Paging - Introduction

- Paging : Address Space를 고정된 크기를 가진 페이지로 나눔
- 페이징을 하면 물리 메모리도 Page Frame이라는 단위로 나눠짐
- 프로세스별로 페이지 테이블(Virtual Address -> Physical Address)이 필요함

Advantages

- 유연성 : Address Space의 추상화가 효율적 -> Heap/Stack이 어느방향으로 커지는지 생각할 필요 없음
- 간단함 : Free-Space의 관리가 쉬움 Address Space의 Page Size와 물리적 메모리의 Page Frame Size가 같음 할당이 쉽고, free list 관리가 쉬움

Address Translation

- 가상 주소에 두가지 구성요소가 있음 : VPN(Virtual Page Number), Offset
- 물리 메모리의 주소로 바꿀때 VPN을 PFN(Page Frame Number)로 대체

Page Table이 저장된 곳

- Page Table은 그 사이즈가 지나치게 커질 수 있음 ex) 32bit address space / 4KB pages / 20 bit VPN -> Page Entry 하나가 4 byte라고 치면 2^20 * 4 = 4MB
- 페이지 테이블은 메모리에 저장

구성요소

- 페이지 테이블은 Virtual Address -> Physical Address를 매핑파는 자료구조 ex) Linear Page Table / array
- 운영체제가 VPN에 따라 배열을 인덱싱하고, PTE(Page Table Entry)를 찾아본다

PTE의 일반적인 Flag

- Valid Bit : 해당 엔트리가 Valid 한지
- Protection Bit : 페이지가 Read/Write/Execute 될 수 있는지
- Present Bit : 페이지가 Physical Memory에 있는지 Disk에 있는지
- Dirty Bit : 페이지가 메모리에 불려온뒤로 수정 됐는지
- Reference Bit(Accessed Bit) : 페이지가 접근 됐는지

Paging의 속도

- PTE의 위치를 찾기 위해서 페이지 테이블의 시작 주소를 알아야 함 -> 한번의 메모리 레퍼런스를 할 때마다 운영체제는 한번의 메모리 레퍼런스 연산을 더 더 해야함

Translation Lookaside Buffers

- TLB : CPU 또는 IC Chip에 있는 MMU의 일부
- Virtual -> Physical 주소 변환의 가장 많이 사용되는 하드웨어 캐시
- Spatial Locality 적용 -> 성능 향상

Algorithm

1. VPN을 추출해서
2. TLB가 해당 VPN에 관한 변환법을 가지고 있는지 확인
3. [TLB HIT] TLB entry에서 PFN을 가져와, 해당하는 물리주소를 가져오고 메모리에 접근
4. [TLB Miss] 하드웨어가 페이지 테이블에 접근해 변환주소를 찾음
5. TLB를 업데이트 함

Locality

- Temporal locality : 최근에 접근된 명령어/데이터아이템은 곧 다시 접근될 가능성이 높음
- Spatial locality : 프로그램이 주소 X에 접근한 뒤에, 곧 x 주변에 있는 메모리에 접근할 가능성이 높음

TLB Miss Handle

CISC

- 하드웨어가 CISC에서 TLB Miss를 handle함
- 하드웨어가 Page Table이 메모리 어느곳에 위치하고 있는지 정확히 알아야 함
- 하드웨어가 Page Table을 걸으면서 맞는 PTE를 찾아서 변환주소를 추출하고, 업데이트 한 뒤 같은 명령 반복
- Hardware-managed TLB

RISC

- TLB Miss가 발생하면 하드웨어가 Exception을 던짐 -> Trap Handler -> OS가 이를 확인하고 Trap handler 가동(TLB Miss에 관한)
- Software-managed TLB

TLB Entry

- TLB는 Full Associative(완전 연관) 방식으로 관리 됨
- 주로 32/64/128개의 Entry로 구성
- 하드웨어가 평행적으로 전체 TLB를 검색해서 원하는 변환을 찾아냄
- 다른 비트들 : Valid bits, Protection bits, Address-space identifier, dirty bit
- VPN + PFN + Other bits로 구성

문제점 : Context Switching

- TLB가 프로세스별로 저장되는게 아니기 때문에, Context Switching을 하게되면 VPN이 겹치는 수가 있음 -> 무슨 프로세스의 Entry인지 알 수가 없음
- 해결법 : ASID(Address Space Identifier) 필드를 TLB에 추가해 Address Space를 구분

다른 케이스

- 두개의 프로세스가 물리적 페이지를 공유하고 있을 때
- 이렇게 물리적 페이지를 공유하는 것은 유용하고, 사용하고 있는 물리페이지 숫자를 줄인다

TLB Replacement Policy

- LRU(Least Recently Used) : 사용한지 오래 된 엔트리를 없앰 -> Locality를 생각한 결정

실제 TLB Entry

- VPN + PFN + Global Bit(프로세스들이 공유하는 페이지) + ASID + Coherence Bit(하드웨어가 페이지를 어떻게 캐시할지 결정) + Dirty Bit + Valid Bit