(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(51) Int. Cl.[6]
G06F 12/08

(45) 공고일자 2001년01월15일
(11) 등록번호 10-0277818
(24) 등록일자 2000년10월13일

(21) 출원번호	10-1993-0703312	(65) 공개번호	특1994-0700705
(22) 출원일자	1993년11월02일	(43) 공개일자	1994년02월23일
번역문제출일자	1993년11월02일		
(86) 국제출원번호	PCT/SE 92/00282	(87) 국제공개번호	WO 92/20027
(86) 국제출원일자	1992년04월29일	(87) 국제공개일자	1992년11월12일
(81) 지정국	EP 유럽특허 : 오스트리아 벨기에 스위스 리히텐슈타인 사이프러스 독일 덴마크 스페인 핀랜드 프랑스 영국 그리스 아일랜드 이탈리아 룩셈부르크 모나코 네덜란드 포르투칼 스웨덴 국내특허 : 일본 대한민국 미국		
(30) 우선권주장	9101325-0 1991년05월02일 스웨덴(SE)		

(73) 특허권자 스웨디시 인스티튜트 오브 컴퓨터 사이언스 선드스트롬 시워트
스웨덴 키스타에스-164 28 박스1263
(72) 발명자 하저스텐, 에릭
스웨덴왕국, 하저스텐 에스-126 57, 트레스코바켄 23
(74) 대리인 나영환, 이상섭

심사관 : 김세영

(54) 컴퓨터 시스템의 데이터 처리 속도를 증가시키는 방법

요약

본 발명은 적어도 하나의 마이크로프로세서(1)와, 메모리 장치(3)와, 상기 프로세서에 접속되는 캐시 메모리(2, 4)를 포함하는 컴퓨터 시스템의 데이터 처리 속도를 증가시키는 방법에 관한 것으로서, 상기 캐시 메로리(2, 4)는 프로세서(1)에 의해 요구되는 메모리 장치(3)내의 어드레스로부터 데이터를 페치하고, 또 프로세서(1)에 의해 요구되지 않는 메모리 장치(3)내의 하나 또는 몇 개의 어드레스로부터 데이터를 페치하도록 배열되어 있다. 본 발명은 스트림 검출 회로(5)라고 불리우는 회로가 캐시 메모리(2, 4)와 상호 접속하여, 프로세서(1)가 캐시 메로리(2, 4)내에서 요구하는 어드레스를 검출하는 단계와, 상기 어드레스가 캐시 메모리(2, 4)내에 이미 존재하는지의 여부를 레지스터 하는 단계와, 캐시 메모리(2, 4)내의 프로세서(1)에 의해 요구되는 하나 또는 몇 개의 순차적 어드레스열을 검출하는 단계와, 상기 어드레스열을 검출시 다음 어드레스에 대응하는 메모리 장치(3)로부터 데이터를 페치하고 상기어드레스를 캐시 메모리(2, 4)내에 삽입하기 위해 상기 캐시 메모리(2, 4)를 커맨드하는 단계를 포함하는 것을 특징으로 한다.

대표도

도1

명세서

[발명의 명칭]

컴퓨터 시스템의 데이터 처리 속도를 증가시키는 방법

[발명의 상세한 설명]

본 발명은 컴퓨터 시스템의 데이터 처리 속도를 증가시키는 방법에 관한 것이다. 최근 들어, VLSI(Very Large Scale Integrated)회로의 발달로 마이크로프로세서 및 메모리 장치 사이에 액세스(access) 시간의 격차가 넓어지게 되었다. 메모리 장치는 고속인 마이크로프로세서와 비교할 때 비교적 느리다. 이와 같은 속도의 차이를 좁히기 위해서 소위 캐시(Cache)라는 메모리가 도입되었다. 캐시는 마이크로 프로세서와 메모리 장치 사이에 설치된다. 캐시는 칩의 형태로서 비교적 소형이고 고속인 메모리 장치이다. 예컨대, 캐시 내에는 마이크로프로세서에 의해 자주 사용되는 데이터가 저장된다. 이러한 캐시는 그 캐시 및 마이크로프로세서와 비교할 때 느린 대용량 메모리 장치로부터 데이터를 페치(fetch)한다. 종종, 마이크로프로세서와 대용량 메모리 장치 사이에는 둘 또는 몇몇 캐시가 계층적으로 배열되기도 한다.

또한, 캐시는 다중 프로세서 시스템 내에서도 사용될 수 있는데, 여기서 마이크로프로세서는 각각의 캐시에 접속되며, 그 각각의 캐시에 의해 정보를 대용량 메모리 장치에 저장하고 그 대용량 메모리 장치로부터 정보를 검색할 수 있다.

이러한 기술의 사용예로는 각각의 프로세서가 프로그램의 상이한 부분을 실행함으로써 메모리 장치로부터 상이한 데이터를 페치해야 하는 다중 프로세서 시스템이 있다. 프로세서가 하나의 실행을 완료하고 새로운 실행을 개시할 때, 그 프로세서는 단지 메모리 장치에 저장된 데이터의 일부분만을 필요로 하게 된다.

이 경우, 프로세서는 먼저 캐시에 데이터의 첫 번째 부분을 요구한다. 만일 캐시가 데이터의 첫 번째 부분을 가지고 있지 않을 경우, 캐시는 메모리 장치로부터 데이터를 페치하여 이를 저장한다. 프로세서가 캐시 내에 저장되어 있지 않은 데이터를 요구함에 따라 캐시 내의 데이터는 증가하게 된다. 프로세서가 단지 프로그램의 특정 부분만을 실행하기 때문에, 프로그램의 실행이 진행됨에 따라 프로세서가 원하는 데이터는 캐시가 그 데이터를 이전의 요청에 의해 이미 프로세서로부터 페치하였을 경우가 많아져서 캐시 내에 존재할 확률이 높아진다. 캐시로부터 데이터의 부분을 페치하기 위한 액세스 시간은 프로세서가 대용량 메모리 장치로부터 직접 데이터의 부분을 페치하기 위한 액세스 시간에 비해 현저히 짧다. 따라서, 메모리와 프로세서간의 데이터 처리 속도는 증가되고, 메모리 장치와 프로세서간의 속도차를 감소시켜 데이터 처리 속도를 증가시킨다.

캐시를 사용하여 메모리 속도를 더욱 증가시키기 위해서, 메모리 장치의 어드레스에서 데이터를 페치할 때 프로세서가 요구할 그 다음 데이터 부분을 예측하여 그 근처 어드레스 데이터를 동시에 페치하여 캐시에 미리 저장하려는 시도가 있어 왔다. 또한, 데이터의 단일 부분이 요구되는 경우 데이터의 전체 블록을 페치할 수도 있을 것이다. 위의 경우에, 프로세서가 어떠한 블록 내의 하나의 어드레스를 요구할 때 동일한 블록 내의 몇몇 어드레스를 반드시 요구한다는 보장을 할 수 있다면 이로울 것이다. 그러나, 이는 많은 양의 사용되지 않는 데이터가 페치된다는 것을 의미하며, 이는 요구되는 캐시의 필요 메모리 용량이 증가하게 되는 원인이 되어, 메모리 속도를 감소시키게 된다.

본 발명은 마이크로프로세서가 데이터를 요구하기 전에 캐시로 그 데이터를 미리 페치하는 경우에, 이 프리페치(prefetch)된 데이터가 프로세서에 의해 요구될 확률이 데이터를 프리페치하는 공지된 기타의 시스템보다 훨씬 더 높아서, 캐시 크기가 작고, 따라서 캐시 속도가 매우 빠르게 되는 방법을 제공한다.

따라서, 본 발명은, 적어도 하나의 마이크로프로세서, 메모리 장치 및 그 프로세서에 접속되는 캐시 메모리를 포함하며, 상기 캐시 메모리는 프로세서가 요구한 메모리 장치 내의 어드레스로부터 데이터를 페치하며, 이로 인해 프로세서가 요구하지 않은 메모리 장치 내의 하나 또는 몇몇 어드레스로부터의 데어터도 페치하도록 구성되는 컴퓨터 시스템의 데이터 처리 속도를 증가시키기 위한 방법에 있어서, 스트림 검출 회로(stream-detection circuit)라고 불리는 회로를 캐시와 상호 접속하여, 캐시 내의 프로세서가 요구하는 어드레스를 모니터(monitor)하고, 그 어드레스가 캐시 내에 이미 존재하는지의 여부를 레지스터(register)하고, 캐시 내에서 프로세서에 의해 요구되는 하나 또는 몇몇의 순차적인 어드레스 열을 검출하고, 상기 열을 검출하면 그 캐시를 명령하여, 상기 열의 다음 어드레스에 대응하는 데이터를 메모리 장치로부터 페치하고 그 어드레스를 캐시 내에 삽입시키도록 상기 스트림 검출 회로를 구성한 것을 특징으로 한다.

본 발명은 통상적으로 상이한 컴퓨터 시스템 및 개별적인 캐시 메모리와 계층적으로 배열된 캐시 메모리 양자를 모두 갖는 시스템과 접속하여 사용될 수 있다. 이하에서는, 첨부된 본 발명의 일실시예에 관한 도면을 부분적으로 참조하여 본 발명을 보다 상세히 설명한다.

제1도는 컴퓨터 및 메모리 장치에 부가된 캐시 메모리를 도시한 도면.

제2도는 스트림 오브젝트 회로(stream-object circuit)를 도시한 도면.

제3도는 체커 오브젝트 회로(checker-object circuit)를 도시한 도면.

제4도는 스트림 검출 회로를 도시한 도면.

제5도는 소프트웨어로 구현한 단순화된 스트림 검출 회로의 자유 형식의 시방서.

제1도는 캐시(2)가 부착된 컴퓨터(1)를 도시하고 있다. 컴퓨터와 메모리 장치(3) 사이에는 캐시 2로 불리는 또 다른 캐시(4)가 존재한다. 본 발명에 따르면, 스트림 검출 회로(5)가 컴퓨터와 캐시 2 사이에 접속된다. 이 스트림 검출 회로에는 인텔리전트 스트림(Intelligent Stream)을 나타내는 아이스트림(Istream)이라는 명칭이 붙여져 있다.

상기 스트림 검출 회로는 소위 스트림 오브젝트 회로 및 체커 오브젝트 회로를 포함한다.

체커 오브젝트 회로는 프로세서로부터의 트래픽(traffic)에서 어드레스 패턴(pattern), 즉 순차적인 어드레스 열의 존재 여부를 검출한다. 이러한 검출은 현재 어드레스와 더불어 가장 공통적으로 발생한 요구된 어드레스들 사이의 간격의 목록, 예컨대(+1, -1, -17)과 같은 목록에 기초하여 어떠한 어드레스가 요구될 것인지를 추정함으로써 달성된다. 상기 목록은 체커 오브젝트 회로를 활성화시켜, 어드레스 A가 요구될 때 어드레스 A+1, A-1 및 A+17이 앞으로 요구될 것이라고 추정한다. 미리 정해진 패턴을 이용한 추정이 소정의 횟수동안 옳다고 판정되면, 소위 스트림 오브젝트가 개시 및 시작될 것이다.

상기 체커 오브젝트 회로는 체크될 어드레스(Caddr)와, 미리 언급된 스텝(Step)과, 카운터(Ctr)에 대한 정보를 포함한다. Caddr 은 체크되거나 감시될 하나의 어드레스를 포함하지만, 프리페치(prefetch)를 발생시키지는 않는다. Caddr내의 어드레스가 프로세서와 캐시 2 사이에 나타나는 경우, 어드레스의 스트림이 식별된 것으로 간주될 수 있는 것인가 또는 그렇지 않는 것인가에 대한 경계치인 임계값과 동일한 수치까지 Ctr이 카운트되었는지가 체크된다. 만일 Ctr이 임계값을 초과하지 않으면, Caddr은 스텝 내의 간격만큼 증가되고, Ctr은 1만큼 증가된다.

상기 스트림 오브젝트 회로는 가장 오래전에 페치된 어드레스(Oaddr), 가장 최근에 페치된 어드레스(Naddr) 및 다음 어드레스를 계산하기 위한 함수에 관한 정보를 포함한다. 이러한 함수의 예로는, 예컨대 +1 또는 -1과 같은 프리페치된 어드레스간의 일정한 차이(간격)의 함수가 있다. 스트림 오브젝트 회로는, 프로세서로 부터의 Oaddr에 대한 각 요구가 프리페치된 데이터가 사용 가능하다는 것을 나타내어 추가적인 프리페치를 활성화시키도록 배치된다. 프로세서가 Oaddr에 대해 요구했으나 Oaddr이 캐시 내에 있지 않으면, 즉 캐시 미스(miss)가 발생하면, 프리페치는 충분히 빨리 수행되지 않게 되고 하나 이상의 어드레스를 프리페치하게 된다. 프로세서로부터 요구된 어드레스가 캐시 내에 있는 경우에는, 프리페치는 적절히 수행된 것이고 단지 하나의 추가 어드레스만이 프리페치된다.

따라서, 스트림 오브젝트 회로는 특정 패턴, 예컨대 A+1 및 A-1과 같은 인접 어드레스들에 따라 어드레스

스트림 또는 데이터의 스트림의 페치를 발생한다. 이 패턴은 응용에 따라 간단히 또는 복잡하게 만들어질 수 있다.

그러나, 어드레스 또는 데이터의 스트림의 존재 여부를 검출하는 것은 체커 오브젝트 회로이다. 이 체커 오브젝트 회로는 프로세서가 캐시 내에 있지 않은 어드레스를 요구하였을 때, 체커 오브젝트 회로가 감시 또는 관찰할 하나 또는 몇몇 어드레스, 즉 소위 체커 오브젝트를 선택하는 방식으로 검출을 수행한다. 이 선택은 가장 최근에 요구된 어드레스에 1을 가산하거나 1을 감산하는 것과 같은 특정 패턴에 따라 행해진다(즉, A+1 또는 A-1을 선택함). 상기 체커 오브젝트 회로가 소정의 회수에 걸쳐 올바른 어드레스의 추정에 성공하면, 어드레스 또는 데이터의 스트림은 검출된 것으로 여겨진다. 그 성공 횟수는 카운트되어 미리 정해진 수치(본 명세서에서는 임계값)와 비교된다.

바람직한 실시예에 있어서, 어드레스 열은 둘 또는 몇몇 어드레스로 구성될 수 있다.

특별한 실시예에 있어서, 스트림 검출 회로는 프로세서가 캐시 내의 어드레스를 요구할 때, 상기 스트림 검출 회로가 어드레스 열이 증가하고 있는지 또는 감소하고 있는지를 검출할 때까지, 추정된 어드레스 열 내의 상위 및 하위 어드레스 모두에 해당하는 데이터를 메모리 장치로부터 페치하도록 구성될 수 있다.

또 다른 바람직한 실시예에 있어서, 검출 가능한 전술한 열은 부가적, 산술 또는 기하학적이거나, 또는 기타 논리적으로 구성될 수 있다.

또 다른 바람직한 실시예에 있어서, 새롭고 관심있는 패턴이 실행(execution) 중에 검출되어, 체커 오브젝트 회로가 앞으로 요구될 어드레스에 대한 어드레스를 추정하는데 사용되는 함수의 목록에 첨가된다. 이러한 경우의 한가지 예는 요구된 어드레스들간의 새로운 고정 간격이 이용되기 시작될 때이다.

이러한 스트림 검출 회로는 아래에 설명하는 예에서와 같이 비교적 간단하게 하드웨어로 구성될 수 있으나, 공지된 형태의 완전히 적절한 프로세서에 의해 처리되는 소프트웨어로도 구성될 수도 있다. 이 경우, 특정 하드웨어의 구현은 필요없다. 소프트웨어 구현의 경우에 있어서, 스트림 검출 회로를 프로그래밍하는 것은 명백히 큰 이점이 있는데, 이럼으로써 여러 가지의 더 복잡하거나 덜 복잡한 다양한 데이터의 스트림을 검출하는 소프트웨어로서의 스트림 검출 회로를 구현할 수 있다. 이에 따라, 스트림은, 예컨대 어드레스들간의 고정 스텝에 제한되지 않으며, 다양한 어드레스 열이 검출될 수 있다. 예컨대, A+1, A+2, A+4, A+8, . . . A+n의 어드레스 열 또는 A+1, A+2, A+3, A+11, A+12, A+13, A+21, A+22, ...의 어드레스 열이 검출될 수 있다.

따라서, 본 발명은 검출된 열을 사용하여 캐시 내의 어느 데이터가 높은 확률로 프로세서에 의해 요구될 것인지를 예측하고, 이를 통해 그 데이터를 프리페치하기 위해서 꼭 검출되어야 하는 특별한 구성의 열의 검출에 대해 전혀 제한이 없다.

어드레스를 적절한 시간에 프리페치하기 위해서, 즉 프로세서에 의해 요구될 때 프리페치된 어드레스가 캐시 내에 제때 도달하도록 하기 위해서, 또 다른 바람직한 실시예에서 프리페치된 데이터의 양은 메모리 장치로부터 캐쉬로 막 도달한 데이터에 대한 캐시 내의 어드레스를 마이크로프로세서가 요구할 때까지 증가된다.

이하에서는 캐시와 함께 스트림 검출 회로의 기능에 대한 알고리즘을 설명하는 간단한 예로써 캐시가 본 발명에서 어떻게 기능하는 지를 설명한다. 예시가 제1도의 오른쪽 표에 도시되어 있으며, 프로세서와 캐시 2간의 트래픽과, 소위 캐시 히트(hit)가 발생했는지 여부 및 어드레스를 프리페치하는 동작을 설명하고 있다. 아래의 예에서, 간단한 스트림 검출 회로는 두 개의 체커 오브젝트 회로 및 네 개의 스트림 오브젝트 회로를 포함한다. 스텝 간격은 +1 및 -1로 설정된다. 이 예에 있어서, 스트림의 발견에 대한 소위 임계치는 수치 1로 설정된다. 즉, 체커 오브젝트가 프로세서에 의해 요구된 캐시 2 내의 어드레스와 일치할 때 즉시 스트림이 생성된다.

이러한 동작은 '연구 트래픽(studied traffic)'이란 컬럼에 따라 한줄씩 기재되어 있다.

'연구 트래픽' 컬럼 및 '프리페치 동작' 컬럼에는 '판독' 및 '기록'이 있다. 이는 캐시에 대한 프리페치된 데이터가 '판독' 또는 '기록' 모드에서 프리페치된다는 것을 의미한다. 이는 본 발명을 멀티프로세서에서 구현할 때에 큰 중용성을 갖는다.

판독 A- 캐시 내의 어드레스 A에 대한 프로세서로부터의 요구는 캐시 2 내에서의 미스를 초래한다. 즉 어드레스는 캐시 2 내에 존재하지 않는다. 그러면, 캐시 2는 메모리 장치(3)로부터 A를 페치한다. 어드레스 A는 모든 Oaddr과 비교되는데, 상기 Oaddr은 현재 아무런 어드레스를 포함하지 않으므로 일치가 발생하지 않는다. 그 다음 소위 체커 오브젝트 C(A-1) 및 C(A+1)이 형성된다.

기록 B - 어드레스 B는 이미 캐시 2 내에 존재한다고 가정하면, 소위 캐시 히트가 발생한다. 어드레스 B는 현재 아무런 어드레스를 포함하지 않는 모든 Oaddr과 비교된다.

판독 A+1 - 캐시 2 내의 미스는 캐시 2로 하여금 A+1을 페치하도록 한다. 어드레스 A+1는 두 개의 체커 오브젝트 C(A-1) 및 C(A+1)과 비교되어, A+1과의 일치가 발생한다. 하나의 스트림이 Naddr = A+2 및 Oaddr = A+1 및 Step = 1로 생성된다. 스트림 검출 회로(5)는 캐시 2로 하여금 A+2를 프리페치하게 한다.

판독 A+2 - 캐시 2 내의 미스로 캐시 2는 A+2를 페치하게 된다. 정확하게는, 이 페치는 이전 단계인 판독 A+1에서 개시되었지만, A+2는 캐시 2에 도달하기 위해 중분한 시간을 아직 갖지 못한 것이다. A+2는 모든 Oaddr과 비교되어 일치를 발생한다. 이는 스트림 검출 회로로 하여금 프리페치한 데이터 준비 정도를 증가시키기 위하여 캐시 2가 2개의 어드레스, 즉 A+3 및 A+4를 프리페치하도록 한다. Oaddr은 A+3으로 변경되고 Naddr은 A+4로 변경된다.

기록 C - 이 요구는 캐시 2 내의 미스를 발생한다. 스트림 검출 회로는 캐시 2로 하여금 C를 페치하도록 한다. 어드레스 C는 모든 Oaddr(A+3)과 비교되나, 어떠한 일치를 발생하지 않는다. 또한, 어드레스 C는 2개의 체커 오브젝트(A+1, A+2)와 비교되지만, 어느 것과도 일치하지 않게 된다. 따라서, 2개의 새로운 체커 오브젝트 C(C+1) 및 C(C-1)이 생성된다.판독 D - 어드레스 D가 이미 캐시 2 내에 존재한다고

가정하면, 캐시 2의 히트가 발생한다. 어드레스 D는 모든 Oaddr(A+3)과 비교되나, 어떠한 일치도 발생하지 않는다.

판독 A+3 - 이는 캐시 2 내의 히트를 발생한다. 이 어드레스는 모든 Oaddr(A+3)과 비교되어, 일치를 발생한다. 그 다음, Oaddr은 A+4로, Naddr은 A+5로 변경된다. 또한, 스트림 검출 회로는 캐시 2로 하여금 판독 인스트럭션 (A+5)로 A+5를 프리페치하도록 한다.

판독 A+1 - 이는 캐시 2내의 히트를 발생한다. 이 어드레스 A+1 은 모든 Oaddr(A+4)과 비교되어, 어떠한 일치도 발생하지 않는다. 더 이상 아무 일도 발생하지 않는다.

기록 C-1 - 이는 캐시 2 내의 미스를 발생한다. 어드레스 C-1은 모든 Oaddr(A+4)과 비교되어, 어떠한 일치도 발생하지 않는다. C-1은 모든 체커 오브젝트(A+1, A-1, C+1, C-1)와 비교되어 히트를 발생한다. 그 다음, Naddr = C-2, Oaddr = C-2 및 Step = -1인 스트림이 개시된다. 또한, 어드레스 C-2가 기록 인스트럭션(C-2)으로 프로페치된다.

판독 A+4 - 이는 캐시 2 내의 히트를 발생한다. 어드레스 A+4는 모든 Oaddr, 즉, (A+4) 및 (C-2)와 비교되어, C-2상에서 일치를 발생한다. Oaddr은 A+5로 변경되고, Naddr,은 A+6으로 변경되며, 어드레스 A+6이 판독 인스트럭션 (A+6)으로 프리페치된다.

기록 C-2 - 이는 캐시 2 내의 히트를 발생한다. 어드레스 C-2는 모든 Oaddr(A+5, C-2)와 비교되어 일치를 발생한다. 그 다음, Oaddr은 C-3으로 변경되고 Naddr은 C-3으로 변경된다. 어드레스 C-3이 기록 인스트럭션 (C-3)으로 프리페치된다.

전술한 알고리즘은 특히 하드웨어 구현에 적합하다. 더욱이, 이 알고리즘은 소위 제2 레벨 캐시에 어드레스를 페치하는 데 특히 적합하다. 프로세서로부터의 대부분의 요구는 캐시 히트를 발생하게 된다. 캐시 히트가 발생하면, 매 스트림 오브젝트마다 하나의 비교만이 필요하다. 캐시 미스가 발생하면, 즉 요구된 어드레스가 캐시 내에 없으면, 매 스트림 오브젝트 및 매 체커 오브젝트마다 비교가 요구된다.

제5도는 소프트웨어로 구현된 간단한 스트림 오브젝트 회로의 자유로운 형식의 시방서(specification)이다. 제5도는 간단한 경우의 소프트웨어 구성을 도시하고 있다. 그러나, 당업자는 이 구성을 소프트웨어로 구현된 보다 복잡한 스트림 검출 회로에 사용하기를 바랄 것이다. 제5도는 '단순화된 아이 스트림(I-stream)의 개략 설명'으로 명명하였다.

이하에서는 본 발명에 따른 하드웨어로 구현한 스트림 검출 회로의 예를 설명한다. 3개의 도면을 참조하여 설명되는데, 3개의 도면은 스트림 오브젝트 회로를 도시하는 제2도, 체커 오브젝트 회로를 도시하는 제3도 및 체커 오브젝트 회로 및 스트림 오브젝트 회로를 포함하는 스트림 검출 회로를 도시하는 제4도이다.

제3도 및 제4도에 있어서, '제어(control)' 기호는 데이터의 실행 경로 및 데이터의 흐름을 지시하는 논리 회로(logic)를 나타낸다.

제4도는 스트림 검출 회로의 데이터 경로에 대한 상위 레벨 구성을 도시하고 있다. 입력 신호 'address'는 프로세서에 의해 캐시에 요구된 어드레스를 포함하며, 입력 신호 'miss'는 요구된 어드레스가 캐시 내에 존재하는지 여부를 나타낸다.

'체커 어레이(checker-array)' 블록은 몇몇 체커 오브젝트를 포함하는 것으로 가정한다. 이 블록의 목적은 프로세서로부터의 요구 내에서 새로운 패턴을 발견하는 것이고, 캐시 내의 미스가 이미 액티브(active)된 스트림에 포함되지 않은 요구와 함께 검출되었을 때 '인에이블(enable)' 신호로 활성화된다.

임의의 패턴에 따른 새로운 요구가 검출되면, 's.create' 신호가 LRU-BOX 블록으로 전송된다. 이 LRU 박스는 가장 오랫동안 동작하지 않았던 스트림 오브젝트를 선택하는 일을 한다. 이를 결정하기 위해서, 각 스트림 오브젝트는 매번 동작할 때마다 신호(hit0, hit1, hit2, hit3)를 전송한다. LRU 박스는 선택된 스트림 오브젝트에 신호(create0, create1, create2, create3)를 전송함으로써 선택된 스트림 오브젝트를 개시한다.

이 'create' 신호로 동작하면 선택된 스트림 오브젝트의 초기 어드레스값(init. addr)은 현재 요구된 어드레스 및 검출된 패턴의 스텝(step)의 합으로서 계산된다. 또한, 어드레스는 프리페치를 위해 '출력 버퍼'로 전송된다. 또한, 상기 패턴의 스텝이 스트림 오브젝트를 개시하기 위해 전송된다(init.step).

입력 신호 'address' 및 'miss'는 모든 스트림 오브젝트에 전송되는데, 상기 스트림 오브젝트는 그 신호를 액티브 스트림을 체크하는 데 사용한다. 스트림 오브젝트는 프리페치될 새로운 어드레스를 '출력 버퍼'로 전송한다.

제2도는 스트림 오브젝트로 회로를 도시하고 있다. 이 회로는 3개의 레지스터, 즉 step, naddr 및 oaddr을 포함한다. 이 레지스터들은 입력 신호인 create가 활성일 때 개시되어, 'init step'(initial step), 'init naddr'(initial new address) 및 'init oaddr'(initial old address)가 각 레지스터에 기록되게 한다. 입력 신호 s.create가 비활성이면, oaddr의 내용은 프로세서에 의해 곧 요구되는 'address' 신호와 비교될 것이다. 여기서, 일치가 검출되면, 출력 신호 'stream-hit'가 발생하는데, 이는 이전에 프리페치된 어드레스가 지금 요구되고 있다는 것을 나타낸다. 출력 신호 ' prefetch-addr'는 naddr과 step을 가산하여 생성된다. stream-hit가 활성이면, step과 oaddr을 가산함으로써 프리페치 어드레스(prefetch-addr)를 계산하는 데이터 경로가 활성될 것이다. 또한, Naddr 레지스터는 prefetch-addr로 갱신될 것이다. 'miss' 입력 신호가 활성이면, 새로운 naddr을 발생하기 위해 step 및 naddr을 더하는 데이터 경로가 두 번 활성화되고 추가의 프리페치 어드레스(prefetch addr)가 생성될 것이다.

제3도는 체커 오브젝트 회로의 어레이를 포함하는 체커 오브젝트 회로를 도시하고 있다. 입력 신호로는 체크될 어드레스(address), 활성화 신호(enable) 및 전술한 create 신호가 있다. 출력 신호로는 step, 스트림을 생성하기 위한 신호(s.create) 및 프리페치된 어드레스가 검출되었다는 것을 나타내는 'hit' 신호가 있다. 제3도는 체커 오브젝트 회로의 어레이를 도시하고 있는데, 각 체커 오브젝트 회로는 3개의 레지

스터, 즉 caddr(체크될 어드레스), ctr(이는 이전에 언급된 임계치에 도달하였는지 여부를 체크하기 위한 수치를 포함함) 및 step를 포함한다. 하나의 오브젝트는 'forall.ctr' 레지스터 또는 'fifo.repl' 레지스터 중 어느 하나에 의해 선택된다.

'enable' 입력 신호가 활성화되면, 체크가 수행된다. forall.ctr 카운터는 매번 어드레스가 체크될 때마다 전체 어레이를 걸쳐 동작한다. 선택된 오브젝트의 caddr 레지스터의 내용은 인입 어드레스와 비교된다. 선택된 오브젝트의 ctr의 내용은 상수값 c-threshold, 즉 임계치와 비교된다. 선택된 오브젝트의 step 레지스터의 내용은 'step' 출력 신호로서 출력된다. 선택된 caddr이 인입 어드레스와 동일하고 선택된 str 이 c-threshold 내의 임계치보다 크면, c.create 출력 신호가 발생한다. 선택된 caddr이 인입 어드레스와 동일하고 선택된 ctr이 c-threshold 내의 임계치보다 크지 않으면, 출력 신호 int.inc(internal increment)가 발생한다.

int.inc 신호는 선택된 ctr 내의 수치를 증가시키고 새로운 수치를 ctr 레지스터에 입력시키는 데이터 경로를 활성화한다. 또한, int.inc 신호는 선택된 step을 인입 어드레스에 더하여 그 결과를 caddr 레지스터에 기록하는 데이터 경로를 활성화한다.

'hit' 출력 신호는 caddr(체크되는 어드레스) 및 인입 어드레스간의 어드레스에 관한 일치가 검출되었는지를 나타낸다.

입력 신호 creat가 활성화되면, fifo.repl의 내용에 의해 오브젝트가 선택된다. 그 다음, 선택된 레지스터의 caddr이 인입 어드레스와 c-interval의 목록의 첫 번째 값을 더한 값으로 설정되며, ctr은 1로 설정되고, step은 c-interval 목록의 첫 번째 값으로 설정된다. 그 다음, fifo.repl 레지스터는 증가되고, c-interval 목록의 두 번째 값에 대하여 동일한 작업을 수행하도록 사용된다. 이 과정은 c-interval상의 모든 값이 사용될 횟수만큼 반복된다.

제4도는 체커 오브젝트 회로의 어레이 및 몇몇 스트림 오브젝트 회로를 포함하는 스트림 검출 회로를 도시하고 있다. 인입 어드레스는 스트림 오브젝트 회로 및 체커 오브젝트 회로의 어레이로 전달된다. 인입 어드레스는 그 어드레스가 캐시내에서 미스를 발생하였는지를 알려준다. 프리페치될 데이터에 대한 어드레스는 출력 레지스터인 '출력 버퍼'내에 출력 신호로서 표시된다.

캐시 내의 미스가 일어나고 스트림 오브젝트 회로 내에서 히트가 발생하지 않으면, 즉 일치되지 않으면, 체커 어레이가 활성화될 것이다. 활성화된 체커 어레이가 히트를 생성하지 않으면, 새로운 체커 오브젝트, 즉 체크될 어드레스가 체커 어레이에 전달된 's.create' 신호에 의해 생성될 것이다.

체커 어레이 내의 히트(즉, 어드레스가 예상됨)는 'hit' 출력 신호로 도시되어 있다. 's.create' 출력 신호는 히트가 존재하고 스트림을 생성하기 위한 임계치가 초과되었다는 것을 표시한다. 이 s.create 신호는 소위 LUR(least recentrly used) 박스로 전달되고, 상기 LUR 박스는 create 신호를 그 다음 개시될 스트림 오브젝트 회로로 전달한다. LUR 박스는 가장 오랫동안 활성화되지 않은 스트림 오브젝트 회로를 선택한다. 선택된 스트림 오브젝트 회로는 체커 어레이로부터의 init.step을 수신하고, 프로세서에 의해 현재 요구된 어드레스와 step의 합으로부터의 init.oaddr 및 init.addr를 수신한다.

스트림 오브젝트 회로의 개시와 동시에, 데이터의 첫 번째 부분이 프리페치된다. 스트림이 검출되면, 프리페치될 데이터의 어드레스는 레지스터 출력 버퍼로 전달된다. 스트림이 검출되면 하나 이상의 어드레스가 프리페치될 수 있다.

상기 구현은 어드레스의 수와 프로세서 및 캐시간의 전송 속도에 의존한다.

스트림 검출 회로는 버스에 접속되어 프로세서와 캐시 사이에서 동작하는 개별적인 기능 유닛으로 구현되거나, 이와 달리 캐시에 통합된 일부로 구현될 수 있다. 이 차이는 어드레스를 프리페치하도록 발행된 스트림 검출 회로의 커멘드의 목적지가 어디냐에 의존한다.

개별적인 유닛의 경우에 있어서, 스트림 검출 회로는 버스에 접속되는 임의의 기타 장치로서 버스상에 커맨드를 발행한다. 적절한 버스의 예로는 모토롤라(Motorola)의 Mbus 및 Futurebus+가 있다.

스트림 오브젝트 회로를 캐시 내에 통합시키는 것은 캐시에 대해 버스상에 추가적인 트래픽을 발생하지 않아도 된다는 장점이 있다.

전술한 캐시의 실시예는 몇몇 응용 분야에 매우 적합하다.

그 중 한 분야는 어레이 또는 파일을 검색하는 경우와 같이 어드레스 요구가 완전히 순차적일 때이다. 다른 분야는 각 요소가 수회에 걸쳐 발생하고 다시 사용되지 않는 큰 어레이를 요구하는 루프(loop)의 경우이다. 또 다른 분야는 행렬 계산과 같은 과학적 계산의 경우이다. 예컨대, 행렬의 곱셈에 있어서는 3개의 스트림이 발생된다. 제1 행렬로부터 판독 후 그 결과 행렬에 기록하여 그 값이 1인 하나의 스텝의 스트림이 발생하고, 다른 오퍼랜드를 판독하여 열의 수와 동일한 수치의 스텝을 발생한다.

또 다른 분야는 서서히 증가하는 소위 스택의 생성에 관한 이용이다.

스트림 검출 회로는 시동 단계(startup stage), 즉 소위 콜드 스타트(cold start)거나 프로세서가 다른 프로세서에 결합 또는 재접속된 후에 바로 시동되는 경우에 가장 많이 사용된다.

이제까지, 여러 응용예를 설명하였다. 본 발명은 본 발명의 기본적인 사상을 벗어남이 없이 상기 실시예들과 상이하게, 데이터 또는 어드레스의 스트림을 검출하고 이를 사용하여 데이터 또는 어드레스에 대한 후속 요구를 예측하여 프리페치하도록 변형될 수 있다.

따라서, 본 발명은 전술한 예시적인 실시예에 제한되는 것이 아니라, 첨부된 청구 범위 내에서 변형이 가능하다.

(57) 청구의 범위

청구항 1

적어도 하나의 마이크로프로세서(1), 메모리 장치(3) 및 상기 마이크로프로세서에 접속되는 캐시(cache)(2, 4)를 포함하며, 상기 캐시(2, 4)는 마이크로프로세서(1)가 요구하는 메모리 장치(3) 내의 어드레스(address)로부터 데이터를 페치(fetch)하고, 이로 인해 마이크로프로세서(1)가 요구하지 않은 메모리 장치(3) 내의 하나 또는 몇몇 어드레스로부터의 데어터도 페치하도록 배치되는 컴퓨터 시스템의 데이터 처리 속도를 증가시키는 방법에 있어서, 캐시(2, 4)와 상호 동작하기 위해 접속되는 스트림 검출(stream-detection)회로(5)를 구비하는데,

상기 스트림 검출 회로는

상기 마이크로프로세서(1)가 캐시(2, 4)로 요구하는 어드레스를 감지하고, 상기 요구되는 어드레스가 상기 캐시(2, 4) 내에 존재하는지의 여부를 레지스터(register)하는 단계와,

상기 마이크로프로세서(1)에 의해 요구되는 어드레스 들의 하나 또는 몇몇의 순차적인 열을 상기 캐시(2, 4) 내에서 검출하는 단계와,

상기 어드레스 열의 검출시에, 상기 캐시(2, 4)를 명령하여 상기 열의 다음 어드레스에 대응하는 데이터를 메모리 장치(3)로부터 페치하고 상기 어드레스를 캐시(2, 4) 내에 삽입하도록 하는 단계를 포함하는 절차를 수행하는 것임을 특징으로 하는 컴퓨터 시스템의 데이터 처리 속도 증가 방법.

청구항 2

제1항에 있어서,

상기 어드레스 열은 둘 이상의 어드레스로 구성되는 것인 컴퓨터 시스템의 데이터 처리 속도 증가 방법.

청구항 3

제1항 또는 제2항에 있어서,

상기 어드레스 열은 부가적, 산술적 또는 기하학적 또는 기타 논리적으로 구성되는 것인 컴퓨터 시스템의 데이터 처리 속도 증가 방법.

청구항 4

제1항 또는 제2항에 있어서,

프리페치되는 데이터의 양은 마이크로프로세서(1)가 메모리 장치(3)로부터 캐쉬(2, 4)로 막 수신된 데이터의 어드레스를 요구할 때까지 증가하는 것인 컴퓨터 시스템의 데이터 처리 속도 증가 방법.

청구항 5

제1항 또는 제2항에 있어서,

상기 스트림 검출 회로(5)는 상기 마이크로프로세서(1)가 캐시(2, 4)로 어드레스를 요구할 때, 상기 캐시(2, 4)를 명령하여 상기 스트림 검출 회로(5)가 어드레스 열의 증가 또는 감소 여부를 검출할 때까지, 인정된 어드레스 열내의 상위 및 하위 차순에 대응하는 데이터를 메모리 장치(3)로부터 페치하도록 배치되는 것인 컴퓨터 시스템의 데이터 처리 속도 증가 방법.

청구항 6

제3항에 있어서,

프리페치되는 데이터의 양은 마이크로프로세서(1)가 메모리 장치(3)로부터 캐시(2, 4)로 막 수신된 데이터의 어드레스를 요구할 때까지 증가하는 것인 컴퓨터 시스템의 데이터 처리 속도 증가 방법.

청구항 7

제3항에 있어서,

상기 스트림 검출 회로(5)는 상기 마이크로프로세서(1)가 캐시(2, 4)로 어드레스를 요구할 때, 그 캐시(2, 4)를 명령하여 상기 스트림 검출 회로(5)가 어드레스 열의 증가 또는 감소 여부를 검출할 때까지, 인정된 어드레스 열 내의 상위 및 하위 차순에 대응하는 데이터를 메모리 장치(3)로부터 페치하도록 배치되는 것인 컴퓨터 시스템의 데이터 처리 속도 증가 방법.

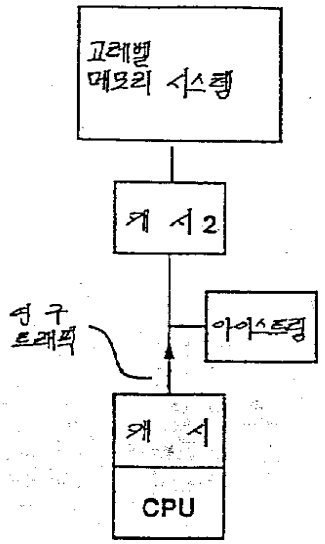
청구항 8

제4항에 있어서,

상기 스트림 검출 회로(5)는 상기 마이크로프로세서(1)가 캐시(2, 4)로 어드레스를 요구할 때, 그 캐시(2, 4)를 명령하여 상기 스트림 검출 회로(5)가 어드레스 열의 증가 또는 감소 여부를 검출할 때까지, 인정된 어드레스 열 내의 상위 및 하위 차순에 대응하는 데이터를 메모리 장치(3)로부터 페치하도록 배치되는 것인 컴퓨터 시스템의 데이터 처리 속도 증가 방법.

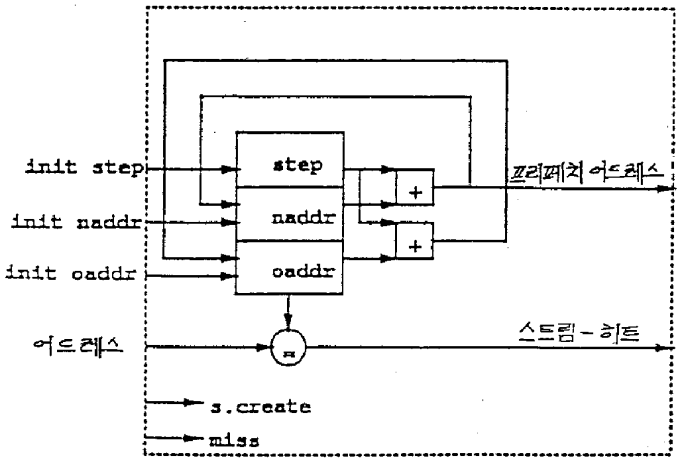
도면

도면1

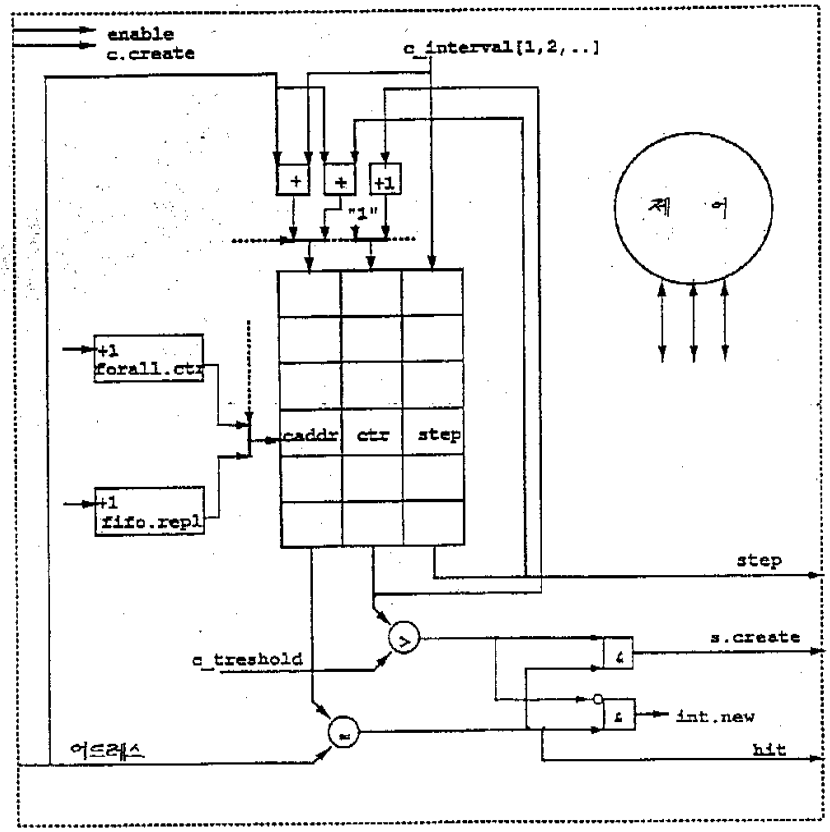


연구 트레픽		캐시2 레에서의 히트	프리페치 등작
판	득 A		
기	득 B	x	
판	득 A+1		판 득 A+2
판	득 A+2		판 득 A+3, 판 득 A+4
기	득 C		
판	득 D	x	
판	득 A+3	x	판 득 A+5
판	득 A+1	x	
기	득 C-1		기 득 C-2
판	득 A+4	x	판 득 A+6
기	득 C-2	x	기 득 C-3

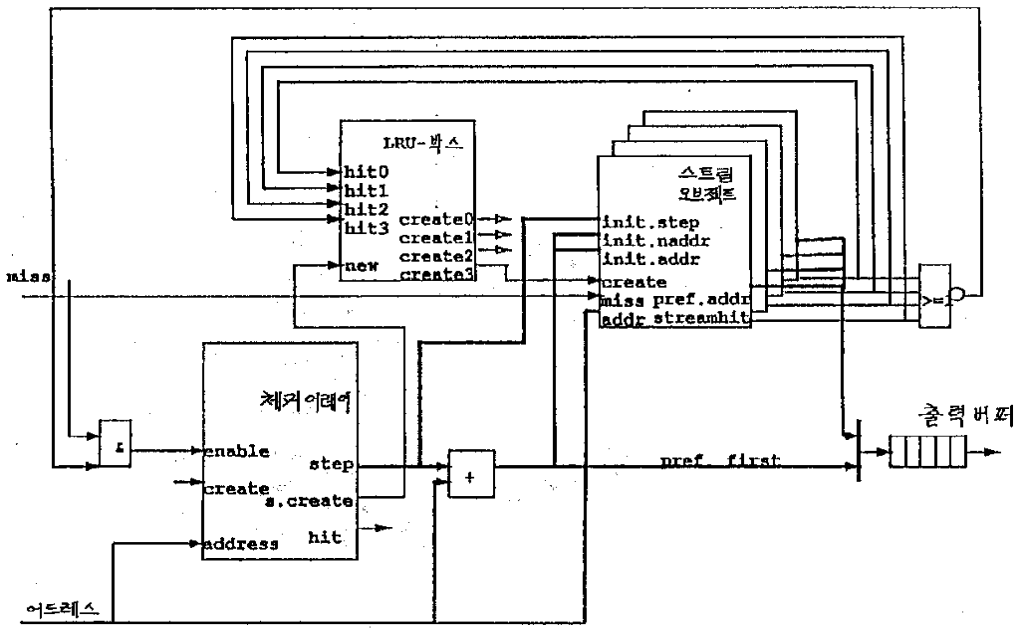
도면2



도면3


enable
c.create
c_interval[1,2,..]
+
+
+1
"1"
제 어
+1
forall.ctr
caddr
ctr
step
+1
fifo.repl
step
c_treshold
>
&
s.create
&
int.new
=
어드레스
hit

도면4


LRU-박스
hit0
hit1
hit2
hit3
create0
create1
create2
create3
new
스트림
오브젝트
init.step
init.naddr
init.addr
create
miss pref.addr
addr streamhit
miss
>=1
체커어레이
&
enable
step
create
s.create
address
hit
+
pref. first
출력버퍼
어드레스

도면5

대순화된 아이스트림(I-STREAM)의 개략설명

```
PARAMETERS:
c_treshold : How many c_hits are required to create a stream
c_interval : Array of interesting steps for streams (normally +1 and -1)

Checker Object, C(addr,step,ctr)
--------------------
addr: the address to check for
step : The interval between prefetches
ctr : How many hits have been recorded
OPERATIONS:
new_c(addr,step,ctr) : Reclaim and initiates a C object. Old C objects are
                       reclaimed by FIFO replacement.
create_c(A) :  for( i=1, i < sizeof(c_interval), i++)
               {new_c(A+c_interval[i],c_interval[i].1))%step = c_interval[i], ctr=1
c.snoop(A,op) :
               if (c.addr==A) {
                 if (c.ctr > c_treshold)
                    create_s(A.op.c.step)}              %create a stream
                 else
                   new_c(A+c.step,c.step,c.ctr+1);      %create a new c_obj
                 return true}                           %report a hit
               else {return false}

Stream Object, S(oaddr.naddr.step)
-----------------------------------
oaddr: The address prefetched the longest ago, (old-addr)
naddr: The address prefetched last (new-addr)
step : The step to take
OPERATIONS:
new_s(oaddr.naddr,step) : Reclaims and initiates an S object. Old S objects are
                          reclaimed by LRU replacement.
create_s(A,op,step):
               {prefetch(op,A+step);            %fetch the next one
                new_s(A+step,A+step,step)       %set up the stream
               }
s.snoop(A,op,cache-hit):
          if (s.oaddr == A){                    %is the stream successful
            if cache-hit (                      %is the prefetching done early enough
               prefetch(op, s.naddr + s.step); %fetch one more
               s.oaddr = s.oaddr + s.step;     %update this stream
               s.naddr = s.naddr + s.step
               }
            else (                              %increase the prefetching
               prefetch(op, s.naddr + s.step)   %fetch two
               prefetch(op, s.naddr + 2 * s.step)
               s.oaddr = s.oaddr + s.step       %update this stream
               s.naddr = s.naddr + 2 * s.step
               }
            return true                         %report hit
            }
          else {return false}

THE BUS SNOOPING:
-----------------
bus_snoop(A,op,cache-hit):
        forall S  {while !(snoop_hit=s.snoop(A,op,cache-hit))}
        if (!cache_hit && !snoop_hit) {
          forall C {while !(snoop_hit=c.snoop(A))
          if !snoop_hit {create_c(A)}
```