chap2

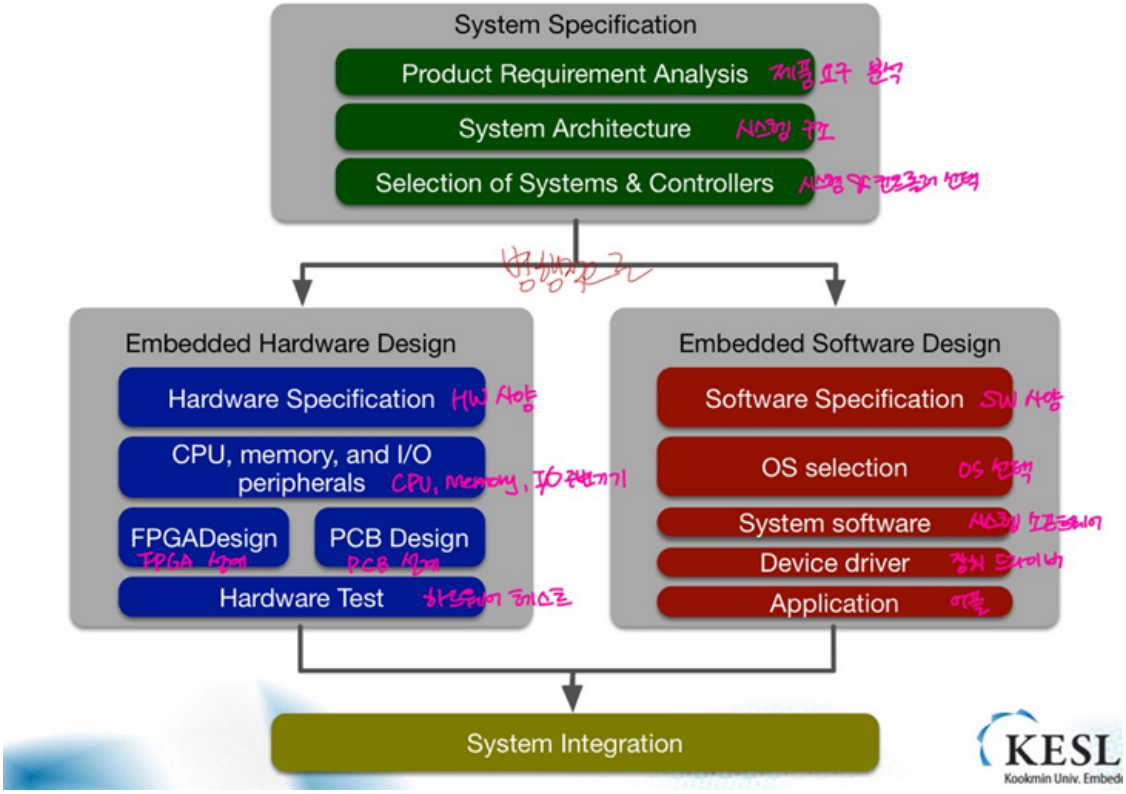
- mission critical: 필수 불가결한 요소, 작은 function의 오류만으로도 전체 시스템에 치명적인 결과를 초래하는 분야를 말함
- RTOS(Real Time Operating System): 원하는 작업을 시간 내에 처리할 수 있는 걸 보장하는 운영체제
- hard real time: 경성 실시간 프로그램, 시스템이 주어진 시간에 작업을 완료하지 못하면 막대한 피해를 줌
- soft real time: 연성 실시간 프로그램, 시간적 제약 조건을 지키지 못하더라도 피해가 거의 없다. 처리 결과에 의미가 있는 경우
- IFTTT(if this then that): 타 소프트웨어를 관리할 수 있도록 도와주는 프로그램
- 임베디드 시스템 정의
    - 전통적: 범용x 특수 목적을 제어하기 위해 제작된 시스템
    - 오늘날: 범용적 목적과 임베디드 시스템의 경계가 모호해짐
        - 운영체제 사용
        - 전력 소비량
        - 성능
- 임베디드 시스템 응용
    - Control system: factory automation, home automation
    - Automotive system: Car embedded system
    - Medical system: medical imaging devices
    - Portable devices: Phone, camera
    - Network system: Routers
- 임베디드 특성
    - 낮은 성능의 CPU
    - 사전에 정의된 기능
    - 가벼움, 저전력, 저비용
    - mission critical
    - 거의 hard real time

chap3

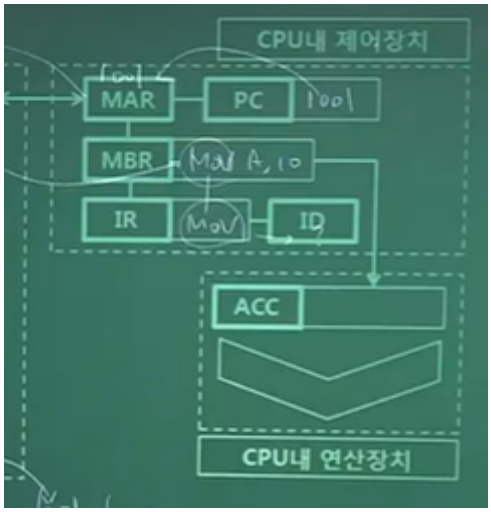
- PCB (printed circuit board): 저항기, 집적 회로 등의 전자 부품을 표면에 고정하고 연결시켜 전자 회로를 구성한 판
- Soc(System on chip): CPU를 포함한 다양한 시스템 구성요소를 포함하는 단일 칩
- Host system: embedded 소프트웨어가 개발되는 위치
- Target system: embedded 실행파일이 실행되는 위치
- Cross compiler: Embedded System의 특성상 큰 용량의 저장 장치를 가지지 못하기 때문에 host system에서 embedded 소프트웨어를 개발하여 target system에서 실행파일을 실행시킨다.
- on chip bus: cpu, 메모리, gpu 등이 하나의 칩 안에 들어있는 형태, 규모 및 복잡도가 커지게 된다.
- Register: CPU가 요청을 처리하는 데 필요한 데이터를 일시적으로 저장하는 기억장치 (general purpose 레지스터, control 레지스터, status 레지스터)
- ALU(Arithmetic Logic Unit): CPU에서 산술연산, 논리연산, 비트연산을 수행하는 연산장치
- 기계어: CPU의 decoder로 인식 가능한 언어
- ISA(Instruction Set Architecture): 연산자, 피연산자(레지스터, 메모리, immediate value로 구성)

- pipeline: 명령어 처리를 여러 단계로 나누어 병렬로 처리하는 구조
- BUS: 두개 이상의 장치를 상호 연결하는 통신 경로
  - 내부버스

    레지스터와 ALU 사이의 bus
  - 외부 버스
    - data 버스: 프로세서와 외부간에 데이터 값 전송 (양방향)
    - address 버스: 주소값 전송 (단방향)
    - control 버스: 명령 또는 제어 신호를 전송 (단방향)
- Embedded systems design workflow



- CPU 구성
  - 레지스터
  - ALU(Arithmetic logic unit)
  - CU(control unit)
  - processor bus
- CU 구성



- CISC(complex instruction set computer) vs RISC(reduced instruction set computer)

구 분	CISC	RISC
구조	복잡한 구조	단순한 구조
구성	복잡, 많은 명령어	간단, 최소 명령어
명령어 길이	다양한 길이	고정된 길이
레지스터	작음	많음
속도	느림	빠름
용도	개인용 컴퓨터	서버, 워크스테이션

- 폰노이만 vs 하버드
  - 폰노이만
    - 데이터 메모리와 프로그램 메모리가 구분되어있지 않고 하나의 버스를 가지고 있는 구조
    - 명령과 데이터를 동시에 전송할 수 없다.
  - 하버드
    - 데이터 메모리와 프로그램 메모리가 구분되어있다.
    - 데이터와 명령을 동시에 전송할 수 있다.
    - pin의 수가 더 많음 32 * 4 = 128pin

chap4

- memory system: 단일 메모리는 효율적인 메모리를 구현할 수 없음(CPU 1 clock에 read/write 수행) 따라서 캐시, DRAM, 디스크 등 계층적 메모리로 구현
- memory hierarchy: 메모리 계층 구조, 싸고 빠른 컴퓨터를 목표로, 여러가지 기억장치를 속도, 용량, 성능에 따라 계층적으로 분류
- RAM(random access memory): 읽고 쓰기가 가능한 주기억장치, 전원을 끄면 데이터가 지워진다.
- ROM(read only memory): 읽기 전용의 주기억장치, 전원을 꺼도 데이터가 지워지지 않는다.
- cache memory: CPU의 처리속도와 주기억장치의 접근 속도 차이에 따른 병목현상을 줄이기 위해 사용하는 소규모 고속 메모리
- Access Time = hit time + (miss rate) * (miss penalty)
- MMU (Memory Management Unit): 가상 메모리 주소를 실제 메모리 주소로 변환하며, 할당되지 않은 메모리에 대한 접근 보호를 담당
- MC (Memory Controller): 메인메모리로 오고가는 데이터 관리를 담당 ex) 도서관 사서
- I/O-mapped I/O (= port-mapped I/O): 메모리와 I/O 주소 공간을 분리하여 엑세스 하는 방식
- MMIO(memory mapped I/O): CPU가 I/O 장치를 엑세스 할 때, I/O와 메모리 주소 공간을 분리하지 않고 하나의 메모리로 취급하여 배치하는 방식, 대부분 이 방법을 많이 사용
- Polling: CPU가 다른 장치의 상태를 일정한 조건을 만족할 때 까지 주기적으로 검사하는 방식 (busy waiting = CPU의 자원 낭비)
- Interrupt: I/O 장치나 Exception이 발생하여 처리가 필요할 경우에 CPU에게 처리해달라하고 요청하는 것
- DMA(Direct Memory Access): CPU의 개입 없이 I/O 장치와 메모리 간의 데이터 전송
- Interrupt controller: 인터럽트 우선순위를 인터럽트 출력에 할당하는 담당을 하거나, 인터럽트 병합하여 출력을 담당
- ISR(Interrupt Service Routine) = interrupt handler: 접수되는 인터럽트에 대응하여 특정 기능을 처리하는 기계어 코드 루틴
- Interrupt vector: 인터럽트가 발생했을 때, 그 서비스 루틴들의 주소를 가지고있는 공간
- fixed interrupt: ISR의 주소가 고정된 위치에 있음
- Vectored interrupt: 우선순위 부여 방식으로 ISR의 주소가 고정되어 있지 않음, I/O장치 상태변화에 사용

- Locality of reference (참조의 지역성) - 메모리 계층 설계의 핵심
  - Temporal Locality (시간 지역성)

    한 번 접근이 이루어진 메모리 영역은 가까운 미래에 다시 접근될 확률이 높다는 경향
  - Spatial Locality (공간 지역성)

    한 번 접근이 이루어진 메모리 영역의 주변은 가까운 미래에 접근될 확률이 높다는 경향
  - optimal

    미래를 안다면 제일 효율적인 메모리 관리 방법
- memory hierarchy: 메모리 계층적 구조

	speed	**size**	**cost**
cpu	very fast	512 byte	very expensive
cache	very fast	12MB	very expensive
RAM	fast	8GB	inexpensive
hard disk	slow	2TB	very inexpensive
off line storage	very slow	PB	least expensive

- cache hit, miss 단위

	size	**단위**
hit	32 or 64bit (= 8byte)	word
miss	512bit (= 64 byte)	cache line

- SRAM(static random access memory) vs DRAM(dynamic random access memory)

SRAM	**DRAM**
4개의 트랜지스터	1개의 트렌지스터, 1개의 capacitor
휘발성(전원 공급x시)	휘발성(전원 공급x시)
stable(전원 들어와있으면 데이터 유지)	unstable (주기적인 refresh가 없으면 데이터 사라짐)
매우 빠름 (tens of ns)	빠름 (hundreds of ns)
매우 비쌈	쌈
	읽으면서 데이터 파괴하기 때문에, 읽은 후 다시 써주어야함

- Flash memory
  - 원리: electorns trapped in the floating gate (floating gate에 전하를 가둠으로써 데이터를 저장)
  - 특성
    - 비휘발성
    - 큰 전압으로 인해 데이터가 지워질 수 있음

        - 페이지 단위로 읽기/쓰기 (1KB~4KB)
        - 블럭 단위로 지우기 (64 page) -> 지우는 횟수도 한계가 있음
        - out place update
            - garbage collection 필요 이유!!

            읽기/쓰기는 페이지 단위로 발생하지만, 지우기는 블록 단위로 발생하기 때문에 in place update, 즉 덮어쓰기가 불가능하다. 따라서 플래시 디스크는 데이터가 업데이트 될 때, 해당 데이터를 지우고 다시 쓰거나, 새로운 공간에 데이터를 저장해야한다. garbage collection을 통해 데드 페이지들이 차지하고 있는 공간을 확보해 메모리 공간을 효율적으로 사용할 수 있게 한다.

- MLC (Multi-Level Cell Flash)
    - 정의: cell 당 2비트 이상 저장
    - 특성
        - 비트당 비용 절감
        - 프로그램 속도 저하
        - 읽기 속도 느림
        - 쓰기 내구성 감소
        - 데이터 보존 시간 단축
- Cache Writing Policy
    - Write hit

write through	**write back**
캐시 내용 기록과 동시에 주기억 장치 값도 변경	캐시에 먼저 기록하고, 나중에 주기억장치에(관련 캐시 블록이 캐시에서 제거될 때) 기록
단: 주기억장치의 접근 횟수 늘어남	단: 메모리를 이용하는 입출력 장치는 항상 캐시를 통해야함

    - Write miss

write allocate	**no write allocate**
miss 된 캐시 블럭을 먼저 캐시에 로드 (overwrite) 하고, hit	miss 된 캐시 블록을 캐시에 로드 (overwrite) 하지 않고, hit

- MC(memory controller)의 기능
    - 주소 decode
        - target memory device 결정
        - 적절한 memory control signal 주장
    - 주소 multiplexting
        - 주소를 어떻게 접근할지 제어: 행, 열로 나눠서 특정 배열 자료로 제어
    - 주기적인 읽기/쓰기를 통해 RAM의 내용을 refresh
- MMIO 주의사항
    - I/O 메모리 영역은 "non-cacheable(캐시할 수 없음)"로 구성되어야함

      캐시를 사용하면 I/O 장치의 내용이 변경되었음에도 변겨오딘 내용을 가져오지 못하고 최근 엑세스한 내용만 나올 것이기 때문에 "non-cacheable"로 구성되어야한다.
    - I/O 변수는 "volatile(휘발성)"로 선언되어야함

컴파일러에 의한 임의적인 최적화를 방지한다. 여기서 임의적인 최적화란, 동일한 주소에 대해 중복되는 쓰기 명령을 없애는 행위를 말한다.

chap5

- ARM (Acron RISC Machines // Advanced RISC Machines)
- Exception: 프로세서 내부에서 발생하는 예측하지 못한 상황을 알리는 목적의 시그널
- CPSR(Current Program Status Register): 현재 프로그램 상태를 나타내는 레지스터
- SPSR(Saved Program Status Register): 현재 프로그램 상태를 저장하는 레지스터
    - CPSR, SPSR은 구조가 같다 (ALU계산 결과 상태 flags, 인터럽트 활성화 제어 flag, 동작모드 등을 제어 flag)

- Programmer's model (DORIE)
    - Data access architecture
    - Operation mode
    - Register architecture
    - Instruction set
    - Exception handing mechanism
- Instruction 예시
    1. SUB R0, R1, R2 (OPcode Destination register, Operand1, Operand2)
    2. LDR R0, [R4, R5] (OPcode Destination, [Base, Offset]) //(LDR / STR에서만 메모리 엑세스 가능 RISC기 때문)
    3. STR R0, [R4, R5] (OPcode Destination, [Base, Offset])
- Pipeline 단계
    1. Fetch: 명령어 fetch
    2. decode: 명령어 해동
    3. execute: 연산
    4. memory : memory access(cpu에서는 break가 걸림)
    5. write: register write
- Operating Mode
    - USR(User Mode): 사용자 프로그램을 동작시키는 가장 일반적인 모드
    - FIQ(Fast Interrupt reQuest): fast interrupt 신호가 하드웨어에서 발생했을 때 변경되는 모드
    - IRQ (Interrupt ReQuest): 일반적인 interrupt 신호가 하드웨어에서 발생했을 때 변경되는 모드
    - SVC(Supervisor): 운영체제 등에서 시스템 코드(SWI)를 수행하기 위한 보호모드, system call을 실행할 때 SVC모드에서 실행한다.
    - ABT(Abort Mode): 메모리에서 데이터나 명령어를 읽어오다가 문제가 생기면 변경되는 모드
    - UND (UNDefined Mode): ARM 집합에 없는 명령어를 읽을 경우 변경되는 모드
    - SYS(System Mode): 운영체제등에서 사용자 프로세스가 특권모드를 획득해야할 경우, 소프트웨어에 의해서만 진입할 수 있는 모드
- ARM 명령어
    - 32bit
    - 16bit Thumb 명령어 (ARM state와 Thumb state를 BX 명령을 통해 바꿀 수 있음)
- Java 지원
    - 16 bit Jazelle 명령어 기반
    - 하드웨어 기반 JVM 처리 최적화
- ARM 명령어 set 특징

    - Conditional execution (명령어들이 특정 조건이 만족했을 때에만 실행)
    - LD/ST, Branch 가 간접 주소(indirect address)를 사용
    - 11개의 명령어 타입
- Exception vector(우선순위 순서)
    1. Reset (SVC)
    2. Data Abort (ABT)
    3. FIQ (FIQ)
    4. IRQ (IRQ)
    5. Prefetch Abort(ABT)
    6. SoftWare Interrupt(SWI) (SVC)
- SP, R13
    - 특수 목적을 가진 general register
    - ARM에서는 push/pop 지원x -> LDM/STM으로 작동 구현
    - 모드마다 SP 존재
- Link Register, R14
    - 함수호출, 예외발생시 return address를 저장
    - 루틴 마치고 PC <-- LR해서 복귀
    - 모드마다 LR 존재
- Program Counter, R15
    - 다음에 실행될 명령어의 주소를 저장
    - 시스템에서 1개
- Program Status Register (PSR)
    - CPSR 1개, SPSR 5개
    - Condition flag: ALU 연산 결과 나타내는 flag
    - Control bits: CPU 상태 제어