19 Translation Lookaside Buffers

가상 주소 -> 물리 주소 변환을 위해 매핑을 읽어야하면, load/store 마다 메모리 읽기가 필요하며 이는 엄청난 낭비

변환-색인 버퍼 (Translation-lookaside buffer) TLB

TLB는 칩의 **메모리 관리부(memory-management unit, MMU)**의 일부이다. (주소변환 캐시 라고도함)

가상 메모리 참조시에 TLB에 원하는 변환 정복 있는지 확인하고 변환을 수행한다.

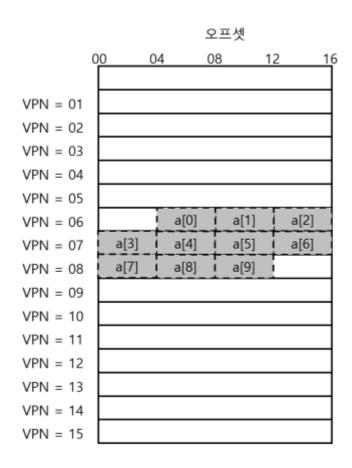
TLB의 기본 알고리즘

- 1. 가상 주소에서 VPN을 추출한다.
- 2. 만약 TLB 존재하면 TLB 히트로 TLB 항목에서 PFN을 추출한다.
- 3. 가상 주소 오프셋과 합쳐 원하는 물리 주소 (PA)를 구성해 메모리에 접근한다.

2-2 만약 TLB가 존재하지 않으면 저번장에 언급한 메모리 참조를 실행하고, 이를 TLB에 저장한다.

기본적인 캐시의 목표를 달성하는 것이 주 목표이다. (캐시에서 히트를 최대한 달성)

TLB 예제: 배열 접근



〈그림 22.2〉 예제: 작은 주소 공간 내의 배열

a[0], a[3], a[7] 을 접근할 때에만 TLB 미스가 일어나고, 남은 경우에는 TLB에 로드되어있어서 히트가 발생

히트율 70%

이는 배열의 **공간 지역성(spatial locality)**로 인해 성능이 개선되었다.

당연하게도, 페이지가 커지면 TLB의 효용성은 더 증가한다. (미스 횟수 감소)

만약 루프 후에도 배열을 사용한다면, **시간 지역성(temporal locality)** 로 인해 TLB의 히트율이 높아진다 (단 배열 전체를 저장할 정도로 크다면). 이는 한번 참조된 메모리 영역이 짧은 시간 내 재참조 되는 현상을 의미한다.

본질적으로 캐시인 TLB의 성공 여부는 이 두 공간 지역성, 시간 지역성에 달려있다.

TLB 미스

- 하드웨어가 처리하는 경우
 - ㅇ 하드웨어에 페이지 테이블에 대한 명확한 정보를 가지고 있음
 - ㅇ 페이지 테이블에서 원하는 엔트리 찾음
 - ㅇ 필요한 변혼 정보 추출
 - o TLB 갱신, 미스 발생 명령어 재실
- 운영체제가 처리
 - o TLB 미스 처리 트랩 핸들러를 추가해서 관리한다.
 - 시스템 콜 호출 트랩 핸들러와 다르게, 리턴 후 시스템 콜 호출 명령어를 재 실행 해야한다. (기 본적으로는 다음 명령어) -> 이는 운영 체제가 트랩 종류에 따라 현재 혹은 다음 PC값을 저장해 야 함을 의미
 - o TLB 미스 핸들러 실행시에 TLB 미스가 반복되지 않도록, TLB 미스 핸들러를 핸들러의 물리 메모리에 위치시킨다.
 - o 혹은 일부분을 핸들러 코드 주소를 위해 영구히 할당 (연결(wired) 변환)
 - ㅇ 상대적으로 유연하다.

TLB 구성

VPN | PFN | other

• 만약 VPN이 같으면, 무슨 프로세스를 위한 PFN을 처리?

VPN	PFN	valid	\mathbf{prot}
10	100	1	rwx
		0	
10	170	1	rwx
		0	

문맥 교환 수행시에 다음 프로세스를 실행하기 전 기존 TLB 정보를 비우는 것으로 해결 하지만, 그때마다 TLB 미스 발생 이를 개선하기 위해 프로세스 식별자(PID)와 유사한 **주소 공간 식별자(ASID)** 필드를 추가해 관리한다.

VPN	PFN	\mathbf{valid}	\mathbf{prot}	ASID
10	100	1	rwx	1
		0		-
10	170	1	rwx	2
		0		-

• PFN이 동일하다면?

VPN	PFN	valid	\mathbf{prot}	ASID
10	101	1	r-x	1
		0		-
50	101	1	r-x	2
		0		_

TLB 교체

캐시 교체 정책의 구체적인 부분은 다음 장에서.

일반적으로 최저 사용 빈도(LRU) 항목을 교체해 메모리 참조 패턴 지역성을 최대한 활용해야한다. 혹은 random 교체 정책