운영체제 9-1장 과제

1731011 김민경.

Q. 메모리 계층구조의 목적 즉 어떤 성능이 향상되는가?
그리고 메모리 계층 구조와 참조의 지역성이 어떤 관계인가?

메모리 계층 구조의 목적은 CPU의 메모리 엑세스 속도를 높이기 위해서이다.
메모리 계층 구조가 없다면 CPU와 보조기억장치의 처리 속도 차이로 인해 문제가 발생한다. CPU는 작업을 위해 I/O 장치에서 데이터를 가져와야하는데 CPU에 비해 속도가 느린 I/O로 인해 CPU가 대기하는 현상이 발생한다. 이는 작업의 처리 시간이 많이 걸리고 응답시간이 증가하는 악영향을 끼칠 수 있다. 따라서 이와 같은 처리 속도의 병목현상을 완화하여 CPU의 처리 효율을 증대시키고자 하였다. 따라서 당장 실행할 데이터를 가지고 있는 CPU 레지스터, 가격이 높아 용량이 작은 대신 바른 캐시 메모리, 메인 메모리, 그리고 가격이 싸고 용량이 큰 보조기억장치로 이루어진 계층 구조를 두어 메모리 엑세스 속도를 높였다. 그러나 캐시 메모리는 크기가 작기 때문에 당장 실행할 프로그램의 코드와 데이터 일부분만 들 수 밖에 없는데, 코드를 다 실행하고 캐시 메모리 데이터를 교체하는 과정에서 CPU의 대기 시간이 길어져 결국 효율이 떨어질 수 있다. 이와 같은 문제를 해결하는 것이 참조의 지역성이다.
메모리는 CPU가 참조하는 데이터들을 저장하는 매체이다. 시스템이 필요로 하는 데이터들은 특정한 패턴이 반복되는데 이러한 특성으로 인하여 CPU가 참조하는 데이터들 역시 특정 데이터가 자주 참조되는 특성을 참조의 지역성이라고 한다.
참조의 지역성은 공간, 시간, 순차 유형으로 분류된다.

2. 운영체제의 메모리 관리의 목표는 무엇이며 왜 필요한가?

운영체제 메모리 관리의 목표는 크게 4가지로 나뉜다.

첫째, 메모리는 여러 프로세스에 의해 사용되는 공유자원이다. 그러므로 프로세스가 마음대로 임의의 메모리 영역을 사용하도록 놓아둘 수 없다. 따라서 운영체제 커널은 각 프로세스가 사용하는 공간이 겹치지 않도록 전체적으로 메모리 할당과 관리를 담당한다.

둘째, 메모리를 보호하기 위해서이다. 프로세스가 다른 프로세스에게 할당된 메모리를 접근하거나 사용자 모드에서 커널의 영역에 접근하는 것을 막는다.

셋째, 메모리 용량을 극복하기 위해서이다. 시스템에 설치된 메모리보다 더 큰 프로세스나 작지만 많은 프로세스들을 동시에 실행시킬 때 메모리가 이들을 수용하지 못하는 경우가 발생한다. 따라서 가상 메모리와 같은 메모리 관리 정책이 필요하다.

넷째, 메모리 효율성을 높이기 위해서이다. 정해진 양의 메모리에 가능하면 많은 프로세스들을 실행시켜 시스템 처리율을 높이는 메모리 관리가 필요하다.

3. 논리주소(가상주소)와 물리주소의 정의를 말하고 논리주소가 필요한 이유는 무엇인가?

물리주소는 하드웨어적으로 고정된 메모리 주소이다. 반면 논리주소는 개발자나 프로그램(프로세스)에서 사용하는 주소로, 코드나 변수를 엑세스할 때 사용하는 주소이다. 프로그램을 실행할 때 컴파일러는 사용자가 작성한 프로그램을 논리주소로 컴파일한다. 컴파일하는 시점에서 응용프로그램이 메모리 몇번지에 로딩될지 알수 없기 때문에 물리주소로 컴파일하는 것은 불가능하다. 그러므로 컴파일 후 생성된 실행 파일 내에 모든 코드와 변수들을 논리주소로만 구성한다. 그리고 MMU(Memory Management Unit)을 통해 논리주소를 물리주소로 변환하는 과정을 거친다. 이 외에도, 프로세스별 가상주소를 가지게 해서 메모리 영역을 보호할 수 있다. 그리고 사용자가 물리주소를 알 수 없기 때문에 보안성이 향상된다. 사용자의 관점에서본때, 프로세스 내 요소들이 연속적으로

적재되었다고 가정하여 실행하기 때문에 프로그램 작성 및 실행에 효율적이다.

4. 여러 프로세스를 메모리에 할당하는 전략 중 하나로서, 연속 메모리 할당 기법을 간단히 설명하고, 다른 프로세스가 할당된 영역을 침범하지 않게 보호하는 방법은 무엇인가?

연속 메모리 할당은 각 프로세스에게 메모리 한 덩어리씩 할당하는 기법이다. 여기서 연속이라는 뜻은 모든 프로세스들이 연속이 된다는 의미가 아니라 할당 받은 메모리가 한 덩어리처럼 연속된 메모리라는 의미이다.
연속 메모리 할당은 다시 고가지로 구분된다. 고정 크기 할당은 메모리를 파티션으로 불리는 고정 크기의 영역으로 나누고 프로세스마다 1개의 파티션을 할당하는 방법이다. 가변 크기 할당은 프로세스 크기의 연속된 메모리 공간을 할당하는 방법이다. 프로세스가 다른 프로세스의 메모리를 침범하거나 엑세스하는 것을 막기위해 MMU는 논리주소와 limit 레지스터 값을 비교하여 할당된 메모리 범위를 넘어섰다면 바로 시스템 오류 신호를 발생시키는 회로를 포함한다. 시스템 오류 신호가 발생하면, CPU는 오류를 처리하는 커널 코드를 실행하고 현재 프로세스는 강제로 종단시킨다.

5. 가변 크기 할당 전략을 사용할때, 동적으로 발생하는 메모리 할당 요청에 대해 적합한 공간을 찾는 3가지 알고리즘을 간단히 설명하라.

1. first-fit
홀 리스트를 검색하여 처음으로 만나는, 요청 크기보다 큰 홀을 선택한다.
알고리즘의 속도는 빠르지만 비효율로 인한 메모리 낭비가 높다.

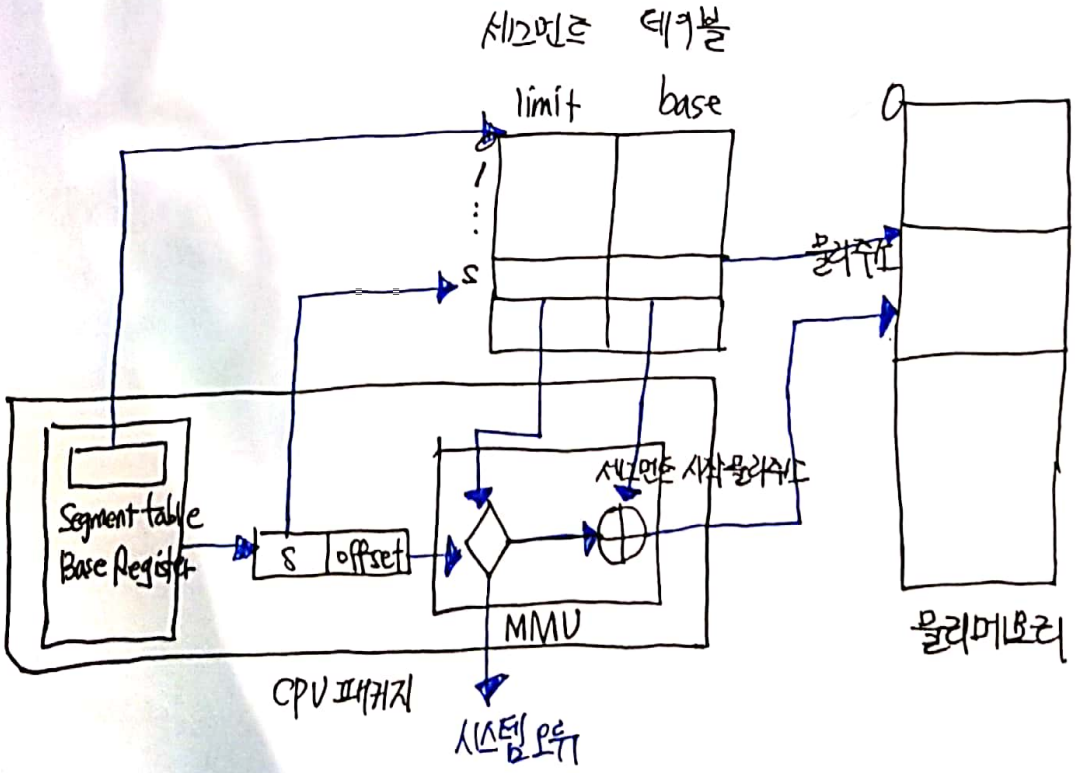
2. best-fit
홀 리스트를 검색하여, 요청 크기를 수용하는 것 중 가장 작은 홀을 선택한다.
그결과 할당된 메모리 영역(홀) 내에 가장 작은 홀이 새로 생긴다.
이경우 홀 리스트가 크기별로 정렬 되어있지 않으면 홀을 전부 검색해야하는 부담이 있다.

3. Worst-fit

홀 리스트를 검색하여, 요청 크기를 수용하는것 중 가장 큰 홀을 선택하는 알고리즘.
이다. 새로 할당된 메모리 영역(홀)에 가장 큰 홀이 새로 생긴다.
이는 홀 리스트가 크기 별로 정렬되어있지 않으면 홀을 전부 검색해야하는
부담이 있다.

6. 운영체제의 메모리 관리 정책 중 세그멘테이션 방법을 간단히 설명하라.
논리주소의 구성과 논리주소에서 물리 주소로 변환되는 과정을 그리고 설명하라.

세그멘테이션 방법이란 프로세스의 주소 공간을 크기가 다른 여러 개의 세그먼트로
나누고 각 세그먼트를 하나의 연속된 물리 메모리 블록에 배치하는 메모리 관리
기법이다. 세그 멘테이션은 코드/데이터/스택/동적 할당 세그먼트로 나뉜다.
그리고 컴파일가 코드 세그먼트와 데이터 세그먼트를 구성한 후 실행파일을
생성한다. 프로그램 실행이 될 때는 운영체제의 로더가 실행파일 내
논리 세그먼트를 물리 세그먼트로 로딩시켜준다. 운영체제는 스택 세그먼트와 동적
할당 세그먼트를 필요시 물리 메모리에 할당해준다.



1) CPU에서 Segment table base register 이용하여 메모리에 저장된 세그먼트 테이블을 찾는다.

2) 논리 주소의 세그먼트의 번호로 세그먼트 테이블에서 limit 값을 검색한다. 그리고 이것을 논리 주소의 옵셋과 비교하여 옵셋이 limit 보다 크다면 시스템 오류를 발생시키고 아니라면 다음 과정으로 넘어간다.

3) 세그먼트 테이블의 base 값과 옵셋을 더하여 물리 주소를 구한다.

7. 페이징 기법이 세그먼테이션 기법보다 선호된다. 그 이유는 무엇인가?

세그멘테이션은 영역별 크기가 가변적이기 때문에 외부 단편화가 발생한다.
반면, 페이징은 고정된 크기로 메모리를 분할하기 때문에 내부 단편화가 발생하지만 외부 단편화 발생 확률이 없다. 여기서 외부 단편화는 내부 단편화에 비해 메모리에 끼치는 영향이 크다.

8. MMU의 기능은 무엇인가? MMU는 하드웨어인가 소프트웨어인가? 그리고 컴퓨터 어느 곳에 존재하는가?

MMU는 주소변환 하드웨어이다. 오늘날 CPU 패키지 내에 구성되어 있다.
MMU의 기능은 운영체제가 프로세스마다 만들어둔 물리 메모리의 주소와 크기 정보를 참고하여 논리 주소를 물리주소로 변환하는 것이다.

9. 연속 메모리 할당, 세그먼테이션, 페이징 방법에서 외부 단편화와 내부 단편화 중 어떤 것이 발생하는지 설명하라.

연속 메모리 할당의 경우, 먼저 고정 크기할당은 내부 단편화가 일어난다.
모든 파티션이 동일한 크기를 할당 받으므로 파티션 내부에 홀이 발생하기 때문이다.
가변 크기 할당은 외부 단편화가 일어난다. 크기가 다른 파티션이 할당되고 반환되면서 파티션과 파티션 사이의 작은 홀이 발생하기 때문이다.
세그멘테이션은 가변 크기 할당 방식이므로, 각 세그먼트와 세그먼트 사이에 작은 홀이 발생하는 외부단편화가 발생한다.

페이징은 내부 단편화가 발생한다. 고정 크기 할당 방식이므로 세그먼테이션과 달리 외부 단편화는 일어나지 않지만 프레임 내부에 흔히 발생한다.

10. 32Bit CPU 에서 페이징을 사용하는 시스템이 있다. 한 페이지의 크기가 8KB이고 물리 메모리가 1GB를 장착한 컴퓨터에서 다음 물음에 답하라.

(1) 프로세스의 주소 공간의 크기는 얼마인가?

$2^{32} = 4GB$

(2) 물리 메모리의 최대 크기는 얼마까지 가능한가?

최대로 엑세스 가능한 물리 주소 범위는 $0 \sim 2^{32}$ 로 총 2^{32} 개이다.

따라서 최대 4GB 까지 가능하다.

(3) 한 프로세스는 몇개의 페이지로 구성되는가?

한 페이지의 크기가 8KB 이므로

$2^{32} / 2^{13} = 2^{19}$ 개로 약 50 만개의 페이지로 구성된다.

(4) 물리 메모리가 1GB 일때, 메모리 프레임의 개수는?

프레임의 크기는 8KB로 페이지의 크기와 동일하다.

따라서 $1GB / 8KB = 2^{30} / 2^{13} = 2^{17}$ 개이다.

약 12만 5천개이다.

(5) 하나의 프로세스를 위한 프로세스 테이블의 항목 개수는?

프로세스 테이블의 항목 개수는 프로세스의 총 페이지 개수와 동일하다.

2^{19} 개로 약 50 만개이다.

(6) 페이지 테이블의 한 항목의 크기는 합리적으로 얼마이면 적당하겠는가? 이유는?

32bit CPU에 한 페이지의 크기가 8KB 이므로, 총 32비트에서 하위 13비트는 옵셋으로, 상위 19비트는 페이지번호로 두어 항목의 크기는 32bit = 4byte로 두어야 한다.

(7) 앞의 (6)의 질문의 답에 4byte라고 하면, 한 프로세스를 위한 페이지 테이블의 크기는?

2^{19} 개의 페이지가 4byte 만큼 있으므로 한 프로세스의 페이지 테이블 크기는

2^{21} byte = 4MB 이다

(8) 페이지 크기와 페이지 테이블의 크기 관계는?

페이지의 크기가 증가할수록 페이지 개수가 줄어들기 때문에 테이블 인덱스는 감소되어 테이블의 크기는 작아지며, 페이지의 크기가 감소할수록 테이블의 크기는 증가한다.

(9) 페이지 크기가 클수록 좋은가 나쁜가?

현대의 운영체제는 메모리가 부족하면 할당된 프레임을 디스크에 저장해두고 빈 프레임을 만들어 필요할 때 디스크에서 다시 메모리로 읽어들이는 가상 메모리 기법을 활용하므로, 이때 디스크에 읽고 쓰는 횟수를 줄이기 위해서 페이지의 크기를 키우는 추세이다. 따라서 페이지의 크기는 클수록 좋다.