되었고 나머지 활성 스레드들이 순방향 점프 명령 또는 균일하게 충족된 순방향 분기 명령 (즉, 분기 조건이 모든 활성 스레드들에 대해 균일하게 충족되는 순방향 분기 명령) 의 어느

하나를 실행하는 경우들에 있어서, 제어 흐름을 핸들링하는 것에 대한 하나의 접근법은, 모든 활성 스레드들이 활성으로 남아 있을 것이므로 순방향 점프 또는 순방향 분기 명령에서 특정된 타겟 명령으로 항상 점프하는 것일 수도 있다. 그러나, 이러한 접근법은 "최저 값의 어드레스 우선" 스레드 프로세싱 순서를 보장하지 않는다. 특히, 일부의 경우들에 있어서, 하나 이상의 비활성 스레드들은 점프 또는 분기 명령의 현재의 프로그램 카운터 값과 타겟 프로그램 카운터 값 (즉, 분기 또는 점프 명령에서 특정된 타겟 명령과 연관된 프로그램 카운터 값) 사이에 있는 재개 카운터 값들을 가질 수도 있다. 제어 흐름이 이러한 비활성 스레드들 위로 점프하였을 경우, 이러한 스레드들은 프로그램의 실행을 종료하기 이전에 재활성화되지 않을 것이라는 것이 가능하다.

[0022] 이러한 상황을 회피하기 위하여, 시스템에서의 모든 스레드들에 대응하는 재개 카운터 값들의 세트로부터의 가장 작은 재개 카운터 값을 표시하는 값을 저장하는 글로벌 (global) MINRC 가 이용될 수도 있다. 재개 카운터들 중의 임의의 것이 (예를 들어, 스레드의 비활성화 시에) 새로운 값으로 설정될 때, MINRC 는 새로운 가장 작은 재개 카운터 값을 반영하기 위하여 업데이트될 수도 있다. 프로세싱 시스템은 현재 실행된 순방향 점프 또는 순방향 분기 명령과 그 명령에서 특정된 타겟 명령 사이에 임의의 재개 포인트들이 있는지 여부를 결정하기 위하여 MINRC 를 이용할 수도 있고, 이것은 궁극적으로, 프로세싱 시스템이 시스템으로 하여금 비활성 스레드들에 대한 하나 이상의 재개 포인트들을 건너뛰지 않게 하고 타겟 명령으로 직접 점프할 수 있는지 여부를 결정하기 위하여 이용될 수도 있다.

[0023] 예를 들어, 균일하게 충족되는 순방향 점프 명령 또는 순방향 분기 명령을 실행할 때, 프로세싱 시스템은 점프 또는 분기 명령에 의해 특정된 타겟 프로그램 카운터 값을 MINRC 값과 비교할 수도 있고, 비교에 기초하여 프로그램 카운터로 로딩하기 위한 타겟 프로그램 카운터 값 또는 MINRC 값 중의 어느 하나를 선택할 수도 있다. 타겟 프로그램 카운터 값이 MINRC 값 이하일 때, 프로세싱 시스템은 프로그램 카운터로 로딩하기 위한 값으로서 타겟 프로그램 카운터 값을 선택할 수도 있다. 타겟 프로그램 카운터 값이 MINRC 값 이하가 아닐 때, 프로세싱 시스템은 프로그램 카운터로 로딩하기 위한 값으로서 MINRC 를 선택할 수도 있다. MINRC 값이 모든 비활성 스레드들 중의 가장 작은 재개 카운터 값을 표시하므로, 순방향 점프들 및 균일한 순방향 분기들을 상기 설명된 방식으로 실행하는 것은 제어 흐름이 임의의 비활성 스레드들에 대한 재개 포인트들 위로 점프하지 않는다는 것을 보장할 것이다. 이러한 방법으로, MINRC 는 비활성화된 스레드들의 재활성화를 제어하기 위해 재개 카운터들을 이용하는 프로세싱 시스템에서 순방향 점프 및 순방향 분기 명령들을 실행할 때에 적당한 제어 흐름을 보장하기 위하여 이용될 수도 있다.

[0024] 순방향 점프 및 순방향 분기 명령들의 적당한 제어 흐름을 보장하기 위한 MINRC-기반 접근법은 또한, 동일한 목적을 위해 이용될 수도 있는 다른 기법들보다 더욱 효율적일 수도 있다. 예를 들어, 적당한 제어 흐름을 보장하기 위한 또 다른 기법은, 순방향 점프 명령 또는 균일하게 충족된 순방향 분기 명령이 실행될 때마다, 모든 스레드들을 비활성화하는 것과, 현재의 프로그램 카운터 값과 타겟 프로그램 카운터 값 사이의 재개 카운터들 값들을 가지는 임의의 비활성 스레드들이 타겟 명령을 실행하기 이전에 적당하게 비활성화 및 실행되도록, 현재의 프로그램 카운터 값과 타겟 프로그램 카운터 값 사이의 각각의 프로그램 카운터 값을 통해 프로그램 카운터를 순차적으로 증분시키는 것을 포함할 수도 있다. 이러한 접근법은 "최저 값의 어드레스 우선" 스레드 프로세싱 순서를 보장할 수도 있지만, 이러한 접근법은 MINRC-기반 접근법보다 덜 효율적일 수도 있다. 예를 들어, 일부의 경우들에 있어서, 비활성 스레드들의 전부에 대한 재개 카운터 값들은 쟁점이 되고 있는 순방향 점프 또는 분기 명령과 연관된 타겟 프로그램 카운터 값 이상일 수도 있다. 이러한 상황에서 현재의 명령과 타겟 명령 사이의 프로그램 카운터 값들의 각각을 통해 순차적으로 횡단하는 것은, 임의의 스레드의 실행이 발생하지 않는 낭비된, 휴면기의 명령 사이클들로 인해 비효율적일 수도 있다.

[0025] 현재의 명령과 타겟 명령 사이의 프로그램 카운터 값들의 각각을 통해 순차적으로 횡단하기 보다는, MINRC-기반 접근법은 순방향 점프 명령 또는 균일하게 충족된 순방향 분기 명령을 프로세싱하는 것에 응답하여 타겟 프로그램 카운터 값 또는 MINRC 값 중의 어느 하나를 프로그램 카운터로 직접 로딩할 수도 있다. 이것은 현재의 명령의 실행과 다음 명령의 실행 사이에 여분의, 휴면기의 명령 사이클들을 가질 필요 없이 다음 명령 사이클 동안에 다음 명령이 프로세싱되도록 한다. 다음 명령 사이클 동안에 다음 명령이 프로세싱되도록 함으로써, 제어 흐름을 위한 MINRC-기반 접근법은 MINRC 들을 이용하지 않는 재개 카운터-기반 시스템들에 비해, 순방향 점프 및 순방향 분기 명령들의 성능을 개선시킬 수도 있다.

[0026] 그러나, 단일의 글로벌 MINRC 값을 이용하는 하나의 단점은, 이러한 값이 본질적으로 그리고 저절로, 서브루틴들을 포함하는 프로그램들에 대한 적당한 제어 흐름을 보장하기에 타당하지 않을 수도 있다는 점이다. 프로그램이 서브루틴을 포함하는 경우, 글로벌 MINRC 값은 현재 실행되고 있는 서브루틴의 프로그램 공간의 외부에

위치되는 프로그램 카운터 값을 지시할 수도 있다는 것이 가능하다. 예를 들어, 하나 이상의 스레드들은 메인 프로그램 모듈에서 비활성화되었을 수도 있고, MINRC 는 이러한 스레드들에 의해 실행되어야 할 다음 명령에 대응하는 메인 프로그램 공간에서의 프로그램 카운터 값으로 설정되었을 수도 있다. 비활성화된 스레드들이 재활성화되도록 스케줄링되는 메인 프로그램 공간에서의 프로그램 카운터 값에서의 명령을 실행하기 이전에, 나머지 활성 스레드들은 서브루틴을 실행하기 시작할 수도 있다. 순방향 점프 또는 순방향 분기 명령이 서브루틴 동안에 실행되는 경우, MINRC 값은 서브루틴의 프로그램 공간의 외부에 있는 메인 프로그램 공간에서의 프로그램 카운터 값을 여전히 지시할 수도 있다는 것이 가능하다. 이것은 제어 흐름이 서브루틴 외부로, 그리고 복귀 명령 이외의 메인 프로그램으로 다시 점프하게 할 수도 있다.

[0027] 일반적으로, 특화된 호출 및 복귀 명령들은 프로그램의 실행 동안에 메인 프로그램과 서브루틴들 사이, 또는 상이한 서브루틴들 사이에서 제어 흐름을 전달하기 위하여 이용된다. 이 명령들은 하나의 프로그램 모듈 (즉, 메인 프로그램 또는 서브루틴) 에 대한 시스템 상태가 제어를 또 다른 프로그램 모듈로 전달하기 이전에 저장되도록 하고, 다른 프로그램 모듈의 시스템 상태가 그것이 이전에 저장되었을 경우에 복원되도록 한다. 상기 예에서 설명된 바와 같이, 제어 흐름이 서브루틴 외부로, 그리고 호출 명령 또는 복귀 명령 이외의 메인 프로그램으로 다시 점프하게 하는 것은 시스템으로 하여금 부정확한 상태로 동작하게 할 수 있다. 그러므로, 단일의 글로벌 MINRC 가 재개 카운터들을 이용하는 시스템에서 순방향 분기 및 순방향 점프 명령들의 성능을 개선가능할 수도 있지만, 글로벌 MINRC 는 단독으로, 프로그램 내의 실행 서브루틴들을 타당하게 지원가능하지 않을 수도 있다.

[0028] 개시물은 MINRC-기반 제어 흐름 기법들을 이용하는 프로세싱 시스템에서 서브루틴들을 실행하기 위한 기법들을 설명한다. 서브루틴들을 실행하기 위한 기법들은 하나 이상의 프로그램 모듈-특정 MINRC 들을 유지하는 것을 포함할 수도 있다. 각각의 프로그램 모듈-특정 MINRC 는 실행되어야 할 프로그램의 특별한 프로그램 모듈에 대응할 수도 있고, 현재 실행하는 프로그램 모듈의 실행이 개시되었을 때에 활성이었던 모든 스레드들에 대응하는 하나 이상의 재개 카운터 값들의 세트로부터의 가장 작은 재개 카운터 값을 표시할 수도 있다. 메인 프로그램에 대해 그리고 서브루틴들의 각각에 대해 별도의 MINRC 들을 이용함으로써, 프로세싱 시스템은 서브루틴들이 서브루틴으로부터 메인 프로그램으로, 또는 호출 또는 복귀 명령의 실행 이외의 또 다른 서브루틴으로 분기하지 않는다는 것을 보장할 수도 있다. 이러한 방법으로, 서브루틴들의 실행을 위한 적당한 제어 흐름은 순방향 점프 및 순방향 분기 명령들의 실행을 제어하기 위하여 MINRC-기반 접근법을 이용하는 프로세싱 시스템에서 보장될 수도 있다.

[0029] 발산 스레드들을 관리하기 위한 다른 기법들은 동기화 토큰 (synchronization token) 들 및 발산 토큰 (divergence token) 들을 저장하기 위하여 스택 (stack) 을 이용하는 것과, 비활성화된 스레드들의 재활성화를 제어하기 위하여 소프트웨어-트리거링된 스레드 재활성화 시스템을 이용하는 것을 포함할 수도 있다. 예를 들어, 발산을 허용하는 분기 명령이 발생할 때마다, 동기화 토큰은 분기 명령이 최초로 조우되었을 때에 활성이었던 스레드들을 표시하는 스택 상으로 푸시 (push) 될 수도 있다. 분기 명령이 발산 분기 명령인 경우, 시스템은 발산 토큰을, 분기를 취하지 않았던 스레드들과, 분기를 취하지 않았던 스레드들에 대한 다음 명령에 대응하는 프로그램 카운터 값을 표시하는 스택 상으로 푸시할 수도 있다. 스택에서 발산 토큰을 팝핑 (popping) 하도록 시스템에 지시하는 특화된 소프트웨어 플래그 및/또는 소프트웨어 명령이 조우될 때까지, 시스템은 분기를 취하지 않았던 나머지 스레드들을 실행하는 것을 계속할 수도 있다. 스택에서 발산 토큰을 팝핑하는 것에 응답하여, 시스템은 분기를 취하지 않았던 스레드들을 비활성화하며, 분기를 취하지 않았던 스레드들을 재활성화 및 실행하도록 진행할 수도 있다. 스택에서 동기화 토큰을 팝핑하도록 시스템에 지시하는 특화된 소프트웨어 플래그 및/또는 소프트웨어 명령이 조우될 때까지, 시스템은 분기를 취하지 않았던 나머지 스레드들을 실행하는 것을 계속할 수도 있다. 스택에서 동기화 토큰을 팝핑하는 것에 응답하여, 시스템은 스레드 상태가 발산 분기 명령이 최초로 조우되었을 때와 동일하도록 스레드들을 재활성화하도록 진행할 수도 있다.

[0030] 그러나, 이 접근법의 하나의 단점은 스레드들의 재활성화를 제어하기 위하여 특수한 소프트웨어 명령들이 필요하다는 것이다. 또한, 이 접근법은 발산 분기가 발생할 때마다 스택에서 엔트리 (entry) 를 배치하므로, 시스템이 핸들링할 수 있는 내포된 발산 분기 (nested divergent branch) 들의 수는 스택의 사이즈에 기초하여 제한된다. 내포된 발산 분기는 또 다른 발산 분기 명령의 취해진 경로 또는 취해지지 않은 경로 중의 어느 하나의 실행 동안에 발생하는 발산 분기를 지칭할 수도 있다. 즉, 내포된 발산 분기는, 하나 이상의 스레드들이 이전에 발생하는 발산 분기 명령으로 인해 이미 비활성화되었으며 이러한 스레드들이 아직 재활성화되지 않았을 때에 발생하는 발산 분기이다.

[0031] 위에서 설명된 발산 스레드들을 재활성화하기 위한 소프트웨어-트리거링된 스택-기반 접근법들과 대조적으로, 특화된 소프트웨어 명령들은 본 개시물의 재개 카운터-기반 접근법을 구현하기 위하여 반드시 필요하지는 않다.

실제로, 일부의 예들에서는, 재개 검사가 각각의 명령 사이클에서 수행되어, 임의의 비활성화된 스레드들이 그 사이클에 대해 재활성화되도록 스케줄링되는지 여부를 결정할 수도 있다. 이것은 발산 스레드들이 비활성화 및 재활성화되는 방식이 프로그래머 및/또는 컴파일러로부터 효과적으로 은닉되게 할 수도 있고, 프로그래머 및/또는 컴파일러가 발산 스레드들을 프로세싱하도록 설계된 병렬 시스템 및 발산 스레드들을 프로세싱하도록 설계되지 않은 비-병렬 시스템의 양자 상에서 실행될 수 있는 실행가능한 코드의 단일 세트를 생성하도록 할 수도 있다. 추가적으로, 재개 카운터-기반 접근법은 발산 스레드 핸들링을 가능하게 하기 위하여 레거시 코드를 재컴파일 (recompile) 및/또는 재기록할 필요 없이 비-병렬 시스템에 대해 최초로 설계되었던 코드를 실행할 수 있다.

[0032] 또한, 재개 카운터-기반 접근법은 비활성화된 스레드들의 재활성화를 제어하기 위하여 스택과는 반대로 재개 카운터들의 유한 (finite) 세트를 이용하므로, 이러한 접근법이 핸들링할 수 있는 내포된 발산 분기들의 수는 개념적으로 무한하고 스택의 사이즈에 기초하여 제한되지 않는다. 일부의 예들에서, 본 개시물의 기법들은 MINRC 값들을 저장하기 위하여 스택을 이용할 수도 있다는 것에 주목해야 한다. 그러나, MINRC 값들은 발산 분기들 자체에 응답하는 것이 아니라 서브루틴 호출 및 복귀 명령들에 응답하여 이러한 스택 상으로 푸시되고 이러한 스택에서 팝핑된다. 그러므로, MINRC 스택이 본 개시물의 기법들을 구현하기 위하여 이용될 수도 있더라도, 이러한 스택은 이러한 시스템에서 발생할 수도 있는 내포된 발산 분기들의 수를 제한하지 않는다.

[0033] 도 1 은 본 개시물의 서브루틴 실행 기법들을 구현하기 위하여 이용될 수도 있는 일 예의 프로세싱 시스템 (10) 을 예시하는 블록도이다. 프로세싱 시스템 (10) 은 프로그램을 위한 명령들을 병렬 방식으로 실행하도록 구성된다. 프로세싱 시스템 (10) 은 제어 유닛 (12), 프로세싱 엘리먼트들 (14A 내지 14D) (집합적으로 "프로세싱 엘리먼트들 (14)"), 명령 저장소 (16), 데이터 저장소 (18), 및 통신 경로들 (20, 22, 24, 26A 내지 26D) 을 포함한다. 통신 경로들 (26A 내지 26D) 은 "통신 경로들 (26)" 로서 집합적으로 지칭될 수도 있다.

[0034] 프로세싱 시스템 (10) 은 개인용 컴퓨터, 데스크톱 컴퓨터, 랩톱 컴퓨터, 컴퓨터 워크스테이션, 태블릿 컴퓨팅 디바이스, 비디오 게임 플랫폼 또는 콘솔, (예를 들어, 소위 스마트폰, 이동 전화, 셀룰러 전화, 위성 전화, 및/또는 이동 전화 핸드셋과 같은) 무선 통신 디바이스, 랜드라인 (landline) 전화, 인터넷 전화, 휴대용 비디오 게임 디바이스 또는 개인 정보 단말 (personal digital assistant; PDA) 과 같은 핸드헬드 디바이스, 개인용 음악 플레이어, 비디오 플레이어, 디스플레이 디바이스, 텔레비전, 텔레비전 셋톱 박스, 서버, 중간 네트워크 디바이스, 메인프레임 컴퓨터, 그래픽 데이터를 프로세싱 및/또는 디스플레이하는 임의의 다른 타입의 디바이스, 또는 계산들을 수행하는 임의의 타입의 디바이스에서 구현될 수도 있다.

[0035] 일부의 예들에서, 제어 유닛 (12) 및 프로세싱 엘리먼트들 (14) 은 프로그래밍가능한 프로세서 또는 프로그래밍가능한 프로세서의 일부를 형성하는 하드웨어 컴포넌트들일 수도 있다. 예를 들어, 제어 유닛 (12) 및 프로세싱 엘리먼트들 (14) 은 그래픽 프로세싱 유닛 (GPU) 또는 GPU 의 일부를 함께 형성할 수도 있다.

[0036] 일부의 예들에서, 프로세싱 시스템 (10) 은 프로세싱 엘리먼트들 (14) 을 이용하여 프로그램을 위한 복수의 실행 스레드들을 실행하도록 구성되는 단일 명령 다중 데이터 (SIMD) 프로세싱 시스템일 수도 있다. 이러한 SIMD 시스템에서, 프로세싱 엘리먼트들 (14) 은 상이한 데이터 항목들에 대하여 단일 명령을 한 번에 함께 프로세싱할 수도 있다. 프로그램과 연관된 스레드들의 전부가 실행을 완료한 후에 프로그램이 중단될 수도 있다.

[0037] 제어 유닛 (12) 은 명령 저장소 (16) 에 저장된 프로그램을 위한 명령들을 실행하기 위하여 프로세싱 시스템 (10) 을 제어하도록 구성된다. 프로그램의 각각의 명령에 대하여, 제어 유닛 (12) 은 통신 경로 (20) 를 통해 명령 저장소 (16) 로부터 명령을 취출 (retrieve) 할 수도 있고, 그 명령을 프로세싱할 수도 있다. 일부의 예들에서, 제어 유닛 (12) 은 명령과 연관된 동작이 프로세싱 엘리먼트들 (14) 중의 하나 이상 상에서 실행하게 함으로써, 명령을 프로세싱할 수도 있다. 예를 들어, 제어 유닛 (12) 에 의해 취출된 명령은 그 명령에 의해 특정된 데이터 항목들에 대하여 산술 동작 (arithmetic operation) 을 수행하도록 프로세싱 시스템 (10) 에 지시하는 산술 명령일 수도 있고, 제어 유닛 (12) 은 프로세싱 엘리먼트들 (14) 중의 하나 이상으로 하여금, 특정된 데이터 항목들에 대해 산술 동작을 수행하게 할 수도 있다. 추가의 예들에서, 제어 유닛 (12) 은 동작이 프로세싱 엘리먼트들 (14) 상에서 수행되게 하지 않고 명령을 프로세싱할 수도 있다.

[0038] 제어 유닛 (12) 은 명령을 통신 경로 (22) 를 통해 프로세싱 엘리먼트들 (14) 에 제공함으로써, 동작이 프로세싱 엘리먼트들 (14) 의 하나 이상 상에서 수행되게 할 수도 있다. 명령은 프로세싱 엘리먼트들 (14) 에 의

해 수행되어야 할 동작을 특정할 수도 있다. 프로세싱 엘리먼트들 (14) 의 하나 이상에 제공된 명령은 명령 저장소 (16) 로부터 취출된 명령과 동일하거나 이와 상이할 수도 있다. 일부의 예들에서, 제어 유닛 (12) 은, 동작이 수행되어야 하는 프로세싱 엘리먼트들 (14) 의 특별한 서브세트를 활성화하는 것, 및 동작이 수행되지 않아야 하는 프로세싱 엘리먼트들 (14) 의 또 다른 서브세트를 비활성화하는 것 중의 하나 또는 양자 모두에 의해 프로세싱 엘리먼트들 (14) 의 특별한 서브세트 상에서 동작이 수행되게 할 수도 있다. 제어 유닛 (12) 은 각각의 활성화 및/또는 비활성화 신호들을 통신 경로 (22) 를 통해 프로세싱 엘리먼트들 (14) 의 각각에 제공함으로써 프로세싱 엘리먼트들 (14) 을 활성화 및/또는 비활성화할 수도 있다. 일부의 예들에서, 제어 유닛 (12) 은 명령을 프로세싱 엘리먼트들 (14) 에 제공하는 것과 함께, 활성화 및/또는 비활성화 신호들을 프로세싱 엘리먼트들 (14) 에 제공함으로써, 프로세싱 엘리먼트들 (14) 을 활성화 및/또는 비활성화할 수도 있다.

추가의 예들에서, 제어 유닛 (12) 은 명령을 프로세싱 엘리먼트들 (14) 에 제공하기 전에, 프로세싱 엘리먼트들 (14) 을 활성화 및/또는 비활성화할 수도 있다.

[0039] 제어 유닛 (12) 은 프로세싱 엘리먼트들 (14) 을 이용하여 프로그램을 위한 복수의 실행 스레드들을 실행할 수도 있다. 프로세싱 엘리먼트들 (14) 의 각각은 복수의 스레드들의 각각의 스레드에 대한 프로그램의 명령들을 프로세싱하도록 구성될 수도 있다. 예를 들어, 제어 유닛 (12) 은 프로세싱을 위하여 각각의 실행 스레드를 프로세싱 엘리먼트들 (14) 의 개별적인 하나에 배정할 수도 있다. 프로그램을 위한 상이한 실행 스레드들은 데이터 항목들의 세트에서의 상이한 데이터 항목들에 대하여 명령들의 동일한 세트를 실행할 수도 있다.

예를 들어, 프로세싱 엘리먼트 (14A) 는 복수의 데이터 항목들에서의 데이터 항목들의 제 1 서브세트에 대하여 명령 저장소 (16) 에 저장된 프로그램을 위한 제 1 실행 스레드를 실행할 수도 있고, 프로세싱 엘리먼트 (14B) 는 복수의 데이터 항목들에서의 데이터 항목들의 제 2 서브세트에 대하여 명령 저장소 (16) 에 저장된 프로그램을 위한 제 2 실행 스레드를 실행할 수도 있다. 제 1 실행 스레드는 제 2 실행 스레드와 상이할 수도 있고, 데이터 항목들의 제 1 서브세트는 데이터 항목들의 제 2 서브세트와 상이할 수도 있다.

[0040] 일부의 예들에서, 제어 유닛 (12) 은 복수의 실행 스레드들에서의 개별적인 스레드들을 활성화 및 비활성화할 수도 있다. 제어 유닛 (12) 이 스레드를 비활성화할 때, 제어 유닛 (12) 은 또한, 스레드를 실행하도록 배정되는 프로세싱 엘리먼트 (14A 내지 14D) 를 비활성화 및/또는 디스에이블 (disable) 할 수도 있다. 유사하게, 제어 유닛 (12) 이 스레드를 활성화할 때, 제어 유닛 (12) 은 또한, 스레드를 실행하도록 배정되는 프로세싱 엘리먼트 (14A 내지 14D) 를 활성화할 수도 있다. 제어 유닛 (12) 은 본 개시물에서 더 이후에 더욱 상세하게 설명된 바와 같이, 발산 분기 조건들의 핸들링 (handling) 을 보조하기 위하여 하나 이상의 스레드들의 다양한 조합들을 활성화 및 비활성화할 수도 있다.

[0041] 본원에서 이용된 바와 같이, 활성 스레드는 활성화되어 있으며 프로그램의 명령들을 실행하도록 현재 구성되어 있는 스레드를 지칭할 수도 있다. 비활성 스레드는 비활성화되어 있으며 프로그램의 명령들을 실행하지 않도록 현재 구성되어 있는 스레드를 지칭할 수도 있다. 주어진 프로세싱 사이클 동안에 프로세싱 시스템 (10) 에서 실행되는 복수의 스레드들에 대하여, 활성 스레드들의 각각은 프로세싱 사이클 동안에 복수의 스레드들에 대하여 글로벌 프로그램 카운터 레지스터 (global program counter register) 에 의해 식별된 프로그램의 명령을 프로세싱하도록 구성될 수도 있다. 예를 들어, 제어 유닛 (12) 은 프로세싱 사이클 동안에 프로그램의 명령을 프로세싱하도록 이러한 프로세싱 엘리먼트들 (14) 을 구성하기 위하여 활성 스레드들에 배정되는 프로세싱 엘리먼트들 (14) 을 활성화할 수도 있다. 다른 한편으로, 비활성 스레드들의 각각은 프로세싱 사이클 동안에 프로그램의 명령을 프로세싱하지 않도록 구성될 수도 있다. 예를 들어, 제어 유닛 (12) 은 프로세싱 사이클 동안에 프로그램의 명령을 프로세싱하지 않도록 이러한 프로세싱 엘리먼트들 (14) 을 구성하기 위하여 비활성 스레드들에 배정되는 프로세싱 엘리먼트들 (14) 을 비활성화할 수도 있다.

[0042] 일부의 예들에서, 명령 프로세싱 사이클은 프로그램 카운터의 연속적인 부하들 사이의 시간 간격을 지칭할 수도 있다. 예를 들어, 명령 프로세싱 사이클은, 프로그램 카운터가 제 1 명령과 연관된 제 1 값으로 로딩될 때와, 프로그램 카운터가 제 2 명령과 연관된 제 2 값으로 로딩될 때와의 사이의 시간을 지칭할 수도 있다. 제 2 명령은 제 1 명령 직후에 시스템에 의해 프로세싱되는 명령일 수도 있다. 제 1 및 제 2 값들은 동일하거나 상이한 값들일 수도 있고, 제 1 및 제 2 명령들은 동일하거나 상이한 명령들일 수도 있다. 일부의 예들에서, 명령 프로세싱 사이클은 프로그램 카운터의 연속적인 동기식 부하들 사이의 시간 간격을 지칭할 수도 있다. 일부의 예들에서, 프로그램 카운터의 동기식 부하는 클록 신호에 의해 트리거링 (trigger) 되는 부하를 지칭할 수도 있다. 명령 프로세싱 사이클은 명령 사이클 또는 프로세싱 사이클이라고 본원에서 대안적으로 지칭될 수도 있다. 일부의 예들에서, 명령 프로세싱 사이클은 하나 이상의 클록 사이클들에 대응할 수도 있다.

[0043] 때때로, 다음 명령의 프로세싱 전에, 제어 유닛 (12) 은 프로세싱 시스템 (10) 에 의해 프로세싱되어야 할 다음 명령을 결정한다. 제어 유닛 (12) 이 프로세싱되어야 할 다음 명령을 결정하는 방식은 가장 최근에 실행된 명령이 제어 흐름 명령인지 여부에 따라 상이하다. 가장 최근에 실행된 명령이 제어 흐름 명령이 아닌 경우, 제어 유닛 (12) 은 프로세싱 시스템 (10) 에 의해 프로세싱되어야 할 다음 명령이 명령 저장소 (16) 에 저장된 다음의 순차적인 명령에 대응하는 것으로 결정할 수도 있다. 예를 들어, 명령 저장소 (16) 는 프로그램을 위한 명령들을 순서화된 시퀀스 (ordered sequence) 로 저장할 수도 있고, 다음의 순차적인 명령은 명령들의 순서화된 시퀀스로 가장 최근에 실행된 명령 직후에 발생하는 명령일 수도 있다.

[0044] 가장 최근에 실행된 명령이 제어 흐름 명령인 경우, 제어 유닛 (12) 은 제어 흐름 명령에서 특정된 정보에 기초하여 프로세싱 시스템 (10) 에 의해 프로세싱되어야 할 다음 명령을 결정할 수도 있다. 예를 들어, 제어 흐름 명령은 무조건부 제어 흐름 명령 (예를 들어, 무조건부 분기 명령 또는 점프 명령) 일 수도 있고, 이 경우에, 제어 유닛 (12) 은 프로세싱 시스템 (10) 에 의해 프로세싱되어야 할 다음 명령이 제어 흐름 명령에 의해 식별된 타겟 명령인 것으로 결정할 수도 있다. 또 다른 예로서, 제어 흐름 명령은 조건부 제어 흐름 명령 (예를 들어, 조건부 분기 명령) 일 수도 있고, 이 경우에, 제어 유닛 (12) 은 명령 저장소 (16) 로부터, 제어 흐름 명령에 의해 식별된 타겟 명령, 또는 프로세싱하기 위한 다음 명령으로서 명령 저장소 (16) 에 저장된 다음의 순차적인 명령 중의 하나를 선택할 수도 있다.

[0045] 본원에서 이용된 바와 같이, 제어 흐름 명령은 다음의 순차적인 명령을 무조건적으로 선택하는 것 이외의 기법에 기초하여 실행하기 위한 다음 명령을 결정하도록 제어 유닛 (12) 에 지시하는 명령을 지칭할 수도 있다. 제어 흐름 명령은 명령 저장소 (16) 에 저장된 타겟 명령을 특정할 수도 있다. 예를 들어, 제어 흐름 명령은 명령 저장소 (16) 에 저장된 타겟 명령에 대응하는 타겟 프로그램 카운터 값을 표시하는 값을 포함할 수도 있다. 또 다른 예로서, 제어 흐름 명령은 스택 저장 구조에서 복귀 어드레스를 팝핑하도록 제어 유닛 (12) 에 지시함으로써 타겟 명령을 특정할 수도 있다. 복귀 어드레스는 명령 저장소 (16) 에 저장된 타겟 명령에 대응할 수도 있다. 일부의 예들에서, 타겟 명령은 명령 저장소 (16) 에 저장된 다음의 순차적인 명령과는 상이할 수도 있다.

[0046] 하이-레벨 (high-level) 프로그램 코드는 예를 들어, if, switch, do, for, while, continue, break, 및 goto 명령문들과 같은 제어 흐름 명령문들을 포함할 수도 있다. 컴파일러는 하이-레벨 제어 흐름 명령문들을 로우-레벨 (low-level), 예를 들어, 머신-레벨 (machine-level) 제어 흐름 명령들로 변환할 수도 있다. 제어 흐름 명령이 아닌 명령은 본원에서 순차적인 명령으로서 지칭될 수도 있다. 순차적인 명령은 제어 유닛 (12) 이 실행하기 위한 다음 명령인 것으로 다음의 순차적인 명령을 반드시 선택하는 명령을 지칭할 수도 있다. 일부의 예들에서, 순차적인 명령은 타겟 명령을 식별하는 정보를 포함하지 않을 수도 있다.

[0047] 제어 흐름 명령들에 대하여, 타겟 명령을 식별하는 정보는 명령 저장소 (16) 에 저장된 타겟 명령을 표시하는 값일 수도 있다. 일부의 예들에서, 타겟 명령을 표시하는 값은 명령 저장소 (16) 에 저장된 타겟 명령에 대응하는 명령 어드레스를 표시하는 값일 수도 있다. 일부의 경우들에 있어서, 명령 어드레스를 표시하는 값은 명령 저장소 (16) 에 저장된 타겟 명령의 어드레스일 수도 있다. 추가적인 경우들에 있어서, 명령 어드레스를 표시하는 값은 타겟 명령의 어드레스를 계산하기 위하여 이용된 값일 수도 있다. 추가의 예들에서, 명령 어드레스를 표시하는 값은 타겟 명령에 대응하는 타겟 프로그램 카운터 값을 표시하는 값일 수도 있다. 일부의 경우들에 있어서, 타겟 프로그램 카운터 값을 표시하는 값은 타겟 명령에 대응하는 타겟 프로그램 카운터 값일 수도 있다. 추가적인 경우들에 있어서, 타겟 프로그램 카운터 값을 표시하는 값은 타겟 프로그램 카운터 값을 계산하기 위하여 이용된 값일 수도 있다. 일부의 예들에서는, 타겟 명령에 대응하는 타겟 프로그램 카운터 값이 명령 저장소 (16) 에 저장된 타겟 명령의 어드레스와 동일할 수도 있다.

[0048] 제어 흐름 명령은 순방향 제어 흐름 명령 또는 역방향 제어 흐름 명령일 수도 있다. 제어 흐름 명령이 순방향인지 또는 역방향인지 여부의 속성은 제어 흐름 명령의 방향으로서 지칭될 수도 있다. 순방향 제어 흐름 명령은, 타겟 명령이 명령 저장소 (16) 에 저장된 명령들의 순서화된 시퀀스에서 제어 흐름 명령 후에 발생하는 제어 흐름 명령일 수도 있다. 역방향 제어 흐름 명령은, 타겟 명령이 명령 저장소 (16) 에 저장된 명령들의 순서화된 시퀀스에서 다음의 순차적인 명령 이전에 발생하는 제어 흐름 명령일 수도 있다. 다음의 순차적인 명령은 명령들의 순서화된 시퀀스에서 제어 흐름 명령 직후에 발생할 수도 있다.

[0049] 제어 흐름 명령은 조건부 제어 흐름 명령 또는 무조건부 제어 흐름 명령일 수도 있다. 조건부 제어 흐름 명령은 제어 흐름 명령과 연관된 타겟 명령으로 점프하기 위한 조건을 특정하는 정보를 포함한다. 조건부 제어 흐름 명령을 프로세싱할 때, 제어 유닛 (12) 이 조건이 충족되는 것으로 결정하는 경우에는, 제어 유닛 (12)

은 프로세싱되어야 할 다음 명령이 타겟 명령인 것으로 결정할 수도 있다. 다른 한편으로, 제어 유닛 (12) 이 조건이 충족되지 않는 것으로 결정하는 경우에는, 제어 유닛 (12) 은 프로세싱되어야 할 다음 명령이 명령 저장소 (16) 에 저장된 다음의 순차적인 명령인 것으로 결정할 수도 있다. 무조건부 제어 흐름 명령은 제어 흐름 명령과 연관된 타겟 명령으로 점프하기 위한 조건을 특정하는 정보를 포함하지 않는다. 무조건부 제어 흐름 명령을 프로세싱할 때, 제어 유닛 (12) 은 프로세싱하기 위한 다음 명령이 제어 흐름 명령에 의해 식별된 타겟 명령인 것으로 무조건적으로 결정할 수도 있다. 다시 말해서, 이러한 경우에 있어서의 결정은 무조건부 제어 흐름 명령 자체에서 특정된 임의의 조건에 따라 좌우되지 않는다.

[0050] 조건부 제어 흐름 명령의 일 예는 조건부 분기 명령을 포함한다. 본 개시물에서의 일반 용어 분기 명령의 이용은, 분기 명령이 이와 다르게 무조건부 분기 명령으로서 지정되지 않으면, 조건부 분기 명령을 전형적으로 지칭한다. 무조건부 제어 흐름 명령들의 예들은 점프 명령들, 호출 명령들, 및 복귀 명령들을 포함한다.

[0051] 조건부 분기 명령은 하나 이상의 데이터 항목 값들에 대해 특정되는 조건들을 포함할 수도 있다. 예를 들어, 조건의 하나의 타입은 프로세싱 시스템 (10) 에서 실행되는 각각의 활성 스레드에 대해 제 1 데이터 항목 값을 제 2 데이터 항목 값과 비교하는 비교 조건일 수도 있다. 데이터 항목 값들을 비교하는 것은 예를 들어, 제 1 데이터 항목 값이 제 2 데이터 항목 값보다 더 큰지, 더 작은지, 더 크지 않은지, 더 작지 않은지, 이와 동일한지, 또는 이와 동일하지 않은지 여부를 결정하는 것을 포함할 수도 있다. 또 다른 타입의 조건은 프로세싱 시스템 (10) 에서 실행되는 각각의 활성 스레드에 대한 데이터 항목 값이 제로와 동일한지 또는 이와 동일하지 않은지 여부를 결정하는 제로 검사 조건일 수도 있다. 프로세싱 엘리먼트들 (14) 의 각각은 상이한 데이터 항목들에 대해 동작하므로, 조건을 평가하는 결과는 프로세싱 시스템 (10) 에서 실행되는 각각의 활성 스레드에 대해 상이할 수도 있다. 프로세싱 시스템 (10) 에서 실행되는 활성 스레드들의 전부가 분기 조건을 충족하거나, 또는 프로세싱 시스템 (10) 에서 실행되는 활성 스레드들의 전부가 분기 조건을 충족하지 않는 경우, 균일한 분기 조건이 발생하고, 분기 명령에 대한 분기 발산이 균일하다고 말해진다. 다른 한편으로, 프로세싱 시스템 (10) 에서 실행되는 활성 스레드들의 적어도 하나가 분기 조건을 충족하고, 프로세싱 시스템 (10) 에서 실행되는 활성 스레드들의 적어도 하나가 분기 조건을 충족하지 않는 경우, 발산 분기 조건이 발생하고, 분기 명령에 대한 분기 발산이 발산한다고 말해진다.

[0052] 프로세싱 시스템 (10) 에서 실행되는 스레드들은 동일한 명령을 록스텝 방식으로 실행할 수도 있다. 다시 말해서, 프로세싱 엘리먼트들 (14) 의 각각은 프로세싱 사이클 동안에 모든 활성 스레드들에 대해 동일한 명령을 함께 실행할 수도 있다. 그러나, 발산 분기 조건이 발생할 때, 그 분기 조건을 충족하는 스레드들은 분기 조건을 충족하지 않는 스레드들에 의해 실행되도록 스케줄링된 다음 명령들과는 상이한 다음의 명령들을 실행하도록 스케줄링될 수도 있다. 이것은 프로세싱 시스템 (10) 에서의 스레드들이 단일 명령을 록스텝 방식으로 실행하는 것을 방해할 수도 있다.

[0053] 발산 분기 조건을 처리하기 위하여, 일부의 예들에서, 제어 유닛 (12) 은 나머지 활성 스레드들이 동일한 프로그램 카운터 어드레스에 모두 동기화되도록, 분기 조건을 충족하였거나 이를 충족하지 않았던 스레드들의 하나의 서브세트를 비활성화할 수도 있다. 스레드들의 재활성화를 제어하기 위하여, 제어 유닛 (12) 은, 프로세싱 시스템에서 실행되는 각각의 스레드에 대한 재개 카운터를 할당하는 것, "최저 값의 어드레스 우선" 스레드 프로세싱 순서에 따라 발산 스레드들을 프로세싱하는 것, 및 매 명령의 실행 이전에 재개 검사 동작을 수행하는 것을 포함하는 재개 카운터-기반 접근법을 이용할 수도 있다.

[0054] 더욱 구체적으로, 제어 유닛 (12) 은 발산 분기 조건에 응답하여 하나 이상의 스레드들을 비활성화할 수도 있고, 비활성화되고 있는 각각의 스레드에 대하여, 각각의 스레드에 대한 재개 카운터 (예를 들어, 레지스터) 를, 각각의 스레드가 재활성화되도록 스케줄링되는 프로그램 카운터 값을 표시하는 값으로 설정할 수도 있다.

일부의 예들에서, 각각의 스레드가 재활성화되도록 스케줄링되는 프로그램 카운터 값은 비활성화된 스레드에 의해 실행되어야 할 다음 명령에 대응하는 프로그램 카운터 값일 수도 있다. 스레드가 활성인 경우, 스레드에 대한 재개 카운터는, 프로그램의 어드레스 범위보다 더 큰 값 (예를 들어, 최대 레지스터 값) 에 대응할 수도 있는 디폴트 값으로 설정될 수도 있다. 프로그램 카운터 레지스터가 새로운 프로그램 카운터 값으로 로딩될 때마다, 제어 유닛 (12) 은 재개 검사 동작을 수행할 수도 있고, 이것은 스레드에 대한 재개 카운터 값이 새로운 프로그램 카운터 값과 동일한 임의의 스레드들을 재활성화할 수도 있다. 일부의 예들에서, 비활성화된 스레드들 중의 임의의 것이 명령을 실행하기 이전에 재활성화되도록 스케줄링되는지를 결정하기 위하여, 재개 검사 동작은 각각의 비활성화된 스레드와 연관된 재개 카운터 값을 새롭게 로딩된 프로그램 카운터 값과 비교할 수도 있다.

[0055] "최저 값의 어드레스 우선" 스레드 프로세싱 순서는 분기 명령의 방향 (즉, 순방향 또는 역방향) 에 기초하여 어느 스레드들이 발산 분기 명령에 응답하여 비활성화되는지를 구별할 수도 있다. 발산 역방향 분기 명령에 대하여, 제어 유닛 (12) 은 분기 조건이 충족되지 않는 스레드들을 비활성화할 수도 있고, 비활성화되고 있는 각각의 스레드에 대한 재개 카운터 값을, 분기 명령 후에 발생하는 다음의 순차적인 명령과 연관된 값으로 설정할 수도 있고, 분기 명령에 의해 특정된 타겟 명령과 연관된 값으로 프로그램 카운터를 로딩할 수도 있고, 분기 조건이 충족되는 그러한 스레드들을 실행하도록 진행할 수도 있다. 발산 순방향 분기 명령에 대하여, 제어 유닛 (12) 은 분기 조건이 충족되는 스레드들을 비활성화할 수도 있고, 비활성화되고 있는 각각의 스레드에 대한 재개 카운터 값을, 분기 명령에 의해 특정된 타겟 명령과 연관된 값으로 설정할 수도 있고, 분기 명령 후에 발생하는 다음의 순차적인 명령과 연관된 값으로 프로그램 카운터를 로딩할 수도 있고, 분기 조건이 충족되지 않는 그러한 스레드들을 실행하도록 진행할 수도 있다. 스레드들을 이러한 방식으로 비활성화하는 것은, 더 낮은 값의 어드레스들에서의 명령들을 프로세싱하도록 스케줄링되는 발산 스레드들이 더 높은 값의 어드레스들에서의 명령들을 프로세싱하도록 스케줄링되는 스레드들 이전에 실행되는 것을 보장한다 (즉, "최저 값의 어드레스 우선" 스레드 프로세싱 순서). 이러한 프로세싱 순서는 제어 흐름이 이러한 스레드들을 재활성화하지 않고, 그리고 프로그램을 조기에 종료하지 않고 비활성 스레드들에 대한 임의의 재개 포인트들 위로 점프하는 것을 방지할 수도 있다. 다시 말해서, 이러한 프로세싱 순서는, 모든 스레드들이 활성일 것이고 최종 프로그램 명령문이 실행을 종료한 시간까지 프로세싱을 완료하였을 것이라는 것을 보장한다.

[0056] 본 개시물에 따르면, 제어 유닛 (12) 은 프로그램과 연관된 제 1 MINRC 에 기초하여 프로그램의 실행을 제어할 수도 있고, 서브루틴과 연관된 제 2 MINRC 에 기초하여 프로그램의 서브루틴의 실행을 제어할 수도 있다. 제 1 MINRC 는 프로세싱 시스템 (10) 에서 실행되는 복수의 스레드들과 연관된 복수의 재개 카운터 값들 중의 가장 작은 재개 카운터 값을 표시하는 값을 특정할 수도 있다. 제 2 MINRC 는, 서브루틴의 실행이 개시될 때에 활성인 스레드들의 전부에 대응하는 복수의 재개 카운터 값들의 서브세트로부터의 가장 작은 재개 카운터 값을 표시하는 값을 특정할 수도 있다. 일부의 경우들에 있어서, 제 1 MINRC 에 대한 복수의 재개 카운터 값들은 프로그램의 실행이 개시될 때에 활성인 스레드들의 전부에 대응할 수도 있다. 일부의 예들에서, 프로그램이 최상위-레벨 (top-level) 프로그램 (예를 들어, 메인 프로그램) 일 때, 모든 스레드들은 프로그램이 개시될 때에 활성일 수도 있다. 이러한 예들에서, 제 1 MINRC 는 프로세싱 시스템 (10) 에서 실행되는 모든 스레드들과 연관된 재개 카운터 값들의 세트 중의 가장 작은 재개 카운터 값일 수도 있다.

[0057] 일반적으로, 제어 유닛 (12) 은 복수의 MINRC 들에 기초하여 프로세싱 시스템 (10) 의 실행을 제어할 수도 있다. 각각의 MINRC 는 전체 프로그램 내의 특별한 프로그램 모듈을 제어하기 위하여 이용될 수도 있다. 본원에서 이용된 바와 같이, 프로그램 모듈은 메인 프로그램 모듈 (즉, 프로그램의 실행이 개시될 때에 프로그램이 초기에 실행하는 프로그램 모듈), 및/또는 메인 프로그램 모듈에 의해 또는 또 다른 서브루틴 프로그램 모듈에 의해 호출되는 서브루틴 프로그램 모듈을 지칭할 수도 있다. 일부의 예들에서, 제어 유닛 (12) 은 실행되고 있는 현재의 프로그램 모듈에 대한 MINRC 값을 유지할 수도 있고, 현재의 프로그램 모듈에 대응하는 MINRC 값에 기초하여 현재의 프로그램 모듈의 실행을 제어할 수도 있다. MINRC 값은 현재의 프로그램 모듈의 실행이 개시될 때에 활성인 모든 스레드들에 대응하는 하나 이상의 재개 카운터 값들의 세트로부터의 가장 작은 재개 카운터 값을 표시할 수도 있다.

[0058] 제어 유닛 (12) 이 상이한 프로그램 모듈을 실행하는 것으로 전환할 때, 제어 유닛 (12) 은 프로세싱 시스템 (10) 의 실행을 제어하기 위하여 이용되는 MINRC 를 전환할 수도 있다. 예를 들어, 제어 유닛 (12) 이 호출자 프로그램 모듈을 실행하는 것으로부터 서브루틴 프로그램 모듈을 실행하는 것으로 전환하는 경우, 제어 유닛 (12) 은 프로세싱 시스템 (10) 의 실행을 제어하기 위하여 이용되는 MINRC 를, 호출자 프로그램 모듈과 연관된 제 1 MINRC 로부터 서브루틴 모듈과 연관된 제 2 MINRC 로 전환할 수도 있다. 유사하게, 서브루틴 프로그램 모듈의 실행을 완료한 후, 제어 유닛 (12) 은 프로세싱 시스템 (10) 의 실행을 제어하기 위하여 이용되는 MINRC 를, 서브루틴 모듈과 연관된 제 2 MINRC 로부터 호출자 프로그램 모듈과 연관된 제 1 MINRC 값으로 전환할 수도 있다. 호출자 프로그램 모듈은 메인 프로그램 모듈 또는 서브루틴 프로그램 모듈일 수도 있다.

[0059] 일부의 예들에서, 특별한 MINRC 에 기초하여 프로그램 모듈의 실행을 제어하는 것은 그 특별한 MINRC 에 기초하여 프로그램 모듈에 포함된 순방향 제어 흐름 명령들의 실행을 제어하는 것을 포함할 수도 있다. 예를 들어, 분기 조건이 균일하게 충족되는 순방향 점프 명령 또는 순방향 조건부 분기 명령을 실행하는 것에 응답하여, 제어 유닛 (12) 은 MINRC 값에 기초하여 실행하기 위한 다음 명령을 결정할 수도 있다. 제어 유닛 (12) 은 예를 들어, 분기 또는 점프 명령에 의해 특정된 타겟 프로그램 카운터 값을 MINRC 값과 비교할 수도 있고, 프로그램 카운터로 로딩하기 위한 타겟 프로그램 카운터 값 또는 MINRC 값 중의 어느 하나를 선택할 수도 있다.

하나의 예에서, 타겟 프로그램 카운터 값이 MINRC 값 이하일 때, 제어 유닛 (12) 은 프로그램 카운터로 로딩하기 위한 값으로서 타겟 프로그램 카운터 값을 선택할 수도 있다. 이러한 예에서, 타겟 프로그램 카운터 값이 MINRC 값 이하가 아닐 때, 제어 유닛 (12) 은 프로그램 카운터로 로딩하기 위한 값으로서 MINRC 값을 선택할 수도 있다. 타겟 프로그램 카운터 값 또는 MINRC 값 중의 어느 하나를 프로그램 카운터로 로딩하는 것은 프로세싱 시스템 (10) 이 실행이 전혀 발생하지 않는 값들을 통해 프로그램 카운터를 증분시키는 것으로 인해 프로그램 사이클들을 낭비하는 것을 방지한다. 또한, MINRC 값이 모든 비활성 스레드들 중의 가장 작은 재개 카운터 값을 표시하므로, 순방향 점프들 및 균일한 순방향 분기들을 상기 설명된 방식으로 실행하는 것은 제어 흐름이 임의의 비활성 스레드들의 재개 포인트들 위로 점프하지 않는다는 것을 보장할 것이다. 이러한 방법으로, 하나 이상의 MINRC 값들에 기초하여 프로세싱 시스템 (10) 의 실행을 제어하는 것은 발산 스레드 핸들링을 위하여 재개 카운터들을 활용하는 시스템에서 순방향 점프 명령들 및 순방향 분기 명령들의 성능을 개선시킬 수도 있다.

[0060] 본 개시물의 일부의 양태들에 따르면, 제어 유닛 (12) 은 서브루틴의 진입 및/또는 진출 시에 프로세싱 시스템 (10) 의 실행을 제어하기 위하여 이용되는 MINRC 를 전환하도록 구성될 수도 있다. 예를 들어, 호출 명령을 실행하는 것에 응답하여, 제어 유닛 (12) 은 호출자 프로그램에 대응하는 제 1 MINRC 의 상태를 저장하고, 서브루틴 프로그램에 대응하는 제 2 MINRC 를 초기화하고, 서브루틴 프로그램에 대응하는 제 2 MINRC 에 기초하여 서브루틴 프로그램의 실행을 제어하도록 구성될 수도 있다. 복귀 명령을 실행하는 것에 응답하여, 프로세싱 시스템은 호출자 프로그램에 대응하는 제 1 MINRC 의 저장된 상태를 복원하고, 제 1 MINRC 의 복원된 상태에 기초하여 메인 프로그램의 실행을 제어하는 것을 재개하도록 구성될 수도 있다.

[0061] 일부의 예들에서, 제어 유닛 (12) 은 제 1 MINRC 에 대해 MINRC 레지스터에 저장된 값을 스택 저장 구조 상으로 푸시함으로써 적어도 부분적으로 호출자 프로그램에 대응하는 제 1 MINRC 의 상태를 저장할 수도 있다. 이러한 예들에서, 제어 유닛 (12) 은 스택 저장 구조로부터 제 1 MINRC 의 저장된 상태를 팝핑함으로써, 그리고 제 1 MINRC 의 저장된 상태에 대응하는 값으로 MINRC 레지스터에 저장된 값을 겹쳐쓰기 (overwrite) 함으로써 적어도 부분적으로 호출자 프로그램에 대응하는 MINRC 의 저장된 상태를 복원할 수도 있다. 호출 명령을 실행할 때, 제어 유닛 (12) 은 제 2 MINRC 를 디폴트 값 (예를 들어, 최대 레지스터 값 또는 프로그램의 어드레스 범위보다 더 큰 값) 과 동일하게 설정함으로써 적어도 부분적으로 서브루틴 프로그램에 대응하는 제 2 MINRC 를 초기화할 수도 있다.

[0062] 프로세싱 시스템 (10) 에서의 재개 카운터 값들 중의 임의의 것이 새로운 값으로 설정될 때, 제어 유닛 (12) 은 새로운 가장 작은 재개 카운터 값을 반영하기 위하여 MINRC 값을 업데이트할 수도 있다. 제어 유닛 (12) 은 예를 들어, 하나 이상의 스레드들을 비활성화하는 것에 응답하여, 및/또는 재개 검사 동작을 수행하는 것에 응답하여 MINRC 값을 업데이트할 수도 있다.

[0063] 본 개시물의 일부의 양태들에 따르면, MINRC 값을 업데이트할 때, 제어 유닛 (12) 은 서브루틴의 실행이 개시되었을 때에 비활성이었던 스레드들과 연관되는 재개 카운터 값들을, 서브루틴과 연관된 MINRC 를 업데이트하기 위하여 이용되는 것으로부터 배제하기 위하여 다양한 기법들을 이용할 수도 있다. MINRC 를 업데이트하는 동안의 고려사항으로부터 이러한 재개 카운터 값들을 배제함으로써, 본 개시물의 기법들은 서브루틴-특정 MINRC 가 서브루틴과 연관된 프로그램 공간 내에 있는 값들로 업데이트되는 것을 보장할 수도 있다.

[0064] 일부의 예들에서, 제어 유닛 (12) 은 서브루틴의 진입 포인트 이상인 재개 카운터 값들을, MINRC 업데이트 동안에 결과적인 MINRC 값에 영향을 주는 것으로부터 배제할 수도 있다. 추가의 예들에서, 제어 유닛 (12) 은 플래그들의 세트에서의 각각의 플래그가 서브루틴의 실행이 개시되었을 때에 각각의 스레드가 활성이었는지 여부를 표시하는 플래그들의 세트를 유지할 수도 있다. 이러한 예들에서, 제어 유닛 (12) 은 서브루틴의 실행이 개시되었을 때에 활성이 아니었던 재개 카운터 값들을, MINRC 업데이트 동안에 결과적인 MINRC 값에 영향을 주는 것으로부터 배제할 수도 있다.

[0065] 제어 유닛 (12) 은 통신 경로 (20) 를 통해 명령 저장소 (16) 에, 통신 경로 (22) 를 통해 프로세싱 엘리먼트들 (14) 에, 그리고 통신 경로 (24) 를 통해 데이터 저장소 (18) 에 통신가능하게 결합된다. 제어 유닛 (12) 은 판독 명령들을 명령 저장소 (16) 로 전송하기 위하여 통신 경로 (20) 를 이용할 수도 있다. 판독 명령은 명령이 취출되어야 하는 명령 저장소 (16) 에서의 명령 어드레스를 특정할 수도 있다. 제어 유닛 (12) 은 판독 명령을 전송하는 것에 응답하여 명령 저장소 (16) 로부터 하나 이상의 프로그램 명령들을 수신할 수도 있다. 제어 유닛 (12) 은 명령들을 프로세싱 엘리먼트들 (14) 에 제공하기 위하여, 그리고 일부의 예들에서, 프로세싱 엘리먼트들 (14) 로부터 데이터 (예를 들어, 분기 조건을 평가하기 위한 비교 명령의 결과) 수신하기

위하여, 통신 경로 (22) 를 이용할 수도 있다. 일부의 예들에서, 제어 유닛 (12) 은 (예를 들어, 분기 조건을 평가하기 위하여) 데이터 저장소 (18) 로부터 직접 데이터 항목 값들을 취출하기 위하여 통신 경로 (24) 를 이용할 수도 있다. 도 1 은 통신 경로 (24) 를 포함하는 것으로 프로세싱 시스템 (10) 을 예시하지만, 다른 예들에서는, 프로세싱 시스템 (10) 이 통신 경로 (24) 를 포함하지 않을 수도 있다.

[0066] 프로세싱 엘리먼트들 (14) 의 각각은 명령 저장소 (16) 에 저장된 프로그램을 위한 명령들을 프로세싱함에 있어서 프로세싱 시스템 (10) 을 보조하기 위한 동작들을 수행하도록 구성될 수도 있다. 일부의 예들에서, 프로세싱 엘리먼트들 (14) 의 각각은 동작들의 동일한 세트를 수행하도록 구성될 수도 있다. 예를 들어, 프로세싱 엘리먼트들 (14) 의 각각은 동일한 명령 세트 아키텍처 (ISA) 를 구현할 수도 있다. 추가적인 예들에서, 프로세싱 엘리먼트들 (14) 의 각각은 산술 논리 유닛 (arithmetic logic unit; ALU) 일 수도 있다. 추가의 예들에서, 프로세싱 시스템 (10) 은 벡터 프로세서 (예를 들어, 그래픽 프로세싱 유닛 (graphics processing unit; GPU) 벡터 프로세서) 일 수도 있고, 프로세싱 엘리먼트들 (14) 의 각각은 벡터 프로세서 내의 프로세싱 엘리먼트일 수도 있다. 추가적인 예들에서, 프로세싱 시스템 (10) 은 SIMD 실행 유닛일 수도 있고, 프로세싱 엘리먼트들 (14) 의 각각은 SIMD 실행 유닛 내의 SIMD 프로세싱 엘리먼트일 수도 있다.

[0067] 프로세싱 엘리먼트들 (14) 에 의해 수행된 동작들은 산술 동작들, 논리 동작들, 비교 동작들 등을 포함할 수도 있다. 산술 동작들은 예를 들어, 가산 동작, 감산 동작, 승산 동작, 제산 동작, 등과 같은 동작들을 포함할 수도 있다. 산술 동작들은 또한, 예를 들어, 정수 산술 동작들 및/또는 부동 소수점 (floating-point) 산술 동작들을 포함할 수도 있다. 논리 동작들은 예를 들어, 비트별 AND 동작, 비트별 OR 동작, 비트별 XOR 동작, 등과 같은 동작들을 포함할 수도 있다. 비교 동작들은 예를 들어, 보다 더 큰 동작, 보다 더 작은 동작, 제로와 동일한 동작, 제로와 동일하지 않은 동작, 등과 같은 동작들을 포함할 수도 있다. 보다 더 큰 그리고 보다 더 작은 동작들은 제 1 데이터 항목이 제 2 데이터 항목보다 더 큰지 또는 더 작은지 여부를 결정할 수도 있다. 제로와 동일한 그리고 제로와 동일하지 않은 동작들은 데이터 항목이 제로와 동일한지 또는 동일하지 않은지 여부를 결정할 수도 있다. 동작들을 위해 이용된 피연산자 (operand) 들은 데이터 저장소 (18) 에 포함된 레지스터들에 저장될 수도 있다.

[0068] 프로세싱 엘리먼트들 (14) 의 각각은 통신 경로 (22) 를 통해 제어 유닛 (12) 으로부터 명령을 수신하는 것에 응답하여 동작을 수행하도록 구성될 수도 있다. 일부의 예들에서, 프로세싱 엘리먼트들 (14) 의 각각은 다른 프로세싱 엘리먼트들 (14) 에 독립적으로 활성화 및/또는 비활성화되도록 구성될 수도 있다. 이러한 예들에서, 프로세싱 엘리먼트들 (14) 의 각각은, 각각의 프로세싱 엘리먼트 (14A 내지 14D) 가 활성화될 때에 제어 유닛 (12) 으로부터 명령을 수신하는 것에 응답하여 동작을 수행하고, 각각의 프로세싱 엘리먼트 (14A 내지 14D) 가 비활성화 (즉, 활성화되지 않음) 될 때에 제어 유닛 (12) 으로부터 명령을 수신하는 것에 응답하여 동작을 수행하지 않도록 구성될 수도 있다.

[0069] 프로세싱 엘리먼트들 (14A 내지 14D) 의 각각은 각각의 통신 경로 (26A 내지 26D) 를 통해 데이터 저장소 (18) 에 통신가능하게 결합될 수도 있다. 프로세싱 엘리먼트들 (14) 은 데이터 저장소 (18) 로부터 데이터를 취출하고 통신 경로들 (26) 을 통해 데이터를 데이터 저장소 (18) 에 저장하도록 구성될 수도 있다. 일부의 예들에서, 데이터 저장소 (18) 로부터 취출된 데이터는 프로세싱 엘리먼트들 (14) 에 의해 수행된 동작들을 위한 피연산자들일 수도 있다. 일부의 예들에서, 데이터 저장소 (18) 에 저장된 데이터는 프로세싱 엘리먼트들 (14) 에 의해 수행된 동작들의 결과들일 수도 있다.

[0070] 명령 저장소 (16) 는 프로세싱 시스템 (10) 에 의한 실행을 위한 프로그램을 저장하도록 구성된다. 프로그램은 명령들의 순서화된 시퀀스로서 저장될 수도 있다. 일부의 예들에서, 각각의 명령은 고유의 명령 어드레스에 의해 어드레싱될 수도 있다. 이러한 예들에서, 명령들의 시퀀스에서의 더 이후의 명령들을 위한 명령 어드레스들은 명령들의 시퀀스에서의 더 이전의 명령들을 위한 명령 어드레스들보다 더 크다. 일부의 예들에서, 프로그램 명령들은 머신-레벨 (machine-level) 명령들일 수도 있다. 즉, 이러한 예들에서는, 명령들이 프로세싱 시스템 (10) 의 ISA 에 대응하는 포맷일 수도 있다. 명령 저장소 (16) 는 통신 경로 (20) 를 통해 제어 유닛 (12) 으로부터 판독 명령을 수신하도록 구성된다. 판독 명령은 명령이 취출되어야 하는 명령 어드레스를 특정할 수도 있다. 판독 명령을 수신하는 것에 응답하여, 명령 저장소 (16) 는 통신 경로 (20) 를 통해 판독 명령에서 특정된 명령 어드레스에 대응하는 명령을 제어 유닛 (12) 에 제공할 수도 있다.

[0071] 명령 저장소 (16) 는 임의의 타입의 메모리, 캐시 (cache) 또는 그 조합일 수도 있다. 명령 저장소 (16) 가 캐시일 때, 명령 저장소 (16) 는 프로세싱 시스템 (10) 외부의 프로그램 메모리에 저장되는 프로그램과 연관된 명령들을 캐시할 수도 있다. 명령 저장소 (16) 는 프로세싱 시스템 (10) 내에 있는 것으로 예시되어

있지만, 다른 예들에서는, 명령 저장소 (16) 가 프로세싱 시스템 (10) 의 외부에 있을 수도 있다.

[0072] 데이터 저장소 (18) 는 프로세싱 엘리먼트들 (14) 에 의해 이용된 데이터 항목들을 저장하도록 구성된다. 일부의 예들에서, 데이터 저장소 (18) 는 복수의 레지스터들을 포함할 수도 있고, 각각의 레지스터는 프로세싱 시스템 (10) 에 의해 동작되는 복수의 데이터 항목들 내의 각각의 데이터 항목을 저장하도록 구성된다. 데이터 저장소 (18) 는 데이터 저장소 (18) 에서의 레지스터들과 메모리 또는 캐시 (도시되지 않음) 사이에서 데이터를 전달하도록 구성되는 하나 이상의 통신 경로들 (도시되지 않음) 에 결합될 수도 있다.

[0073] 통신 경로들 (20, 22, 24, 26) 은 도 1 에서 예시된 바와 같이, 프로세싱 시스템 (10) 에서의 상이한 컴포넌트들 사이에서의 신호들, 명령들 및/또는 데이터의 통신을 제공하도록 구성될 수도 있다. 통신 경로들 (20, 22, 24, 26) 은 예를 들어, 도 1 에 도시된 상이한 컴포넌트들 사이에서 전기적 신호들을 반송하는 하나 이상의 버스들 (예를 들어, 온-칩 (on-chip) 버스들) 및/또는 전기적 상호접속부 (interconnect) 들 (예를 들어, 와이어들 및/또는 회로 트레이스들) 로서 각각 구현될 수도 있다.

[0074] 도 1 은 프로세싱 엘리먼트들 (14) 에 의해 이용된 데이터를 저장하기 위한 단일 데이터 저장소 (18) 를 예시하지만, 다른 예들에서는, 프로세싱 시스템 (10) 이 프로세싱 엘리먼트들 (14) 의 각각에 대한 별도의 전용 데이터 저장소들을 포함할 수도 있다. 도 1 은 예시적인 목적들을 위하여 4 개의 프로세싱 엘리먼트들 (14) 을 갖는 프로세싱 시스템 (10) 을 예시한다. 그러나, 다른 예들에서는, 프로세싱 시스템 (10) 이 동일하거나 상이한 구성인 동일하거나 상이한 수의 프로세싱 엘리먼트들 (14) 을 가질 수도 있다.

[0075] 도 2 는 본 개시물에 따라 도 1 의 일 예의 프로세싱 시스템 (10) 에서의 제어 유닛 (12) 을 더욱 상세하게 예시하는 블록도이다. 제어 유닛 (12) 은 프로그램 카운터 (28), 페치 (fetch) 모듈 (30), 디코드 모듈 (32) 및 제어 흐름 모듈 (34) 을 포함한다. 제어 흐름 모듈 (34) 은 대안적으로 제어 흐름 유닛으로서 본원에서 지칭될 수도 있다.

[0076] 프로그램 카운터 (28) 는 프로그램 카운터 값을 저장하도록 구성된다. 일부의 예들에서, 프로그램 카운터 (28) 는 예를 들어, 프로그램 카운터 레지스터와 같은 하드웨어 레지스터일 수도 있다. 프로그램 카운터 값은 명령 저장소 (16) 에 저장된 명령을 표시할 수도 있다. 일부의 경우들에 있어서, 프로그램 카운터 값은 명령 저장소 (16) 에 저장된 명령의 명령 어드레스와 동일할 수도 있다. 추가적인 경우들에 있어서, 프로그램 카운터 값은 명령 저장소 (16) 에 저장된 명령의 명령 어드레스를 컴퓨팅하기 위하여 이용될 수도 있다. 예를 들어, 프로그램 카운터 값은 명령 어드레스를 생성하기 위하여 오프셋 (offset) 값에 가산될 수도 있다. 프로그램 카운터 (28) 는 프로세싱 엘리먼트들 (14) 의 전부에 대한 단일 프로그램 카운터로서 이용될 수도 있으므로, 프로그램 카운터 (28) 는 "글로벌 프로그램 카운터 (global program counter)" 또는 "글로벌 프로그램 카운터 레지스터 (global program counter register)" 로서 본원에서 지칭될 수도 있다.

[0077] 페치 모듈 (30) 은 프로그램 카운터 (28) 에 저장된 프로그램 카운터 값에 기초하여 명령 저장소 (16) 로부터 명령을 페치 (예를 들어, 취출) 하도록 구성된다. 예를 들어, 페치 모듈 (30) 은 프로그램 카운터 (28) 에 저장된 프로그램 카운터 값에 의해 식별된 명령 어드레스로부터 명령을 페치할 수도 있다. 페치 모듈 (30) 은 추가의 프로세싱을 위하여 페치된 명령을 디코드 모듈 (32) 에 제공할 수도 있다.

[0078] 디코드 모듈 (32) 은 페치 모듈 (30) 로부터 수신된 명령을 디코딩하도록 구성된다. 명령을 디코딩하는 것은 명령이 프로세싱 엘리먼트들 (14) 에 의해 프로세싱될 수 있는 명령의 타입인지 여부를 결정하는 것을 포함할 수도 있다. 명령이 프로세싱 엘리먼트들 (14) 에 의해 프로세싱될 수 있는 명령의 타입인 경우, 디코드 모듈 (32) 은 명령이 프로세싱 엘리먼트들 (14) 의 하나 이상 상에서 실행하게 할 수도 있다. 일부의 예들에서, 디코드 모듈 (32) 은 명령이 프로세싱 엘리먼트들 (14) 의 전부 상에서 실행하게 할 수도 있다. 다른 예들에서, 디코드 모듈 (32) 은 명령이 전부보다 적은 프로세싱 엘리먼트들 (14) 상에서 실행하게 할 수도 있다. 일부의 경우들에 있어서, 명령이 프로세싱 엘리먼트들 (14) 의 하나 이상 상에서 실행하게 하는 것은 실행을 위하여 프로세싱 엘리먼트들 (14) 의 하나 이상에 대해 명령을 발행하는 것을 포함할 수도 있다. 예를 들어, 디코드 모듈 (32) 은 프로세싱을 위하여 활성 스레드들에 대응하는 모든 프로세싱 엘리먼트들 (14) 에 대해 순차적인 명령을 발행할 수도 있다. 명령이 프로세싱 엘리먼트들 (14) 에 의해 프로세싱될 수 있는 명령의 타입이 아닌 경우, 제어 유닛 (12) 은 프로세싱을 위하여 프로세싱 엘리먼트들 (14) 중의 임의의 것에 대해 명령을 발행하지 않고 명령을 프로세싱할 수도 있다. 예를 들어, 명령은 프로세싱 엘리먼트들 (14) 에 의한 프로세싱을 요구하지 않는 타입의 제어 흐름 명령일 수도 있고, 이 경우, 제어 유닛 (12) 은 프로세싱 엘리먼트들 (14) 중의 임의의 것에 대해 명령을 발행하지 않고 명령을 프로세싱할 수도 있다.

[0079] 어느 하나의 경우에 있어서, 디코드 모듈 (32) 은 추가의 프로세싱을 위하여 제어 정보를 제어 흐름 모듈 (34) 에 포워딩 (forwarding) 할 수도 있다. 일부의 예들에서, 제어 정보는 명령 자체일 수도 있다. 추가의 예들에서, 제어 정보는 예를 들어, 명령이 제어 흐름 명령인지 또는 순차적인 명령인지 여부를 표시하는 정보를 포함할 수도 있다. 명령이 제어 흐름 명령인 경우, 제어 정보는 예를 들어, 명령이 분기 명령, 점프 명령, 호출 명령, 또는 복귀 명령인지 여부를 표시하는 정보를 더 포함할 수도 있다. 명령이 분기 또는 점프 명령인 경우, 제어 정보는 예를 들어, 분기 또는 점프 명령이 순방향 또는 역방향 분기 또는 점프 명령인지 여부를 표시하는 정보를 더 포함할 수도 있다. 명령이 분기 명령인 경우, 제어 정보는 예를 들어, 분기 조건을 특정하는 정보를 더 포함할 수도 있다.

[0080] 프로세싱 엘리먼트들 (14) 에 의해 프로세싱될 수 있는 타입인 명령들은 산술 명령들 및 논리 명령들을 포함할 수도 있다. 산술 명령은 산술 동작을 수행하도록 프로세싱 엘리먼트들 (14) 에 지시하는 명령을 지칭할 수도 있고, 논리 명령은 논리 동작을 수행하도록 프로세싱 엘리먼트들 (14) 에 지시하는 명령을 지칭할 수도 있다. 일부의 예들에서, 제어 흐름 명령은 프로세싱 엘리먼트들 (14) 에 의해 프로세싱될 수 있는 명령일 수도 있다 (예를 들어, 제어 흐름 명령은 프로세싱 엘리먼트들 (14) 에 의해 평가되는 분기 조건을 포함할 수도 있음). 프로세싱 엘리먼트들 (14) 에 의해 프로세싱될 수 있는 타입이 아닌 명령들은, 분기 조건이 제어 유닛 (12) 에 의해 평가되는 제어 흐름 명령들, 및/또는 분기 조건을 가지지 않는 제어 흐름 명령들을 포함할 수도 있다.

[0081] 제어 흐름 모듈 (34) 은 제어 유닛 (12) 에 의해 프로세싱되어야 할 다음 명령과 연관된 프로그램 카운터 값을 결정할 수도 있고, 프로그램 카운터 값을 프로그램 카운터 (28) 로 로딩할 수도 있다. 이전에 페치된 명령이 순차적인 명령인 경우, 제어 흐름 모듈 (34) 은 프로그램 카운터 (28) 로 로딩하기 위하여 다음의 순차적인 명령을 표시하는 프로그램 카운터 값을 선택할 수도 있다. 이전에 페치된 명령이 분기 또는 점프 명령인 경우, 제어 흐름 모듈 (34) 은 프로그램 카운터 (28) 로 로딩하기 위한 새로운 프로그램 카운터 값을 선택하기 위하여 MINRC 를 활용할 수도 있다. 예를 들어, 제어 흐름 모듈 (34) 은 제어 흐름 명령에 의해 특정된 타겟 명령과 연관된 타겟 프로그램 카운터 값, 다음의 순차적인 명령을 표시하는 프로그램 카운터 값, 또는 프로그램 카운터 (28) 로 로딩하기 위한 MINRC 값 중의 하나를 선택할 수도 있다. 이전에 페치된 명령이 호출 명령인 경우, 제어 흐름 모듈 (34) 은 프로그램 카운터 (28) 로 로딩하기 위하여 호출 명령에 의해 특정된 타겟 명령을 표시하는 타겟 프로그램 카운터 값을 선택할 수도 있다. 이전에 페치된 명령이 복귀 명령인 경우, 제어 흐름 모듈 (34) 은 프로그램 카운터 (28) 로 로딩하기 위하여 서브루틴 호출 스택으로부터 팝핑되는 복귀 어드레스를 표시하는 프로그램 카운터 값을 선택할 수도 있다.

[0082] 제어 흐름 모듈 (34) 은 프로세싱 시스템 (10) 에서 실행되는 각각의 스레드에 대한 재개 카운터 값을 저장할 수도 있다. 예를 들어, 제어 흐름 모듈 (34) 에 저장된 재개 카운터 값들의 수는 프로세싱 시스템 (10) 에 포함된 프로세싱 엘리먼트들 (14) 의 수와 동일할 수도 있다. 각각의 재개 카운터 값에 대하여, 각각의 재개 카운터 값에 대응하는 스레드가 비활성인 경우, 재개 카운터 값은 비활성 스레드가 활성화 또는 재활성화되도록 스케줄링되는 프로그램 카운터 값을 표시할 수도 있다. 이와 다르게, 각각의 재개 카운터 값에 대응하는 스레드가 활성인 경우, 일부의 예들에서, 재개 카운터 값은 디폴트 값 (예를 들어, 최대 레지스터 값, 또는 재개 카운터를 위한 저장 슬롯 또는 레지스터에서 표현될 수 있는 가장 큰 값인 값) 으로 설정될 수도 있다.

[0083] 제어 흐름 모듈 (34) 은 프로세싱 시스템 (10) 에서 현재 실행되고 있는 프로그램 모듈을 위한 MINRC 값을 저장할 수도 있다. MINRC 값은 프로세싱 모듈의 실행이 개시될 때에 활성인 스레드들과 연관된 재개 카운터 값들의 세트로부터의 가장 작은 재개 카운터 값을 표시할 수도 있다. 모든 스레드들이 활성인 경우, 일부의 예들에서, 최소 재개 카운터 값은 최대 값, 즉, 최소 재개 카운터를 위한 저장 슬롯에서 표현될 수 있는 가장 큰 값인 값으로 설정될 수도 있다.

[0084] 제어 흐름 모듈 (34) 은 프로세싱 시스템 (10) 에서 실행되는 각각의 스레드에 대한 활성 플래그를 저장할 수도 있다. 예를 들어, 제어 흐름 모듈 (34) 에 저장된 활성 플래그들의 수는 프로세싱 시스템 (10) 에 포함된 프로세싱 엘리먼트들 (14) 의 수와 동일할 수도 있다. 각각의 활성 플래그는 활성 플래그와 연관된 스레드가 활성 또는 비활성인지 아닌지 여부를 표시할 수도 있다. 일부의 예들에서, 활성 플래그는, 활성 플래그와 연관된 스레드가 활성임을 표시하도록 설정 (set) 되며, 활성 플래그와 연관된 스레드가 비활성임을 표시하도록 재설정 (reset) 되는 단일 비트일 수도 있다. 제어 흐름 모듈 (34) 은 스레드를 활성화 및 비활성화할 때에 특별한 스레드에 대한 활성 플래그를 설정 및 재설정할 수도 있다.

[0085] 제어 흐름 모듈 (34) 은 서브루틴 호출 명령들 및 서브루틴 복귀 명령들을 프로세싱함에 있어서 보조하기 위하

여 하나 이상의 스택들을 관리할 수도 있다. 예를 들어, 제어 흐름 모듈 (34) 은 호출 명령을 실행하는 것에 응답하여, 복귀 어드레스를 서브루틴 스택 상으로, 그리고 MINRC 값을 MINRC 스택 상으로 푸시할 수도 있다.

또 다른 예로서, 제어 흐름 모듈 (34) 은 복귀 명령을 실행하는 것에 응답하여, 서브루틴 스택으로부터 복귀 어드레스를, 그리고 MINRC 스택으로부터 MINRC 값을 팝핑할 수도 있다. 서브루틴 스택 및 MINRC 스택은 동일하거나 상이한 스택들일 수도 있다.

[0086] 일부의 예들에서, 제어 흐름 모듈 (34) 은, 플래그들의 세트에서의 각각의 플래그가 현재 실행되고 있는 프로그램 모듈의 실행이 활성화되었을 때에 각각의 스레드가 활성이었는지 여부를 표시하는 플래그들의 세트를 저장할 수도 있다. 이러한 예들에서, 제어 흐름 모듈 (34) 은 호출 명령을 실행하는 것에 응답하여 플래그들의 세트에 대한 플래그 값들을 스택 상으로 푸시할 수도 있고, 복귀 명령을 실행하는 것에 응답하여 스택에서 플래그들 세트에 대한 플래그 값들을 팝핑할 수도 있다.

[0087] 일부의 예들에서, 제어 흐름 모듈 (34) 은 프로그램 상태를 저장할 수도 있다. 예를 들어, 제 1 프로그램 상태는 모든 스레드들이 활성임을 표시할 수도 있고, 제 2 프로그램 상태는 적어도 하나의 스레드가 활성이며 적어도 하나의 스레드가 비활성임을 표시할 수도 있고, 제 3 프로그램 상태는 모든 스레드들이 비활성임을 표시할 수도 있다. 이러한 예들에서, 프로그램 상태는 프로그램 카운터 (28) 로 로딩하기 위한 프로그램 카운터 값을 선택하기 위하여 이용될 수도 있다.

[0088] 일부의 예들에서, 제어 흐름 모듈 (34) 은 통신 경로 (22) 를 통해 프로세싱 엘리먼트들 (14) 의 하나 이상을 활성화 및 비활성화하도록 구성될 수도 있다. 추가적인 예들에서, 제어 흐름 모듈 (34) 은 특별한 프로세싱 엘리먼트들 (14) 을 활성화 및 비활성화하도록 디코드 모듈 (32) 에 지시할 수도 있다.

[0089] 일부의 예들에서, 도 1 및 도 2 의 프로세싱 시스템 (10) 은 그래픽 프로세싱 유닛 (GPU) 에 포함될 수도 있다. 이러한 예들에서, 프로세싱 시스템 (10) 은 예를 들어, 정점 쉐이더 (vertex shader) 유닛, 픽셀 쉐이더 (pixel shader) 유닛, 프래그먼트 쉐이더 (fragment shader) 유닛, 기하구조 쉐이더 (geometry shader) 유닛, 통합된 쉐이더 (unified shader) 유닛, 등과 같이, GPU 내에 포함된 쉐이더 유닛을 구현하기 위하여 이용될 수도 있다. 이러한 예들에서, 프로세싱 시스템 (10) 은 예를 들어, 정점 쉐이더 프로그램들, 프래그먼트 쉐이더 프로그램들, 기하구조 쉐이더 프로그램들, 등과 같은 쉐이더 프로그램들을 실행하도록 구성될 수도 있다.

[0090] 도 3 은 본 개시물의 서브루틴 실행 기법들을 구현하기 위하여 이용될 수도 있는 일 예의 제어 흐름 모듈 (34) 을 예시하는 블록도이다. 제어 흐름 모듈 (34) 은 분기 조건 평가기 (40), 최소 재개 카운터 (MINRC) 레지스터 (42), 다음 명령 생성기 (44), 재개 카운터 레지스터들 (46), 스레드 상태 관리기 (48), 서브루틴 핸들러 (50) 및 스택 저장 구조 (52) 를 포함한다.

[0091] 분기 조건 평가기 (40) 는 프로세싱 시스템 (10) 에서 실행되는 각각의 활성 스레드에 대한 조건부 분기 명령에 의해 특정된 분기 조건을 평가하도록 구성된다. 분기 조건 평가기 (40) 는 디코드 모듈 (32) 로부터 현재 프로세싱된 명령이 분기 명령인지 여부를 표시하는 정보를 수신할 수도 있고, 현재 프로세싱된 명령이 분기 명령인 경우, 분기 조건 평가기 (40) 는 또한, 현재 프로세싱된 분기 명령에 대한 분기 조건을 표시하는 정보를 디코드 모듈 (32) 로부터 수신할 수도 있다. 일부의 예들에서, 현재 프로세싱된 명령이 분기 명령인지 여부를 표시하는 정보, 및 분기 조건을 표시하는 정보 중의 하나 또는 양자는 명령 자체의 표현일 수도 있다. 추가의 예들에서, 이 정보 컴포넌트들 중의 하나 또는 양자는 디코드 모듈 (32) 에 의해 생성되는 신호일 수도 있다.

[0092] 분기 조건 평가기 (40) 는 스레드-특정 데이터를 이용하여 각각의 스레드에 대해 동일한 분기 조건을 평가할 수도 있다. 일부의 예들에서, 분기 조건 평가기 (40) 는 각각의 스레드에 대한 분기 조건을 평가하기 위해 필요한 임의의 데이터를 얻을 수도 있고, 각각의 스레드에 대한 분기 조건 평가 결과를 생성하기 위하여 각각의 스레드에 대한 분기 조건을 내부적으로 평가할 수도 있다. 추가의 예들에서, 분기 조건 평가기 (40) 는 활성 스레드에 대응하는 각각의 프로세싱 엘리먼트 (14) 에 대해, 각각의 스레드에 대한 분기 조건을 평가하기 위해 필요한 데이터를 얻도록, 분기 조건을 평가하도록, 그리고 각각의 스레드에 대한 분기 조건 평가 결과를 분기 조건 평가기 (40) 에 제공하도록 지시할 수도 있다. 어느 하나의 경우에 있어서, 분기 조건 평가기 (40) 는 각각의 활성 스레드에 대하여, 분기 조건이 각각의 스레드에 대해 충족되는지 여부를 결정할 수도 있다.

[0093] 일부의 예들에서, 분기 조건 평가기 (40) 는 분기 명령에 대한 분기 발산이 균일한지 또는 발산하는지 여부를 결정할 수도 있다. 예를 들어, 분기 조건 평가기 (40) 는 모든 활성 스레드들이 분기 조건을 충족하였는지 여부와, 모든 활성 스레드들이 분기 조건을 충족하지 않았는지 여부를 결정할 수도 있다. 모든 활성 스레드

들이 분기 조건을 충족하였거나 충족하지 않았을 경우, 분기 조건 평가기 (40) 는 분기 명령에 대한 분기 발산이 균일한 것으로 결정할 수도 있다. 일부의 활성 스레드들이 분기 조건을 충족하였으며 일부의 활성 스레드들이 분기 조건을 충족하지 않았을 경우, 분기 조건 평가기 (40) 는 분기 명령에 대한 분기 발산이 발산하는 것으로 결정할 수도 있다. 분기 발산이 균일한 예들에서는, 분기 조건 평가기 (40) 가 분기 조건이 균일하게 충족되는지 균일하게 충족되지 않는지 여부를 결정할 수도 있다.

[0094] 분기 조건 평가기 (40) 는 분기 조건 상태 정보를 스레드 상태 관리기 (48) 에 제공할 수도 있다. 분기 조건 상태 정보는 프로세싱 시스템 (10) 에서 실행되는 각각의 활성 스레드에 대하여, 각각의 스레드가 분기 조건을 충족하였는지 또는 분기 조건을 충족하지 않았는지 여부 (즉, 스레드에 대한 분기 조건 평가 결과) 를 표시할 수도 있다. 스레드 상태 관리기 (48) 는 분기 명령을 수신하는 것에 응답하여 특별한 스레드들을 활성화 및/또는 비활성화할 것인지 여부를 결정하기 위하여 분기 조건 상태 정보를 이용할 수도 있다.

[0095] 분기 조건 평가기 (40) 는 분기 발산 정보를 다음 명령 생성기 (44) 에 제공할 수도 있다. 분기 발산 정보는 분기 명령에 대한 분기 발산이 균일한지 또는 발산하는지 여부를 표시하는 정보를 포함할 수도 있다. 분기 명령에 대한 분기 발산이 균일한 경우, 분기 발산 정보는 또한, 분기 조건이 균일하게 충족되는지 또는 균일하게 충족되지 않는지 여부를 표시하는 정보를 포함할 수도 있다. 일부의 예들에서, 분기 발산 정보는 활성 스레드들의 각각에 대한 분기 조건 상태 정보의 형태를 취할 수도 있다. 다른 예들에서, 분기 발산 정보는 개별적인 스레드들에 대한 분기 조건 상태 정보를 반드시 포함하지 않을 수도 있다.

[0096] MINRC 레지스터 (42) 는 프로세싱 시스템 (10) 에서 현재 실행하는 프로그램 모듈에 대한 MINRC 값을 저장할 수도 있다. MINRC 값은 현재 실행하는 프로그램 모듈의 실행이 개시될 때에 활성인 모든 스레드들 중의 가장 작은 재개 카운터 값을 표시할 수도 있다. MINRC 값은 스레드 상태 관리기 (48), 서브루틴 핸들러 (50) 및 스택 저장 구조 (52) 중의 하나 이상에 의해 업데이트 및/또는 수정될 수도 있다. 다음 명령 생성기 (44) 에 대해 더욱 상세하게 설명되는 바와 같이, 제어 흐름 모듈 (34) 은 MINRC 레지스터 (42) 에 기초하여 현재 실행하는 프로그램 모듈의 실행을 제어하도록 구성된다. MINRC 레지스터 (42) 는 다음 명령 생성기 (44), 재개 카운터 레지스터들 (46), 서브루틴 핸들러 (50) 및 스택 저장 구조 (52) 에 통신가능하게 결합된다.

[0097] 다음 명령 생성기 (44) 는 현재 실행되고 있는 명령의 타입을 표시하는 정보, 명령이 분기 명령인 경우에 현재 실행되고 있는 명령의 분기 발산을 표시하는 정보, 만약 있을 경우, 현재 실행되고 있는 명령에 의해 특정된 타겟 명령을 표시하는 정보, 명령이 복귀 명령인 경우에 복귀 어드레스를 표시하는 정보, 및 명령이 순방향 분기 또는 점프 명령인 경우에 MINRC 레지스터 (42) 에 저장된 MINRC 값에 기초하여, 실행되어야 할 다음 명령에 대응하는 프로그램 카운터 값을 생성하도록 구성된다. 다음 명령 생성기 (44) 는 다음 명령 생성기 (44) 에 의해 생성된 프로그램 카운터 값이, 다음 명령 사이클의 실행이 개시될 때에 프로그램 카운터 (28) 로 로딩되게 할 수도 있다.

[0098] 현재 실행되고 있는 명령의 타입을 표시하는 정보는 디코드 모듈 (32) 로부터 수신될 수도 있고, 예를 들어, 명령이 순차적인 명령 또는 제어 흐름 명령인지 여부를 표시하는 정보를 포함할 수도 있다. 명령이 제어 흐름 명령인 경우, 명령의 타입을 표시하는 정보는 예를 들어, 명령이 분기 명령, 점프 명령 또는 서브루틴 명령 (예를 들어, 호출 또는 복귀 명령) 인지 여부를 표시하는 정보를 포함할 수도 있다. 명령이 분기 또는 점프 명령인 경우, 명령의 타입을 표시하는 정보는 예를 들어, 명령이 순방향 분기 또는 점프 명령인지 여부, 또는 명령이 역방향 분기 또는 점프 명령인지 여부를 표시하는 정보를 포함할 수도 있다.

[0099] 명령의 분기 발산을 표시하는 정보는 분기 조건 평가기 (40) 로부터 수신될 수도 있고, 예를 들어, 분기 발산이 균일한지 또는 발산하는지 여부를 표시하는 정보를 포함할 수도 있다. 분기 발산이 균일한 경우, 명령의 분기 발산을 표시하는 정보는 예를 들어, 분기 조건이 균일하게 충족되는지 또는 균일하게 충족되지 않는지 여부를 표시하는 정보를 포함할 수도 있다.

[0100] 타겟 명령을 표시하는 정보는 디코드 모듈 (32) 로부터 수신될 수도 있고, 예를 들어, 타겟 프로그램 카운터 값, 또는 타겟 프로그램 카운터 값을 표시하는 오프셋 값을 포함할 수도 있다. 오프셋 값은 예를 들어, 타겟 프로그램 카운터 값을 생성하기 위하여 프로그램 카운터에 추가되는 값일 수도 있다. 타겟 명령을 표시하는 정보는 현재의 명령이 타겟 명령을 특정할 때에 실행되어야 할 다음 명령에 대한 프로그램 카운터를 결정하기 위하여 이용될 수도 있다. 이 명령들은 예를 들어, 조건부 분기 명령들, 점프 명령들, 및 호출 명령들을 포함할 수도 있다.

[0101] 복귀 어드레스를 표시하는 정보는 스택 저장 구조 (52) 로부터 수신될 수도 있다. 예를 들어, 서브루틴 핸

들러 (50) 는 서브루틴 호출 명령이 실행될 때에 복귀 어드레스를 스택 저장 구조 (52) 상으로 푸시할 수도 있다. 서브루틴에 대한 복귀 명령이 실행될 때, 서브루틴 핸들러 (50) 는 스택 저장 구조 (52) 에서 복귀 어드레스를 팝핑할 수도 있고, 실행되어야 할 다음 명령에 대한 프로그램 카운터 값을 결정하기 위하여 복귀 어드레스를 다음 명령 생성기 (44) 에 제공할 수도 있다.

[0102] 순차적인 명령들에 대하여, 다음 명령 생성기 (44) 는 프로그램 카운터 (28) 로 로딩하기 위한 프로그램 카운터 값으로서, 다음의 순차적인 명령에 대응하는 프로그램 카운터 값을 선택한다. 다음의 순차적인 명령은 명령 저장소 (16) 에 저장된 프로그램에 대한 명령들의 순서화된 시퀀스에서 현재 실행되고 있는 명령 직후에 발생하는 명령을 지칭할 수도 있다.

[0103] 역방향 점프 명령에 대하여, 다음 명령 생성기 (44) 는 프로그램 카운터 (28) 로 로딩하기 위한 프로그램 카운터 값으로서, 역방향 점프 명령에 의해 특정된 타겟 명령을 표시하는 타겟 프로그램 카운터 값을 선택할 수도 있다. 순방향 점프 명령에 대하여, 이하에서 더욱 상세하게 설명된 바와 같이, 다음 명령 생성기 (44) 는 MINRC 레지스터 (42) 에 저장된 MINRC 값에 기초하여 프로그램 카운터 (28) 로 로딩하기 위한 프로그램 카운터 값을 선택할 수도 있다.

[0104] 역방향 분기 명령에 대하여, 다음 명령 생성기 (44) 는 역방향 분기 명령에 대한 분기 조건이 균일하게 충족되지 않는지 여부를 결정할 수도 있다. 역방향 분기 명령에 대한 분기 조건이 균일하게 충족되지 않는 경우, 다음 명령 생성기 (44) 는 프로그램 카운터 (28) 로 로딩하기 위한 프로그램 카운터 값으로서, 다음의 순차적인 명령에 대응하는 프로그램 카운터 값을 선택할 수도 있다. 다른 한편으로, 역방향 분기 명령에 대한 분기 조건이 균일하게 충족되거나 발산하는 경우, 다음 명령 생성기 (44) 는 프로그램 카운터 (28) 로 로딩하기 위한 프로그램 카운터 값으로서, 역방향 점프 명령에 의해 특정된 타겟 명령을 표시하는 타겟 프로그램 카운터 값을 선택할 수도 있다.

[0105] 순방향 분기 명령들에 대하여, 다음 명령 생성기 (44) 는 순방향 분기 명령에 대한 분기 조건이 균일하게 충족되지 않거나 발산하는지 여부를 결정할 수도 있다. 순방향 분기 명령에 대한 분기 조건이 균일하게 충족되지 않거나 발산하는 경우, 다음 명령 생성기 (44) 는 프로그램 카운터 (28) 로 로딩하기 위한 프로그램 카운터 값으로서, 다음의 순차적인 명령에 대응하는 프로그램 카운터 값을 선택할 수도 있다. 순방향 분기 명령에 대한 분기 조건이 균일하게 충족되는 경우, 이하에서 더욱 상세하게 설명된 바와 같이, 다음 명령 생성기 (44) 는 MINRC 레지스터 (42) 에 저장된 MINRC 값에 기초하여 프로그램 카운터 (28) 로 로딩하기 위한 프로그램 카운터 값을 선택할 수도 있다.

[0106] 호출 명령들에 대하여, 다음 명령 생성기 (44) 는 프로그램 카운터 (28) 로 로딩하기 위한 프로그램 카운터 값으로서, 호출 명령에 의해 특정된 타겟 명령을 표시하는 타겟 프로그램 카운터 값을 선택할 수도 있다. 복귀 명령들에 대하여, 다음 명령 생성기 (44) 는 프로그램 카운터 (28) 로 로딩하기 위한 프로그램 카운터 값으로서, 스택 저장 구조 (52) 로부터 팝핑된 복귀 어드레스를 표시하는 프로그램 카운터를 선택할 수도 있다.

[0107] 위에서 논의된 바와 같이, 분기 조건이 균일하게 충족되는 순방향 점프 명령 또는 순방향 분기 명령을 실행하는 것에 응답하여, 다음 명령 생성기 (44) 는 MINRC 레지스터 (42) 에 저장된 MINRC 값에 기초하여 프로그램 카운터 (28) 로 로딩하기 위한 프로그램 카운터 값을 선택할 수도 있다. 일부의 예들에서, 다음 명령 생성기 (44) 는 MINRC 값에 기초하여 프로그램 카운터 (28) 로 로딩하기 위한 프로그램 카운터 값으로서, 명령에 의해 특정된 타겟 프로그램 카운터 값 또는 MINRC 레지스터 (42) 에 저장된 MINRC 값 중의 하나를 선택할 수도 있다. 예를 들어, 일부의 예들에서, 다음 명령 생성기 (44) 는 타겟 프로그램 카운터 값이 MINRC 값 이하인지 여부를 결정할 수도 있다. 타겟 프로그램 카운터 값이 MINRC 값 이하인 경우, 다음 명령 생성기 (44) 는 프로그램 카운터 (28) 로 로딩하기 위한 프로그램 카운터 값으로서 타겟 프로그램 카운터 값을 선택할 수도 있다. 다른 한편으로, 타겟 프로그램 카운터 값이 MINRC 값 이하가 아닌 경우, 다음 명령 생성기 (44) 는 프로그램 카운터 (28) 로 로딩하기 위한 프로그램 카운터 값으로서 MINRC 값을 선택할 수도 있다. 예를 들어, 타겟 프로그램 카운터 값이 MINRC 값보다 더 작은지 여부, 또는 MINRC 값이 타겟 프로그램 카운터 값보다 더 큰지 여부를 결정하는 것을 포함하는 다른 비교 동작들이 다른 예들에서 또한 가능하다. 이러한 방식으로, 다음 명령 생성기 (44) 는 MINRC 레지스터 (42) 에 저장된 MINRC 값에 기초하여 현재 실행하는 프로그램 모듈의 실행을 제어할 수도 있다.

[0108] 재개 카운터 레지스터들 (46) 은 프로세싱 시스템 (10) 에서 실행되는 스레드들에 대한 복수의 재개 카운터 값들을 저장한다. 각각의 재개 카운터 값은 프로세싱 시스템 (10) 에서 실행되는 각각의 스레드에 대응할 수도 있고, 각각의 스레드가 비활성인 경우에 각각의 스레드가 활성화되도록 스케줄링되는 프로그램 카운터 값을

표시할 수도 있다. 스레드가 활성인 경우, 재개 카운터 값은, 일부의 경우들에 있어서 프로그램들을 실행하기 위해 이용된 유효한 프로그램 카운터 값들의 범위보다 더 큰 값일 수도 있는 디폴트 값으로 설정된다. 예를 들어, 스레드가 활성인 경우, 재개 카운터는 최대 값인 값 (즉, 재개 카운터를 위한 저장 슬롯 또는 레지스터에서 표현될 수 있는 가장 큰 값인 값) 으로 설정될 수도 있다. 대응하는 스레드에 대한 재개 카운터는 스레드가 활성일 때에 디폴트 값으로 설정되므로, 각각의 재개 카운터는 또한, 각각의 재개 카운터에 대응하는 스레드가 활성인지 여부를 표시할 수도 있다.

[0109] 일부의 예들에서, 재개 카운터 레지스터들 (46) 은 복수의 재개 카운터 값들을 저장하도록 구성된 복수의 레지스터들을 포함할 수도 있다. 예를 들어, 각각의 레지스터는 프로세싱 시스템 (10) 에서 실행되는 복수의 스레드들 중의 각각의 하나에 대한 재개 카운터 값을 저장하도록 구성되는 재개 카운터 레지스터일 수도 있다. 재개 카운터 레지스터들 (46) 은 스레드 상태 관리기 (48) 에 통신가능하게 결합된다.

[0110] 스레드 상태 관리기 (48) 는 프로세싱 시스템 (10) 에서 실행되는 스레드들의 상태를 관리하도록 구성된다. 예를 들어, 스레드 상태 관리기 (48) 는 적절하게, 프로세싱 시스템 (10) 에서 실행되는 스레드들을 활성화 및 비활성화할 수도 있고, 재개 카운터 레지스터들 (46) 을 업데이트할 수도 있고, MINRC 레지스터 (42) 를 업데이트할 수도 있다.

[0111] 스레드 상태 관리기 (48) 는 프로세싱 시스템 (10) 이 발산 분기 조건을 갖는 분기 명령을 실행하는 것에 응답하여 하나 이상의 스레드들을 비활성화할 수도 있다. 예를 들어, 스레드 상태 관리기 (48) 는 분기 조건 평가기 (40) 로부터 발산 분기 조건이 발생하였는지 여부를 표시하는 정보를, 분기 조건 평가기 (40) 또는 디코드 모듈 (32) 의 어느 하나로부터 분기 명령이 순방향 분기 명령 또는 역방향 분기 명령인지 여부를 표시하는 정보를, 그리고 어느 스레드들이 분기 조건을 충족하였는지와, 어느 스레드들이 분기 조건을 충족하지 않았는지를 표시하는 정보를 수신할 수도 있다. 스레드 상태 관리기 (48) 는 발산 분기 명령이 순방향 분기 명령 또는 역방향 분기 명령인지 여부를 결정할 수도 있다. 발산 분기 명령이 순방향 분기 명령인 경우, 스레드 상태 관리기 (48) 는 분기 조건을 충족하였던 각각의 활성 스레드를 비활성화할 수도 있다. 발산 분기 명령이 역방향 분기 명령인 경우, 스레드 상태 관리기 (48) 는 분기 조건을 충족하지 않았던 각각의 활성 스레드를 비활성화할 수도 있다.

[0112] 비활성화되고 있는 각각의 스레드에 대하여, 스레드 상태 관리기 (48) 는 각각의 스레드에 대응하는 재개 카운터 레지스터들 (46) 에 저장된 재개 카운터 값을, 각각의 스레드가 재활성화되어야 하는 프로그램 카운터 값 (예를 들어, 각각의 스레드가 재활성화되도록 스케줄링되는 프로그램 카운터 값) 을 표시하는 값으로 설정할 수도 있다. 발산 순방향 분기 명령에 응답하여 스레드를 비활성화할 때, 스레드 상태 관리기 (48) 는 스레드에 대한 재개 카운터 값을, 순방향 분기 명령에 의해 특정된 타겟 프로그램 카운터 값을 표시하는 값으로 설정할 수도 있다. 발산 역방향 분기 명령에 응답하여 스레드를 비활성화할 때, 스레드 상태 관리기 (48) 는 스레드에 대한 재개 카운터 값을, 다음의 순차적인 명령에 대응하는 프로그램 카운터 값을 표시하는 값으로 설정할 수도 있다. 비활성화된 스레드들에 대한 재개 카운터 값들을 설정한 후, 이하에서 더욱 상세하게 설명된 바와 같이, 스레드 상태 관리기 (48) 는 현재 실행하는 프로그램 모듈에 대해 MINRC 레지스터 (42) 에 저장된 MINRC 값을 업데이트할 수도 있다.

[0113] 일부의 예들에서, 특별한 스레드를 비활성화하기 위하여, 스레드 상태 관리기 (48) 는 특별한 스레드에 대응하는 프로세싱 엘리먼트들 (14) 중의 각각의 하나를 비활성화할 수도 있다. 추가적인 예들에서는, 특별한 스레드를 비활성화하기 위하여, 스레드 상태 관리기 (48) 는 데이터 저장소 (16) 가 특별한 스레드에 대응하는 임의의 연산 결과들을 저장하지 않아야 함을 표시하는 신호를, 특별한 스레드에 대응하는 데이터 저장소 (16) 의 일부분으로 전송할 수도 있다. 스레드들을 비활성화할 때, 일부의 예들에서, 스레드 상태 관리기 (48) 는 비활성화되어야 할 스레드에 대응하는 활성 플래그를, 스레드가 비활성화되었음을 표시하는 값으로 설정할 수도 있다.

[0114] 스레드 상태 관리기 (48) 는 임의의 비활성화된 스레드들이 각각의 명령 사이클에 대해 재활성화될 필요가 있는지 여부를 결정하기 위하여, 각각의 명령 사이클에 대해 재개 검사 동작을 수행하도록 구성될 수도 있다. 일부의 예들에서, 재개 검사 동작을 수행하기 위하여, 스레드 상태 관리기 (48) 는 복수의 재개 카운터 값들의 각각을, 현재 프로세싱된 명령과 연관된 프로그램 카운터 값 (즉, 프로그램 카운터 (28) 로 현재 로딩되는 프로그램 카운터 값) 과 비교할 수도 있다. 예를 들어, 스레드 상태 관리기 (48) 는 각각의 재개 카운터 값이 프로그램 카운터 (28) 에 저장된 현재의 프로그램 카운터 값과 동일한지 여부를 결정할 수도 있다. 특별한 스레드에 대한 재개 카운터 값이 현재의 프로그램 카운터 값과 동일한 경우, 스레드 상태 관리기 (48) 는 스레

드를 재활성화할 수도 있다. 이와 다르게, 특별한 스레드에 대한 재개 카운터 값이 현재의 프로그램 카운터 값과 동일하지 않은 경우, 스레드 상태 관리기 (48) 는 스레드의 비활성화된 상태를 유지할 수도 있다.

[0115] 재활성화되고 있는 각각의 스레드에 대하여, 스레드 상태 관리기 (48) 는 각각의 스레드에 대응하는 재개 카운터 값을, 스레드가 활성임을 표시하는 디폴트 값으로 설정할 수도 있다. 예를 들어, 디폴트 값은 재개 카운터 값에 대해 레지스터에서 표현될 수 있는 가장 큰 값일 수도 있다. 임의의 재활성화된 스레드들에 대한 재개 카운터 값들을 설정한 후, 이하에서 더욱 상세하게 설명된 바와 같이, 스레드 상태 관리기 (48) 는 MINRC 레지스터 (42) 에 저장된 MINRC 값을 업데이트할 수도 있다.

[0116] 일부의 예들에서, 특별한 스레드를 재활성화하기 위하여, 스레드 상태 관리기 (48) 는 특별한 스레드에 대응하는 프로세싱 엘리먼트들 (14) 중의 각각의 하나를 활성화할 수도 있다. 추가의 예들에서, 특별한 스레드를 재활성화하기 위하여, 스레드 상태 관리기 (48) 는 데이터 저장소 (16) 가 특별한 스레드에 대응하는 연산 결과들을 저장해야 함을 표시하는 신호를, 특별한 스레드에 대응하는 데이터 저장소 (16) 의 일부분으로 전송할 수도 있다. 스레드들을 재활성화할 때, 일부의 예들에서, 스레드 상태 관리기 (48) 는 스레드에 대응하는 활성 플래그를, 스레드가 활성화되었음을 표시하는 값으로 설정할 수도 있다.

[0117] 일부의 예들에서, 재개 검사 동작은 프로그램 카운터 값을 프로그램 카운터 (28) 로 로딩하는 것에 응답하여 개시될 수도 있다. 일부의 예들에서, 명령 사이클은 프로세싱 엘리먼트들 (14) 이 재개 검사 동작이 완료한 후에 재개 검사 동작의 일부로서 재활성화되었던 임의의 스레드들에 대한 연산 동작들을 수행하도록 하기에 충분한 길이일 수도 있다. 추가의 예들에서, 프로그램 카운터 (28) 에 저장된 프로그램 카운터 값에 대응하는 명령의 실행은, 재개 검사 동작이 완료되고 명령에 대해 재활성화되도록 스케줄링되는 임의의 스레드들이 활성화된 후까지 지연될 수도 있다. 이러한 예들에서, 재개 검사 동작이 완료된 후, 스레드 상태 관리기 (48) 는 프로세싱 엘리먼트들 (14) 로 하여금, 현재의 명령과 연관된 임의의 연산 동작들을 수행하는 것을 시작하게 할 수도 있다.

[0118] 위에서 논의된 바와 같이, 스레드들을 비활성화 및/또는 재활성화할 때에 하나 이상의 재개 카운터 값들을 업데이트하는 것에 응답하여, 스레드 상태 관리기 (48) 는 MINRC 레지스터 (42) 에 저장된 MINRC 값을 업데이트할 수도 있다. 일반적으로, 스레드 상태 관리기 (48) 는 업데이트된 MINRC 값을 결정함에 있어서 이용하기 위한 재개 카운터 값들의 후보 세트를 결정할 수도 있고, MINRC 값을, 재개 카운터 값들의 후보 세트로부터의 가장 작은 재개 카운터 값을 표시하는 값으로 설정할 수도 있다. 재개 카운터 값들의 후보 세트는, 프로세싱 시스템 (10) 에서 실행되는 모든 스레드들에 대한 재개 카운터 값을 포함하는, 재개 카운터 값들의 전체 세트의 서브세트일 수도 있다. 서브세트는 부모 세트 (parent set) 의 엘리먼트들의 전부 또는 전부보다 더 적은 것을 포함할 수도 있다. 일부의 경우들에 있어서, 재개 카운터 값들의 후보 세트는, 현재 실행하는 프로그램 모듈의 실행이 개시되었을 때에 활성이 아니었던 스레드들에 대응하는 하나 이상의 재개 카운터 값들을 배제할 수도 있다. MINRC 를 업데이트하는 동안의 고려사항으로부터 이러한 재개 카운터 값들을 배제함으로써, 본 개시물의 기법들은 서브루틴-특정 MINRC 가 서브루틴과 연관된 프로그램 공간 내에 있는 값들로 업데이트되는 것을 보장할 수도 있다.

[0119] 일부의 예들에서, 스레드 상태 관리기 (48) 는 재개 카운터 값들의 전체 세트로부터의 각각의 재개 카운터 값이 현재 실행하는 프로그램 모듈의 진입 포인트 (예를 들어, 서브루틴의 진입 포인트) 이상인지 여부를 결정할 수도 있고, 재개 카운터 값이 현재 실행하는 프로그램 모듈의 진입 포인트 이상인 그러한 재개 카운터들을, 재개 카운터 값들의 후보 세트인 것으로 선택할 수도 있다. 현재 실행하는 프로그램 모듈의 진입 포인트는 현재 실행하는 프로그램 모듈에 대응하는 프로그램 공간에 대한 시작 어드레스를 표시하는 프로그램 카운터 값일 수도 있다. 프로그램 모듈이 메인 프로그램 (즉, 최상위-레벨 프로그램) 인 경우들에는, 프로그램 모듈의 진입 포인트가 최상위-레벨 프로그램의 시작 어드레스일 수도 있다. 프로그램 모듈이 서브루틴 프로그램 모듈인 경우들에는, 프로그램 모듈의 진입 포인트가 서브루틴 프로그램 모듈의 시작 어드레스들일 수도 있다.

[0120] 추가적인 예들에서, 스레드 상태 관리기 (48) 는 플래그들의 세트에서의 각각의 플래그가 프로그램 모듈 (예를 들어, 서브루틴) 의 실행이 개시되었을 때에 각각의 스레드가 활성이었는지 여부를 표시하는 플래그들의 세트를 유지할 수도 있다. 예를 들어, 서브루틴 호출 명령을 실행하는 것에 응답하여, 스레드 상태 관리기 (48) 는 플래그들의 세트에서의 각각의 플래그를, 각각의 플래그에 대응하는 스레드가 호출 명령이 실행될 때에 활성인지 여부를 표시하는 값으로 설정할 수도 있다. 이러한 예들에서, 스레드 상태 관리기 (48) 가 MINRC 값을 업데이트할 때, 스레드 상태 관리기 (48) 는 재개 카운터 값들의 전체 세트로부터의 각각의 재개 카운터 값이 프로그램 모듈의 실행이 개시되었을 때에 재개 카운터 값에 대응하는 스레드가 활성이었음을 표시하는 각각의

플래그에 대응하는지 여부를 결정할 수도 있고, 스레드가 활성이었음을 표시하는 플래그 값을 갖는 그러한 재개 카운터들 값들을, 재개 카운터 값들의 후보 세트인 것으로 선택할 수도 있다.

[0121] 서브루틴 핸들러 (50) 는 프로세싱 시스템 (10) 에서의 서브루틴 제어 흐름 명령문들의 실행을 관리하도록 구성된다. 서브루틴 핸들러 (50) 는 현재 프로세싱된 명령이 호출 명령 또는 복귀 명령인지 여부를 표시하는 정보를 디코드 모듈 (32) 로부터 수신할 수도 있다. 호출 명령은 서브루틴 프로그램 모듈의 실행을 시작하도록 프로세싱 시스템 (10) 에 지시하는 명령을 지칭할 수도 있다. 복귀 명령은, 현재 실행하는 서브루틴 모듈의 실행을 종료하도록, 그리고 서브루틴 프로그램의 실행을 개시하였던 호출 명령 직후에 발생하는 호출자 프로그램에서의 명령으로 호출자 프로그램 모듈을 실행하는 것을 재개하도록 프로세싱 시스템 (10) 에 지시하는 명령을 지칭할 수도 있다. 호출자 프로그램은 메인 프로그램 또는 또 다른 서브루틴 프로그램의 어느 하나일 수도 있다.

[0122] 호출 명령을 실행하는 것에 응답하여, 서브루틴 핸들러 (50) 는 호출자 프로그램과 연관된 MINRC 레지스터 (42) 의 상태를 저장할 수도 있다. 예를 들어, 서브루틴 핸들러 (50) 는 MINRC 레지스터 (42) 에 저장된 MINRC 값을 스택 저장 구조 (52) 내의 스택 상으로 푸시할 수도 있다. MINRC 값을 스택 상으로 푸시하는 것은 스택 저장 구조 (52) 의 스택에 MINRC 값을 저장하는 것을 포함할 수도 있다. 또한, 호출 명령을 실행하는 것에 응답하여, 서브루틴 핸들러 (50) 는 복귀 어드레스를 스택 저장 구조 (52) 내의 스택 상으로 푸시할 수도 있다. 복귀 어드레스는 호출자 프로그램에서의 호출 명령 이후의 다음의 순차적인 명령에 대응하는 프로그램 카운터 값을 표시할 수도 있다.

[0123] 일부의 예들에서, MINRC 값이 그 상으로 푸시되는 스택은 복귀 어드레스가 그 상으로 푸시되는 스택과 동일한 스택일 수도 있다. 예를 들어, 서브루틴 핸들러 (50) 는 스택 프레임을 스택 저장 구조 (52) 에서의 스택 상으로 푸시할 수도 있고, 여기서, 스택 프레임은 호출 명령에 대응하는 MINRC 값 및 호출 명령에 대응하는 복귀 어드레스 양자를 포함한다. 추가적인 예들에서, 서브루틴 핸들러 (50) 는 MINRC 값을 스택 저장 구조 (52) 에서의 제 1 스택 상으로 푸시할 수도 있고, 복귀 어드레스를 스택 저장 구조 (52) 에서의 제 2 스택 상으로 푸시할 수도 있다. 이러한 예들에서, 제 1 스택은 제 2 스택과는 상이할 수도 있다.

[0124] 또한, 호출 명령을 실행하는 것에 응답하여, 서브루틴 핸들러 (50) 는 서브루틴에 대응하는 MINRC 에 기초하여 서브루틴의 실행을 제어하도록 프로세싱 시스템 (10) 을 구성할 수도 있다. 예를 들어, 서브루틴 핸들러 (50) 는 MINRC 레지스터 (42) 에 저장된 MINRC 값을, 서브루틴에 대응하는 MINRC 에 대한 초기 값으로 겹쳐쓰기 할 수도 있다. 다시 말해서, 서브루틴 핸들러 (50) 는 서브루틴의 실행을 위하여 디폴트 MINRC 값을 저장하기 위한 MINRC 레지스터 (42) 를 초기화할 수도 있다. 일부의 예들에서, 디폴트 값은 MINRC 레지스터 (42) 에서 표현될 수 있는 가장 큰 값일 수도 있다.

[0125] 복귀 명령을 실행하는 것에 응답하여, 서브루틴 핸들러 (50) 는 호출자 프로그램에 대응하는 MINRC 의 저장된 상태를 복원할 수도 있다. 예를 들어, 서브루틴 핸들러 (50) 는 스택 저장 구조 (52) 에서의 스택으로부터 호출자 프로그램에 대응하는 MINRC 의 저장된 상태를 팝핑할 수도 있다. 스택으로부터 MINRC 의 저장된 상태를 팝핑하는 것은 MINRC 의 저장된 상태에 대응하는 MINRC 값을 스택으로부터 팝핑하는 것을 포함할 수도 있다. 스택으로부터 MINRC 값을 팝핑하는 것은 스택 저장 구조 (52) 로부터 가장 최근에 저장된 MINRC 값을 취출하는 것을 포함할 수도 있다. 또한, 복귀 명령을 실행하는 것에 응답하여, 서브루틴 핸들러 (50) 는 복귀 어드레스를 스택 저장 구조 (52) 내의 스택에서 팝핑할 수도 있다. 복귀 어드레스는 호출자 프로그램에서의 호출 명령 이후의 다음의 순차적인 명령에 대응하는 프로그램 카운터 값을 표시할 수도 있다.

[0126] 서브루틴 핸들러 (50) 가 복귀 어드레스가 푸시되는 스택과 동일한 스택 상으로 MINRC 값을 푸시하는 예들에서는, 서브루틴 핸들러 (50) 가 동일한 스택으로부터 MINRC 값 및 복귀 어드레스를 팝핑할 수도 있다. 예를 들어, 서브루틴 핸들러 (50) 는 스택 저장 구조 (52) 에서의 스택에서 스택 프레임을 팝핑할 수도 있고, 여기서, 스택 프레임은 MINRC 값 및 복귀 어드레스 양자를 포함한다. 서브루틴 핸들러 (50) 가 MINRC 값 및 복귀 어드레스를 상이한 스택들 상으로 푸시하는 예들에서는, 서브루틴 핸들러 (50) 가 스택 저장 구조 (52) 에서의 제 1 스택에서 MINRC 값을 팝핑할 수도 있고, 스택 저장 구조 (52) 에서의 제 2 스택에서 복귀 어드레스를 팝핑할 수도 있다.

[0127] 서브루틴 핸들러 (50) 는 스택 저장 구조 (52) 로부터 팝핑되었던 복귀 어드레스를 다음 명령 생성기 (44) 에 제공할 수도 있고, 이 다음 명령 생성기 (44) 는 프로그램 카운터 (28) 로 로딩하기 위한 다음 프로그램 카운터 값으로서 프로그램 카운터 값을 선택하기 위하여 복귀 어드레스를 이용할 수도 있다. 선택된 프로그램 카운터 값은 서브루틴의 실행을 개시하였던 호출자 프로그램에서의 호출 명령 이후의 다음의 순차적인 명령을 표시

할 수도 있다.

[0128] 또한, 복귀 명령을 실행하는 것에 응답하여, 서브루틴 핸들러 (50) 는 호출자 프로그램에 대응하는 MINRC 의 저장된 상태에 기초하여 호출자 프로그램의 실행을 제어하도록 프로세싱 시스템 (10) 을 구성할 수도 있다. 예를 들어, 서브루틴 핸들러 (50) 는 스택 저장 구조 (52) 로부터 팝핑된 MINRC 값의 저장된 상태에 대응하는 값으로 MINRC 레지스터 (42) 에 저장된 MINRC 값을 겹쳐쓰기할 수도 있다.

[0129] 스택 저장 구조 (52) 는 MINRC 값들 및 복귀 어드레스들에 대한 저장을 제공하도록 구성된다. 일부의 예들에서, 스택 저장 구조 (52) 는 복귀 어드레스들을 저장하도록 구성되는 제 1 스택 저장 구조, 및 MINRC 값들을 저장하도록 구성되는 제 2 스택 저장 구조를 포함할 수도 있다. 이러한 예들에서, 제 1 스택 저장 구조는 제 2 스택 저장 구조와는 상이할 수도 있다. 추가적인 예들에서, 스택 저장 구조 (52) 는 스택 프레임들을 저장하도록 구성되는 스택 저장 구조를 포함할 수도 있고, 여기서, 각각의 스택 프레임은 특별한 호출 명령에 대응하는 복귀 어드레스 및 특별한 호출 명령에 대응하는 MINRC 값을 포함할 수도 있다.

[0130] 스택 저장 구조는 후입선출 (Last In First Out; LIFO) 프로세싱 방식에 따라 데이터를 저장 및 취출하도록 구성될 수도 있다. LIFO 프로세싱 방식에 따르면, 스택 저장 구조가 스택 저장 구조로부터 데이터 유닛 (예를 들어, 스택 프레임, 복귀 어드레스, MINRC 값, 등) 을 취출하기 위한 요청 (예를 들어, 팝 요청) 을 수신할 때마다, 스택 저장 구조는 스택 상에 저장되었던 가장 최근 데이터 유닛 (예를 들어, 스택 상으로 푸시되어야 할 가장 최근 데이터 유닛) 을 반환할 수도 있다. 스택 저장 구조는 푸시 커맨드 (command) 들 및 팝 커맨드들을 프로세싱하도록 구성될 수도 있고, 이 커맨드들은 데이터 유닛들을 저장 또는 취출할 것인지를 특정하지만, 데이터 유닛들을 저장하거나 데이터 유닛들을 취출하기 위한 특별한 데이터 어드레스를 반드시 특정하지는 않는다.

[0131] 일부의 예들에서, 스택 저장 구조 (52) 는 하드웨어-기반 스택 저장 구조 (52) 일 수도 있다. 예를 들어, 스택 저장 구조 (52) 는 하나 이상의 레지스터들 및/또는 시프트 레지스터들로서 구현될 수도 있다. 스택 저장 구조 (52) 는 제어 흐름 모듈 (34) 과 동일한 프로세서 상에서 구현되는 것으로 도 3 에 도시되지만, 다른 예들에서는, 스택 저장 구조 (52) 의 전부 또는 일부가 온-칩 캐시 또는 외부 메모리 디바이스에서 구현될 수도 있다. 예를 들어, 스택 저장 구조 (52) 는 유한한 수의 스택 저장 슬롯들을 위한 저장 공간을 포함할 수도 있다. 오버플로우 (overflow) 조건이 이러한 예에서 발생하는 경우, 서브루틴 핸들러 (50) 는 스택 저장 구조 (52) 에 포함된 유한한 수의 스택 저장 슬롯들을 통해 그리고 이 스택 저장 슬롯들 위에 추가적인 데이터를 저장하기 위하여 온-칩 캐시 또는 외부 메모리를 활용할 수도 있다.

[0132] 도 4 는 본 개시물의 서브루틴 실행 기법들을 위한 일 예의 제어 흐름을 예시하는 개념도이다. 도 4 에 도시된 바와 같이, 호출자 프로그램 공간 (60) 은 호출자 프로그램 모듈과 연관되고, 피호출자 프로그램 공간 (62) 은 서브루틴 프로그램 모듈과 연관된다. 호출자 프로그램 모듈은 메인 프로그램 또는 호출자 서브루틴의 어느 하나일 수도 있다. 호출자 프로그램 공간 (60) 은 시작 명령 (64) 으로 시작하고 종료 명령 (66) 으로 종료되는 프로그램 명령들의 시퀀스를 포함한다. 프로그램 명령들의 시퀀스에서 시작 명령 (64) 과 종료 명령 (66) 사이에는 호출 명령 (68) 이 있고, 이것은 호출자 프로그램 모듈로부터 피호출자 서브루틴 모듈로 제어를 전달하도록 프로세싱 시스템 (10) 에 지시한다. 프로그램 명령들의 시퀀스에서의 호출 명령 (68) 직후의 명령은 명령 (70) 이고, 이것은 임의의 타입의 명령일 수도 있다. 피호출자 프로그램 공간 (62) 은 시작 명령 (72) 으로 시작하고 복귀 명령 (74) 으로 종료되는 프로그램 명령들의 시퀀스를 포함한다. 복귀 명령 (74) 은 피호출자 서브루틴 모듈로부터 다시 호출자 프로그램 모듈로 제어를 전달하도록 프로세싱 시스템 (10) 에 지시한다. 시작 명령 (64) 은 호출자 프로그램 모듈의 진입 포인트로서 본원에서 지칭될 수도 있고, 시작 명령 (72) 은 피호출자 서브루틴 모듈의 진입 포인트로서 본원에서 지칭될 수도 있다.

[0133] 호출자 프로그램 모듈은 시작 명령 (64) 으로 명령들을 실행하는 것을 시작하고, 호출 명령 (68) 이 조우될 때까지 호출자 프로그램 공간 (60) 에서 명령들을 실행하는 것을 계속한다. 호출 명령 (68) 은 호출자 프로그램 모듈로부터 피호출자 서브루틴으로 제어를 전달하도록 프로세싱 시스템 (10) 에 지시한다. 호출 명령 (68) 은 시작 명령 (72) 에 대응하는 프로그램 카운터 값을 표시하는 값을 포함한다. 호출 명령 (68) 을 실행하는 것에 응답하여, 명령 (70) (즉, 호출자 프로그램에서의 호출 명령 (68) 이후의 다음의 순차적인 명령) 에 대응하는 복귀 어드레스는 호출 스택 상으로 푸시된다 (즉, 에 저장된다). 또한, 호출 명령 (68) 을 실행하는 것에 응답하여, 호출자 프로그램 모듈에 대응하는 MINRC 값은 스택 상으로 푸시된다. 또한, 호출 명령 (68) 을 실행하는 것에 응답하여, 피호출자 서브루틴 프로그램 모듈에 대응하는 MINRC 는 디폴트 값으로 초기화된다. 또한, 호출 명령 (68) 을 실행하는 것에 응답하여, 프로세싱 시스템 (10) 을 위한 프로그램 카운

터는 피호출자 서브루틴의 시작 명령 (72) 에 대응하는 프로그램 카운터 값으로 로딩된다. 시작 명령 (72) 에 대응하는 프로그램 카운터 값은 호출 명령 (68) 에 포함된 값에 의해 특정될 수도 있다.

[0134] 피호출자 서브루틴 모듈은 시작 명령 (72) 으로 명령들을 실행하는 것을 시작하고, 복귀 명령 (74) 이 조우될 때까지 피호출자 프로그램 공간 (62) 에서 명령들을 실행하는 것을 계속한다. 복귀 명령 (74) 을 실행하는 것에 응답하여, 복귀 어드레스는 호출 스택에서 팝핑되고 (즉, 으로부터 취출되고), 프로그램 카운터로 로딩된다. 다시, 복귀 어드레스는 호출자 프로그램 공간 (60) 에서의 명령 (70) 에 대응한다. 또한, 복귀 명령 (74) 을 실행하는 것에 응답하여, 호출자 프로그램 모듈에 대응하는 MINRC 값은 스택에서 팝핑된다. 호출자 프로그램 모듈은 프로그램의 종료를 표시하는 종료 명령 (66) 이 조우될 때까지 명령 (70) 으로 명령들을 실행하는 것을 재개한다.

[0135] 도 5 는 본 개시물의 서브루틴 실행 기법들을 위한 또 다른 예의 제어 흐름을 예시하는 개념도이다. 호출자 프로그램 모듈 (76) 및 피호출자 프로그램 모듈 (78) 이 도 5 에 도시되어 있다. 호출자 프로그램 모듈 (76) 및 피호출자 프로그램 모듈 (78) 은 동일한 프로그램의 일부일 수도 있고, 피호출자 프로그램 모듈 (78) 은 프로그램의 서브루틴일 수도 있다. 호출자 프로그램 모듈 (76) 은 메인 프로그램 모듈 또는 서브루틴 프로그램 모듈일 수도 있다. 호출자 프로그램 모듈 (76) 은 호출 명령 (80), 및 호출 명령 (80) 직후의 다음 명령 (82) 을 포함한다. 호출 명령 (80) 및 다음 명령 (82) 의 각각의 실행 이전에, 재개 검사 동작들 (84 및 86) 이 각각 수행된다. 실행 동안에, 호출 명령 (80) 은 후속의 명령들의 실행을 위하여 프로세싱 시스템의 제어를 피호출자 프로그램 모듈 (78) 로 전달할 수도 있다. 최종 명령이 피호출자 프로그램 모듈 (78) 에서 실행된 후, 제어는 다음 명령 (82) 에 대한 재개 검사 동작 (86) 이전의 포인트에서 호출자 프로그램 모듈 (76) 로 다시 전달될 수도 있다.

[0136] 도 5 의 포인트들 A 및 B 는 프로그램의 실행 동안의 시간에서의 2 개의 상이한 포인트들을 도시한다. 포인트 A 는 호출 명령 (80) 에 대한 재개 검사 동작 (84) 의 완료 후, 그리고 호출자 프로그램 모듈 (76) 로부터 피호출자 프로그램 모듈 (78) 로의 제어의 전달 이전인, 호출 명령 (80) 의 명령 사이클 동안에 있는 시간에서의 포인트를 정의한다. 포인트 B 는 다음 명령 (82) 에 대한 재개 검사 동작 (86) 의 개시 이전, 그리고 다시 피호출자 프로그램 모듈 (78) 로부터 호출자 프로그램 모듈 (76) 로의 제어의 전달 후인, 다음 명령 (82) 의 명령 사이클 동안에 있는 시간에서의 포인트를 정의한다.

[0137] 포인트 A 에서 활성인 스레드들은 피호출자 프로그램 모듈의 실행을 개시하기 직전에 활성인 스레드들의 세트를 정의한다. 유사하게, 포인트 B 에서 활성인 스레드들은 피호출자 프로그램 모듈 (78) 의 실행을 종료한 직후에 활성인 스레드들의 세트를 정의한다.

[0138] 서브루틴 수렴 (subroutine convergence) 은 서브루틴의 실행을 개시하기 직전의 프로세싱 시스템 (10) 에서의 스레드들의 전부에 대한 스레드 상태가 서브루틴의 실행을 종료한 직후의 스레드 상태와 동일하다는 속성을 지칭한다. 즉, 스레드가 도 5 의 포인트 A 에서 활성인 경우, 서브루틴 수렴은 스레드가 또한 도 5 의 포인트 B 에서 활성이어야 함을 요구한다. 유사하게, 스레드가 도 5 의 포인트 A 에서 비활성인 경우, 서브루틴 수렴은 스레드가 또한 도 5 의 포인트 B 에서 비활성이어야 함을 요구한다. 일부의 예들에서, 본 개시물의 기법들은 서브루틴 수렴을 보장가능할 수도 있어, 프로세싱 시스템 (10) 에서 서브루틴 명령문들의 안정적인 동작을 보장할 수도 있다.

[0139] 일반적으로, 서브루틴 환경에서 적당한 동작을 보장하기 위하여, 호출자와 피호출자 사이의 제어 흐름은 호출 및 복귀 명령들을 통해 전달되어야 한다. 다시 말해서, 호출자는 호출 명령 이외에 피호출자로 분기 또는 점프할 수 없고, 피호출자는 복귀 명령 이외에 호출자로 분기 또는 점프할 수 없다. 이것은, 호출 및 복귀 명령들이 프로그램 카운터를 수정하는 것 뿐만 아니라, 예를 들어, 복귀 어드레스들 및/또는 다른 변수들을 스택 상으로 푸시 및 팝핑함으로써 시스템의 상태를 수정하기도 하기 때문이다. 동적 호출자-피호출자 분기가 호출 및 복귀 명령들 이외에 허용되었을 경우, 시스템 상태는 정확한 것으로 보장되지 않을 것이며, 스레드들의 모두가 실행을 완료하지 않고 프로그램이 조기에 종결될 수 있는 것이 가능하다. 그러나, 도 6 에 대하여 이하에서 설명된 바와 같이, 본 개시물의 프로그램 모듈-특정 MINRC 기법들은 이러한 동적 호출자-피호출자 분기들이 호출 및 복귀 명령들 이외에 발생하는 것을 방지한다.

[0140] 도 6 은 본 개시물의 기법들에 따라 일 예의 프로그램 공간 배치들 (88, 90) 을 예시하는 개념도이다. 프로그램 공간 배치들 (88, 90) 의 각각은 호출자 프로그램 및 피호출자 프로그램을 포함한다. 도 6 에 도시된 바와 같이, 호출자 및 피호출자에 대한 프로그램 공간들은 프로그램 메모리에서 중첩되지 않는다. 그러므로, 도 3 에 도시된 바와 같이, 호출자 프로그램 공간은 피호출자 프로그램 공간의 위 또는 아래 중의 어

느 하나에 있다. 호출자 프로그램 공간이 피호출자 프로그램 공간의 위에 있는 좌측의 경우를 고려한다.

도 6 은 호출자 프로그램 공간에서 3 개의 상이한 명령들 (A, B, C) 을 예시한다. 명령 B 는 호출자 프로그램에 대한 명령들의 시퀀스에서 명령 A 이후에 발생하고, 명령 C 는 호출자 프로그램에 대한 명령들의 시퀀스에서 명령 B 이후에 발생한다. 명령 A 로, 호출자 프로그램이 발산 분기 명령의 실행에 응답하여 하나 이상의 스레드들을 비활성화한다고 가정하고, 비활성화된 스레드들에 대한 MINRC 가 명령 C 를 지시한다고 가정한다. 명령 C 를 실행하기 이전에, 호출자 프로그램은 호출 명령인 명령 B 를 실행한다. 명령 B 를 실행하는 것에 응답하여, 제어는 피호출자 서브루틴으로 전달된다.

[0141] 동일한 MINRC 가 호출자 프로그램의 실행을 제어하기 위해, 그리고 피호출자 서브루틴의 실행을 제어하기 위해 이용되는 경우를 고려한다. 이러한 경우에 있어서, 피호출자 서브루틴이 균일하게 충족되는 임의의 순방향 점프 명령들 또는 순방향 분기 명령들을 포함하는 경우, 이러한 명령들은 MINRC 및 타겟 명령 중의 더 작은 것으로 점프하도록 구성된다. 이 경우, MINRC 는 호출자 프로그램 공간에서의 명령 (즉, 명령 C) 을 지시하므로, MINRC 는 피호출자 프로그램 공간에서의 임의의 타겟 프로그램 명령보다 항상 더 작다. 그러므로, 균일하게 충족되는 순방향 점프 또는 순방향 분기 명령은 피호출자 프로그램으로 하여금 복귀 명령 이외에 호출자 프로그램 공간으로 분기하게 할 것이다. 위에서 논의된 바와 같이, 이러한 분기는 시스템의 적당한 동작을 보장하지 않을 것이다.

[0142] 그러나, 본 개시물의 프로그램 모듈-특정 MINRC 기법들은 호출자-피호출자 분기가 호출 및 복귀 명령들 이외에 발생하는 것을 방지한다. 예를 들어, 피호출 프로그램에 대한 서브루틴-특정 MINRC 는 이 예에서 명령 C 를 지시하는 값을 저장하지 않을 것이다. 실제로, 서브루틴의 실행이 시작될 때, 서브루틴의 실행을 제어하기 위해 이용되는 새로운 서브루틴-특정 MINRC 가 초기화된다. 서브루틴-특정 MINRC 는 서브루틴의 실행이 개시될 때에 활성인 모든 스레드들 중의 가장 작은 재개 카운터 값을 표시한다. 서브루틴의 실행이 시작되었을 때, 명령 C 를 지시하는 재개 카운터 값들을 가지는 스레드들이 이미 비활성화되었으므로, 이러한 재개 카운터 값들은 서브루틴-특정 MINRC 에 영향을 주지 않을 것이다. 이러한 방법으로, 본 개시물의 기법들은 호출자와 피호출자 사이의 제어 흐름이 호출 및 복귀 명령들을 통해 전달되는 것을 보장한다.

[0143] 도 7 내지 도 18 은 본 개시물의 서브루틴 실행 기법들을 활용하는 일 예의 명령 프로세싱 기법들을 예시하는 흐름도들이다. 일부의 예들에서는, 도 7 내지 도 18 에 도시된 일 예의 기법들이 도 2 및 도 3 중의 어느 하나의 제어 흐름 유닛 (34) 에서, 및/또는 도 1 내지 도 3 중의 임의의 것의 프로세싱 시스템 (10) 내에서 구현될 수도 있다. 설명의 용이함을 위하여, 기법들은 도 2 에 도시된 제어 흐름 유닛 (34) 에 대하여 설명될 것이지만, 기법들은 동일하거나 상이한 구성인 동일하거나 상이한 컴포넌트들을 갖는 다른 시스템들에서 수행될 수도 있다는 것을 이해해야 한다.

[0144] 도 7 은 본 개시물에 따라 실행되어야 할 다음 명령에 대한 프로그램 카운터 값을 결정하기 위한 일 예의 기법을 예시하는 흐름도이다. 제어 흐름 모듈 (34) 은 명령, 및/또는 명령과 연관된 제어 정보를 수신한다 (100). 제어 흐름 모듈 (34) 은 명령이 제어 흐름 명령인지 여부를 결정한다 (102). 명령이 제어 흐름 명령이 아닌 것으로 결정하는 것에 응답하여, 제어 흐름 모듈 (34) 은 프로그램 카운터 (28) 를 증분시킨다 (104). 예를 들어, 제어 흐름 모듈 (34) 은 다음의 순차적인 명령을 표시하는, 프로그램 카운터 (28) 로 로딩하기 위한 프로그램 카운터 값을 선택할 수도 있다.

[0145] 다른 한편으로, 명령이 제어 흐름 명령인 것으로 결정하는 것에 응답하여, 제어 흐름 모듈 (34) 은 제어 흐름 명령이 서브루틴 명령인지 여부를 결정한다 (106). 일 예의 서브루틴 명령들은 호출 명령 및 복귀 명령을 포함할 수도 있다. 명령이 서브루틴 명령인 것으로 결정하는 것에 응답하여, 제어 흐름 모듈 (34) 은 명령이 호출 명령인지 여부를 결정한다 (108). 명령이 호출 명령인 것으로 결정하는 것에 응답하여, 제어 흐름 모듈 (34) 은 도 8 에 설명된 기법들에 따라 호출 명령을 프로세싱한다 (110). 다른 한편으로, 명령이 호출 명령이 아닌 것으로 결정하는 것 (즉, 명령이 복귀 명령인 것으로 결정하는 것) 에 응답하여, 제어 흐름 모듈 (34) 은 도 9 에 설명된 기법들에 따라 복귀 명령을 프로세싱한다 (112).

[0146] 판정 박스 (106) 로 복귀하면, 명령이 서브루틴 명령이 아닌 것으로 결정하는 것에 응답하여, 제어 흐름 모듈 (34) 은 제어 흐름 명령이 점프 명령인지 여부를 결정한다 (114). 점프 명령은 대안적으로 무조건부 분기 명령으로서 지칭될 수도 있다. 제어 흐름 명령이 점프 명령인 것으로 결정하는 것에 응답하여, 제어 흐름 모듈 (34) 은 도 10 에 설명된 기법들에 따라 점프 명령을 프로세싱한다 (116). 이와 다르게, 제어 흐름 명령이 점프 명령이 아닌 것으로 결정하는 것 (즉, 제어 흐름 명령이 조건부 분기 명령인 것으로 결정하는 것) 에 응답하여, 제어 흐름 모듈 (34) 은 도 12 및 도 13 에 설명된 기법들에 따라 조건부 분기 명령을 프로세싱한다

(118).

[0147] 도 8 은 본 개시물의 프로그램 모듈-특정 MINRC 실행 기법들에 따라 호출 명령을 프로세싱하기 위한 일 예의 기법을 예시하는 흐름도이다. 제어 흐름 모듈 (34) 은 호출자 프로그램과 연관된 MINRC 에 대해 MINRC 레지스터에 저장된 값을 MINRC 스택 상으로 푸시한다 (120). 제어 흐름 모듈 (34) 은 MINRC 레지스터를 서브루틴 프로그램 모듈에 대응하는 초기 값으로 초기화한다 (122). 다시 말해서, 제어 흐름 모듈 (34) 은 피호출자 프로그램과 연관된 MINRC 에 대한 초기 값으로 MINRC 레지스터에 저장된 값을 겹쳐쓰기할 수도 있다. 제어 흐름 모듈 (34) 은 복귀 어드레스를 호출 스택 상으로 푸시한다 (124). 복귀 어드레스는 호출 명령 직후에 발생하는 호출자 프로그램에서의 다음의 순차적인 명령을 표시할 수도 있다. 제어 흐름 모듈 (34) 은 프로그램 카운터를, 타겟 명령 (즉, 타겟 프로그램 카운터 값) 을 표시하는 값으로 설정한다 (126). 타겟 명령 및/또는 타겟 프로그램 카운터는 호출 명령에서 특정될 수도 있다.

[0148] 도 9 는 본 개시물의 프로그램 모듈-특정 MINRC 실행 기법들에 따라 복귀 명령을 프로세싱하기 위한 일 예의 기법을 예시하는 흐름도이다. 제어 흐름 모듈 (34) 은 호출 스택이 비어 있는지 여부를 결정한다 (128). 호출 스택이 비어 있는 것으로 결정하는 것에 응답하여, 제어 흐름 모듈 (34) 은 프로세스를 종결시킨다 (130). 호출 스택이 비어 있는 경우, 이것은 복귀 명령이 최상위-레벨 프로그램 (즉, 메인 프로그램) 의 종료 명령임을 의미할 수도 있다. 호출 스택이 비어 있지 않은 것으로 결정하는 것에 응답하여, 제어 흐름 모듈 (34) 은 MINRC 스택에서 MINRC 값을 팝핑하고 (132), MINRC 레지스터를 팝핑된 MINRC 값과 동일하게 설정한다 (134). 제어 흐름 모듈 (34) 은 호출 스택에서 복귀 어드레스를 팝핑하고 (136), 프로그램 카운터 레지스터를 팝핑된 복귀 어드레스와 동일하게 설정한다 (138). 복귀 어드레스는 현재 실행되고 있는 복귀 명령을 포함하는 서브루틴의 실행을 개시하였던 호출 명령 직후에 발생하는 명령을 표시할 수도 있다.

[0149] 도 10 은 본 개시물에 따라 점프 명령을 프로세싱하기 위한 일 예의 기법을 예시하는 흐름도이다. 제어 흐름 모듈 (34) 은 점프 명령이 역방향 점프 명령인지 여부를 결정한다 (140). 일부의 예들에서, 제어 흐름 모듈 (34) 은 점프 명령에 대한 타겟 프로그램 카운터 값이 점프 명령을 식별하는 프로그램 카운터 값보다 더 큰지 여부를 결정함으로써, 점프 명령이 역방향 점프 명령인지 여부를 결정할 수도 있다. 추가의 예들에서, 점프 명령에 대한 타겟 프로그램 카운터 값은 상대적인 타겟 프로그램 카운터 값일 수도 있고, 이것은 타겟 명령과, 점프 명령을 식별하는 프로그램 카운터 값과의 사이의 차이를 표시할 수도 있다. 이러한 예들에서, 제어 흐름 모듈 (34) 은 점프 명령에 대한 상대적인 타겟 프로그램 카운터 값이 제로보다 더 작은지 여부를 결정함으로써, 점프 명령이 역방향 점프 명령인지 여부를 결정할 수도 있다. 추가적인 예들에서, 순방향 및 역방향 점프 명령들은 상이한 동작 코드들, 즉, 연산코드 (opcode) 들을 포함할 수도 있다. 이러한 예들에서, 제어 흐름 모듈 (34) 은 명령의 연산코드에 기초하여, 점프 명령이 역방향 점프 명령인지 여부를 결정할 수도 있다.

[0150] 제어 흐름 모듈 (34) 이 점프 명령이 역방향 점프 명령인 것으로 결정하는 경우, 제어 흐름 모듈 (34) 은 적어도 하나의 스레드가 활성인지 여부를 결정한다 (142). 제어 흐름 모듈 (34) 이 스레드들이 활성이 아닌 것으로 결정하는 경우, 제어 흐름 모듈 (34) 은 프로그램 카운터 (28) 를 증분시킨다 (144). 예를 들어, 제어 흐름 모듈 (34) 은 다음의 순차적인 명령을 표시하는, 프로그램 카운터 (28) 로 로딩하기 위한 프로그램 카운터 값을 선택할 수도 있다. 다른 한편으로, 제어 흐름 모듈 (34) 이 적어도 하나의 스레드가 활성인 것으로 결정하는 경우, 제어 흐름 모듈 (34) 은 타겟 명령으로 점프한다 (146). 예를 들어, 제어 흐름 모듈 (34) 은 프로그램 카운터 (28) 로 로딩하기 위하여 점프 명령에 의해 식별된 타겟 명령을 표시하는 타겟 프로그램 카운터 값을 선택할 수도 있다.

[0151] 제어 흐름 모듈 (34) 이 점프 명령이 역방향 점프 명령이 아닌 것 (즉, 점프 명령이 순방향 점프 명령인 것) 으로 결정하는 경우, 제어 흐름 모듈 (34) 은 타겟 프로그램 카운터 값이 MINRC 값 이하인지 여부를 결정한다 (148). 제어 흐름 모듈 (34) 이 타겟 프로그램 카운터 값이 MINRC 값 이하가 아닌 것으로 결정하는 경우, 제어 흐름 모듈 (34) 은 모든 활성 스레드들을 비활성화한다 (150). 일부의 예들에서, 제어 흐름 모듈 (34) 은 활성 스레드들의 전부를 비활성화하기 위하여 도 11 에서 예시된 기법을 이용할 수도 있다. 추가의 예들에서, 활성 스레드들을 비활성화하는 것은 비활성화되고 있는 각각의 스레드에 대하여, 각각의 스레드에 대한 명령들을 프로세싱하도록 배정되는 프로세싱 엘리먼트 (14) 를 비활성화 및/또는 디스에이블하는 것을 포함할 수도 있다. 제어 흐름 모듈 (34) 은 MINRC 값에 의해 식별된 명령으로 점프하도록 진행한다 (152). 예를 들어, 제어 흐름 모듈 (34) 은 타겟 프로그램 카운터 값이 MINRC 값 이하가 아닌 것으로 결정하는 것에 응답하여, 프로그램 카운터 (28) 로 로딩하기 위한 값으로서 MINRC 값을 선택할 수도 있다. 다른 한편으로, 제어 흐름 모듈 (34) 이 타겟 프로그램 카운터 값이 MINRC 값 이하인 것으로 결정하는 경우, 제어 흐름 모듈 (34)

은 타겟 명령으로 점프한다 (154). 예를 들어, 제어 흐름 모듈 (34) 은 타겟 프로그램 카운터 값이 MINRC 값 이하인 것으로 결정하는 것에 응답하여, 프로그램 카운터 (28) 로 로딩하기 위한 값으로서, 점프 명령에 의해 식별된 타겟 명령을 표시하는 타겟 프로그램 카운터 값을 선택할 수도 있다.

[0152] 이 예에서, 제어 흐름 모듈 (34) 은 더 낮은 값의 어드레스들에서의 명령들을 프로세싱하도록 스케줄링되는 발산 스레드들이 더 높은 값의 어드레스들에서의 명령들을 프로세싱하도록 스케줄링되는 스레드들 이전에 실행되는 것 (즉, "최저 값의 어드레스 우선" 스레드 프로세싱 순서) 을 보장하기 위하여, MINRC 가 타겟 프로그램 카운터 값보다 더 작을 때에 프로그램 카운터 (28) 로 로딩하기 위한 MINRC 값을 선택한다.

[0153] 도 11 은 본 개시물에 따라 모든 스레드들을 비활성화하기 위한 일 예의 기법을 예시하는 흐름도이다. 일부의 예들에서, 도 11 에서 예시된 기법은 도 10 에서 예시된 프로세스 박스 (150) 또는 도 13 에서 예시된 프로세스 박스 (198) 를 구현하기 위하여 이용될 수도 있다. 제어 흐름 모듈 (34) 은 스레드를 선택한다 (156).

제어 흐름 모듈 (34) 은 선택된 스레드가 활성인지 여부를 결정한다 (158). 제어 흐름 모듈 (34) 이 선택된 스레드가 활성인 것으로 결정하는 경우, 제어 흐름 모듈 (34) 은 스레드와 연관된 활성 플래그를 거짓 (false) 의 값으로 재설정하고 (160), 스레드와 연관된 재개 카운터를, 점프 또는 분기 명령에 의해 식별된 타겟 명령을 표시하는 타겟 프로그램 카운터 값으로 설정하고 (162), 판정 박스 (164) 로 진행한다. 다른 한편으로, 제어 흐름 모듈 (34) 이 선택된 스레드가 활성이 아닌 것으로 결정하는 경우, 제어 흐름 모듈 (34) 은 스레드에 대한 활성 플래그를 재설정하지 않고, 그리고 스레드에 대한 재개 카운터를 설정하지 않고, 판정 박스 (164) 로 진행한다. 어느 하나의 경우에 있어서, 제어 흐름 모듈 (34) 은 프로세싱하기 위한 임의의 더 많은 스레드들이 있는지 여부를 결정한다 (164). 제어 흐름 모듈 (34) 이 프로세싱하기 위한 더 많은 스레드들이 있는 것으로 결정하는 경우, 제어 흐름 모듈 (34) 은 또 다른 스레드를 프로세싱하기 위하여 프로세스 박스 (156) 로 복귀한다.

[0154] 이와 다르게, 제어 흐름 모듈 (34) 이 프로세싱하기 위한 활성 스레드들이 더 이상 없는 것으로 결정하는 경우, 제어 흐름 모듈 (34) 은 MINRC 를 업데이트한다 (166). 예를 들어, 제어 흐름 모듈 (34) 은 현재 실행하는 프로그램 모듈의 실행이 개시될 때에 활성인 스레드들에 대응하는 재개 카운터 값들의 세트로부터의 가장 작은 재개 카운터 값을 결정할 수도 있고, MINRC 를, 가장 작은 재개 카운터 값을 표시하는 값으로 설정할 수도 있다. 일부의 예들에서, 제어 흐름 모듈 (34) 은 MINRC 를 업데이트하기 위하여 도 17 및 도 18 에서 예시된 기법들의 어느 하나를 이용할 수도 있다.

[0155] MINRC 값을 업데이트한 후, 제어 흐름 모듈 (34) 은 비활성화 프로세스를 종료하고, 호출 프로세스 예를 들어, 도 10 의 프로세스 박스 (152) 또는 도 13 의 프로세스 박스 (200) 로 복귀한다. 도 11 은 스레드들의 각각을 순차적으로 비활성화함으로써 다중 스레드들을 비활성화하는 일 예의 기법을 예시하지만, 다른 예들에서는, 다중 스레드들이 예를 들어, 스트로브 (strobe) 또는 공통 제어 라인 (common control line) 을 이용함으로써 부분적으로 또는 완전히 병렬로 비활성화될 수도 있다.

[0156] 도 12 및 도 13 은 본 개시물에 따라 분기 명령을 프로세싱하기 위한 일 예의 기법을 예시하는 흐름도들이다.

제어 흐름 모듈 (34) 은 분기 명령이 역방향 분기 명령인지 여부를 결정한다 (168). 제어 흐름 모듈 (34) 이 분기 명령이 역방향 분기 명령인지 여부를 결정할 수도 있는 방식은, 점프 명령이 역방향 점프 명령인지 여부를 결정하기 위한 도 10 에 대하여 위에서 설명되었던 것과 실질적으로 유사할 수도 있고, 간결함을 위하여, 더욱 상세하게 설명되지 않을 것이다.

[0157] 제어 흐름 모듈 (34) 이 분기 명령이 역방향 분기 명령인 것으로 결정하는 경우, 제어 흐름 모듈 (34) 은 적어도 하나의 스레드가 활성인지 여부를 결정한다 (170). 제어 흐름 모듈 (34) 이 스레드들이 활성이 아닌 것으로 결정하는 경우, 제어 흐름 모듈 (34) 은 프로그램 카운터 (28) 를 증분시킨다 (172). 예를 들어, 제어 흐름 모듈 (34) 은 다음의 순차적인 명령을 표시하는, 프로그램 카운터 (28) 로 로딩하기 위한 프로그램 카운터 값을 선택할 수도 있다. 이 예에서, 제어 흐름 모듈 (34) 은 더 낮은 값의 어드레스들에서의 명령들을 프로세싱하도록 스케줄링되는 발산 스레드들이 더 높은 값의 어드레스들에서의 명령들을 프로세싱하도록 스케줄링되는 스레드들 이전에 실행되는 것을 보장하기 위하여, 최저 값의 재개 카운터가 검출될 때까지 프로그램 카운터 값들을 통해 순차적으로 순환할 수도 있다.

[0158] 다른 한편으로, 제어 흐름 모듈 (34) 이 적어도 하나의 스레드가 활성인 것으로 결정하는 경우, 제어 흐름 모듈 (34) 은 발산 조건이 균일한지 여부 (즉, 분기 조건이 균일하게 충족되는지 또는 균일하게 충족되지 않는지 여부) 를 결정한다 (174). 제어 흐름 모듈 (34) 이 발산 조건이 균일하지 않은 것 (즉, 분기가 발산하는 것) 으로 결정하는 경우, 제어 흐름 모듈 (34) 은 분기 조건을 충족하지 않는 임의의 활성 스레드들을 비활성화할

수도 있다 (176). 일부의 예들에서, 제어 흐름 모듈 (34) 은 분기 조건을 충족하지 않는 활성 스레드들을 비활성화하기 위하여 도 14 에서 예시된 기법을 이용할 수도 있다. 추가의 예들에서, 분기 조건을 충족하지 않는 활성 스레드들을 비활성화하는 것은 비활성화되고 있는 각각의 스레드에 대하여, 각각의 스레드에 대한 명령들을 프로세싱하도록 배정되는 프로세싱 엘리먼트 (14) 를 비활성화 및/또는 디스에이블하는 것을 포함할 수도 있다. 제어 흐름 모듈 (34) 은 타겟 명령으로 점프하도록 진행한다 (178). 예를 들어, 제어 흐름 모듈 (34) 은 프로그램 카운터 (28) 로 로딩하기 위하여 분기 명령에 의해 식별된 타겟 명령을 표시하는 타겟 프로그램 카운터 값을 선택할 수도 있다.

[0159] 이 예에서, 제어 흐름 모듈 (34) 은 더 낮은 값의 어드레스들에서의 명령들을 프로세싱하도록 스케줄링되는 발산 스레드들이 더 높은 값의 어드레스들에서의 명령들을 프로세싱하도록 스케줄링되는 스레드들 이전에 실행되는 것 (즉, "최저 값의 어드레스 우선" 스레드 프로세싱 순서) 을 보장하기 위하여, 분기 조건을 충족하지 않는 스레드들을 비활성화한다. 더욱 구체적으로, 분기 조건을 충족하지 않는 활성 스레드들은 다음의 순차적인 명령을 실행하도록 스케줄링되고, 다음의 순차적인 명령에 대한 프로그램 카운터 값은 타겟 명령과 연관된 타겟 프로그램 카운터 값보다 더 크다. 따라서, 역방향 분기 명령에서는, 분기 조건을 충족하는 활성 스레드들이 분기 조건을 충족하지 않는 스레드들 이전에 실행되도록 스케줄링된다.

[0160] 판정 박스 (174) 로 복귀하면, 제어 흐름 모듈 (34) 이 발산 조건이 균일한 것으로 결정하는 경우, 제어 흐름 모듈 (34) 은 분기 조건이 충족되는지 여부를 결정한다 (180). 제어 흐름 모듈 (34) 이 분기 조건이 충족되지 않는 것으로 결정하는 경우, 제어 흐름 모듈 (34) 은 프로그램 카운터 (28) 를 증분시킨다 (182). 예를 들어, 제어 흐름 모듈 (34) 은 다음의 순차적인 명령을 표시하는, 프로그램 카운터 (28) 로 로딩하기 위한 프로그램 카운터 값을 선택할 수도 있다. 이 경우, 모든 활성 스레드들은 균일하게 충족되지 않은 분기 조건으로 인해 다음의 순차적인 명령을 실행하도록 스케줄링되므로, 제어 흐름 모듈 (34) 은 프로그램 카운터 (28) 를 증분시킨다. 다른 한편으로, 제어 흐름 모듈 (34) 이 분기 조건이 충족되는 것으로 결정하는 경우, 제어 흐름 모듈 (34) 은 타겟 명령으로 점프한다 (184). 예를 들어, 제어 흐름 모듈 (34) 은 프로그램 카운터 (28) 로 로딩하기 위하여 분기 명령에 의해 식별된 타겟 명령을 표시하는 타겟 프로그램 카운터 값을 선택할 수도 있다. 이 경우, 모든 활성 스레드들은 균일하게 충족된 분기 조건으로 인해 타겟 명령을 실행하도록 스케줄링되므로, 제어 흐름 모듈 (34) 은 타겟 명령으로 점프한다.

[0161] 판정 박스 (168) 로 복귀하면, 제어 흐름 모듈 (34) 이 분기 명령이 역방향 분기 명령이 아닌 것 (즉, 분기 명령이 순방향 분기 명령인 것) 으로 결정하는 경우, 제어 흐름 모듈 (34) 은 도 13 의 판정 박스 (186) 로 진행하고, 여기서, 제어 흐름 모듈 (34) 은 발산 조건이 균일한지 여부 (즉, 분기 조건이 균일하게 충족되는지 또는 균일하게 충족되지 않는지 여부) 를 결정한다 (186). 제어 흐름 모듈 (34) 이 발산 조건이 균일하지 않은 것 (즉, 분기가 발산하는 것) 으로 결정하는 경우, 제어 흐름 모듈 (34) 은 분기 조건을 충족하는 임의의 활성 스레드들을 비활성화할 수도 있다 (188). 일부의 예들에서, 제어 흐름 모듈 (34) 은 분기 조건을 충족하는 활성 스레드들을 비활성화하기 위하여 도 15 에서 예시된 기법을 이용할 수도 있다. 추가의 예들에서, 분기 조건을 충족하는 활성 스레드들을 비활성화하는 것은 비활성화되고 있는 각각의 스레드에 대하여, 각각의 스레드에 대한 명령들을 프로세싱하도록 배정되는 프로세싱 엘리먼트 (14) 를 비활성화 및/또는 디스에이블하는 것을 포함할 수도 있다. 제어 흐름 모듈 (34) 은 프로그램 카운터 (28) 를 증분시킨다 (190).

[0162] 이 예에서, 제어 흐름 모듈 (34) 은 더 낮은 값의 어드레스들에서의 명령들을 프로세싱하도록 스케줄링되는 발산 스레드들이 더 높은 값의 어드레스들에서의 명령들을 프로세싱하도록 스케줄링되는 스레드들 이전에 실행되는 것 (즉, "최저 값의 어드레스 우선" 스레드 프로세싱 순서) 을 보장하기 위하여, 분기 조건을 충족하는 스레드들을 비활성화한다. 더욱 구체적으로, 분기 조건을 충족하지 않는 활성 스레드들은 다음의 순차적인 명령을 실행하도록 스케줄링되고, 다음의 순차적인 명령에 대한 프로그램 카운터 값은 타겟 명령과 연관된 타겟 프로그램 카운터 값보다 더 작다. 따라서, 순방향 분기 명령에서는, 분기 조건을 충족하지 않는 활성 스레드들이 분기 조건을 충족하는 스레드들 이전에 실행되도록 스케줄링된다.

[0163] 판정 박스 (186) 로 복귀하면, 제어 흐름 모듈 (34) 이 발산 조건이 균일한 것으로 결정하는 경우, 제어 흐름 모듈 (34) 은 분기 조건이 충족되는지 여부를 결정한다 (192). 제어 흐름 모듈 (34) 이 분기 조건이 충족되지 않는 것으로 결정하는 경우, 제어 흐름 모듈 (34) 은 프로그램 카운터 (28) 를 증분시킨다 (194). 예를 들어, 제어 흐름 모듈 (34) 은 다음의 순차적인 명령을 표시하는, 프로그램 카운터 (28) 로 로딩하기 위한 프로그램 카운터 값을 선택할 수도 있다. 이 경우, 모든 활성 스레드들은 균일하게 충족되지 않은 분기 조건으로 인해 다음의 순차적인 명령을 실행하도록 스케줄링되므로, 제어 흐름 모듈 (34) 은 프로그램 카운터 (28) 를

증분시킨다.

[0164] 다른 한편으로, 제어 흐름 모듈 (34) 이 분기 조건이 충족되는 것으로 결정하는 경우, 제어 흐름 모듈 (34) 은 타겟 프로그램 카운터 값이 MINRC 값 이하인지 여부를 결정한다 (196). 제어 흐름 모듈 (34) 이 타겟 프로그램 카운터 값이 MINRC 값 이하가 아닌 것으로 결정하는 경우, 제어 흐름 모듈 (34) 은 모든 활성 스레드들을 비활성화한다 (198). 일부의 예들에서, 제어 흐름 모듈 (34) 은 활성 스레드들의 전부를 비활성화하기 위하여 도 11 에서 예시된 기법을 이용할 수도 있다. 추가의 예들에서, 활성 스레드들을 비활성화하는 것은 비활성화되고 있는 각각의 스레드에 대하여, 각각의 스레드에 대한 명령들을 프로세싱하도록 배정되는 프로세싱 엘리먼트 (14) 를 비활성화 및/또는 디스에이블하는 것을 포함할 수도 있다. 제어 흐름 모듈 (34) 은 MINRC 값에 의해 식별된 명령으로 점프하도록 진행한다 (200). 예를 들어, 제어 흐름 모듈 (34) 은 타겟 프로그램 카운터 값이 MINRC 값 이하가 아닌 것으로 결정하는 것에 응답하여, 프로그램 카운터 (28) 로 로딩하기 위한 값으로서 MINRC 값을 선택할 수도 있다. 다른 한편으로, 제어 흐름 모듈 (34) 이 타겟 프로그램 카운터 값이 MINRC 값 이하인 것으로 결정하는 경우, 제어 흐름 모듈 (34) 은 타겟 명령으로 점프한다 (202). 예를 들어, 제어 흐름 모듈 (34) 은 타겟 프로그램 카운터 값이 MINRC 값 이하인 것으로 결정하는 것에 응답하여, 프로그램 카운터 (28) 로 로딩하기 위한 값으로서, 분기 명령에 의해 식별된 타겟 명령을 표시하는 타겟 프로그램 카운터 값을 선택할 수도 있다.

[0165] 이 예에서, 제어 흐름 모듈 (34) 은 더 낮은 값의 어드레스들에서의 명령들을 프로세싱하도록 스케줄링되는 발산 스레드들이 더 높은 값의 어드레스들에서의 명령들을 프로세싱하도록 스케줄링되는 스레드들 이전에 실행되는 것 (즉, "최저 값의 어드레스 우선" 스레드 프로세싱 순서) 을 보장하기 위하여, MINRC 가 타겟 프로그램 카운터 값보다 더 작을 때에 프로그램 카운터 (28) 로 로딩하기 위한 MINRC 값을 선택한다.

[0166] 도 14 는 본 개시물에 따라 분기 조건을 충족하지 않는 활성 스레드들을 비활성화하기 위한 일 예의 기법을 예시하는 흐름도이다. 일부의 예들에서, 도 14 에서 예시된 기법은 도 12 에서 예시된 프로세스 박스 (176) 를 구현하기 위하여 이용될 수도 있다. 제어 흐름 모듈 (34) 은 활성 스레드를 선택한다 (204). 제어 흐름 모듈 (34) 은 분기 조건이 선택된 스레드에 대해 충족되는지 여부를 결정한다 (206). 제어 흐름 모듈 (34) 이 분기 조건이 선택된 스레드에 대해 충족되지 않는 것으로 결정하는 경우, 제어 흐름 모듈 (34) 은 스레드와 연관된 활성 플래그를 거짓 (false) 의 값으로 재설정하고 (208), 스레드와 연관된 재개 카운터를, 다음의 순차적인 명령을 표시하는 프로그램 카운터 값 (예를 들어, "PC +1") 으로 설정하고 (210), 판정 박스 (212) 로 진행한다. 이 경우, 분기 조건은 스레드에 대해 충족되지 않았으므로, 재개 카운터는 다음의 순차적인 명령을 표시하는 값으로 설정된다.

[0167] 다른 한편으로, 제어 흐름 모듈 (34) 이 분기 조건이 선택된 스레드에 대해 충족되는 것으로 결정하는 경우, 제어 흐름 모듈 (34) 은 스레드에 대한 활성 플래그를 재설정하지 않고, 그리고 스레드에 대한 재개 카운터를 설정하지 않고, 판정 박스 (212) 로 진행한다. 어느 하나의 경우에 있어서, 제어 흐름 모듈 (34) 은 프로세싱하기 위한 임의의 더 많은 활성 스레드들이 있는지 여부를 결정한다 (212). 제어 흐름 모듈 (34) 이 프로세싱하기 위한 더 많은 활성 스레드들이 있는 것으로 결정하는 경우, 제어 흐름 모듈 (34) 은 또 다른 활성 스레드를 프로세싱하기 위하여 프로세스 박스 (204) 로 복귀한다. 이와 다르게, 제어 흐름 모듈 (34) 이 프로세싱하기 위한 활성 스레드들이 더 이상 없는 것으로 결정하는 경우, 제어 흐름 모듈 (34) 은 MINRC 를 업데이트한다 (214). 제어 흐름 모듈 (34) 은 도 11 의 프로세스 박스 (166) 에 대하여 위에서 설명되는 것과 유사한 방식으로 MINRC 를 업데이트할 수도 있다. MINRC 를 업데이트한 후, 제어 흐름 모듈 (34) 은 비활성화 프로세스를 종료하고, 호출 프로세스 예를 들어, 도 12 의 프로세스 박스 (178) 로 복귀한다.

[0168] 도 15 는 본 개시물에 따라 분기 조건을 충족하는 활성 스레드들을 비활성화하기 위한 일 예의 기법을 예시하는 흐름도이다. 일부의 예들에서, 도 15 에서 예시된 기법은 도 13 에서 예시된 프로세스 박스 (188) 를 구현하기 위하여 이용될 수도 있다. 제어 흐름 모듈 (34) 은 활성 스레드를 선택한다 (216). 제어 흐름 모듈 (34) 은 분기 조건이 선택된 스레드에 대해 충족되는지 여부를 결정한다 (218). 제어 흐름 모듈 (34) 이 분기 조건이 선택된 스레드에 대해 충족되는 것으로 결정하는 경우, 제어 흐름 모듈 (34) 은 스레드와 연관된 활성 플래그를 거짓의 값으로 재설정하고 (220), 스레드와 연관된 재개 카운터를, 분기 명령에 의해 식별된 타겟 명령을 표시하는 타겟 프로그램 카운터 값으로 설정하고 (222), 판정 박스 (224) 로 진행한다. 이 경우, 분기 조건은 스레드에 대해 충족되었으므로, 재개 카운터는 타겟 명령을 표시하는 값으로 설정된다.

[0169] 다른 한편으로, 제어 흐름 모듈 (34) 이 분기 조건이 선택된 스레드에 대해 충족되지 않는 것으로 결정하는 경우, 제어 흐름 모듈 (34) 은 스레드에 대한 활성 플래그를 재설정하지 않고, 그리고 스레드에 대한 재개 카운터

를 설정하지 않고, 판정 박스 (224) 로 진행한다. 어느 하나의 경우에 있어서, 제어 흐름 모듈 (34) 은 프로세싱하기 위한 임의의 더 많은 활성 스레드들이 있는지 여부를 결정한다 (224). 제어 흐름 모듈 (34) 이 프로세싱하기 위한 더 많은 활성 스레드들이 있는 것으로 결정하는 경우, 제어 흐름 모듈 (34) 은 또 다른 활성 스레드를 프로세싱하기 위하여 프로세스 박스 (216) 로 복귀한다. 이와 다르게, 제어 흐름 모듈 (34) 이 프로세싱하기 위한 활성 스레드들이 더 이상 없는 것으로 결정하는 경우, 제어 흐름 모듈 (34) 은 MINRC 를 업데이트한다 (226). 제어 흐름 모듈 (34) 은 도 11 의 프로세스 박스 (166) 에 대하여 위에서 설명되는 것과 유사한 방식으로 MINRC 를 업데이트할 수도 있다. MINRC 를 업데이트한 후, 제어 흐름 모듈 (34) 은 비활성화 프로세스를 종료하고, 호출 프로세스 예를 들어, 도 13 의 프로세스 박스 (190) 로 복귀한다.

[0170] 도 16 은 본 개시물에 따라 스레드들을 재활성화하기 위한 일 예의 재개 검사 기법을 예시하는 흐름도이다. 일부의 예들에서, 도 16 에서 예시된 기법은 새로운 프로그램 카운터 값이 프로그램 카운터 (28) 로 로딩될 때마다 수행될 수도 있다. 제어 흐름 모듈 (34) 은 비활성 스레드를 선택한다 (228). 제어 흐름 모듈 (34) 은 비활성 스레드에 대한 재개 카운터 값이 프로그램 카운터 값과 동일한지 여부를 결정한다 (230). 제어 흐름 모듈 (34) 이 비활성 스레드에 대한 재개 카운터 값이 프로그램 카운터 값과 동일한 것으로 결정하는 경우, 제어 흐름 모듈 (34) 은 스레드와 연관된 활성 플래그를 참 (true) 의 값으로 설정하고 (232), 스레드와 연관된 재개 카운터를 최대 값으로 설정하고 (234), 판정 박스 (236) 로 진행한다. 일부의 예들에서, 최대 값은 재개 카운터를 위한 저장 슬롯 또는 레지스터에서 표현될 수 있는 가장 큰 값인 값일 수도 있다.

[0171] 다른 한편으로, 제어 흐름 모듈 (34) 이 비활성 스레드에 대한 재개 카운터 값이 프로그램 카운터 값과 동일하지 않은 것으로 결정하는 경우, 제어 흐름 모듈 (34) 은 스레드에 대한 활성 플래그를 설정하지 않고, 그리고 스레드에 대한 재개 카운터를 설정하지 않고, 판정 박스 (236) 로 진행한다. 어느 하나의 경우에 있어서, 제어 흐름 모듈 (34) 은 프로세싱하기 위한 임의의 더 많은 비활성 스레드들이 있는지 여부를 결정한다 (236). 제어 흐름 모듈 (34) 이 프로세싱하기 위한 더 많은 비활성 스레드들이 있는 것으로 결정하는 경우, 제어 흐름 모듈 (34) 은 또 다른 비활성 스레드를 프로세싱하기 위하여 프로세스 박스 (228) 로 복귀한다. 이와 다르게, 제어 흐름 모듈 (34) 이 프로세싱하기 위한 비활성 스레드들이 더 이상 없는 것으로 결정하는 경우, 제어 흐름 모듈 (34) 은 MINRC 를 업데이트한다 (238). 제어 흐름 모듈 (34) 은 도 11 의 프로세스 박스 (166) 에 대하여 위에서 설명되는 것과 유사한 방식으로 MINRC 를 업데이트할 수도 있다. MINRC 값을 업데이트한 후, 제어 흐름 모듈 (34) 은 재개 검사 프로세스를 종료한다.

[0172] 도 17 은 본 개시물에 따라 MINRC 를 업데이트하기 위한 일 예의 기법을 예시하는 흐름도이다. 일부의 예들에서, 도 17 에서 예시된 기법은 도 11 에서 예시된 프로세스 박스 (166), 도 14 에서 예시된 프로세스 박스 (214), 도 15 에서 예시된 프로세스 박스 (226), 및/또는 도 16 에서 예시된 프로세스 박스 (238) 를 구현하기 위하여 이용될 수도 있다. 일반적으로, 도 17 에서 예시된 기법은 하나 이상의 재개 카운터들을 업데이트하는 것에 응답하여 수행될 수도 있고, 이것은 재개 검사 동작의 일부로서 하나 이상의 스레드들을 활성화하는 것에 응답하여, 및/또는 발산 분기 조건에 응답하여 하나 이상의 스레드들을 비활성화하는 것에 응답하여 발생할 수도 있다.

[0173] 제어 흐름 모듈 (34) 은 비활성 스레드를 선택한다 (240). 제어 흐름 모듈 (34) 은 비활성 스레드에 대한 재개 카운터 값이 MINRC 값보다 더 작은지 여부를 결정한다 (242). 제어 흐름 모듈 (34) 이 비활성 스레드에 대한 재개 카운터 값이 MINRC 값보다 더 작은 것으로 결정하는 경우, 제어 흐름 모듈 (34) 은 비활성 스레드에 대한 재개 카운터 값이 현재 실행하는 프로그램 모듈에 대한 진입 포인트 이상인지 여부를 결정한다 (244). 현재 실행하는 프로그램 모듈에 대한 진입 포인트는 현재 실행되고 있는 프로그램 모듈의 시작 어드레스에 대응하는 프로그램 카운터 값을 지칭할 수도 있다. 제어 흐름 모듈 (34) 이 비활성 스레드에 대한 재개 카운터 값이 현재 실행하는 프로그램 모듈에 대한 진입 포인트 이상인 것으로 결정하는 경우, 제어 흐름 모듈 (34) 은 MINRC 를 비활성 스레드에 대한 재개 카운터 값과 동일하게 설정하고 (246), 판정 박스 (248) 로 진행한다.

[0174] 판정 박스 (242) 로 복귀하면, 제어 흐름 모듈 (34) 이 비활성 스레드에 대한 재개 카운터 값이 MINRC 값보다 더 작지 않은 것 (즉, 비활성 스레드에 대한 재개 카운터 값이 MINRC 값 이상인 것) 으로 결정하는 경우, 제어 흐름 모듈 (34) 은 MINRC 를 비활성 스레드에 대한 재개 카운터 값과 동일하게 설정하지 않고 판정 박스 (248) 로 진행할 수도 있다. 판정 박스 (244) 로 복귀하면, 제어 흐름 모듈 (34) 이 비활성 스레드에 대한 재개 카운터 값이 현재 실행하는 프로그램 모듈에 대한 진입 포인트 이상이 아닌 것 (즉, 재개 카운터가 진입 포인트보다 더 작은 것) 으로 결정하는 경우, 제어 흐름 모듈 (34) 은 MINRC 를 비활성 스레드에 대한 재개 카운터 값

과 동일하게 설정하지 않고 판정 박스 (248) 로 진행할 수도 있다.

[0175] 어느 하나의 경우에 있어서, 제어 흐름 모듈 (34) 은 프로세싱하기 위한 임의의 더 많은 비활성 스레드들이 있는지 여부를 결정한다 (248). 제어 흐름 모듈 (34) 이 프로세싱하기 위한 더 많은 비활성 스레드들이 있는 것으로 결정하는 경우, 제어 흐름 모듈 (34) 은 또 다른 비활성 스레드를 프로세싱하기 위하여 프로세스 박스 (240) 로 복귀한다. 이와 다르게, 제어 흐름 모듈 (34) 이 프로세싱하기 위한 비활성 스레드들이 더 이상 없는 것으로 결정하는 경우, 제어 흐름 모듈 (34) 은 MINRC 업데이트 프로세스를 종료한다 (250).

[0176] 도 17 에 도시된 흐름도는 재개 카운터 값들과 현재 프로세싱된 프로그래밍 모듈의 진입 포인트들과의 비교에 기초하여 MINRC 를 업데이트하기 위한 기법의 단지 하나의 예인 것에 주목해야 한다. 추가적인 예들에서는, 판정 박스들 (242 및 244) 의 순서가 전환될 수도 있거나, 판정 박스들 (242 및 244) 이 동시에 수행될 수도 있다.

[0177] 도 18 은 본 개시물에 따라 MINRC 를 업데이트하기 위한 또 다른 예의 기법을 예시하는 흐름도이다. 일부의 예들에서, 도 18 에서 예시된 기법은 도 11 에서 예시된 프로세스 박스 (166), 도 14 에서 예시된 프로세스 박스 (214), 도 15 에서 예시된 프로세스 박스 (226), 및/또는 도 16 에서 예시된 프로세스 박스 (238) 를 구현하기 위하여 이용될 수도 있다. 일반적으로, 도 18 에서 예시된 기법은 하나 이상의 재개 카운터들을 업데이트하는 것에 응답하여 수행될 수도 있고, 이것은 재개 검사 동작의 일부로서 하나 이상의 스레드들을 활성화하는 것에 응답하여, 및/또는 발산 분기 조건에 응답하여 하나 이상의 스레드들을 비활성화하는 것에 응답하여 발생할 수도 있다.

[0178] 제어 흐름 모듈 (34) 은 비활성 스레드를 선택한다 (252). 제어 흐름 모듈 (34) 은 비활성 스레드에 대한 재개 카운터 값이 MINRC 값보다 더 작은지 여부를 결정한다 (254). 제어 흐름 모듈 (34) 이 비활성 스레드에 대한 재개 카운터 값이 MINRC 값보다 더 작은 것으로 결정하는 경우, 제어 흐름 모듈 (34) 은 현재 실행하는 프로그램 모듈의 실행이 개시되었을 때에 비활성 스레드가 최초에는 활성이었는지 여부를 결정한다 (256). 예를 들어, 제어 흐름 모듈 (34) 은 플래그들의 세트를 유지할 수도 있고, 여기서, 각각의 플래그는 현재 실행하는 프로그램 모듈의 실행이 개시되었을 때에 각각의 플래그에 대응하는 스레드가 활성이었는지 여부를 표시하는 값을 저장한다. 이러한 예에서, 제어 흐름 모듈 (34) 은, 현재 프로세싱되고 있는 비활성 스레드에 대응하는 플래그 값이, 현재 실행하는 프로그램 모듈의 실행이 개시되었을 때에 그 스레드가 활성이었음을 표시하는지 여부를 결정할 수도 있다. 제어 흐름 모듈 (34) 이 현재 실행하는 프로그램 모듈의 실행이 개시되었을 때에 비활성 스레드가 최초에는 활성이었던 것으로 결정하는 경우, 제어 흐름 모듈 (34) 은 MINRC 를 비활성 스레드에 대한 재개 카운터 값과 동일하게 설정하고 (258), 판정 박스 (260) 로 진행한다.

[0179] 판정 박스 (254) 로 복귀하면, 제어 흐름 모듈 (34) 이 비활성 스레드에 대한 재개 카운터 값이 MINRC 값보다 더 작지 않은 것 (즉, 비활성 스레드에 대한 재개 카운터 값이 MINRC 값 이상인 것) 으로 결정하는 경우, 제어 흐름 모듈 (34) 은 MINRC 를 비활성 스레드에 대한 재개 카운터 값과 동일하게 설정하지 않고 판정 박스 (260) 로 진행할 수도 있다. 판정 박스 (256) 로 복귀하면, 제어 흐름 모듈 (34) 이 현재 실행하는 프로그램 모듈의 실행이 개시되었을 때에 비활성 스레드가 최초에는 활성이 아니었던 것으로 결정하는 경우, 제어 흐름 모듈 (34) 은 MINRC 를 비활성 스레드에 대한 재개 카운터 값과 동일하게 설정하지 않고 판정 박스 (260) 로 진행할 수도 있다.

[0180] 어느 하나의 경우에 있어서, 제어 흐름 모듈 (34) 은 프로세싱하기 위한 임의의 더 많은 비활성 스레드들이 있는지 여부를 결정한다 (260). 제어 흐름 모듈 (34) 이 프로세싱하기 위한 더 많은 비활성 스레드들이 있는 것으로 결정하는 경우, 제어 흐름 모듈 (34) 은 또 다른 비활성 스레드를 프로세싱하기 위하여 프로세스 박스 (252) 로 복귀한다. 이와 다르게, 제어 흐름 모듈 (34) 이 프로세싱하기 위한 비활성 스레드들이 더 이상 없는 것으로 결정하는 경우, 제어 흐름 모듈 (34) 은 MINRC 업데이트 프로세스를 종료한다 (262).

[0181] 도 18 에 도시된 흐름도는 재개 카운터 값들과 현재 프로세싱된 프로그래밍 모듈의 진입 포인트들과의 비교에 기초하여 MINRC 를 업데이트하기 위한 기법의 단지 하나의 예인 것에 주목해야 한다. 추가적인 예들에서는, 판정 박스들 (254 및 256) 의 순서가 전환될 수도 있거나, 판정 박스들 (254 및 256) 이 동시에 수행될 수도 있다.

[0182] 도 19 는 본 개시물의 서브루틴 실행 기법들을 구현하기 위하여 이용될 수도 있는 일 예의 제어 흐름 모듈 (34) 을 예시하는 블록도이다. 일부의 예들에서, 도 19 에서 예시된 일 예의 제어 흐름 모듈 (34) 은 도 7 내지 도 18 에 대하여 위에서 설명된 기법들을 구현하기 위하여 이용될 수도 있다. 제어 흐름 모듈 (34) 은 스레

드 레지스터들 (302), 활성 플래그들 (304A 내지 304D; 집합적으로 "활성 플래그들 (304)"), 재개 카운터들 (306A 내지 306D; 집합적으로 "재개 카운터들 (306)"), 최소 재개 카운터 (MINRC; 308), 재개 검사 모듈 (310), 분기 조건 평가기 (312), 이벤트 정보 생성기 (314), 프로그램 상태 레지스터 (316), 상태 천이 블록 (318), 스레드 비활성화기 (320), 서브루틴 핸들러 (322), 호출 스택 (324), MINRC 스택 (326), 및 다음 명령 블록 (328) 을 포함한다.

[0183] 일부의 예들에서, 도 19 에서 예시된 일 예의 제어 흐름 모듈 (34) 은 도 3 에 도시된 일 예의 제어 흐름 모듈 (34) 을 구현하기 위하여 이용될 수도 있다. 이러한 예들에서, 도 19 에 도시된 재개 카운터들 (306) 은 도 3 에 도시된 재개 카운터들 (46) 에 대응할 수도 있고, 도 19 에 도시된 MINRC (308) 는 도 3 에 도시된 MINRC 레지스터 (42) 에 대응할 수도 있고, 도 19 에 도시된 분기 조건 평가기 (312) 는 도 3 에 도시된 분기 조건 평가기 (40) 에 대응할 수도 있고, 도 19 에 도시된 서브루틴 핸들러 (322) 는 도 3 에 도시된 서브루틴 핸들러 (50) 에 대응할 수도 있다. 따라서, 간결함을 위하여 그리고 중복성을 회피하기 위하여, 이 공유된 컴포넌트들의 구성 및 동작은 더욱 상세하게 설명되지 않는다. 또한, 이벤트 정보 생성기 (314), 프로그램 상태 레지스터 (316), 상태 천이 블록 (318), 및 다음 명령 블록 (328) 은 도 3 에 도시된 다음 명령 생성기 (44) 의 기능성을 구현하도록 구성될 수도 있다. 유사하게, 재개 검사 모듈 (310), 이벤트 정보 생성기 (314), 프로그램 상태 레지스터 (316), 상태 천이 블록 (318), 및 스레드 비활성화기 (320) 는 도 3 에 도시된 스레드 상태 관리기 (48) 의 기능성을 구현하도록 구성될 수도 있다. 추가적으로, 호출 스택 (324) 및 MINRC 스택 (326) 은 도 3 에 도시된 스택 저장 구조 (52) 에 대응할 수도 있다.

[0184] 스레드 레지스터들 (302) 은 프로세싱 시스템 (10) 에서 실행되는 스레드들의 각각에 대한 스레드 상태를 저장하도록 구성된다. 도 19 에 도시된 바와 같이, 스레드 레지스터들 (302) 은 활성 플래그들 (304) 및 재개 카운터들 (306) 을 포함한다. 활성 플래그들 (304) 의 각각은 각각의 활성 플래그 (304A 내지 304D) 에 대응하는 스레드의 상태가 활성인지 여부를 표시하는 활성 플래그를 저장한다. 재개 카운터들 (306) 의 각각은 각각의 스레드에 대한 재개 카운터 값을 저장한다. 일부의 예들에서, 각각의 스레드는 프로세싱 엘리먼트들 (14) 중의 각각의 하나에 배정될 수도 있다. 이러한 예들에서, 활성 플래그들 (304) 및 재개 카운터들 (306) 의 각각은 프로세싱 엘리먼트들 (14) 의 각각의 하나에 대응할 수도 있다. 예를 들어, 활성 플래그 (304A) 및 재개 카운터 (306A) 는 도 1 에서 예시된 프로세싱 엘리먼트 (14A) 에 각각 대응할 수도 있고, 활성 플래그 (304B) 및 재개 카운터 (306B) 는 도 1 에서 예시된 프로세싱 엘리먼트 (14B) 에 각각 대응할 수도 있다. 도 19 에서 예시된 일 예의 제어 흐름 모듈 (34) 이 4 개의 활성 플래그들 (304) 및 4 개의 재개 카운터들 (306) 을 갖는 시스템을 예시하지만, 다른 예들에서는, 제어 흐름 모듈 (34) 이 동일하거나 상이한 수들의 활성 플래그들 (304) 및 재개 카운터들 (306) 을 가질 수도 있다.

[0185] 스레드 레지스터들 (302) 은 또한 MINRC (308) 를 포함한다. MINRC (308) 는 현재 실행하는 프로그램 모듈의 실행이 개시되었을 때에 활성이었던 재개 카운터들 (306) 의 세트로부터의 가장 작은 값을 표시하는 값을 저장하도록 구성된다.

[0186] 재개 검사 모듈 (310) 은 프로그램 카운터 (28) 가 새로운 프로그램 카운터 값으로 로딩되는 것에 응답하여 재개 검사 동작을 수행하도록 구성된다. 일부의 예들에서, 재개 검사 모듈 (310) 은 도 16 에서 예시된 재개 검사 기법들에 따라 재개 검사 동작을 수행할 수도 있다. 재개 검사 모듈 (310) 은 재개 검사 동작을 수행하기 위하여, 프로그램 카운터 (28) 로부터 현재의 프로그램 카운터 값, 그리고 스레드 레지스터들 (302) 로부터 현재의 활성 플래그들 (304) 및 재개 카운터 값들 (306) 을 수신할 수도 있다. 재개 검사 모듈 (310) 은 또한, 재개 검사 동작의 일부로서, 활성 플래그들 (304), 재개 카운터들 (306) 및 MINRC (308) 를 수정할 수도 있다. 추가적으로, 재개 검사 모듈 (310) 은 재개 검사 동작의 결과에 기초하여 프로그램 상태 레지스터 (316) 를 업데이트할 수도 있다.

[0187] 일부의 예들에서, 재개 검사 동작을 완료한 후, 재개 검사 모듈 (310) 은 재개 검사 동작이 완료하였음을 표시하는 신호를, 페치 모듈 (30) 및 디코드 모듈 (32) 중의 하나 또는 양자에 전송할 수도 있다. 페치 모듈 (30) 이 재개 검사 동작이 완료하였다는 신호를 수신할 때, 페치 모듈 (30) 은 추가의 프로세싱을 위하여 페치된 명령을 디코드 모듈 (32) 로 포워딩할 수도 있다. 명령을 수신하는 것에 응답하여, 디코드 모듈 (32) 은 활성 플래그들 (304) 을 검사할 수도 있고, 재개 검사 동작에 의해 수정되었을 수도 있는 활성 플래그들 (304) 의 현재 성태에 기초하여 프로세싱 엘리먼트들 (14) 의 활성 및 비활성 상태를 업데이트할 수도 있다. 명령이 프로세싱 엘리먼트들 (14) 에 발행가능한 타입인 경우, 디코드 모듈 (32) 은 프로세싱 엘리먼트들 (14) 의 활성 및 비활성 상태를 업데이트하는 것과 함께, 또는 상기 업데이트 후에 프로세싱 엘리먼트들 (14) 에 명령을 발행할 수도 있다. 일 예의 제어 흐름 모듈 (34) 은 재개 검사 동작의 완료 시에 페치 모듈 (30) 에 시그널

링하는 것으로 재개 검사 모듈 (310) 을 예시하지만, 다른 예들에서는, 재개 검사 모듈 (158) 이 재개 검사가 완료하였음을 표시하는 신호를 디코드 모듈 (32) 에 전송할 수도 있다. 이러한 예들에서는, 디코드 모듈 (32) 이 신호를 수신할 때, 디코드 모듈 (32) 은 활성 플래그들 (304) 을 검사할 수도 있고, 활성 플래그들 (304) 의 현재의 상태에 기초하여 프로세싱 엘리먼트들 (14) 의 활성 및 비활성 상태를 업데이트할 수도 있다.

[0188] 디코드 모듈 (32) 이 명령을 디코딩할 때, 디코드 모듈 (32) 이 명령이 분기 명령인 것 (즉, 조건부 분기 명령) 으로 결정하는 경우, 디코드 모듈 (32) 은 현재의 명령이 조건부 분기 명령임을 표시하는 신호를 분기 조건 평가기 (312) 로 전송할 수도 있고, 추가의 프로세싱을 위하여 분기 조건을 표시하는 정보를 분기 조건 평가기 (312) 에 제공할 수도 있다. 일부의 예들에서, 디코드 모듈 (32) 이 명령이 분기 명령이 아닌 것 (예를 들어, 점프 명령, 서브루틴 명령, 또는 순차적인 명령) 으로 결정하는 경우, 디코드 모듈 (32) 은 현재의 명령이 조건부 분기 명령이 아님을 표시하는 신호를 분기 조건 평가기 (160) 로 전송할 수도 있다.

[0189] 디코드 모듈 (32) 은 추가의 프로세싱을 위하여 제어 정보를 이벤트 정보 생성기 (162) 에 제공한다. 일부의 예들에서, 제어 정보는 명령 자체일 수도 있다. 추가의 예들에서, 제어 정보는 예를 들어, 명령이 제어 흐름 명령 또는 순차적인 명령인지 여부를 표시하는 정보; 명령이 제어 흐름 명령인 경우, 명령이 분기 명령, 점프 명령, 호출 명령, 또는 복귀 명령인지 여부를 표시하는 정보; 및 명령이 분기 또는 점프 명령인 경우, 분기 또는 점프 명령이 순방향 또는 역방향 분기 또는 점프 명령인지 여부를 표시하는 정보; 및 명령이 분기 명령인 경우, 분기 조건을 특정하는 정보와 같은 정보를 포함할 수도 있다.

[0190] 현재 프로세싱된 명령이 조건부 분기 명령인 경우, 분기 조건 평가기 (312) 는 각각의 활성 스레드에 대해 분기 조건을 평가할 수도 있다. 일부의 예들에서, 분기 조건 평가기 (312) 는 통신 경로 (22) 를 통해 프로세싱 엘리먼트들 (14) 로부터 비교 동작 또는 제로 검사 동작의 결과를 수신할 수도 있다. 추가의 예들에서, 분기 조건 평가기 (312) 는 통신 경로 (24) 를 통해 데이터 저장소 (18) 에서의 하나 이상의 레지스터들에 액세스할 수도 있고, 비교 동작 또는 제로 검사 동작을 수행할 수도 있다. 임의의 경우에 있어서, 분기 조건 평가기 (312) 는 분기 조건이 각각의 활성 스레드에 대해 충족되는지 또는 충족되지 않는지 여부를 결정할 수도 있다. 일부의 예들에서, 분기 조건 평가기 (312) 는 분기 조건이 각각의 활성 스레드에 대해 충족되는지 또는 충족되지 않는지 여부를 표시하는 정보를 이벤트 정보 생성기 (314) 로 포워딩할 수도 있다. 추가적인 예들에서, 분기 조건 평가기 (312) 는 현재의 명령에 대한 분기 발산이 균일한지 또는 발산하는지 여부를 결정할 수도 있고, 분기 발산이 균일한지 또는 발산하는지 여부를 표시하는 정보를 이벤트 정보 생성기 (314) 로 포워딩할 수도 있다. 추가의 예들에서, 분기 발산이 분기 명령에 대해 균일한 경우, 분기 조건 평가기 (312) 는 분기 조건이 균일하게 충족되는지 또는 균일하게 충족되지 않는지 여부를 결정할 수도 있고, 분기 조건이 균일하게 충족되는지 또는 균일하게 충족되지 않는지 여부를 표시하는 정보를 이벤트 정보 생성기 (314) 로 포워딩할 수도 있다.

[0191] 이벤트 정보 생성기 (314) 는 디코드 모듈 (32) 로부터 제어 정보를 수신하고, 현재 프로세싱된 명령이 분기 명령인 경우에는, 분기 조건 평가기 (312) 로부터 분기 조건 정보를 수신한다. 일부의 예들에서, 현재 프로세싱된 명령이 분기 명령인 경우, 이벤트 정보 생성기 (314) 는 또한 분기 조건 평가기 (312) 로부터 분기 발산 정보를 수신할 수도 있다. 이벤트 정보 생성기 (314) 가 분기 조건 평가기 (312) 로부터 분기 발산 정보를 수신하지 않는 경우에는, 이벤트 정보 생성기 (314) 는 현재의 명령에 대한 분기 발산이 균일한지 또는 발산하는지 여부를 결정할 수도 있다. 이벤트 정보 생성기 (314) 는 또한, 현재 프로세싱된 명령에 대한 타겟 프로그램 카운터 값이 MINRC (308) 이하인지 여부를 결정할 수도 있다. 이벤트 정보 생성기 (314) 는 수신된 정보에 기초하여 이벤트들을 생성하고, 이벤트들을 상태 천이 블록 (318), 스레드 비활성화기 (320), 서브루틴 핸들러 (322) 및 다음 명령 블록 (328) 에 제공한다.

[0192] 일부의 예들에서, 이벤트 정보 생성기 (314) 는 다음의 이벤트들을 생성할 수도 있다:

[0193] *Jb*: 역방향 점프 명령

[0194] *JfL*: 순방향 점프 명령, 타겟은 MINRC 이하이다

[0195] *JfG*: 순방향 점프 명령, 타겟은 MINRC 보다 더 크다

[0196] *BbuT*: 역방향 분기 명령, 모든 스레드들은 균일하고, 조건은 참이다

[0197] *BbuF*: 역방향 분기 명령, 모든 스레드들은 균일하고, 조건은 거짓이다

[0198] *BfuTL*: 순방향 분기 명령, 모든 스레드들은 균일하고, 조건은 참이고,

[0199] 타겟은 MINRC 이하이다

[0200] *BfuTG*: 순방향 분기 명령, 모든 스레드들은 균일하고, 조건은 참이고,

[0201] 타겟은 MINRC 보다 더 크다

[0202] *BfuF*: 순방향 분기 명령, 모든 스레드들은 균일하고, 조건은 거짓이다

[0203] *Bbd*: 역방향 분기 명령, 스레드들은 발산한다

[0204] *Bfd*: 순방향 분기 명령, 스레드들은 발산한다

[0205] *S*: 순차적인 명령

[0206] *Call*: 서브루틴 진입 로케이션으로 점프

[0207] *Ret*: 호출자에서의 호출 직후에 다음 명령으로 점프

[0208] 상기 식별된 이벤트들에 따르면, 명령은 순차적인 명령 (S), 점프 명령 (J), 분기 명령 (B), 호출 명령 (Call), 또는 복귀 명령 (Ret) 일 수도 있다. 점프 또는 분기 명령들에 대하여, 점프 또는 분기 방향은 역방향 (b) 또는 순방향 (f) 의 어느 하나일 수도 있다. 분기 명령들에 대하여, 분기 발산은 균일 (u) 또는 발산 (d) 의 어느 하나일 수도 있다. 분기 명령들에 대하여, 분기 조건은 참 (T) 또는 거짓 (F) 의 어느 하나일 수도 있다. 참 분기 조건은 충족된 분기 조건에 대응할 수도 있고, 거짓 분기 조건은 충족되지 않은 분기 조건에 대응할 수도 있다. 순방향 점프 범위는 타겟이 MINRC 보다 더 큰지 여부에 의존할 수도 있다. 따라서, 비교 결과는 타겟이 MINRC 이하임 (L), 또는 타겟이 MINRC 보다 더 큼 (G) 을 표시하도록 정의될 수도 있다.

[0209] 프로그램 상태 레지스터 (316) 는 프로세싱 시스템 (10) 에서 실행되는 프로그램에 대한 프로그램 상태를 저장할 수도 있다. 일부의 예들에서, 프로그램 상태 레지스터 (316) 는 다음의 3 개의 상태들을 저장할 수도 있다:

[0210] 상태 0: 모든 스레드들이 활성이다.

[0211] 상태 1: 적어도 하나의 스레드가 활성이고 적어도 하나의 스레드가 비활성이다.

[0212] 상태 2: 모든 스레드들이 비활성이다.

[0213] 일부의 예들에서, 프로세싱 시스템 (10) 은 프로그램의 초기 상태 및 최종 상태가 각각 상태 0 이도록 구성될 수도 있다.

[0214] 상태 천이 블록 (318) 은 이벤트 정보 생성기 (314) 로부터 이벤트를, 그리고 프로그램 상태 레지스터 (316) 로부터 현재의 프로그램 상태를 수신할 수도 있고, 수신된 이벤트들 및 현재의 프로그램 상태에 기초하여 새로운 프로그램 상태를 생성할 수도 있고, 새로운 프로그램 상태를 프로그램 상태 레지스터 (316) 에 저장할 수도 있다. 상태 천이 블록 (318) 은 도 20 에 대하여 더욱 상세하게 설명된 상태 천이도에 따라, 및/또는 도 21 에 대하여 더욱 상세하게 설명된 상태 천이표에 따라 새로운 프로그램 상태를 생성할 수도 있다.

[0215] 스레드 비활성화기 (320) 는 이벤트 정보 생성기 (314) 로부터 이벤트를, 그리고 프로그램 상태 레지스터 (316) 로부터 현재의 프로그램 상태를 수신할 수도 있고, 이벤트 및 현재의 프로그램 상태에 기초하여 하나 이상의 스레드들을 비활성화할 것인지 여부를 결정할 수도 있고, 이벤트들 및 현재의 프로그램 상태들의 어떤 조합들에 응답하여 하나 이상의 스레드들을 비활성화할 수도 있다. 스레드들을 비활성화할 때, 스레드 비활성화기 (320) 는 비활성되고 있는 스레드들에 대한 활성 플래그들 (304) 및 재개 카운터들 (306) 을 업데이트할 수도 있다. 스레드 비활성화기 (320) 는 도 21 에 대하여 더욱 상세하게 설명된 상태 천이표에 따라 스레드들을 비활성화할 수도 있다.

[0216] 서브루틴 핸들러 (322) 는 이벤트 정보 생성기 (314) 로부터 이벤트를 수신할 수도 있고, 수신된 이벤트에 기초하여 호출 스택 (324), MINRC 스택 (326) 및 MINRC (308) 를 관리할 수도 있다. 예를 들어, *Call* 이벤트를 수신하는 것에 응답하여, 서브루틴 핸들러 (322) 는 MINRC (308) 에 현재 저장된 MINRC 값을 MINRC 스택 (326) 상으로 푸시할 수도 있고, MINRC (308) 를 디폴트 값으로 초기화할 수도 있다. MINRC (308) 를 디폴트 값으로 초기화하는 것은 MINRC (308) 에 이전에 저장되어 있었던, 호출자 프로그램에 대응하는 MINRC 값을 겹쳐쓰기하는 것을 포함할 수도 있다. 또한, *Call* 이벤트를 수신하는 것에 응답하여, 서브루틴 핸들러 (322) 는 복귀 어드레스를 호출 스택 (324) 상으로 푸시할 수도 있다. 복귀 어드레스는 현재 프로세싱되고 있는 호출

명령 직후에 발생하는 호출자 프로그램에서의 다음의 순차적인 명령에 대응할 수도 있다.

[0217] *Ret* 이벤트를 수신하는 것에 응답하여, 서브루틴 핸들러 (322) 는 MINRC 스택 (326) 에서 가장 최근에 저장된 MINRC 값을 팝핑할 수도 있고, 팝핑된 MINRC 값을 MINRC (308) 에 저장할 수도 있다. 팝핑된 MINRC 값을 MINRC (308) 에 저장하는 것은 MINRC (308) 에 이전에 저장되어 있었던, 피호출자 서브루틴 프로그램에 대응하는 MINRC 값을 겹쳐쓰기하는 것을 포함할 수도 있다. 또한, *Ret* 이벤트를 수신하는 것에 응답하여, 서브루틴 핸들러 (322) 는 호출 스택 (324) 에서 가장 최근에 저장된 복귀 어드레스를 팝핑할 수도 있고, 프로그램 카운터 (28) 로 하여금, 팝핑된 복귀 어드레스에 대응하는 값으로 로딩되게 할 수도 있다. 예를 들어, 서브루틴 핸들러 (322) 는 팝핑된 복귀 어드레스를 다음 명령 블록 (328) 으로 전달할 수도 있고, 이 다음 명령 블록 (328) 은 다음 명령 사이클 상에서 프로그램 카운터 (28) 로 로딩하기 위하여 팝핑된 복귀 어드레스에 대응하는 프로그램 카운터 값을 선택할 수도 있다.

[0218] 다음 명령 블록 (328) 은 이벤트 정보 생성기 (314) 로부터 이벤트를, 그리고 프로그램 상태 레지스터 (316) 로부터 현재의 프로그램 상태를 수신할 수도 있고, 프로그램 카운터 (28) 로 로딩하기 위한 새로운 프로그램 카운터 값을 결정할 수도 있고, 새로운 프로그램 카운터 값을 프로그램 카운터 (28) 로 로딩할 수도 있다. 새로운 프로그램 카운터 값은 제어 유닛 (12) 에 의해 프로세싱되어야 할 다음 명령을 표시할 수도 있다. 다음 명령 블록 (328) 은 도 21 에 대하여 더욱 상세하게 설명된 상태 천이표에 따라 새로운 프로그램 카운터 값을 결정할 수도 있다.

[0219] 위에서 논의된 바와 같이, 재개 검사 모듈 (310) 은 재개 검사 동작의 결과에 기초하여 프로그램 상태 레지스터 (316) 를 업데이트할 수도 있다. 이 업데이트는 재개 검사 모듈 (310) 에 의해 비동기 방식으로 수행될 수도 있다. 예를 들어, 프로그램 상태가 재개 검사 동작을 수행하기 이전에 상태 1 이었고, 모든 비활성 스레드들이 재활성화되는 경우, 프로그램 상태 레지스터 (316) 는 모든 스레드들이 활성화되어 있음을 반영하기 위하여 프로그램 상태 레지스터 (316) 를 상태 0 으로 비동기 방식으로 변경시킬 수도 있다. 상태 천이 블록 (318) 은 재개 검사 모듈 (310) 에 의한 임의의 업데이트 후에 이용가능한 현재의 프로그램 상태에 기초하여 새로운 프로그램 상태를 생성한다는 것에 주목해야 한다. 유사하게, 스레드 비활성화기 (320) 는 재개 검사 모듈 (310) 에 의한 임의의 업데이트 후에 이용가능한 현재의 프로그램 상태에 기초하여 하나 이상의 스레드들을 비활성화할 것인지 여부를 결정하고, 다음 명령 블록 (328) 은 재개 검사 모듈 (310) 에 의한 임의의 업데이트 후에 이용가능한 현재의 프로그램 상태에 기초하여 새로운 프로그램 카운터 값을 결정한다. 이와 같이, 프로그램 상태는 재개 검사로 인해 단일 프로세싱 사이클 동안에 2 개의 상이한 상태들 사이에서 변경될 수도 있지만, 프로세싱 사이클에 대한 최종 상태, 즉, 재개 검사가 완료한 후에 발생하는 상태는 상태 천이 블록 (318), 스레드 비활성화기 (320) 및 다음 명령 블록 (328) 의 각각에 의한 프로세싱을 위해 현재의 프로그램 상태로서 이용된다.

[0220] 일부의 예들에서, 제어 흐름 모듈 (34) 에서의 컴포넌트들의 각각은 프로세서 내의 하나 이상의 하드웨어 컴포넌트들로서 구현될 수도 있다. 예를 들어, 스레드 레지스터들 (302), 활성 플래그들 (304), 재개 카운터들 (306), MINRC (308), 프로그램 상태 레지스터 (316), 호출 스택 (324), 및/또는 MINRC 스택 (326) 은 하나 이상의 하드웨어 레지스터들로서 각각 구현될 수도 있다. 또 다른 예로서, 재개 검사 모듈 (310), 분기 조건 평가기 (312), 이벤트 정보 생성기 (314), 상태 천이 블록 (318), 스레드 비활성화기 (320), 서브루틴 핸들러 (322) 및/또는 다음 명령 블록 (328) 은 조합 논리 하드웨어로서, 및/또는 조합 논리 하드웨어 및 하드웨어 레지스터들의 조합으로서 구현될 수도 있다.

[0221] 도 20 은 본 개시물에 따라 도 19 에서 예시된 제어 흐름 모듈 (34) 의 예시적인 동작을 특징화하는 상태 천이도이다. 도 20 에서 도시된 화살표들은 원들에 의해 식별된 상이한 상태들 사이의 천이들을 나타낸다. 화살표들은 이벤트 정보 생성기 (314) 에 의해 생성된 이벤트들 및 재개 이벤트 중의 하나 또는 양자와 연관된다. 재개 이벤트는 하나 이상의 스레드들이 재활성화되는 재개 검사 동작의 결과로서 발생하는 비동기 상태 천이일 수도 있다. 이벤트 정보 생성기 (314) 에 의해 생성된 나머지 이벤트들과 연관된 상태 천이들은 동기 상태 천이들일 수도 있다. 동기 상태 천이는 프로세싱 사이클들 사이에서 발생할 수도 있고, 비동기 상태 천이는 프로세싱 사이클 동안에 발생할 수도 있다. 비동기 상태 천이가 재개 검사로 인해 프로세싱 사이클 동안에 발생하는 경우, 비동기 천이 후에 발생하는 상태는 다음 프로세싱 사이클에 대한 다음 상태를 결정하기 위하여 이용된다.

[0222] 도 21 은 본 개시물에 따라 도 19 에서 예시된 제어 흐름 모듈 (34) 의 예시적인 동작을 특징화하는 상태 천이표이다. 도 21 의 상태 천이표는 현재의 프로그램 상태를 나타내는 "과거 상태" 열 (column) 과, 다음 프로

세싱 사이클에 대한 새로운 프로그램 상태 또는 재개 검사 동작으로 인해 비동기 천이 후에 발생하는 프로그램 상태 중의 어느 하나를 나타내는 "새로운 상태" 열을 포함한다. 상태 천이표는 또한, 이벤트 정보 생성기 (314) 에 의해 생성된 이벤트들을 포함하는 "이벤트" 열을 포함한다. "이벤트" 열에서의 표시자 "n/a" 는 상태 천이 및 액션 (action) 이 재개 검사 동작으로 인해 발생한다는 것과, 이벤트가 이러한 천이에 대해 무관하다는 것을 의미한다. 상태 천이표는 또한, 현재의 프로그램 상태 및 이벤트의 특별한 조합에 응답하여 어떤 액션이 발생하는지를 표시하는 "액션" 열을 포함한다. "액션" 열에서 "재개" 라고 표기된 액션은 재개 검사 동작으로 인해 비동기 상태 천이가 발생하는 것을 의미한다.

[0223] 도 20 및 도 21 에 도시된 바와 같이, 상태 천이 블록 (318) 은 현재의 상태가 상태 0 인 것과, *S* 이벤트, *Jb* 이벤트, *BfuF* 이벤트, *BbuT* 이벤트, *BbuF* 이벤트, *JfL* 이벤트, *BfuTL* 이벤트, *Call* 이벤트, 또는 *Ret* 이벤트를 수신하는 것에 응답하여, 프로그램 상태 레지스터 (316) 로 로딩하기 위한 새로운 상태로서 상태 0 을 선택한다. 상태 천이 블록 (318) 은 현재의 상태가 상태 0 인 것과, *Bbd* 이벤트 또는 *Bfd* 이벤트를 수신하는 것에 응답하여, 프로그램 상태 레지스터 (316) 로 로딩하기 위한 새로운 상태로서 상태 1 을 선택한다. 상태 천이 블록 (318) 은 또한, 현재의 상태가 상태 1 인 것과, *S* 이벤트, *Jb* 이벤트, *BbuF* 이벤트, *BbuT* 이벤트, *BfuF* 이벤트, *Bbd* 이벤트, *Bfd* 이벤트, *JfL* 이벤트, *BfuTL* 이벤트, *Call* 이벤트, 또는 *Ret* 이벤트를 수신하는 것에 응답하여, 프로그램 상태 레지스터 (316) 로 로딩하기 위한 새로운 상태로서 상태 1 을 선택한다. 상태 천이 블록 (318) 은 현재의 상태가 상태 1 인 것과, *JfG* 이벤트 또는 *BfuTG* 이벤트를 수신하는 것에 응답하여, 프로그램 상태 레지스터 (316) 로 로딩하기 위한 새로운 상태로서 상태 2 를 선택한다. 상태 천이 블록 (318) 은 또한, 현재의 상태가 상태 2 인 것과, 임의의 이벤트를 수신하는 것에 응답하여, 프로그램 상태 레지스터 (316) 로 로딩하기 위한 새로운 상태로서 상태 2 를 선택한다. 재개 검사 동작의 일부로서 하나 이상의 스레드들을 재활성화하는 것에 응답하여, 상태 천이 블록 (318) 은 비동기 방식으로 상태 0 또는 상태 1 로 천이할 수도 있다.

[0224] 도 24 에 도시된 바와 같이, 스레드 비활성화기 (320) 는 현재의 상태가 상태 0 또는 상태 1 의 어느 하나인 것과, *Bbd* 이벤트 또는 *Bfd* 이벤트를 수신하는 것에 응답하여, 하나 이상의 스레드들을 비활성화하는 것으로 결정할 수도 있다. *Bbd* 이벤트 및 *Bfd* 이벤트들은 분기 명령에 대한 분기 조건의 평가로부터 기인하는 발산 이벤트들로서 지칭될 수도 있다. 스레드 비활성화기 (320) 는, *Bbd* 이벤트를 수신하는 것에 응답하여 분기 조건 (즉, 거짓 조건) 을 충족하지 않는 모든 활성 스레드들을 비활성화하는 것과, *Bfd* 이벤트를 수신하는 것에 응답하여 분기 조건 (즉, 참인 조건) 을 충족하는 모든 활성 스레드들을 비활성화는 것으로 결정될 수도 있다. 스레드 비활성화기 (320) 는 분기 조건을 충족하지 않는 모든 활성 스레드들을 비활성화하기 위하여 도 14 에서 예시된 기법과, 분기 조건을 충족하는 모든 활성 스레드들을 비활성화하기 위하여 도 15 에서 예시된 기법을 활용할 수도 있다. 스레드 비활성화기 (320) 는 현재의 상태가 상태 1 인 것과, *JfG* 이벤트 또는 *BfuTG* 이벤트를 수신하는 것에 응답하여, 모든 활성 스레드들을 비활성화하는 것으로 결정할 수도 있다. 스레드 비활성화기 (320) 는 모든 활성 스레드들을 비활성화하기 위하여 도 11 에서 예시된 기법을 활용할 수도 있다.

[0225] 도 24 에 도시된 바와 같이, 다음 명령 블록 (328) 은 현재의 프로그램 상태 및 이벤트들의 다양한 조합들에 응답하여 프로그램 카운터 (28) 로 로딩하기 위한 다음의 프로그램 카운터 값들 중의 하나를 선택할 수도 있다: (1) 다음의 순차적인 명령을 표시하는 프로그램 카운터 값 (즉, "PC + 1"); (2) 타겟 명령을 표시하는 프로그램 카운터 값 (즉, 타겟 프로그램 카운터 값); (3) 최소 재개 카운터 값 (MINRC), 또는 복귀 어드레스를 표시하는 프로그램 카운터 값. 예를 들어, 다음 명령 블록 (328) 은 현재의 상태가 상태 0 인 것과, *S* 이벤트, *BfuF* 이벤트, *BbuF* 이벤트, 또는 *Bfd* 이벤트를 수신하는 것에 응답하여, 프로그램 카운터 (28) 로 로딩하기 위하여 다음의 순차적인 명령을 표시하는 프로그램 카운터 값 (즉, "PC +1") 을 선택할 수도 있다. 다음 명령 블록 (328) 은 또한, 현재의 상태가 상태 1 인 것과, *S* 이벤트, *BbuF* 이벤트, *BfuF* 이벤트, 또는 *Bfd* 이벤트를 수신하는 것에 응답하여, 프로그램 카운터 (28) 로 로딩하기 위하여 다음의 순차적인 명령을 표시하는 프로그램 카운터 값을 선택할 수도 있다. 다음 명령 블록 (328) 은 또한, 현재의 상태가 상태 2 인 것과, *JfL* 이벤트 또는 *JfG* 이벤트 이외의 임의의 이벤트를 수신하는 것에 응답하여, 프로그램 카운터 (28) 로 로딩하기 위하여 다음의 순차적인 명령을 표시하는 프로그램 카운터 값을 선택할 수도 있다.

[0226] 다음 명령 블록 (328) 은 현재의 상태가 상태 0 인 것과, *Jb* 이벤트, *BbuT* 이벤트, *JfL* 이벤트, *BfuTL* 이벤트, *Bbd* 이벤트, 또는 *Call* 이벤트를 수신하는 것에 응답하여, 프로그램 카운터 (28) 로 로딩하기 위하여 타겟 명령을 표시하는 프로그램 카운터 값 (즉, 타겟 프로그램 카운터 값) 을 선택할 수도 있다. 이벤트가 *Call* 이벤트인 경우에 있어서, 타겟 명령은 서브루틴 진입 포인트에 대응할 수도 있다. 다음 명령 블록 (328) 은 또한, 현재의 상태가 상태 1 인 것과, *Jb* 이벤트, *BbuT* 이벤트, *Bbd* 이벤트, *JfL* 이벤트, *BfuTL* 이벤트, 또는

Call 이벤트를 수신하는 것에 응답하여, 프로그램 카운터 (28) 로 로딩하기 위하여 타겟 명령을 표시하는 프로그램 카운터 값을 선택할 수도 있다. 다음 명령 블록 (328) 은 또한, 현재의 상태가 상태 2 인 것과, *JfL* 이벤트를 수신하는 것에 응답하여, 프로그램 카운터 (28) 로 로딩하기 위하여 타겟 명령을 표시하는 프로그램 카운터 값을 선택할 수도 있다.

[0227] 다음 명령 블록 (328) 은 현재의 상태가 상태 1 인 것과, *JfG* 이벤트 또는 *BfuTG* 이벤트를 수신하는 것에 응답하여, 프로그램 카운터 (28) 로 로딩하기 위한 MINRC 값을 선택할 수도 있다. 다음 명령 블록 (328) 은 현재의 상태가 상태 2 인 것과, *JfG* 이벤트를 수신하는 것에 응답하여, 프로그램 카운터 (28) 로 로딩하기 위한 MINRC 값을 선택할 수도 있다.

[0228] 다음 명령 블록 (328) 은 현재의 상태가 상태 0 인 것과, *Ret* 이벤트를 수신하는 것에 응답하여, 프로그램 카운터 (28) 로 로딩하기 위하여 복귀 어드레스를 표시하는 프로그램 카운터 값을 선택할 수도 있다. 다음 명령 블록 (328) 은 또한, 현재의 상태가 상태 1 인 것과, *Ret* 이벤트를 수신하는 것에 응답하여, 프로그램 카운터 (28) 로 로딩하기 위하여 복귀 어드레스를 표시하는 프로그램 카운터 값을 선택할 수도 있다.

[0229] 도 22 내지 도 28 은 본 개시물의 서브루틴 실행 기법들을 구현하기 위한 일 예의 의사-코드 (pseudo-code) 를 예시한다. 특히, 도 22 는 본 개시물에 따라 재개 검사 동작을 구현하기 위한 일 예의 의사-코드를 예시한다. 일부의 예들에서, 도 22 에서 예시된 의사-코드는 도 16 에서 예시된 흐름도에 대응할 수도 있다. 도 23 은 본 개시물에 따라 점프 명령 프로세싱을 구현하기 위한 일 예의 의사-코드를 예시한다. 일부의 예들에서, 도 23 에서 예시된 의사-코드는 도 10 및 도 11 에서 예시된 흐름도들에 대응할 수도 있다. 도 24 는 본 개시물에 따라 분기 명령 프로세싱을 구현하기 위한 일 예의 의사-코드를 예시한다. 일부의 예들에서, 도 24 에서 예시된 의사-코드는 도 11 내지 도 15 에서 예시된 흐름도들에 대응할 수도 있다.

[0230] 도 25 는 본 개시물에 따라 호출 명령 프로세싱을 구현하기 위한 일 예의 의사-코드를 예시한다. 일부의 예들에서, 도 25 에서 예시된 의사-코드는 도 8 에서 예시된 흐름도에 대응할 수도 있다. 도 25 에서 도시된 바와 같이, 제어 흐름 유닛 (34) 은 호출자 프로그램에 대응하는 MINRC 의 상태를 저장하기 위하여, 호출자 프로그램에 대한 MINRC 레지스터 (42) 에 저장된 값을 스택 저장 구조 (52) 에서의 MINRC 스택 상으로 푸시할 수도 있다. 추가적으로, 호출 흐름 유닛 (34) 은 디폴트 값 (즉, "MAX") 으로 MINRC 레지스터 (42) 에 저장된 값을 겹쳐쓰기할 수도 있다. 디폴트 값은 피호출자 프로그램에 대한 MINRC 를 초기화하기 위하여 이용될 수도 있다. 피호출자 프로그램의 MINRC 는 서브루틴의 실행이 개시될 때에 활성인 모든 스레드들에 대한 가장 작은 재개 카운터에 대응하므로, 피호출자 프로그램에 대한 MINRC 를 계산하기 위하여 이용된 모든 재개 카운터들은 피호출자 프로그램의 MINRC 가 초기화될 때에 활성 스레드들과 연관된다. 위에서 논의된 바와 같이, 스레드가 활성일 때, 재개 카운터는, 이 경우에 "MAX" 에 의해 나타내는 "무한한 값" (예를 들어, 최대 레지스터 값) 과 동일할 수도 있다. 그러므로, 초기화될 때, 피호출자 프로그램에 대한 MINRC 는 "MAX" 와 동일하고, 이것은 피호출자 프로그램이 실행을 시작할 때에 활성 스레드들과 연관된 모든 재개 카운터들의 값이다. 피호출자 프로그램의 실행이 진전됨에 따라, 하나 이상의 스레드들이 비활성화될 수도 있고, 다음으로, 이것은 MINRC 로 하여금 다른 값들로 업데이트되게 할 것이다.

[0231] 호출 명령은 피호출자 서브루틴의 제 1 명령에 대응하는 타겟 프로그램 명령을 표시하는 타겟 값을 포함할 수도 있다. 호출 명령을 실행할 때, 제어 흐름 유닛 (34) 은 피호출자 서브루틴의 타겟 명령에 대응하는 값으로 프로그램 카운터 (28) 를 로딩할 수도 있다. 또한, 호출 명령을 실행할 때, 제어 흐름 유닛 (34) 은 호출자 프로그램에서의 호출 명령 이후의 다음의 순차적인 명령에 대응하는 복귀 어드레스를 스택 저장 구조 (52) 에서의 호출 스택 상으로 푸시할 수도 있다.

[0232] 의사-코드에서 특정된 호출 스택은 메인 프로그램으로의 진입 포인트에서 초기에 비어 있을 수도 있다. 복귀 어드레스를 호출 스택 상으로 푸시하는 것에 추가하여, 다른 상태 변수들은 또한, 호출 명령의 실행 시에 호출 스택 상으로 푸시될 수도 있고, 복귀 명령의 실행 후에 호출 스택에서 팝핑될 수도 있다.

[0233] 도 26 은 본 개시물에 따라 복귀 명령 프로세싱을 구현하기 위한 일 예의 의사-코드를 예시한다. 일부의 예들에서, 도 26 에서 예시된 의사-코드는 도 9 에서 예시된 흐름도에 대응할 수도 있다. 도 26 에 도시된 바와 같이, 제어 흐름 유닛 (34) 은 스택 저장 구조 (52) 에서의 MINRC 스택에서 가장 최근에 저장된 MINRC 값을 팝핑할 수도 있다. 가장 최근에 저장된 MINRC 값은 호출자 프로그램에 대응하는 MINRC 의 저장된 상태에 대응할 수도 있다. 제어 흐름 유닛 (34) 은 팝핑된 MINRC 값으로 MINRC 레지스터 (42) 에 저장된 값을 겹쳐쓰기할 수도 있다. 추가적으로, 제어 흐름 유닛 (34) 은 스택 저장 구조 (52) 에서의 호출 스택으로부터 가장 최근에 저장된 복귀 어드레스를 팝핑할 수도 있고, 팝핑된 복귀 어드레스에 대응하는 값을 프로그램 카운터

(28) 로 로딩할 수도 있다.

[0234] 복귀 명령을 실행한 후, MINRC 레지스터 (42) 는 서브루틴 호출 명령을 실행하기 이전에 그것이 있었던 상태로 복원된다. 위에서 설명된 바와 같이 스택 상으로 그리고 스택에서 MINRC 값들을 푸시 및 팝핑함으로써, 본 개시물의 기법들은 일부의 예들에서, 단지 단일의 MINRC 레지스터 (42) 를 유지하면서 서브루틴-특정 MINRC 들을 구현가능할 수도 있다.

[0235] 위에서 재현된 일 예의 의사-코드는 복귀 어드레스의 상태 및 MINRC 의 상태를 2 개의 상이한 스택들에 저장한다. 즉, 복귀 어드레스는 호출 스택 상으로 푸시되고, MINRC 는 MINRC 스택 상으로 푸시된다. 그러나, 다른 예들에서는, 복귀 어드레스 및 MINRC 가 단일 스택 프레임의 일부와 동일한 스택 상으로 푸시될 수도 있다.

[0236] 도 27 은 본 개시물에 따라 MINRC 업데이트 동작을 수행하기 위한 일 예의 의사-코드를 예시한다. 일부의 예들에서, 도 27 에서 예시된 의사-코드는 도 17 에서 예시된 흐름도에 대응할 수도 있다. 도 27 에 도시된 바와 같이, 제어 흐름 유닛 (34) 은 프로세싱 시스템 (10) 에서 실행되는 모든 비활성 스레드들과 연관된 재개 카운터들의 전부를 통해 순환한다. 각각의 비활성 스레드에 대하여, 다음의 조건들의 양자가 충족되는 경우, 제어 흐름 유닛 (34) 은 MINRC 를 각각의 비활성 스레드에 대응하는 재개 카운터 값과 동일하게 설정할 수도 있다: (1) 비활성 스레드에 대한 재개 카운터 값은 현재의 MINRC 값보다 더 작고; (2) 비활성 스레드에 대한 재개 카운터 값은 서브루틴의 진입 포인트 (즉, 서브루틴의 시작 어드레스에 대응하는 프로그램 카운터 값) 이상이다. 비활성 스레드에 대한 재개 카운터 값이 서브루틴의 진입 포인트 이상인지 여부를 MINRC 의 업데이트의 조건으로 함으로써, 도 27 에 도시된 MINRC 업데이트 동작은 서브루틴의 실행을 개시하기 이전에 비활성이었던 스레드들과 연관되는 하나 이상의 재개 카운터들이 결과적인 MINRC 값에 영향을 주는 것을 방지할 수도 있다. 이러한 방법으로, 제어 흐름 유닛 (34) 은 서브루틴-특정 MINRC 에 대한 업데이트된 MINRC 값이 서브루틴에 대해 할당된 프로그램 공간 내에 있는 것을 보장할 수도 있다.

[0237] 도 28 은 본 개시물에 따라 MINRC 업데이트 동작을 수행하기 위한 일 예의 의사-코드를 예시한다. 일부의 예들에서, 도 27 에서 예시된 의사-코드는 도 18 에서 예시된 흐름도에 대응할 수도 있다. 도 28 에서 도시된 바와 같이, 제어 흐름 유닛 (34) 은 프로세싱 시스템 (10) 에서 실행되는 모든 비활성 스레드들과 연관된 재개 카운터들의 전부를 통해 순환한다. 각각의 비활성 스레드에 대하여, 다음의 조건들의 양자가 충족되는 경우, 제어 흐름 유닛 (34) 은 MINRC 를 각각의 비활성 스레드에 대응하는 재개 카운터 값과 동일하게 설정할 수도 있다: (1) 비활성 스레드에 대한 재개 카운터 값은 현재의 MINRC 값보다 더 작고; (2) 각각의 비활성 스레드와 연관된 플래그는 서브루틴 프로그램의 실행이 개시되었을 때에 스레드가 활성이었음을 표시한다. 따라서, 위에서 설명된 MINRC 업데이트 동작은 서브루틴 프로그램의 실행이 개시되었을 때에 비활성이었던 스레드들과 연관되는 그러한 재개 카운터들을 효과적으로 배제한다. 이러한 방법으로, 제어 흐름 유닛 (34) 은 서브루틴-특정 MINRC 에 대한 업데이트된 MINRC 값이 서브루틴에 대해 할당된 프로그램 공간 내에 있는 것을 보장할 수도 있다.

[0238] 도 29 는 본 개시물에 따라 프로그램 모듈-특정 MINRC 들에 기초하여 프로세싱 시스템을 제어하기 위한 일 예의 기법을 예시하는 흐름도이다. 제어 유닛 (12) 은 제 1 MINRC 에 기초하여 프로그램의 실행을 제어한다 (330). 제 1 MINRC 는 복수의 스레드들과 연관된 복수의 재개 카운터 값들 중의 가장 작은 재개 카운터 값을 표시하는 값을 특정한다. 제어 유닛 (12) 은 서브루틴과 연관된 제 2 MINRC 에 기초하여 프로그램의 서브루틴의 실행을 제어한다 (332). 제 2 MINRC 는, 서브루틴의 실행이 개시될 때에 활성인 스레드들의 전부에 대응하는 복수의 재개 카운터 값들의 서브세트로부터의 가장 작은 재개 카운터 값을 표시하는 값을 특정한다. 일부의 예들에서, MINRC 에 기초하여 프로그램 모듈 (예를 들어, 메인 프로그램 또는 서브루틴) 의 실행을 제어하는 것은 분기 조건이 MINRC 에 기초하여 모든 활성 스레드들에 대해 충족되는 순방향 분기 명령 또는 순방향 점프 명령에 응답하여 실행하기 위한 다음 명령을 선택하는 것을 포함할 수도 있다.

[0239] 일부의 예들에서, 재개 카운터 값들의 각각은, 각각의 재개 카운터 값에 대응하는 스레드들의 각각의 하나가 스레드들의 각각의 하나가 비활성인 경우에 활성화되도록 스케줄링되는 프로그램 카운터 값을 표시할 수도 있다.
추가의 예들에서, 각각의 재개 카운터 값에 대응하는 스레드들의 각각의 하나가 활성인 경우에는, 재개 카운터 값들의 각각이 디폴트 값과 동일하다.

[0240] 도 30 은 본 개시물에 따라 서브루틴 호출 명령을 실행하기 위한 일 예의 기법을 예시하는 흐름도이다. 제어 유닛 (12) 은 호출 명령을 실행한다 (334). 호출 명령을 실행하는 것에 응답하여, 제어 유닛 (12) 은 호출자 프로그램에 대응하는 제 1 MINRC 의 상태를 저장한다 (336). 예를 들어, 제어 유닛 (12) 은 MINRC 레

지스터 (42) 에 저장된 값을 스택 저장 구조 (52) 상으로 푸시할 수도 있다. 호출 명령을 실행하는 것에 응답하여, 제어 유닛 (12) 은 피호출자 서브루틴 프로그램의 실행이, 피호출자 서브루틴 프로그램에 대응하는 제 2 MINRC 에 기초하여 제어되게 한다 (338). 즉, 제어 유닛 (12) 은 제 2 MINRC 로 하여금, 서브루틴 프로그램의 실행을 제어하도록 (예를 들어, 분기 조건이 모든 활성 스레드들에 대해 충족되는 순방향 분기 명령 또는 순방향 점프 명령에 응답하여 다음 명령의 선택을 제어하도록) 이용되게 할 수도 있다. 일부의 예들에서는, 제 2 MINRC 로 하여금 서브루틴 프로그램의 실행을 제어하도록 이용되게 하기 위하여, 제어 유닛 (12) 은 MINRC 레지스터 (42) 가 피호출자 프로그램에 대응하는 제 2 MINRC 에 대한 초기 MINRC 값을 저장하도록 MINRC 레지스터 (42) 를 초기화할 수도 있다. 일부의 예들에서, MINRC 레지스터 (42) 를 초기화하는 것은 제 2 MINRC 에 대응하는 초기 MINRC 값으로 제 1 MINRC 에 대응하는 MINRC 레지스터 (42) 에 저장된 값을 겹쳐쓰기하는 것을 포함할 수도 있다. MINRC 레지스터 (42) 에 저장된 초기 MINRC 값은 디폴트 MINRC 값 (예를 들어, 최대 레지스터 값, 또는 프로그램을 위해 필요한 프로그램 카운터 범위보다 더 큰 값) 일 수도 있다.

[0241] 도 31 은 본 개시물에 따라 서브루틴 복귀 명령을 실행하기 위한 일 예의 기법을 예시하는 흐름도이다. 제어 유닛 (12) 은 복귀 명령을 실행한다 (340). 복귀 명령을 실행하는 것에 응답하여, 제어 유닛 (12) 은 현재 실행되고 있는 서브루틴의 호출자 프로그램에 대응하는 제 1 MINRC 의 저장된 상태를 복원한다 (342). 예를 들어, 제어 유닛 (12) 은, 제 1 MINRC 의 저장된 상태에 대응하는 스택 저장 구조 (52) 에 저장된 MINRC 값을 팝핑할 수도 있다. 복귀 명령을 실행하는 것에 응답하여, 제어 유닛 (12) 은 호출자 프로그램의 실행이, 호출자 프로그램에 대응하는 제 1 MINRC 에 기초하여 제어되게 한다 (344). 즉, 제어 유닛 (12) 은 제 1 MINRC 로 하여금, 호출자 프로그램의 실행을 제어하도록 (예를 들어, 분기 조건이 모든 활성 스레드들에 대해 충족되는 순방향 분기 명령 또는 순방향 점프 명령에 응답하여 다음 명령의 선택을 제어하도록) 이용되게 할 수도 있다. 일부의 예들에서, 호출자 프로그램의 실행이 호출자 프로그램에 대응하는 제 1 MINRC 에 기초하여 제어되게 하기 위하여, 제어 유닛 (12) 은 팝핑된 MINRC 값을 MINRC 레지스터 (42) 에 저장할 수도 있다. 일부의 예들에서, 팝핑된 MINRC 값을 MINRC 레지스터 (42) 에 저장하는 것은 팝핑된 MINRC 값에 대응하는 값으로, 서브루틴에 대한 MINRC 에 대응하는 MINRC 레지스터 (42) 에 저장된 값을 겹쳐쓰기하는 것을 포함할 수도 있다.

[0242] 본 개시물에서 설명된 기법들은 적어도 부분적으로 하드웨어, 소프트웨어, 펌웨어, 또는 그 임의의 조합으로 구현될 수도 있다. 예를 들어, 설명된 기법들의 다양한 양태들은 하나 이상의 마이크로프로세서들, 디지털 신호 프로세서 (digital signal processor; DSP) 들, 주문형 집적 회로 (application specific integrated circuit; ASIC) 들, 필드 프로그래밍가능한 게이트 어레이 (field programmable gate array; FPGA) 들, 또는 임의의 다른 등가의 집적 또는 개별 논리 회로부 뿐만 아니라 이러한 컴포넌트들의 임의의 조합들을 포함하는 하나 이상의 프로세서들 내에서 구현될 수도 있다. 용어 "프로세서" 또는 "프로세싱 회로부" 는 단독 또는 다른 논리 회로부와 조합하는 상기한 논리 회로부, 또는 프로세싱을 수행하는 개별 하드웨어와 같은 임의의 다른 등가의 회로부 중의 임의의 것을 일반적으로 지칭할 수도 있다.

[0243] 이러한 하드웨어, 소프트웨어, 및 펌웨어는 본 개시물에서 설명된 다양한 동작들 및 기능들을 지원하기 위하여 동일한 디바이스 내에서 또는 별도의 디바이스들 내에서 구현될 수도 있다. 추가적으로, 설명된 유닛들, 모듈들 또는 컴포넌트들 중의 임의의 것은 개별적이지만 상호 동작가능한 논리 디바이스들로서 함께 또는 별도로 구현될 수도 있다. 모듈들 또는 유닛들로서의 상이한 특징들의 묘사는 상이한 기능적 양태들을 강조하도록 의도된 것이고, 이러한 모듈들 또는 유닛들이 별도의 하드웨어 또는 소프트웨어 컴포넌트들에 의해 실현되어야 하는 것을 반드시 암시하지는 않는다. 오히려, 하나 이상의 모듈들 또는 유닛들과 연관된 기능성은 별도의 하드웨어, 펌웨어, 및/또는 소프트웨어 컴포넌트들에 의해 수행될 수도 있거나, 공통적인 또는 별도의 하드웨어 또는 소프트웨어 컴포넌트들 내에서 통합될 수도 있다.

[0244] 본 개시물에서 설명된 기법들은 또한, 명령들을 저장하는 컴퓨터 판독가능한 저장 매체와 같은 컴퓨터 판독가능한 매체에서 저장, 구체화, 또는 인코딩될 수도 있다. 컴퓨터 판독가능한 매체에서 내장되거나 인코딩된 명령들은 예를 들어, 명령들이 하나 이상의 프로세서들에 의해 실행될 때, 하나 이상의 프로세서들로 하여금 본원에서 설명된 기법들을 수행하게 할 수도 있다. 컴퓨터 판독가능한 저장 매체들은 랜덤 액세스 메모리 (random access memory; RAM), 판독 전용 메모리 (read only memory; ROM), 프로그래밍가능한 판독 전용 메모리 (programmable read only memory; PROM), 소거가능 프로그래밍가능한 판독 전용 메모리 (erasable programmable read only memory; EPROM), 전자적 소거가능 프로그래밍가능한 판독 전용 메모리 (elctronically erasable programmable read only memory; EEPROM), 플래시 메모리, 하드 디스크, CD-ROM, 플로피 디스크, 카세트, 자기 매체들, 광학 매체들, 또는 유형의 (tangible) 다른 컴퓨터 판독가능한 저장 매체들을 포함할 수도

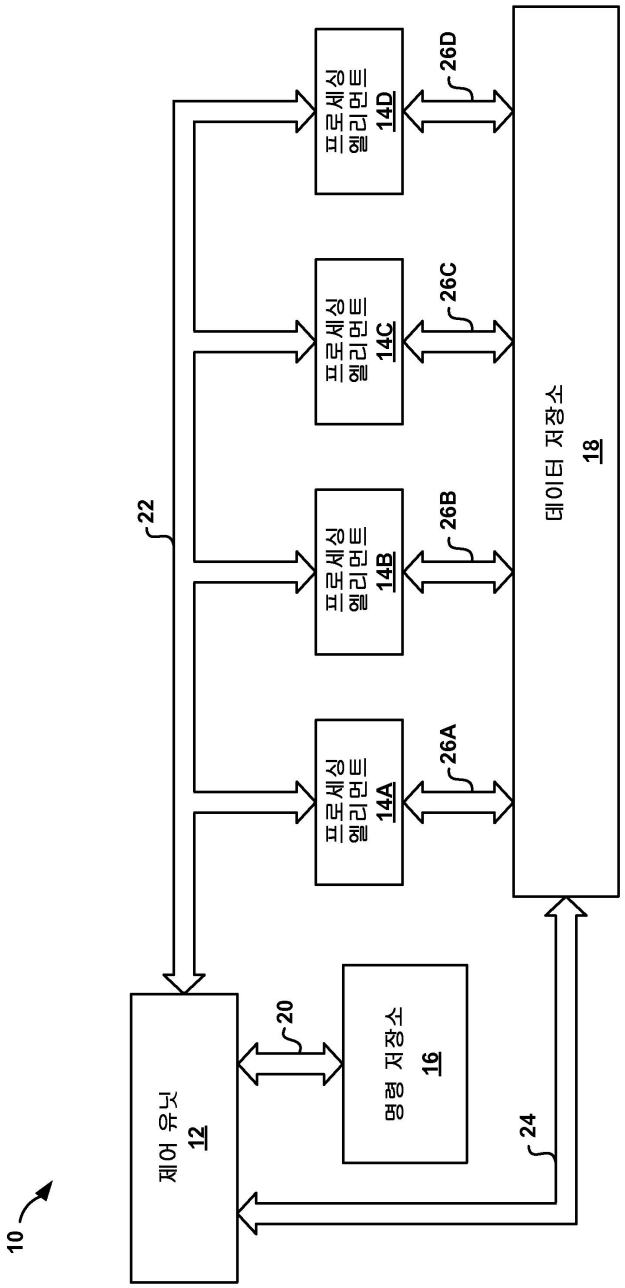
있다.

[0245] 컴퓨터 판독가능한 매체들은 위에서 열거된 것들과 같은 유형의 저장 매체에 대응하는 컴퓨터 판독가능한 저장 매체들을 포함할 수도 있다. 컴퓨터 판독가능한 매체들은 또한, 예를 들어, 통신 프로토콜에 따라 하나의 장소로부터 또 다른 장소로 컴퓨터 프로그램의 전송을 용이하게 하는 임의의 매체를 포함하는 통신 매체들을 포함할 수도 있다. 이러한 방식으로, 어구 "컴퓨터 판독가능한 매체들" 은 일반적으로 (1) 비일시적인 유형의 컴퓨터 판독가능한 저장 매체들, 및 (2) 일시적인 신호 또는 반송파와 같은 비유형의 (non-tangible) 컴퓨터 판독가능한 통신 매체에 대응할 수도 있다.

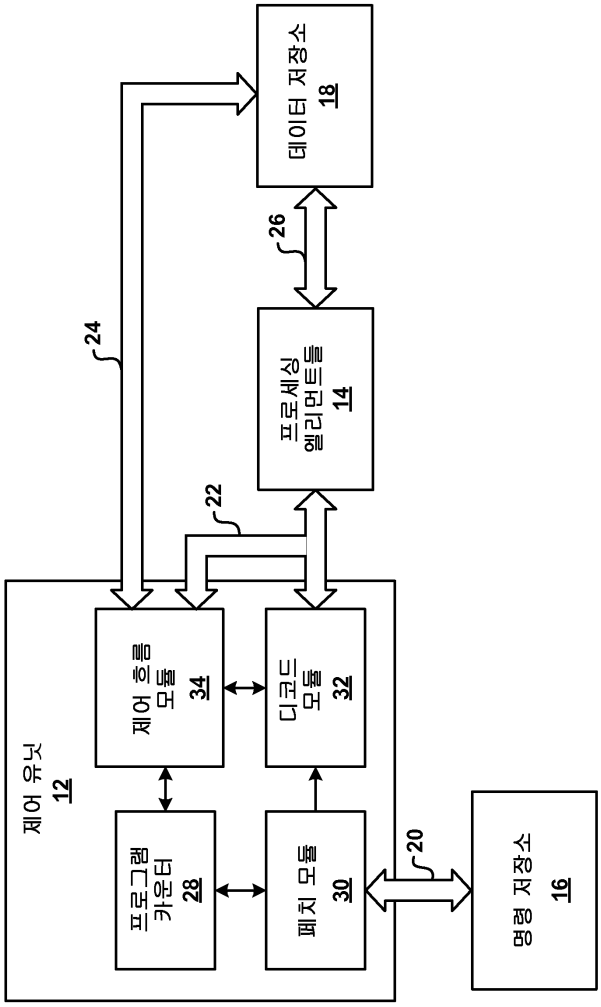
[0246] 다양한 양태들 및 예들이 설명되었다. 그러나, 다음의 청구항들의 범위로부터 이탈하지 않고 본 개시물의 구조 또는 기법들에 대해 수정들이 행해질 수 있다.

도면

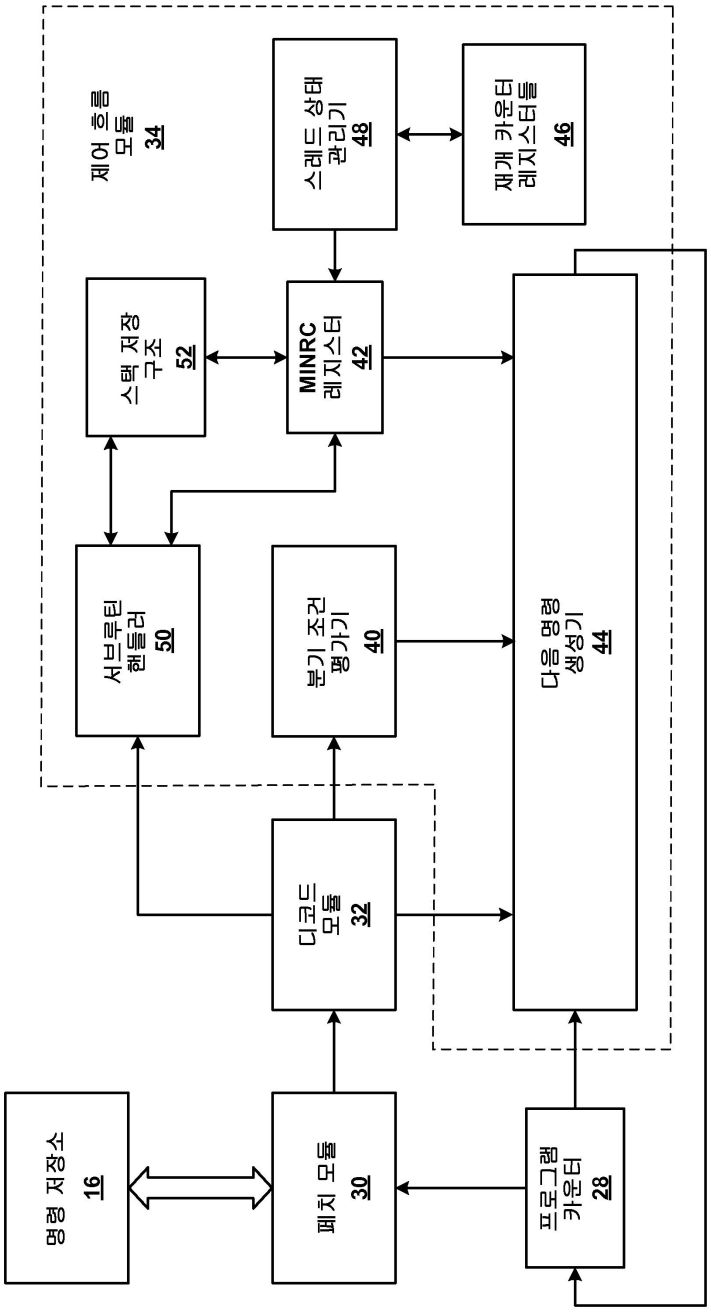
도면1



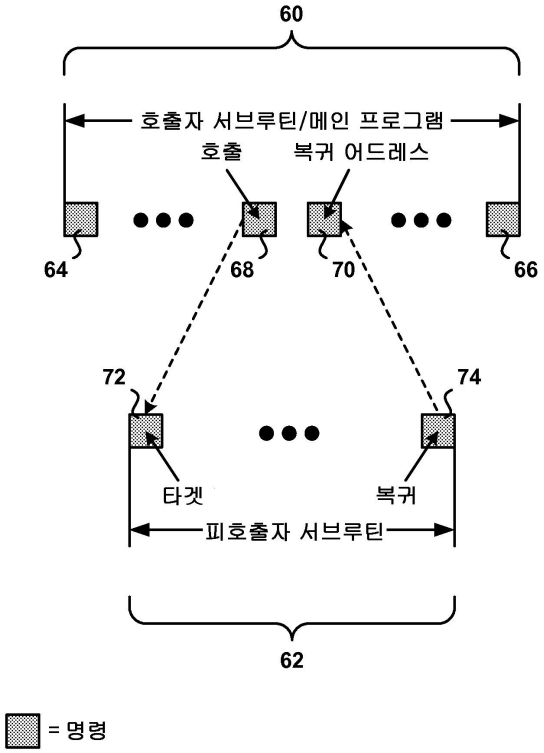
도면2


데이터 저장소
18
26
프로세싱 엘리먼트들
14
24
22
제어 유닛
12
제어 흐름 모듈
34
디코드 모듈
32
프로그램 카운터
28
페치 모듈
30
20
명령 저장소
16

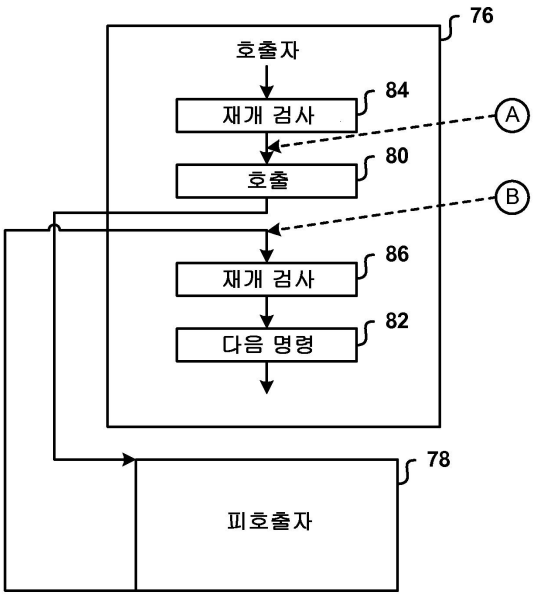
도면3


제어 흐름
모듈
34
스레드 상태
관리기
48
재개 카운터
레지스터들
46
스택 저장
구조
52
MINRC
레지스터
42
서브루틴
핸들러
50
분기 조건
평가기
40
다음 명령
생성기
44
디코드
모듈
32
명령 저장소
16
페치 모듈
30
프로그램
카운터
28

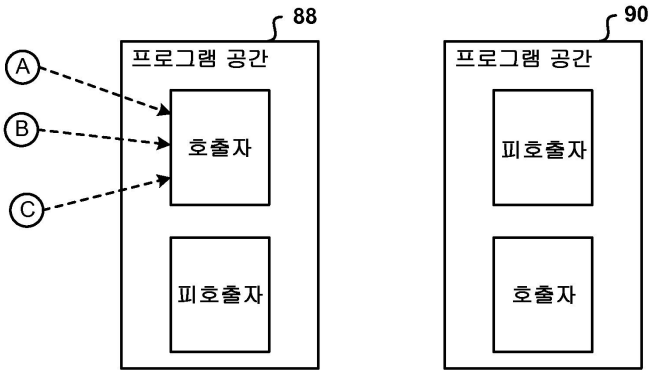
도면4


60
호출자 서브루틴/메인 프로그램
호출
복귀 어드레스
64
68
70
66
72
74
타겟
복귀
피호출자 서브루틴
62
= 명령

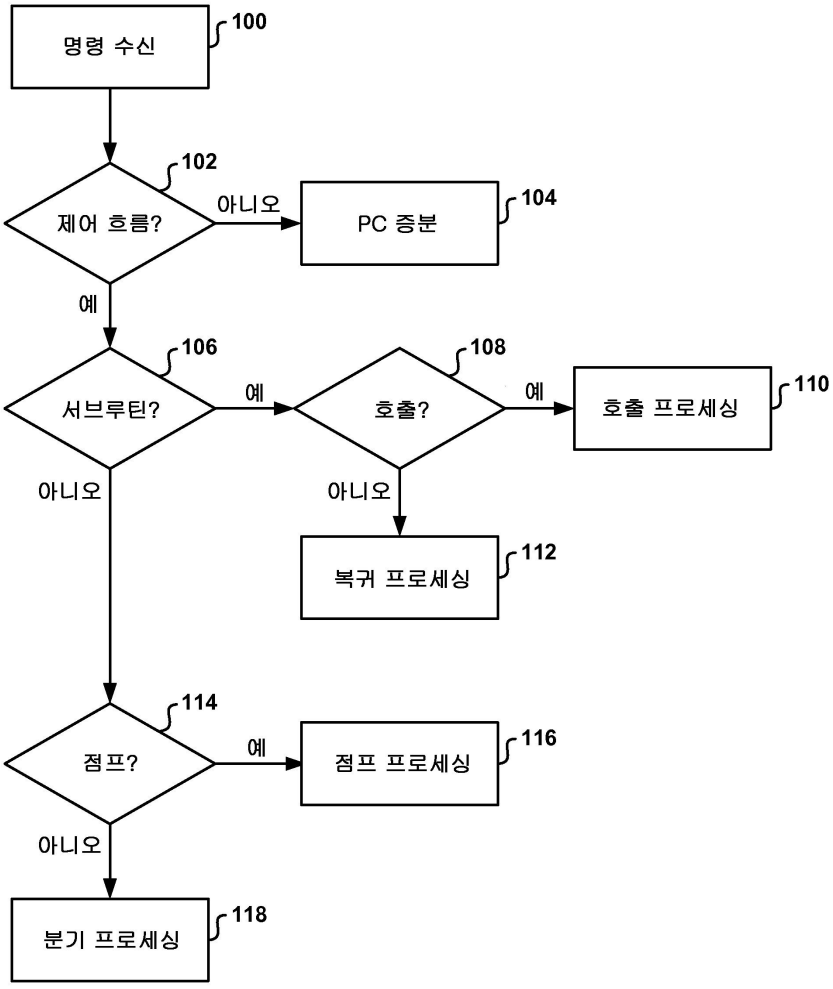
도면5


76
호출자
84
재개 검사
A
80
호출
B
86
재개 검사
82
다음 명령
78
피호출자

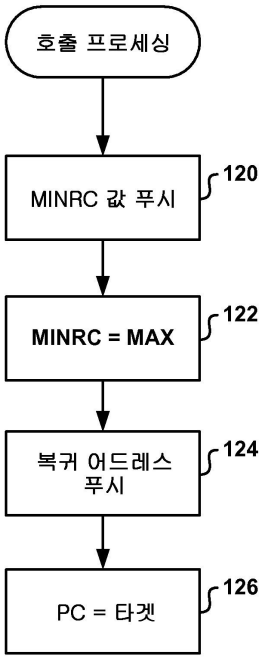
도면6


88
프로그램 공간
A
B
C
호출자
피호출자
90
프로그램 공간
피호출자
호출자

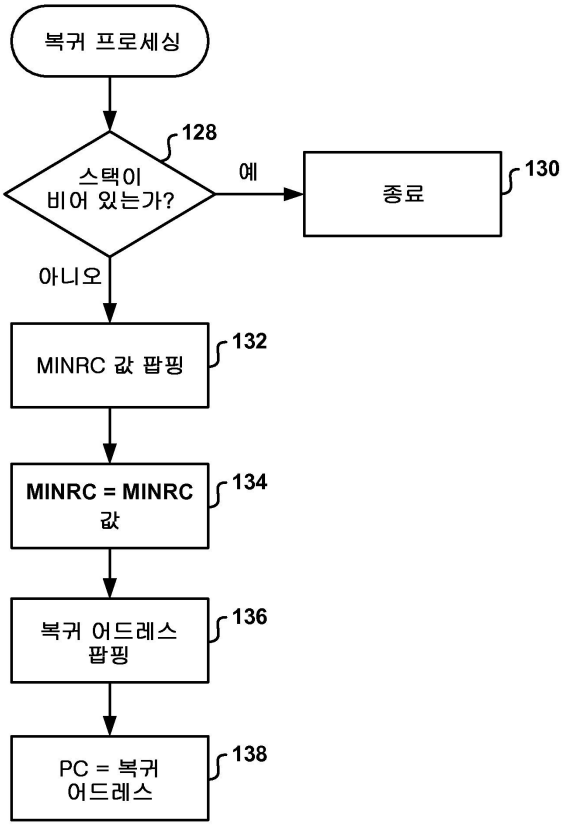
도면7


명령 수신
100
제어 흐름?
102
아니오
PC 증분
104
예
서브루틴?
106
예
호출?
108
예
호출 프로세싱
110
아니오
아니오
복귀 프로세싱
112
점프?
114
예
점프 프로세싱
116
아니오
분기 프로세싱
118

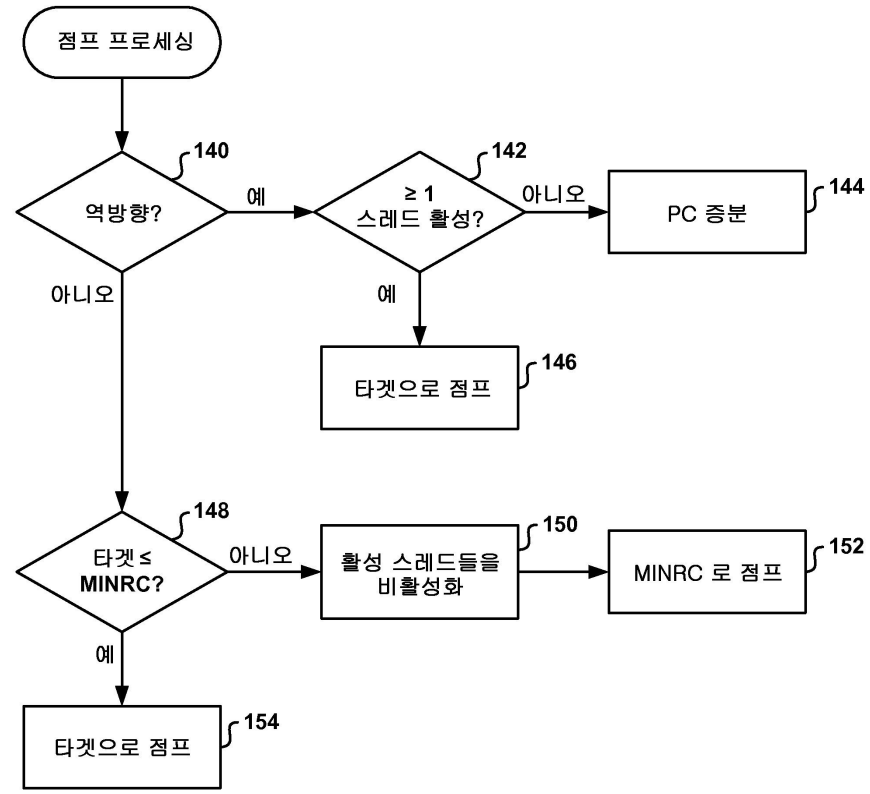
도면8


호출 프로세싱
MINRC 값 푸시
120
MINRC = MAX
122
복귀 어드레스 푸시
124
PC = 타겟
126

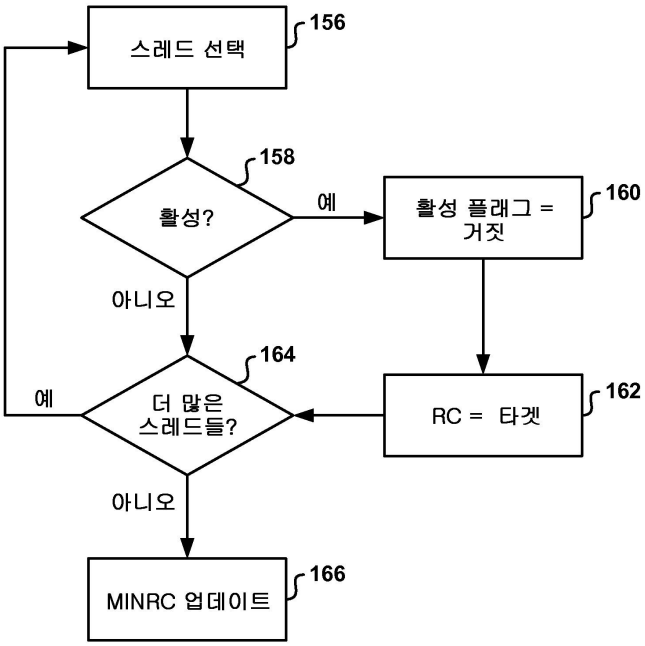
도면9


복귀 프로세싱
128
스택이 비어 있는가?
예
종료
130
아니오
MINRC 값 팝핑
132
MINRC = MINRC 값
134
복귀 어드레스 팝핑
136
PC = 복귀 어드레스
138

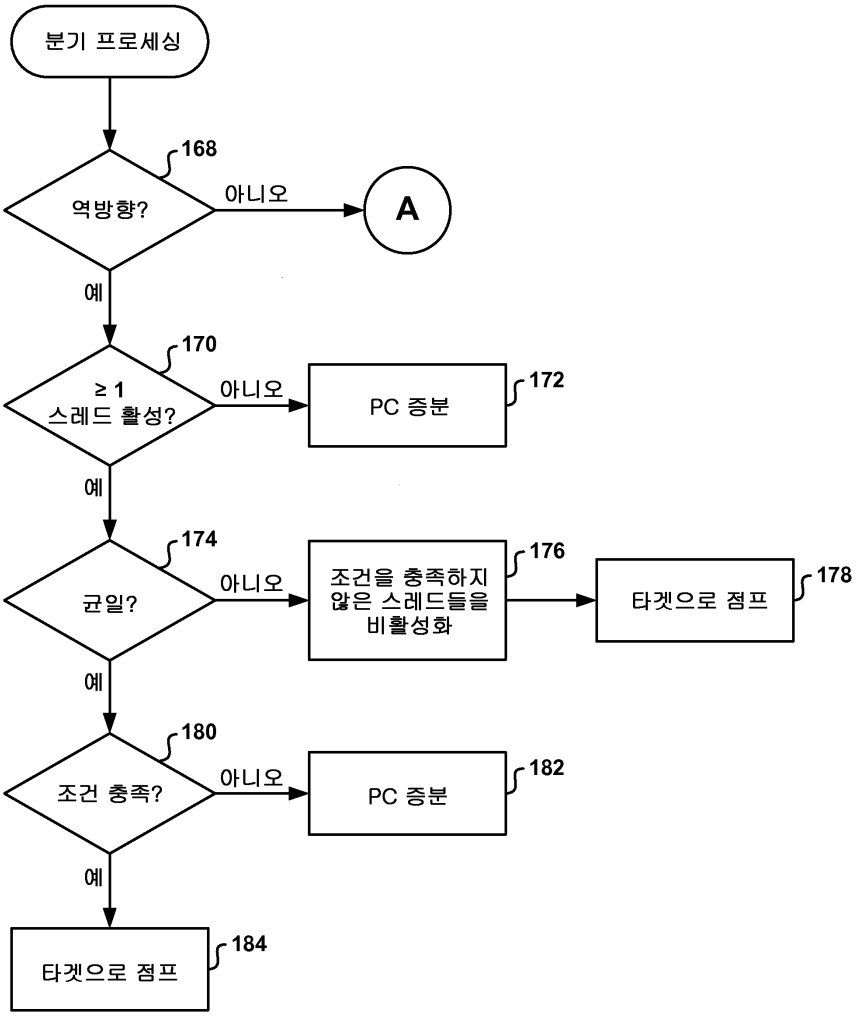
도면10


점프 프로세싱
140
역방향?
예
142
≥ 1
스레드 활성?
아니오
144
PC 증분
아니오
예
146
타겟으로 점프
148
타겟 ≤
MINRC?
아니오
150
활성 스레드들을
비활성화
152
MINRC 로 점프
예
154
타겟으로 점프

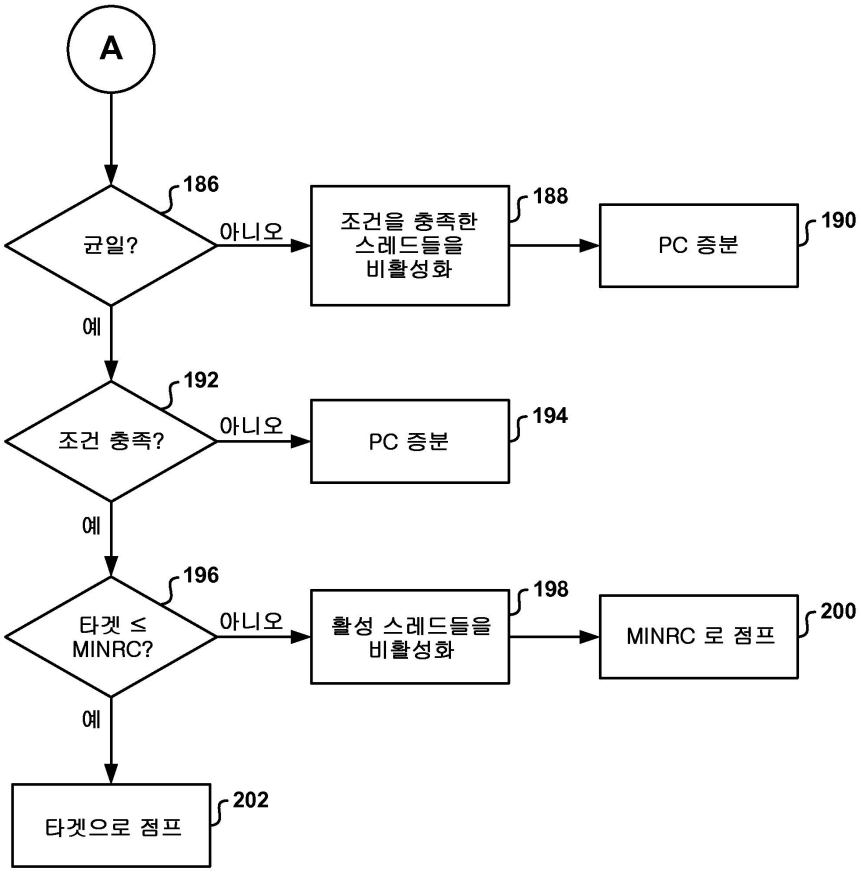
도면11


156
스레드 선택
158
활성?
예
160
활성 플래그 =
거짓
아니오
164
더 많은
스레드들?
예
162
RC = 타겟
아니오
166
MINRC 업데이트

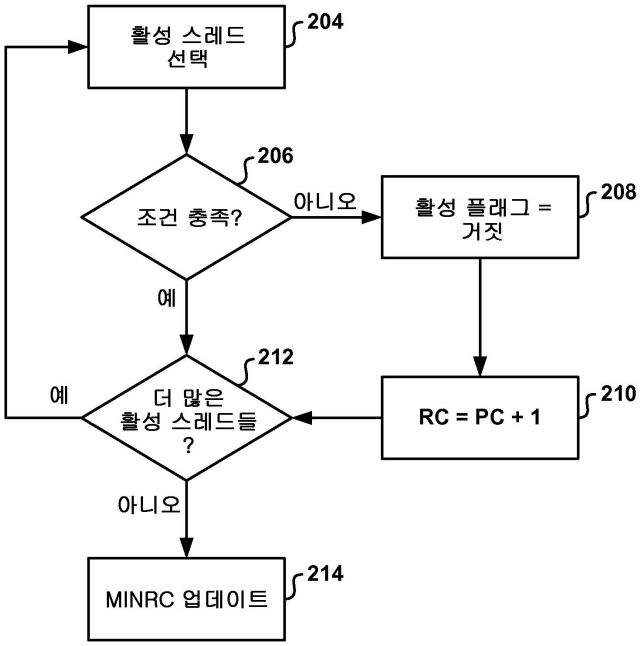
도면12


분기 프로세싱
168
역방향?
아니오
A
예
170
≥ 1
스레드 활성?
아니오
PC 증분
172
예
174
균일?
아니오
조건을 충족하지
않은 스레드들을
비활성화
176
타겟으로 점프
178
예
180
조건 충족?
아니오
PC 증분
182
예
타겟으로 점프
184

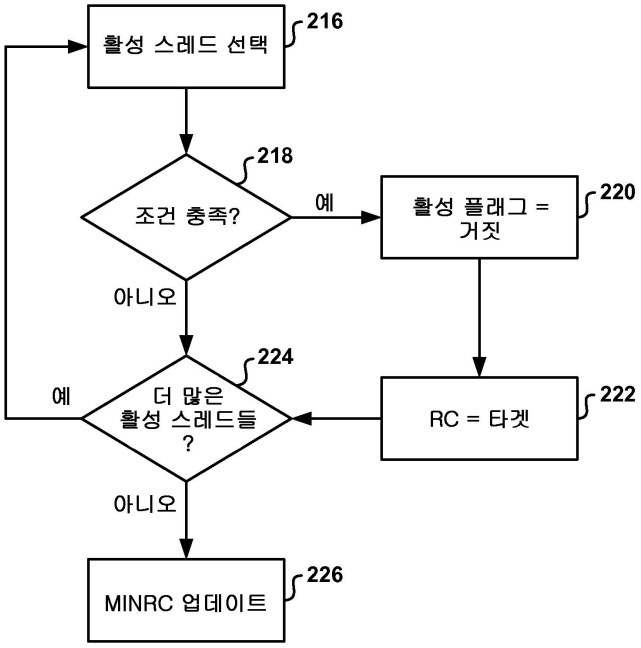
도면13


A
186
균일?
아니오
188
조건을 충족한 스레드들을 비활성화
190
PC 증분
예
192
조건 충족?
아니오
194
PC 증분
예
196
타겟 ≤ MINRC?
아니오
198
활성 스레드들을 비활성화
200
MINRC 로 점프
예
202
타겟으로 점프

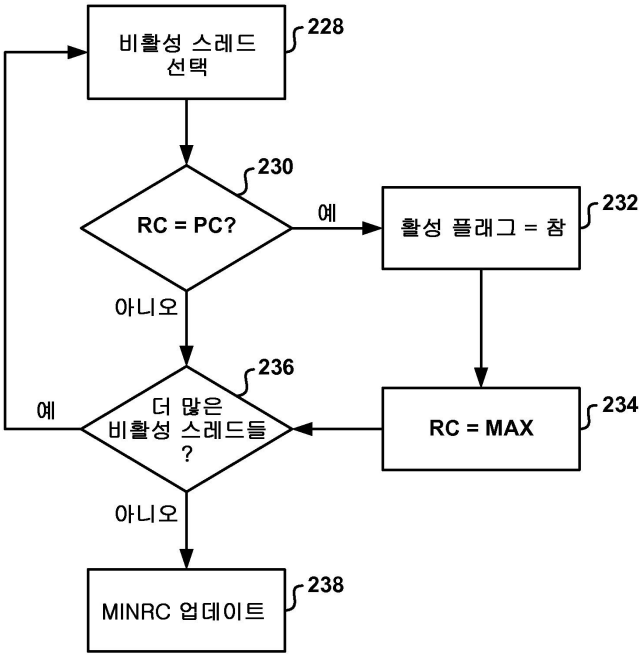
도면14


204
활성 스레드 선택
206
조건 충족?
아니오
208
활성 플래그 = 거짓
예
212
더 많은 활성 스레드들 ?
예
210
RC = PC + 1
아니오
214
MINRC 업데이트

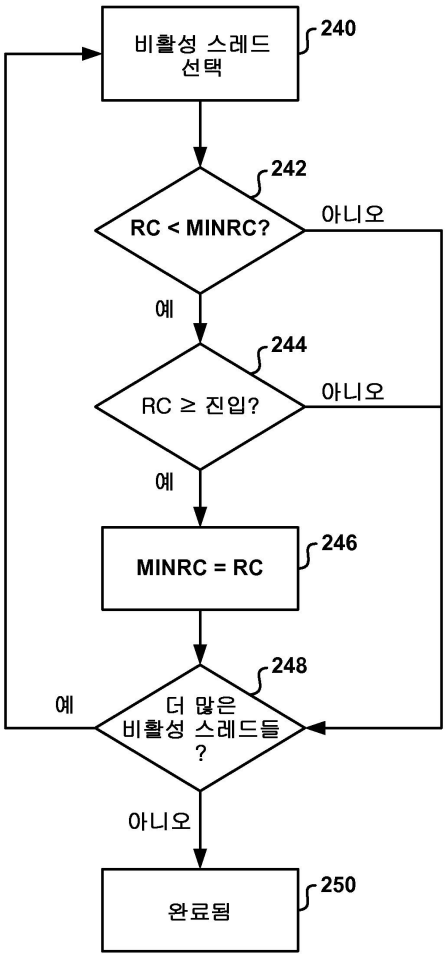
도면15


활성 스레드 선택
216
조건 충족?
218
예
활성 플래그 = 거짓
220
아니오
더 많은 활성 스레드들 ?
224
예
RC = 타겟
222
아니오
MINRC 업데이트
226

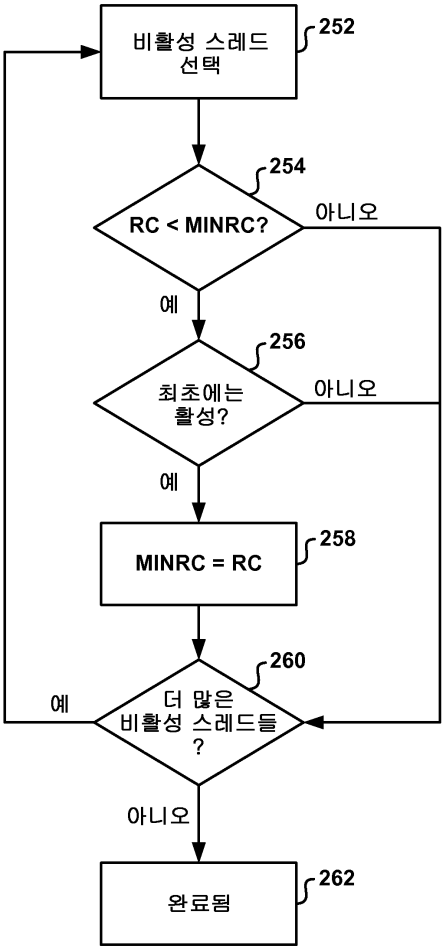
도면16


비활성 스레드 선택
228
RC = PC?
230
예
활성 플래그 = 참
232
아니오
더 많은 비활성 스레드들 ?
236
예
RC = MAX
234
아니오
MINRC 업데이트
238

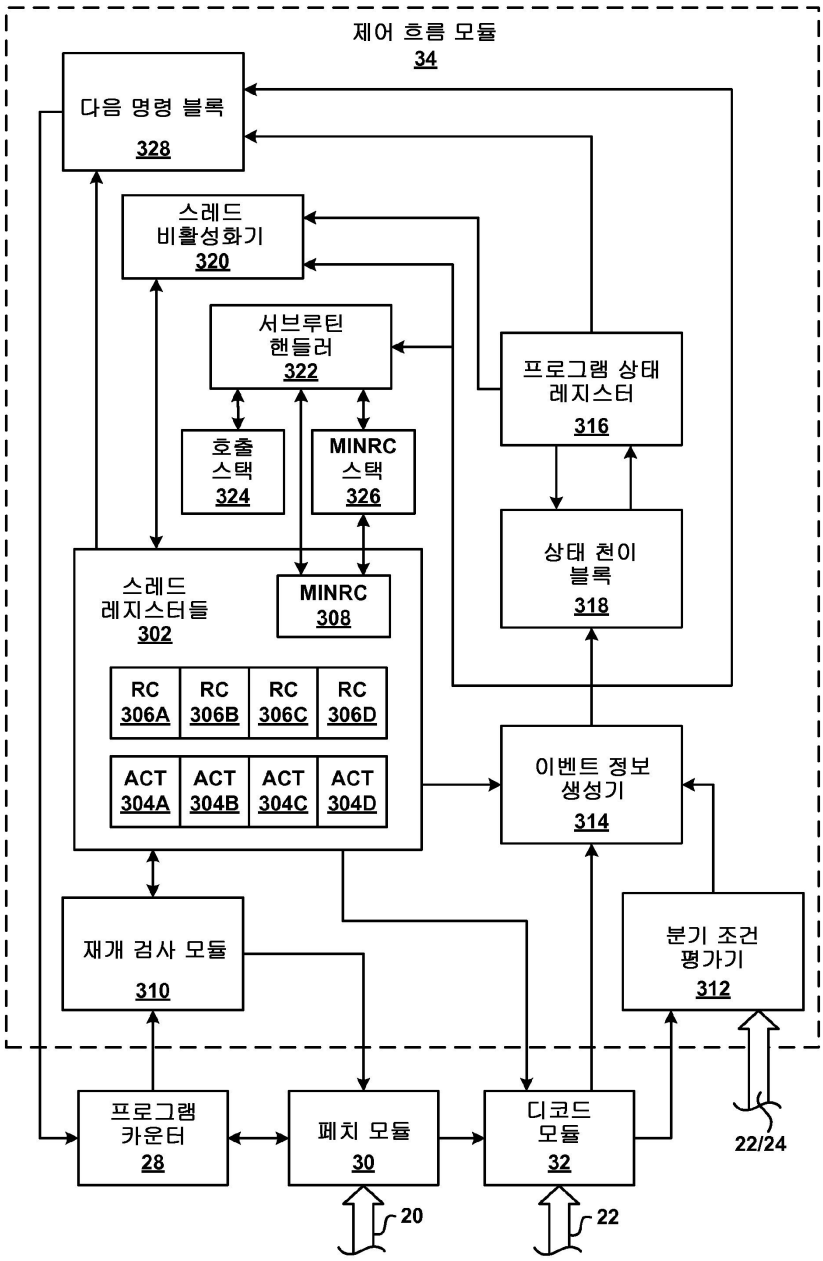
도면17


비활성 스레드 선택
240
242
RC < MINRC?
아니오
예
244
RC ≥ 진입?
아니오
예
MINRC = RC
246
248
예
더 많은 비활성 스레드들 ?
아니오
완료됨
250

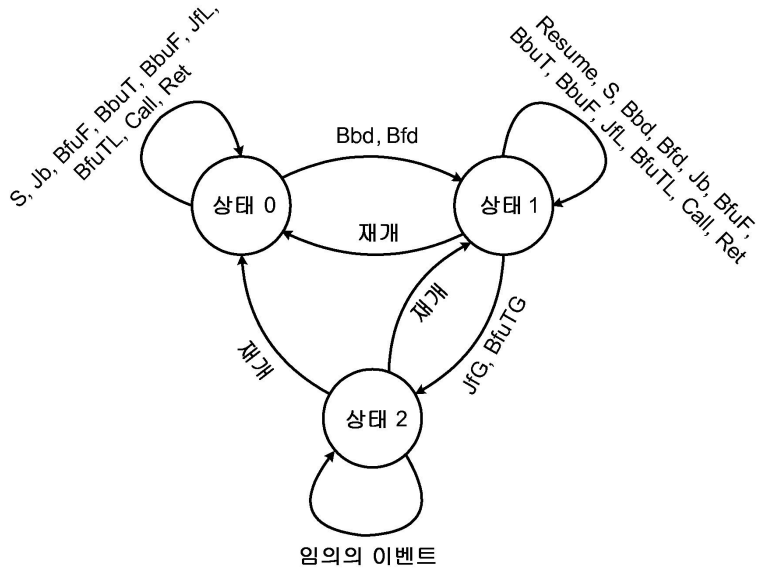
도면18


비활성 스레드 선택
252
254
RC < MINRC?
아니오
예
256
최초에는 활성?
아니오
예
MINRC = RC
258
260
더 많은 비활성 스레드들 ?
예
아니오
완료됨
262

도면19


제어 흐름 모듈
34
다음 명령 블록
328
스레드
비활성화기
320
서브루틴
핸들러
322
프로그램 상태
레지스터
316
호출
스택
324
MINRC
스택
326
상태 천이
블록
318
스레드
레지스터들
302
MINRC
308
RC
306A
RC
306B
RC
306C
RC
306D
ACT
304A
ACT
304B
ACT
304C
ACT
304D
이벤트 정보
생성기
314
재개 검사 모듈
310
분기 조건
평가기
312
프로그램
카운터
28
페치 모듈
30
디코드
모듈
32
22/24
20
22

도면20



도면21

과거 상태	새로운 상태	이벤트	액션
상태 0	상태 0	S	PC+1
		Jb	타겟으로 점프
		BfuF	PC+1
		BbuT	타겟으로 균일하게 분기
		BbuF	PC+1
		JfL	타겟으로 점프
		BfuTL	타겟으로 균일하게 분기
		Call	서브루틴 진입 포인트로 점프
		Ret	복귀 어드레스로 점프
상태 0	상태 1	Bbd	타겟으로 분기. 거짓 조건을 갖는 활성 스레드들은 비활성으로 됨.
		Bfd	PC+1. 참인 조건을 갖는 활성 스레드들은 비활성으로 됨.
상태 1	상태 0	n/a	재개. 모든 비활성 스레드들은 활성으로 됨
상태 1	상태 1	S	PC+1
		Jb	타겟으로 점프
		BbuF	PC+1
		BbuT	타겟으로 균일하게 분기
		BfuF	PC+1
		Bbd	타겟으로 분기. 거짓 조건을 갖는 활성 스레드들은
		Bfd	비활성으로 됨.
		n/a	PC+1. 참인 조건을 갖는 활성 스레드들은 비활성으로 됨.
		JfL	재개. 일부의 비활성 스레드들은 활성으로 됨.
		BfuTL	타겟으로 점프
		Call	타겟으로 균일하게 분기
		Ret	서브루틴 진입 포인트로 점프
			복귀 어드레스로 점프
상태 1	상태 2	JfG	MINRC 로 점프. 모든 활성 스레드들은 비활성으로 됨.
		BfuTG	MINRC 로 점프. 모든 활성 스레드들은 비활성으로 됨.
상태 2	상태 0	n/a	재개. 모든 비활성 스레드들은 활성으로 됨
상태 2	상태 1	n/a	재개. 일부의 비활성 스레드들은 활성으로 됨
상태 2	상태 2	JfL	타겟으로 점프
		JfG	MINRC 로 점프
		그 외 이벤트	PC+1

도면22

각각의 명령을 실행하기 전의 재개 검사의 거동:

```
{
   for each inactive thread {
      if(RC == PC) {
         active flag = TRUE;
         RC = MAX;
      }
   }
   updateMINRC();
}
```

도면23

점프 명령의 거동:

```
{
   if(backward jump) {// Jb
      if(any thread is active) // 상태 0 또는 상태 1
         jump to target; //전체 범위 점프
      else // 상태 2
         PC++;
   }
   else { //순방향 점프
      if(target <= MINRC) // JfL , 상태 0, 1, 또는 2
         jump to target; // 전체 범위 점프
      else { // JfG, 상태 1 또는 2
         for each active thread {
            active flag = FALSE;
            RC = target;
         }
         Jump to MINRC;
      }
   }
}
```

도면24

```
분기 명령의 거동 :
{
  if(backward branching) {
    if(any thread is active) { // 상태 0 또는 상태 1
      if(all active threads are uniform) {
        if(condition satisfied) // BbuT
          jump to target;
        else // BbuF
          PC++;
      } else { // Bbd
        for each active thread {
          if(condition not satisfied) {
            active flag = FALSE;
            RC = PC + 1;
          }
        }
        updateMINRC(); // 이 경우에 MINRC = PC + 1
        jump to target;
      }
    } else // 상태 2
      PC++;
  } else { // 순방향 분기
    if(all active threads are uniform) {
      if(condition satisfied) { // 상태 0 또는 상태 1
        if(target <= MINRC) // BfuTL
          jump to target; // 전체 범위 점프
        else { // BfuTG, 오직 상태 1
          for each active thread {
            active flag = FLASE;
            RC = target;
          }
          jump to MINRC; // 부분적인 범위 점프
        }
      } else // BfuF, 상태 0, 1, 또는 2
        PC++; // 점프 없음
    } else { // Bfd, 상태 0 또는 상태 1
      for each active thread with condition satisfied {
        active flag = FALSE;
        RC = target;
      }
      updateMINRC();
      PC++; // 점프 없음
    }
  }
}
```

도면25

```
호출 명령의 거동:
{
    MINRC_stack.push(MINRC);          // 호출자의 MINRC 를 저장
    MINRC = MAX;                      // 피호출자의 MINRC 를 초기화
    call_stack.push(return address);  // 복귀 어드레스를 저장
    PC = target;                      // 서브루틴 진입 포인트로 점프
}
```

도면26

```
복귀 명령의 거동:
{
    if(call_stack is not empty)
    {
        MINRC = MINRC_stack.pop();   // 호출자의 MINRC 를 복원
        PC = call_stack.pop();       // 복귀 어드레스를 얻고 그 어드레스로 점프
    }
    else
        terminates;
}
```

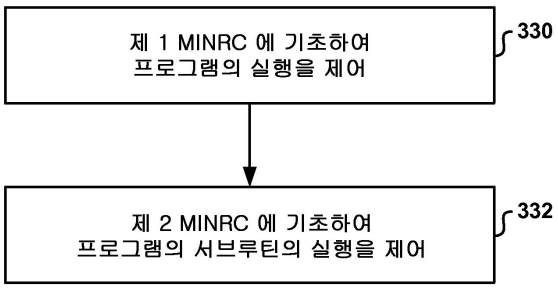
도면27

```
updateMINRC()
{
  for each inactive thread {
    if(RC < MINRC and RC >= entry point)
      MINRC = RC;
  }
}
```

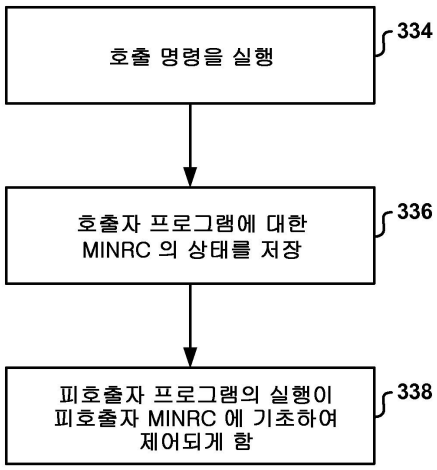
도면28

```
updateMINRC()
{
  for each inactive thread {
    if(RC < MINRC and originalActiveFlag == ACTIVE)
      MINRC = RC;
  }
}
```

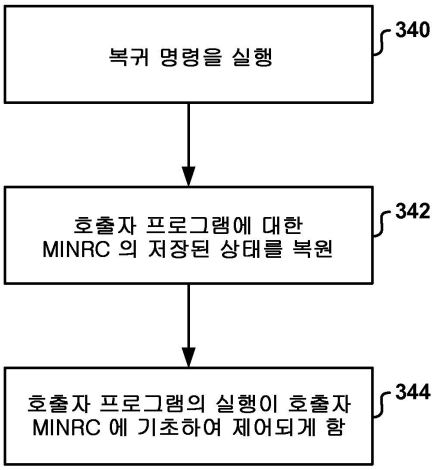
도면29



도면30


호출 명령을 실행
334
호출자 프로그램에 대한
MINRC 의 상태를 저장
336
피호출자 프로그램의 실행이
피호출자 MINRC 에 기초하여
제어되게 함
338

도면31


복귀 명령을 실행
340
호출자 프로그램에 대한
MINRC 의 저장된 상태를 복원
342
호출자 프로그램의 실행이 호출자
MINRC 에 기초하여 제어되게 함
344