문제 1. MMU와 베이스/한계 레지스터의 역할 및 메모리 보호

다음 중 MMU가 주소 변환 시 베이스 레지스터와 한계 레지스터를 활용하여 메모리 보호 기능을 구현하는 방식에 대한 올바른 설명은 무엇인가?

① 베이스 레지스터는 논리 주소에 상수를 단순히 더해 물리 주소를 산출하고, 한계 레지스터는 변환된 물리 주소가 메모리의 최종 주소를 초과하지 않도록 검사한다.

② 베이스 레지스터는 프로세스가 할당받은 실제 메모리 시작 주소를 저장하며, 한계 레지스터는 해당 프로세스가 접근 가능한 논리 주소 범위의 최대값을 제한하여 잘못된 접근을 방지한다.

③ 베이스 레지스터는 각 논리 주소의 오프셋을 재계산하는 역할을 하며, 한계 레지스터는 물리 주소의 페이지 번호를 동적으로 결정한다.

④ 베이스와 한계 레지스터는 주소 변환 후, 물리 주소가 캐시 영역 내에 존재하는지를 확인하는 보조 기능을 수행한다.

⑤ 베이스 레지스터와 한계 레지스터는 프로세스 간 주소 공간 공유를 위해 동적으로 재할당되며, 메모리 보호 대신 메모리 효율성을 극대화한다.

문제 2. 가상 메모리 주소 매핑 – 세그먼테이션과 페이징

가상 메모리 시스템에서 논리 주소를 물리 주소로 매핑하는 주요 기법인 세그먼테이션과 페이징에 대해, 다음 설명 중 옳은 것은 무엇인가?

① 세그먼테이션은 고정 크기의 블록으로 논리 주소를 분할하여 1:1 매핑하는 반면, 페이징은 동적 크기로 분할되어 내부 단편화를 최소화한다.

② 페이징은 논리 주소 공간을 동일한 크기의 페이지로 나누어 물리 메모리의 동일 크기 프레임에 할당하며, 세그먼테이션은 의미 단위로 분할되어 크기가 가변적이며 접근 권한 관리에 유리하다.

③ 페이징 모두 논리 주소를 가변 크기로 분할하지만, 세그먼테이션은 주로 캐시 효율성을 위해 고정 크기로 분할 한다.

④ 세그먼테이션은 전적으로 소프트웨어에서 관리되며, 페이징은 MMU와 TLB를 활용해 하드웨어적으로 빠른 매핑을 제공한다.

⑤ 페이징 기법은 메모리 보호 기능이 없기 때문에, 반드시 세그먼테이션과 병행하여 사용해야 한다.

문제 3. 메모리 계층 구조와 접근 시간 최적화 전략

다음 중 메모리 계층 구조에 따른 접근 시간 최적화 원리를 올바르게 설명한 것은 무엇인가?

① 레지스터와 캐시 메모리는 용량 차이는 크지만, 접근 시간 차이는 미미하여 성능 최적화에 크게 기여하지 않는다.

② CPU는 모든 데이터 접근 시 주 메모리에서 직접 레지스터로 데이터를 로드하므로, 캐시 메모리는 오히려 불필요한 중간 단계로 작동한다.

③ 캐시 메모리는 주 메모리에서 자주 접근하는 데이터를 임시 저장하여 평균 메모리 접근 시간을 단축시키며, 각 계층 간의 접근 시간 차이는 수십 배에서 수백 배에 달한다.

④ 보조기억장치는 메모리 계층의 최상위에 위치하여, 레지스터보다 빠른 접근 속도를 제공함으로써 전체 시스템 성능을 주도한다.

⑤ 캐시 메모리는 레지스터와 주 메모리의 중간 역할을 하지만, 접근 시간 최적화에는 병렬 처리 기법만이 주된 역할을 한다.

문제 4. 캐시 Write 정책과 데이터 일관성

Write-back과 Write-through 캐시 정책이 데이터 일관성과 성능에 미치는 영향을 고려할 때, 다음 중 올바른 설명은 무엇인가?

① Write-through 정책은 모든 데이터 변경을 캐시와 주 메모리에 동시에 반영하여 높은 일관성을 보장하지만, 그로 인해 쓰기 작업 성능이 저하된다.

② Write-back 정책은 데이터 변경 시 캐시 내에만 수정된 내용을 기록하고, 주 메모리로의 반영을 지연시켜 성능을 향상시키지만, TLB 동기화에 문제가 발생할 수 있다.

③ Write-through 정책은 캐시 미스 발생 시 데이터를 주 메모리에서 강제로 갱신하는 반면, Write-back은 갱신 주기가 고정되어 있어 일관성이 낮다.

④ Write-through 정책은 일관성을 보장하여 메모리와 캐시 모두 데이터를 쓰기 때문에 성능 면에서 유리하다

⑤ Write-through 정책은 캐시 업데이트와 동시에 주 메모리에 데이터를 기록하여 높은 데이터 일관성을 유지하는 반면, Write-back은 성능 향상을 위해 지연 기록 방식을 사용한다.

문제 5. 캐시 교체 알고리즘 – LRU의 한계와 보완 기법

다음 설명 중 LRU의 한계와 이를 보완하기 위한 보완 기법에 대한 옳은 설명은 무엇인가?

① LRU는 구현이 단순해 항상 최적의 성능을 보이며, 추가적인 보완 기법이 필요 없다.

② LRU는 순차적 데이터 접근(스트리밍) 시 동일 데이터를 반복 교체하는 단점이 있으며, 이를 보완하기 위해 참조 카운트를 사용하는 LFU(Least Frequently Used) 방식이 고려될 수 있다.

③ LRU의 높은 구현 비용 문제를 해결하기 위해 단순 FIFO(First-In-First-Out) 알고리즘으로 대체하면, 항상 더 나은 캐시 적중률을 얻을 수 있다.

④ LRU는 캐시 상태를 정밀하게 관리하기 어려워 순차적 접근 시 오히려 불필요한 교체가 발생할 수 있으며, 이를 완화하기 위해 근사 LRU 알고리즘(예: Clock 알고리즘)이 널리 사용된다.

⑤ LRU는 캐시 교체 시 가장 최근에 사용된 항목을 교체하는 방식이며, LRU는 효율적인 캐시 교체 정책일 수 있지만, 특정 상황이나 접근 패턴에 따라 최적의 선택이 아닐 수도 있다.

문제 6. 시간적 지역성과 공간적 지역성을 고려한 캐시 설계

시간적 지역성과 공간적 지역성은 각각 다른 메모리 접근 패턴을 설명하는데, 다음 중 이 두 개념과 캐시 설계에의 적용 원리를 올바르게 설명한 것은 무엇인가?

① 시간적 지역성은 과거에 참조된 데이터가 가까운 미래에 재참조될 가능성을, 공간적 지역성은 인접한 메모리 블록의 데이터가 함께 참조될 가능성을 나타내며, 캐시 설계 시 블록 크기 및 교체 전략에 항상 반영된다.

② 시간적 지역성은 인접 주소 간 연관성을 설명하며, 공간적 지역성은 동일 주소의 반복 참조를 의미하여, 캐시 설계에서는 블록 크기 조정보다는 교체 정책에만 영향을 미친다.

③ 시간적 지역성은 캐시 적중률 개선에 기여하지 않고, 공간적 지역성만이 데이터 전송 최적화를 위해 활용된다.

④ 시간적 지역성과 공간적 지역성은 서로 독립적인 현상이므로, 캐시 설계 시 일반적으로 두 가지를 고려해야 전체 성능이 최적화된다.

⑤ 공간적 지역성은 연속적 데이터 접근을 보장하지만, 시간적 지역성은 예측이 어렵기 때문에 캐시 설계에 적용하기 어렵다.

문제 7. 가상 메모리 관리와 TLB의 효율성

가상 메모리 시스템에서 TLB(Translation Lookaside Buffer)가 주소 변환 효율성을 향상시키는 이유와 한계에 대해, 다음 중 올바른 설명은 무엇인가?

① TLB는 전체 페이지 테이블 항목을 저장하여 주소 변환을 1:1 매핑하지만, 그 용량이 커지면 접근 속도가 오히려 느려진다.

② TLB는 최근에 참조한 주소 변환 정보를 임시 저장해 페이지 테이블 접근 시간을 획기적으로 단축시키지만, TLB 미스 시 전체 페이지 테이블 검색으로 인한 성능 저하가 발생할 수 있다.

③ TLB는 가상 메모리의 모든 페이지를 미리 로드해 높은 캐시 적중률을 유지하므로, TLB 미스 발생 빈도가 매우 낮아 성능에 거의 영향을 주지 않는다.

④ TLB는 소프트웨어적으로 관리되기 때문에 운영체제의 스케줄링 정책에 따라 주소 변환 성능이 크게 달라진다.

⑤ TLB는 물리 주소와 논리 주소 간의 복잡한 매핑을 단순화하나, 캐시 계층 구조와 직접적인 연관성은 없어 메모리 접근 최적화에 영향을 미치지 않는다.

1. 정답: 2 이유: MMU는 프로세스별로 할당된 실제 메모리 시작 주소(베이스)를 기록하고, 해당 프로세스가 접근 가능한 주소 범위를 한계 레지스터로 제한하여 주소 변환 시 범위 초과나 잘못된 접근을 방지합니다.

2. 정답: 2 이유: 페이징은 논리 주소를 동일 크기의 페이지로 분할해 물리 메모리의 프레임에 할당함으로써 내부 단편화를 줄이고, 세그먼테이션은 의미 단위(코드, 데이터, 스택 등)로 나누어 가변 크기를 허용하며 각 세그먼트에 대해 접근 권한 등 부가적인 제어가 가능합니다.

3. 정답: 3 이유: 캐시 메모리는 주 메모리보다 훨씬 빠른 접근 속도를 제공하며, 레지스터와 캐시 간 및 캐시와 주 메모리 간의 접근 시간 차이가 크기 때문에, 캐시 계층을 효율적으로 설계하는 것이 전체 메모리 성능 최적화에 매우 중요합니다.

4. 정답: 5 이유: Write-through 정책은 데이터 변경 시 캐시와 주 메모리에 동시에 기록하여 일관성을 유지하지만 성능에 영향을 주고, Write-back은 캐시 내에서만 수정 후 나중에 일괄 기록해 성능을 향상시키는 방식입니다.

5. 정답: 4 이유: LRU는 순차적 접근 시 불필요한 교체(thrashing) 현상을 유발할 수 있으므로, 이를 완화하기 위해 Clock 알고리즘과 같이 근사 LRU 방식을 적용하는 것이 일반적입니다.

6. 정답: 4 이유: 특정 상황에서는 하나의 지역성만 고려해도 성능을 최적화할 수 있지만, 일반적으로 두 가지 모두 고려하는 것이 성능 최적화에 유리합니다.

7. 정답: 2 이유: TLB는 최근 참조된 주소 변환 정보를 저장하여 매핑 시간을 단축시키지만, TLB 미스 발생 시 페이지 테이블 전체 검색이 필요해 성능 저하가 발생할 수 있는 한계가 있습니다.