Embedded HW Structure

DMA: 주인인척 하는 애 , 끝났습니다! 하는 메시지 interrupt

Memory Systems

역할

- 컴퓨터 프로그램 작업공간
- 영구 저장소
- 추상화
  - 프로그래밍을 할 때 쉽게 메모리를 사용할 수 있도록 abstraction을 제공
- 구현
  - single type의 메모리 디바이스에서는 구현 불가능
  - memory hierachry (계층적 메모리)로 잘 구현 가능
  - 메모리 하위 시스템의 집합으로 구성
    - caches, DRAM, disk, ,,,

Memory Hierarchy(계층적 메모리)

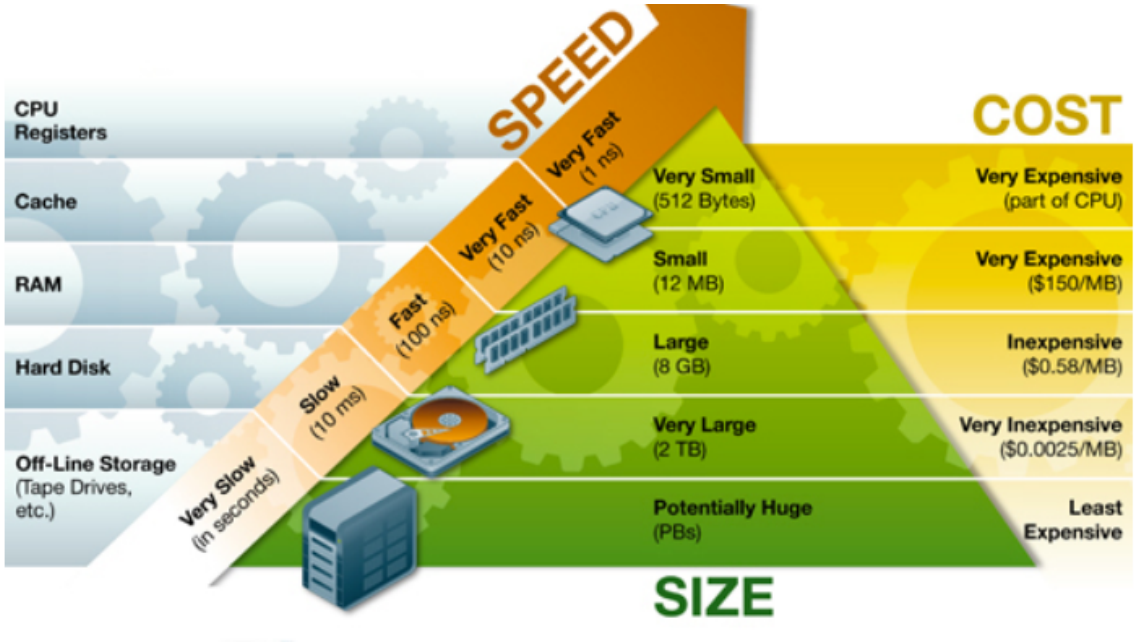
- 궁극적 목표: 속도와 용량 동시 달성
  - Speed: 가장 빠른 성능 제공
  - Capacity: 가장 저렴하게
- Locality of reference (지역성의 원리)
  - Temporal Locality (시간 지역성)

    한 메모리 위치를 참조하면 가까운 미래에 다시 참조될 가능성이 높다.
  - Spatial Locality (공간 지역성)

    한 메모리 위치를 참조하면, 가까운 미래에 그 이웃이 참조될 가능성이 높다.

    단위는 512byte

미래를 안다면 제일 효율적인 메모리 관리 방법: optimal



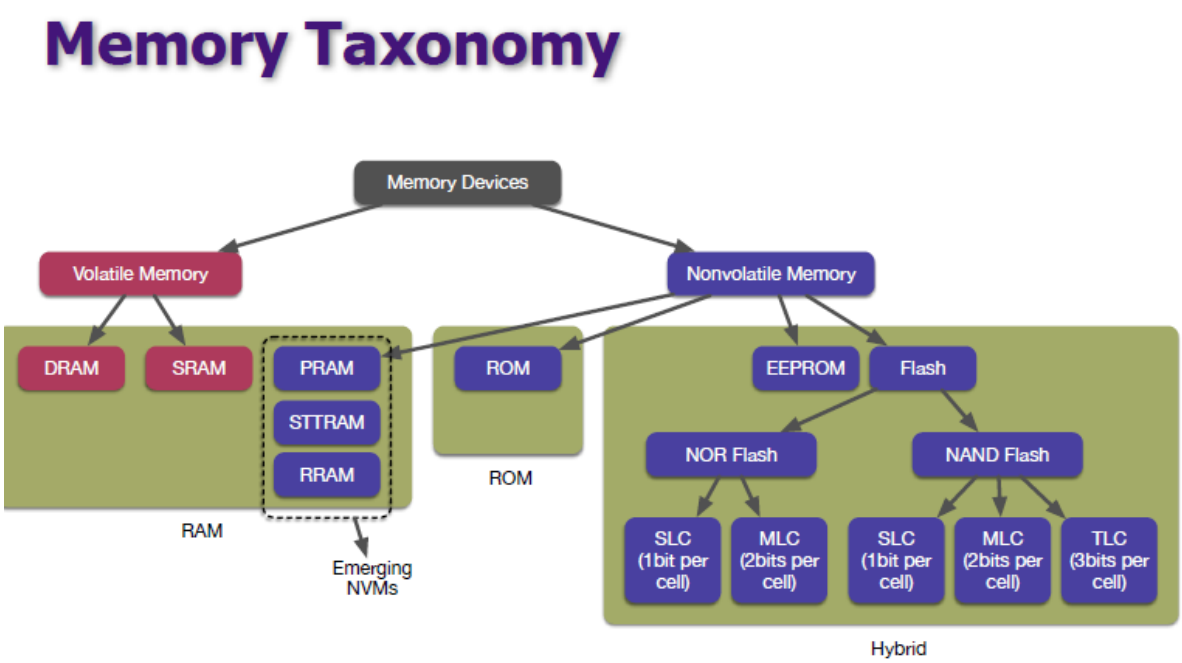
메모리 --- storage : 페이지단위: 4KB

cache (hit, miss 둘다 spacial locality)

hit: 32bit, or 64bit (= 8byte) 단위: word

miss: 512bit (=64byte) 단위: cache line

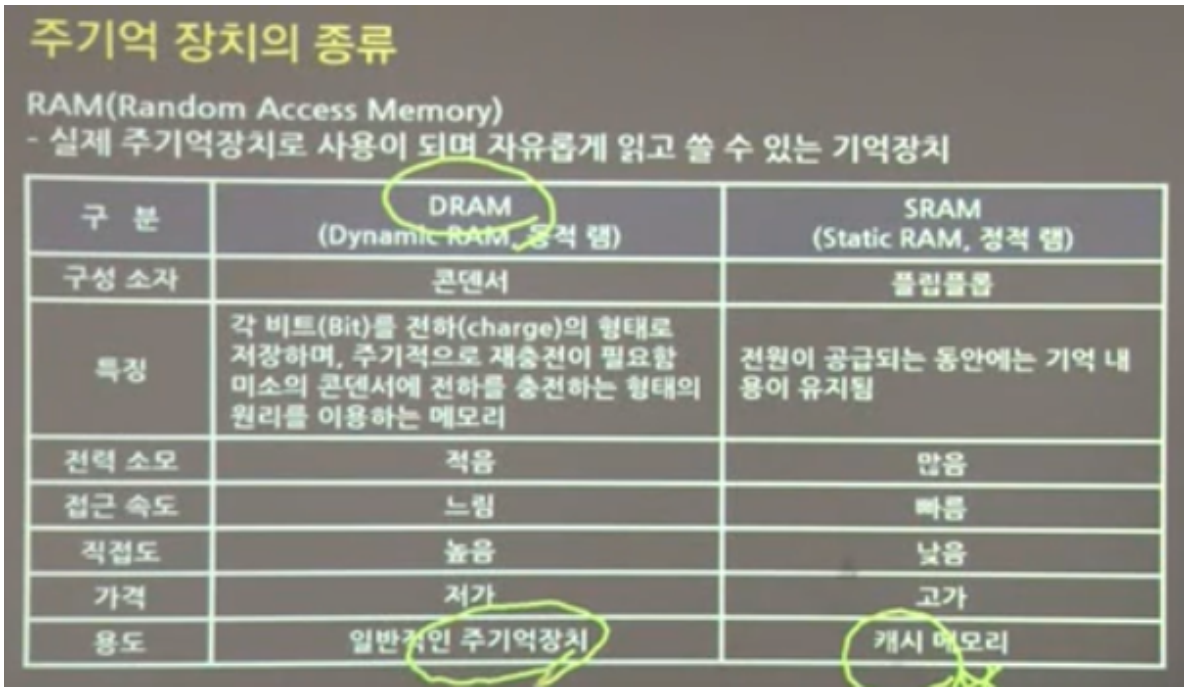
Memory Taxonomy(분류법)



ROM: 기본 입출력 시스템 (BIOS), 자가 진단 프로그램 (POST) 과 같이 변경 가능성이 없는 시스템 소프트웨어

SRAM(Static RAM)

DRAM (Dynamic RAM)



SRAM	DRAM
4개의 트렌지스터로 구성	1개의 트랜지스터와 메모리가 저장되는 capacitor로 구성
휘발성	휘발성
전원 공급하면 재충전(refresh)필요 x	전원 공급 되더라도 주기적인 재충전(refresh)필요 o
매우빠름(~tens of ns)	빠름(~hundreds of ns)
비쌈	쌈
	읽으면 내용이 파괴되어 읽은 후에 다시 써야한다.

Flash Memory

- 원리
  - electrons trapped in the floating gate(플로팅게이트에 전하를 저장함으로써 데이터를 저장)
- 특성
  - 비휘발성
  - 큰 전압으로 인해 지워질 수 있음
  - 읽기/ 쓰기 페이지 단위로 실행
- 한계 (out place 방식)
  - 블록 단위로 지워야함

    다른 페이지로 이동하지 않고 변경 될 수 없다. = in-place update 불가능
  - 덮어 쓸 수 x -> 모든 블록을 지우기 전까지 해당 자료를 변경 할 수 x
  - garbage collection 필요 이유

    free 공간이 필요할 때 자동적으로 내부 명령을 실행해서 garbage collection을 실행하면서 빈공간 유지

Multi-Level Cell Flash

- 정의: 한 셀 당 1비트 이상의 정보를 저장

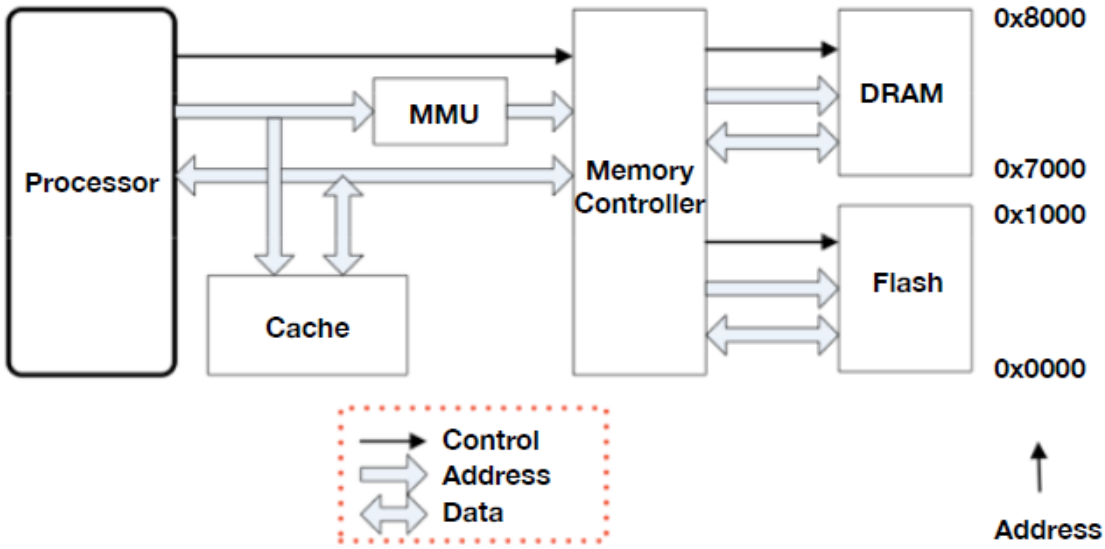
  2비트 -> 4가지로 표현 가능
- 특성

  • 비트당 비용 절감 (= 똑같은 가격으로 기술 상승) • 프로그램 속도 저하 • 읽기 속도 느림 • 쓰기 내구성 감소 -> dead cell • 데이터 보존 시간 단축
  - MLC -> TLC로 넘어갈 때 좋아지는 것은 lower cost per bit 단 한개뿐

    따라서 MLC > TLC

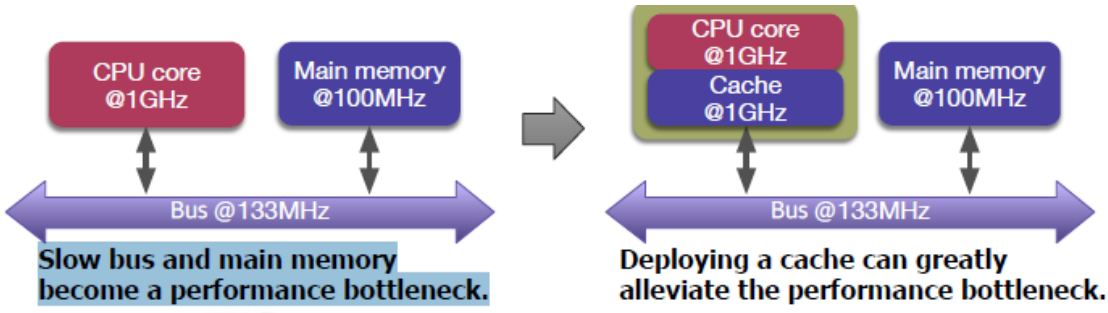
Memory System Structure

Memory System Structure



Cache Memory

- 많이 찾을 것 같은(알고리즘으로 처리) 데이터 저장
- 소규모&고속 메모리
- SRAM으로 구성



1. CPU는 빠른데 메인메모리 처리 속도는 느려서 병목현상이 일어남
2. 캐시 구축하면 현상 완화

Cache Performance

- cache 목표: 최대한 빨리 CPU에 명령어와 데이터 제공
- Cache hit: 요청된 데이터가 캐시에서 발견
- Cache miss: 요청된 데이터를 캐시에서 찾을 수 없음
- miss penalty: 메인 메모리에서 cache line을 가져오는데 걸리는 시간
- Access Time = (hit time) + (miss rate)*(miss penalty)

  (1 - miss rate)*(hit time) + (miss rate)(miss penalty + hit time)

Cache Writing Policy

- Write hit
  - Write through

캐시 내용 기록함과 동시에 주기억장치 값도 변경

항상 캐시의 내용과 주기억장치의 내용을 동일하게 =주기억장치가 항상 최신

단: 주기억 장치의 접근 횟수가 많아짐

- Write back

  캐시에 먼저 기록하고 나중에 주기억장치에 (관련 캐시 블록이 캐시에서 제거될 때)

  어느 순간의 시점에서 캐시의 내용과 주기억장치의 내용이 다를 수 있음

  따라서 메모리를 이용하는 입출력 장치는 항상 캐시를 통해야한다.

- Write miss
  - Write allocate

    miss된 캐시 블럭을 먼저 캐시에 로드(overwrite)하고, hit
  - No-write allocate

    misse된 캐시 블록이 캐시에 로드되지 않는다. 메인메모리에 직접 기록됨

MMU (Memory Management Unit)

- 주소 변환

  가상 메모리 --> 물리적 메모리
- 메모리 보고

  프로세스가 할당되지 않은 메모리에 접근하지 못하도록 방지

Memory Controller

- 역할: 메인메모리로 오고가는 데이터 관리

  즉, 예를 들자면, 도서관에 책이 꼽혀있는 상태를 그냥 메모리라고 한다면, 사용자가 이런책 찾아주세요 혹은, 책을 반납했을 때 그게 있어야 할 위치에 꼽아주는 작업을 하는 사서가 일종의 컨트롤러라 보시면 됩니다.
- 기능
  - 주소 decode
    - target memory device 결정
    - 적절한 memory control signal을 주장
  - 주소 multiplexing (DRAM을 위한)
    - 주소를 어떻게 접근할지 제어: 행,열로 나눠서 특정 자기 배열자료로
  - 주기적으로 RAM의 내용을 refresh (DRAM을 위해)

I/O Devices

- 주소 버스, 데이터 버스, 제어 버스 사용: 메모리 장치와 유사
- I/O mapped

  I/O 장치를 위한 전용공간
- Memory mapped I/O

  메모리와 io 장치는 같은 주소 버스를 씀
- I/O memory 영역은 "캐시할 수 없음"으로 구성되어야함

  캐시를 사용하면 입출력 장치의 내용이 변경되었음에도 변경된 내용을 가져오지 못하고 최근 엑세스한 내용만 나올 것이기 때문에 "non-cacheable"로 구성되어야한다.
- I/O variables는 "휘발성"으로 선언해야함

이유: 컴파일러에 의한 임의적인 최적화를 방지한다.

I/O Resource Management

- Poling
  - CPU가 다른 장치의 상태를 일정한 조건을 만족할 때 까지 주기적으로 검사하는 방식
  - CPU의 busy waiting = CPU의 자원이 낭비
- Interrupt
  - 노예가 "주인님 저 다 일 마쳤습니다. 여기 한번만 봐주세요"
  - 입출력 장치나 예외상황이 발생하여 처리가 필요할 경우에 CPU에게 처리해달라고 요청하는 개념.
- DMA (Direct Memory Access)
  - CPU의 개입 없이 I/O 장치와 메모리간의 데이터 전송
  - CPU의 개입 없이 주변 장치들이 메모리에 직접 접근하여 읽거나 쓸 수 있도록 하는 기능

Interrupt Interface

interrupt controller가 interrupt를 모으고, 제어: interrupt가 마구잡이로 올 수 있음

Interrupt Flow Control

ISR: Interrupt Service Routine

Interrupt Vector

지금은 가볍게, 나중에 할 것임

- interrupt Vector table
- fixed interrupt
- vectored interrupt

DMA (Direct Memory Access)

별 설명 안함

Bus

전에 설명해서 안할거임