3부. ARM 프로세서와 임베디드 하드웨어 설계

07장. ARM 프로세서 코어
08장. ARM 프로세서
09장. SoC 구조
10장. 임베디드 시스템 하드웨어 설계

DIGNSYS
www.dignsys.com

교 육 목 표

- **ARM** 프로세서 코어의 구조를 배운다
- 캐시, **MMU,** 쓰기 버퍼의 구조와 제어 방법을 배운다**.**
- **ARM** 프로세서 기반의 **SoC** 구조를 배운다
- 임베디드 시스템 하드웨어 설계 방법을 배운다**.**

목 차

ARM 프로세서

- **ARM 프로세서(Processor)**
  - 프로세서 코어에 캐시, MMU(Memory Management Unit), Write Buffer, TCM(Tightly Coupled Memory), BIC(Bus Interface Unit)과 같은 주변 회로를 구성한 독립된 형체
- **ARM 프로세서의 종류**
  - ARM920T 프로세서, ARM940T 프로세서, ARM946EJ 프로세서, ARM1020E 프로세서, ARM1136JF-S 프로세서 등
- **프로세서 Family**
  - 동일한 프로세서를 사용한 제품군
  - ARM9TDMI core를 사용한 ARM940T나 ARM920T와 같은 프로세서를 ARM9TDMI 프로세서 family라고 부른다.

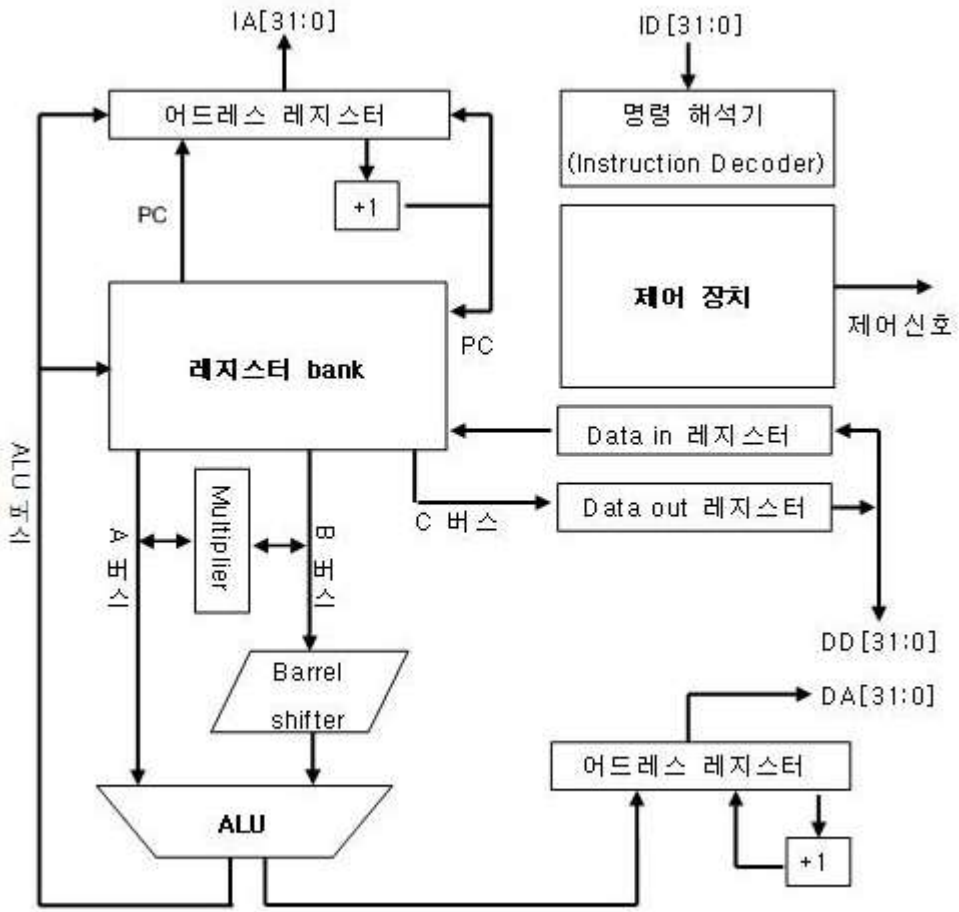
ARM9TDMI 프로세서의 특징

- **ARM Architecture v4T 사용**
- **5 단의 pipeline을 사용**
- **Harvard Architecture 사용**
  - 메모리 인터페이스 분리
    - 명령어를 읽는 메모리 인터페이스
    - 데이터를 읽고 쓰기 위한 메모리 인터페이스
  - 동시에 명령어의 fetch와 데이터의 억세스가 가능
- **Clock speed 향상**
  - 반도체 기술의 발달
  - 늘어난 pipeline stage
- **1.5 CPI(Clock Cycle Per Instruction)**

ARM9TDMI 프로세서

- **ARM9TDMI는 ARM9 core에 다음과 같은 기능이 확장된 프로세서이다**
  - T : Thumb 명령 지원
    - 32 비트 ARM 명령과 16 비트 Thumb 명령 모두 지원
  - D : Debug 지원
    - Core 내부에 디버그를 지원하기 위한 기능과 디버그를 위한 입출력 신호가 추가
    - 디버거를 사용하기 위해서는 반드시 필요한 부분
  - M : Enhanced Multiplier 지원
    - 64 비트 결과를 낼 수 있는 32x8의 Enhanced Multiplier 지원
    - 기본은 2비트, M 확장된 8비트로 계산 시간 단축
  - I : Embedded ICE Logic
    - 디버거를 이용하여 명령어 breakpoint나 데이터 watchpoint를 설정 가능하도록 한다.
    - 디버거를 사용하기 위해서는 Debug 지원(D 확장)과 함께 반드시 필요

ARM9TDM 내부 구조



ARM9TDM core 내부 구조

- 레지스터 **Bank**
  - 37개의 32비트 레지스터
  - 데이터 처리를 위한 저장 공간으로 사용
- 명령어 해석기
  - 입력되는 명령을 해석하는 장치
- 제어 장치
  - 명령에 따라 필요한 제어 동작 수행하고 필요에 따라 제어 신호를 외부로 구동
- **ALU**
  - 32비트 산술 및 논리 연산 수행
- **Multiplier**
  - 32비트의 Booth's 곱셈기가 연결되어 32비트 곱셈 연산 수행

ARM9TDM 코어 내부 구조

- 어드레스 **Incrementer**
  - 순차적인 명령어를 fetch 할 때 사용 (Instruction 용)
  - LDM 이나 STM 같은 block 단위의 데이터 전송명령이 실행될 때 사용 (Data 용)
- 메모리 인터페이스 분리
  - Harvard Architecture
    - 명령어를 읽는 메모리 인터페이스
      - ✓ *어드레스 IA[31:0], 데이터 ID[31:0]*
    - 데이터를 읽고 쓰기 위한 메모리 인터페이스
      - ✓ *어드레스 DA[31:0], 데이터 DD[31:0]*

ARM9TDM 코어의 CPU 내부 버스

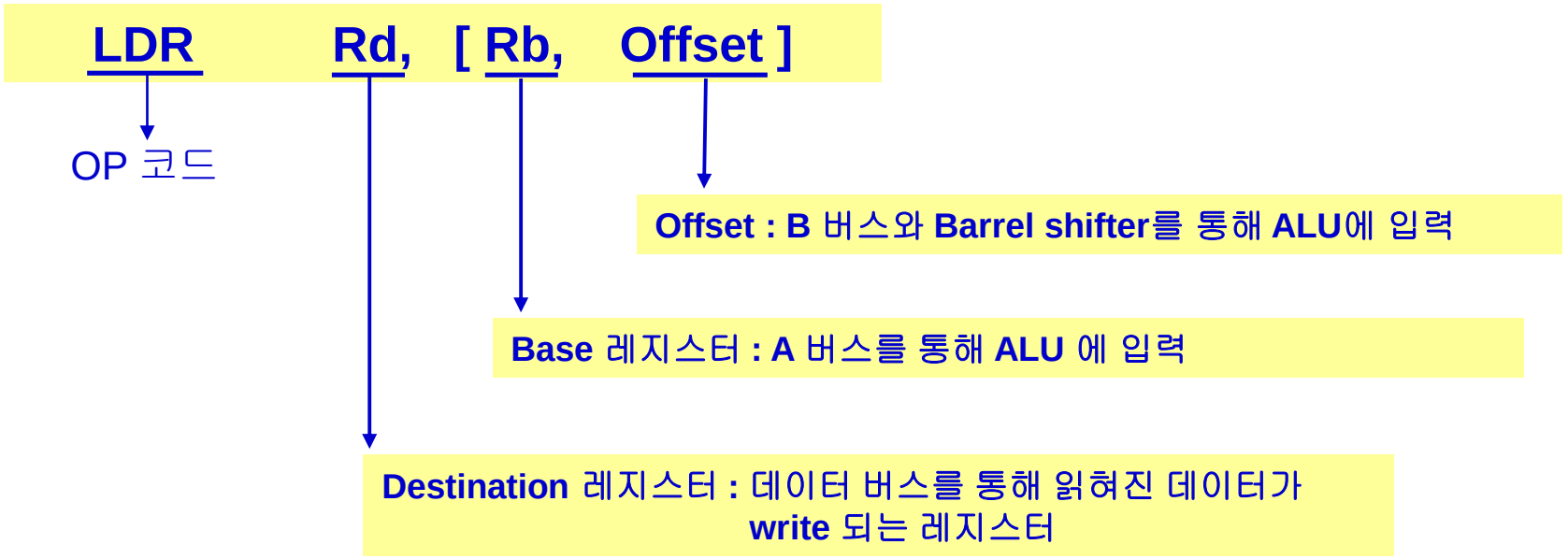
- **A 버스**
  - 레지스터 뱅크에서 바로 ALU에 연결 된다.
  - Data Processing 명령의 경우 Operand 1(Source 레지스터)가 전달되는 경로
  - Data Transfer 명령의 경우 Base 레지스터 값이 전달 되는 경로
- **B 버스**
  - 레지스터 뱅크에서 Barrel Shifter를 통하여 ALU에 연결 된다
  - Data Processing 명령의 경우 Operand 2가 전달되는 경로
  - Data Transfer 명령의 경우 Offset 값이 전달 되는 경로
  - Barrel shifter :
    - Operand 2와 Offset 을 지정할 때 shift operation을 같이 사용할 수 있도록 한다.

ARM9TDM 코어의 CPU 내부 버스

- **C 버스**
  - 데이터를 위한 전용 버스
- **데이터 입력 버스**
  - 외부 데이터 버스에서 데이터를 읽어 내부 레지스터로 전달하는 경로
- **ALU 버스**
  - ALU의 연산 결과가 전달되는 경로
  - Data Processing 명령의 경우 레지스터 뱅크의 Destination 레지스터에 결과 값 저장
  - Data Transfer 명령의 경우 어드레스 레지스터에 저장되어 어드레스를 생성

데이터 처리 명령과 내부 버스

데이터 전송 명령과 내부 버스

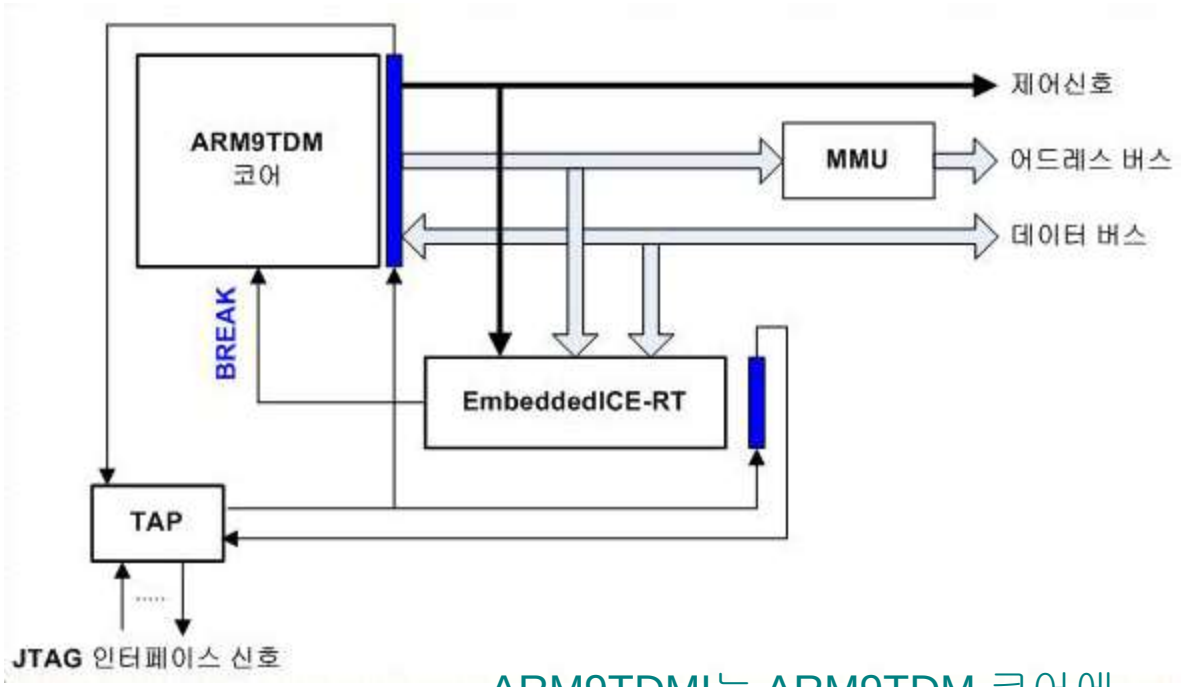


Multiplier 와 Shifter

- **Booth's 곱셈기**
  - 32비트의 두 입력을 받아서 곱하여, 결과가 32비트를 넘더라도 넘는 부분은 버리고 32비트만 남긴다.

- **Barrel Shifter**
  - 한번의 연산으로 데이터 워드 내에 있는 다수의 비트를 shift하거나 rotate 시킬 수 있다.
  - 1 clock 사이클 동안에 최대 32비트 shift 가능

ARM9TDMI 코어와 EmbeddedICE 로직



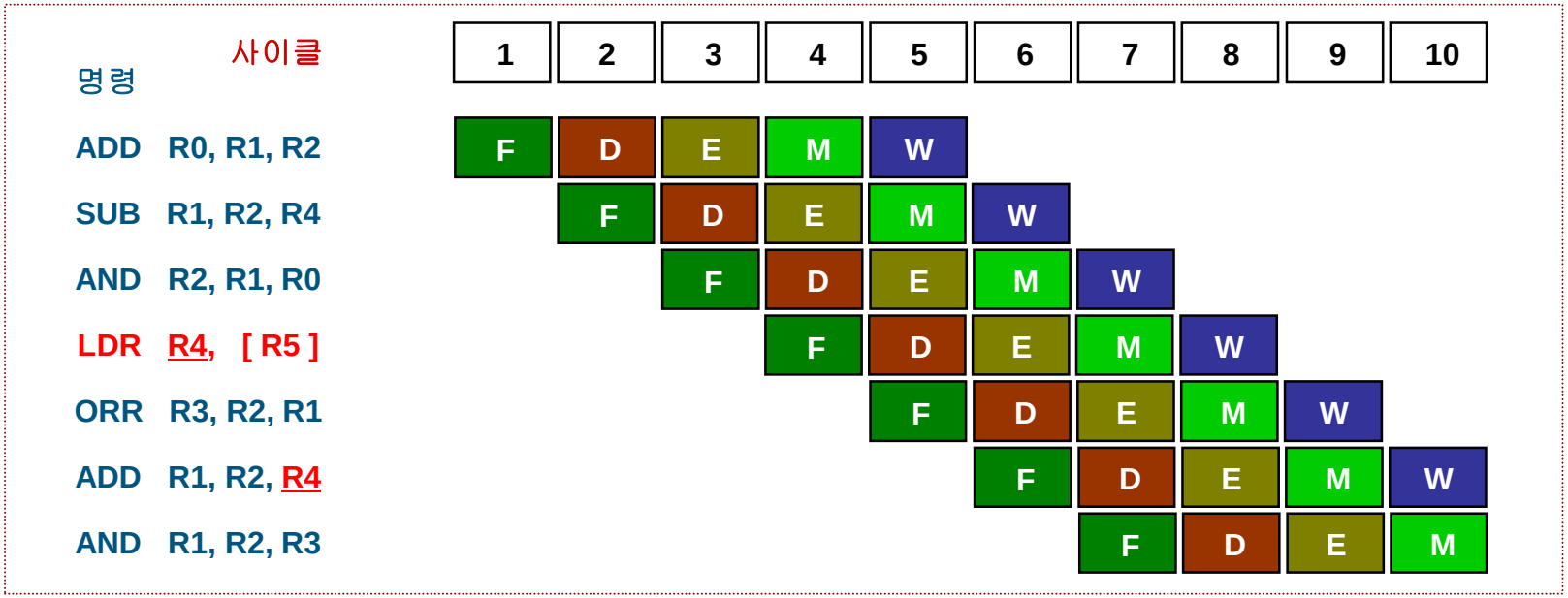
ARM9TDMI는 ARM9TDM 코어에
EmbeddedICE 로직이 추가된 프로세서이다

ARM9TDMI 파이프라인

	FETCH	DECODE	EXECUTE
ARM :	PC	PC-4	PC-8
Thumb :	PC	PC-2	PC-4

- **ARM9**의 경우 **ARM7**과 달리 **32** 비트 **ARM** 명령어와 **16** 비트 **Thumb** 명령어 **decoder**를 별도로 가지고 있다.
- **Memory stage**가 추가되어 **LDR** 명령이 사용되더라도 매 사이클 마다 하나의 명령이 실행된다.

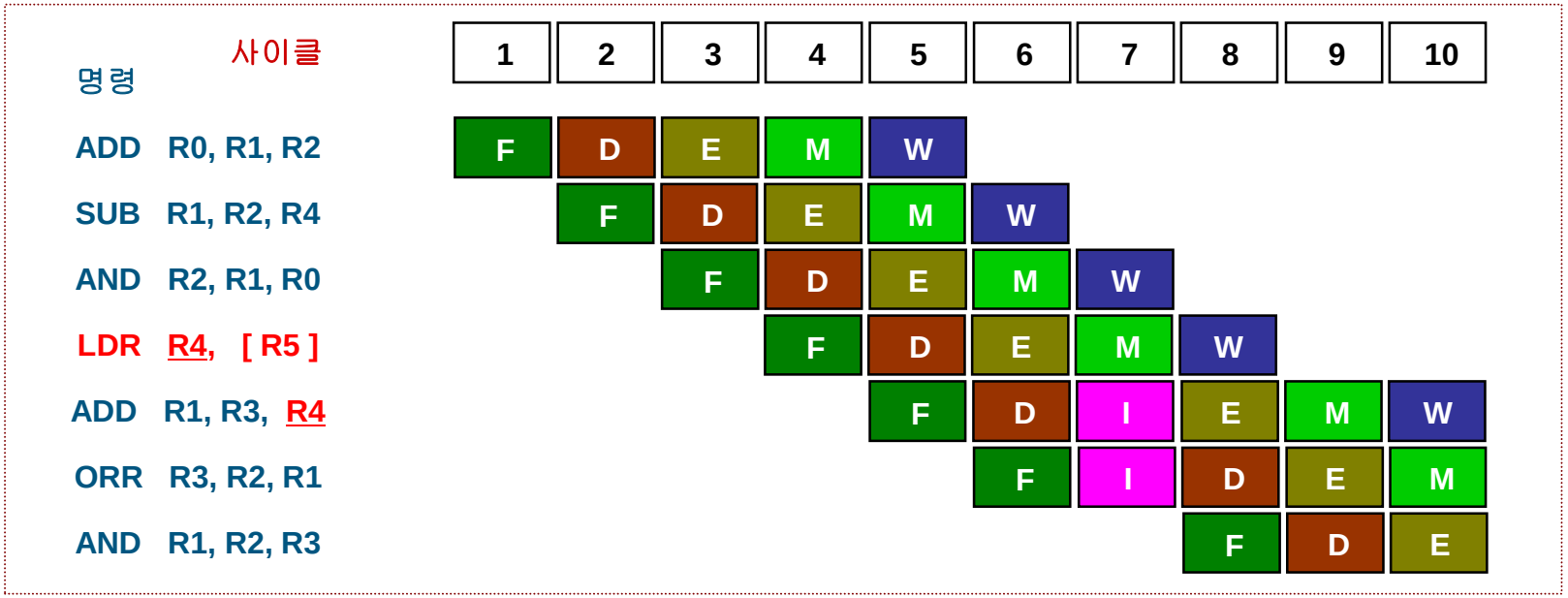
ARM9TDMI의 기본 파이프라인 동작



F : Fetch D : Decode E : Execute M : Memory W : Write I : Interlock

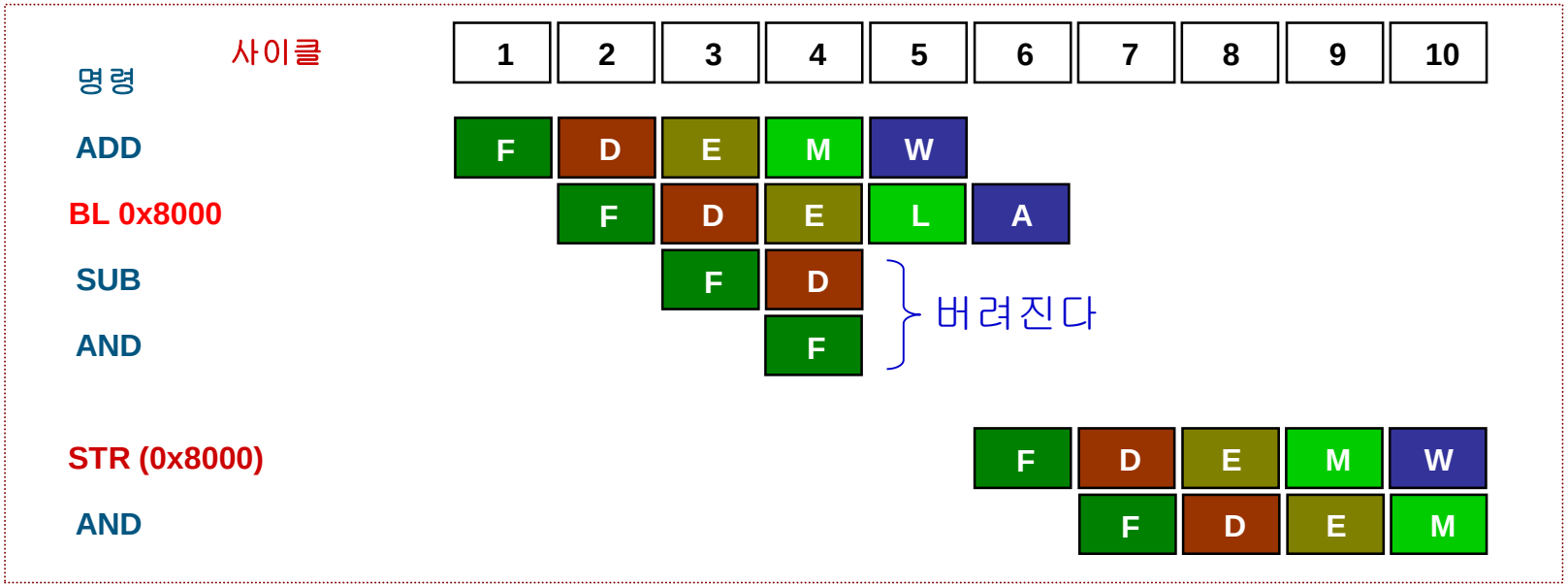
- LDR 명령이 사용되더라도 매 사이클 마다 하나의 명령어 실행
  - CPI = 1
  - 단 LDR 명령의 Destination 레지스터가 다음 명령에서 사용되면 않 된다.

LDR 명령과 인터락 사이클



- LDR 명령과 Interlock 사이클
  - LDR 명령의 Destination 레지스터가 다음 명령에서 사용되는 경우 발생
  - CPI = 1.2

분기 명령과 파이프라인 동작



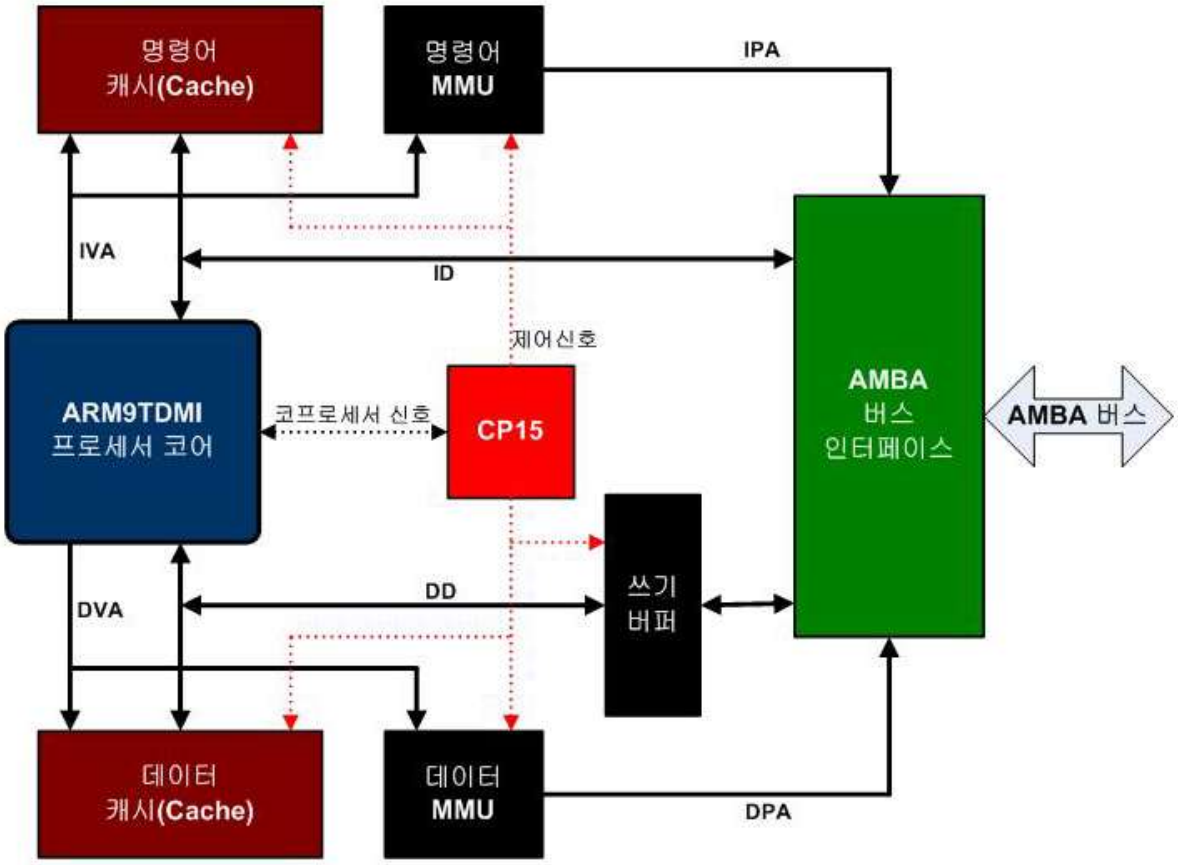
F : Fetch D : Decode E : Execute L : Link Return A : Adjust

- Branch 명령이 사용되면
  - Fetch한 명령을 모두 버리고 지정된 분기하여 새로운 명령을 읽는다
    - Pipeline이 깨진다(break)

ARM9TDMI 프로세서 Family

- **ARM9TDMI 프로세서 Family**
  - ARM9TDMI core를 사용하여 만들어진 프로세서
- **Cached ARM9TDMI Processor**
  - ARM9TDMI core와 Cache를 가지고 있는 프로세서
  - ARM920T
    - 16K Instruction/Data Cache
    - Memory Management Unit (MMU)
    - Write Buffer
  - ARM922T
    - 8K Instruction/Data Cache
    - Memory Management Unit (MMU)
    - Write Buffer
  - ARM940T
    - 4K Instruction/Data Cache
    - Memory Protection Unit (MPU)
    - Write Buffer

ARM9 프로세서의 구조



IVA : 명령어 용 가상 주소 (Instruction Virtual Address)
IPA : 명령어 용 물리 주소 (Instruction Physical Address)
ID : 명령어 용 데이터 버스(Instruction Data bus)

DVA : 데이터 용 가상 주소 (Data Virtual Address)
DPA : 데이터 용 물리 주소 (Data Physical Address)
DD : 데이터 용 데이터 버스(Data Data bus)

ARM9E 프로세서

- **ARM Architecture v5TE 사용**
  - ARM/Thumb interwork 개선 : BLX 명령 추가
  - DSP 명령 확장
  - Saturation 연산 추가
  - Count Leading Zero 명령 추가 : CLZ 명령
  - 32x16, 16x16 곱셈 명령 추가
- **ARM9E 프로세서 Family**
  - ARM946E-S
  - ARM966E-S
  - ARM926EJ-S

ARM9E 프로세서 코어

프 로 세 서	특 징
ARM946E-S	❑ **Instruction/Data Cache** 지원 ❑ **Instruction/Data TCM** 지원 ❑ **Memory Protection Unit(MPU)** 지원
ARM966E-S	❑ **Instruction/Data TCM** 지원 ❑ **Cache**와 **MMU** 나 **MPU** 가 없다.
ARM926EJ-S	❑ **Instruction/Data Cache** 지원 ❑ **Instruction/Data TCM** 지원 ❑ **Memory Management Unit(MMU)** 지원 ❑ **Java** 바이트 코드 지원

ARM9E 파이프라인

	FETCH	DECODE	EXECUTE
ARM :	PC	PC-4	PC-8
Thumb :	PC	PC-2	PC-4

- **Memory stage**가 추가되어 **LDR** 명령이 사용되더라도 매 사이클 마다 하나의 명령이 실행된다.
- **MAC(Multiply and Accumulate)** 와 **SAT(Saturation)**이 **pipeline stage**에 추가 되어있다.

목 차

ARM11 프로세서 (1)

- **ARM Architecture v6** 사용
- **8** 단의 **pipeline** 사용
  - Pipeline이 병렬로 처리 가능
- **Branch prediction** 지원
  - Static 과 Dynamic branch prediction 지원
- **VFP(Vectored Floating Point)** 지원
- **Jazelle**이 기본적으로 지원
- **Non-blocking** 지원
- **HUM(Hit under miss)** 지원
- **64** 비트 메모리 인터페이스 사용

ARM11 프로세서 (2)

- **ARM TrustZone technology**
  - On-chip security 지원
  - ARM1176JZ-S, ARM1176JZF-S core
- **Thumb-2 core technology**
  - 향상된 performance와 code density
  - ARM1156T2-S, ARM1156T2F-S core
- **Intelligent Energy Manager(IEM) technology**
  - Dynamic power management
  - ARM1176JZ-S, ARM1176JZF-S core

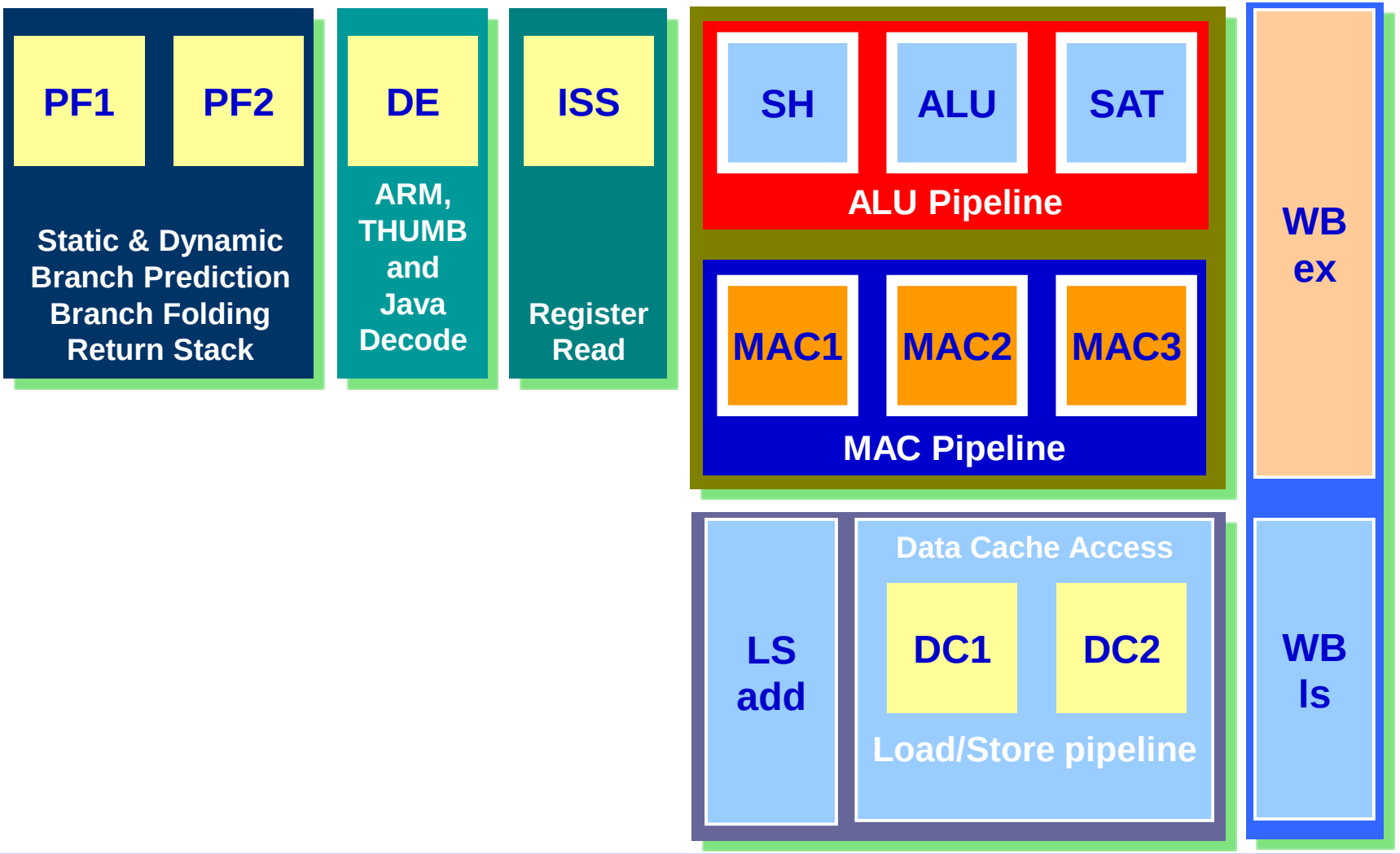
v6 Architecture

- 향상된 명령어 구조
  - Backward compatibility
  - T : Thumb 명령 지원
  - E : DSP 연산 처리 가능
  - J : Java 바이트 코드 처리 가능
- 미디어 처리 기능 확장
  - SIMD(Single Instruction Multiple Data) 명령 추가
  - SAD(Sum of Absolute Difference) 명령 추가
- 예외처리와 과 인터럽트 처리 개선
  - Real-time 처리 성능 증가
  - Vectored interrupt 방식 지원
  - Interrupt Latency 개선
  - 예외처리를 위한 명령어 추가
- 메모리 인터페이스 개선
  - Un-aligned access 지원
  - Mixed-endian 지원

ARM11의 파이프라인

- **ARM11 파이프라인의 구성 : 총 8단**
  - fetch stage 2단
  - Decode stage
  - Issue stage
  - execute stage 4단
- **ARM11 파이프라인의 동작**
  - Fetch 1(Fe1) : 1번째 명령을 fetch, dynamic branch prediction
  - Fetch 2(Fe2) : 2번째 명령을 fetch, static branch prediction
  - Decode(De) : 명령을 decode
  - Issue(Iss) : 레지스터 read 와 명령어 issue
  - Shifter(Sh), MAC1 : shift 동작, 1단계 Multiply-Accumulate
  - ALU, MAC2 : ALU 동작, 2단계 Multiply-Accumulate
  - Saturation(Sat), MAC3 : 연산 결과 saturation, 3단계 Multiply-Accumulate
  - Write Back(Wbex) : 연산이나 Multiply 결과를 레지스터에 write, main execution stage이다.

ARM11의 파이프라인



ARM11 파이프라인의 병렬성

- 명령어 처리 **unit** 분리
  - ALU (Arithmetical Logical Unit)
    - 산술/논리 연산 처리
  - MAC (Multiply-Accumulate)
    - 곱셈 연산 처리
  - LS (Load-Store)
    - Load 명령과 Store 명령 처리 전용
    - Non-blocking 지원 가능

명령어 분기 관리

- 파이프라인의 단계 수의 증가에 따른 시스템 성능 감소
  - 파이프라인의 단계 수가 증가 할 수록 Branch 명령에 따른 성능 감소 발생
- **Branch prediction(분기 예측)**
  - 명령의 흐름을 예측하여 pipeline의 stall 현상을 줄여 효율성을 높인다.
- **Static branch prediction**
  - CP15 레지스터에 의해서 enable/disable 가능
  - 명령어 pipeline의 fetch stage에 이루어 진다.
  - PC-relative한 분기 명령만 예측이 가능하다.
  - Backward branch
    - Taken : 분기를 실행
  - Forward branch
    - Not taken : 분기를 실행하지 않는다.
  - Prefetch buffer에서의 분기 명령 fold
    - 분기예측 결과 분기를 실행하지 않게 되면 branch 명령 삭제

ARM11 프로세서 계열 (1)

- **ARM1136JF-S**
  - Synthesizable core
  - 4개의 Physically-tagged 64k I & D Caches
  - Internal Configurable TCMs
  - 4개의 main memory ports를 가진다.
  - Jazelle core를 가지고 있어 Java 바이트 코드 실행 가능
  - VFP coprocessor를 가지고 있다.
- **ARM1136J-S**
  - ARM1136JF-S와 동일하고 VFP를 지원하지 않는 부분이 다르다.

ARM11 프로세서 계열 (2)

- **ARM1156T2F-S**
  - Instruct/Data Cache 와 TCM
  - Memory Protection Unit (MPU)
  - 4개의 AXI 버스 사용
  - Thumb-2 명령 지원
  - Java 바이트 코드 지원 않 됨
- **ARM1176ZJF-S**
  - Instruct/Data Cache 와 TCM
  - Memory Management Unit (MMU) 와 TrustZone 지원
  - 4개의 AXI 버스 사용
  - Java 바이트 코드 지원
- **ARM1176ZJ-S**
  - ARM1176ZJF-S와 동일하고 VFP를 지원하지 않는 부분이 다르다.

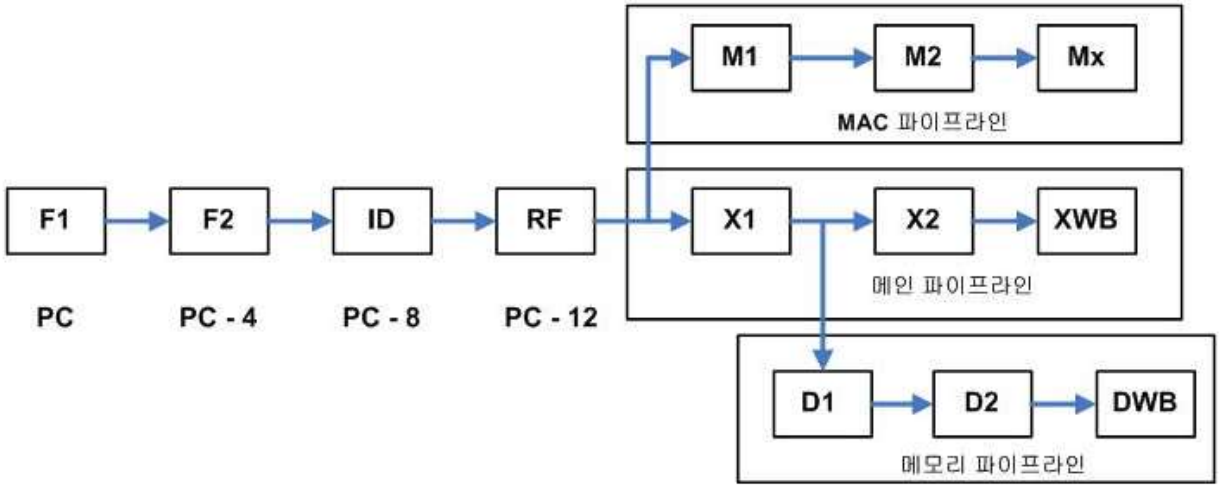
목 차

Intel XScale™ Micro-architecture

- **ARM** 아키텍쳐 **v5TE ISA(Instruction Set Architecture)** 호환
- **32**비트 **ARM** 명령과 **16**비트 **Thumb** 명령 지원
- **ARM**의 **DSP** 확장 기능 명령 지원
- 변형된 하버드 버스 구조 사용
- **7** 또는**8** 단계의 가변적 파이프라인 사용
- 인텔의 **Media Processing Technology** 지원
- 개선된 **16**비트 곱셈기
- **40** 비트 누산기
- **32 KB** 명령 캐시와 데이터 캐시
- **2 KB** 미니 캐시 지원
- **MMU(Memory Management Unit)** 지원
- 분기 예측을 위한 **Branch Target** 버퍼 지원

XScale의 파이프라인

- **3개의 처리 유닛으로 분리**
  - 주 파이프라인
    - ALU 연산 실행하는 파이프라인으로 7 단계로 구성된다.
  - MAC 파이프라인
    - Multiply Accumulate 파이프라인으로, 7 단계로 구성된다.
  - 메모리 파이프라인
    - Load/Store 같은 메모리 참조 파이프라인으로, 8 단계로 구성된다.



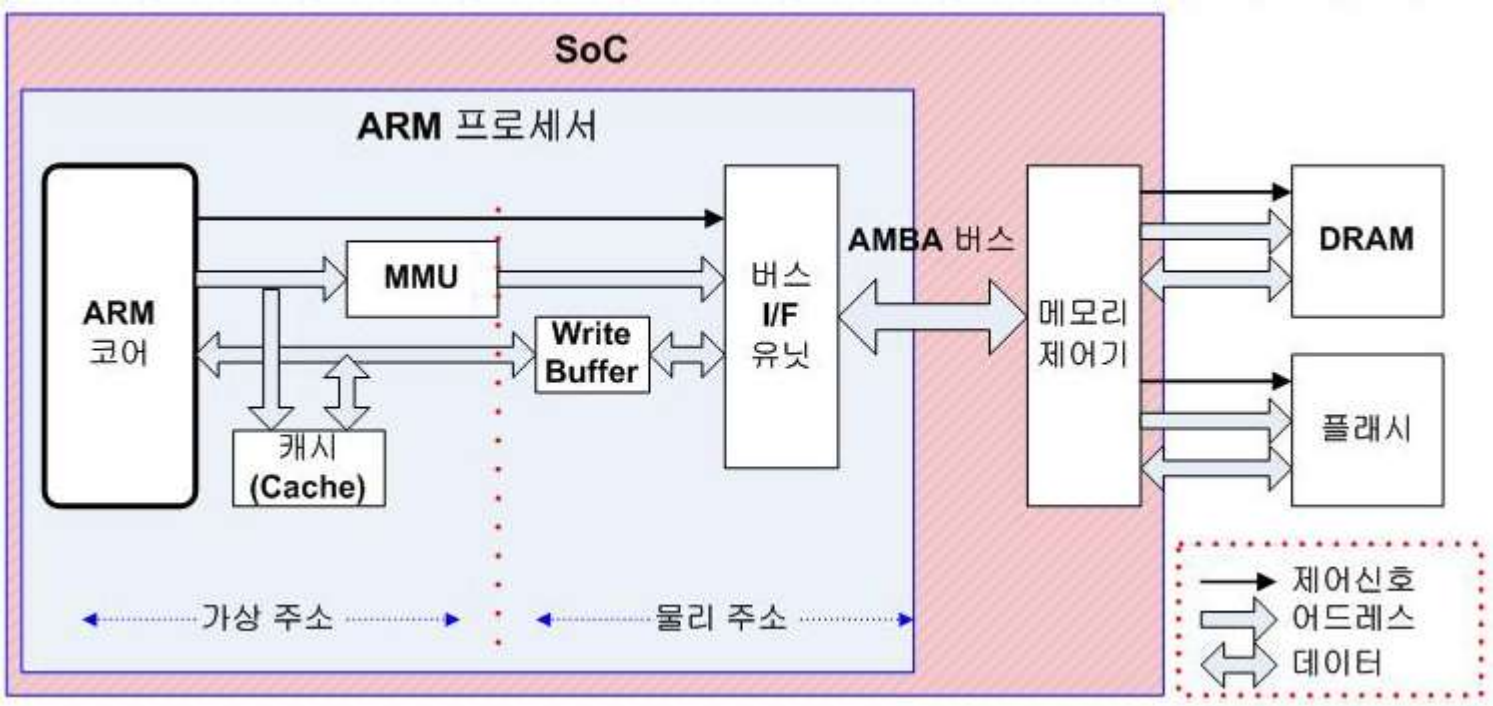
목 차

ARM 프로세서

- **ARM 프로세서(Processor)**
  - 프로세서 코어에 캐시, MMU(Memory Management Unit), Write Buffer, TCM(Tightly Coupled Memory), BIC(Bus Interface Unit)과 같은 주변 회로를 구성한 독립된 형체
- **ARM 프로세서의 종류**
  - ARM920T 프로세서, ARM940T 프로세서, ARM946EJ 프로세서, ARM1020E 프로세서, ARM1136JF-S 프로세서 등

ARM 프로세서와 SoC 그리고 메모리 시스템

- 208페이지, [그림 8-1] 참조



목 차

ARM 프로세서의 제어

- **CP15 인터페이스를 통해서 제어**
  - Cache, MPU or MMU, endian 제어 등
- **Coprocessor Register Transfer 명령**
  - MRC : Move to Register from Coprocessor
    - Coprocessor 레지스터 내용을 ARM 레지스터로 전송
  - MCR : Move to Coprocessor from Register
    - ARM 레지스터의 내용을 Coprocessor 레지스터로 전송

MCR/MRC{cond} p15, opcode_1, Rd, CRn, CRm, <opc_2>

p15 – coprocessor 15를 나타낸다

opcode_1 – always zero

Rd – ARM의 source 또는 destination 레지스터

CRn – primary CP15 register

CRm – additional register name

<opc_2> – additional information 표시

ARM920T 의 CP15 레지스터 들

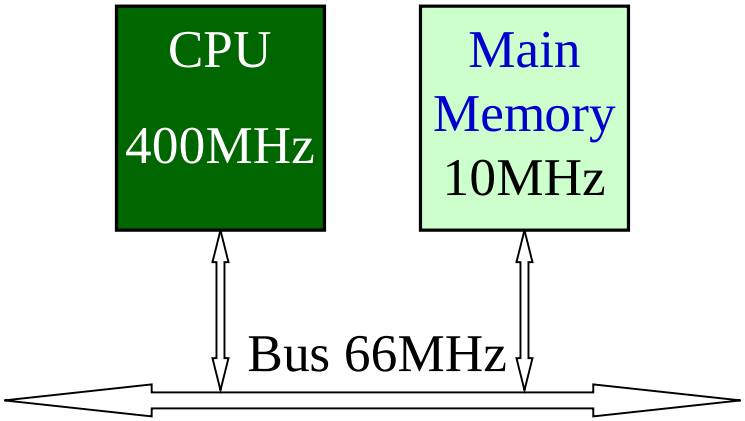
Register	용 도	비 고
0	ID code register	<opc_2>=0
	Cache type register	<opc_2>=1
1	Control Register	Cache, MMU enable, Endian Clock, 제어 등
2	Translation table base register	
3	Domain access control register	
5	Fault status register	
6	Fault address register	
7	Cache operation register	Cache control
8	TLB operation register	
9	Cache lockdown register	
10	TLB lockdown register	
13	FSCE PID register	Fast Context Switching Extension
14	Debug support register	DCC enabled
4, 11, 12	Reserved	

ARM920T 의 CP15 레지스터 억세스

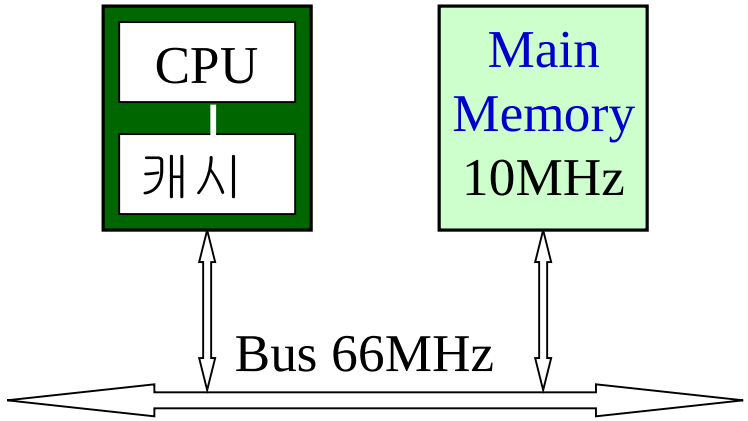
- 각각의 **CP15** 레지스터에 대한 자세한 설명과 억세스 방법은 책 참조
  - 페이지 210에서 페이지 217
  - CP15 레지스터 1 ~ CP15 레지스터 13

목 차

캐시 메모리



고속의 CPU가 버스 및 메모리 속도에 의존적이며 늦다



CPU 주변에 고속의 메모리를 두고 자주 사용되는 명령과 데이터를 저장하여 시스템 성능을 개선

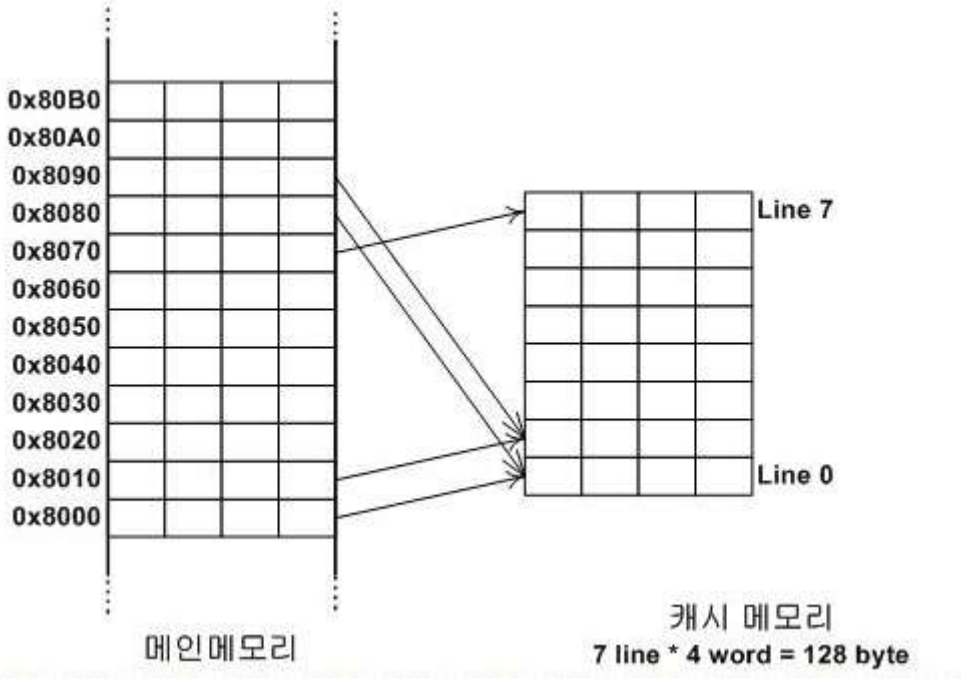
- **캐시 메모리**
  - 프로세서가 최근에 억세스 한 메모리의 내용을 보관하고 있다가 다시 요청하면 메모리 억세스 없이 전달
- **재 사용되는 메모리 공간에 대한 억세스가 없다**
  - 속도가 느린 메모리 시스템의 성능개선
  - 버스의 사용량을 줄여 시스템 성능 향상
  - 전력 소모를 줄일 수 있다

캐시와 성능

- 프로세서가 읽고자 하는 명령이나 데이터가 캐시 내에 존재**(Cache Hit)** 하는 회수가 많아야 캐시의 성능이 우수하다**.**
- **CPU**가 데이터나 명령을 읽고자 하는데 캐시 내에 원하는 명령이나 데이터가 없으면**(Cache Miss),** 캐시 제어기는 시스템 메모리 장치에서 **line** 크기 만큼 명령이나 데이터를 읽어 캐시 메모리에 저장**(Line Fill)** 한다**.**
  - Cache Line : 캐시가 관리하는 최소한의 데이터 단위
    - 4 word, 8 word 정도의 크기를 가진다.
  - Cache miss 가 많으면 시스템 성능이 떨어 진다.

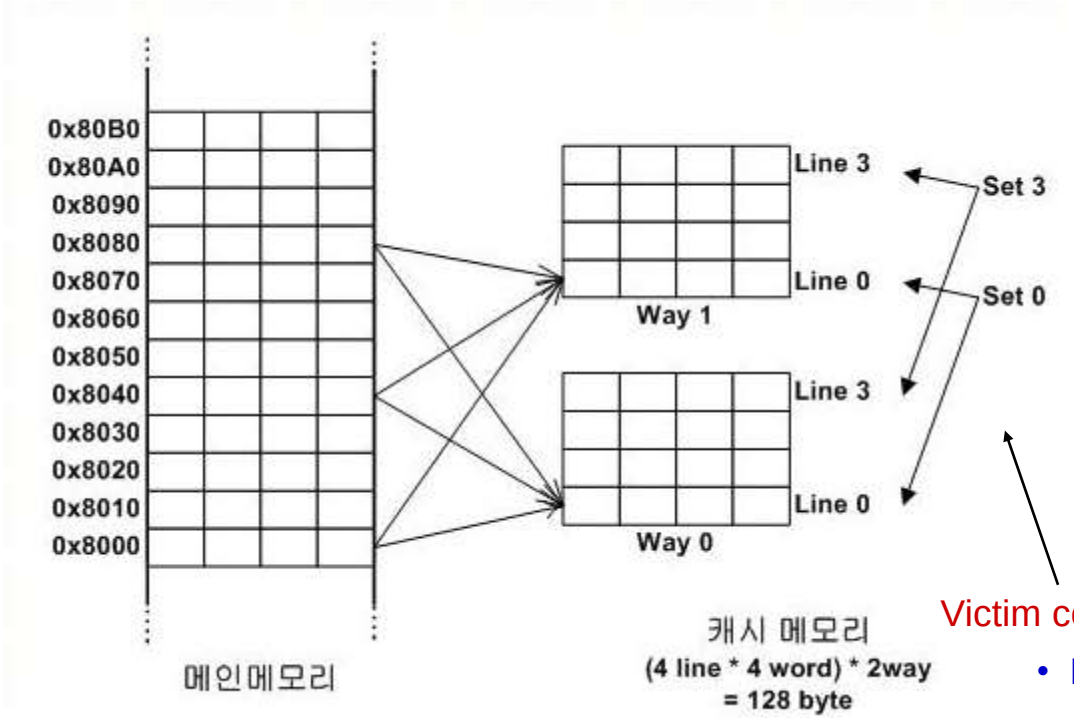
직접 맵핑 된 캐시

- 메모리 성능은 개선 되지만 미스(**miss**)가 발생할 확률이 높다



Set Associative 캐시

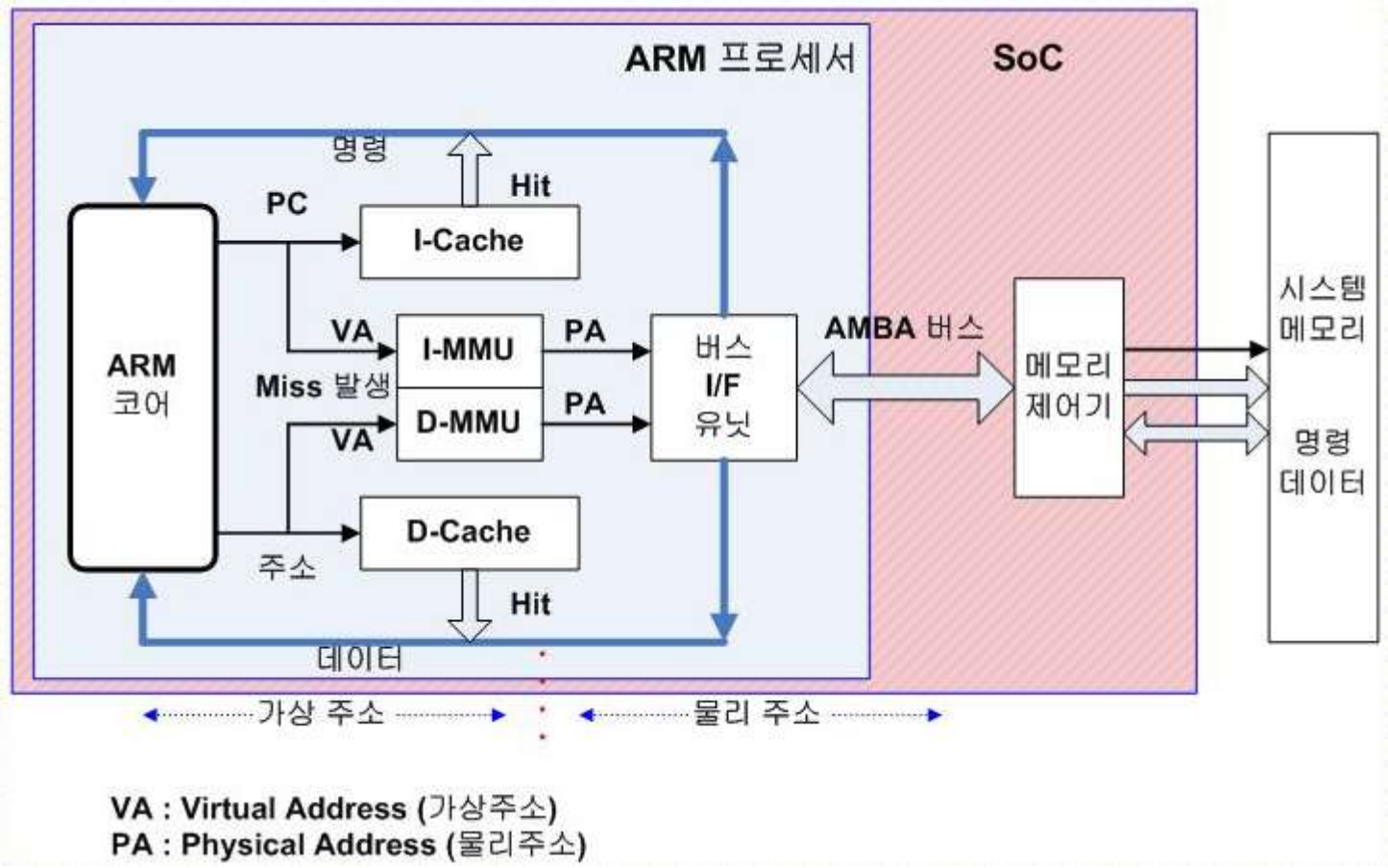
- 직접 맵핑 된 캐시보다 히트(hit)될 확률이 높다



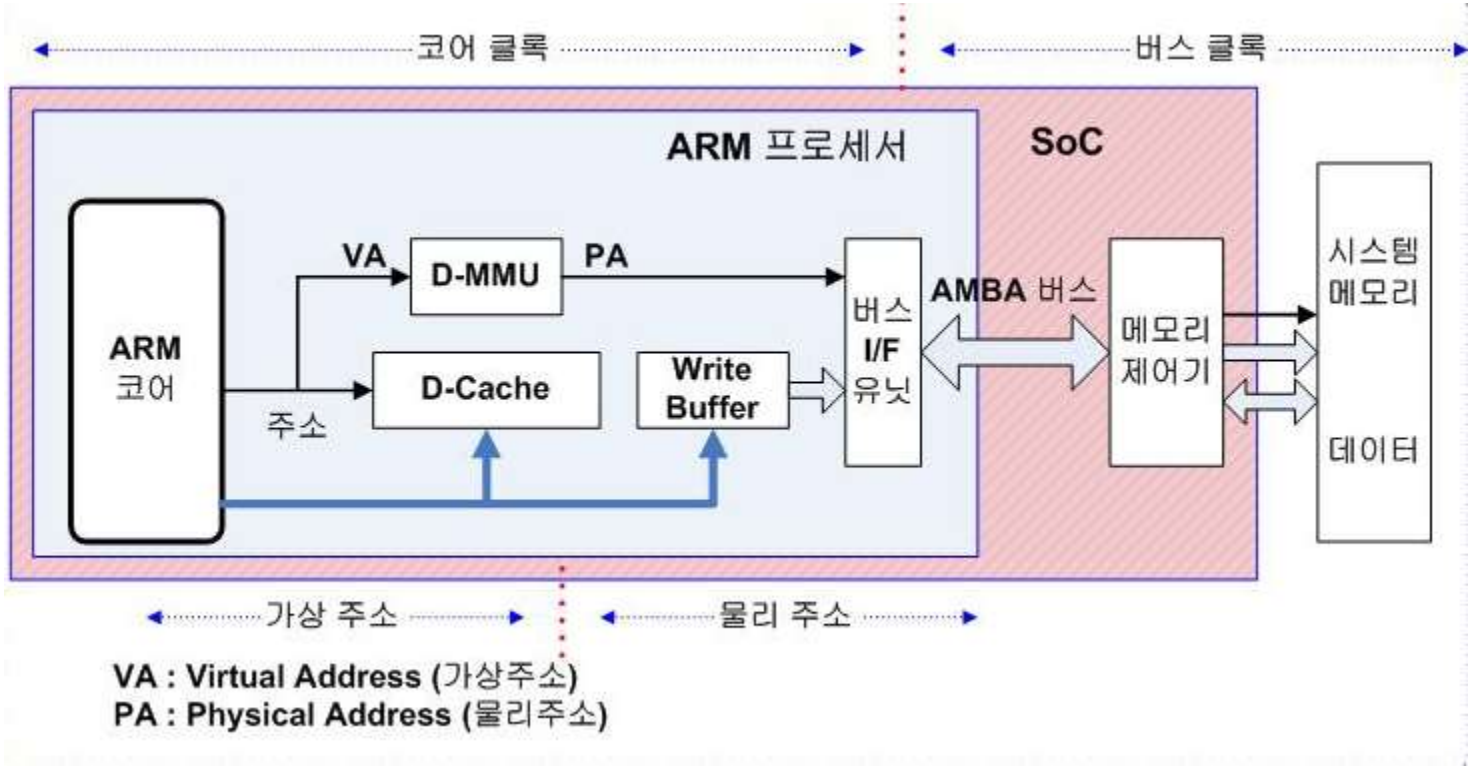
Victim counter : 어느 way에 replace 할지 결정

- Random
- cyclic*(round robbin)
- Least Recently Used(LRU) : 사용 않 됨

캐시와 메모리 일기 동작

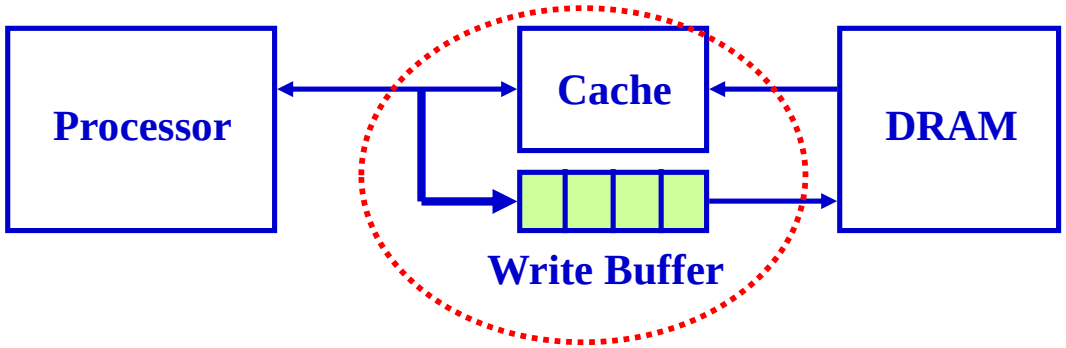

ARM 프로세서
SoC
명령
Hit
PC
I-Cache
ARM
코어
VA
I-MMU
PA
Miss 발생
D-MMU
PA
VA
버스
I/F
유닛
AMBA 버스
메모리
제어기
시스템
메모리
명령
데이터
D-Cache
주소
Hit
데이터
가상 주소
물리 주소
VA : Virtual Address (가상주소)
PA : Physical Address (물리주소)

캐시와 메모리 쓰기 동작


코어 클록
버스 클록
ARM 프로세서
SoC
ARM
코어
VA
D-MMU
PA
주소
D-Cache
Write
Buffer
버스
I/F
유닛
AMBA 버스
메모리
제어기
시스템
메모리
데이터
가상 주소
물리 주소
VA : Virtual Address (가상주소)
PA : Physical Address (물리주소)

쓰기 버퍼(Write Buffer)

- **Core와 버스의 서로 다른 속도차 극복**
  - Core speed로 write buffer에 데이터를 write
    - Write 후 CPU는 다른 작업 처리 가능
  - Bus speed로 write buffer의 내용이 메모리로 write
  - Write buffer는 FIFO 형태의 구조를 가진다



쓰기 버퍼와 캐시

- **Write Through**
  - CPU가 특정 주소에 명령이나 데이터를 write하는 경우, 해당하는 명령이나 데이터가 캐시 메모리에 있을 때, 캐시 메모리와 외부 메모리에 모두 쓰기 동작을 한다.
  - Write buffer를 거치지 않고 메모리에 저장
- **Write Back**
  - CPU가 특정 주소에 명령이나 데이터를 write하는 경우, 해당하는 명령이나 데이터가 캐시 메모리에 있을 때, 캐시 메모리에만 쓰기 동작을 하고, 외부의 메모리에는 나중에 기록 된다.
  - 쓰기 버퍼를 사용한다.

캐시의 플러시(Flush)

- **Cache**의 내용과 메모리의 내용이 다른 경우 **Cache**를 비우고 새로 데이터를 메모리에서 읽어와야 한다.
  - Self modifying code가 사용된 경우
  - MPU나 MMU의 값이 변경된 경우
- **Cache Flush** 및 **Invalidate**는 **CP15** 레지스터에 의해서 한다.

```
MCR     p15, 0, r0, c7, c7, 0        ; ID 캐시 무효화
```

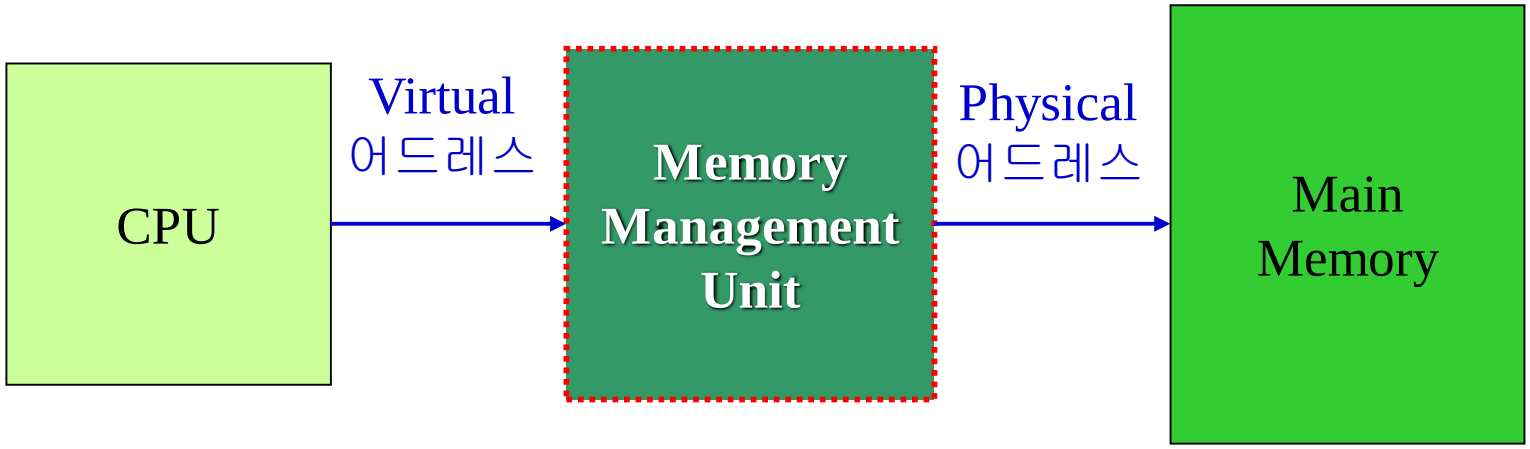
캐시의 락다운(Lockdown)

- **Cache의 일정 부분을 update 되지 않도록 한다.**
  - 중요한 명령이나 데이터를 항상 Cache에 있도록 하여 성능 증가 및 보장
- **Victim counter가 지정한 위치에 가지 않도록 하여 가능**
  - ARM core의 경우 CP15의 register 9번으로 제어
- **Cache flush 전에는 반듯이 Lockdown을 해제 하여야 한다.**

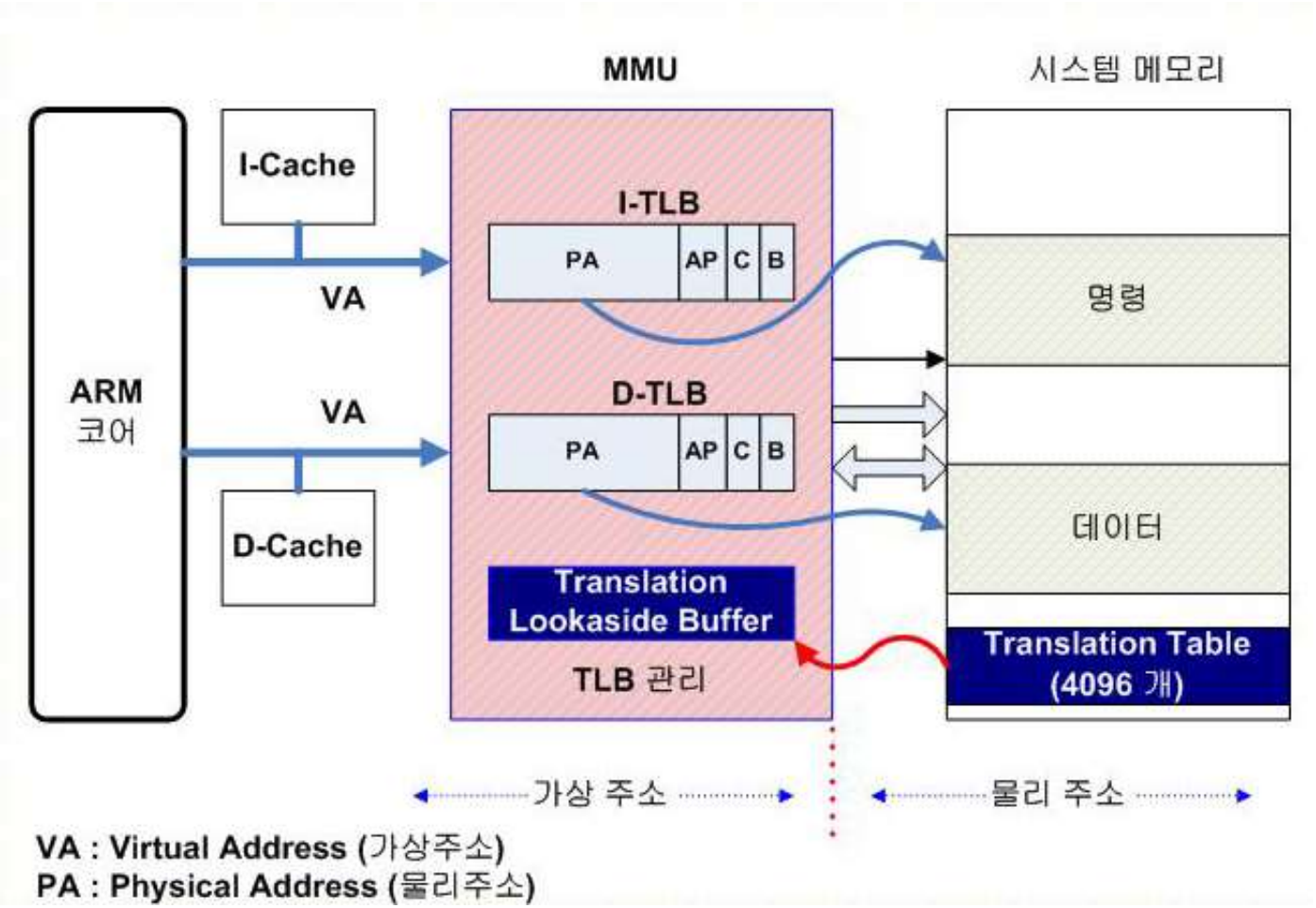
목 차

MMU (Memory Management Units)

- 메모리 보호(protection) 기능
- 어드레스 변환(translation) 기능
  - CPU에서 사용되는 logical 한 Virtual 어드레스를 physical 어드레스로 변환



MMU의 주소 변환


MMU
시스템 메모리
I-Cache
I-TLB
PA
AP
C
B
VA
명령
ARM
코어
VA
D-TLB
PA
AP
C
B
데이터
D-Cache
Translation
Lookaside Buffer
TLB 관리
Translation Table
(4096 개)
가상 주소
물리 주소
VA : Virtual Address (가상주소)
PA : Physical Address (물리주소)

MMU의 구성

- **Translation Lookaside Buffer (TLB)**
  - 최근에 사용된 Virtual address를 physical address로 변화하는 정보와 access permission에 대한 정보를 저장하고 있는 일종의 Cache
- **Translation Table Walking Logic**
  - TLB를 update 하고 관리하는 기능을 가진 logic
- **Access Control Logic**

변환 테이블(Translation Table)

- **Physical 메모리에 있는 translation 정보를 가지고 있는 Table**
- **Level 1 Translation Table**
  - 4096개의 32비트 translation table entry
    - 4GB 메모리를 virtual address 1MB 단위로 나누어 관리
    - Virtual address 비트 [31:20]로 정렬
  - Physical memory에 대한 1MB section 단위의 address translation 정보와 access control 정보를 가지거나, 레벨 2 table에 대한 주소 정보를 가진다.
- **Level 2 Translation Table**
  - 64KB(large page), 4KB(small page), 1KB(tiny page) 단위의 translation table을 정보를 가지고 있다.
  - 각 translation table에는 address translation 정보와 access control 정보를 가진다.

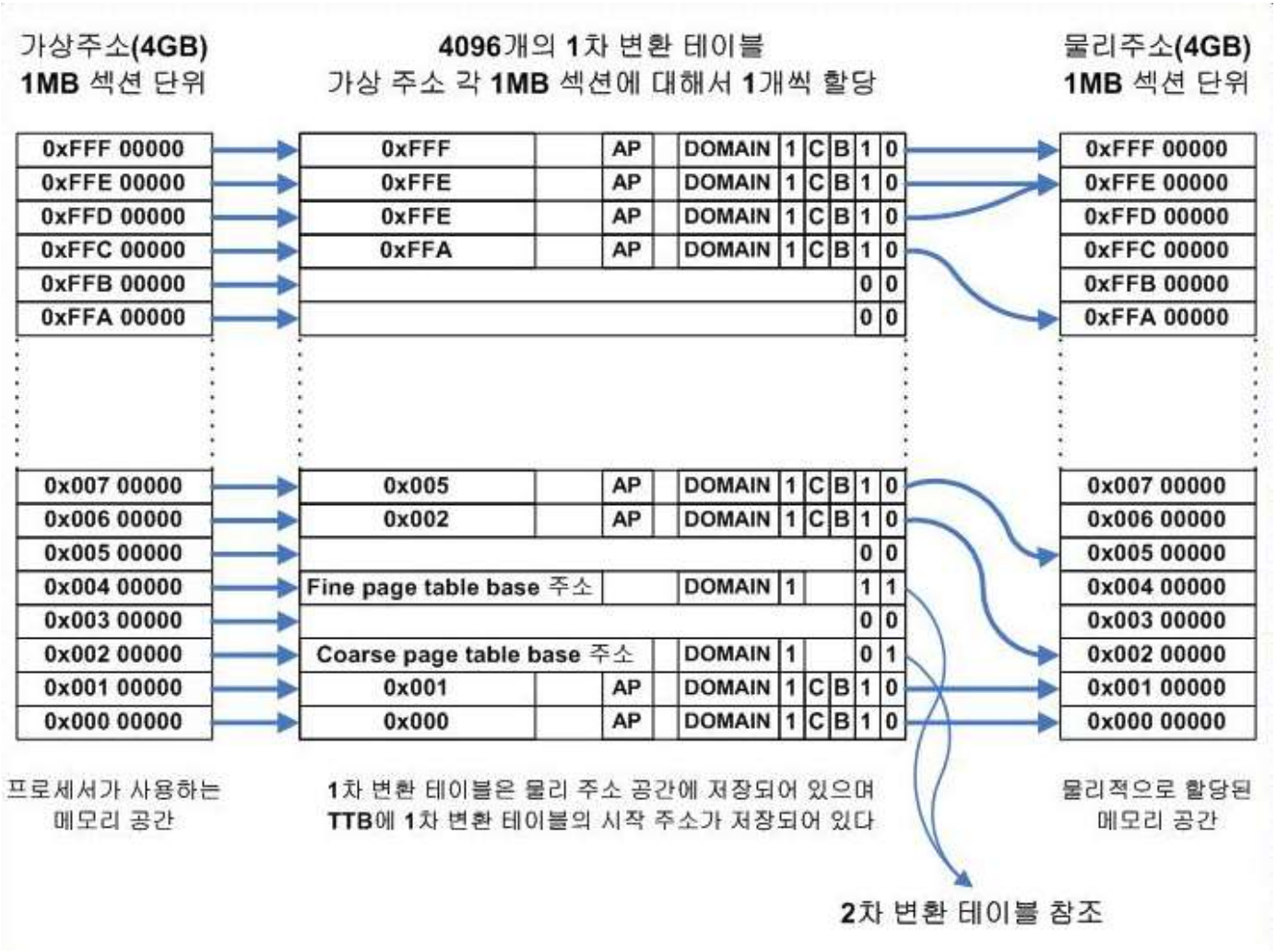
Translation Lookaside Buffer (TLB)

- 최근에 사용된 **Virtual address**를 **physical address**로 변화하는 정보와 **access permission**에 대한 정보를 저장하고 있는 일종의 **Cache**
- **TLB**가 **Virtual** 어드레스에 대한 **translation table entry**를 가지고 있으면 **access control logic**이 **access** 가능을 판단
  - 접근이 허용되면 virtual address를 physical address로 변환 후 access
  - 접근의 허용이 않 되면 CPU에 Abort 구동
- **TLB**에 **virtual** 어드레스에 대한 정보가 없으면 **translation table walking logic**에서 **table** 정보를 **physical** 메모리에서 읽어 **TLB update**

메모리 접근 관리

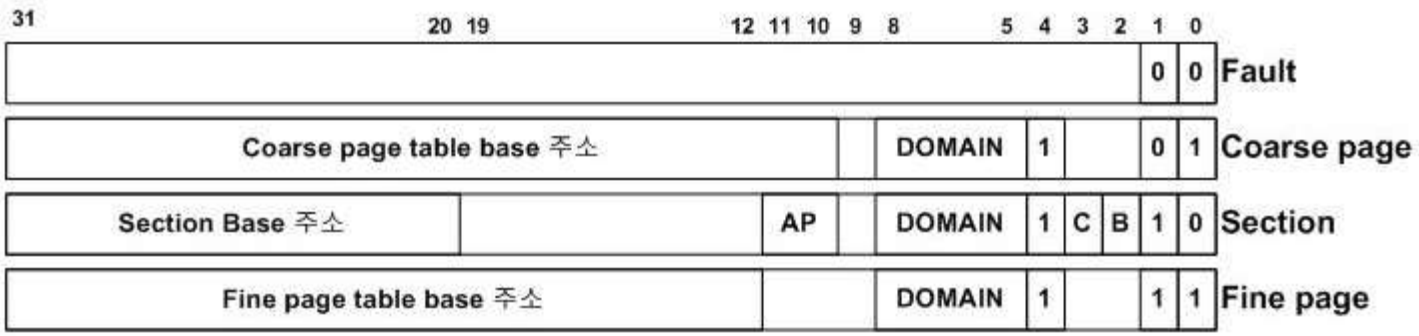
- 섹션(section)과 페이지(page)의 두 가지 메모리 접근 방식
- 섹션(Section)
  - 1MB 단위로 메모리 제어
- 페이지(Page)
  - Tiny page : 1KB 단위로 메모리 관리
  - Small page : 4KB 단위로 메모리 관리
  - Large page : 64KB 단위로 메모리 관리

어드레스 변환


가상주소(4GB)
1MB 섹션 단위
4096개의 1차 변환 테이블
가상 주소 각 1MB 섹션에 대해서 1개씩 할당
물리주소(4GB)
1MB 섹션 단위
0xFFF 00000
0xFFE 00000
0xFFD 00000
0xFFC 00000
0xFFB 00000
0xFFA 00000
0x007 00000
0x006 00000
0x005 00000
0x004 00000
0x003 00000
0x002 00000
0x001 00000
0x000 00000
0xFFF
0xFFE
0xFFE
0xFFA
0x005
0x002
Fine page table base 주소
Coarse page table base 주소
0x001
0x000
AP
DOMAIN
1 C B 1 0
0 0
1 1
0 1
프로세서가 사용하는
메모리 공간
1차 변환 테이블은 물리 주소 공간에 저장되어 있으며
TTB에 1차 변환 테이블의 시작 주소가 저장되어 있다
물리적으로 할당된
메모리 공간
2차 변환 테이블 참조

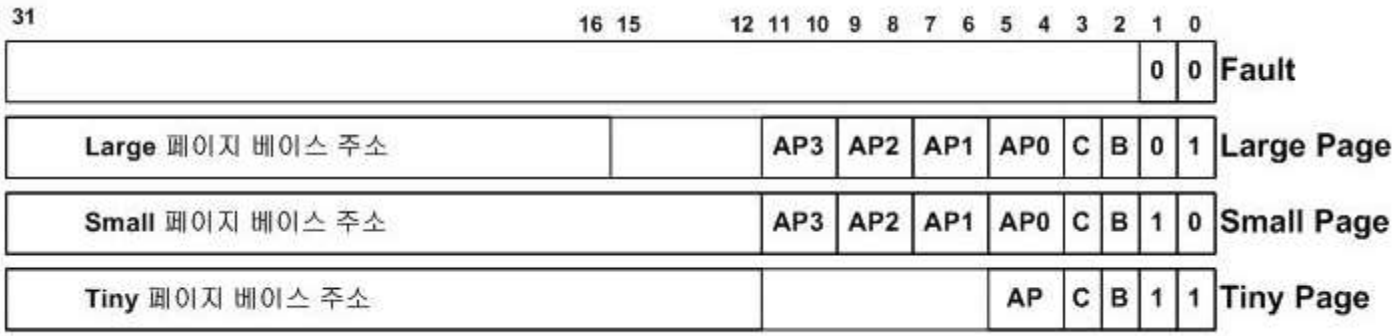
레벨 1 Descriptor

- **Level 1 Descriptor의 종류**
  - Fault
  - Section descriptor
  - Coarse or fine page descriptor
    - Coarse page 는 256 entry의 Level 2 table을 가진다
    - Fine page는 1024 entry의 Level 2 table을 가진다



레벨 2 Descriptor

- **Level 2 Descriptor의 종류**
  - Fault
  - Large page : 64KB 단위 관리
  - Small page : 4KB 단위 관리
  - Tiny page : 1KB 단위 관리



캐시와 쓰기 버퍼 제어

- **Section 또는 page 별로 캐시와 쓰기 버퍼의 사용여부 결정**
  - Cacheable
    - Page 내의 데이터가 Cache될 수 있음을 나타낸다
  - Bufferable
    - Page 내의 데이터가 write buffer에 write될 수 있음을 나타낸다.
    - Memory mapped I/O 장치의 경우에는 반드시 disable 되어 있어야 한다.
- **Cacheable 과 Bufferable에 의한 메모리 시스템 특징**

C	B	의 미	Cache의 Write 동작
0	0	Cache 불가, 쓰기 버퍼 불가	
0	1	Cache 불가, 쓰기 버퍼 동작	
1	0	Cache 동작, 쓰기 버퍼 불가	Write-through Cache
1	1	Cache 동작, 쓰기 버퍼 동작	Write-back Cache

점근 권한(Access Permission)

- **Section 또는 page 별로 메모리의 access permission 제한**
  - Access 권한은 각 descriptor의 AP 정보와 S(System) 비트와 R(Rom) 비트에 의해서 제어
    - S, R비트는 control 레지스터의 비트 8과 9이다
  - Access가 불가하면 permission fault가 발생

AP	S	R	Access Permission	
			Supervisor	User
00	0	0	No access	No access
00	1	0	Read only	No access
00	0	1	Read only	Read only
00	1	1	Reserved	
01	X	X	Read / Write	No Access
10	X	X	Read / Write	Read only
11	X	X	Read / Write	Read / Write
XX	1	1	Reserved	

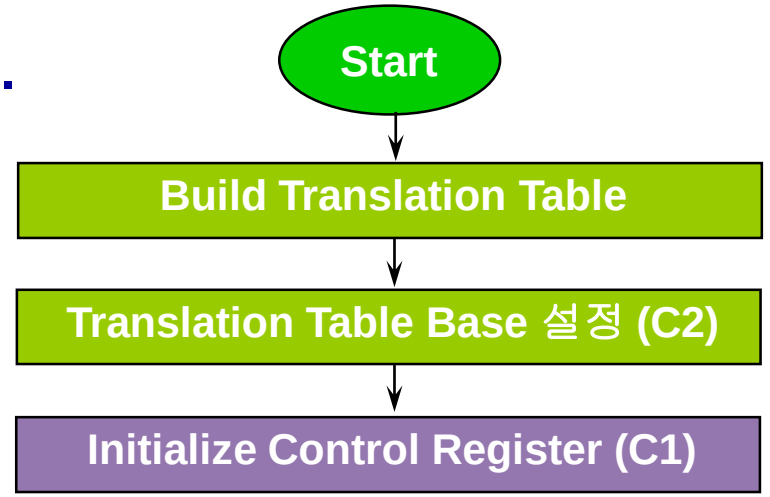
도메인 관리

- **MMU의 Access는 기본적으로 "DOMAIN"의 의해서 제어 된다.**
  - 개별적인 Access permission을 갖도록 제어 하는데 사용
  - 16개까지의 domain 지정 가능
    - DACR(Domain Access Control Register)는 각 domain 별로 2비트씩 할당
- **Domain 과 Access Permission**

Domain	의 미	설 명
00	No Access	모든 access에 대하여 domain fault 발생
01	Client	Page Table의 Section descriptor 나 page descriptor의 AP(Access Permission) 비트 정보를 따른다.
10	Reserved	Reserved (No Access)
11	Manager	Page Table의 Section descriptor 나 page descriptor의 AP(Access Permission) 비트 정보를 무시하고 무조건 access를 허용한다.

MMU 설정

- **MMU를 설정하는 과정은 다음과 같다.**
  1. 변환 테이블 생성
     - Physical 메모리에 설정
     - Virtual to physical translation 정보
     - Cacheable / Bufferable 정보
     - Access Permission
  2. 변환 테이블 주소 지정
     - CP15의 c2 사용
  3. 제어용 레지스터 초기화
     - Enable Cache, MMU
     - Clocking 모드 설정
     - Endian 모드 설정
     - Vector location 설정




```
MOV    r0,  =TTBase          ; Translation table base
MCR    p15, 0, r0, c2, c0, 0 ; set TT base
```

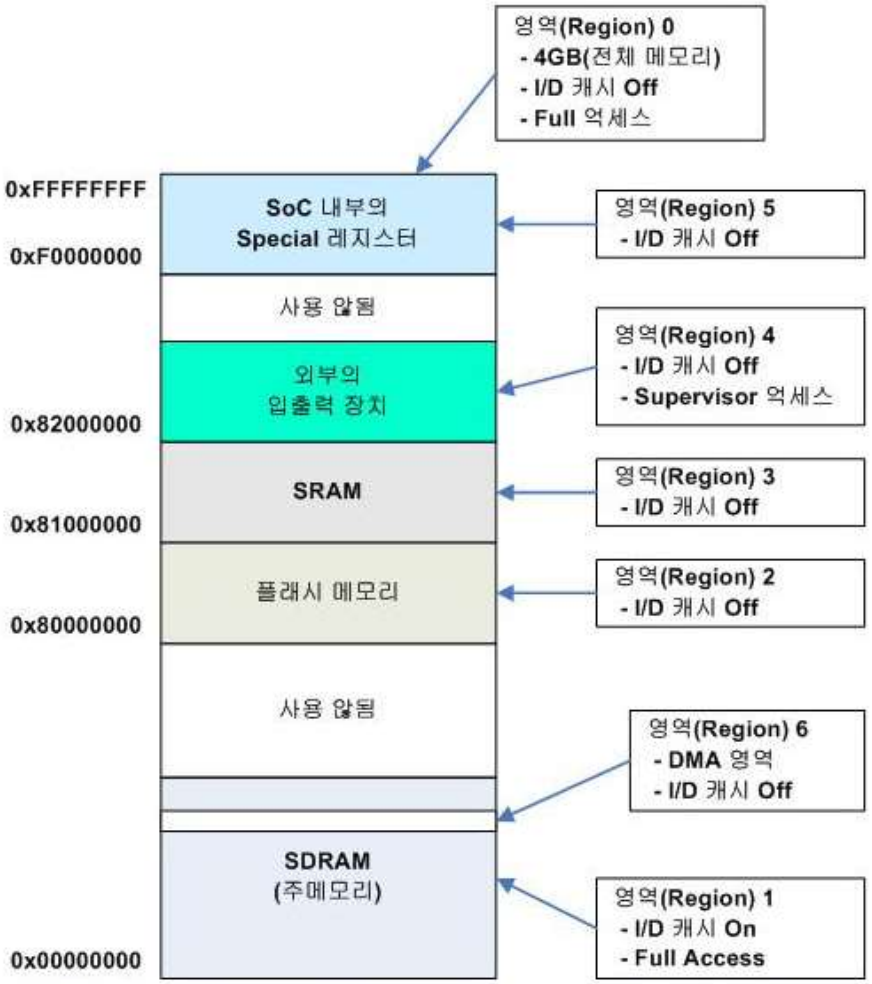
```
MRC    p15, 0, r0, c1, c0, 0 ; read control register
ORR    r0, r0, #0x1
MCR    p15, 0, r0, c1, c0, 0 ; enable MMU
```

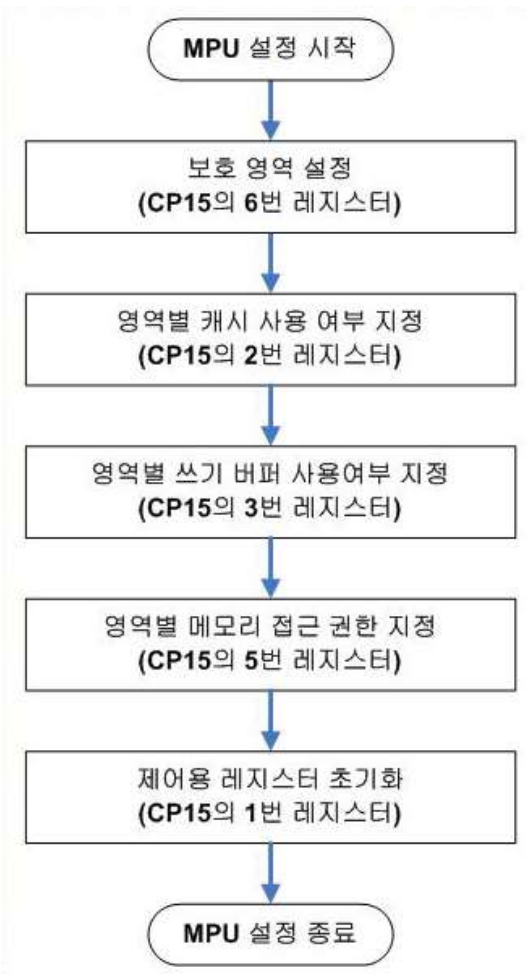
목 차

Memory Protection Unit (MPU)

- **메모리 partition**
  - 메모리를 일정 크기의 region으로 partition
    - 위치와 크기는 프로그램에 의해서 지정
      - ✓ *Region의 크기는 4KB에서 4GB까지*
    - Instruction region과 memory region 설정 필요
  - Region 별로 Cache attribute를 별도로 지정 가능
  - Region 별로 Access permission 별도로 지정 가능
- **Region 설정**
  - MPU가 사용되는 프로세서는 최소 1개 이상의 instruction region과 data region이 설정되어 있고 MPU가 enable 되어 있어야 한다.
  - Protection unit은 Cache를 enable하기 전에 enable 되어 있어야 한다.

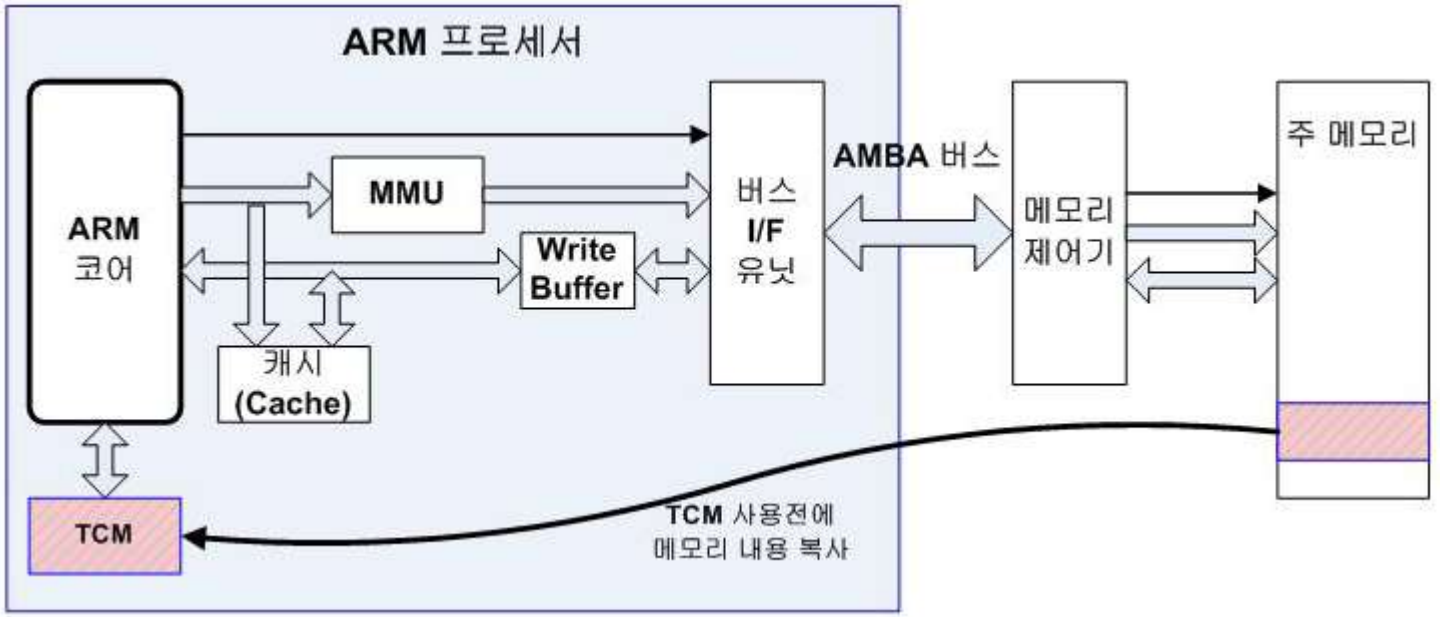
MPU의 보호 영역 지정-페이지242





목 차

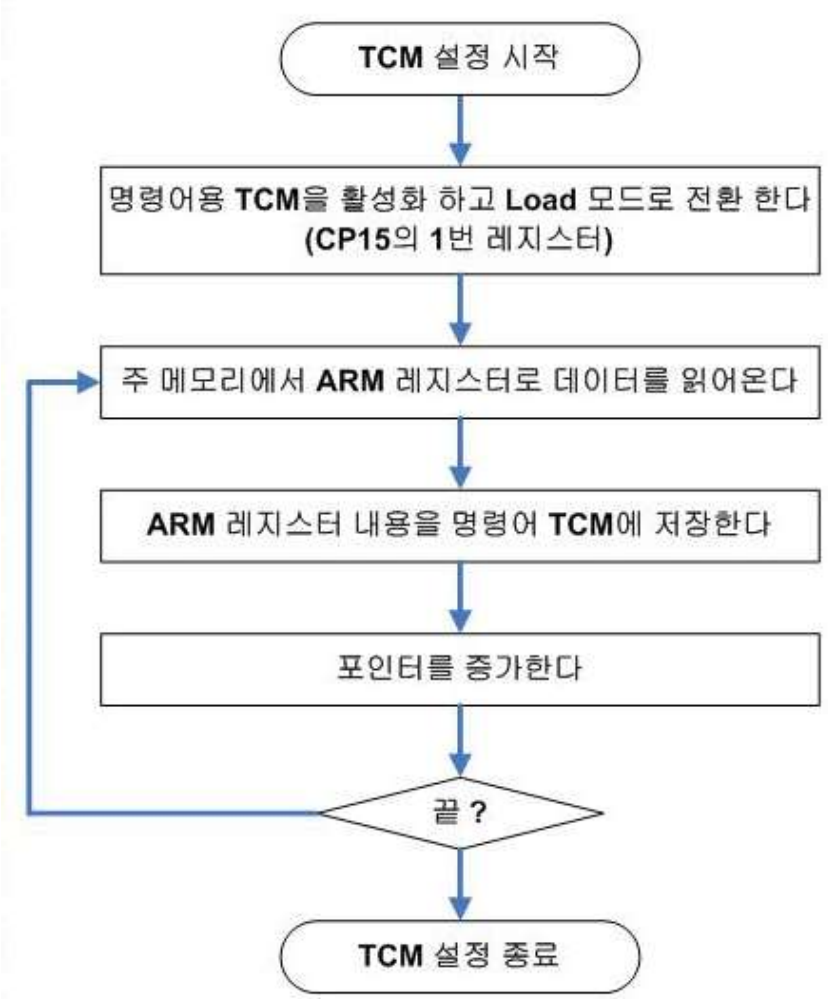
Tightly Coupled Memory (TCM)



메모리의 특정 부분을 **CPU** 바로 옆에 두고
사용하여 최대의 성능을 올린다.

•캐시는 계속 **update** 되므로, 성능을 보장해야 하는데 필요한
명령이나 데이터가 항상 캐시에 있다고 보장 할 수 없다.

TCM 설정


TCM 설정 시작
명령어용 TCM을 활성화 하고 Load 모드로 전환 한다
(CP15의 1번 레지스터)
주 메모리에서 ARM 레지스터로 데이터를 읽어온다
ARM 레지스터 내용을 명령어 TCM에 저장한다
포인터를 증가한다
끝 ?
TCM 설정 종료

목 차

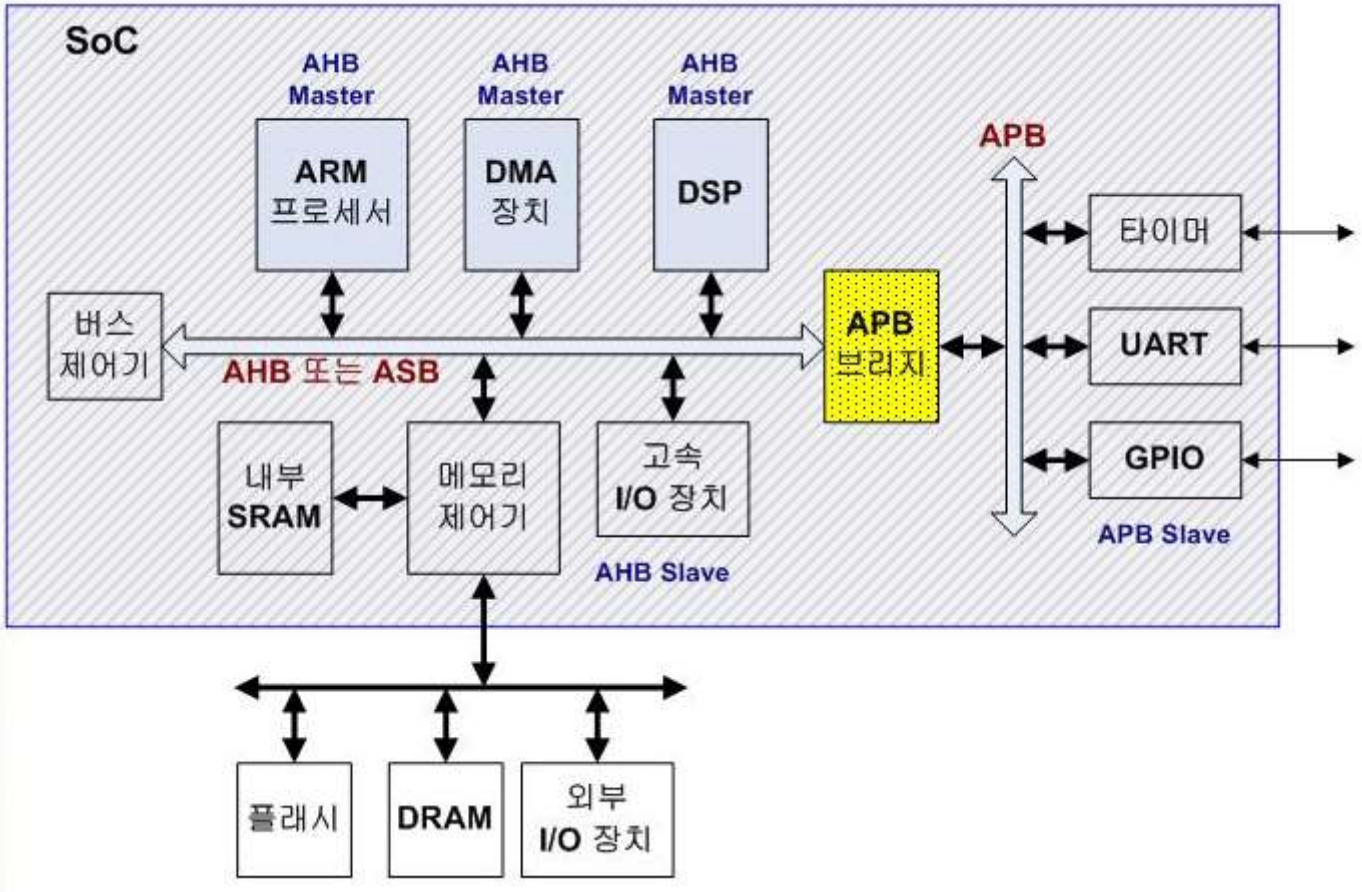
SoC(System On a Chip)

- **SoC(System On a Chip)는 CPU와 함께 각종 입출력 제어장치를 하나의 칩(IC)에 내장한 반도체 부품**
- **버스(Bus)**
  - 고속 SRAM, DMA 제어기, 타이머, UART를 비롯한 여러 입출력 장치와 같은 고속 및 저속으로 동작하는 장치를 서로 효과적으로 연결할 수 있는 신호선의 집합이 필요
- **버스의 신호군**
  - 데이터 버스(data bus)
  - 어드레스 버스(address bus)
  - 제어 버스(control bus)

AMBA 버스

- **AMBA(Advanced Microcontroller Bus Architecture)**
  - ARM에서 개발하여 오픈 한 표준 시스템 버스 설계 규격
  - SoC를 구성하는 기능 블록들 간의 연결 및 관리 방법을 제공
- **표준 AMBA 버스의 장점**
  - SoC 설계자 간의 의사 소통 용이
    - 설계자 간의 idea 공유로 SoC 설계 시간 단축
  - IP(Intellectual Property)의 재사용이 용이
    - 외부의 표준 IP 도입으로 SoC 설계 시간 단축

AMBA 버스의 구성


SoC
AHB Master
AHB Master
AHB Master
ARM 프로세서
DMA 장치
DSP
APB
타이머
버스 제어기
AHB 또는 ASB
APB 브리지
UART
GPIO
내부 SRAM
메모리 제어기
고속 I/O 장치
APB Slave
AHB Slave
플래시
DRAM
외부 I/O 장치

AHB 버스

- **AHB (Advanced High-performance Bus)**
  - 고속(High performance)으로 동작
  - 파이프라인(Pipelined) 동작 지원
  - 여러 개의 버스 마스터 지원
  - 버스트(Burst) 전송 지원
  - 단일 Rising 클록 에지 사용

AHB 버스의 구성

- **AHB 마스터**
  - Read 또는 Write 동작을 요청하는 주체
  - 한번에 하나의 Master 만이 버스를 사용할 수 있다.
- **AHB 슬레이브**
  - 주어진 어드레스 범위 내에서 요청에 따라 Read 또는 Write 동작 수행
  - 데이터 전송 결과를 Master에 보고
    - Success, failure, waiting 상태
- **AHB 중재기**
  - 여러 Master의 요청을 중재하여 한번에 하나의 Master 만이 버스를 사용 하도록 조정한다.
- **AHB 디코더**
  - Master의 요청에 따라 전송하고자 하는 Slave의 어드레스를 Decode
  - AHB 버스에는 하나의 Decoder가 있고, 이 하나의 Decoder에서 모든 Slave를 Decode 한다.

ASB 버스

- **ASB(Advanced System Bus)**
  - 고속 버스
  - ARM의 내부 구조에 기본
  - 파이프라인 동작 지원
  - 여러 개의 버스 마스터 지원
  - 버스트(Burst) 전송 지원
  - Rising/falling 클록 에지 사용

APB 버스

- **APB (Advanced Peripheral Bus)**
  - Low power
  - Latched address and control
  - Simple interface
- **APB 버스의 구성**
  - APB Bridge
    - 어드레스 hold, 어드레스 decode
    - APB Control 신호 생성
  - APB Slave
    - APB의 요청에 의한 응답

AXI 버스

- **AXI (Advanced eXtensible Interface)**
  - AMBA 버스 Spec. 3.0
- **Address,control 와 data phases를 분리**
- **Unaligned data transfer 지원**
- **Burst transfer 지원**
- **Read 와 write data channel 분리**
- **High frequency operation**
- **Low power**

목 차

상용 SoC 제품

- 페이지 **258** 참조

목 차

제품의 요구 사항 분석

- 디버깅 및 시스템 콘솔 용 시리얼 **UART** 사용, 외부와 **D-SUB9** 커넥터로 연결
- 시리얼 통신을 위한 **UART** 확장, **4**핀 헤더 사용
- **USB** 호스트를 지원
- **USB** 슬레이브를 지원
- 현재 시간을 알수 있는 **RTC(Real Time Clock)** 기능 지원
- 시스템의 동작 상태를 나타내기 위한 **LED**를 지원
- 시스템 리셋을 위한 스위치와, 외부 키 입력을 위한 한 개의 푸시 버튼 스위치 지원
- 별도의 하드웨어 확장 보드로**TFT LCD**와 터치스크린 지원
- 별도의 하드웨어 확장 보드로 **10/100Mbps** 이더넷 통신 지원
- 메인보드와 서브보드를 분리하여 설계
- 플래시 및 주메모리는 임베디드 리눅스나 **WinCE**의 동작에 무리가 없도록 충분한 크기를 갖도록 설계

제품 요구에 따른 하드웨어 사양
(페이지 270, [표 10-2] 참조)

항목	설명	공급자 및 부품명
MCU(SoC)	ARM920T, 266MHz @ core 1.8V, I/O 3.3V	삼성, S3C2410
플래시	2MB NOR 플래시,	AMD, AM29LV160
	64MB NAND 플래시,	삼성, K9F1208U0A
주 메모리	64MB SDRAM	삼성, K4S561632E-TC75
시리얼	콘솔 UART	
	RS-232C	SIPEX, SP 3232
	DSUB-9 커넥터	
	4 핀 헤더	2.54mm 간격, Header
USB 호스트	USB 1.1 호환	SoC에 포함
	USB 호스트 커넥터	A 타입
USB 디바이스	USB 1.1 호환	SoC에 포함
	USB 디바이스 커넥터	B 타입

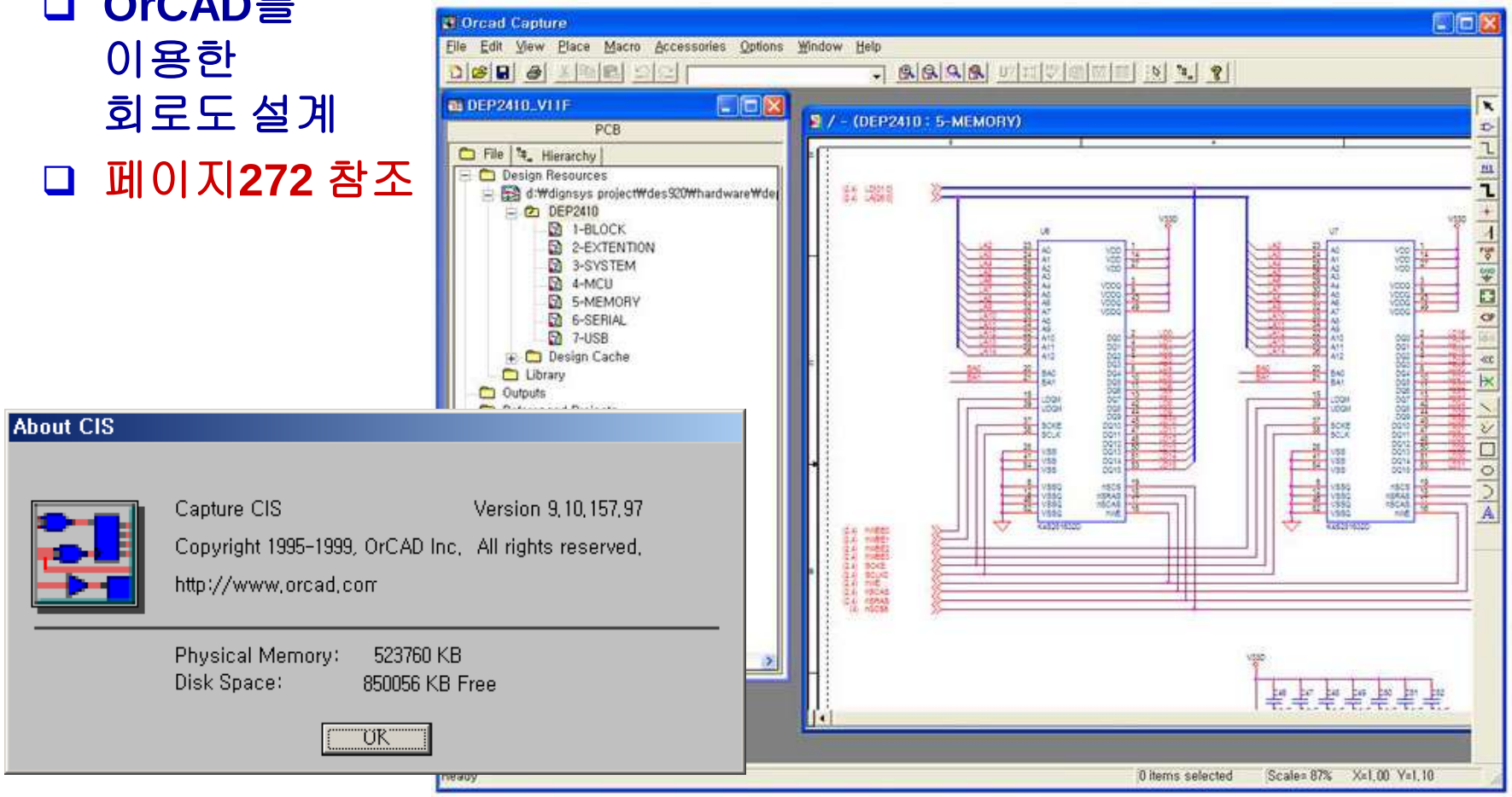
제품 요구에 따른 하드웨어 사양
(페이지 270, [표 10-2] 참조)

항목	설명	공급자 및 부품명
RTC	Real Timer Clock	SoC에 포함
LED	전원 공급 상태 LED	적색
	시스템 동작 상태 LED	녹색
	확장용 녹색 LED	녹색
스위치	리셋 스위치	PUSH 버튼
	입력 스위치	PUSH 버튼
JTAG	20 핀JTAG 커넥터	박스형
	ICE 인터페이스	
확장 커넥터	확장 보드 연결	4개의 50핀, 2.54mm 간격
전원	DC +5V 전원 아답터	
	보드내장 레귤레이터	LM1117-3.3, LM1117-1.8
TFT LCD	3.5인치, 24bpps	
	ADC 사용	
이더넷 제어기	10/100Mbps	SMC91111

목 차

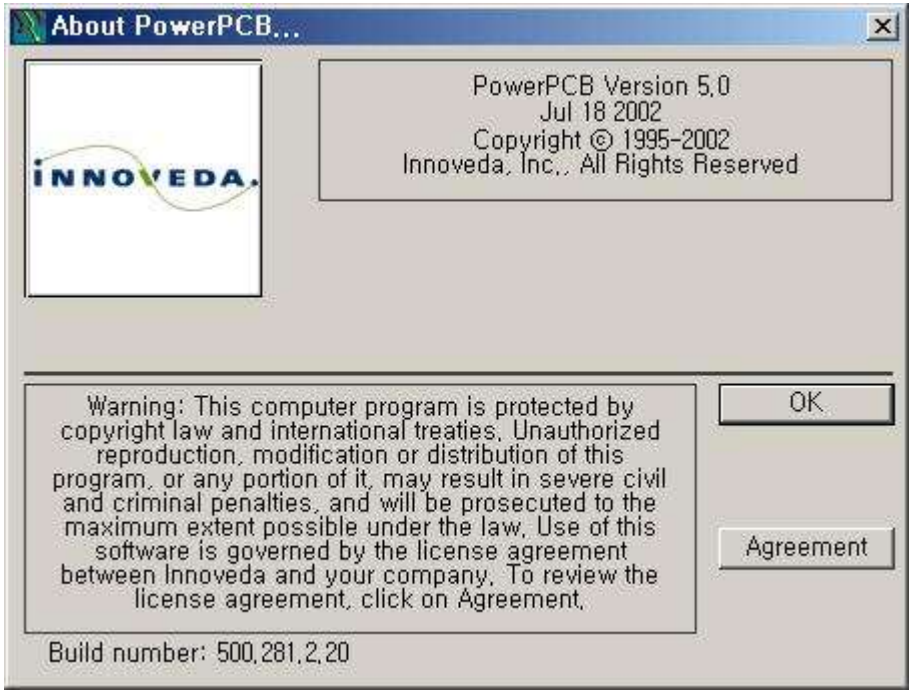
회로도 설계

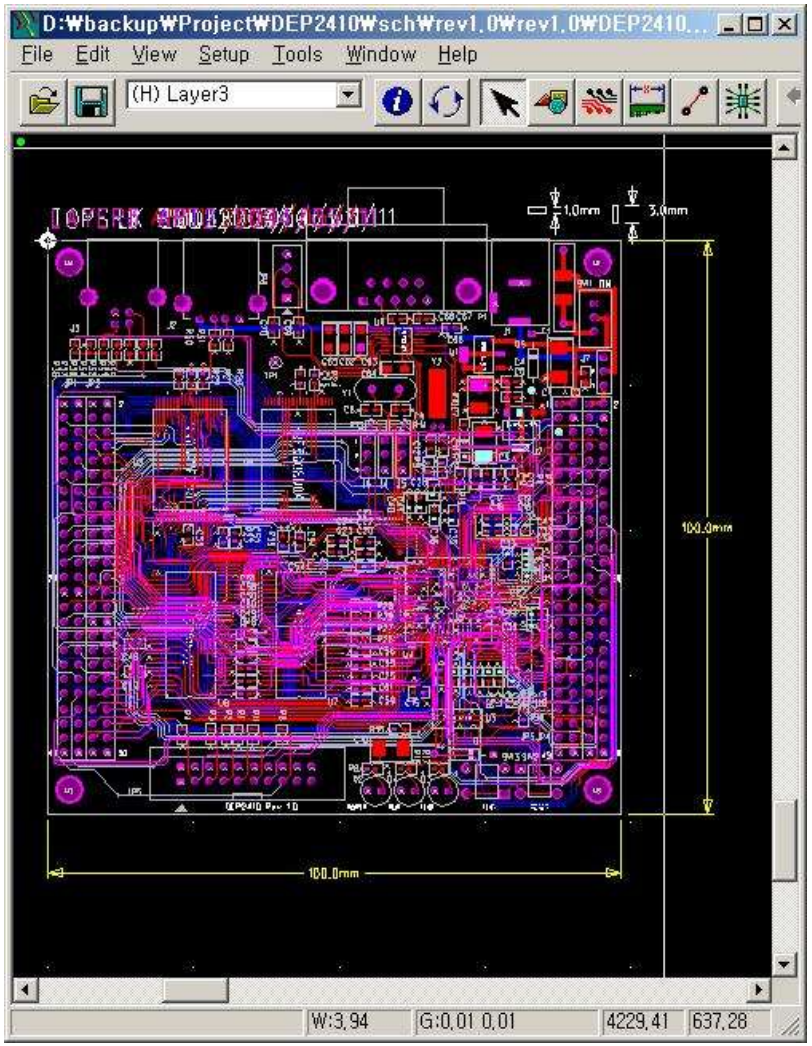
- **OrCAD를 이용한 회로도 설계**
- **페이지272 참조**



PCB 설계

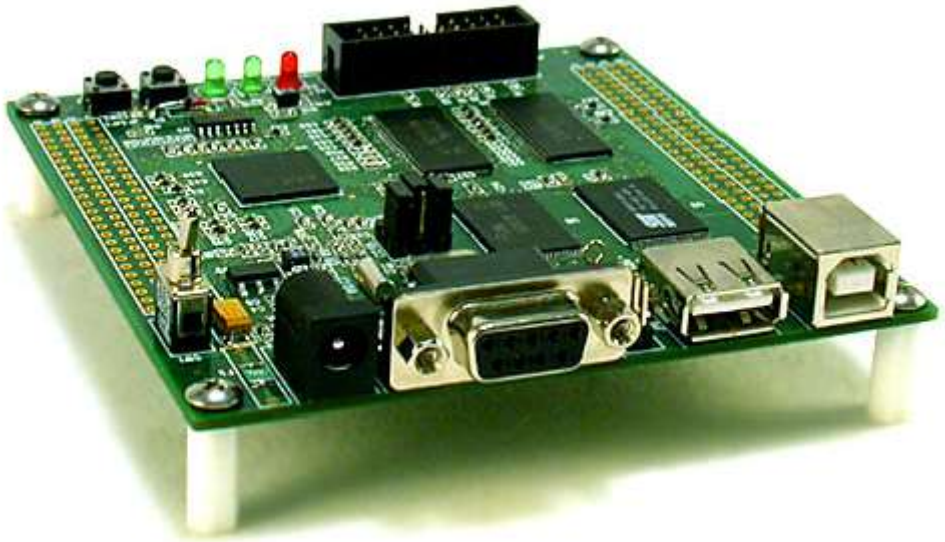
- **PowerPCB를 이용한 PCB 설계**





하드웨어 조립 및 시험

- **PCB**가 완료되면 각 부품을 조립하고 시험한다.



질의 응답