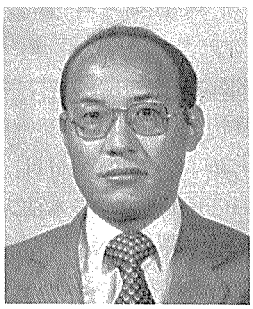

吳 吉 祿
韓國電子通信研究所
주전산기개발본부장 / 工博

MC68851을 이용한 Demand 페이지 관리 기억장치

1. 서 론

Unix V Rel 2.2는 커널에서의 기억장치 관리 부분을 완전히 새로운 설계를 구현하였다. 이 기억장치 관리 부분에서 크게 달라진 두가지 변화는 demand 페이지 기법의 도입과 호환성(portability)을 증가시킨 것이다.

Unix SVR 2.2 이전의 것들은 프로세스 스워핑 기억장치 관리 기법에 바탕을 두고 있다. 스워핑을 기반으로 하는 시스템에서는 큰 프로세스들의 수행이 종종 불가능할 때가 있다. 이것은 프로세스의 크기가 사용 가능한 실제 기억장치 크기에 제한을 받기 때문이다. SVR 2.2에서는 프로세스의 크기를 최대 주소 공간까지 확장해서 사용할 수 있게 한 demand 페이지 방식을 사용하고 있다.

그리고 SVR 2.2 이전에는 표준 기억장치 접속(standard memory management interface)를 정의하지 않고, 기억장치 관리 하드웨어에 접근할 수 있는 커널 기능만 구현하였다. 이로 인하여 Unix를 새로운 MMU에 이식하기가 힘들고, 커널에서 하드웨어와 관련된 부분을 분리시키기가 어렵게 되어 있었다 [4, 5].

그래서 SVR 2.2에서는 기억장치 관리부분과 다른 부분과의 접속을 기억장치 관리기(memory manager)에서 할 수 있게 하고, 이 부분을 커널에서 다른 부분과 분리시켰다. 즉 기억장치 관리기의 설계는 기억장치 관리장치(MMU) 하드웨어에 직접 접근하는 부분(기억장치 관리를 구현하기 위해서 사용되어지는 MMU 하드웨어의 동작, 페이징 알고리즘 및 자료 구조)을 커널의 다른 부분과 완전히 분리하여 행하게 하였다.

이러한 기능의 첨가로 Unix를 새로운 시스템

에 이식할 때 기억장치 관리부분에서 수정해야 할 양이 줄어들고 커널의 다른 부분과의 접속 관계도 고려할 필요가 없게 되었다. 특히 최근에는 페이지 기억장치 관리방법을 효율적으로 구현할 수 있게 한 칩(PMMU : paged memory management unit)들을 여러 반도체 회사에서 만들어내고 있어서, 이를 더욱 쉽게 해 주고 있다. Motorola사의 MC68851, National Semiconductor사의 NS32082, NS32382 등이 이러한 칩들이다.

본 논문에서는 이들 중에서 Motorola 사의 MC68851을 이용한 demand 페이지 기억장치 관리의 구현에 관해서 기술한다. 이미 MC68851의 subset인 MMB851을 이용해서 구현한 시스템은 이미 나와 있지만, 이것은 기억장치 관리 시스템(MMU) 설계에 상당한 제약이 있어서 Unix 시스템에서 요구하는 기능을 충분히 지원해 주지 못하고 있다[2]. 그리고 MC68851을 이용한 MMU 설계는 기존의 MMB851을 이용한 것보다 전체 구조가 간단해지며, Unix 시스템 V의 region 개념을 쉽게 적용시킬 수 있다[6,7].

Unix 시스템 커널에 대한 상세한 내용은 이미 여러 문헌을 통해서 잘 알려져 있기 때문에 여기서는 생략하기로 하고, 다만 MMU 설계에 따른 여러 고려사항 및 기본 구조에 관해서만 기술하기로 한다. 우선 Unix 시스템 V의 MMU의 기본 구조 및 region에 관해서 알아 보고, MC68851의 구조 및 주소의 변환 과정에 대해서 간략하게 기술한다. 그리고 이 PMMU를 이용한 demand 페이지 가상 기억장치의 설계에 관해서 기술한다.

2. Unix 시스템 V의 MMU

MMU의 기능은 기억장치의 초기화, 기억장치 할당 및 회수, 페이지 관리, 그리고 프로세스 region 관리로 나눌 수 있다. 기억장치 초기화에서는 주기억장치 내에서의 커널위치 및 각종 표들의 위치와 크기를 정의해 주고, 시스템 페이지 표를 초기화시켜 주게 된다.

시스템이나 프로세스에 의해서 요구되는 기억장치의 할당 및 회수는 페이지 단위로 이루어지며, 시스템은 사용가능한 기억장치의 용량을 적정 수준이상으로 항상 유지시키기 위해서 page-out daemon을 사용한다. 이것은 일정 시간마다 사용 가능한 기억장치의 양을 검사하여 적정 수준이하로 내려가면 수행되는 것으로, 각 사용자 프로세스의 페이지들 중에서 일정 시간동안 사용되지 않은 모든 페이지에 대해서 page-out을 시킨다.

프로세스의 주소 공간을 관리하기 위해서 region 개념을 사용하고 있다. 이것은 커널에서 기억장치의 경영을 독립시키므로서, MMU 하드웨어가 상이한 시스템에서도 Unix의 이식을 용이하게 한다. Region은 논리 주소 공간에서 연속적으로 존재하는 기억장치의 덩어리로서, 커널의 루틴들이 기억장치에 접근할 때 사용되는 논리적인 단위이다. 프로세스의 각 기억장치 세그먼트마다 별개의 region이 존재하며, 커널에서의 주소 변환은 이들의 기본 단위에서 이루어진다. 프로세스가 가질 수 있는 region의 종류는 text, data, stack 그리고 shared memory가 있다.

프로세스에 관련된 기억장치 관리는 곧 해당 프로세스의 region 관리를 의미하며, 실제 MMU 중 반가량이 이에 관련된 것이다. 그리고 나머지는 페이지 관리를 위한 부분으로 region의 마지막 표인 페이지 표와 직접적인 연관을 가지게 된다. 프로세스의 기억장치의 할당 및 회수, 확장 및 축소 등을 region에서 행하므로해서, 기억장치 관리를 실제 시스템의 하드웨어와 분리가 가능하다. 단지 페이지표 관리 측면에서만 실질적인 하드웨어와 연관성을 가지게 된다. 이로 인해서 서로 다른 하드웨어에 Unix의 이식이 용이하게 되어 있다.

3. MC68851 Architecture

가. PMMU Overview

MC68851은 MC68020 32-bit 마이크로 프로세서에서 demand 페이지 가상 기억장치 관리 환경을 효율적으로 지원하기 위해서 설계된

high-performance paged virtual memory management unit (PMMU)이다[2].

MC68851에서 제공하는 특성들은 다음과 같다.

- fast logical-to-physical address translation
- 32-bit logical and physical addresses
- Eight available page sizes ranging from 256 to 32K bytes
- fully associative 64-entry address translation cache (ATC)
- address translation cache support for multi-tasking
- H/W maintenance of external translation tables and cache
- MC68020 instruction set extention
- hierarchical protection mechanism with upto 8 level
- instruction breakpoints for S/W debug and program control
- support for logical and/or physical data cache
- support for multiple logical and/or physical bus masters

그리고 MC68851 programming model은

- three 64-bit root point registers (각64bits)
- a 32-bit translation control register (TC)
- 64-entry address translation cache (ATC)
- a 16-bit cache status register (PCSR)
- a 16-bit status register (PCSR)
- three 8-bit protection control registers (CAL VAL SCC)
- a 16-bit access control register (AC)
- eight 16-bit breakpoint acknowledge data registers
- eight 16-bit breakpoint acknowledge control registers

나. 주소변환(Address Translation)

MC68851은 CPU에서의 S/W 지원없이 4G bytes까지의 논리 주소 공간에서의 logical-to-physical 주소 변환을 지원한다. 최근 사용한 logical-to-physical 변환 주소를 저장하는 고속력 기억장치를, 64개의 ATC(address translation cache), 사용해서 직접 논리 주소를 실제 주소로 변환시켜 준다. Logical bus master의 버스 사이클동안 해당 논리 주소가 ATC에 있으면 ATC에 저장되어 있던 실제 주소가 그대로 실제 주소 버스에 전달되고, 없을 때에는 변환표들을 이용해서 실제 주소를 찾아내어 ATC에 저장해 주고 버스사이클을 다시 수행해서 실제 주소를 만들어 주게 된다.

주소 전환표들은 트리 형태로 구성되어 있으며, 이들 표들은 실제 기억장치에 위치한다. 각 표들은 포인트들로 구성되어 있으며, 이들 포인터들은 다른 표를 지정하거나 실제 페이지를 지정하게 된다. 각 표들의 포인터들이 표 지정자(table descriptor)인가 페이지 지정자(page descriptor)인가에 따라서 포인터표(pointer table) 혹은 페이지표(page table)로 나누어져 있다. MC68851은 동시에 active한 3개의 트리가 있고, 각 트리의 루트(root)로서는 슈퍼바이저, 사용자 그리고 DMA access를 위해서 SRP, CRP, 그리고 DRP가 사용되어진다.

변환 트리(translation tree)의 일반적인 구조는 TC(translation contol) 레지스터에 의해서 결정되어진다. 그림 1은 TC의 구조이다.

2 bits	12bits	8 bits	10bits
region selection	segment offset	page offsct	byte offset

그림 1. Translation Control Register

IS(initial shift) 필드는 논리 주소공간의 크기를 결정하는데 사용되고, PS(page size) 필드는 시스템에서 사용되는 페이지 크기를 결정한다. 그리고 table index field(TIA, TIB, TIC, TID)는 변환 트리의 각 레벨에서 사용되는 논리 주소의 비트 수를 나타내고, 각 레벨에서 논리 주소 공간의 분할을 규정한다. Logical-to-physical 변환을 위한 변환 트리는 5-레벨까지 사용가능하며, 레벨을 줄일려면 TID부터 역으로 0을 채워나오면 된다. 하지만 TC레지스터의 각 필드 합은 32가 되어야 한다.

CRP는 현재 사용자 모드에서의 변환 트리의 루트로서, OS가 새로운 task를 dispatch하기 전에 해당 task를 위한 변환 트리의 루트로 바꾸어 주어야 한다. 그리고 MC68851 내부에는 최근 사용한 8개까지의 CRPs를 저장할 수 있어서, 8개까지의 사용자 주소 공간을 가질 수 있다. 하지만 특정한 시점에서는 항상 하나만의 CRP가 선택되어 사용되어진다. 이를 위해서 ATC의 각 엔트리는 tag 필드(TA)를 가지고 있으며, 이 TA는 각 task에 대한 alias 값을 가지고 있어서 해당 변환표를 찾아갈 수 있다.

SRP와 DRP도 CRP와 기능이 동일하며 단지 각 하나만 존재한다. SRP의 경우는 모든 task에 대해서 동일하게 사용할 수 있어서 Unix에서의 슈퍼바이저 모드로 사용이 가능하게 되어 있다.

지정자 (descriptor)는 표지정자와 페이지 지정자로 나누어지고, 이들 각각은 64 bits의 long format이나 32 bits의 short format일 수 있다. 표 지정자는 다음 표의 주소를 가지고, 그 표의 인덱스의 위나 아래의 제한점을 둘 수 있다. 페이지 지정자는 실제 페이지 프레임의 주소와 lock, modified, used bit 등을 가진다.

RPT(root pointer)와 표 지정자 그리고 페이지 지정자를 이용하여 논리 주소를 실제 주소로 변환하는 과정은 그림 2와 같다. FNC(function code)에 의해서 루트 포이터가 결정되어지고, 루트 포인트에 있는 주소로 첫번째 변환표를 찾아간다. 32 bit 논리 주소중 IS 필드는 사용하지 않고, TIA를 이용해서 해당 지정자를 찾아서 그 다음 변환표의 주소를 찾아낸다.

TIB, TIC에 대해서도 이 과정을 계속해서 페이지 테이블을 찾게되고, TID를 이용해서 페이지 테이블에서 페이지 지정자를 찾으면 실제 페이지 프레임의 주소를 구할 수 있다. 이 주소와 PS 값을 더하면 논리 주소에 해당되는 실제 주소가 구해지게 된다. ATC에 저장한 후 버스 사이클을 다시 수행하면, 논리 주소에서 실제 주소로의 변환이 일어나서 실제 기억장치에 접근할 수 있다.

4. MMU 설계

가. 주소 공간

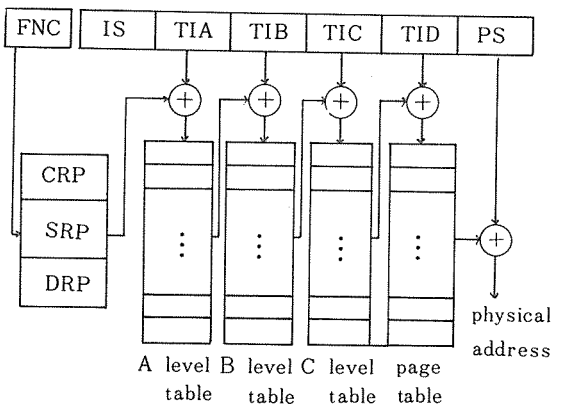

그림 2. 주소 변환 과정

주소 공간은 시스템 주소 공간과 사용자 주소 공간으로 나누고, 시스템과 각 사용자는 서로 다른 4G 바이트의 가상 주소 공간을 가질 수 있다. 즉 시스템이나 각 사용자는 자신의 가상 주소 공간에 접근하기 위해서 32비트까지 사용이 가능하다. 이것은 MC68851의 지원으로 가능하다. 하지만 일반적으로 Unix V를 사용하는 시스템의 주소 공간은 VAX/750 경우, 4G 바이트 모두 사용하지 않고, 실제 기억장치의 2배정도 사용하고 있다. 그리고 사용자 주소의 경우는 Unix V에서 제공하는 region을 구현하기 위해서 여러단계의 변환 과정을 거쳐야 하지만, 시스템 주소의 경우는 시스템 페이지 표만 관리해 주면 된다. 그래서 사용자 모드와 시스템 모드의 주소 변환 방법이 달리 적용되어야 한다.

시스템 주소 공간의 크기는 실제 기억장치의 n배로 하고, 1K바이트 단위의 페이지로 나누어 접근할 수 있게 하였다. 에드레싱 비트와 주소 공간과의 관계는 그림 3과 같다. M+N+10은 항상 32가 되어야 하며, M과 N은 tunable factor로서 시스템에 이식할 때 조정될 수 있는 것으로, 시스템 성능에도 영향을 주게 된다. 그리고 N은 시스템 페이지표의 크기를 결정하는 요소가 된다.

IS(M bits)	N bits	10 bits
initial shift	page offset	byte offset

그림 3. 시스템 모드에서의 논리 주소 비트형태

시스템 페이지 표는 실제 기억장치 페이지 수의 2배 만큼의 엔트리를 가지며, 실제 기억장치에 상주하게 되고 시스템이 부팅될 때 커널 바로 아래 위치하게 된다. 각 엔트리는 8 바이트로 되어 있으며, 각 엔트리들의 포맷은 경우에 따라 달라질 수 있다. 이 표의 엔트리가 시스템 주소 공간의 실제 페이지를 지정하는 경우와 각 사용자 프로세스의 세그먼트 표의 엔트리로 사용되어지는 경우가 있다. 시스템 페이지 표의 엔트리가 각 사용자 프로세스의 세그먼트 엔트리가 될 때 그 엔트리는 다시 사용자 페이지 표를 지정하는 표 지정자가 된다.

사용자 공간은 텍스터, 데이타, 스텍, 공유 데이타 region으로 나뉘어지고, 각 region은 각각 1G 바이트씩의 가상 공간이 할당되어진다. 또 각 region은 256K바이트의 세그먼트로 나뉘어지며, 세그먼트는 256개의 페이지로 구성되어진다. 그림 4는 사용자 가상 주소의 필드와 가상 주소 공간과의 관계를 나타낸다.

2 bits	12 bits	8 bits	10 bits
region selection	segment offset	page offset	byte offset

그림 4. 사용자 모드에서의 논리 주소 비트형태

시스템 모드와 사용자 모드의 주소 bit-pattern이 달리 할 수 있는 것은 MC68851의 TC 레지스터의 변환이 가능하기 때문이다. 즉 사용자 모드에서 시스템 모드로 바뀌거나 그 반대의 경우에 해당 bit-pattern에 맞게 TC 레지스터의 내용을 변경시켜 주면 된다. 이를 위해서 커널 중에서 시스템 모드에서 사용자 모드로 들어가는 부분과 그 반대의 경우의 부분을 찾아서 TC 레지스터를 변환시키는 작업을 수행해야 한다.

나. 주소 변환

MC68020에서는 function code를 이용하여 슈퍼 바이즈와 사용자 모드로 나눌 수 있고, 각각은 다시 텍스터와 데이타로 분리될 수 있으며, 이들 모든 경우에 독자적인 4G 바이트까지의 가상 주소 공간을 가질 수 있다. 시스템 모드(슈퍼 바이즈 mode)에서의 주소 공간의 크기는 성능에 상당한 영향을 주기 때문에(시스템 페이지 표의 크기가 비례로 증가하기 때문에) 크기를 제한하는 것이 필요하게 된다. 따라서 시스템 가상 주소 공간은 앞에서 언급했듯이 4G 바이트를 전부 사용하지 않고, 실제 기억장치의 2배 정도만 사용한다. 그래서 사용자 모드보다도 주소 변환에 따른 표도 간단하고, 그 과정도 간단하게 된다. 물론 이것은 MC68851에서 사용자가 변환과정의 제어를 허용하고 있기 때문이다. 그리고 시스템 모드의 경우는 텍스터와 데이타를 구분하지 않는다.

사용자 모드에서는 텍스트나 데이타의 구분없이 한 프로세스당 4G 바이트의 가상 주소 공간을 사용한다. 따라서 function code에 의해서 이들을 구분하지 않고, 앞에서 언급했듯이 주소의 첫 두 비트로 텍스터 데이타 스택 및 공유 데이타 부분으로 나누어준다. 그러면 이들 두 모드에서의 변환 과정에 관해서 알아 보기로 한다.

1) 시스템 모드

시스템 모드에서는 주소 변환을 위하여 하나의 페이지 표 즉, 시스템 페이지 표만 필요하게 된다. 그리고 주소 변환을 위해서 MC68851의 TC register에서 IS, TIA, PS, E, SRE 필드만 이용하게 된다. 시스템 모드에서 가상 주소 공간을 16M bytes로 가정하고, 페이지 크기를 1K 바이트로 할 때 TC 레지스터의 각 필드 값은 E, SRE는 각각 1이 되고, IS는 8 (unused bits), PS는 10(1 page : 1K), 그리고 TIA는 14가 된다. 그리고 TIB, TIC, TID는 모두 0이 된다. 따라서 시스템 모드에서의 변환 과정은 1 레벨 변환으로 가능하게 되고, 전체 시스템 페이지 표의 크기는 128K 바이트가 된다. TC 레지스터의 내용은 사용자 모드에서 시스템 모드로 바뀔 때 앞에서 지정한 값으로 바꾸어 주게 된다.

주소 변환 과정은 그림 5와 같다. 주어진 가상 주소에 대해서 페이지 폴트가 발생하면, MC

68851의 SRP를 이용해서 시스템 페이지 표를 찾아가서, 가상 주소의 offset(23 : 10)을 index로 해당 페이지 엔트리를 찾아가면 실제 페이지 프레임의 주소를 찾을 수 있다. 이 가상 및 실제 주소를 ATC에 넣고, 버스 사이클을 다시 수행하면 원하는 기억장치에 접근할 수 있다. 여기서의 페이지 폴트는 커널에서 말하는 것과 의미가 다른 하드웨어적인 것을 말한다.

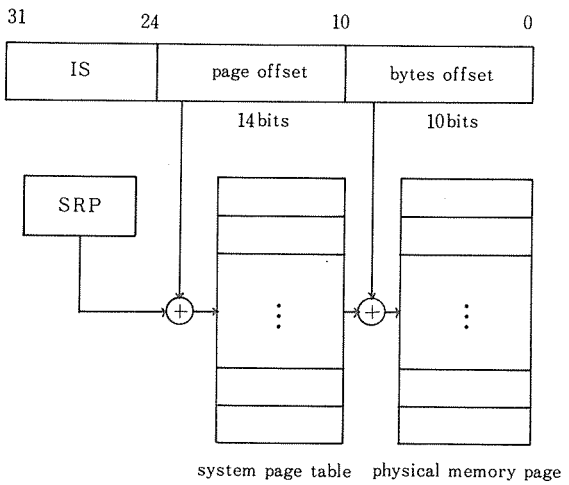

그림 5. 시스템 모드에서의 주소 변환 과정

시스템 페이지 표의 일부는 사용자 모드의 세그먼트 표로 사용되어지는데, 이 경우는 해당 엔트리의 포맷이 달라지게 된다. 여기에 대해서는 사용자 모드에서 상세하게 기술하기로 한다.

2) 사용자 모드

모든 프로세스는 사용자 모드에서 독자적인 4G 바이트의 가상 주소 공간을 가진다. 가상 주소는 32 비트를 사용하고, 3 레벨의 주소 변환 과정을 거친다. 주소 변환을 위해서 필요한 표로써는 region 지정자 표, 세그먼트 표, 그리고 페이지 표가 있게 된다. 이들 각 표의 크기는 TC 레지스터에 정의되어진다. TC 레지스터의 E, SRE는 1이 되고, FCL은 0, PS는 10, IS는 0, TIA는 2, TIB는 12, TIC는 8이고 TID는 0이 된다.

사용자 모드에서 논리 주소가 발생하면 그 주소는 직접 PMMU (MC68851)의 ATC 엔트리와 비교해서 존재 유무를 조사한다. 이 때 function code도 논리 주소와 함께 들어가서 논리 주소와 동시에 비교하게 된다. ATC에 해당 엔트리가 존재하면 실제 주소를 PMMU의 실제 주소 버스에 실어 주게 되고, 이것으로 실제 기억장치에 접근할 수 있게 된다.

ATC에 해당 엔트리가 없을 때는 두가지 경우로 실제 기억장치에 있지만 ATC에 등재되지 않은 경우와 실제 기억장치에 없는 경우이다. 첫번째 경우는 PMMU가 자동적으로 해당 표를 찾아 가면서 실제 주소를 구해서 ATC에 등재시켜 준다. 하지만 두번째 경우는 첫번째 과정 중 해당 페이지가 없거나 invalid한 경우 PMMU에 버스 에러가 발생하여 시스템에게 페이지 폴트임을 알려 준다. 그러면 첫번째 경우에 대해서 먼저 기술하고, 이를 바탕으로 두번째 경우를 설명하기로 한다.

사용자 모드에서의 논리 주소가 ATC에 없으면 PMMU 내에 있는 현재 CRP를 이용해서 region 지정자표를 찾아가서, 논리 주소의 REG 필드의 값으로 해당 엔트리를 찾아 낸다. 그 엔트리는 세그먼트 표의 소재를 알 수 있게 한다. 논리 주소의 세그먼트 옵셋으로 세그먼트 표의 엔트리를 찾아서 그 다음에 있는 페이지 표를 찾아 간다. 페이지 표에는 논리 주소의 페이지에 해당하는 페이지의 실제 페이지 주소가 있게 된다. 이렇게 실제 페이지 주소를 찾으면 그 주소를 PMMU의 ATC에 넣어 주고, 새로운 버스 사이클을 수행시켜 실제 기억장치에 접근하게 된다. 이 과정을 거치는 동안 해당 엔트리가 없거나 invalid하면 PMMU는 버스 에러를 발생시켜 시스템에게 페이지 폴트임을 알려 준다.

페이지 폴트가 발생하면 사용자 모드에서 시스템 모드로 변환되어 관련작업을 수행해 주게 된다. 그림 1에서 처럼 각 사용자의 프로세스 표에서부터 시작해서 region 표를 거쳐 해당 페이지가 실제 있는 디스크에서의 위치에 관한 정보를 DBD 표에서 찾아내어 실제 기억장치에 올려 주게 된다. 이에 관해서 상세한 설명은 생략하고 단지 사용자 모드에서의 세그먼트 표에 관해서만 기술하기로 한다.

프로세스가 생성되면서 그 프로세스의 주소

공간도 동시에 만들어지는데, 이것은 시스템 모드에서 해당 프로세스의 region을 만들어줌으로써 가능하다[6, 7]. 이에 대한 상세한 기술은 생략한다. 각 프로세스의 region 표나 페이지 표는 프로세스 주소 공간을 관리하기 위해서 필요한 것으로 프로세스가 수행중에 발생하는 사용자 주소 공간에서의 논리 주소는 이들 표를 통해서 실제 주소로 변환하지 않고, 하드웨어에서 볼 수 있는 표를 다시 만들어 주게 되는데 이것이 바로 세그먼트 표가 된다. 이 표를 효율적으로 관리하기 위해서 시스템 페이지 표 내에 만들어 사용하게 된다. 그리고 한 프로세스에서 변환표를 위한 기억장치의 크기는 (즉 세그먼트 표나 페이지 표를 위한 기억장치) 다이나믹하게 변하면서 항상 최소 크기로 유지시킨다. 세그먼트 표는 시스템 페이지 표의 일부를 할당받아 사용방법은 VAX SVR2.2와 유사하게 했다.

그림 6은 MC68851의 ATC에 원하는 페이지의 실제 주소가 없을 때 거치는 주소 변환 과정이다. 페이지 표에서 해당 엔트리에서 실제 주소를 찾아 ATC에 넣어 주고, 버스 사이클을 다시 수행하면 원하는 기억장치에 접근할 수 있다.

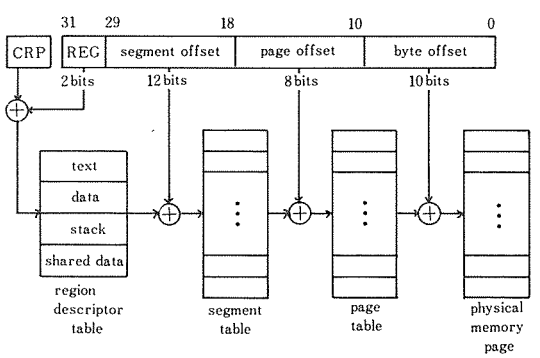

그림 6. 사용자 모드에서의 주소 변환 과정

5. 결 론

MC68020 마이크로 프로세스를 기반으로 하는 시스템에 demand 페이지 가상 기억장치를 구현하였다. 이를 위해서 페이지 기억장치를 지원해 주는 MC68851을 사용하였다. 이것은 MMB 851 사용에 비해서 상당히 flexible하고, MMU의 구조도 간단해지며 region 개념도 쉽게 적용시킬 수 있다[6, 7].

MMU 설계는 가능하면 원래 Unix의 MMU 기본 정책은 변화시키지 않고, 좋은 효율을 올릴 수 있는 방향으로 했다. 하지만 주소 변환에서 변환표 관리 및 실제 기억장치의 효율적인 활용에 대해서 더 연구되어야 한다.

새로이 설계한 MMU는 다중 프로세스에서 운용될 수 있게 semaphore 기능을 첨가할 예정이고, 또 다중 프로세스에서의 효율성을 높이기 위해서 tunning 작업도 행해야 한다[8].

참 고 문 헌

1. Motorola Inc., MC68020 32-bit Microprocessor User's Manual, Prentice-Hall, 1984.
2. Motorola Inc., MC68851 Paged Memory Management Unit User's Manual, Prentice-Hall, 1986.
3. Motorola Inc. M68KVMMB851 Memory Management Board User's Manual, Feb. 1985.
4. Maurice J. Bach, The Design of the UNIX Operating System, Prentice-Hall, 1986.
5. R. S. Jung, "Porting the AT&T Demand Paged UNIX Implementation to Microcomputers," USENIX Proc. pp. 361-370, Summer 1985.
6. 과학기술처, 32-bit UNIX 컴퓨터 시스템 개발에 관한 연구, 과학기술처 1985.
7. 최효진, "UNIX에서 MMB851을 이용한 Virtual Memory Management의 구현," 정보 과학회, 86 봄 학술 발표 논문집, 제 13권 1호, pp. 223-230.
8. 과학기술처, Multiprocessor 컴퓨터 시스템에 관한 연구, 과학기술처, 1987. 4.