* 1. 운영체제에서 장치(device)는 크게 (문자)장치와 (블록)장치로 구분된다. 그리고 이러한 장치를 추상화하는 SW를 (디바이스 드라이버)라고 한다.
  2. 디스크를 스케줄링하는 기법으로 탐색 거리(seek time)와 회전 지연 시간(rotation latency)을 앞 또는 뒤로 움직이면서 요청들을 처리하는 기법이 (SSTF)이다. 반면, 엘리베이터 스케줄러라고도 불리며 모든 트랙을 앞 또는 뒤로 움직이면서 요청들을 처리하는 기법이 (SCAN)이다.
  3. 페이지 테이블 엔트리에는 다양한 비트들이 있다. 이 중에서 페이지 결함 확인에 이용되는 비트가 (V valid bit), Clock 알고리즘에서 사용되는 비트가 (A access bit), 지연 쓰기에서 이용되는 비트가 (D dirty bit)이다.
  4. 플래시 메모리를 관리하는 소프트웨어인 FTL(Flash Translation Later)을 주요 3가지 기능에는 (Translation, Garbage collection, Wear-leveling)
  5. 주소 변환을 도와주는 CPU 내부의 구성요소를 (MMU)라 하며, 여기에는 base/Limit등의 레지스터와 TLB가 포함된다. 메모리와 장치 간에 데이터 복사를 수행하며 CPU가 다른 유용한 일을 할 수 있도록 해 주는 컨트롤러를 (DMA)라고 한다.

2.1 어떤 디스크의 Average seek time, RPM, Max transfer가 각각 5ms, 12000, 200MB/s이라고 가정합시다. 임의 워크로드와 순차 워크로드에서 I/O rate를 계산하시오.

2.2 SCSI 하드 디스크를 구입하여 3개의 파티션으로 나누었다고 가정합시다. “ls -l” 명령을 수행했을 때 “brw-rw----1 root disk 8, 1 6월 8일. /dev/sdal”이라는 정보를 볼 수 있는데, 이때 b, 8, 1, sdal의 의미를 각각 기술하시오.

b: 파일 유형중 블록 장치파일

8: 블록 파일 장치의 주 번호

1: 블록 파일 장치의 부 번호 (장치유형)

Sda1 : SCSI 디스크의 첫번째 디스크의 첫번째 파티션

2.3 리눅스에서 link, mount, fsync의 기능을 설명하시오.

Link : 어떠한 곳에 있는 실행 파일을 거기까지 가지 않더라도 다른 곳에서 실행 시킬 수 있게 하는것. 바로가기 처럼 주소 링크만 한 것이 symbolic link, 원본 데이터와 직접 연결시켜 만드는 것이 hard link 이다.

3.1 VSFS를 구성하는 4가지 구성 부분의 이름을 쓰시오.

Super block, bitmap, inode, userdata

3.2 VSFS에서 /, /lab1.pdf, /lab1.zip라는 3개의 파일을 생성하였습니다. /, /lab1.pdf, /lab1.zip 파일의 크기가 각각 100B, 6KB, 60KB라고 할 때 VSFS의 변화를 그림으로 설명하 시오. (7점, inode에서 direct block pointer는 12개로 가정). 3.3 캐싱과 지연 쓰기를 사용할 때 장점을 "문제 3.2를 이용"하여 설명하시오 (3점) 3.4 FFS에서 /, /a, /b, /a/c, /a/c/d, /a/c/e, /b/f, /g라는 8개의 파일을 생성합니다. 이때 /, /a, /b, /a/c는 디렉터리이고 d, e, f, g는 일반 파일입니다. 디렉터리의 크기는 모 두 4KB이며, 일반 파일의 크기는 d, e, f는 6KB, g는 60KB라고 할 때, FFS에서 어떻게 할당 되는지 실린더 그룹 측면에서 설명하시오. (6점)

4.1 “문제 3.2의 예” 에서 일관성이 지켜지지 않는 상황을 space leak, garbage read, inconsistent, dangling reference라는 용어를 사용하여 살명하시오.

=> bit map만 작성되거나 Data, bitmap만 작성된 경우 space leak이 발생하며 inode만 작성되거나 inode 만 작성이 안되거나 bitmap만 작성이 안된 경우 inconsistent 상태가 된다. 또한 inode 수정이 제대로 되지 않아 반납된 디스크 공간을 가리키는 것이 dangling reference이며 data만 안 쓰여 진 경우 garbage read현상이 나타날 수 있다.

4.2 저널링을 사용하면 일관성을 유지할 수 있는데, 이 이유를 transacation, commit, redo, undo를 사용하여 설명하시오.

=> 저널링은 트랜잭션이나 경신되는 데이터 변경 이력을 남기는 것으로 TxB(트랜잭션 시작블록) 부터 TxE (트랜잭션 종료 블록)까지 작성이 되면 저널링이 완료 된다. 이 때 commit방식을 사용하면 TxE ~ Data까지는 한 번에 작성 후 fsync()를 하고 이후에 TxE가 작성되면 commit하는 방식으로 TxE가 작성되면 앞의 내용이 모두 작성이 되었다는 것을 보장한다. 또한 TxE가 작성이 되어있고 checkpoint단계에 문제가 발생한 경우 Redo를 하면 되며, commit이 되지 않은 상태에서 문제가 발생하면 Undo를 하면 된다.

4.3 Ex3에는 data journal, ordered, writeback이라는 3가지 다른 유형의 저널링이 있는데, 각 저널링의 차이점을 논하시오.

=> Data journal: 메타 데이터 및 데이터 변경에 대해 저널에 모든 로그를 남기는 것으로 모든 정보가 파일 시스템에 쓰이기 전에 저널링이 된다. 가장 정확하게 유지 할 수 있으나 디스크 접근이 많아 성능이 좋지 않다.

Ordered: 데이터 블록들을 메타 데이터보다 먼저 디스크에 기록하도록 한다. 이 때 데이터 블록은 저널링을 하지 않고 바로 스토리지에 저장을 하고 그 후 메타데이터 블록을 저널링 한다.

Writeback: 메타데이터에 대한 변경 로그만 남기며 데이터와 메타 데이터간 쓰기 순서를 보장하지 않기에 속도는 매우 빠르다. 하지만 시스템 문제가 발생하기 전에 쓰던 파일들은 이전 데이터 또는 garbage read현상이 발생 할 수 있다.

4.4 github의 일관성 유지 방법과 OSTEP에서 배운 내용 파일 시스템 일관성 유지 간에 공통점을 논하시오.

5 다음 질문에 답하세요. (총 13점) 5.1 메모리 관리에서 연속 할당과 세그멘테이션(segmentation), 페이징(paging)의 차이점을 기술하시오.

* 연속 할당이란 프로세스 구성요소가 메모리에 연속된 주소로 저장되는 방식이다. 이는 빈 공간이 차지하고 있는 공간이 크다는 단점이 있으며 내부 단편화를 초래한다. 따라서 이를 방지하기 위해 불 연속 할당 반식이 있으며 여기에는 가변크기를 사용하는 세그멘테이션과 고정크기를 사용하는 paging방식이 있다 세그멘테이션은 가변 크기를 사용하기에 외부 단편화가 발생할 수 있으며 하드웨어 구현이 어렵다. 하지만 paging의 경우 고정 크기를 사용하기에 외부 단편화를 방지 할 수 있다.

5.2 가상 메모리(virtual memory) 크기가 32KB이고 물리 메모리(physical memory) 크기가 64KB로 가정합시다. 이 시스템은 세그멘테이션을 사용하며, 각 세그먼트의 최대 크기가 8KB 라고 가정합시다. 한 프로세스가 생성되었는데, 코드, 힙, 스택 세그먼트의 크기가 각각 5KB, 1KB, 6KB이고, 각각 물리 메모리의 10KB, 20KB, 30KB로 적재되었다고 할 때, 세그멘테이션 테이블을 그림으로 설명하세요. (3점)

5.3 가상 주소(virtual address)가 1000, 10000, 30000일 때 물리 주소(physical address)가 어떻게 되는지를 설명하세요. (6점)

가상 메모리크기 32KB -> 2^15, address space size = 15bit

세그먼트 개수: 3-> 2bit

Number of offset : remaining 13 bit, max size of segment : 8KB

물리 메모리 크기 64KB -> 2^16 address space size = 16bit

가상 주소 1000 -> 00 0001 1110 1000

Segment 00 -> code부분

물리주소 : 10kb + 1000

가상주소 10000 -> 01 0111 0001 0000

Segment 01 -> heap부분

Heap 크기를 초과했기에 segmentation fault 발생

가상주소 30000 -> 11 1010 1001 1000

Segment 11 -> stack 부분

물리주소 : 30kb + 5424 – 8192 = 30kB - 2768

5.4 아래 프로그램 예에서 SEG\_MASK, SEG\_SHIFT, OFFSET\_MASK가 얼마입니까? (2점)

/\* 문제 5.4번을 위한 Pseudo Code 시작 \*/

// get top 2 bits of 14-bit VA

Segment = (VirtualAddress & SEG\_MASK) >> SEG\_SHIFT

// now get offset

Offset=VirtualAddress & OFFSET\_MASK

if (Offset >= Bounds[Segment])

RaiseException(PROTECTION\_FAULT)

else

PhysAddr = Base[Segment] + Offset

Register = AccessMemory(PhysAddr)

/\* 문제 5.4번을 위한 Pseudo Code 끝 \*/

9주차 1차시 퀴즈

1. 폴링(polling)과 비교해 볼 때 인터럽트(Interrupt)가 가지는 장점과 단점을 논하시오. 인터럽트에 DMA를 함께 도입하면 제공할 수 있는 추가적인 장점을 논하시오. (1.5점)

2. 문자 장치 드라이버와 블록 장치 드라이버의 차이점을 논하시오. (1점, 2가지 이상 차이점을 설명)

(보너스) 리눅스에서 모듈(module)을 커널에 적재할 때 사용하는 명령어의 이름을 쓰시오. (0.5점)

==>

인터럽트 장점: CPU 수행과 장치 동작을 중첩(overlapping)시켜 CPU를 다른 유용한 작업이 사용 가능.

인터럽트 단점: 인터럽트 처리를 위한 기법 (예를 들어 인터럽트 핸들러와 sleep 관리) 필요 또는 Sleep을 위해서는 문맥 교환이 필요한데 사건이 빨리 발생할 경우 문맥 교환의 부하 때문에 폴링이 유리할 수도 있음.

DMA까지 도입하면 메모리에서 장치로 데이터 복사하는 부분까지 DMA controller가 처리. 이 기간동안 CPU를 다른 유용한 작업이 사용 가능.

==>

차이점 1: 사용자(또는 응용) 직접 접근 vs 파일 시스템을 통해 접근, 차이점 2: 인터페이스 차이 read(), write() vs. rw()

insmod 또는 modprob

9주차 2차시 퀴즈

1. Barracuda 디스크에서 Tseek, Trotation, Ttransfer, Ti/o, Ri/o를 계산하시오. (2점, random, sequential workload에 대해 각각 계산)

2. fsync()가 필요한 이유를 buffer cache 또는 비동기적 쓰기라는 용어를 사용하여 설명하시오. (1점)

==>

임의 쓰기에서

Tseek = 9ms, RPM이 7200이므로 1초에 120바퀴 회전. 결국 한바퀴 회전 시간은 1/120초 = 8.33ms 평균은 반바퀴. 따라서 Trotation= 4.17ms,

비례식 "105MB:1000ms = 4KB:x"에 의해 Ttransfer = 4000/105000 = 0.038ms

Ti/o = 9 + 4.17 + 0.038ms = 대략 13ms,

Ri/o = 4KB / 13ms = 0.31MB/s

순차쓰기에서

Tseek = 9ms, Trotation= 4.17ms, 비례식 "105MB:1000ms = 100MB:x"에 의해 Ttransfer = 100000/105 = 950ms

Ti/o = 9 + 4.17 + 950ms = 대략 950ms,

Ri/o = 100MB / 950ms = 105MB/s

(Ri/o를 100MB/963ms = 103MB/s으로 답변해도 정답 인정)

==>

파일 시스템에서는 버퍼 캐시에서 비동기적 쓰기 연산인 지연 쓰기를 적용하여 성능 향상을 시도.

이때 버퍼 캐시에 있는 수정 내용을 디스크에 써서 durability를 보장 받고 싶을 때 fsync() 사용.

10주차 1차시 퀴즈

1. "mkfs" 명령으로 만든 VSFS는 4 영역으로 구성됩니다. 각 영역의 이름과 기능을 설명하시오. (1점)

2. 오른쪽 하단 그림에서 “-”, “l”, “b”, “c”, ”rw” “5”, “8”, “1”의 의미를 각각 설명하시오. (1점)

==>

Superblock: 파일 시스템에 대한 메타데이터 관리 (파일 시스템 크기, 파일 시스템 유형, 루트 데렉터리 위치 등)

비트맵: 가용 공간을 관리하는 자료 구조, 일반적으로 data bitmap와 inode bitmap으로 구분됨.

아이노드: 파일 시스템에 대한 메타데이터 관리 (파일 크기, 파일 유형, 속성, 소유자 등)

데이터 영역: 사용자가 쓴 데이터 저장 공간

==>

“-”: 파일 유형 중 정규 파일, “l”: 파일 유형 중 링크, “b”: 파일 유형 중 블록 장치 파일, “c”: 파일 유형 중 문자 장치 파일,

”rw”: 속성, 읽기와 쓰기 가능 “5”: 문자 장치 파일의 주 번호, “8”: 블록 장치 파일의 주 번호 (장치 유형), “1”: 블록 장치 파일의 부 번호 (장치 유닛)

10주차 2차시 퀴즈

1. 75KB 크기의 파일을 생성할 때 데이터 영역(data region)에서 몇 개의 디스크 블록을 할당하나요? 이 중에 index block은 어떤 것 인가요? (1점)

2. 파일을 읽을 때 inode와 데이터를 번갈아 읽게 됩니다. 이때 긴 탐색 거리(long seek distance)가 발생하여 성능이 저하될 수 있는데, 이 문제를 해결할 수 있는 자신만의 아이디어를 제안해 보시오. (1점, 창의적인 아이디어 기대합니다.^^)

==>

75KB/4KB = 18.75 결국 19개의 디스크 블록 필요. 이중에 12개는 direct, 7개는 single indirect.

이때 single indirect를 위한 index block 1개 필요.

결국 19 + 1 = 20개 디스크 블록이 데이터 영역에서 할당됨.

13번째 할당 블록이 인덱스 블록.

(예를 들어 11, 12, 13, ..., 30 순서로 20개의 블록이 할당되었다면 11~22번 블록이 direct, 23번 블록이 index, 24~30번 블록이 single indirect로 사용됨.)

==>

11주차 1차시 퀴즈

1. FFS에서 사용하는 할당 규칙 2(rule 2)의 필요성을 이름공간 지역성(namespace locality)라는 용어를 사용하여 설명하시오. (1점)

2. OSTEP 41장 2페이지를 참고하여, 단편화(fragmentation)이 발생하는 이유와 defragmentation 도구의 장점을 설명하시오. (1점)

==>

규칙 2는 새로운 파일을 위한 디스크 공간을 할당할 때 이 파일이 속한 디렉터리와 같은 실린더 그룹에 할당하는 것임.

이 이유는 같은 디렉터리에 존재하는 파일들은 함께 접근되는 이름공간 지역성을 보이며, 이들을 같은 실린더에 할당하여 탐색 거리(seek distance)를 줄이는 장점을 얻기 위함임.

==>

외부 단편화는 많은 파일들의 생성과 삭제가 발생함에 따라 가용 공간이 불연속적으로 떨어져 있게되는 상태.

이때 이 공간들을 한 파일에게 할당해 주면 연속적이지 않고 불연속적으로 저장되며, 이 파일을 읽을 때 도중에 탐색(seek)이 발생하여 성능이 저하될 수 있음.

defragmentation 도구는 흩어져 있는 가용 공간을 연속적으로 모아 외부 단편화를 줄일 수 있어 성능 향상을 제공함.

참고로, 단편화는 외부 단편화(external fragmentation)와 내부 단편화(internal fragmentation)로 구분 가능함.

외부 단편화는 위에서 설명하였으며 내부 단편화는 실제 필요한 데이터보다 더 큰 고정된 크기의 디스크 블록을 할당함에 따라, 내부에서 사용되지 않고 낭비되는 공간을 의미함.

11주차 2차시 퀴즈

1. 왼쪽 하단 그림은 4KB 크기의 새로운 파일이 생성되었을 때 파일 시스템의 변화를 보여줍니다. 이 그림을 사용하여 1) space leak, 2) garbage read, 3) dangling reference, 4) inconsistent라는 용어를 설명하시오. (1점)

2. FTL의 핵심 3가지 기능인 1) address mapping, 2) garbage collection, 3) wear-leveling의 필요성을 플래시 메모리 특성들을 이용하여 설명하시오. (1.5점)

==>

Space leak: 비트맵만 쓰여지고 다른 것은 쓰여지지 않아, 사실상 가용한 공간인데 할당이 되지 않음

Garbage read: 아이노드는 쓰여졌지만 데이터는 쓰여지지 않아, 이후 읽을 때 이상한 데이터가 읽히는 상황

Dangling reference: 존재하지 않는 객체를 참조(reference)하는 상태. 예를 들어 디스크 블록이 반납되어 더 이상 존재하지 않는데, 아이노드 수정이 제대로 이루어 지지 않아 반납된 디스크 블록을 여전히 참조하는 상황.

(LN 6에서는 symbolic link에서 원본 파일은 삭제되어 더 이상 존재하지 않는데, 심볼릭 링크가 여전히 참조하고 있는 상황을 Dangling reference의 예로 들었음.)

Inconsistent: 비트맵, 아이노드, 데이터 블록 중 일부는 쓰여지고 다른 일부는 쓰여지지 않아 일관성이 깨어진 상태

==>

address mapping: 플래시 메모리는 삭제 후 쓰기 제약이 있으며, 삭제 단위(블럭)가 쓰기 단위(페이지)보다 큼.

따라서 쓰기가 발생할 때마다 in place update를 수행하면 부하가 큰 삭제 및 복사가 필요.

이 문제를 완화하기 위해 out place update 즉 다른 주소에 갱신이라는 기법으로 쓰기를 처리하는데, 이를 위해서는 주소 정보를 관리하는 주소 매핑 및 이를 이용한 주소 변환이 필요

garbage collection: out place update를 수행하면 무효화된 기존 데이터가 차지하던 공간을 회수해야 하는데, 이를 위해 garbage collection 사용.

wear-leveling: 플래시 메모리는 삭제 횟수 제한이 있으며, 특정 블록이 집중적으로 삭제되면 마모가 심해져 메모리의 수명을 저하시킴. 이를 해결하기 위해 마모도 평준화 기법 사용.

12주차 1차시 퀴즈

1. 가상 메모리와 물리 메모리의 차이점을 설명하시오. (1점, 차이점을 3가지 이상 설명)

2. 아래 그림을 이용하여 1) 주소 공간(address space), 2) 주소 변환(address translation), 3) base register, 4) limit register, 5) 재배치(relocation)을 설명하시오. (1점)

==>

가상 메모리와 물리 메모리의 차이

1) 가상 메모리는 시스템에서 프로세스 개수만큼 존재, 반면 물리 메모리는 시스템에 하나만 존재

2) 가상 메모리는 한 프로세스가 독점적으로 사용, 반면 물리 메모리는 여러 프로세스들이 공유

3) 가상 메모리는 0번부터 시작, 반면 물리 메모리는 0이 아닌 위치부터 시작하는 경우가 대부분

4) 일반적으로 가상 메모리의 크기가 물리 메모리 크기보다 큼 (가상 메모리 크기는 CPU에서 사용하는 주소 버스 크기에 의해 결정, 물리 메모리는 시스템에 탑재된 DRAM의 용량에 의해 결정)

5) 그 외 다른 차이점들도 정답 인정 가능

==>

1) 주소 공간(address space): 가상 메모리는 텍스트, 데이터 힙, 스택 등 잘 정의된 레이아웃을 가지고 있으며 이를 주소 공간이라고 함

2) 주소 변환(address translation): 가상 메모리의 주소 (가상 주소)를 물리 메모리의 주소 (물리 주소)로 변환하는 작업

3) base register: 연속 할당에서 물리 메모리의 시작 주소를 가리킴

4) limit register: 연속 할당에서 물리 메모리의 끝 주소를 가리킴 (또는 크기를 가리킬 수도 있음. 시스템에 의존적)

5) 재배치(relocation): 물리 메모리가 가용 공간의 어느 위치에든 적재 가능함. 수행 중에 물리 메모리의 위치가 바뀔 수도 있음 (예를 들어 Swap-out 이후 다시 Swap-in이 되었을 때)

12주차 2차시 퀴즈

1. 메모리 가상화에서 1) 컴파일러, 2) 운영체제, 3) CPU (구체적으로 MMU)의 역할을 설명하시오. (1점)

2. 아래 그림을 이용하여 가상 주소 100, 5000, 7000의 물리 주소를 각각 계산하시오. (1.5점. 계산할 때 세그먼트 번호와 오프셋이라는 용어 반드시 사용)

==>

1) 컴파일러: 가상 메모리를 결정. 즉, 함수의 텍스트 상에 위치, 전역 변수의 데이터 상에 위치, 스택에서 인자 위치 등과 같은 주소 공간 결정

2) 운영체제: 수행 중 물리 메모리의 위치, 즉 적재와 재배치 처리.

3) CPU (구체적으로 MMU): 가상 주소를 물리 주소로 변환하는 작업 담당

==>

가상 메모리가 16KB임. 결국 주소는 14비트. 이중에 상위 2비트가 세그먼트 번호. 하위 12 비트가 오프셋

1) 가상 주소 100 --> 이진수로는 00 0000 0110 0100 --> 세그먼트 번호가 0, 오프셋이 100 --> 코드 세그먼트 --> 시작이 32KB --> 사이즈 이하 --> 결국 물리 주소는 32KB + 100

2) 가상 주소 5000 --> 이진수로는 01 0011 1000 1000 --> 세그먼트 번호가 1, 오프셋이 904 --> 힙 세그먼트 --> 시작이 34KB --> 사이즈 이하 --> 결국 물리 주소는 34KB + 904

3) 가상 주소 7000 --> 이진수로는 01 1011 0101 1000 --> 세그먼트 번호가 1, 오프셋이 2904 --> 힙 세그먼트 --> 시작이 34KB --> 사이즈 초과 --> 결국 세그멘테이션 폴트

13주차 1차시 퀴즈  
1. 아래 왼쪽 그림을 이용하여 1) 스왑 아웃, 2) 재배치, 3) 외부 단편화, 4) 집약(compaction), 5) 분할(splitting), 6) 합병(coalescing)의 기능을 논하시오. (1.5점)   
2. 아래 오른쪽 그림에서 SEG\_MASK, SEG\_SHIFT, OFFSET\_MASK의 값이 얼마입니까? (1점)   
==>  
1) 스왑 아웃: 그림 (d)->(e) 단계에서 process 2가 swap out됨 (이후 (g)->(h) 단계에서 swap in)  
2) 재배치: Process 2의 시작 위치가 처음에는 28MB 였는데 (그림 (c)), Swap out/in 이후 8MB로 재배치됨 (그림 (h))    
3) 외부 단편화: 그림 (h)에서 가용 공간 위치가 단편적으로 떨어져 있음 (그림 (e)~(ㅎ)를 예로 설명해도 정답)  
4) 집약(compaction): 그림 (h)에서 떨어져 있는 3개의 가용 공단들을 한 곳으로 모으면 집약  
5) 분할(splitting): 그림 (e)->(f) 단계에서 14MB 크기의 가용 공간을 분할할여 8MB는 Process 4에게 할당하고 남은 6MB는 여전히 가용 공간으로 관리  
6) 합병(coalescing): 예를 들어 그림 (d)에서 Process 3이 끝나거나 스왑 아웃되면 사용하면 공간을 반납하는데, 이때 아래에 있던 가용 공간과 합병하여 하나의 가용 공간으로 합침.  
==>  
SEG\_MASK는 0x3000, SEG\_SHIFT는 12, OFFSET\_MASK는 0xfff  
  
  
13주차 2차시 퀴즈  
1. 연속할당, 세그멘테이션, 페이징의 차이점을 설명하시오. (1점)  
2. 아래와 같은 환경에서 가상 주소 8, 21, 62의 물리 주소를 각각 계산하시오. (1점. VPN, PFN, offset이라는 용어를 사용하여 설명)  
- 가상 메모리 크기: 64B, 물리 메모리 크기: 128B, 페이지/프레임 크기: 16B  
- 페이지 0, 1, 2은 각각 페이지 프레임 2,6,3에 적재됨   
==>  
연속할당: 가상 메모리가 연속적으로 물리 메모리에 적재됨  
세그멘테이션: 가상 메모리를 여러 영역들으로 나누고 각 영역은 불연속적으로 물리 메모리에 적재 가능. 각 영역의 크기는 가변적 (각 영역을 세그먼트라고 부름)  
페이징: 가상 메모리를 여러 영역들으로 나누고 각 영역은 불연속적으로 물리 메모리에 적재 가능. 각 영역의 크기는 고정적 (각 영역을 페이지라고 부름)  
==>  
가상 주소는 6비트, 이중에 상위 2비트는 VPN, 하위 4비트는 offset  
가상 주소 8 --> 이진수로 001000 --> VPN는 0, offset은 8 --> PFN는 2 --> 따라서 물리 주소는 2\*16 + 8 = 40  
가상 주소 21 --> 이진수로 010101 --> VPN는 1, offset은 5 --> PFN는 6 --> 따라서 물리 주소는 6\*16 + 5 = 101  
가상 주소 62 --> 이진수로 111110 --> VPN는 3, offset은 14 --> 적재되어 있지 않음 --> 페이지 부재 결함   
  
  
14주차 1차시 퀴즈  
1. OS가 TLB와 Multi-level page table을 이용하는 이유를 논하시오. (1점)  
2. 다음 질문에 답하시오. (1점)   
LN9의 18 페이지에서 a 배열을 접근할 때 TLB 적중율이 70%라고 배웠습니다.  
(1) 페이지 크기가 16B가 아닌 32B로 증가하였을 때 TLB 적중율은 얼마로 변합니까?  
(2) 이중 반복문을 사용하게되면 TLB 적중율은 얼마로 변합니까? (이 문제에서 페이지 크기는 16B로 가정)    
==>  
- TLB와 이용하는 이유: 페이징을 사용하면 주소 변환을 위해 페이지 테이블 접근이 필요. 이것은 추가적인 메모리 접근을 야기하여 성능 저하. TLB는 이러한 추가적인 메모리 접근을 줄이기 위해 사용.   
- Multi-level page table을 이용하는 이유: 페이지 테이블의 크기가 큼. 중간의 엔트리들을 사실상 사용하지 않음. 디렉터리에서 이 부분을 사용하지 않는다고 표시해두면 페이지 테이블이 메모리를 많이 사용하는 문제 해결 가능     
==>  
페이지 크기가 16B일 때 --> a[0]~[2], a[3]~[6], a[7]~[9]가 같은 페이지 --> a[0], a[3], a[7]이 미스, 그 외는 적중 --> 따라서 적중율이 70%   
페이지 크기가 32B일 때 --> a[0]~[6], a[7]~[9]가 같은 페이지 --> a[0], a[7]이 미스, 그 외는 적중 --> 따라서 적중율이 80%  
이중 반복문일 경우 --> 처음 j=0일 때에는 a[0], a[3], a[7]이 미스, 그 외는 적중 --> 두 번째인 j가 1일 때에는 모두 TLB에 올라와 있음, 모두 적중 --> 따라서 적중율은 7+10/20 = 85%      
    
14주차 2차시 퀴즈  
1. 다음 5가지 용어를 설명하시오: 1) 요구 페이징, 2) 페이지 결함, 3) 교체, 4) 쓰래싱 (thrashing), 5) 워킹셋. (1점)  
2. 다음 질문에 답하시오. (1점)  
- 1) 페이지 참조 스트링이 "1,2,3,4,1,2,5,1,2,3,4,5"이고 가용 프레임 개수가 3개일 때 FIFO, LRU, Optimal(MIN) 교체 정책의 적중율을 각각 계산하시오.  
- 2) 같은 참조 스트링에서 가용 프레임 개수가 4개로 변했을 때 FIFO, LRU, Optimal(MIN) 교체 정책의 적중율을 각각 계산하시오.  
보너스. 이번 학기에서 배운 내용 중에서 가장 흥미로웠던 또는 가장 어려웠던 내용을 e-campus 문의 게시판에 새글 또는 기존 글의 답글로 올리시오. (1점)   
==>  
1) 요구 페이징: 가상 메모리의 모든 페이지를 물리 메모리에 적재하는 것이 아니라, 필요할 때 (즉 요구될 때) 적재하는 기법   
2) 페이지 결함: 아직 적재되지 않은 페이지를 접근하려 할 때 발생하는 트랩  
3) 교체: 물리 메모리의 프레임이 거의 모두 사용되어 일부를 스왑 공간으로 내려 보내는 작업. 어떤 프레임을 선택할 것인가에 대한 정책을 교체 정책이라고 함.   
4) 쓰래싱 (thrashing): 요구되는 메모리 량에 비해 가용 물리 메모리 양이 적어, 교체와 페이지 결함이 주로 발생할 뿐 실재 프로그램이 해야할 수행을 제대로 하지 못하는 상태   
5) 워킹셋: 최근 델타 시간동안 참조된 페이지들의 집합. 이들이 모두 물리 메모리에 올가가 있어야 쓰래싱이 발생하지 않음.   
==>   
프레임 수개 3개일 때  
1) FIFO: 1(miss: 1)->2(miss: 1,2)->3(miss: 1,2,3)->4(miss: 2,3,4)->1(miss:3,4,1)->2(miss:4,1,2)->5(miss:1,2,5)->1(hit:1,2,5)->2(hit:1,2,5)->3(miss:2,5,3)->4(miss:5,3,4)->5(hit, 5,3,4)  
따라서 적중율은 3/12 = 25%    
2) LRU: 1(miss:1)->2(miss:1,2)->3(miss:1,2,3)->4(miss:2,3,4)->1(miss:3,4,1)->2(miss:4,1,2)->5(miss:1,2,5)->1(hit:2,5,1)->2(hit:5,1,2)->3(miss:1,2,3)->4(miss:2,3,4)->5(miss:3,4,5)  
따라서 적중율은 2/12 = 16.7%   
3) MIN: 1(miss:1)->2(miss:1,2)->3(miss:1,2,3)->4(miss:1,2,4)->1(hit:1,2,4)->2(hit:1,2,4)->5(miss:1,2,5)->1(hit:1,2,5)->(hit:1,2,5)->3(miss:3,2,5)->4(miss:3,4,5)->5(hit:3,4,5)  
따라서 적중율은 5/12 = 41.7%   
프레임 수개 4개일 때  
1) FIFO: 1(miss:1)->2(miss:1,2)->3(miss:1,2,3)->4(miss:1,2,3,4)->1(hit:1,2,3,4)->2(hit:1,2,3,4)->5(miss:2,3,4,5)->1(miss:3,4,5,1)->2(miss:4,5,1,2)->3(miss:5,1,2,3)->4(miss:1,2,3,4)->5(miss:2,3,4,5)  
따라서 적중율은 2/12 = 16.7%. 여기가 Belady's anomaly   
2) LRU: 1(miss:1)->2(miss:1,2)->3(miss:1,2,3)->4(miss:1,2,3,4)->1(hit:2,3,4,1)->2(hit:3,4,1,2)->5(miss:4,1,2,5)->1(hit:4,2,5,1)->2(hit:4,5,1,2)->3(miss:5,1,2,3)->4(miss:1,2,3,4)->5(miss:2,3,4,5)  
따라서 적중율은 4/12 = 33.3%  
3) MIN: 1(miss:1)->2(miss:1,2)->3(miss:1,2,3)->4(miss:1,2,3,4)->1(hit:1,2,3,4)->2(hit:1,2,3,4)->5(miss:1,2,3,4)->1(hit:1,2,3,4)->2(hit:1,2,3,4)->3(hit:1,2,3,4)->4(hit:1,2,3,4)->5(miss:2,3,4,5)  
따라서 적중율은 6/12 = 50%  
==>