학습목표

- CPU 처리 속도를 향상 시키기 위한 방법에 대해 설명할 수 있다.
- 캐시에 관한 구성에서 부터 매핑하는 방법에 대해 설명할 수 있다.
- 캐시 관련 알고리즘들을 이해할 수 있다.

학습내용

- Cache 일반
- Mapping
- Replace Algorithm 및 Write Policy
- Quiz, PBL, 탐구주제

Cache의 특징

- CPU와 MM의 속도 차이로 인한 CPU 대기시간을 최소화시키기 위하여 CPU와 MM 사이에 설치하는 High Speed Semiconductor Memory (보통 CPU 내부에 On-Chip Cache의 형태로 설치)
  - MM보다 Access Speed ↑, Cost ↑, Capacity ↓
  - Hit Ratio(H) = Cache에 Hit되는 횟수/전체 기억장치 Access 횟수
  - Miss ratio = 1 - H
  - Average Memory Access Time(T_a)=H×T_c+(1-H)×T_m
    [예] Average Memory Access Time : T_c=50ns, T_m=400ns

H	70%	80%	90%	95%	99%
Ta	155ns	120ns	85ns	67.5ns	53.5ns

  - Cache Hit Ratio가 높아질수록 Average Memory Access Time은 Cache Access Time에 접근

Cache의 특징

- Cache Hit Ratio는 Program과 Data의 Locality에 크게 의존
  - Temporal Locality : 최근에 Access된 Data가 가까운 미래에 다시 Access 될 가능성이 높음
  - Spatial Locality : 인접하여 저장되어 있는 Data들이 연속적으로 Access 될 가능성이 높음
  - Sequential Locality : Branching이 발생하지 않는 한, Instruction들은 저장된 순서대로 인출되어 실행됨
- Cache Design Objectives
  - Hit Ratio의 Maximization= Access Time의 Minimization= Miss에 따른 Delay Time의 최소화
  - Main Memory와 Cache간의 Data Consistency 유지 및 그에 따른 Overhead Minimization

Hierarchical Cache

Hierarchical Cache

- On-Chip Cache를 L1 Cache로 사용하고, Chip 외부에 더 큰 Capacity의 L2 Cache를 설치하는 방식
- L2는 L1의 Super-Set : L2의 Capacity가 L1보다 크며, L1의 모든 내용이 L2에도 존재
- L1은 속도가 빠르지만, Capacity가 작기 때문에 L2 보다 Hit Ratio은 더 낮음
- 먼저 L1을 검사하고, 만약 원하는 정보가 L1에 없다면 L2를 검사하며, L2에도 없는 경우에만 Main Memory를 Access
- 평균 Storage Device Access 시간 :
  $T_a = H_1 \times T_{C1} + (H_2 - H_1) \times T_{C2} + (1 - H_2) \times T_m$
- [예] L1/L2/MM Access Time: 20/60/200ns,
  L1/L2 Hit Ratio: 0.8/0.95
  → T_a = 0.8×20ns + (0.95-0.8)×60ns + (1-0.95)×200ns
  = 35ns

Split Cache

Split Cache

- Cache를 Instruction Cache와 Data Cache로 분리
- Instruction Fetch와 Execution 간에 Cache Access 충돌 현상 제거
- 대부분의 고속 Processor(Pentium계열)에서 사용

Fetch

- Fetch 방식 : MM → Cache
  - Cache의 Capacity가 커질수록 Hit Ratio가 높아지지만,
    Cost가 증가, Address Decoding 및 Data Fetch를 위한
    주변 회로가 더 복잡해지기 때문에 Access Time이 다소 길어짐
  - Demand Fetch 방식 : 필요한 정보만 인출해 오는 방법
  - Prefetch 방식 : 앞으로 필요할 것으로 예측되는 정보도 미리 인출,
    Locality가 높은 경우에 효과가 높음

Mapping(1)

Mapping

- 어떤 MM의 Block들이 어느 Cache Slot을 Share할 것인지를 결정해 주는 방법
- 00000부터 11111까지의 번호(MM의 주소)가 할당된 32명의 학생(MM에 저장된 데이터)이 000부터 111까지의 슬롯(or 세트)번호가 적혀있는 8개의 칸막이 화장실(Cache)을 공평하게 이용하는 방법

Mapping(2)

Mapping의 종류

Fully-Associative Mapping	● 32명중 급한 학생은 누구나 8개의 칸막이 화장실 중 하나를 점유하게 하는 방법
Direct Mapping	● 32명의 학생을 4명씩 8개의 그룹으로 나누고 각 그룹이 사용할 수 있는 화장실을 하나씩 할당하는 방법
Set-Associative Mapping	● 두 가지 방식의 혼합으로서 32명의 학생을 8명씩 4개의 그룹으로 나누고 각 그룹이 사용할 수 있는 화장실을 두 개씩 할당하는 방법

Mapping(3)

- Cache에서 어떻게 데이터를 읽어낼 것인가?

Fully-Associative Mapping	● 화장실을 사용할 때 학생들의 번호(00000~11111: Tag)를 화장실 문에 고지하여 개인을 확인
Direct Mapping	● 화장실을 사용할 때 학생들의 번호 중, 앞 두 자리(00~11: Tag)를 화장실 문에 고지하고 칸막이 화장실 번호(000~111: Slot)와 붙여서 개인을 확인
Set-Associative Mapping	● 화장실을 사용할 때 학생들의 번호 중, 앞 세 자리(000~111: Tag)를 화장실 문에 고지하고 칸막이 화장실 번호(00~11: Set)와 붙여서 개인을 확인