(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2009년12월29일
(11) 등록번호 10-0934533
(24) 등록일자 2009년12월21일

(51) Int. Cl.
G06F 9/48 (2006.01) *G06F 9/50* (2006.01)
(21) 출원번호 10-2004-7001185
(22) 출원일자 2003년04월17일
심사청구일자 2008년04월14일
(85) 번역문제출일자 2004년01월27일
(65) 공개번호 10-2004-0105685
(43) 공개일자 2004년12월16일
(86) 국제출원번호 PCT/JP2003/004886
(87) 국제공개번호 WO 2003/100613
국제공개일자 2003년12월04일
(30) 우선권주장
JP-P-2002-00154313 2002년05월28일 일본(JP)
(56) 선행기술조사문헌
JP08095807 A*
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
소니 가부시끼 가이샤
일본국 도쿄도 미나토쿠 코난 1-7-1
(72) 발명자
도가와아쯔시
일본 141-0001 도쿄도 시나가와꾸 기따시나가와 6쪼메 7-35 소니 가부시끼가이샤 내
(74) 대리인
구영창, 이중희, 장수길

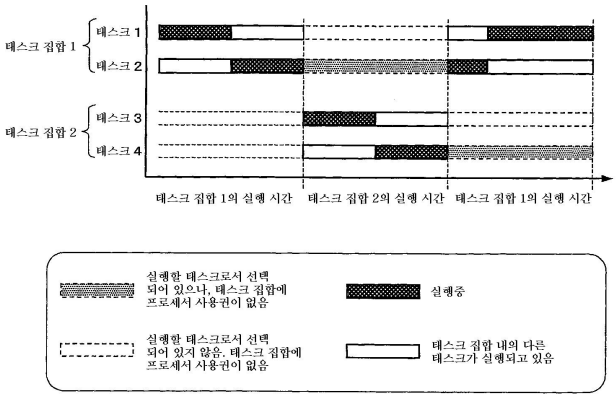
전체 청구항 수 : 총 9 항 심사관 : 황승희

(54) 연산 처리 시스템, 컴퓨터 시스템 상에서의 태스크 제어 방법, 및 컴퓨터 프로그램을 기록한 컴퓨터 판독 가능한 기록 매체

(57) 요 약

긴급성이 높은 처리가 개시되는 타이밍을 기록하는 기구를 구비하여, 긴급성이 낮은 처리 도중에 크리티컬 섹션에 침입할 때에는 기록을 참조하여, 크리티컬 섹션 실행 중에 긴급성이 높은 처리가 개시되지 않는지의 여부를 검사한다. 개시되지 않을 때는 크리티컬 섹션으로 침입하고, 개시되는 경우에는 긴급성이 높은 처리가 완료될 때까지 크리티컬 섹션에의 침입이 연기되도록 제어한다. 복수의 태스크 실행 환경이 존재하는 하에서 크리티컬 섹션에서의 배타 제어를 적합하게 행할 수 있다.

대 표 도 - 도2



특허청구의 범위

청구항 1

상이한 방침에 따라서 스케쥴링이 행해지는 복수의 태스크 실행 환경이 존재하는 연산 처리 시스템으로서,

태스크의 실행 환경을 전환하는 실행 환경 전환 수단과,

다음에 태스크의 실행 환경의 전환을 행하는 예정 시각을 관리하는 예정 시각 관리 수단과,

상기 예정 시각에 따라서 현행의 태스크 실행 환경하에서의 태스크의 실행을 관리하는 태스크 실행 관리 수단

을 포함하는 것을 특징으로 하는 연산 처리 시스템.

청구항 2

제1항에 있어서,

상기 태스크 실행 환경 관리 수단은, 현행의 태스크 실행 환경하에서 실행 중인 태스크가 복수의 태스크로부터 동시에 참조할 수 없는 크리티컬 섹션에 진입할 때에, 상기 예정 시각에 대하여 상기 크리티컬 섹션의 실행 시간에 여유가 있는지의 여부에 따라서 상기 크리티컬 섹션에 진입할지의 여부를 판단하는 것을 특징으로 하는 연산 처리 시스템.

청구항 3

제2항에 있어서,

상기 태스크 실행 환경 관리 수단은, 상기 예정 시각에 대하여 상기 크리티컬 섹션의 실행 시간에 여유가 있는 경우에는 상기 크리티컬 섹션에의 진입을 허용하지만, 여유가 없는 경우에는 상기 실행 환경 전환 수단에 대하여 태스크 실행 환경의 전환을 지시하는 것을 특징으로 하는 연산 처리 시스템.

청구항 4

제1항에 있어서,

상기 실행 환경 전환 수단에 의한 태스크 실행 환경의 전환 시에, 전환 시의 태스크 실행의 상태를 보존하는 콘텍스트 보존 수단을 더 포함하는 것을 특징으로 하는 연산 처리 시스템.

청구항 5

상이한 방침에 따라서 스케쥴링이 행해지는 복수의 태스크 실행 환경이 존재하는 컴퓨터 시스템 상에서의 태스크 제어 방법으로서,

태스크의 실행 환경을 전환하는 실행 환경 전환 단계와,

다음에 태스크의 실행 환경의 전환을 행하는 예정 시각을 관리하는 예정 시각 관리 단계와,

상기 예정 시각에 따라서 현행의 태스크 실행 환경하에서의 태스크의 실행을 관리하는 태스크 실행 관리 단계

를 포함하는 것을 특징으로 하는 컴퓨터 시스템 상에서의 태스크 제어 방법.

청구항 6

제5항에 있어서,

상기 태스크 실행 환경 관리 단계에서는, 현행의 태스크 실행 환경하에서 실행 중인 태스크가 복수의 태스크로부터 동시에 참조할 수 없는 크리티컬 섹션에 진입할 때에, 상기 예정 시각에 대하여 상기 크리티컬 섹션의 실행 시간에 여유가 있는지의 여부에 따라서 상기 크리티컬 섹션에 진입할지의 여부를 판단하는 것을 특징으로 하는 컴퓨터 시스템 상에서의 태스크 제어 방법.

청구항 7

제6항에 있어서,

상기 태스크 실행 환경 관리 단계에서는, 상기 예정 시각에 대하여 상기 크리티컬 섹션의 실행 시간에 여유가 있는 경우에는 상기 크리티컬 섹션에의 진입을 허용하지만, 여유가 없는 경우에는 상기 실행 환경 전환 수단에 대하여 태스크 실행 환경의 전환을 지시하는 것을 특징으로 하는 컴퓨터 시스템 상에서의 태스크 제어 방법.

청구항 8

제5항에 있어서,

상기 실행 환경 전환 단계에 의한 태스크 실행 환경의 전환 시에, 전환 시의 태스크 실행의 상태를 보존하는 콘텍스트 보존 단계를 더 포함하는 것을 특징으로 하는 컴퓨터 시스템 상에서의 태스크 제어 방법.

청구항 9

상이한 방침에 따라서 스케줄링이 행해지는 복수의 태스크 실행 환경이 존재하는 컴퓨터 시스템 상에서의 태스크 제어의 수순이 컴퓨터 판독 가능한 형식으로 기술된 컴퓨터 프로그램을 기록한 컴퓨터 판독 가능한 기록 매체로서,

태스크의 실행 환경을 전환하는 실행 환경 전환 단계와,

다음에 태스크의 실행 환경의 전환을 행하는 예정 시각을 관리하는 예정 시각 관리 단계와,

상기 예정 시각에 따라서 현행의 태스크 실행 환경하에서의 태스크의 실행을 관리하는 태스크 실행 관리 단계

를 포함하는 것을 특징으로 하는 컴퓨터 프로그램을 기록한 컴퓨터 판독 가능한 기록 매체.

명 세 서

기 술 분 야

<1> 본 발명은, 프로그램을 실행하는 것에 의해 소정의 처리 서비스를 제공하는 연산 처리 시스템, 컴퓨터 시스템 상에서의 태스크 제어 방법, 및 컴퓨터 프로그램에 관한 것으로, 특히 프로그램 내에 복수의 제어의 흐름(예를 들면, 인터럽트 처리 프로그램과 통상 처리 프로그램, 또는 복수의 태스크 등)이 존재하는 타입의 연산 처리 시스템, 컴퓨터 시스템 상에서의 태스크 제어 방법, 및 컴퓨터 프로그램에 관한 것이다.

<2> 더 상세하게는, 본 발명은, 복수의 태스크로부터 동시에 참조할 수 없는 프로그램 부분(크리티컬 섹션)에 있어서 배타(排他) 제어를 행하는 연산 처리 시스템, 컴퓨터 시스템 상에서의 태스크 제어 방법, 및 컴퓨터 프로그램에 관한 것으로, 특히 복수의 태스크 실행 환경이 존재하는 하에서 프로그램 내의 크리티컬 섹션에서의 배타 제어를 행하는 연산 처리 시스템, 컴퓨터 시스템 상에서의 태스크 제어 방법, 및 컴퓨터 프로그램에 관한 것이다.

배 경 기 술

<3> 현재의 LSI(Large Scale Integration) 기술의 혁신적인 진보와 함께, 여러가지 타입의 정보 처리 기기나 정보 통신 기기가 개발, 시판되어, 일상 생활에 깊게 침투하기에 이르렀다. 이러한 종류의 기기에서는, 오퍼레이팅 시스템이 제공하는 실행 환경하에서, CPU(Central Processing Unit)나 그 밖의 프로세서가 소정의 프로그램 코드를 실행하여 여러 가지 처리 서비스를 제공하도록 되어 있다.

<4> 그런데, 프로그램 설계에 있어서, 프로그램 내에 제어의 흐름(「태스크」라고도 부름)이 복수 존재하는 것이 유용한 경우가 있다. 여기서, 복수의 제어의 흐름이란, 도 6에 도시한 바와 같이, 프로그램의 처리의 흐름, 즉 흐름도에서 「현재 실행 중인 지점」이 다수 있는 것을 의미한다. 도 6에 도시하는 예에서는, 임의의 시점 T1에서는, 흐름 I에서 단계 S1이, 흐름 Ⅱ에서 단계 S3이 각각 실행된다. 그리고, 이에 대하여, 시간이 경과하여, 다음의 시점 T1에서는, 흐름 I에서 단계 S2가, 흐름 Ⅱ에서 단계 S3이 각각 실행된다.

<5> 일반적으로, 복수의 흐름이 존재하고, 각각이 공통의 데이터를 조작하는 경우, 이들 사이에서 동기를 취하지 않으면 데이터의 일관성을 유지할 수 없다. 여기서 말하는 공통의 데이터로는 태스크 일람이나, 조건 변수 등을 들 수 있다. 조건 변수는 태스크가 기다리고 있는 조건을 추상화한 개념으로, 태스크가 언제 대기 상태로 이행할지, 또는 언제 실행 가능 상태로 복귀할지를 오퍼레이팅 시스템에 전달하는 수단의 하나로서 이용된다.

<6> 예를 들면, 2개의 제어의 흐름 B 및 C이 존재하는 경우에, 각각의 제어의 흐름이 이하의 처리를 행하는 경우에

대하여 고찰하여 본다.

<7> 수순1 : 변수 x의 값을 판독한다.

<8> 수순2 : 판독한 값에 1를 더한 값을 변수 x에 대입한다.

<9> 2개의 흐름 B 및 C가 각각 상기의 처리를 1번 행하면, 동일한 처리가 2번 행해지는 것이 된다. 따라서, 변수 x의 값은 2 증가할 것이다. 그런데, 흐름 B와 흐름 C가 이하와 같이 중첩되는 경우, 변수 x의 값은 1만 증가한다.

<10> ① 흐름 B가 수순 1을 실행한다.

<11> ② 흐름 C가 수순 1을 실행한다.

<12> ③ 흐름 C가 수순 2를 실행한다.

<13> ④ 흐름 B가 수순 2를 실행한다.

<14> 이러한 동작의 오류를 방지하기 위해서는, 임의의 흐름에서 행해진 일련의 참조, 갱신 조작(트랜잭션) 사이에, 다른 흐름으로부터 데이터가 참조, 갱신되는 것을 금할 필요가 있다.

<15> 상술한 예의 경우, 흐름 B에서 수순 1과 수순 2라는 일련의 조작이 완전하게 완료되기 전에 흐름 C가 변수 x를 참조, 갱신하였기 때문에 데이터의 일관성을 잃어버리는 문제가 발생하였다.

<16> 상술한 수순 1~2와 같은 조작은, 바꿔 말하면 복수의 태스크로부터 동시에 참조할 수 없는 프로그램 부분으로, 이하에서는「크리티컬 섹션」이라고도 부른다. 또한, 크리티컬 섹션에서, 데이터의 일관성의 문제를 해결하기 위해서 다른 태스크에 의한 데이터의 참조나 갱신을 금하는 것을 「배타 제어」라고도 부른다. 즉, 임의의 제어의 흐름에 있어서 임의의 데이터에 대하여 일련의 처리가 행해지고 있는 동안 다른 제어의 흐름이 동일한 데이터에 조작을 행하는 것을 지연시키는, 즉 특정 데이터에 대한 조작을 배타적으로 행한다.

<17> 본 발명자는 배타 제어 기구가 이하의 특징을 구비하여야 하는 것으로 생각한다.

<18> (1) 긴급도가 높은 처리가 긴급도가 낮은 처리에 의해서 연기될 가능성(우선도(優先度) 역전 현상의 발생)이 없다.

<19> (2) 다른 방침에 따라서 스케쥴링이 행해지는 복수의 태스크 집합 사이에서도 배타 제어를 행할 수 있다.

<20> 상기 특징 중에서 (1)이 필요한 이유는 당업자에게는 분명하다. 또한, (2)는, 1대의 컴퓨터 시스템 상에서 복수의 오퍼레이팅 시스템을 동시에 가동시키는 경우나, 각각 특성이 다른 복수의 태스크 집합마다 다른 스케쥴링 방법을 이용하는 경우에 필요하게 된다.

<21> 예를 들면, 태스크 사이의 배타 제어를 위해 「mutex 기구」나 「세마포어 기구」가 사용되고 있다. 그런데, 이러한 배타 제어를 이용하는 방법에는, 고 우선도의 처리가 저 우선도의 처리에 의해서 연기될 가능성, 즉 우선도 역전 현상이 발생할 가능성이 있는 문제가 있기 때문에, 상기의 특징 (1)을 만족하지 않는다.

<22> 이러한 우선도 역전의 문제를 완화하기 위한 방법으로서, 우선도 계승 프로토콜(예를 들면, LuiSha, Ragunathan Rajkumar, 및 John P. Lehoczky 공저의 논문 "Priority Inheritance Protocols : An Approach to Real-Time Synchronization", IEEE Transactions on Computers) Vol.39, No.9, pp.1175-1185, September 1990 참조) 등이 제안되어 있다. 우선도 계승 프로토콜이란, 저 우선도 태스크가 일련의 조작을 실행하는 도중에 고 우선도 태스크가 동일 데이터를 조작하려고 하는 경우에는, 저 우선도 태스크의 조작이 완료될 때까지 저 우선도 태스크의 우선도를 일시적으로 고 우선도 태스크와 동일한 우선도로 상승시키는 방법이다.

<23> 도 7에는 우선도 계승 프로토콜의 동작을 도시하고 있다. 이 경우, 우선도가 낮은 태스크 A가 임의의 데이터를 조작하는 도중에 우선도가 높은 태스크 B가 동일한 데이터의 조작을 개시하려고 하여도 부득이하게 지연이 된다. 이 때, 태스크 A의 우선도를 일시적으로 태스크 B와 동일한 레벨까지 상승시킨다. 그 후, 태스크 B보다도 낮지만 태스크 A보다는 높은 우선도를 갖는 태스크 C의 실행이 개시되어도, 태스크 A의 우선도는 태스크 C보다도 상승되어 있기 때문에, 태스크 A의 실행이 중단하는 경우는 없다. 그리고, 태스크 A의 종료 후에, 태스크 B는 데이터의 일관성을 유지하면서 자신보다도 우선도가 낮은 태스크 C에 인터럽트되지 않고 데이터의 조작을 개시할 수 있다.

<24> 그런데, 이 우선도 계승 프로토콜은 우선도라는 공통의 척도에 따라서 모든 태스크의 스케쥴링이 행해지는 것을

전제로 하고 있다. 이 때문에, 예를 들면 단일의 컴퓨터 시스템 상에서 복수의 오퍼레이팅 시스템이 동시에 동작하는 태스크 실행 환경과 같이, 복수의 스케쥴링이 공존하는 시스템(특히, 우선도에 따른 스케쥴링을 행하지 않는 태스크 집합이 존재하고 있는 시스템)에 적용하는 것은 곤란하다. 즉, 상기의 특징 (2)를 만족하지 않는다.

<25> 이러한 문제를 갖지 않는 방법으로서, 스케쥴러 칸셔스 싱크로나이제이션 방법(예를 들면, Leonidas I. Kontothanassis, Robert W. Wisniewski, Michael L. Scott 공저의 논문 "Scheduler-conscious synchronization"(ACM Transactions on Computer Systems, Volume 15, Issue 1, 1997) 참조)을 들 수 있다. 이 방법은, 크리티컬 섹션 실행 중에 다른 태스크가 디스패치되는 것을 금지함으로써, 긴급성이 높은 처리에 대하여 긴급성이 낮은 처리가 미치는 영향을 한정하고 있다. 구체적으로는, 긴급성이 높은 처리의 지연 시간을 최대 크리티컬 섹션 실행 시간 이하로 억제하고 있다. 또한, 이 방법은 디스패치를 금지하는 기구가 갖추어져 있는 것 만을 전제로 하고 있다.

<26> 그러나, 이 방법도 복수의 스케쥴링이 공존하는 시스템에의 적용을 고려한 것이 아니기 때문에, 이러한 태스크 실행 환경하에서는, 긴급성이 높은 처리가 긴급성이 낮은 처리에 의해서 지연될 가능성은 여전히 남아 있다. 즉, 상기 특징 (2)를 만족하는 것은 아니다.

<27> 또한, 상기 특징 (1) 및 (2)를 전부 만족할 수 있는 방법으로서, 논블로킹 싱크로나이제이션 방법(예를 들면, Michael Barry Greenwald의 논문 Non-blocking Synchronization and System Design"(Ph.D. Thesis, Stanford University, 1999) 참조)을 들 수 있다. 그러나, 이것을 적용하기 위해서는 특별한 하드웨어가 필요하여, 비용 증대를 초래한다.

<28> <발명의 개시>

<29> 본 발명의 목적은, 프로그램 내에 복수의 제어의 흐름(예를 들면, 인터럽트 처리 프로그램과 통상 처리 프로그램, 또는 복수의 태스크 등)이 존재하는 타입이 우수한 연산 처리 시스템, 컴퓨터 시스템 상에서의 태스크 제어 방법, 및 컴퓨터 프로그램을 제공하는 것에 있다.

<30> 본 발명의 다른 목적은, 복수의 태스크로부터 동시에 참조할 수 없는 프로그램 부분(크리티컬 섹션)에 있어서 배타 제어를 적합하게 행할 수 있는, 우수한 연산 처리 시스템, 컴퓨터 시스템 상에서의 태스크 제어 방법, 및 컴퓨터 프로그램을 제공하는 것에 있다.

<31> 본 발명의 또 다른 목적은, 스케쥴링 방침이 다른 복수의 태스크 실행 환경이 존재하는 하에서 크리티컬 섹션에서의 배타 제어를 적합하게 행할 수 있는, 우수한 연산 처리 시스템, 컴퓨터 시스템 상에서의 태스크 제어 방법, 및 컴퓨터 프로그램을 제공하는 것에 있다.

<32> 본 발명의 또 다른 목적은, 특별한 하드웨어가 필요없이 복수의 스케쥴링 방침이 공존하는 시스템에 있어서 배타 제어를 적합하게 행할 수 있는, 우수한 연산 처리 시스템, 컴퓨터 시스템 상에서의 태스크 제어 방법, 및 컴퓨터 프로그램을 제공하는 것에 있다.

<33> 본 발명은, 상기 과제를 고려하여 이루어진 것으로, 그 제1 양태는, 복수의 처리 사이에서 배타 제어를 행하는 연산 처리 시스템으로서, 긴급성이 높은 처리가 개시되는 타이밍을 기록하는 개시 타이밍 기록 수단과, 긴급성이 낮은 처리 도중에 복수의 처리로부터 동시에 참조할 수 없는 크리티컬 섹션에 진입할 때에, 상기 개시 타이밍을 참조하여, 그 크리티컬 섹션 실행 중에 긴급성이 높은 처리가 개시되지 않는지의 여부를 검사하는 검사 수단과, 그 검사의 결과에 대응하여, 크리티컬 섹션의 실행을 제어하는 제어 수단을 구비하는 것을 특징으로 하는 연산 처리 시스템이다.

<34> 단, 여기서 말하는 「시스템」이란, 복수의 장치(또는 특정 기능을 실현하는 기능 모듈)가 논리적으로 집합된 것을 말하고, 각 장치나 기능 모듈이 단일 케이스 내에 있는지의 여부는 특별히 묻지 않는다(이하 동일).

<35> 상기 제어 수단은, 크리티컬 섹션 실행 시에 데이터의 일관성을 유지하기 위해서, 그 크리티컬 섹션 실행 중에 긴급성이 높은 처리가 개시되지 않는 경우에는 그 크리티컬 섹션에의 진입을 허용하고, 개시되는 경우에는 그 크리티컬 섹션에의 진입이 연기되도록 하여, 배타 제어를 행한다.

<36> 따라서, 본 발명의 제1 양태에 따른 연산 처리 시스템에 따르면, 긴급성이 높은 처리의 개시가 지연되지 않고서 복수의 처리 사이에서 배타 제어를 적합하게 행할 수 있다.

<37> 또한, 본 발명의 제1 양태에 따른 연산 처리 시스템에 따르면, 우선도 계승을 행하지 않는 mutex나 세마포에 비

하여, 콘텍스트 전환의 횟수를 삭감하는 것에 의해, 오버헤드를 삭감할 수 있다.

<38> 또한, 본 발명의 제1 양태에 따른 연산 처리 시스템에 따르면, 서로 다른 방침에 따라서 스케쥴링이 행해지는 복수의 태스크 집합 사이에서도 배타 제어를 적합하게 행할 수 있다.

<39> 또한, 본 발명의 제1 양태에 따른 연산 처리 시스템에 따르면, 복수의 오퍼레이팅 시스템이 동시에 동작하는 시스템에 있어서, 이들 오퍼레이팅 시스템 상에서 동작하고 있는 태스크 사이의 배타 제어나, 오퍼레이팅 시스템 사이의 배타 제어가 가능하게 된다.

<40> 또한, 본 발명의 제1 양태에 따른 연산 처리 시스템에 따르면, 특별한 하드웨어를 필요로 하지 않고서 복수의 스케쥴링 방침이 공존하는 시스템에 있어서 배타 제어를 적합하게 행할 수 있다.

<41> 또한, 본 발명의 제2 양태는, 서로 다른 방침에 따라서 스케쥴링이 행해지는 복수의 태스크 실행 환경이 존재하는 연산 처리 시스템 또는 컴퓨터 시스템 상에서의 태스크 제어 방법으로서, 태스크의 실행 환경을 전환하는 실행 환경 전환 수단 또는 단계와, 다음에 태스크의 실행 환경의 전환을 행하는 예정 시각을 관리하는 예정 시각 관리 수단 또는 단계와, 상기 예정 시각에 따라서 현행의 태스크 실행 환경하에서의 태스크의 실행을 관리하는 태스크 실행 관리 수단 또는 단계를 포함하는 것을 특징으로 하는 연산 처리 시스템 또는 컴퓨터 시스템 상에서의 태스크 제어 방법이다.

<42> 여기서, 상기 태스크 실행 환경 관리 수단 또는 단계는, 현행의 태스크 실행 환경하에서 실행 중인 태스크가 복수의 태스크로부터 동시에 참조할 수 없는 크리티컬 섹션에 진입할 때에, 상기 예정 시각에 대하여 그 크리티컬 섹션의 실행 시간에 여유가 있는지의 여부에 따라서 그 크리티컬 섹션에 진입할지의 여부를 판단하는 것에 의해, 태스크의 배타 제어를 행하면 바람직하다.

<43> 보다 구체적으로는, 상기 태스크 실행 환경 관리 수단 또는 단계는, 상기 예정 시각에 대하여 그 크리티컬 섹션의 실행 시간에 여유가 있는 경우에는 그 크리티컬 섹션에의 진입을 허용하지만, 여유가 없는 경우에는 상기 실행 환경 전환 수단에 대하여 태스크 실행 환경의 전환을 지시하면 바람직하다.

<44> 여기서, 상기 실행 환경 전환 수단 또는 단계에 의한 태스크 실행 환경의 전환에 있어서, 전환 시에 태스크 실행의 상태를 보존하는 콘텍스트 보존 수단을 더 구비하여도 된다.

<45> 본 발명의 제2 양태에 따른 연산 처리 시스템 또는 컴퓨터 시스템 상에서의 태스크 제어 방법에 따르면, 서로 다른 방침에 따라서 스케쥴링이 행해지는 복수의 태스크 집합 사이에서도 배타 제어를 적합하게 행할 수 있다.

<46> 또한, 본 발명의 제2 양태에 따른 연산 처리 시스템 또는 컴퓨터 시스템 상에서의 태스크 제어 방법에 따르면, 복수의 오퍼레이팅 시스템이 동시에 동작하는 시스템에 있어서, 이들 오퍼레이팅 시스템 상에서 동작하고 있는 태스크 사이의 배타 제어나, 오퍼레이팅 시스템 사이의 배타 제어가 가능하게 된다.

<47> 또한, 본 발명의 제2 양태에 따른 연산 처리 시스템 또는 컴퓨터 시스템 상에서의 태스크 제어 방법에 따르면, 우선도 계승을 행하지 않는 mutex나 세마포에 비하여, 콘텍스트 전환의 횟수를 삭감하는 것에 의해, 오버헤드를 삭감할 수 있다.

<48> 또한, 본 발명의 제2 양태에 따른 연산 처리 시스템 또는 컴퓨터 시스템 상에서의 태스크 제어 방법에 따르면, 특별한 하드웨어가 필요없이 복수의 스케쥴링 방침이 공존하는 시스템에 있어서 배타 제어를 적합하게 행할 수 있다.

<49> 또한, 본 발명의 제3 양태는, 복수의 처리 사이에서 배타 제어를 행하기 위한 처리를 컴퓨터 시스템 상에서 실행하도록 컴퓨터 판독 가능한 형식으로 기술된 컴퓨터 프로그램으로서, 긴급성이 높은 처리가 개시되는 타이밍을 기록하는 개시 타이밍 기록 단계와, 긴급성이 낮은 처리 도중에 복수의 처리로부터 동시에 참조할 수 없는 크리티컬 섹션에 진입할 때, 상기한 개시 타이밍을 참조하여, 그 크리티컬 섹션 실행 중에 긴급성이 높은 처리가 개시되지 않는지의 여부를 검사하는 검사 단계와, 그 검사의 결과에 대응하여, 크리티컬 섹션의 실행을 제어하는 제어 단계를 구비하는 것을 특징으로 하는 컴퓨터 프로그램이다.

<50> 또한, 본 발명의 제4 양태는, 서로 다른 방침에 따라서 스케쥴링이 행해지는 복수의 태스크 실행 환경이 존재하는 컴퓨터 시스템 상에서의 태스크 제어의 수순이 컴퓨터 판독 가능한 형식으로 기술된 컴퓨터 프로그램으로서, 태스크의 실행 환경을 전환하는 실행 환경 전환 단계와, 다음에 태스크의 실행 환경의 전환을 행하는 예정 시각을 관리하는 예정 시각 관리 단계와, 상기 예정 시각에 따라서 현행의 태스크 실행 환경하에서의 태스크의 실행을 관리하는 태스크 실행 관리 단계를 포함하는 것을 특징으로 하는 컴퓨터 프로그램이다.

<51> 본 발명의 제3 및 제4 양태의 각각에 따른 컴퓨터 프로그램은, 컴퓨터 시스템 상에서 소정의 처리를 실현하도록 컴퓨터 판독 가능한 형식으로 기술된 컴퓨터 프로그램을 정의한 것이다. 바꾸어 말하면, 본 발명의 제3 및 제4 양태의 각각에 따른 컴퓨터 프로그램을 컴퓨터 시스템에 인스톨함으로써, 컴퓨터 시스템 상에서는 협동적 작용이 발휘되어, 본 발명의 제1 및 제2 양태에 따른 연산 처리 시스템 또는 컴퓨터 시스템 상에서의 태스크 제어 방법과 마찬가지의 작용 효과를 얻을 수 있다.

<52> 본 발명의 또 다른 목적, 특징이나 이점은, 후술하는 본 발명의 실시 형태나 첨부하는 도면에 기초한 상세한 설명에 의해서 분명하게 될 것이다.

산업상 이용 가능성

<111> 본 발명에 따르면, 프로그램 내에 복수의 제어의 흐름(예를 들면, 인터럽트 처리 프로그램과 통상 처리 프로그램, 또는 복수의 태스크 등)이 존재하는 타입이 우수한 연산 처리 시스템, 컴퓨터 시스템 상에서의 태스크 제어 방법, 및 컴퓨터 프로그램을 제공할 수 있다.

<112> 또한, 본 발명에 따르면, 복수의 태스크로부터 동시에 참조할 수 없는 프로그램 부분(크리티컬 섹션)에서 배타 제어를 적합하게 행할 수 있는, 우수한 연산 처리 시스템, 컴퓨터 시스템 상에서의 태스크 제어 방법, 및 컴퓨터 프로그램을 제공할 수 있다.

<113> 또한, 본 발명에 따르면, 스케쥴링 방침이 다른 복수의 태스크 실행 환경이 존재하는 하에서 크리티컬 섹션에서의 배타 제어를 적합하게 행할 수 있는, 우수한 연산 처리 시스템, 컴퓨터 시스템 상에서의 태스크 제어 방법, 및 컴퓨터 프로그램을 제공할 수 있다.

<114> 또한, 본 발명에 따르면, 특별한 하드웨어가 필요없이 복수의 스케쥴링 방침이 공존하는 시스템에 있어서 배타 제어를 적합하게 행할 수 있는, 우수한 연산 처리 시스템, 컴퓨터 시스템 상에서의 태스크 제어 방법, 및 컴퓨터 프로그램을 제공할 수 있다.

<115> 본 발명에 따르면, 긴급성이 높은 처리의 개시가 지연되지 않고, 복수의 처리 사이에서 배타 제어를 적합하게 행할 수 있다.

<116> 또한, 본 발명에 따르면, 우선도 계승을 행하지 않는 mutex나 세마포에 비하여, 콘텍스트 전환의 횟수를 삭감하는 것에 의해, 오버헤드를 삭감할 수 있다.

<117> 또한, 본 발명에 따르면, 서로 다른 방침에 따라서 스케쥴링이 행해지는 복수의 태스크 집합의 사이에서도 배타 제어를 적합하게 행할 수 있다.

<118> 또한, 본 발명에 따르면, 복수의 오퍼레이팅 시스템이 동시에 동작하는 시스템에 있어서, 이들 오퍼레이팅 시스템 상에서 동작하고 있는 태스크 사이의 배타 제어나, 오퍼레이팅 시스템 사이의 배타 제어가 가능하게 된다.

<119> 본 발명은, 논블로킹 싱크로나이제이션과는 대조적으로, 특별한 하드웨어를 사용하지 않고, 상기 효과를 얻을 수 있다.

도면의 간단한 설명

<53> 도 1은 본 발명의 실시에 제공되는 연산 처리 시스템(10)의 하드웨어 구성을 모식적으로 도시하는 도면.

<54> 도 2는 복수의 오퍼레이팅 시스템이 동시에 가동하고 있는 모습을 모식적으로 도시하는 도면.

<55> 도 3은 본 발명의 실시 형태에 따른 태스크 실행 환경의 구성을 모식적으로 도시하는 도면.

<56> 도 4는 태스크 집합 전환 소프트웨어가 태스크 집합을 전환하기 위한 처리 수순을 도시하는 흐름도.

<57> 도 5는 태스크 집합에 속하는 태스크가 크리티컬 섹션에 침입할 때에 행하는 처리 수속을 도시한 흐름도.

<58> 도 6은 재실행 중인 지점이 복수개 있는 프로그램의 처리의 흐름, 즉 흐름도를 나타낸 도면.

<59> 도 7은 우선도 계승 프로토콜의 동작을 설명하기 위한 도면.

<60> <발명을 실시하기 위한 최량의 형태>

<61> 이하, 도면을 참조하면서 본 발명의 실시 형태에 대하여 상세히 설명한다.

<62> 도 1은 본 발명의 실시에 제공되는 연산 처리 시스템(10)의 하드웨어 구성을 모식적으로 도시하고 있다. 도 1에 도시한 바와 같이, 연산 처리 시스템(10)은 프로세서(11)와, RAM(Random Access Memory)(12)과, ROM(Read Only Memory)(13)과, 복수의 입출력 장치(14-1, 14-2, …)와, 타이머(15)를 포함하고 있다.

<63> 프로세서(11)는 연산 처리 시스템(10)의 메인 컨트롤러이고, 오퍼레이팅 시스템(OS)의 제어하에서 각종 프로그램 코드를 실행하도록 되어 있다.

<64> 오퍼레이팅 시스템이 프로그램 실행을 관리 제어하는 단위를 일반적으로 「태스크」라고 부른다. 본 실시 형태에 따른 연산 처리 시스템(10)에서는, 프로그램 내에 복수의 태스크가 존재하는 것을 허용한다. 따라서, 실제로 계산을 진행시키는 실체인 프로세서(11)의 개수보다도 많은 태스크가 존재하게 된다.

<65> 오퍼레이팅 시스템은 프로세서(11)에 의해 처리되는 태스크를 빈번히 전환하는 것에 의해, 각 태스크를 의사적으로 병렬로 실행시키도록 되어 있다. 각 태스크에는 다른 태스크와 식별 가능한 태스크 ID가 할당되어 있다. 또한, 각 태스크는 스택 상의 데이터 영역을 이용하여 데이터에 대한 일련의 조작, 즉 트랜잭션을 행한다.

<66> 또한, 본 실시 형태에서는, 복수의 오퍼레이팅 시스템을 동시에 가동시킬 수 있다. 이들 오퍼레이팅 시스템은, 각각 특성이 다른 태스크 집합을 구성함과 함께, 서로 다른 방침에 따라서 태스크의 스케쥴링이 행해진다. 본 실시 형태에 따른 연산 처리 시스템(10)에서의 태스크 실행 환경의 상세한 내용에 대해서는 후술한다.

<67> 프로세서(11)는, 버스(16)에 의해서 다른 기기류(후술)와 상호 접속되어 있다. 시스템 버스(16) 상의 각 기기에는 각각 고유의 메모리 어드레스 또는 I/O 어드레스가 부여되어 있고, 프로세서(11)는 이들 어드레스를 지정함으로써 소정의 기기에 액세스 가능하게 된다. 시스템 버스(16)는 어드레스 버스, 데이터 버스, 컨트롤 버스를 포함하는 공통 신호 전송로이다.

<68> RAM(12)은 기입 가능한 메모리이고, 프로세서(11)에서 실행되는 프로그램 코드를 로드하거나, 실행 프로그램의 작업 데이터를 일시 저장하기 위해서 사용된다. 프로그램 코드로는, 예를 들면 BIOS(Basic Input/Output System: 기본 입출력 시스템), 주변 기기를 하드웨어 조작하기 위한 디바이스 드라이버, 오퍼레이팅 시스템, 어플리케이션 등을 들 수 있다.

<69> ROM(13)은 소정의 코드나 데이터를 항구적으로 기억하기 위한 불휘발 메모리이고, 예를 들면 BIOS나 시동 시의 자기 진단 프로그램(Power On Self Test POST) 등을 저장하고 있다.

<70> 입출력 장치(14)는, 디스플레이(21)를 접속하기 위한 디스플레이 인터페이스(14-1), 키보드(22)나 마우스(23)와 같은 사용자 입력 장치를 접속하기 위한 사용자 입력 장치 인터페이스(14-2), 하드 디스크(24)나 미디어 드라이브(25) 등의 외부 기억 장치를 접속하기 위한 외부 기억 장치 인터페이스(14-3), 외부 네트워크와 접속하기 위한 네트워크 인터페이스 카드(NIC)(14-4) 등을 포함한다.

<71> 디스플레이 인터페이스(14-1)는 프로세서(11)가 발행하는 묘화 명령을 실제로 처리하기 위한 전용 인터페이스 컨트롤러이다. 디스플레이 인터페이스(14-1)에서 처리된 묘화 데이터는, 예를 들면 프레임 버퍼(도면 미도시)에 일단 기입된 후, 디스플레이(21)에 의해서 화면 출력된다.

<72> HDD(24)는 기억 저장체로서의 자기 디스크를 고정적으로 탑재한 외부 기억 장치로서(주지), 기억 용량이나 데이터 전송 속도 등에서 다른 외부 기억 장치보다도 우수하다. 통상, HDD(24)에는, 프로세서(11)가 실행하여야 할 오퍼레이팅 시스템의 프로그램 코드나, 어플리케이션 프로그램, 디바이스 드라이버 등이 불휘발적으로 저장되어 있다. 소프트웨어 프로그램을 실행 가능한 상태로 HDD(24)상에 두는 것을 프로그램의 시스템에의 「인스톨」이라고 부른다. 예를 들면, 본 발명을 실현하는 오퍼레이팅 시스템이나, 복수의 태스크가 존재하도록 설계된 어플리케이션 프로그램을 HDD(24)상에 인스톨할 수 있다.

<73> 미디어 드라이브(25)는 CD(Compact Disc)나 MO(Magneto-Optical disc), DVD(Digital Versatile Disc) 등의 가반형 미디어를 장착하고, 그 데이터 기록면에 액세스하기 위한 장치이다.

<74> 가반형 미디어는, 주로 소프트웨어 프로그램이나 데이터 파일 등을 컴퓨터 판독 가능한 형식의 데이터로서 백업하며, 이들을 시스템 사이에서 이동(즉 판매, 유통, 배포를 포함)할 목적으로 사용된다. 예를 들면, 본 발명을 실현하는 오퍼레이팅 시스템이나, 복수의 태스크가 존재하도록 설계된 어플리케이션 프로그램을, 이들 가반형 미디어를 이용하여 복수의 기기 사이에서 물리적으로 유통, 배포할 수 있다.

<75> 네트워크 인터페이스(14-1)는 Ethernet(등록상표) 등의 소정의 통신 프로토콜에 따라서, 시스템(10)을 LAN(Local Area Network) 등의 국소적 네트워크, 나아가서는 인터넷과 같은 광역 네트워크에 접속할 수 있다.

<76> 네트워크 상에서는, 복수의 호스트 단말기(도면 미도시)가 트랜스패어런트인 상태로 접속되어, 분산 컴퓨팅 환경이 구축되어 있다. 네트워크상에서는, 소프트웨어 프로그램이나 데이터, 콘텐츠 등의 배신을 행할 수 있다. 예를 들면, 본 발명을 실현하는 오퍼레이팅 시스템이나, 복수의 태스크가 존재하도록 설계된 어플리케이션 프로그램을 네트워크를 경유하여 다운로드할 수 있다.

<77> 각 입출력 장치(14-1, 14-2, …)에는, 인터럽트 레벨이 할당되고 있고, 소정의 이벤트 발생(예를 들면 키보드 입력이나 마우스 클릭 등의 GUI 처리나, 하드 디스크에서의 데이터 전송의 완료 등)에 응답하여, 인터럽트 요구 신호선(19)을 통하여 프로세서(11)에 통지할 수 있다. 프로세서(11)는 이러한 인터럽트 요구에 응답하여 대응하는 인터럽트 핸들러를 실행한다.

<78> 타이머(15)는 타이머 신호를 소정 주기로 발생하는 장치이다. 타이머(15)에도 인터럽트 레벨이 할당되고 있고, 인터럽트 요구 신호선(19)을 통하여 프로세서(11)에 주기적인 인터럽트를 발생한다.

<79> 또, 도 1에 도시한 바와 같은 연산 처리 시스템(10)의 일례는, 미국 IBM사의 퍼스널 컴퓨터 "PC/AT(Personal Computer/Advanced Technology)"의 호환기 또는 후계기이다. 물론, 다른 아키텍처를 구비한 컴퓨터를 본 실시 형태에 따른 연산 처리 시스템(10)으로서 적용하는 것도 가능하다.

<80> 본 실시 형태에서는 복수의 오퍼레이팅 시스템이 동시에 가동되고 있다. 이들 오퍼레이팅 시스템은, 각각 특성이 다른 태스크 집합을 구성함과 함께, 서로 다른 방침에 따라서 태스크의 스케쥴링이 행해진다.

<81> 도 2는 복수의 오퍼레이팅 시스템이 동시에 가동되고 있는 상태를 모식적으로 도시하고 있다. 도 2에 도시하는 예에서는, 2개의 오퍼레이팅 시스템 OS1 및 OS2가 단일의 연산 처리 시스템(10) 상에서 동시에 가동되고 있다. OS1이 제공하는 태스크 실행 환경하에서는 태스크 1 및 태스크 2가 실행된다. 또한, OS2가 제공하는 태스크 실행 환경하에서는 태스크 3 및 태스크 4가 실행된다. OS1 상에서 실행하는 태스크 1 및 태스크 2는 태스크 집합 1을 구성하고, OS2 상에서 실행하는 태스크 3 및 태스크 4는 태스크 집합 2를 구성한다.

<82> 연산 처리 시스템(10) 상에서는, 예를 들면 콘텍스트 전환 등에 의해서, OS1과 OS2에 대하여 교대로 시스템 사용권이 부여된다. OS1에 콘텍스트가 전환된 기간은 태스크 집합 1의 실행 기간에 상당한다. 그리고, 태스크 집합 1의 실행 기간 내에서는 OS1이 갖는 스케쥴링에 따라서 태스크 집합 1, 즉 태스크 1 및 태스크 2 사이에서의 배타 제어가 행해진다. 배타 제어의 방법으로서는, mutex나 세마포, 우선도 계승 프로토콜, 또는 그 밖의 기존의 혹은 고유한 방식을 적용할 수 있다.

<83> 마찬가지로, OS2에 콘텍스트가 전환된 기간은 태스크 집합 2의 실행 기간에 상당한다. 그리고, 태스크 집합 2의 실행 기간 내에서는, OS2가 갖는 스케쥴링에 따라서, 태스크 집합 2, 즉 태스크 3 및 태스크 4 사이에서의 배타 제어가 행해진다. 배타 제어의 방법으로서는, mutex나 세마포, 우선도 계승 프로토콜, 또는 그 밖의 기존의 혹은 고유한 방식을 적용할 수 있다.

<84> 본 발명은, 도 2에 도시한 바와 같은 서로 다른 방침에 따라서 스케쥴링되는 복수의 태스크 집합이 공존하는 시스템에 있어서 배타 제어를 실현하는 것이다.

<85> 도 3은 본 발명의 실시 형태에 따른 태스크 실행 환경의 구성을 모식적으로 도시하고 있다. 도 3에 도시한 바와 같이, 서로 다른 방침에 따라서 스케쥴링되는 복수의 태스크 집합이 공존하고 있다.

<86> 태스크 집합 1 관리 모듈 및 태스크 집합 2 관리 모듈은 각각 OS1 및 OS2에 상당하며, 시스템 사용권이 부여된 기간 내에서 소정의 스케쥴링 방침에 따라서 태스크 집합 1 및 태스크 집합 2의 배타 제어를 행한다.

<87> 태스크 집합 전환 소프트웨어는, 복수의 오퍼레이팅 시스템 사이에서의 콘텍스트 전환 등 오퍼레이팅 시스템의 총괄 관리를 행하는 소프트웨어 모듈로서, 어떤 태스크 집합으로부터 실행 태스크를 선출할 것인지를 관리할 수 있다. 태스크 집합 전환 소프트웨어는, 타이머(16)로부터 시간 정보를 취득하는 타이머 모듈을 사용하여, 적절한 타이밍에서 시스템의 사용권의 재부여(즉, 태스크 집합의 전환)를 행한다.

<88> 본 실시 형태에서는, 태스크 집합 전환 소프트웨어는 태스크 집합의 전환을 적합하게 관리하기 위해서, 이하에 나타내는 3 종류의 변수를 보유하고 있다.

<89> (1) 현행 태스크 집합:

<90> 현재 프로세서의 사용권이 부여되어 있는 태스크 집합을 식별하는 값이 저장되어 있는 변수이다.

<91> (2) 콘텍스트 보존 어드레스:

<92> 사용권의 재부여 시에는, 그 시점의 프로세서의 상태를 보존할 필요가 있다. 변수 「콘텍스트 보존 어드레스」는 이 보존 영역의 어드레스를 보유하는 변수이다.

<93> (3) 사용권 이행 예정 시각:

<94> 다음에 사용권이 이행하는 시각을 보유하는 변수이다. 이 변수는, 각 태스크 집합 관리 모듈이 값을 판독할 수 있는 변수이다.

<95> 도 4는 태스크 집합 전환 소프트웨어가 태스크 집합을 전환하기 위한 처리 수순을 흐름도의 형식으로 도시하고 있다.

<96> 태스크 집합의 전환 시, 우선 상기의 변수 「콘텍스트 보존 어드레스」에 현재의 상태를 보존한다(단계 S1). 여기서 말하는 상태의 보존은, 보다 구체적으로는 프로세서(11)의 레지스터 값이나 RAM(12)의 메모리 이미지의 퇴피(退避) 등을 의미한다.

<97> 계속해서, 태스크 집합의 전환이 다음에 행해지는 시각을 계산한다(단계 S2). 그리고, 계산한 시각을 상기의 변수 「사용권 이행 예정 시각」에 대입한다(단계 S3).

<98> 또한, 상기의 변수 「현행 태스크 집합」에 소유권 이행 목적지의 태스크 집합의 식별자를 대입한다(단계 S4).

<99> 계속해서, 선행 단계 S2에서 구해진, 태스크 전환이 다음에 행해지는 시각으로 타이머를 설정한다(단계 S5).

<100> 이 타이머는 설정 시각에 인터럽트를 발생한다. 이 인터럽트에 의해서, 다음회의 태스크 집합 전환 처리가 기동된다.

<101> 마지막으로, 사용권의 이행을 소유권 이행 목적지가 되는 태스크 집합 관리 모듈에 통지한다(단계 S6).

<102> 도 5는 임의의 태스크 집합에 속하는 태스크가 크리티컬 섹션에 침입할 때에 행하는 처리 수속을 흐름도의 형식으로 도시하고 있다.

<103> 현행의 태스크 집합에 있어서의 태스크의 실행을 관리하는 태스크 관리 모듈은, 태스크 집합 전환 소프트웨어가 보유하는 변수 「사용권 이행 예정 시각」을 참조하여, 태스크 실행의 여유를 판단한다(단계 S11). 여기서 말하는 여유는, 현재 시각에 크리티컬 섹션 실행 시간을 더한 값을 사용권 이행 예정 시각에서 뺀 값에 상당한다.

<104> 계속해서, 이 여유가 제로보다도 큰 지의 여부를 판단한다(단계 S12).

<105> 여유가 제로보다도 크면, 각 태스크 집합에 고유의 크리티컬 섹션의 진입 처리를 행한다(단계 S13).

<106> 한편, 여유가 제로 이하인 경우에는, 크리티컬 섹션에 돌입하면, 다른 태스크에 의한 동일한 데이터에의 조작이 배제되는 한편, 다음의 태스크 집합의 전환 시각까지 크리티컬 섹션의 처리가 완료되지 않는 모순이 발생한다. 이 때문에, 태스크 집합 관리 모듈은 태스크 집합 전환 소프트웨어에 프로세서 사용권의 포기를 통지한다(단계 S14).

<107> 이 통지에 따라서, 처리(가령 「처리 A」라고 함)의 도중에 다른 태스크 집합 중의 태스크로 제어가 옮겨가게 된다. 처리 A가 완료되는 것은, 재차 처리 A를 실행하고 있었던 태스크가 속하는 태스크에 프로세서(11)의 사용권이 공급되었을 때이다.

<108> 예를 들면, 현행의 태스크 집합에서 실행 중인 태스크에 있어서 긴급성이 낮은 처리 도중에 크리티컬 섹션에 침입할 때에는, 다음에 태스크 집합의 전환이 행해지는 예정 시각을 참조하여, 크리티컬 섹션 실행 중에 긴급성이 높은 처리가 개시되지 않는지의 여부를 검사한다. 그리고, 개시되지 않을 때에는 크리티컬 섹션으로 침입하지만, 개시되는 경우에는 긴급성이 높은 처리가 완료될 때까지 크리티컬 섹션에의 침입이 연기되도록 제어하면 된다.

<109> 따라서, 서로 다른 방침에 따라서 스케쥴링이 행해지는 복수의 태스크 집합의 사이에서, 긴급성이 높은 처리의 개시가 지연되지 않고 복수의 처리 사이에서 배타 제어를 적합하게 행할 수 있다.

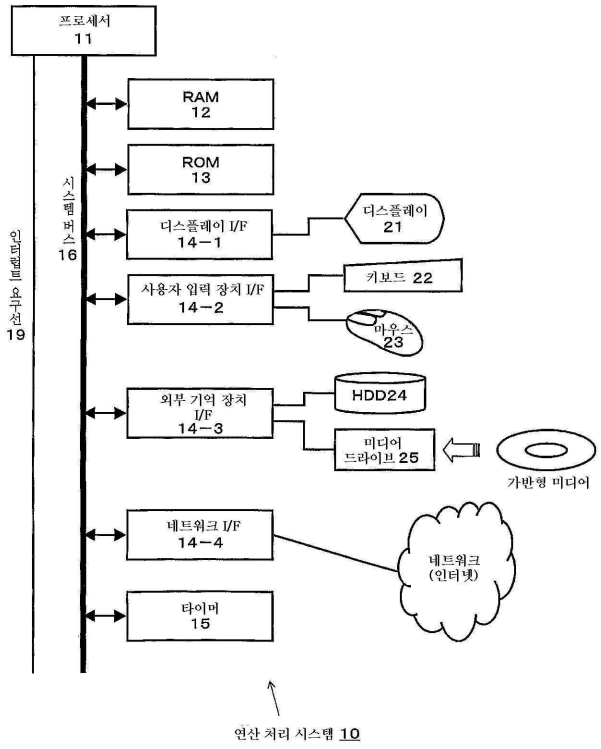
<110> <추보>

이상, 특정한 실시 형태를 참조하여 본 발명에 대하여 상세히 설명하였다. 본 발명의 요지를 일탈하지 않는 범위에서 당업자가 실시 형태의 수정이나 대용을 할 수 있는 것은 자명하다. 즉, 예시라는 형태로 본 발명을 개시하였으며, 본 명세서의 기재 내용을 한정적으로 해석하여서는 안된다. 본 발명의 요지를 판단하기 위해서는

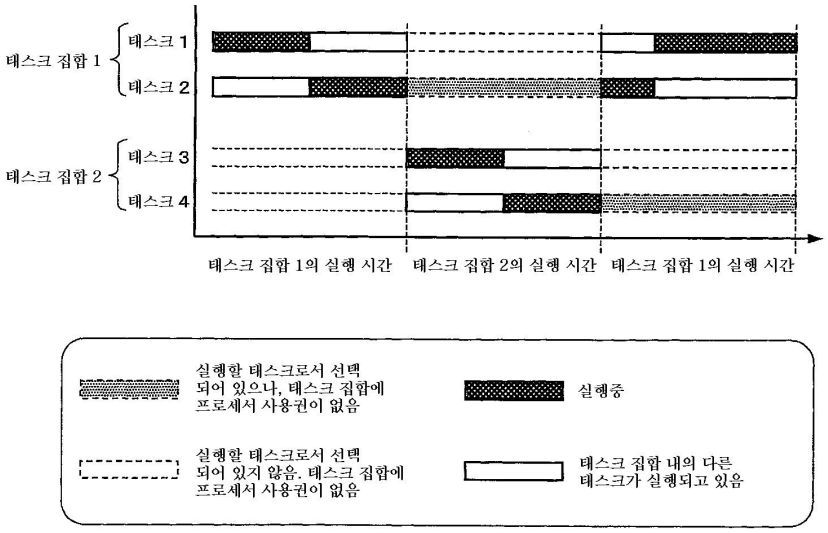
특허 청구의 범위를 참작하여야 한다.

도면

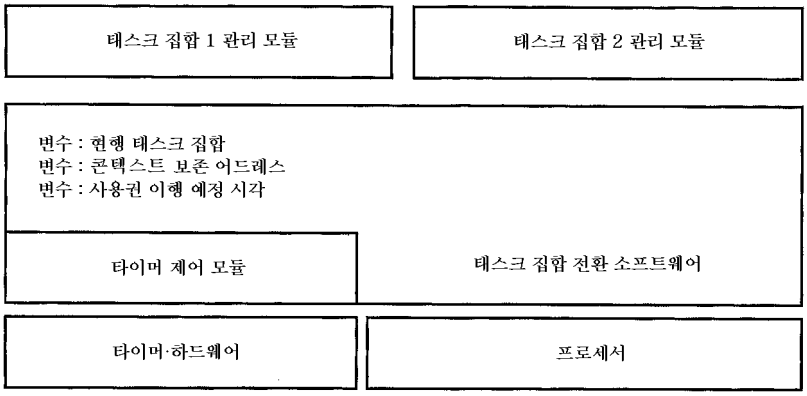
도면1



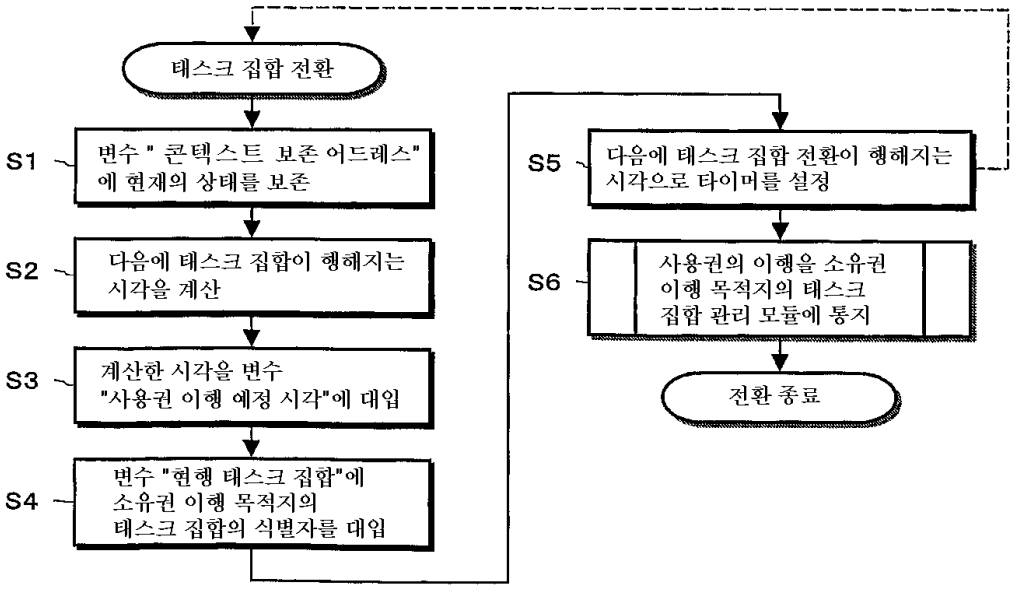
도면2



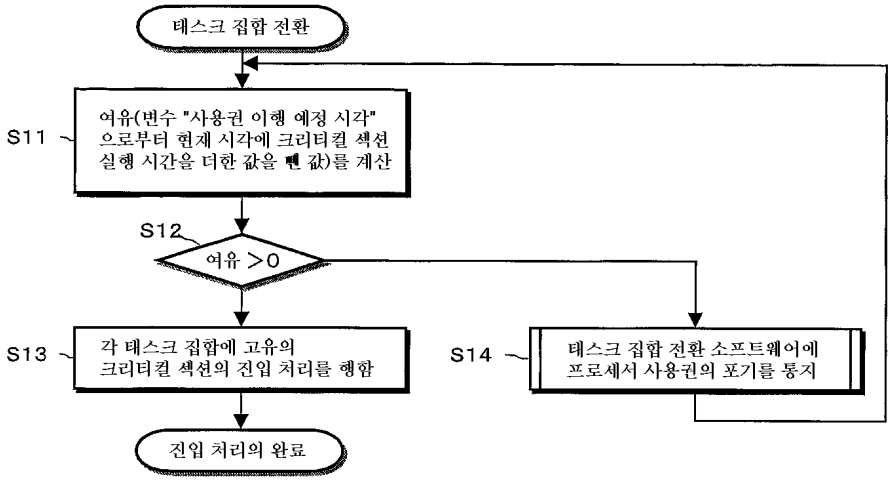
도면3


태스크 집합 1 관리 모듈
태스크 집합 2 관리 모듈
변수 : 현행 태스크 집합
변수 : 콘텍스트 보존 어드레스
변수 : 사용권 이행 예정 시각
타이머 제어 모듈
태스크 집합 전환 소프트웨어
타이머·하드웨어
프로세서

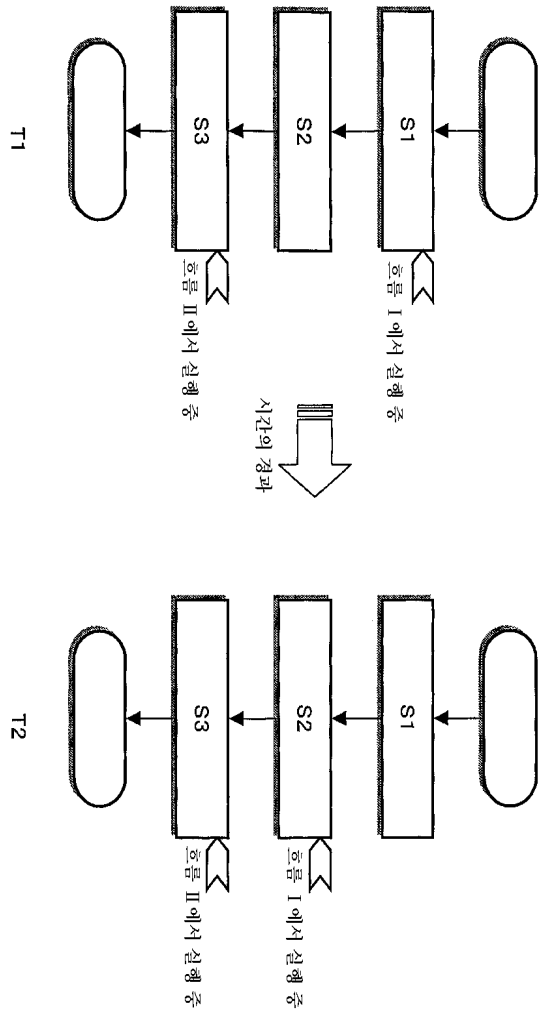
도면4


태스크 집합 전환
S1
변수 " 콘텍스트 보존 어드레스" 에 현재의 상태를 보존
S2
다음에 태스크 집합이 행해지는 시각을 계산
S3
계산한 시각을 변수 "사용권 이행 예정 시각"에 대입
S4
변수 "현행 태스크 집합"에 소유권 이행 목적지의 태스크 집합의 식별자를 대입
S5
다음에 태스크 집합 전환이 행해지는 시각으로 타이머를 설정
S6
사용권의 이행을 소유권 이행 목적지의 태스크 집합 관리 모듈에 통지
전환 종료

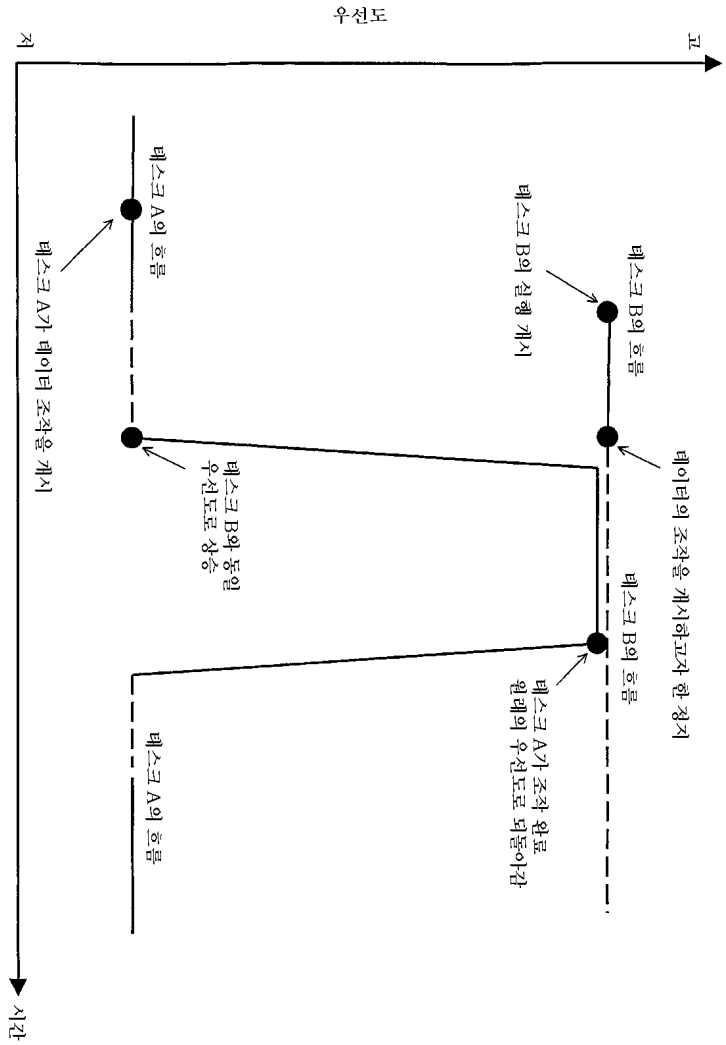
도면5


태스크 집합 전환
S11
여유(변수 "사용권 이행 예정 시각"
으로부터 현재 시각에 크리티컬 섹션
실행 시간을 더한 값을 뺀 값)를 계산
S12
여유 > 0
S13
각 태스크 집합에 고유의
크리티컬 섹션의 진입 처리를 행함
S14
태스크 집합 전환 소프트웨어에
프로세서 사용권의 포기를 통지
진입 처리의 완료

도면6


T1
S3
S2
S1
흐름 II에서 실행 중
흐름 I에서 실행 중
시간의 경과
T2
S3
S2
S1
흐름 II에서 실행 중
흐름 I에서 실행 중

도면7


우선도
저
고
시간
테스크 A가 데이터 조작을 개시
테스크 A의 흐름
테스크 B의 실행 개시
테스크 B의 흐름
데이터의 조작을 개시하고자 한 정지
테스크 B와 동일 우선도로 상승
테스크 B의 흐름
테스크 A가 조작 완료 원래의 우선도로 되돌아감
테스크 A의 흐름