9장 메모리 관리(2) 과제

1731017 김민형

1. 페이징에서 논리주소는 페이지번호와 옵셋으로 구성된다.
한 페이지 크기가 4KB인 경우와 8KB인 경우, 32비트의 논리주소의 구성에 대해 설명하라.

32비트는 가상주소 공간의 크기가 4GB이다.

2^{32}에서 2^{12}(4KB)를 나눈값인 2^{20}개는 한 페이지의 크기가 4KB 인 경우에 페이지의 총 개수가 된다. 32bit 중에서 페이지번호는 20bit가 되며, 옵셋은 12비트가 된다.

2^{32}에서 2^{13}(8KB)를 나눈값인 2^{19}개는 8KB인 경우에 페이지의 총 개수가 된다. 32bit 중에서 페이지번호는 19 bit가 되며, 옵셋은 13 bit가 된다.

2. 다음 C 프로그램이 실행되는 동안

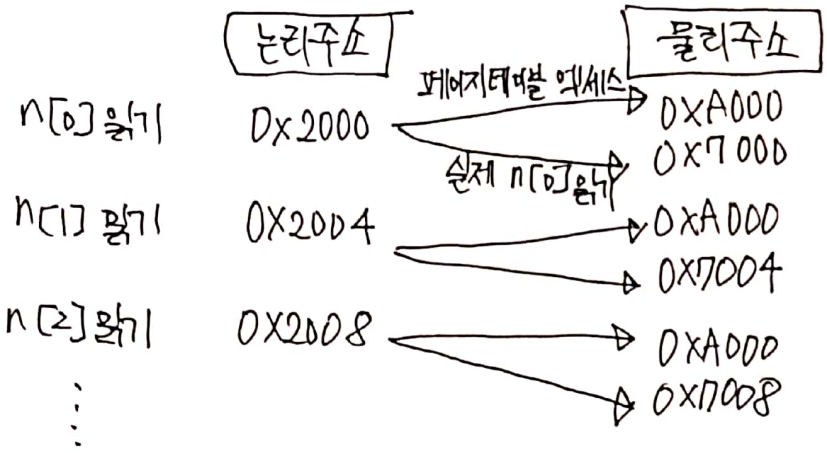
```
int n[1500];
int sum=0;
for(i=0; i<1500; i++)
    sum+=n[i]; //여기서 n[i] 엑세스
```

- 32비트 CPU에서 실행되고, 페이지의 크기는 4KB이다.
- int 는 4바이트이다.
- 배열 n의 논리주소는 0x2000 부터 시작한다.
- 배열 n의 물리주소는 학생이 알아서 마음대로 결정하라.
- 페이지 테이블은 0xA000 에서 부터 시작된다.

(1) n[i]를 엑세스를 위해 물리 메모리는 총 몇번 엑세스되는가? 그림을 그리고 설명하라.

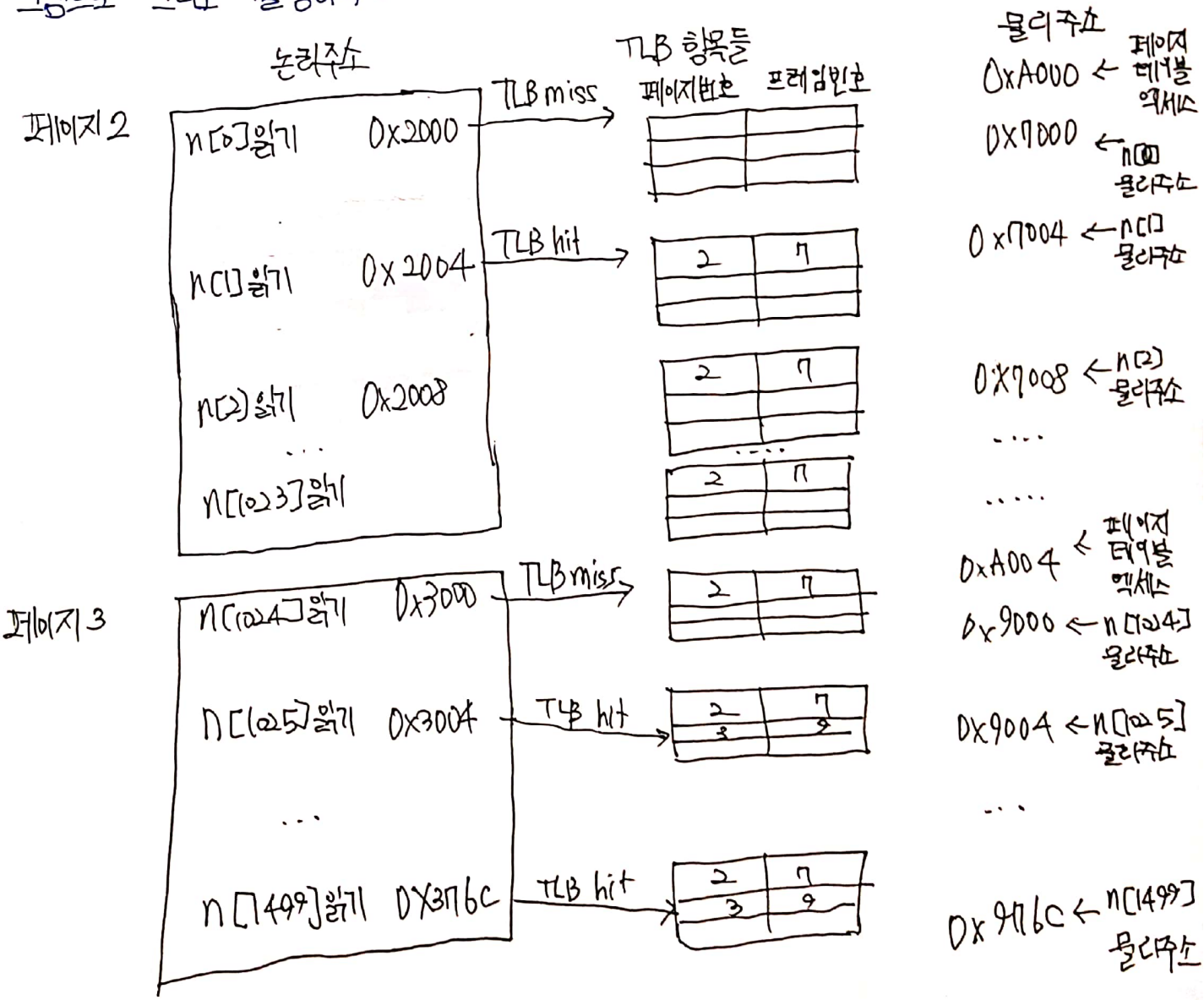
물리메모리는 총 3000번 엑세스 된다.

그림은 아래와 같다.



논리주소에서 물리주소로 바꾸는 MMU에서 페이지 테이블의 물리주소를 가지러 갈 때 한번 엑세스하게 되고, 물리 주소를 이용하여 데이터를 찾으러 갈 때 한 번 엑세스하게 되어 총 두 번 엑세스하게 된다. 따라서 총 물리메모리는 1500 x 2 = 3000번 엑세스하게 된다.

(나) TLB가 있는경우 물리 메모리는 총 몇번 엑세스되는가?
그림으로 그리고 설명하라.



MMU가 논리주소를 물리주소로 변환해줄 때, 페이지번호를 TLB 캐시 내에 있는 모든 페이지 번호들과 비교한다. 그런데 처음으로 실행할 때에는 기존에 저장된 데이터가 없기 때문에 TLB miss가 일어난다. 따라서 (1)번과 동일하게 한번 실행할 때 마다 2번 엑세스하게 된다. 그러나 한번 TLB miss가 일어난 이후에는 miss한 페이지 번호와 프레임 번호를 TLB에 저장한다. 따라서 그 다음 실행이 될 때는 TLB에 일치하는 항목을 찾아 TLB hit가 발생한다. 그 결과 바로 TLB의 프레임 번호로 물리주소를 만들어 엑세스하게 되어 물리메모리에 직접적인 엑세스를 하지 않아 총 한번의 엑세스만 일어난다.
여기서 한 페이지의 크기는 4kB 이기 때문에 6000byte를 엑세스 하려는 위 프로그램의 경우는 2개의 페이지로 되어있다. 따라서 n[0]~n[1023]까지는 TLB hit가 되지만 n[1024]에서는 다른 페이지이기 때문에 TLB miss가 일어나 물리메모리를 2번 엑세스한다. n[1025]~n[1499]까지는 다시 TLB hit 가 되어서 물리메모리를 1번 엑세스한다. 정리하면, 총 2*2+1498씨 =1502번 물리메모리에 엑세스하게 된다.

(3) TLB의 성공과 참조의 지역성의 관계에 대해 설명하라.

참조의 지역성은 프로그램이 실행되는 동안 짧은 시간 범위 내에 일정 구간의 메모리 영역을 반복적으로 엑세스하는 경향을 말한다. 참조의 지역성은 공간에서도 나타나는데 이때 TLB는 이를 활용하여 지금 엑세스 되는 메모리의 주변 번지들이 가까운 미래에 엑세스될 확률이 매우 높게 하기 위해 엑세스한 메모리 번지를 기억하도록 한다. 이는 동일 페이지의 코드, 데이터에 접근할 경우 페이지 테이블 엑세스 횟수를 줄이고 실행 성능을 개선하도록 도와준다.

(4) TLB의 성능과 메모리 엑세스 패턴인 순차 엑세스와 랜덤 엑세스의 관계에 대해서 설명하라.

배열이나 반복문 엑세스와 같이 순차적으로 메모리에 엑세스 하는 경우 같은 페이지의 코드나 데이터를 엑세스하기 때문에 TLB hit가 계속되어서 페이지 테이블을 엑세스하는 횟수가 줄고 프로그램의 실행 속도는 개선된다. 반면, 참조의 지역성이 무시되는 랜덤 메모리 엑세스의 경우, TLB에 저장할 수 있는 항목의 개수가 작고 제한적이기 때문에 TLB miss가 자주 발생하고 페이지 테이블을 엑세스하는 횟수 또한 증가한다. 따라서 랜덤으로 메모리 엑세스하는 프로그램의 경우에는 TLB 사용으로 얻는 프로그램 실행 속도 향상은 크지 않다.

3. 페이징에서 페이지 테이블이 낭비되는 문제를 해결하기 위한 방법 2개를 간단히 설명하라.

페이징에서 페이지 테이블의 일부 항목만 사용해서 발생하는 메모리 낭비를 줄이기 위해 역페이지 테이블 방법과 멀티레벨 페이지 테이블 방법을 사용한다.
먼저, 역 페이지 테이블 방법의 경우를 보자. 이전까지는 페이지 테이블이 페이지별로 할당된 프레임 번호를 사용하였지만, 이제는 거꾸로 모든 프레임에 대해 각 프레임이 어떤 프로세스의 어떤 페이지에 할당되었는지를 나타내어서 페이지 테이블의 한 항목 크기를 줄이도록 한다. 예를 들어서 32bit 주소 체계에서 페이지의 크기가 4MB인 경우, 기존 페이지 테이블은 프로세스 당시 4MB의 크기 이므로 10개의 프로세스가 실행되고 있다면 40MB 의 메모리가 페이지 테이블을 위해 사용된다. 하지만 만약 역 페이지 테이블을 사용하면, 역페이지 테이블의 크기는 2^{20}(총 프레임의 개수) x 8 byte(한 항목의 크기) = 8MB이다.
따라서 역페이지 테이블의 크기는 프로세스의 개수와 무관하므로 프로세스의 개수가 늘어나면 더욱 효과적이다.

그 다음으로는 멀티 레벨 페이지 테이블 방식이다. 이는 작은 페이지 테이블을 여러 단계로 수직적으로 구성하는 방법이다. 이와 같은 방식은 프로세스가 현재 사용 중인 페이지들에 대해서만 페이지 테이블이 구성되도록 하여 페이지 테이블의 낭비를 줄인다.

예를 들어서 2-레벨 테이블의 경우, 논리주소의 페이지 번호 부분을 10비트씩 2개의 레벨로 나눈다. 그리고 page directory 에서 page directory index를 이용하여 저장된 여러 개의 페이지 테이블 중에 하나를 고른다. 그리고 page directory index를 이용하여 물리 메모리 프레임 번호를 알아낸다.

2-레벨 페이지 테이블을 사용하는 경우 소모되는 메모리 량을 계산해보자. 만약 최대 1024개의 페이지가 있고, 1개의 프레임의 크기는 4KB로 가정하면 페이지 테이블 전체 용량은 1024 * 4KB 이고 page directory 인 4KB을 더하여 총 소모되는 메모리는 4MB + 4KB 이다. 하지만, 프로세스는 1024개의 페이지을 모두 사용하지 않고, 실제 사용되는 페이지의 개수가 100개라고 한다면 총 소모되는 메모리는 4KB + 100 x 4KB = 404 KB가 된다. 즉 90% 수준으로 페이지 테이블 공간을 줄일 수 있게 된다.