AVR - Chapter 1

뉴테크놀로지 컴패니
대표 류 대 우
davidryu@newtc.co.kr

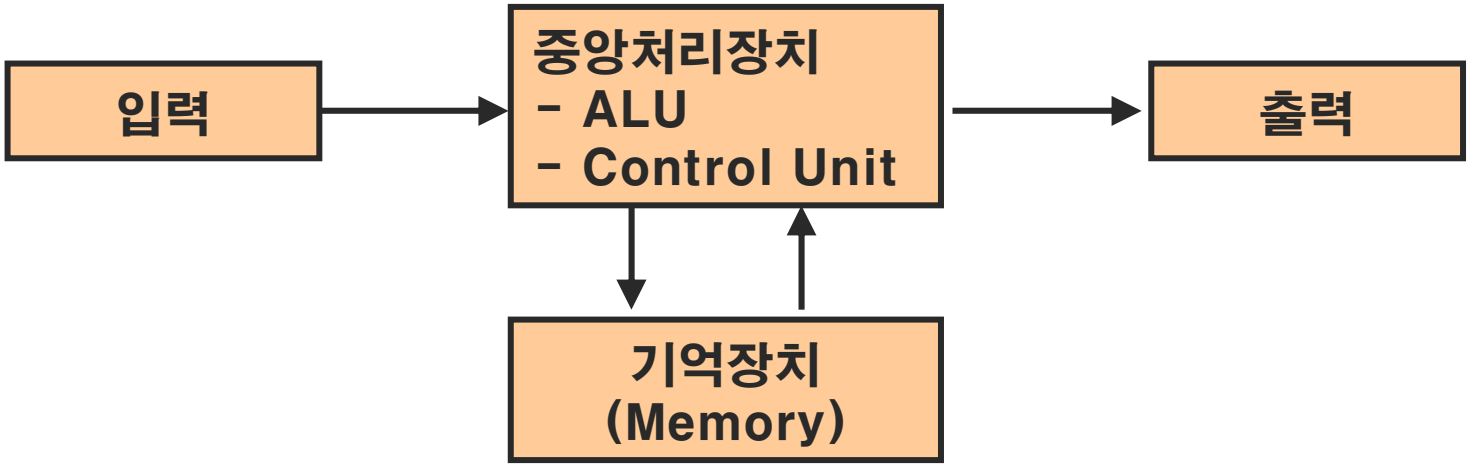
[프로세서의 역사]

- **전자 계산기의 발달**
  - 초창기의 컴퓨터는 부피가 크고, 신뢰성이 다소 떨어졌으나 CPU의 개발로 높은 신뢰도와 빠른 연산을 제공하는 컴퓨터가 개발되고 점차 컴퓨터 크기의 소형화가 이루어지고 있음

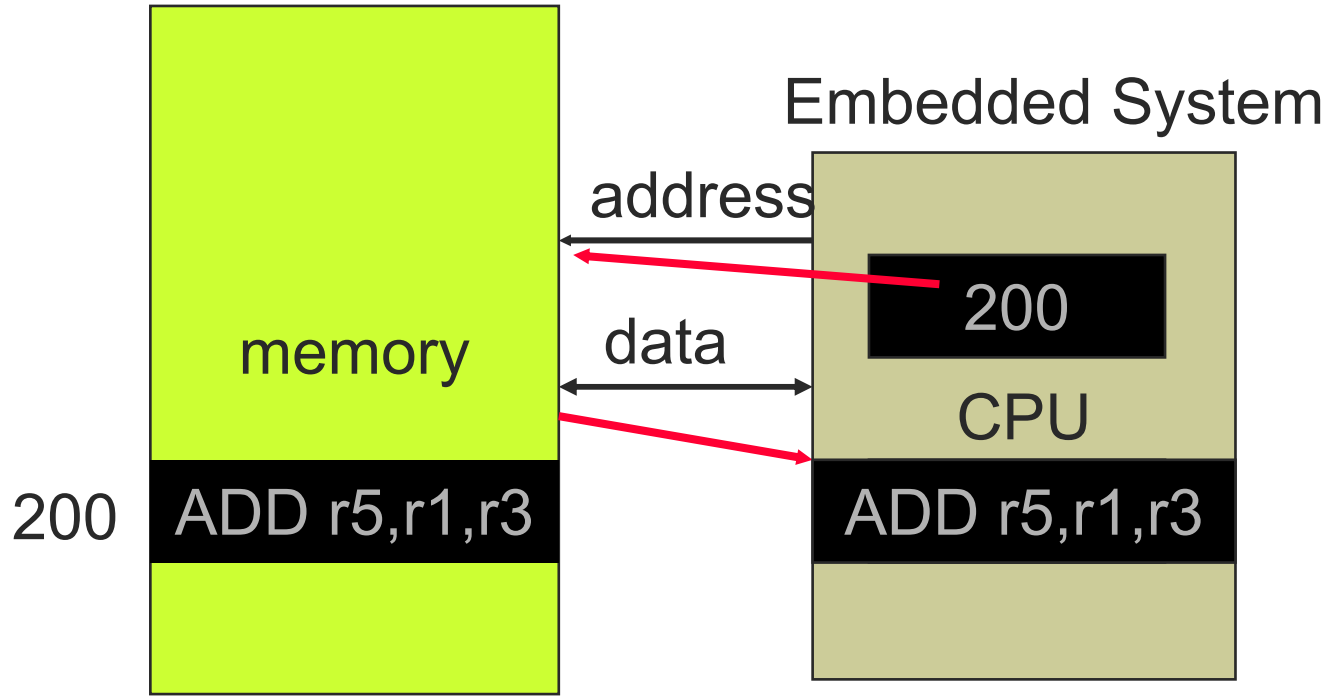
  - 전자 계산기 발달 순서
    - MARK-1
    - ENIAC
    - EDSAC
    - EDVAC
    - UNIVAC-1

[프로세서의 역사]

- **프로그램 내장형 컴퓨터**
  - 1945년 폰 노이만(Von Neumann)
    - 프로그램 내장형 컴퓨터 제창
  - 1949년 영국 캠브리지 대학 교수 윌키스(M. Wilkes)
    - EDSAC – 세계 최초의 프로그램 내장 방식 컴퓨터

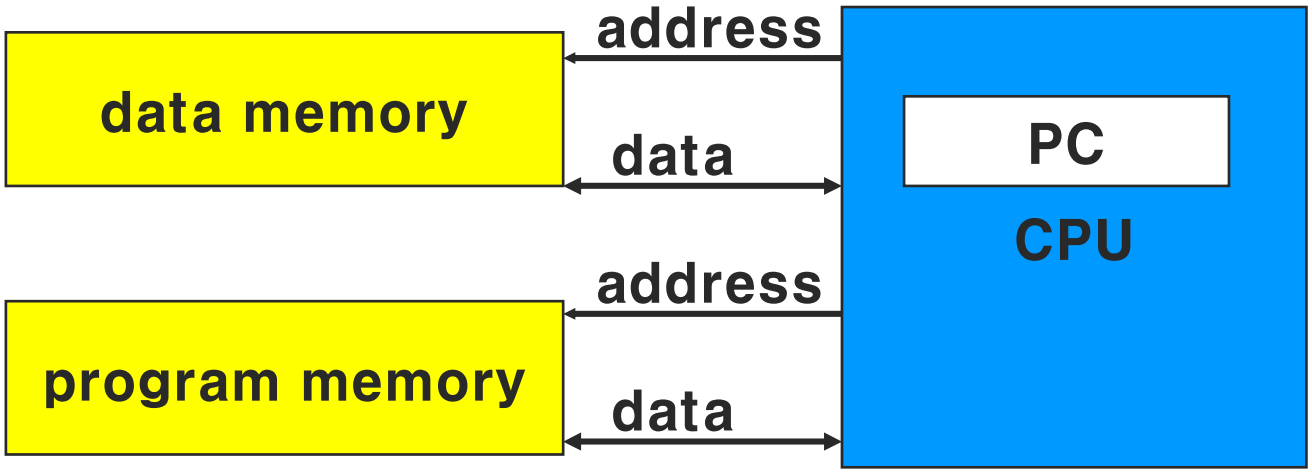


[폰 노이만 아키텍처]


Embedded System
address
data
memory
200
ADD r5,r1,r3
200
CPU
ADD r5,r1,r3

[하버드 아키텍처]

- Harvard can't use self-modifying code.
- Harvard allows two simultaneous memory fetches.
- Most DSP use Harvard architecture for streaming data:
  - greater memory bandwidth;
  - more predictable bandwidth.



[프로세서의 역사]

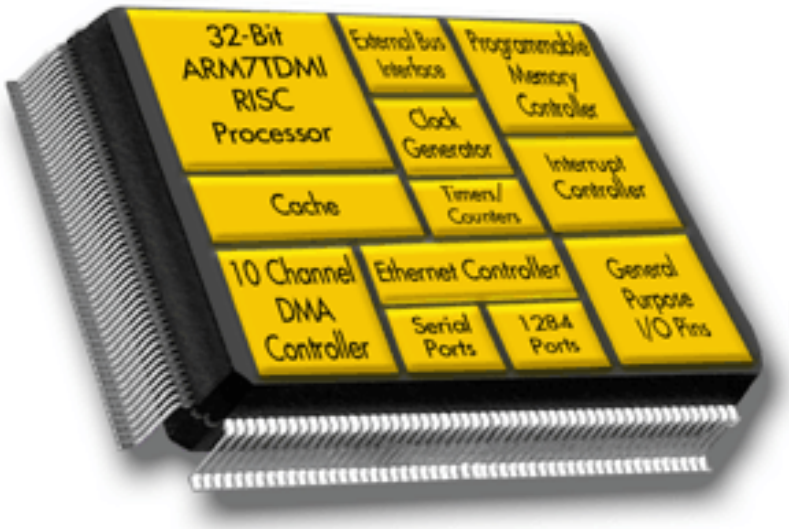
- Intel 의 마이크로 프로세서
  - 1971년 4004, 4bit, 0.5 MHz clock, 16 Kbyte Main Memory
  - 1973년 8080, 8bit
  - 1975년 8085, 8bit
  - 1978년 8086, 16bit
  - 1982년 80286, 16bit
  - 1985년 80386, 32bit
  - 1989년 80486, 32bit
  - 1990년대 80586(Pentium), 32bit
- 기타 마이크로 프로세서 제조업체
  - Motorola, 680계열(68000, 68020…), 주로 산업용으로 사용.
  - Zilog, Z80, Intel의 8085와 유사한 8비트 프로세서

[마이크로프로세서, 마이크로컨트롤러]

- **마이크로프로세서 vs. 마이크로컨트롤러**
  - **마이크로프로세서**
    - 프로세서를 한 개의 칩으로 구현
    - 데이터부와 제어부가 들어있는 코어를 의미
  - **마이크로컨트롤러**
    - 코어 + 여러 가지 크기와 다양한 종류의 메모리 + 여러 종류의 주변장치 + 여러 종류의 입출력 포트
    - 여러 응용분야에 필요로 하는 주변기기들을 한 칩 내에서 모두 제공
    - 시스템보드의 칩의 개수를 줄임
      - 경박단소 (經博短小)
      - 시스템을 성능 및 크기, 그리고 가격을 최적화
      - 제품을 제작 시 제품의 개발기간 및 가격 최소화
      - 제품의 용도에 맞게 프로그램의 변경만으로 적용이 가능하여 적용성 우수
      - 부품수가 적고 시스템 구성이 간단하여 신뢰성이 향상된다.

마이크로컨트롤러 vs. 임베디드프로세서

- **임베디드 프로세서**
  - 원래는 마이크로컨트롤러를 의미
  - 마이크로컨트롤러를 확장한 개념으로도 사용
  - CPU 코어, 메모리, 주변 장치, 입출력장치에 다양한 종류의 네트워크 장치가 추가되는 형태.
    - 10/100BaseT MAC(Medium Access Control) for Internet

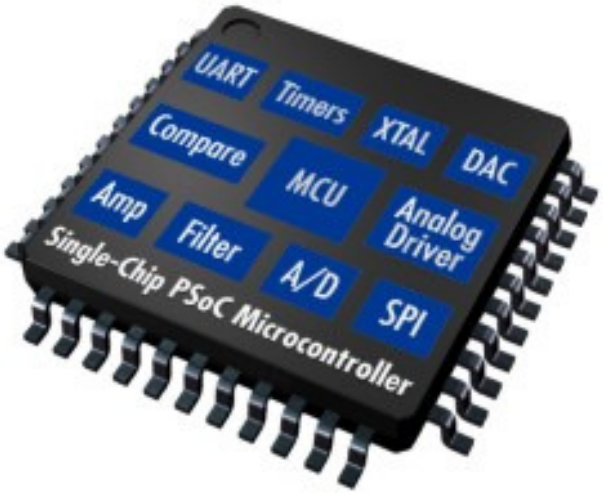


Netsilicon NET+ARM Embedded Processor

[임베디드프로세서 (1)]

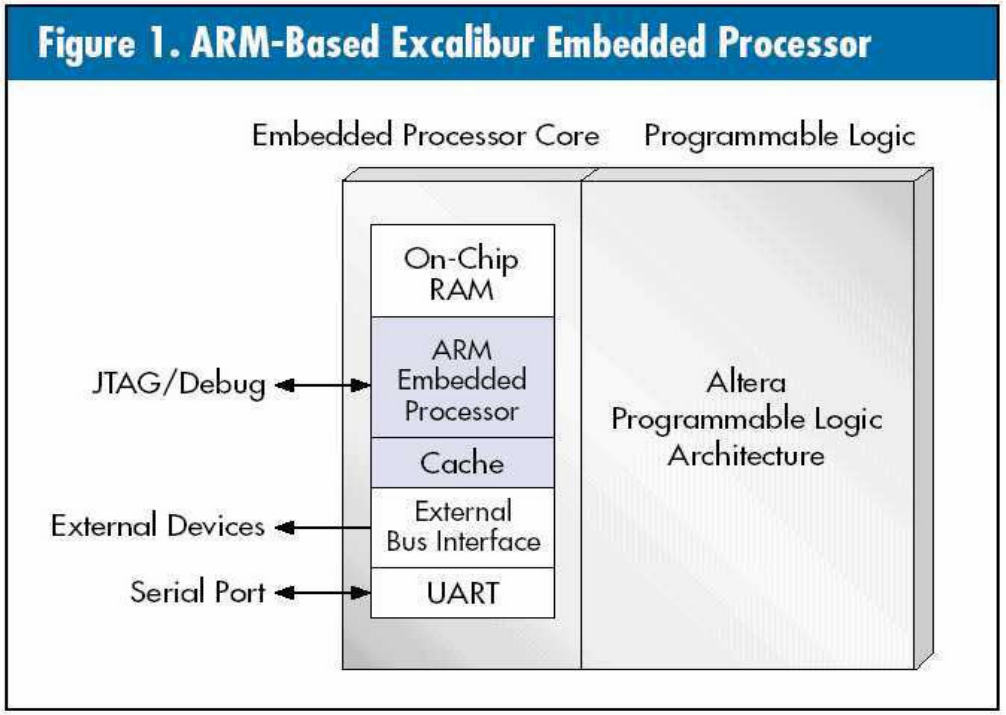
- 최근의 임베디드 프로세서 – 주변장치 선택형
  - 주변 장치들을 프로그래머블(programmable)하게 만들어서 사용자가 원하는 정확한 규격대로 주변장치를 선택적으로 구성
    - 칩 내부의 자원의 낭비를 줄일 수 있음
  - 예 : Cypress MicroSystems의 PSoC™ 마이크로컨트롤러
    - counter 혹은 timer를 구성할 때 8/16/24/32bit counter/timer들 중에서 하나를 선택
    - ADC(Analog to Digital Converter) 경우:

6-Bit SAR ADC, 8-Bit SAR ADC,
8 to 14-Bit Incremental ADC,
12-Bit Incremental ADC,
8-Bit Delta-Sigma ADC,
11-Bit Delta-Sigma ADC,
들 중에서 하나를 선택
(www.cypressmicro.com)



[임베디드프로세서 (2)]

- 알테라(Altera)사의 ARM core를 채택한 SOPC(system-on-a-programmable-chip)
  - PLD를 이용해서 IP를 구현
  - PLD를 ASIC화 해주는 Hard-copy Devices 프로그램을 지원
  - www.altera.com



[프로세서의 분류]

- 기능에 따른 분류 : 의미 없음
  - 많은 업체에서 여러 종류의 칩을 각각 특별한 기능을 내장하고 여러 응용분야에서 사용하기 때문
- 데이터 버스의 크기, 산술처리기(Arithmetic Logic Unit) 및 레지스터들의 크기에 따른 분류
  - 8비트 프로세서 : 응용목적이 간단한 경우
  - 16비트 프로세서 : 중간정도의 복잡도를 가지는 응용분야
  - 32비트 프로세서 : 내장형 시스템 시장에서 주류
  - 64비트 프로세서 : 몇몇 업체에서만 제공
- 명령어 구조에 따른 분류
  - Complex Instruction Set Computer (CISC)
  - Reduced Instruction Set Computer (RISC)
  - Extended Instruction Set Computer (EISC)

[프로세서의 분류]

- **기능에 따른 분류**
  - **MPU – Micro Processor Unit**
    - 중앙처리장치에서 주기억장치를 제외한 연산장치, 제어장치 및 각종 레지스터들을 단지 1개의 IC 소자에 집적시킨 것
    - 용도 : PC
  - **MCU – Micro Controller Unit**
    - 마이크로프로세서 중에 1개의 칩 내에 CPU 기능은 물론이고 일정한 용량의 메모리(ROM, RAM 등)와 입출력 제어 인터페이스 회로까지 를 내장한 것
    - MCU, 단일 칩 마이크로컴퓨터, 마이컴 등으로 불린다.
  - **DSP – Digital Signal Processor**
    - 디지털 신호를 하드웨어적으로 처리할 수 있는 집적회로
    - 용도 : 영상, 음성 등 신호처리

[프로세서의 분류]

- **데이터 버스의 크기, 산술처리기(Arithmetic Logic Unit) 및 레지스터들의 크기에 따른 분류**

  프로세서가 한번에 처리할 수 있는 비트 수로 분류

  - **8-Bit 프로세서**
    - 1970년대 마이크로 프로세서
    - 현재는 소용량 마이크로 컨트롤러가 8-Bit 프로세서이다.
    - 8051, AVR, PIC, SAM88
  - **16-Bit 프로세서**
    - 1980년대 초 마이크로 프로세서
    - 현재는 고속의 마이크로 컨트롤러로 사용된다. 8-Bit 와 32-Bit 의 발달로 많이 사용하지 않는다.
    - AM188, MSP430
  - **32-Bit 프로세서**
    - 1980년대 중반 이후 현재까지 마이크로 프로세서
    - PC, 또는 대용량의 데이터 처리를 하는 산업용 장치에 쓰인다.

[프로세서의 분류]

- 프로세서의 타입
  - 프로세서가 제공하는 명령어 세트 (Instruction Set)의 복잡도
  - Complex Instruction Set Computer(CISC)
    - 복잡한 명령어를 지원하는 프로세서
    - 1980년대 이후 마이크로프로세서 시장에 처음 선을 보임
    - 인텔, 모토롤러 프로세서
  - Reduced Instruction Set Computer(RISC)
    - 단순한 명령어 세트를 지원하는 프로세서
    - MIPS와 ARM의 프로세서
- RISC와 CISC 두 종류의 프로세서의 차이점
  - CISC : 복잡한 많은 수의 명령어를 해독하는데 많은 실리콘을 사용
  - RISC : 작은 명령어 세트로 인하여 남는(CISC에 비해서) 실리콘을 다른 유용한 목적으로 사용
    - 이전 : RISC와 CISC의 관계는 서로의 우월성을 주장하던 관계
    - 최근 : post RISC: 성숙한 단계로 접어들어서 상호간의 장·단점을 CPU 구조에 적절하게 반영하여 최적의 성능 제공

[프로세서의 분류]

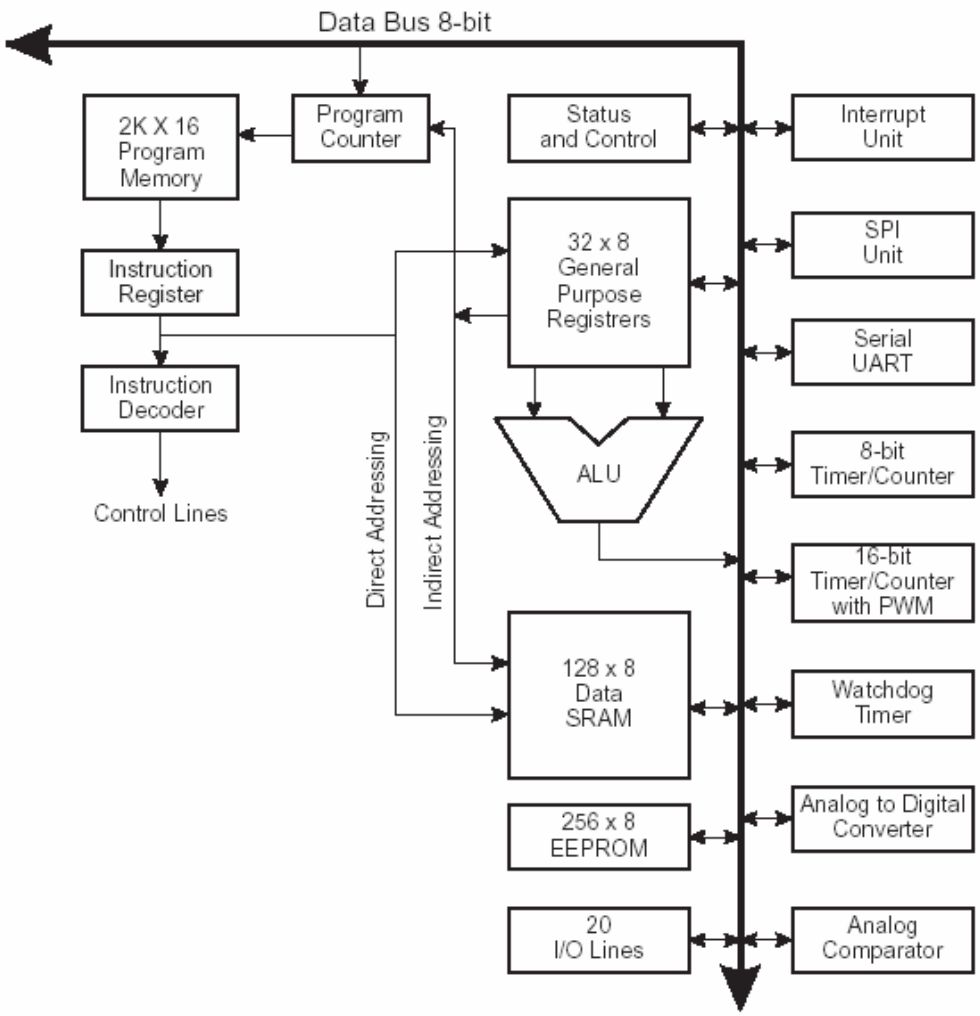
- CISC, RISC, EISC의 특징

구분	CISC	RISC	EISC
CPU instruction	명령어 개수가 많고, 그 길이가 다양하며 실행사이클도 명령어마다 다름	명령어길이는 고정적이며, 워드와 데이터 버스 크기가 모두 동일, 실행사이클도 모두 동일	16비트 명령을 사용하여 32비트 데이터를 처리
회로 구성	복잡	단순	단순
메모리 사용	높은 밀도 메모리 사용이 효율적	낮은 밀도의 명령어사용으로 메모리 사용이 비효율적	코드밀도가 높다(임베디드 시스템에 유리)
프로그램측면	명령어를 적게 사용	상대적으로 많은 명령어가 필요, 파이프라인 사용	RISC보다 더 깊은 파이프 라인 스테이지
컴파일러	다양한 명령을 사용하므로 컴파일러가 복잡해짐	명령어 개수가 적어서 단순한 컴파일러 구현 가능	국산 기술에 의해 개발(ADC corp.)

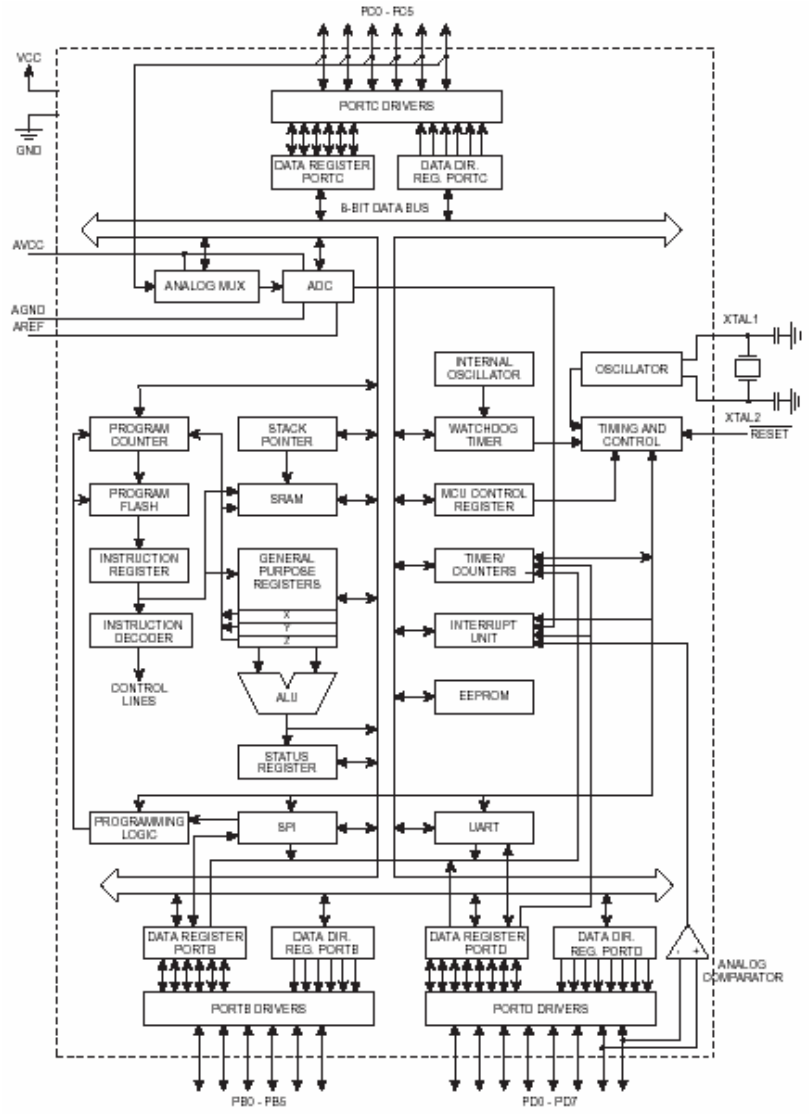
[마이크로 컨트롤러 - AVR]

- AVR은 ATmel사가 개발 제조하고 있는 유사 RISC 구조의 저전력 CMOS 8-bit 마이크로 컨트롤러
  - 임베디드 환경에 적합한 저전력 소모의 칩이며, 내부에 32개의 범용 레지스터 내장
  - 유비쿼터스 및 센서 네트워크를 위한 소형 시스템에 적합
  - 유사 RISC 아키텍처(RISC like Architecture) - 유사 RISC 아키텍처라고 하는 것은 RISC 구조를 따르고는 있으나 명령어와 주소 지정방식의 수가 RISC에 비해 많음
- AVR의 종류
  - Tiny(ATtiny) : 소형시스템에 적합, 외형핀 8 ~ 20개, 메인 마이컴보다 여러 회로를 하나의 소자로 구현한 보조 역할의 칩으로 사용
  - Classic (AT90S): 중간 규모 시스템의 제어에 적합, 모뎀. 스마트 카드리더. 셋탑 박스 및 각종 자동화기기에 응용
  - Mega (ATmega) : 대형이고 입/출력 핀의 수가 많은 시스템 에 적합, 무선 전화기 및 프린터용 제어기. FAX 및 CD-ROM제어기, 통신장비 등에 응용

마이크로 컨트롤러–AVR 내부 블럭도



마이크로 컨트롤러–AVR 내부 블록도 (2)



마이크로 컨트롤러 – AVR 특징

- AVR 의 특징
  - 프로그램을 다운로드하여 저장 할 수 있는 2K ~128K 바이트의 플래시 메모리를 내장
  - 내부 EEPROM 용량 : 128에서 512바이트까지 가능
  - 내부 오실레이터에 의해 동작하는 워치독 타이머와 전 이중방식의 다양한 외부 접속장치를 내장
  - 다양한 인터럽트 소스와 파워 절약모드, 외부 메모리 추가가 가능
  - 내장 디바이스
    - 타이머/카운터, 직렬 통신 장치, 아날로그 비교기, 외부 데이터 장치 인터페이스, 입출력 포트, SPI(Serial Peripheral Interface), 아날로그 디지털 변환기(ADC)

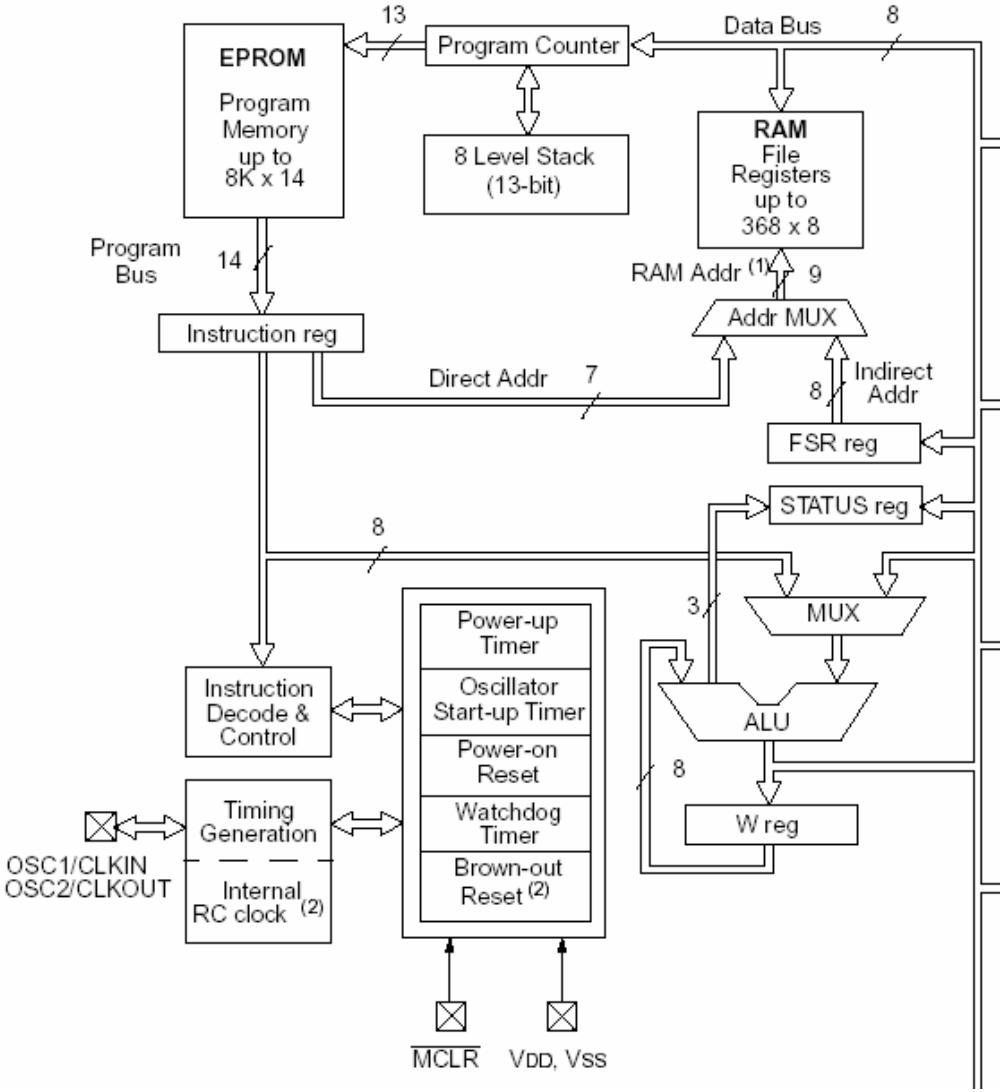
[마이크로 컨트롤러 - PIC]

- PIC 는 마이크로 칩 테크놀로지(Microchip Technology)사에서 개발 제조 판매하는 RISC 구조의 8-bit 마이크로 컨트롤러임
  - 속도, 내부 메모리 용량, 내장 디바이스 특성에 따라 PIC12, PIC14, PIC16, PIC17, PIC18 시리즈가 있음

- 모두 동작 전원 범위가 넓고, 소비 전류는 수 mA 이하이며 파워 ON Reset과 원칩 타이머, ROM 등을 내장하고 있어서, 소형 제품을 만드는 데 적합한 특징을 가지고 있다

- EEPROM, 13개의 I/O 핀, 오실레이터 회로, 리셋 회로, 타이머/카운터 등을 내장하고 있는 아주 편리한 원 칩 마이컴(one chip micom)이다. 동작 범위도 2.0V~6.0V로 확대되었으나, 소비전류는 60μA 정도로 적기 때문에, 건전지 등과 같은 것으로도 동작이 가능

[마이크로 컨트롤러 – PIC]



[마이크로 컨트롤러 - PIC]

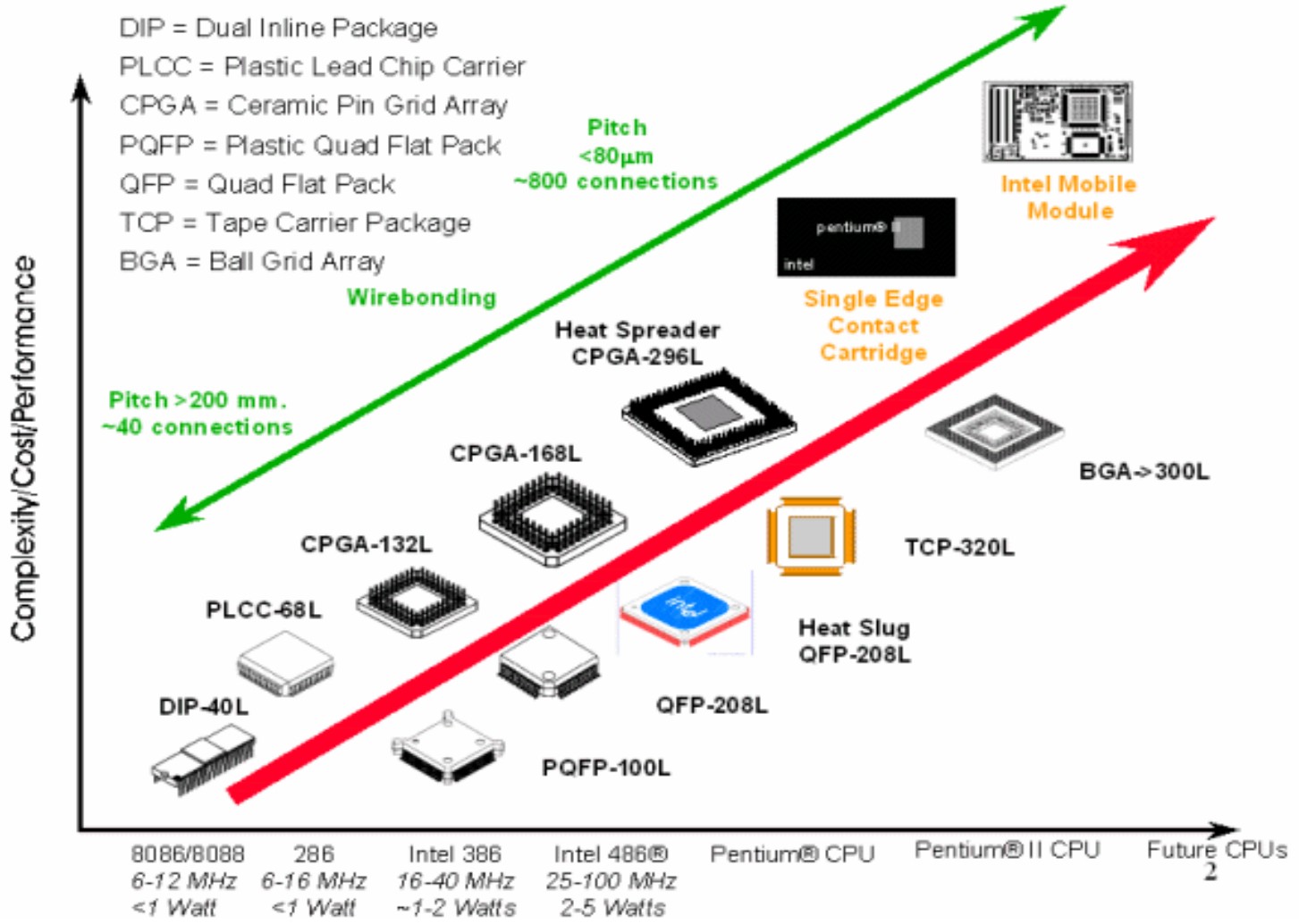
- PIC의 특징
  - 35개의 싱글 워드 명령
  - 브랜치 명령(2 사이클)을 제외하고는 모든 명령 싱글 사이클 (400ns)
  - 동작 속도 : DC-10MHz 클럭 입력, DC-400ns(명령 사이클)
  - 14비트 크기의 명령어
  - 8비트 데이터 폭
  - 1K×14 EEPROM 프로그램 메모리 내장
  - 36×8 범용 레지스터(SRAM)
  - 8레벨의 하드웨어 스택
  - 직접, 간접, 상대 번지 지정
  - 4개의 인터럽트 소스
    - 외부 RB0/INT 핀, TMR0 타이머 오버플로
    - PORTB<7:4>가 변할 경우 인터럽트, 데이터 EEPROM 라이트

[CISC 프로세서]

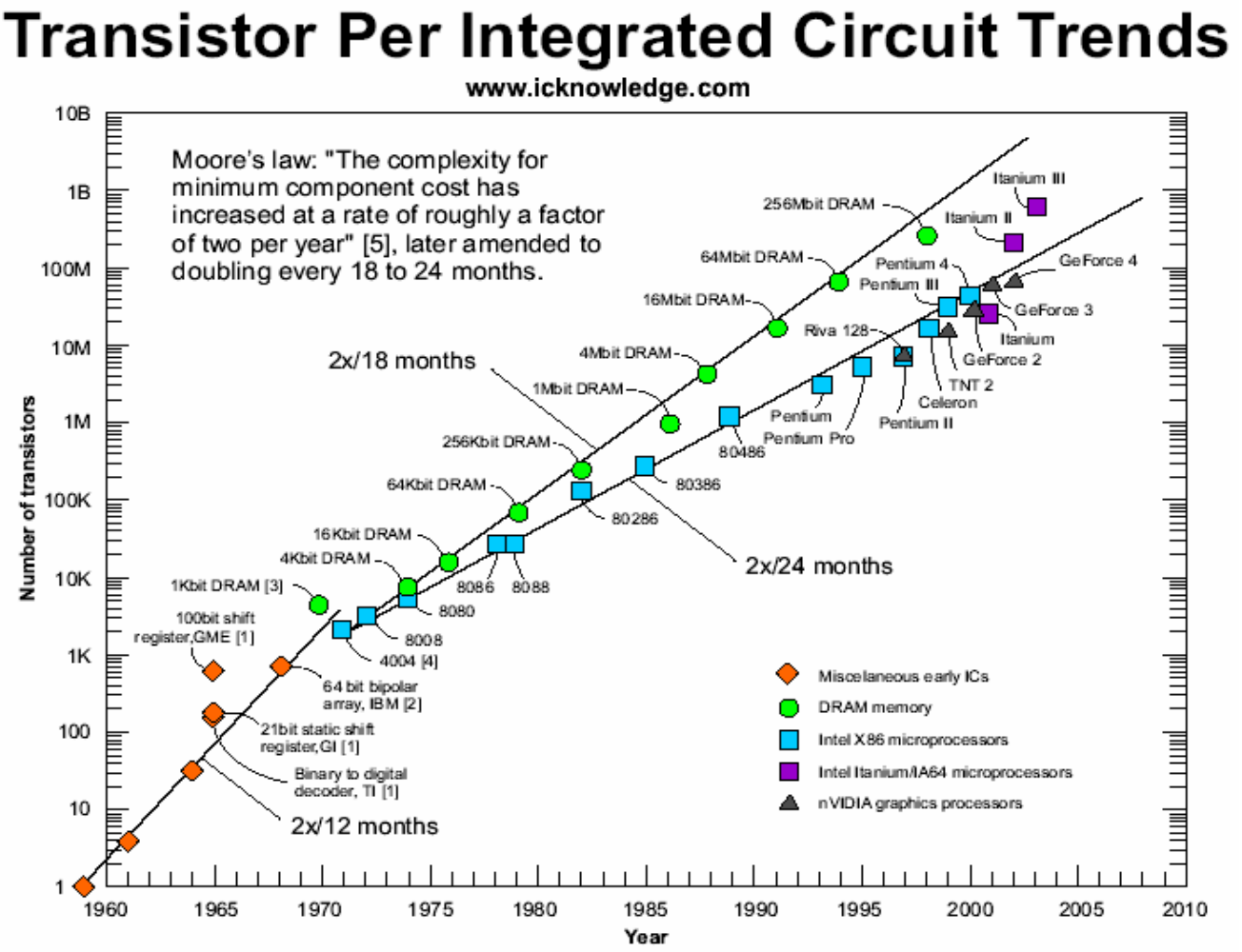
- Intel 계열 마이크로프로세서의 종류 및 역사

연도	프로세서 이름	트렌지스터 개수	특징
1971	4004	2,250	인텔의 첫 마이크로 프로세스, Busicom 계산기에 사용
1972	8008	2,500	Mark-8에서 사용, 최초의 가정용 컴퓨터
1974	8080	5,000	Altair에서 사용
1978	8086/8088	29,000	IBM-PC XT에서 사용, 인텔이 대기업으로 성장
1982	80286	120,000	IBM-PC AT에서 사용, 6년간 천 5백만대 판매
1985	80386	275,000	32비트 멀티 테스킹 지원
1989	80486	1,180,000	수치 보조 프로세서 내장
1993	Pentium	3,100,000	음성, 이미지 처리 기능 강화
1995	Pentium Pro	5,500,000	Dynamic Execution 구조 채택
1997	Pentium 2	7,500,000	MMX 기술 지원
1999	Pentium 3	24,000,000	SIMD 지원, 12 스테이지 파이프라인
2001	Itanium	25,000,000	64비트, Explicitly Parallel Instruction Computing(EPIC)
2002	Pentium 4	55,000,000	20 스테이지 하이퍼 파이프라인, 하이퍼 쓰레딩
2003	Itanium 2	410,000,000	Machine Check Architecture, EPIC, 6MB L3 캐시

[CISC-History:Packaging기술 변천]



[CISC – History]



[CISC - 8051 인터럽트]

- **8051 입출력 및 인터럽트**
  - 8051에서 인터럽트 : 5개 지원
    - 외부 인터럽트 2개(INT0, INT1)
    - 타이머/카운터 인터럽트 2개(TF0, TF1)
    - 시리얼 포트 인터럽트 1개
- **인터럽트 처리 순서**
  - 1. EA(Enable All) bit를 1로 만든다.(인터럽트를 허용하는 상태로 만든다)
  - 2. IE 레지스터의 인터럽트 enable 비트를 1로 만든다.
  - 3. 해당되는 인터럽트 벡터주소로 분기해서 원하는 인터럽트 서비스 루틴을 실행한다.
  - 4. 외부 인터럽트인 경우에는 INT0 혹은 INT1 비트가 '0'이 되어야 한다.

[CISC-8051 인터럽트 레지스터]

- **8051 입출력 및 인터럽트**
  - **인터럽트 발생 source에 따른 벡터 주소**

인터럽트 source	인터럽트 벡터 주소
IE0(외부 인터럽트 0)	0003(H)
TF0(타이머 0)	000B(H)
IE1(외부 인터럽트 1)	0013(H)
TF1(타이머 1)	001B(H)
R1 & T1(시리얼포트)	0023(H)

  - **IE(Interrupt Enable)레지스터**

IE bit 위치	IE7	IE6	IE5	IE4	IE3	IE2	IE1	IE0
Bit 이름	EA	-	-	ES	ET1	EX1	ET0	EX0
의미	전체 INT			시리얼 포트	타이머 1	외부 INT 1	타이머 0	외부 INT 0

[RISC]

- **RISC 개요**
  - RISC의 대표적인 칩 : ARM
  - ARM : Advanced RISC Machine
    - 2002년 현재 RISC(Reduced Instruction Set Computer) 구조를 가진 프로세서들 중에서 세계 시장 점유율 1위
  - ARM 프로세서가 다른 프로세서와 차별화 된 점
    - 구조적인 점
      - 인 텔 (Intel) 이 나 모 토 롤 러 (Motorola) 의 CISC(Complex Instruction Set Computer) 프로세서과 비교
    - 세부적인 (혹은 최적화된) 구현 방법
      - 같은 종류인 RISC 프로세서들과 비교

[RISC vs. CISC]

- load-store architecture:
  - 메모리에서 데이터를 읽고 쓰는 명령어와 CPU 내부에서 연산을 수행하는 명령어들을 분리
  - 따라서 연산을 수행하면서 메모리까지 참조하기위해 많은 시간을 소모하는 명령어가 없게 됨
- fixed length 32-bit 명령어:
  - 모든 명령어는 한 명령어 당32비트만을 점유
- 3-address 명령어 구조:
  - 대부분의 ARM 명령어는 두 개의 source address 와 1개의 destination address, 즉 3 개의 address를 보유

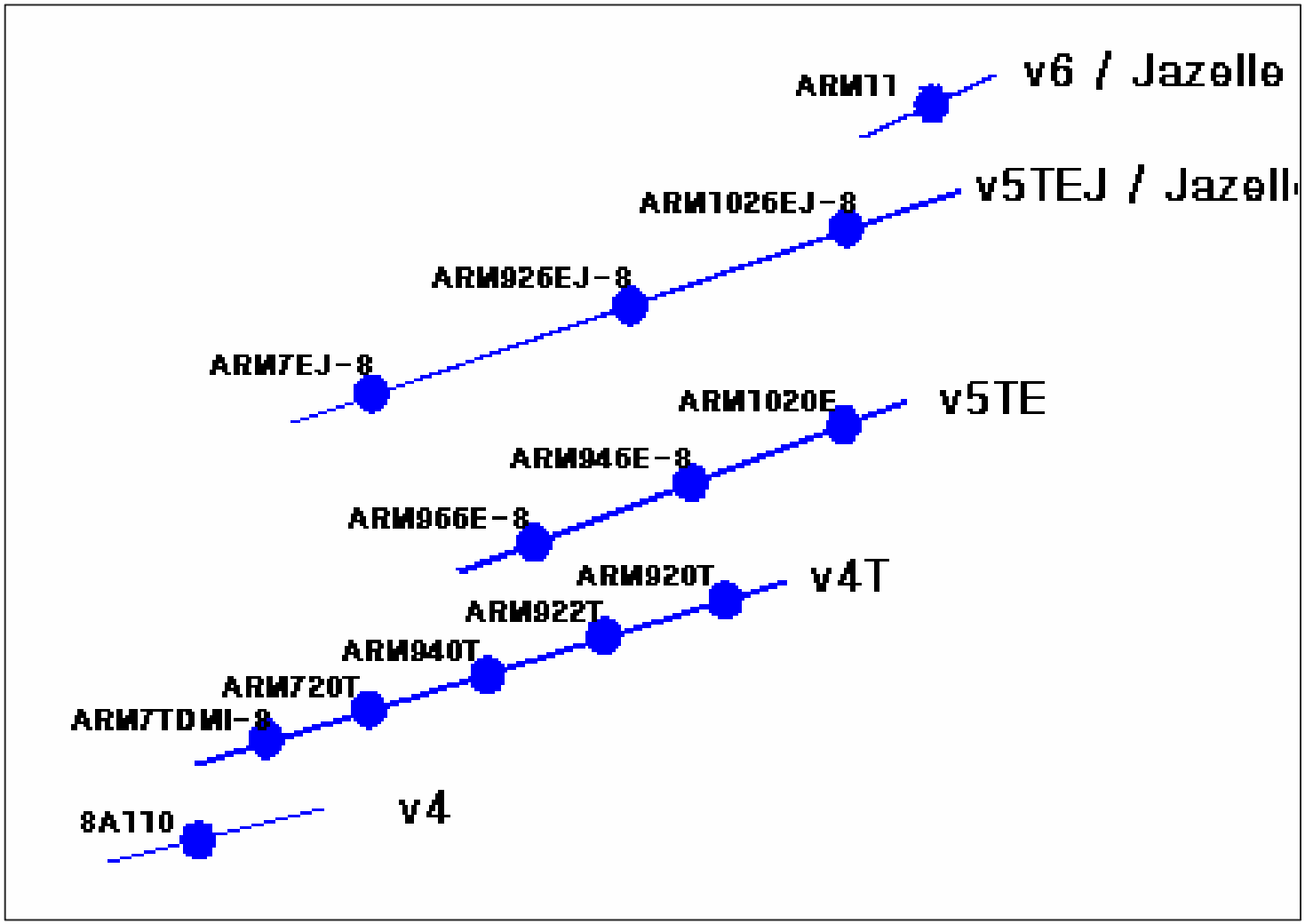
[RISC의 장점]

- **ARM이 임베디드 프로세서 시장을 석권할 수 있었던 이유**
  - 위의 장점들을 살리면서도 동시에 다른 RISC CPU 와 차별화
  - 32-bit RISC의 성능
  - 작은 메모리
  - 저가격
  - 저전력
  - 이동 단말기 시장에서 채택
    - 이동전화시장 즉, GSM과 CDMA 이동통신 시장의 폭발
  - 임베디드 프로세서 시장의 선두주자

[RISC - ARM RISC의 특징]

- **ARM RISC vs. other RISC**
  - 1.16-bit 명령어 사용가능:
    - Thumb모드 제공
    - RISC CPU의 단점인 큰 코드 크기 문제(code size problem)를 완화
  - 2.조건부 명령어 구조:
    - ARM 의 모든 명령어는 조건부 수행명령으로 만들 수 있음
    - 분기 명령을 따로 사용하지 않고도 일반 명령어만 이용해서 프로그램을 구현
    - 명령어의 개수를 줄일 수 있음
  - 3.최적화된 레지스터 수:
    - shadow register 이용
    - 꼭 필요한 레지스터들만 context-swiching에서 활용
  - 4.배럴 쉬프트 레지스터(Barrel shift register)의 활용:
    - 배럴 쉬프트 레지스터를 ALU 입력단에 설치
    - 간단한 상수를 즉석에서 만들거나, 간단한 곱셈이 필요한 경우 혹은 쉬프트동작이 연산명령어와 함께 수행되어야 하는 경우에 명령어를 따로 써야 할 필요가 없게 되었고 프로그램의 수행속도를 줄일 수 있음

[RISC-ARM Processor Roadmap]



[RISC - ARM Core 종류]

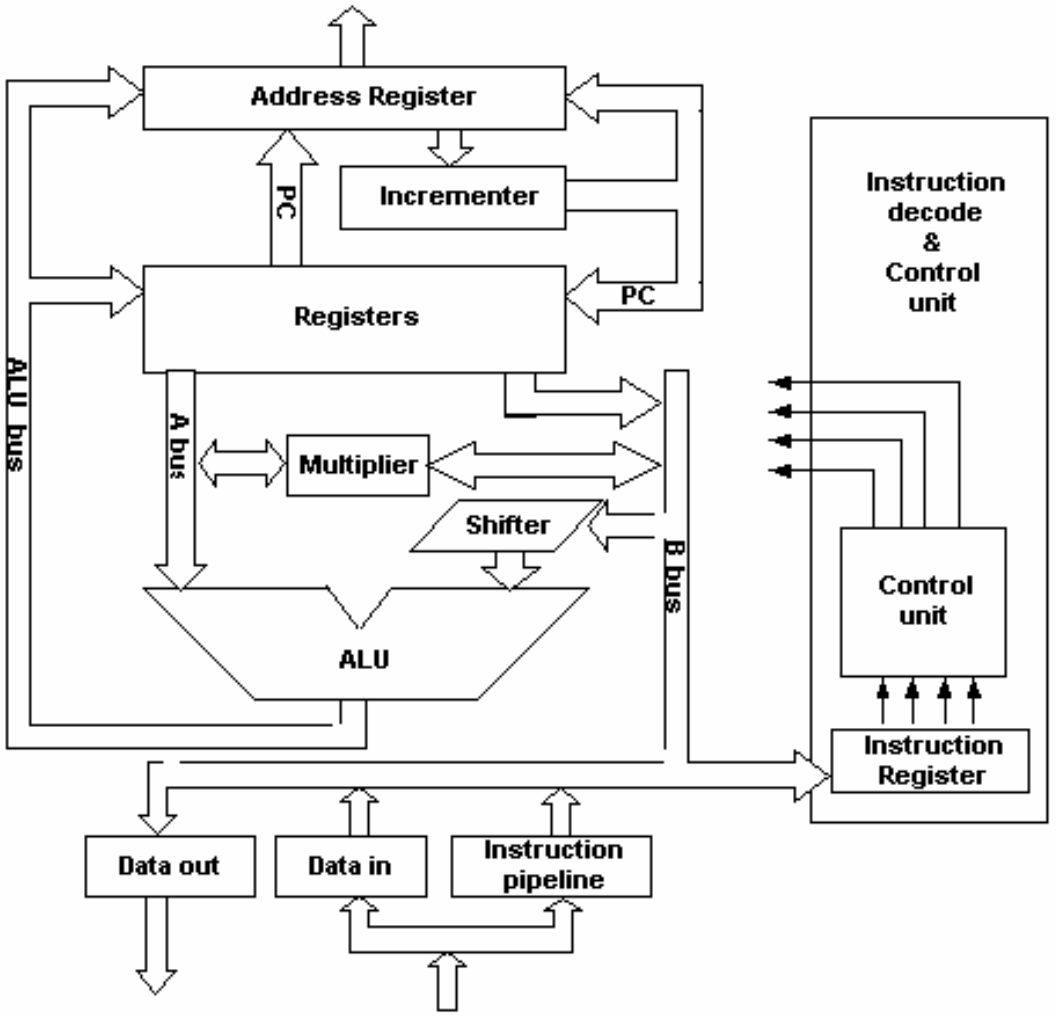
- **ARM 프로세서 코어의 종류**

Product Type	Product	Cache Size (Inst/Data)	Memory Mgnt.	AHB bus I/F	Thumb	DSP	Jazzlle	Clock (MHz)
Embedded Cores	ARM7TDMI	No	No	Yes	Yes	No	No	133
	ARM7TDMI-S	No	No	Yes	Yes	No	No	100-133
	ARM7EJ-S	No	No	Yes	Yes	Yes	Yes	100-133
	ARM966E-S	No	No	Yes	Yes	Yes	No	230-250
	ARM940T	4K/4K	MPU	Yes	Yes	No	No	180
	ARM946E-S	Variable	MPU	Yes	Yes	Yes	No	180-210
	ARM1026EJ-S	Variable	MMU+MPU	Dual AHB	Yes	Yes	Yes	266-325
Platform Cores	ARM720T	8K Unified	MMU	Yes	Yes	No	No	100
	ARM920T	16K/16K	MMU	Yes	Yes	No	No	250
	ARM922T	8K/8K	MMU	Yes	Yes	No	No	250
	ARM926EJ-S	Variable	MMU	Dual AHB	Yes	Yes	Yes	220-250
	ARM1020E	32K/32K	MMU	Dual AHB	Yes	Yes	No	325
	ARM1022E	16K/16K	MMU	Dual AHB	Yes	Yes	No	325
	ARM1026EJ-S	Variable	MMU+MPU	Dual AHB	Yes	Yes	Yes	266-325
	ARM1136J(F)-S	Variable	MMU	Quad AHB	Yes	Yes	Yes	333-400

[RISC - ARM 프로세서 구조]

- **ARM7 데이터부의 구성**
  - Address Register, Data-In Register, Data-Out Register, Instruction Pipe(Pipeline) Register
    - CPU와 메모리와의 인터페이스
  - 주소 레지스터(Address Register)
    - 메모리에서 읽어오거나(memory read) 메모리에 써야 하는(memory write) 경우 데이터의 주소를 지정하는 용도
  - 데이터 레지스터(Data Register)
    - 메모리에서 읽어오거나 저장해야 할 데이터를 보관하는데 사용
    - 데이터 입력 레지스터 (Data-In Register)
    - 데이터 출력 레지스터(Data-Out Register)
  - Instruction Pipeline Register : 명령어가 저장

[RISC - ARM7 블록도]



[RISC - ARM7 Registers]

- 레지스터 (register) : 총 37개의 레지스터
  - 사용목적에 따라서 구분해서 사용
  - 12개의 사용자 레지스터
  - 25개의 시스템 레지스터
  - 총 3개의 입력포트와 2개의 출력포트를 지원
    - 범용 레지스터를 위한 두 개의 입력포트(port)
    - 한 개의 출력포트
    - 프로그램 카운터(Program Counter 또는 PC)를 위한 별도의 입력 및 출력포트

[RISC - ARM7 Datapath]

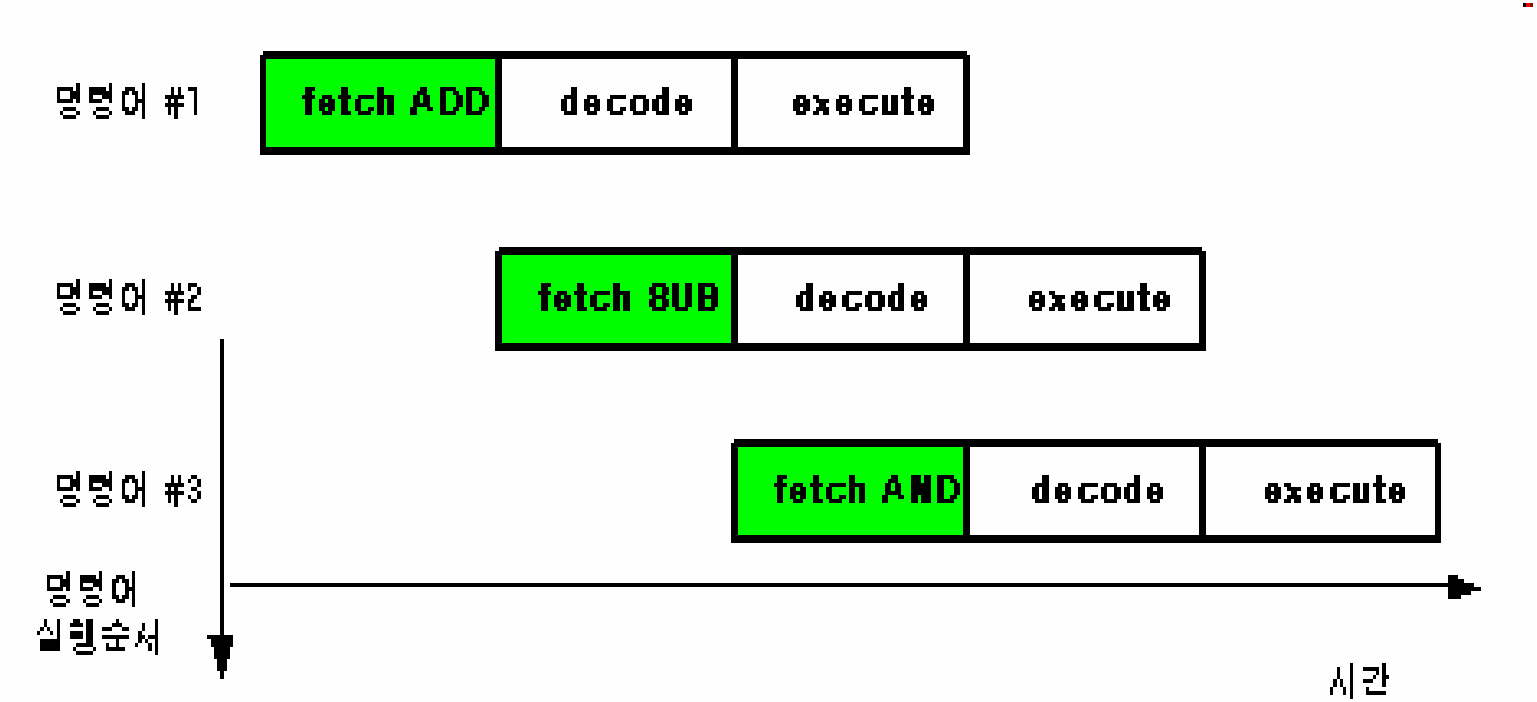
- 쉬프터(shifter)
  - ALU의 두 입력 중에서 왼쪽 입력 단에 위치
  - ALU로 들어가는 데이터를 쉬프트(한비트 자리옮김) 시키는데 사용
  - barrel shifter
    - 소요되는 시간이 비트 수에 관계없이 일정
- ALU(Arithmetic and Logic Unit, 산술논리연산장치)
  - 명령어가 필요로 하는 산술연산 및 논리연산을 수행
- 제어부
  - 명령어 디코딩(decoding) 모듈과 데이터부를 관리하는 제어신호를 만드는 역할을 수행
- 데이터 처리 명령어
  - 단일 명령어 사이클만을 필요
  - 두 개의 레지스터 오퍼랜드
  - ABUS에 실리는 데이터 : ALU로 곧장 보내짐
  - BBUS를 통과하는 데이터 : 배럴 쉬프터에서 쉬프트되어서 ALU로 보내짐
- 프로그램 카운터 (PC)값
  - 주소 레지스터로 보내져서 명령어를 가져오는(fetch)데 사용
  - PC값은 다음 명령어를 위해 Incrementer에서 4만큼 증가 시켜서 PC인 r15에 다시 씌어짐

[RISC – ARM7 Pipeline]

- **ARM7 파이프라인(Pipeline)의 구성**
  - 3단계 파이프라인의 구조
  - 한 CPU는 각 단계에 세 개의 명령어까지 동시에 처리 가능
  - 1.Fetch cycle : 명령어를 메모리에서 읽어서 명령파이프 레지스터 (instruction pipeline register)로 옮긴다.
  - 2.Decode cycle : 명령어를 해독해서 다음 단계에서 데이터부에서 필요로 하는 제어 신호들을 생성한다. 이 단계에서는 데이터부에서는 아무 일도 하지 않고 제어부에서만 일을 한다.
  - 3.Execute cycle : 이 단계에서는 데이터부에서만 일을 한다. 레지스터 뱅크에서 두 오퍼랜드를 읽어서 한 오퍼랜드를 지정된 횟수만큼 쉬프트해서 ALU에 입력한다. ALU는 이 두 데이터를 입력 받고 지정된 연산을 수행해서 결과 값을 다시 destination 레지스터에 저장한다.
  - CPU가 간단한 데이터 처리 명령만 수행하는 경우
    - 한 명령어가 처리되는데 3 클럭 사이클이 소요
    - 3단계 파이프라인에서는 한 클럭 사이클마다 한 명령어가 처리
    - 한 명령어에 대해서 응답시간 또는 latency : 3 클럭 사이클

[RISC - ARM Pipeline 개념도]

- ARM7 파이프라인(Pipeline)의 구성

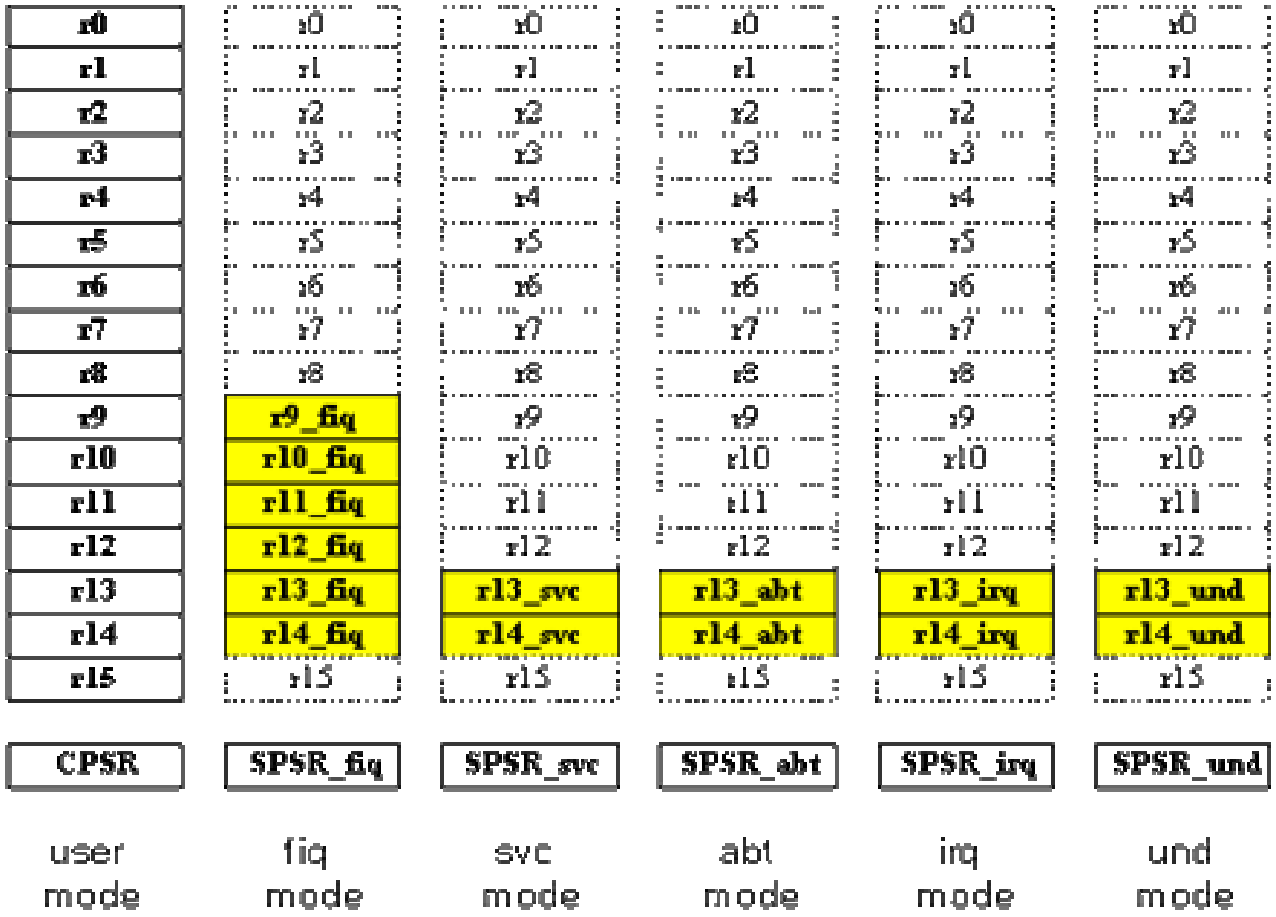


[RISC - ARM7 Operating Modes]

- **ARM 7 레지스터의 종류, flag 및 예외처리**
  - 사용자 프로그램 작성
    - 15개의 범용 32-bit 레지스터 (r0 - r14)
    - program counter(r15)
    - current program status register(CPSR)들을 사용
  - 시스템 프로그램 모드
    - 총 6가지로 분류
    - Irq(Interrupt request), Abt(Abort), Svc(Supervisor), Fiq(Fast Interrupt request), Und(Undefined) 모드 가 지원
    - 7번째 : System 모드 지원
      - system 모드는 user mode와 같은 레지스터를 이용

[RISC – ARM7 Modes]

- ARM 7 레지스터의 종류, flag 및 예외처리



[RISC - ARM7 상태 레지스터 CPSR]

- **CPSR (Current Program Status Register)**
  - user-level 프로그램의 condition code를 저장하는데 사용
  - Condition code의 각 비트 : 산술 및 논리연산결과를 저장
  - 조건분기명령의 조건을 표시
  -

Flag	이름	의미
N	Negative	ALU operation의 결과가 음수인 경우이다.
Z	Zero	ALU operation의 결과가 '0'인 경우이다.
C	Carry	ALU operation 또는 shift operation의 결과 carry 가 발생하는 경우.
V	Overflow	산술연산의 결과 값이 너무 커서 표현가능범위를 벗어나는 경우

  - Condition code의 하위부분
    - CPU의 동작모드, 명령어 세트의 종류(I:32-bit 명령, T:16-bit 명령), interrupt enable (I:interrupt Request, F: Fast interrupt request)
    - 이 비트들은 user-level 프로그램에 의해 변화하지 않음

[RISC – ARM7 Flag]

- condition code flag (또는 줄여서 flag)
  - N,Z,C,V flag은 CPSR(Current Program Status Register)이라는 32 비트 레지스터의 특정한 4비트를 이용하여 표시한다.
  - 프로그래머에게 도움을 주기 위하여 사용
  - destination register에 저장되어 있는 명령어의 실행 결과 값과 함께 현재 CPU의 상태를 표시하는데 사용
  - 예 :
    - 덧셈 연산에서 발생하는 자리올림을 나타내는 Carry라는 flag
      - 자리올림이 발생했을 경우 C flag은 1이 됨
    - 자리 넘침 또는 oVerflow 상태임을 표시하는 V flag
      - 어떤 산술연산에서 양수와 양수를 더했는데 결과가 음수가 되는 경우
      - 음수에서 음수를 뺄 때 결과 값이 양수가 되는 경우
      - 연산의 결과가 부호 비트를 침범하여 주어진 자리수로 표현이 불가능하게 되어 처리불능 상태에 빠진 것을 나타냄
    - N flag
      - 연산결과가 양수인지 음수인지를 구분하는데 용이
      - N flag이 1이면 음수임을 표시
    - Z flag
      - 어떤 산술연산의 결과 값이 0인 경우에 1이 되는 condition code

[RISC - ARM7 16 Condition Codes]

Opcode[31:28]	Mnemonic	의미	플래그
0000	ES	Equal	Z=1
0001	NE	Not Equal	Z=0
0010	CS/HS	Carry Set/Unsigned Higher or Same	C=1
0011	CC/LO	Carry Clear/Unsigned Lower	C=0
0100	MI	Minus/negative	N=1
0101	PL	Plus/positive or zero	N=0
0110	VS	Overflow	V=1
0111	VC	No overflow	V=0
1000	HI	Unsigned Higher	C=1 & Z=0
1001	LS	Unsigned lower or same	C=0 or Z=1
1010	GE	Signed greater than or equal	N=V
1011	LT	Signed less than	N≠V
1100	GT	Signed greater than	Z=0 & N=V
1101	LE	Signed less than or equal	Z=1 or N≠V
1110	AL	Always	Any
1111	NV	Never (Do not use)	none

[RISC - ARM7 Exception 처리]

- **ARM7 Exceptions**
  - Exceptions : interrupt, trap, 그리고 supervisor call
  - 처리순서
    - 1. 현재의 상태가 저장된다 ([PC] -> r14_exc, CPSR ->SPSR_exc, 여기서 exc는 exception 종류를 의미한다.
    - 2. Processor는 동작모드를 exception모드로 바꾼다.
    - 3. PC는 0x00에서 0x1C값 중에서 exception의 종류에 따라 한가지 값이 선택된다.
  - PC의 새로운 주소(vector address라 불림)의 명령어
    - 대개 exception 처리프로그램으로 branch하는 명령어
    - stack을 가리키는 r13_exc를 사용해서 사용자 레지스터들의 값을 저장
  - Exception 처리가 다 끝난 뒤에 다시 사용자 프로그램으로 되돌아오려면,
    - stack에 저장되었던 사용자 레지스터들과 PC와 CPSR을 복구

[RISC – ARM7 명령어]

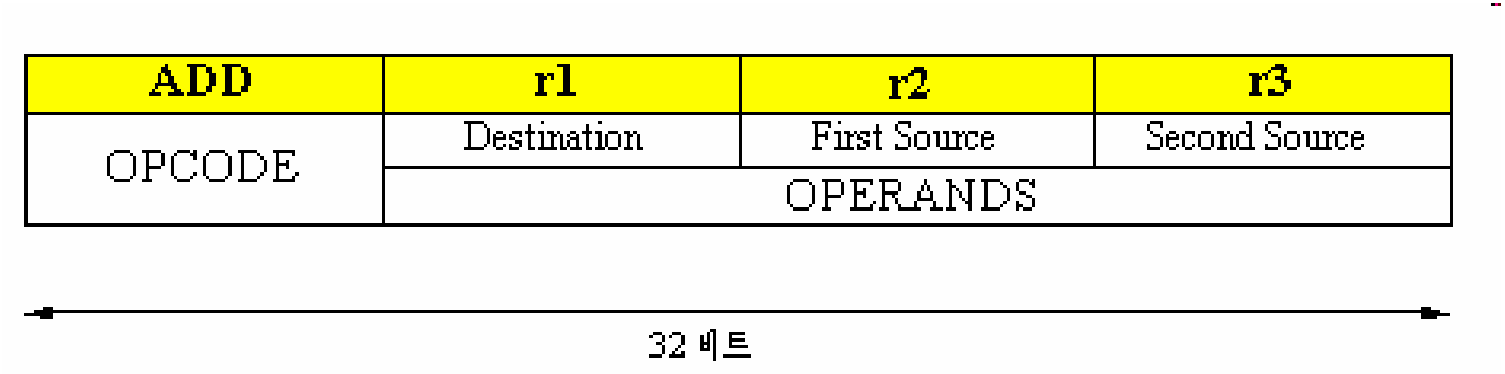
- **명령어 버전: V4**
  - ARM7, ARM9 이 포함됨.
- **ARM 7 명령어 설명 및 어드레싱 모드**
  - 3가지 종류로 분류
  - 데이터 이동 명령어
    - 데이터를 메모리에서 읽거나 쓰는데 사용
  - 데이터 처리 명령어
    - 읽어온 명령어를 처리
  - 분기 명령어
- **데이터 처리 명령어**
  - 데이터에 산술연산 혹은 논리연산을 수행하는 명령
  - 데이터 처리명령을 수행하고 나면 데이터가 변함
    - 데이터 처리명령을 제외한 다른 모든 명령어들은 데이터가 변하지 않음

[RISC – ARM7 데이터 처리 명령어]

- **데이터 처리 명령어**
  - 모든 ARM 명령어 : 32비트
  - 두 가지 중요한 내용
    - 첫 번째는 어떤 명령을 수행하라는 것 : opcode (operation code)로 표현
    - 두 번째는 그 명령수행에 필요한 데이터가 무엇이다 혹은 어떤 장소에 있다는 내용 ; opcode가 필요로 하는 데이터는 operand라고 부름
  - 예 : ADD r1, r2, r3 ;r1 = r2 + r3
    - Opcode : 'add'
    - Operand : r1, r2, r3
    - r : 레지스터(Register)를 의미
    - 세미콜론(;) : 그 이후의 내용이 주석 혹은 코멘트임을 표시
      - 세미콜론 이후부터 그 줄의 끝까지의 내용을 무시

[RISC – ARM7 데이터 처리 명령어 (2)]

- **데이터 처리 명령어**
  - 두 개의 입력 오퍼랜드를 이용해서 한 개의 출력 오퍼랜드를 만들어 냄
    - 일반적으로 ARM에서 모든 오퍼랜드들은 32비트의 크기를 가짐
  - 입력 오퍼랜드 r2와 r3 : source operand
    - r2 : first source operand
    - r3 : second source operand



[RISC - ARM7 데이터 처리 명령어 (3)]

- **데이터 처리 명령어**
  - 레지스터 오퍼랜드 명령어
    - ADD r1, r2, r3 ;r1 = r2 + r3
    - 레지스터 r2의 내용과 r3의 내용을 합하여 그 결과를 r1에 저장
    - 오퍼랜드를 표시하는 순서가 역방향
      - destination register r1이 먼저 표기되고 그 뒤에 first soruce operand r2 와 second source operand r3의 순서
    - 레지스터에 들어있는 데이터가 부호가 없이 양수만 고려한 경우
      - 연산 결과는 carry만 고려
    - 데이터가 부호 비트를 가지는 2의 보수로 간주되는 경우
      - 연산결과는 carry는 물론 자리 넘침(overflow) 까지도 함께 고려해야 함
      - 자리 넘침
        - 양수와 양수를 더했는데 연산결과가 표현 가능한 범위를 넘쳐서 부호 비트로까지 넘쳐서 음수로 표기되는 경우
        - 음수에서 음수를 뺐는데 결과가 양수인 경우도 자리 넘침이 발생한 상태를 의미

[RISC - ARM7 데이터 처리 명령어 (4)]

- 데이터 처리 명령어
  - 레지스터 오퍼랜드 명령어
    - 수행하는 내용에 따라, 즉 opcode의 종류에 따라서 다음의 네 가지로 분류
- 산술 명령
  - 2진 덧셈, 뺄셈
  - 오퍼랜드의 크기는 32비트
  - 부호없이 크기만 사용할 수도 있고 또는 2의 보수를 이용해서 부호있는 숫자로도 사용할 수 있음
    - ADD r1, r2, r3 ;r1 = r2+r3 --ADD
    - ADC r1, r2, r3 ;r1 = r2+r3+C --ADD w/ carry
    - SUB r1, r2, r3 ;r1 = r2-r3 --SUB
    - SBC r1, r2, r3 ;r1 = r2-r3+C-1 --SUB w/ carry
    - RSB r1, r2, r3 ;r1 = r3-r2 --reverse SUB
    - RSC r1, r2, r3 ;r1 = r3-r2+C-1 --rev. SUB w/carry
    - MUL r1, r2, r3 ;r1 = r3×r2 --32bit multiply
  - ADC 명령 : 64비트 덧셈
  - SBC 명령 : 64비트 뺄셈
  - MUL 명령어 : 32비트 두 숫자를 곱해서 32비트의 결과를 저장하는 명령어
    - 결과 값이 32비트보다 큰 경우 : ARM의 다른 곱셈 명령어를 사용

[RISC – ARM7 데이터 처리 명령어 (5)]

- **데이터 처리 명령어**
  - 레지스터 오퍼랜드 명령어
    - 수행하는 내용에 따라, 즉 opcode의 종류에 따라서 다음의 네 가지로 분류
- **산술 명령**
  - 자리올림이 발생
    - 덧셈의 경우 : C flag으로 그 사실을 반영
    - 뺄셈연산의 경우 : 빌려오는 borrow가 발생
      - 뺄셈은 subtraction by addition 즉, second source operand의 2의 보수를 만들어서 더하게 된다.
      - Second operand가 1처럼 작은 숫자인 경우에는 2의 보수는 0xFFFFFFFF처럼 큰 숫자가 된다. 즉 1과 같은 작은 수를 뺄 때는 borrow가 발생하지 않아야 하지만 (즉, borrow=0 이어야 하지만) 실제로는 2의 보수가 더해지므로 C flag이 1이 된다.
      - 반대로 작은 숫자에서 더 큰 숫자를 뺄 때는 borrow=1 이지만, 큰 수의 2의 보수는 더 작은 숫자가 되므로 C flag이 0이 된다.
      - 따라서 SUB 명령의 borrow 조건은 C flag으로도 표현이 가능하며 이는 NOT(C flag)으로 나타낼 수 있다.

[RISC - ARM7 어드레싱 모드]

- 데이터 처리 명령의 11가지 어드레싱 모드
- 어드레싱 모드의 의미?

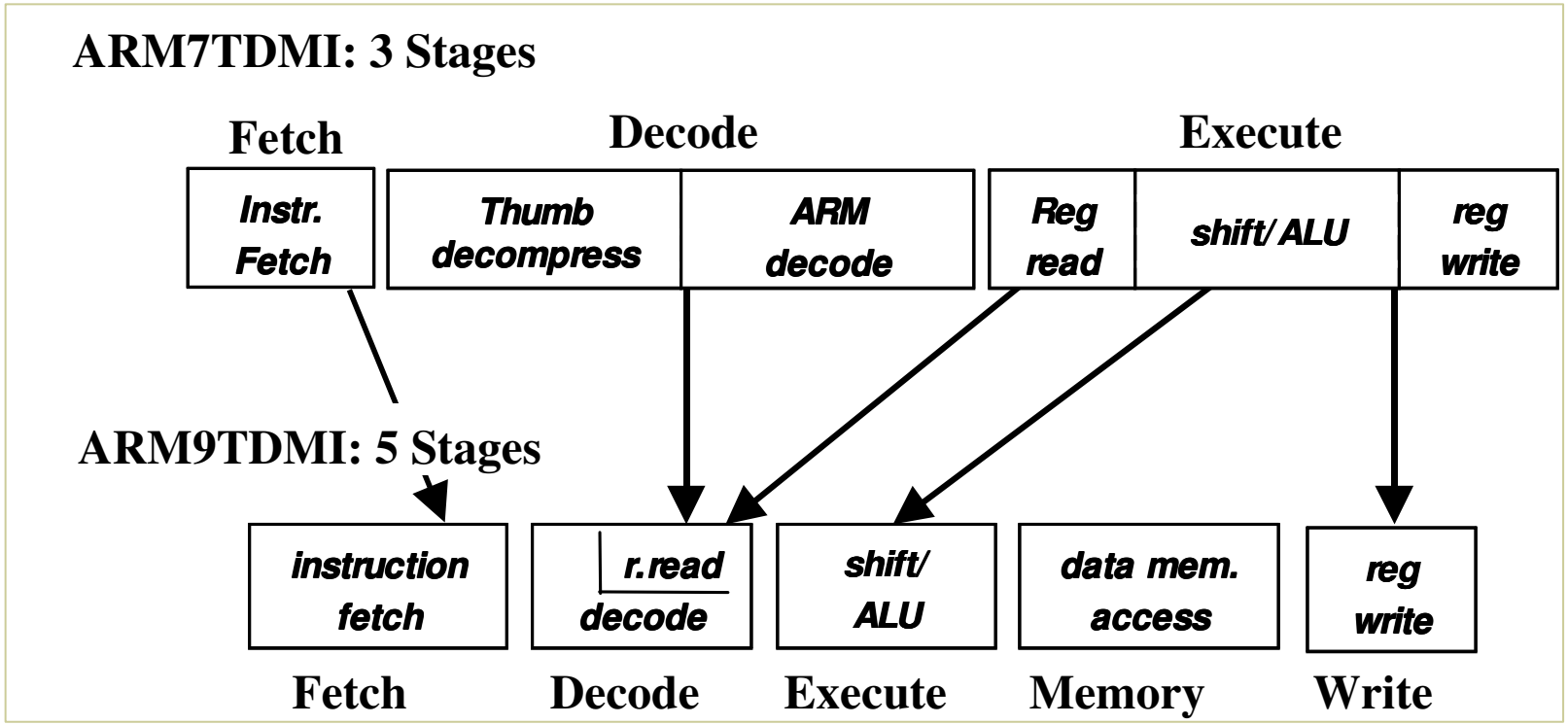
주소지정방식		명령어 문법	명령어 예
1	Register	ADD Rd,Rn,Rm	ADD r1,r2,r3
2	Immediate	ADD Rd,Rn,#<immed8>	ADD r1,r2,#5
3	Register LSL	ADD Rd,Rn,Rm, LSL Rs	ADD r1,r2,r3,LSL r4
4	Immediate LSL	ADD Rd,Rn,LSL #<shift_imm5>	ADD r1,r2,r3,LSL #3
5	Register LSR	ADD Rd,Rn,Rm, LSR Rs	ADD r1,r2,r3,LSR r4
6	Immediate LSR	ADD Rd,Rn,LSR #<shift_imm5>	ADD r1,r2,r3,LSR #3
7	Register ASR	ADD Rd,Rn,Rm, ASR Rs	ADD r1,r2,r3,ASR r4
8	Immediate ASR	ADD Rd,Rn,LSR #<shift_imm5>	ADD r1,r2,r3,ASR #3
9	Register ROR	ADD Rd,Rn,Rm, ROR Rs	ADD r1,r2,r3,ROR r4
10	Immediate ROR	ADD Rd,Rn,ROR #<shift_imm5>	ADD r1,r2,r3,ROR #3
11	Extended ROR	ADD Rd,Rn,Rm, RRX	ADD r1,r2,r3,RRX

[RISC - ARM9 개요]

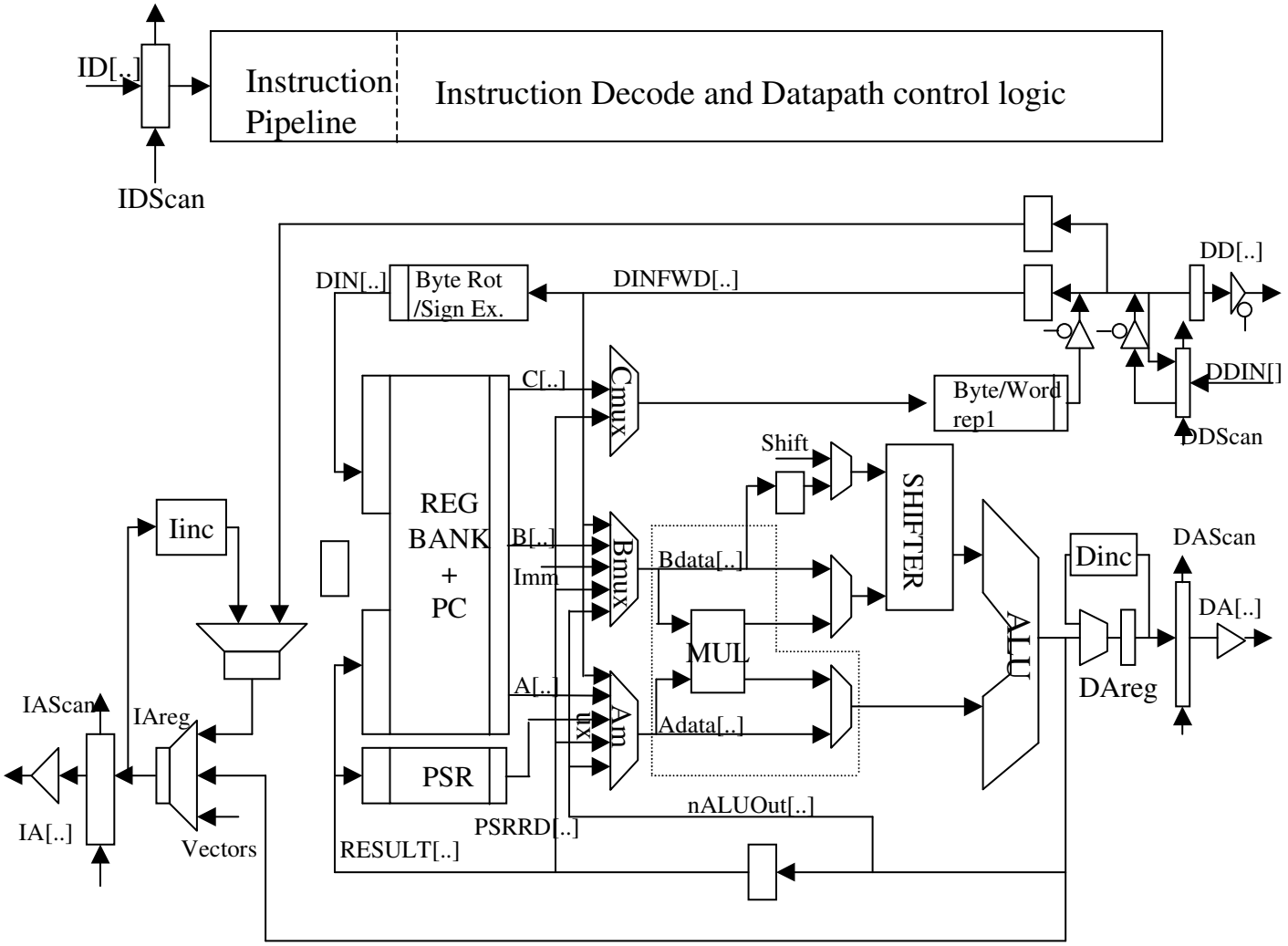
- ARM7과 같은 V4 architecture 채용
  - Share the same instruction set
- 종류
  - ARM9TDMI
    - 프로세서 코어,
  - ARM920T
    - 프로세서 코어 + MMU for Platform OS
    - 16KB code cache & 16KB data cache
    - 각 cache별 TLB 사용
  - ARM940T
    - 프로세서 코어 + MPU for Real-time OS
    - 4KB code cache & 4KB data cache

[RISC - ARM9 vs. ARM7]

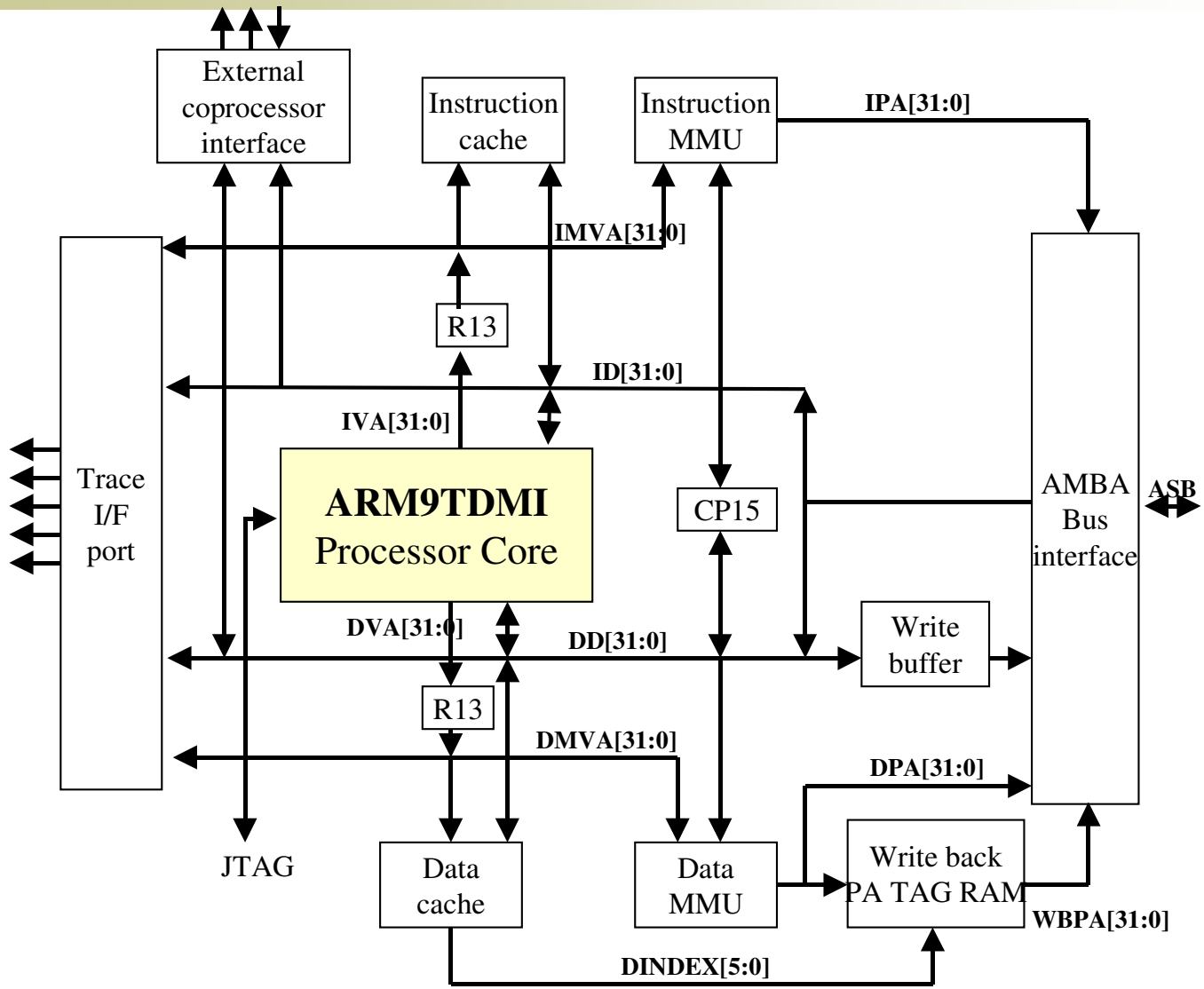
- **Architecture Improvements**
  - ARM7: 3-stage => ARM9: 5-stage pipeline architecture
  - ARM7: unified cache=>ARM9: separate cache (Harvard arch.)



[RISC - ARM9 : 5-stage Pipeline Arch.]



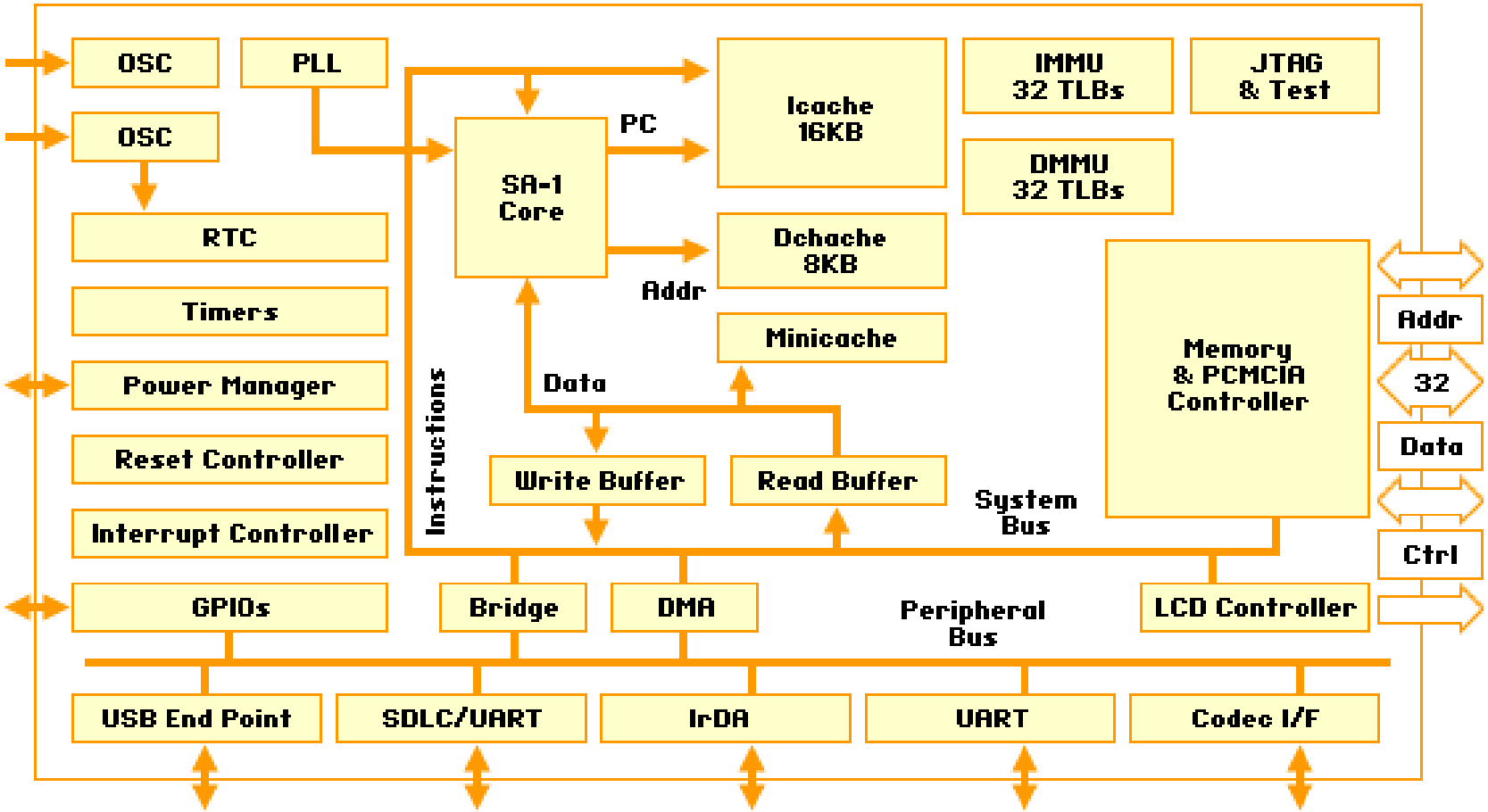
[RISC – ARM920T 구조 블록도]



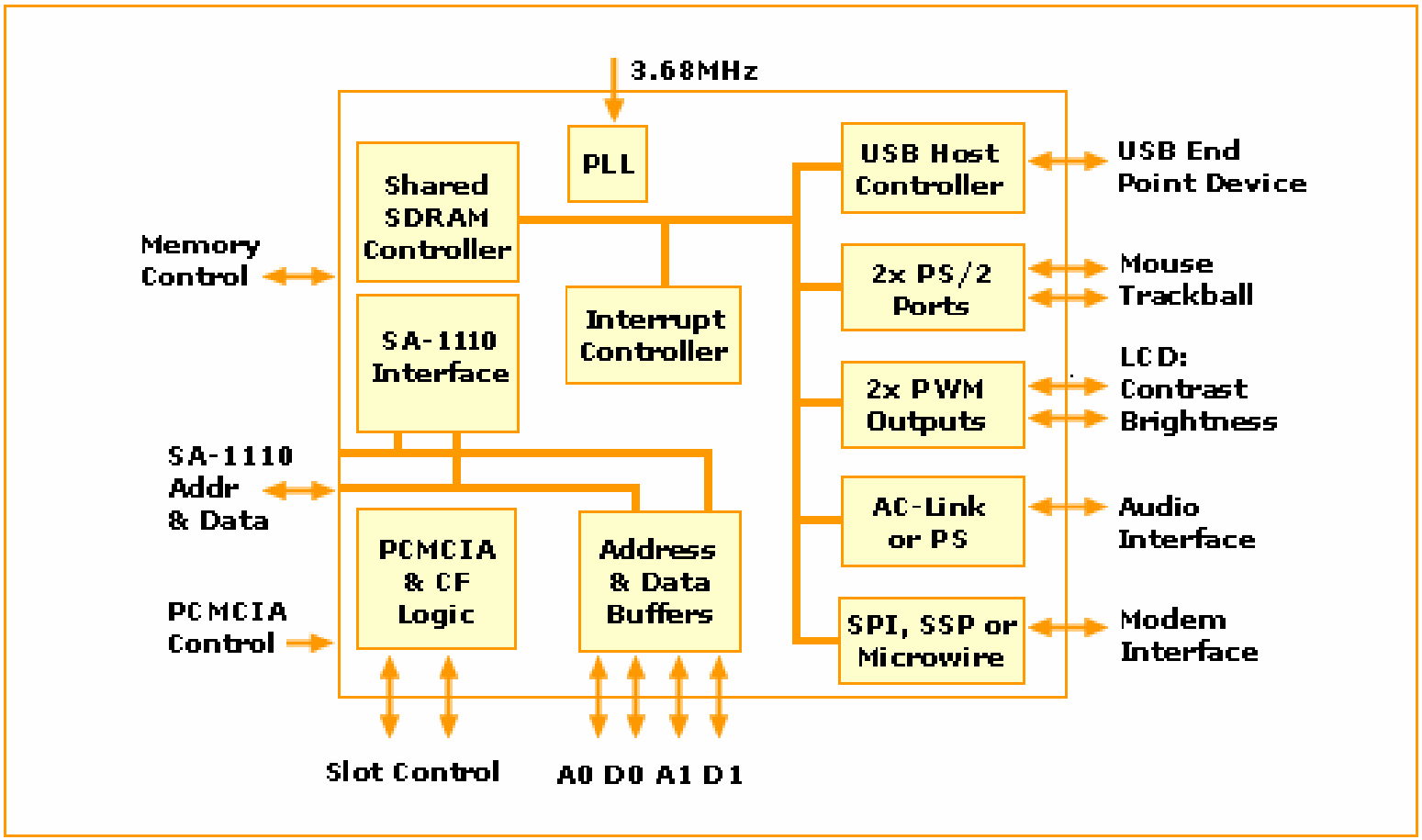
[RISC - StrongARM]

- **ARM v4 프로세서 코어 + PDA용 주변장치**
- **특징**
  - **Forwarding** 가능한 **5**단계 파이프라인
  - **16K** 바이트 **32-way Set Associative** 명령어 캐쉬
  - **8K** 바이트 **32-way Copy-Back** 데이터 캐쉬
  - 명령어와 데이터를 위한 분리된 **32-entry TLB**
  - 데이터 캐쉬 **Thrashing** 방지를 위한 **2-way Set Associative** 미니캐쉬
  - **3**단계 전력 관리 기능
    - 정상 **(Normal)** 모드, 대기 **(Idle)** 모드, 슬립**(Sleep)** 모드

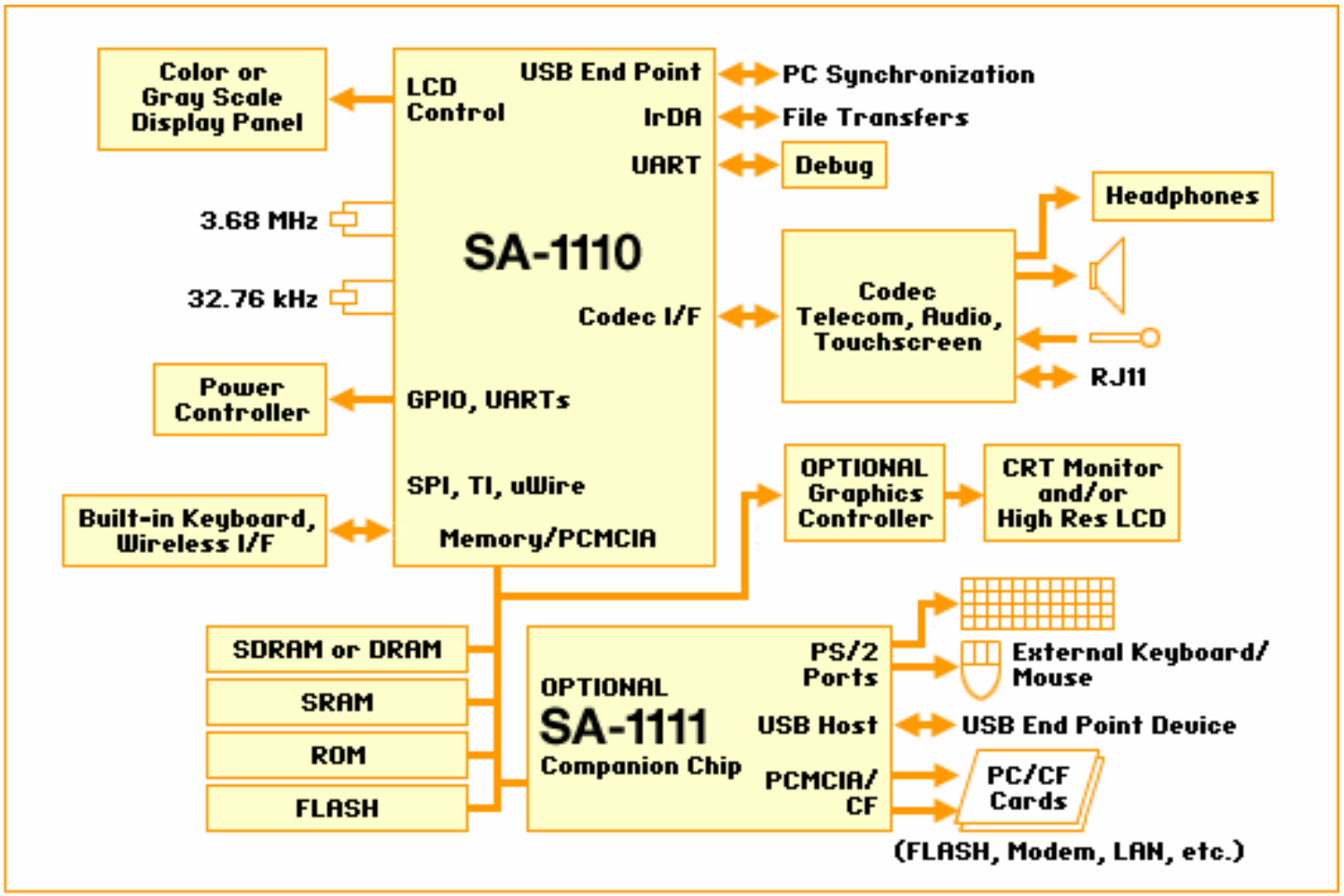
[RISC – SA1110 블록도]



[RISC – SA1111 블록도]



[RISC – SA1110/SA1111 시스템 블록도]



RISC – Xscale Microarchitecture Features

- Arm Architecture Version 5TE ISA 호환
- 저전력 & 고성능(최대 400MHz)
- Modified Harvard Architecture
  - instruction cache와 data cache가 분리(2 caches)
  - 32KB Instruction Cache
  - 32KB Data Cache
- Intel Media Processing Technology
- Instruction and Data Memory Management Unit
- Branch Target Buffer
- Debug Capability via JTAG Port
- 0.35μm 3 Layer metal CMOS, 2.6 million transistor
- 256 PBGA package (17 x 17mm)

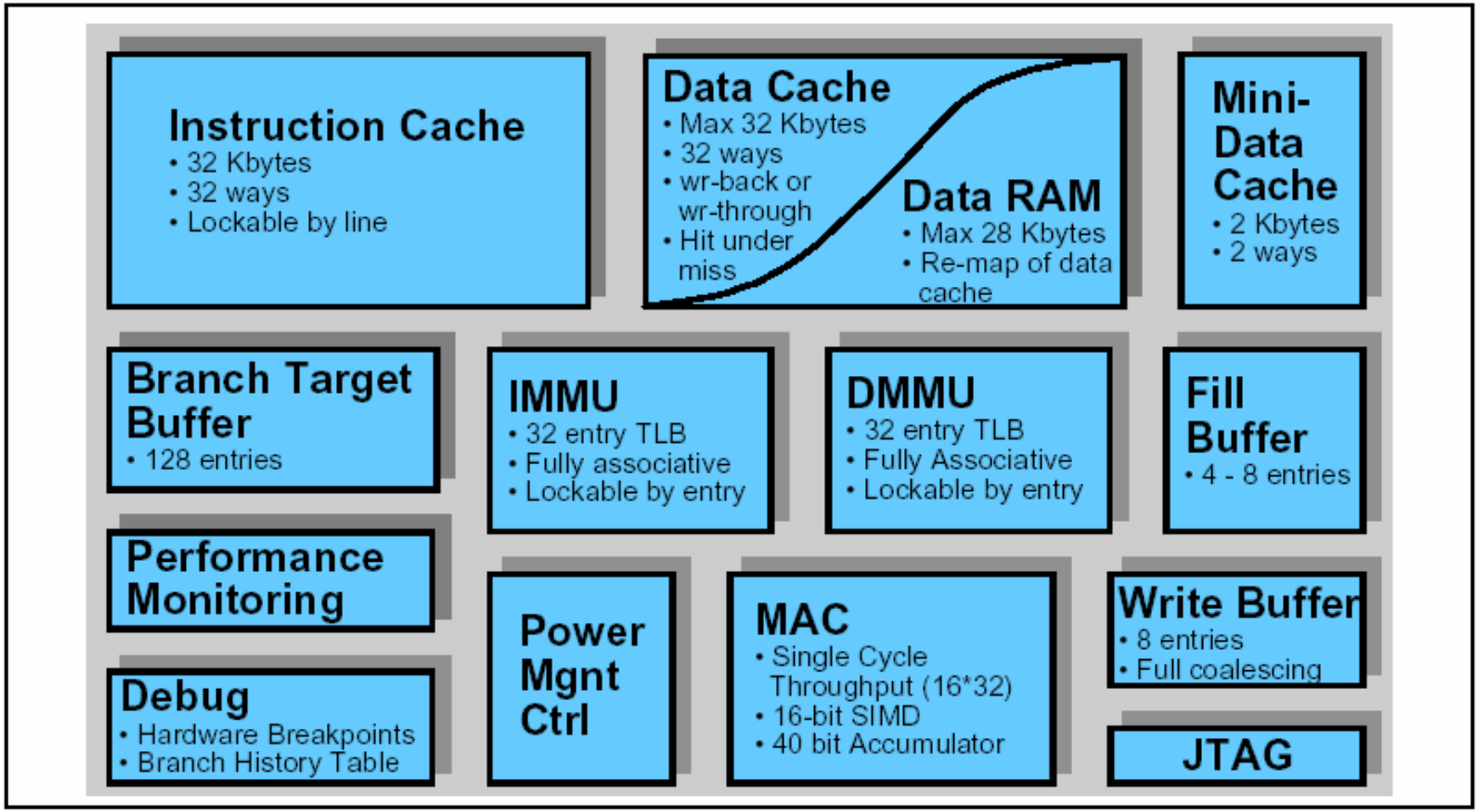
RISC – Xscale System Integration Features

- Memory controller
- Power management controller
  - Normal, idle, sleep mode 지원
- USB client
- Multi channel DMA controller
  - 소프트웨어 프로그램 가능, 외부 DMA 지원
- LCD controller
- AC97 codec
- I2S
- Multimedia card: serial interface to standard memory card, FIFO 포함
- FIR communication: 적외선 통신 포트
- Synchronous serial protocol port
- I2C

RISC – Xscale System Integration Features

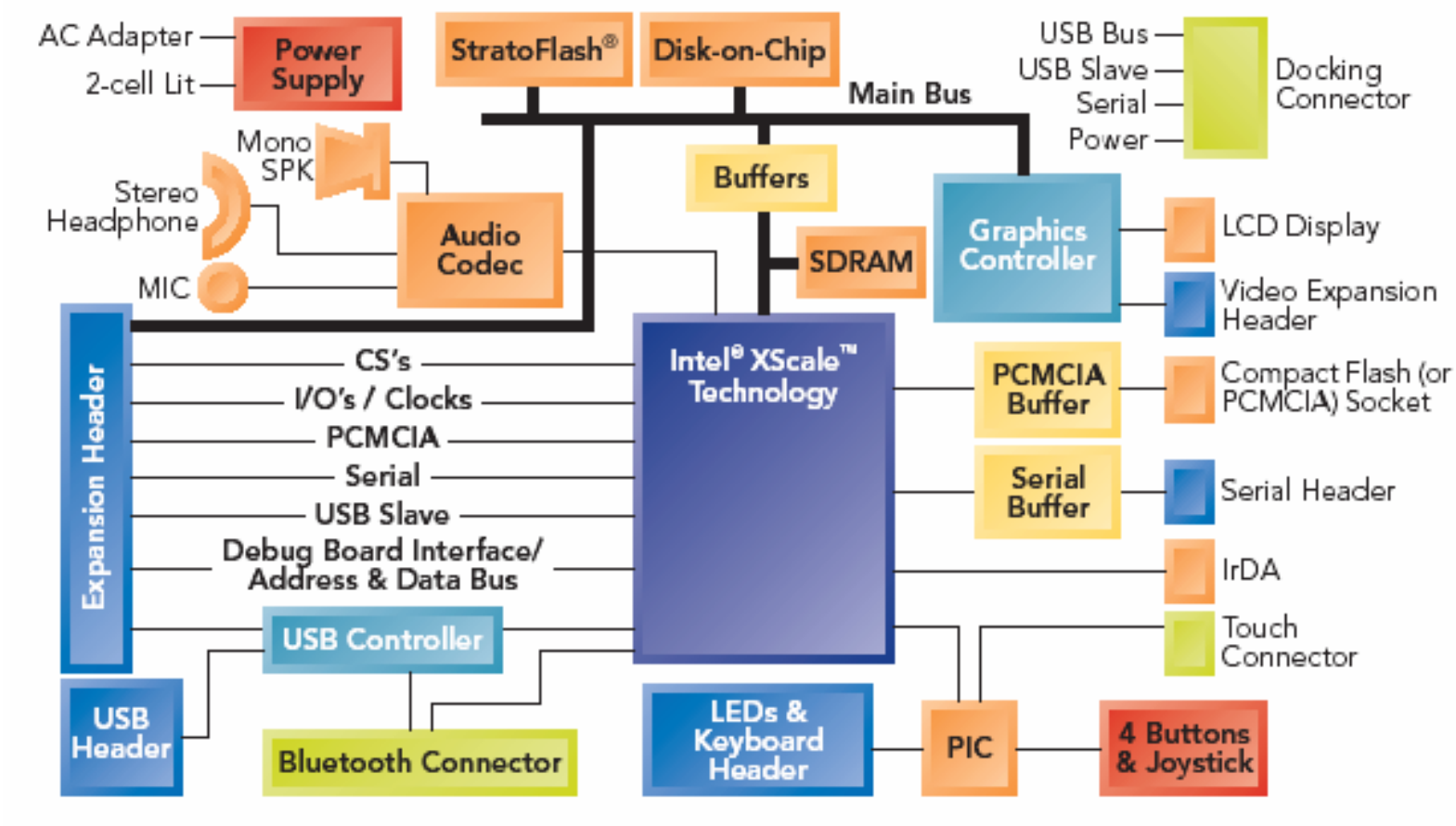
- 85 GPIO ports
  - irq, “wake up” interrupt 생성
- UART
- Real-time clock and timer
  - 32비트 카운터, 32.7kHz 크리스탈, 정밀도 +/- 5sec/mon
- OS timer with alarm register
- Pulse width modulation
- Interrupt controller
  - 모든 시스템 인터럽트를 라우팅

[RISC – XScale 블록도]



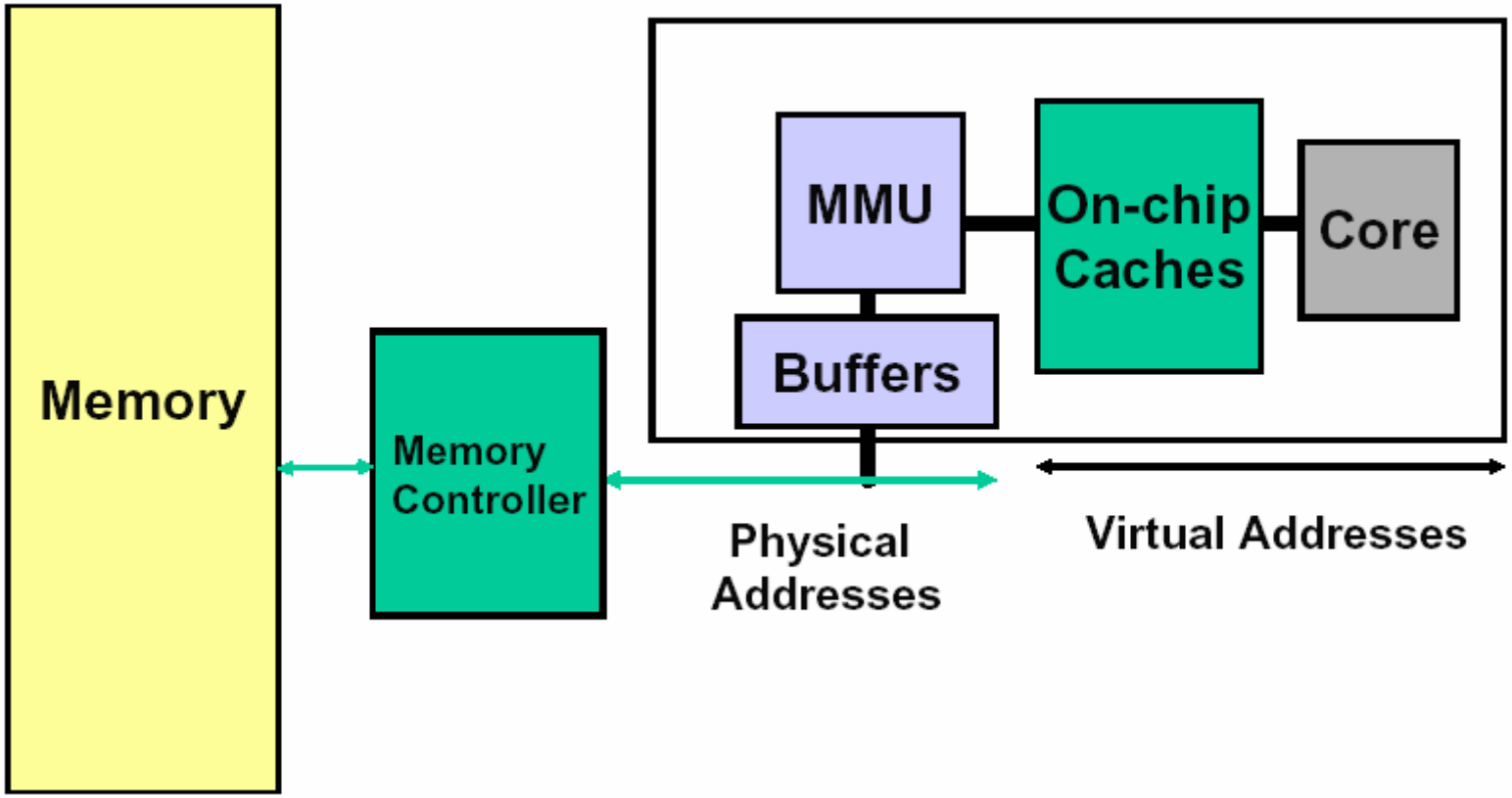
[RISC – Xscale 예제]

- **Palm size device – Example**



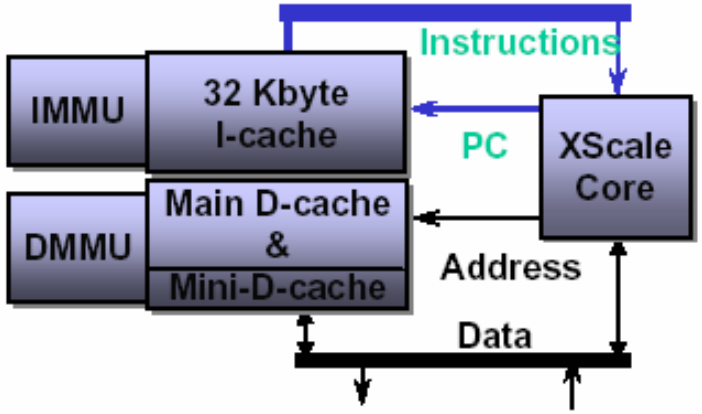
[RISC – Xscale memory model]

- **Memory Model**



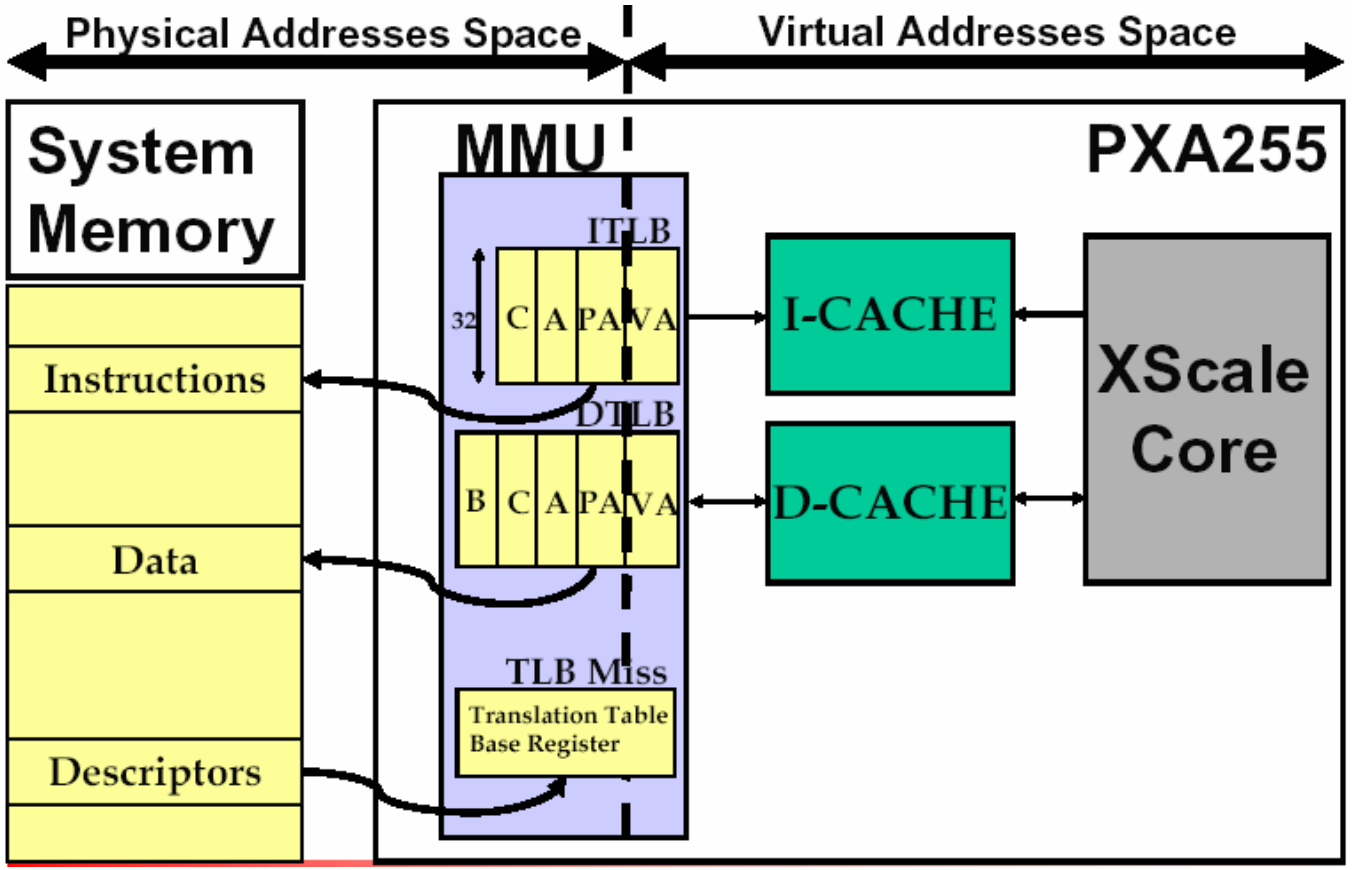
[RISC – Xscale caches]

- **Instruction Cache**
  - 32bit instruction cache
  - 1024 line of 32bytes(8words)
  - 가상 주소 사용
  - 32way 32set associative cache
  - 라운드 로빈 대체 기법
  - MMU가 enable일 경우
    - memory management table에 있는 C 비트에 의해 제어
    - 가상 주소에 해당하는 외부 메모리로부터 single word를 읽어오고 cache에는 쓰여지지 않는다.
  - MMU가 disable일 경우
    - 모든 주소에 대해서 C=1이 된다
    - 8word의 linefetch가 수행돼서 라운드 로빈 대체 기법에 의해 cache bank가 대체된다.

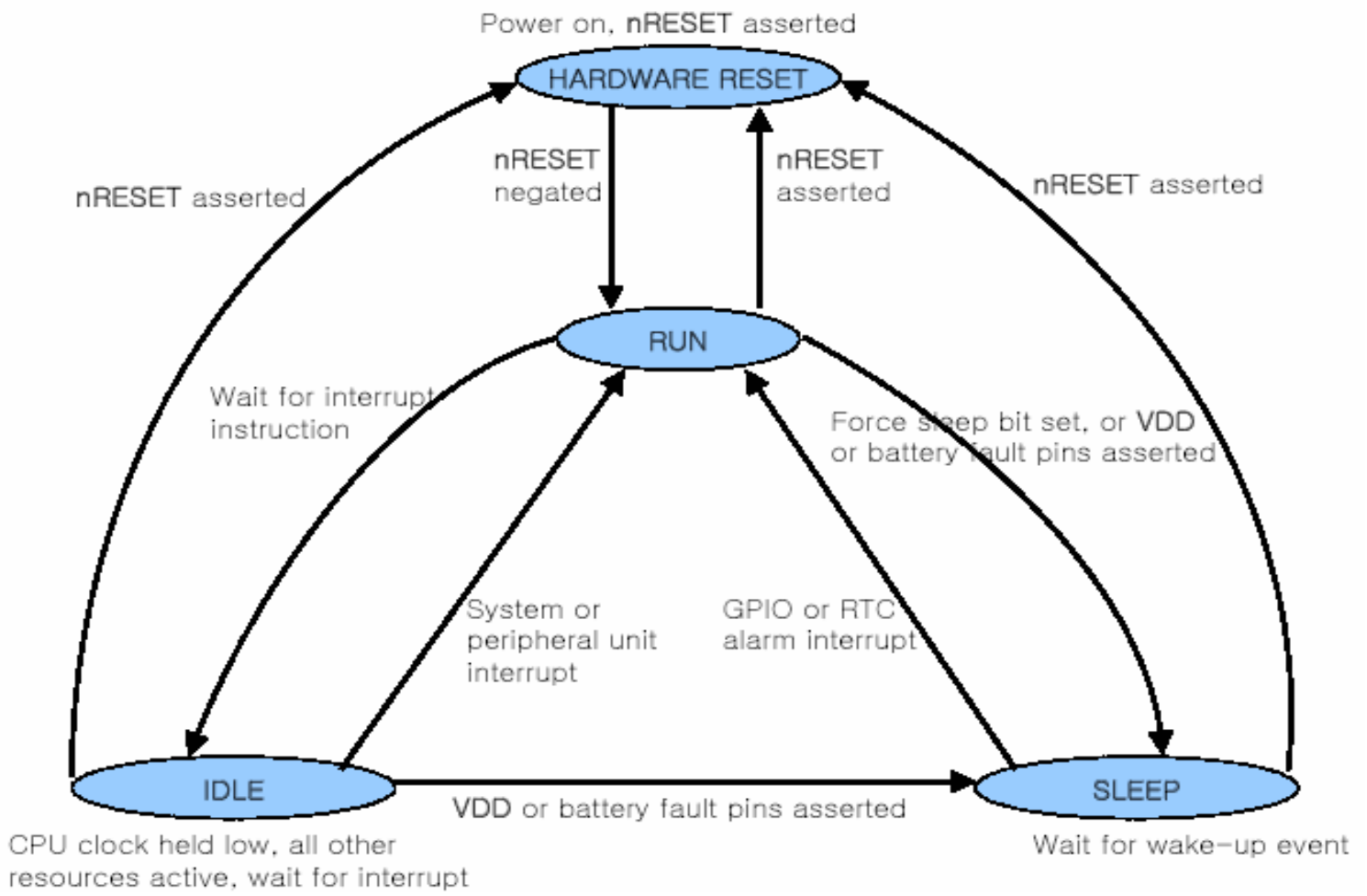


[RISC – Xscale MM]

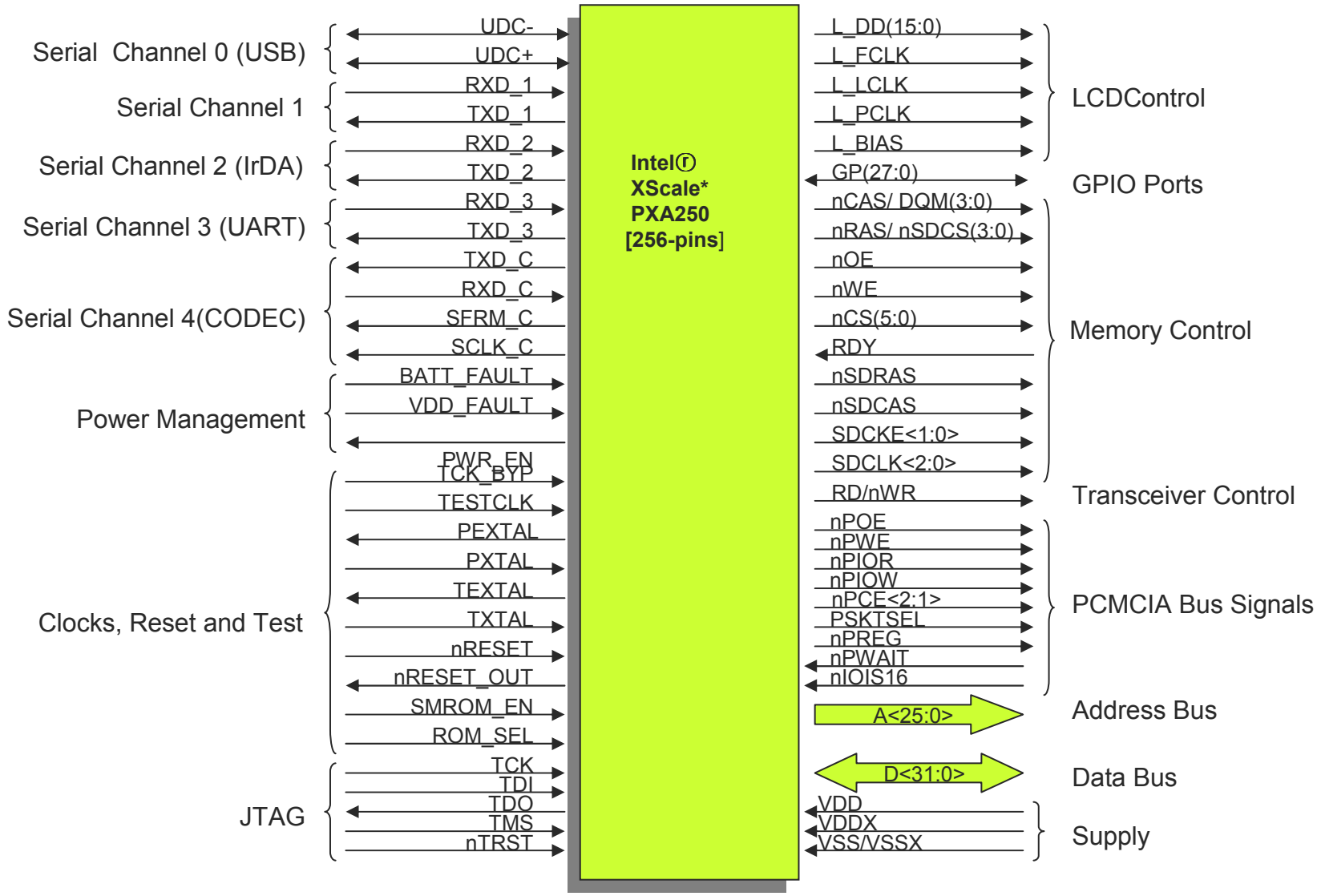
- **Memory Management**



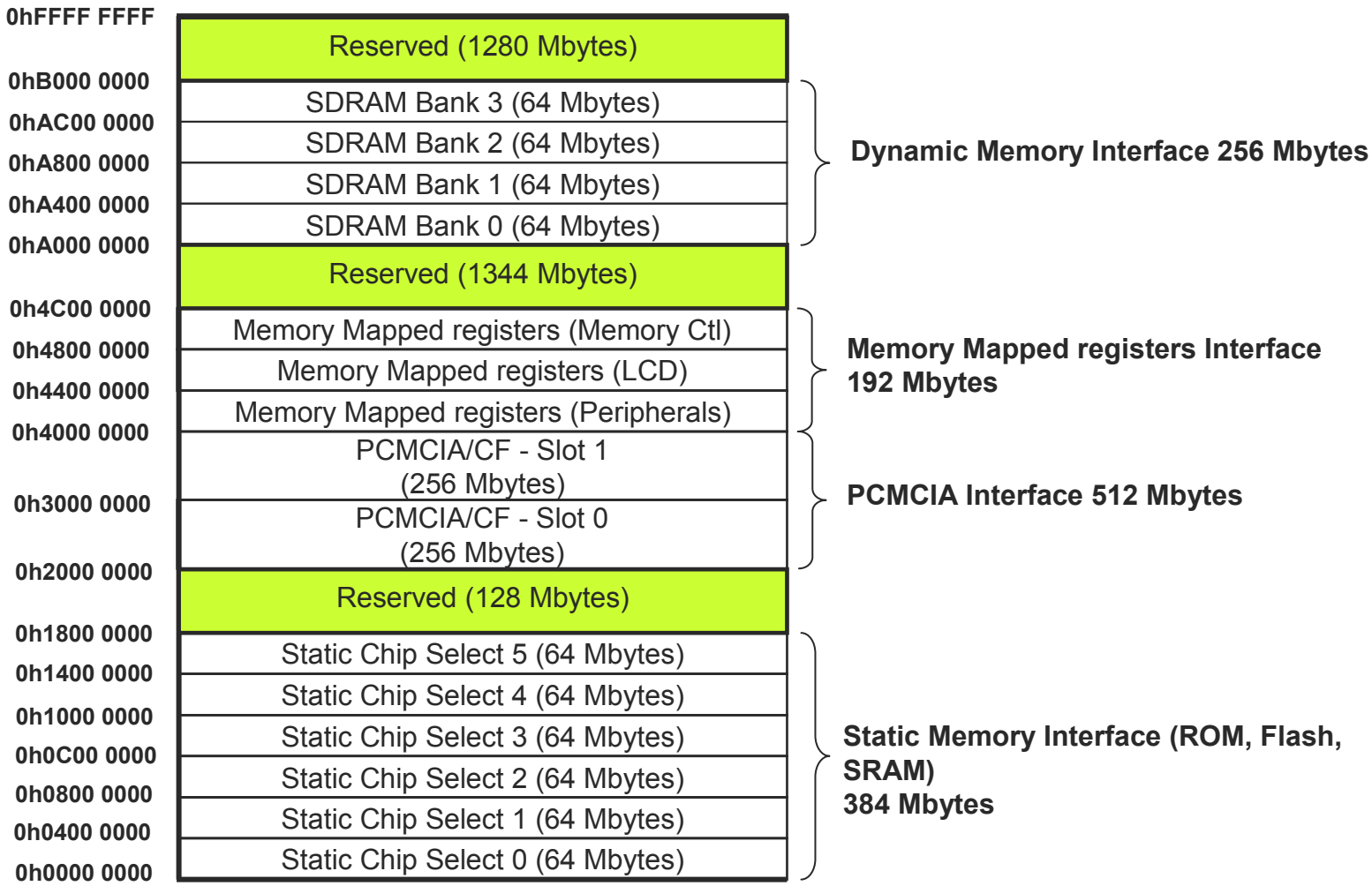
[RISC – Xscale running modes]



[PXA255 Pin]

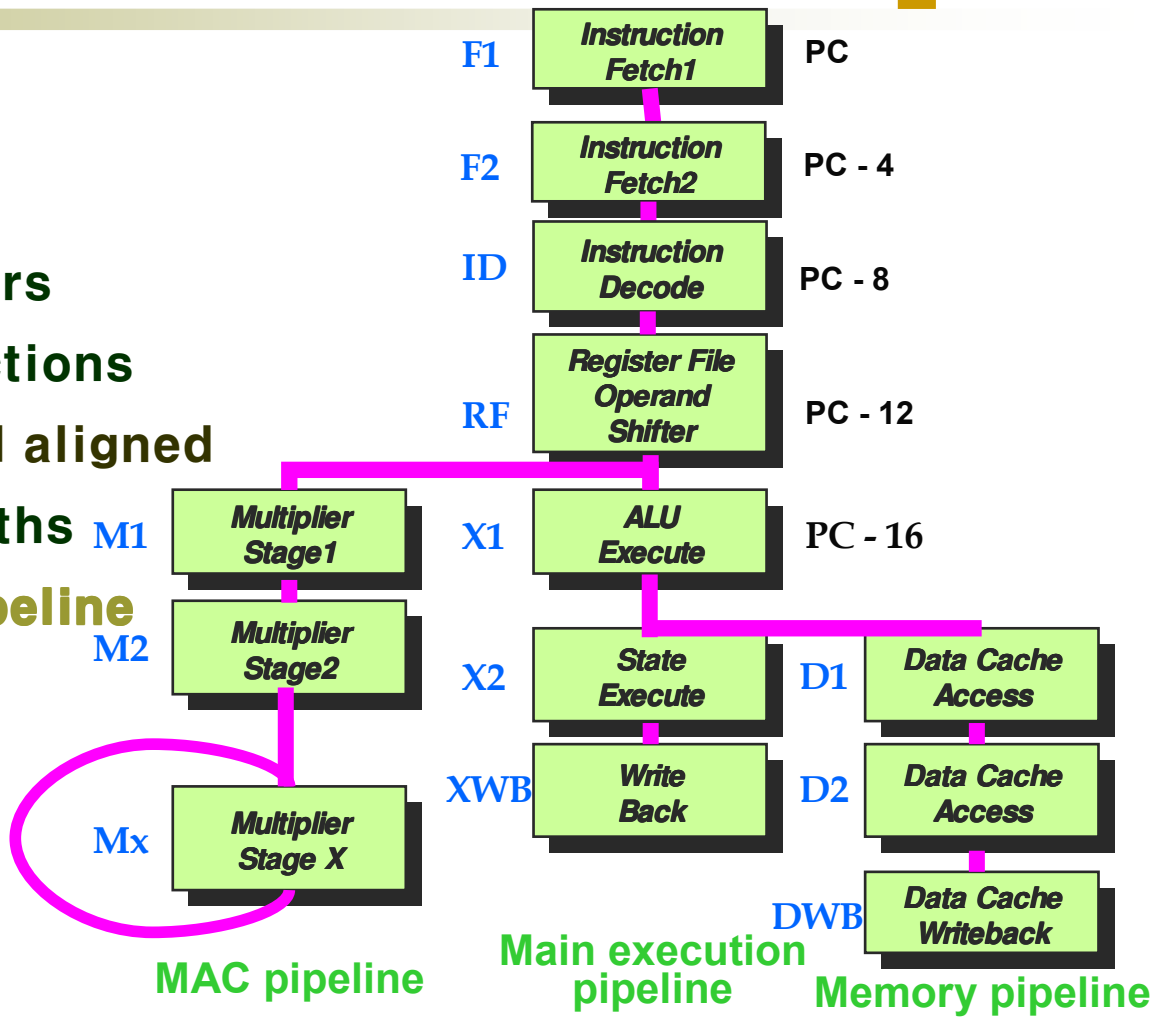

Serial Channel 0 (USB)
UDC-
UDC+
Serial Channel 1
RXD_1
TXD_1
Serial Channel 2 (IrDA)
RXD_2
TXD_2
Serial Channel 3 (UART)
RXD_3
TXD_3
Serial Channel 4(CODEC)
TXD_C
RXD_C
SFRM_C
SCLK_C
Power Management
BATT_FAULT
VDD_FAULT
PWR_EN
Clocks, Reset and Test
TCK_BYP
TESTCLK
PEXTAL
PXTAL
TEXTAL
TXTAL
nRESET
nRESET_OUT
SMROM_EN
ROM_SEL
JTAG
TCK
TDI
TDO
TMS
nTRST
Intel® XScale* PXA250 [256-pins]
L_DD(15:0)
L_FCLK
L_LCLK
L_PCLK
L_BIAS
LCDControl
GP(27:0)
GPIO Ports
nCAS/ DQM(3:0)
nRAS/ nSDCS(3:0)
nOE
nWE
nCS(5:0)
RDY
nSDRAS
nSDCAS
SDCKE<1:0>
SDCLK<2:0>
Memory Control
RD/nWR
Transceiver Control
nPOE
nPWE
nPIOR
nPIOW
nPCE<2:1>
PSKTSEL
nPREG
nPWAIT
nIOIS16
PCMCIA Bus Signals
A<25:0>
Address Bus
D<31:0>
Data Bus
VDD
VDDX
VSS/VSSX
Supply

[PXA255 Address Map]



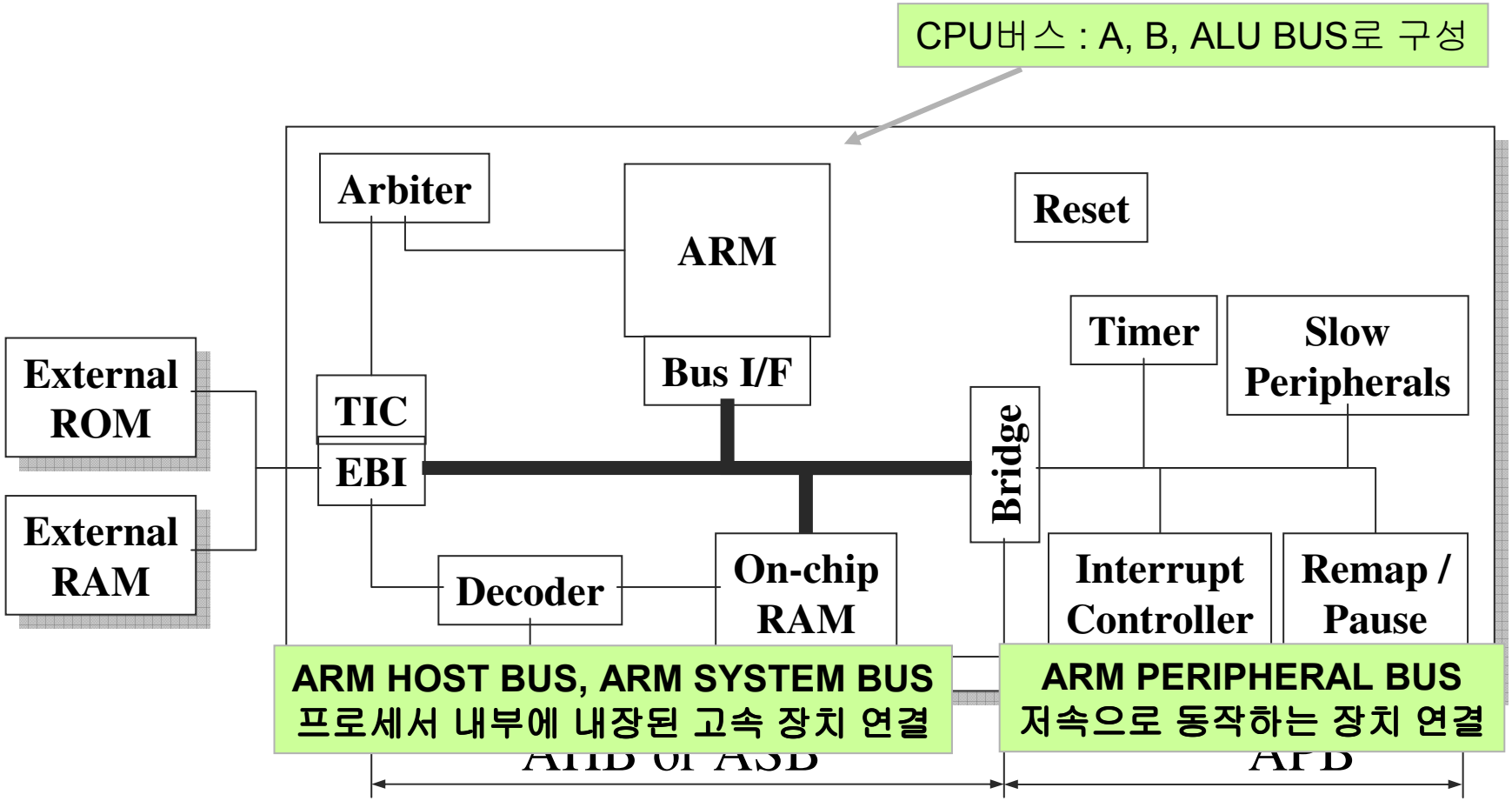
[PXA255 Processor]

- XScale Core
  - 32Bit RISC
  - 32Bit registers
  - 32Bit instructions
    - Longword aligned
  - 32Bit datapaths
  - 7~8 stage pipeline

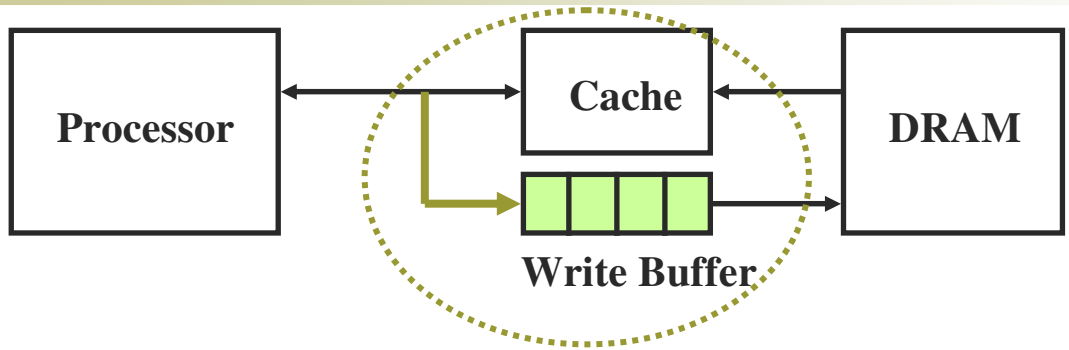


[PXA255 Processor(2)]

- **Advanced Microcontroller Bus Architecture**

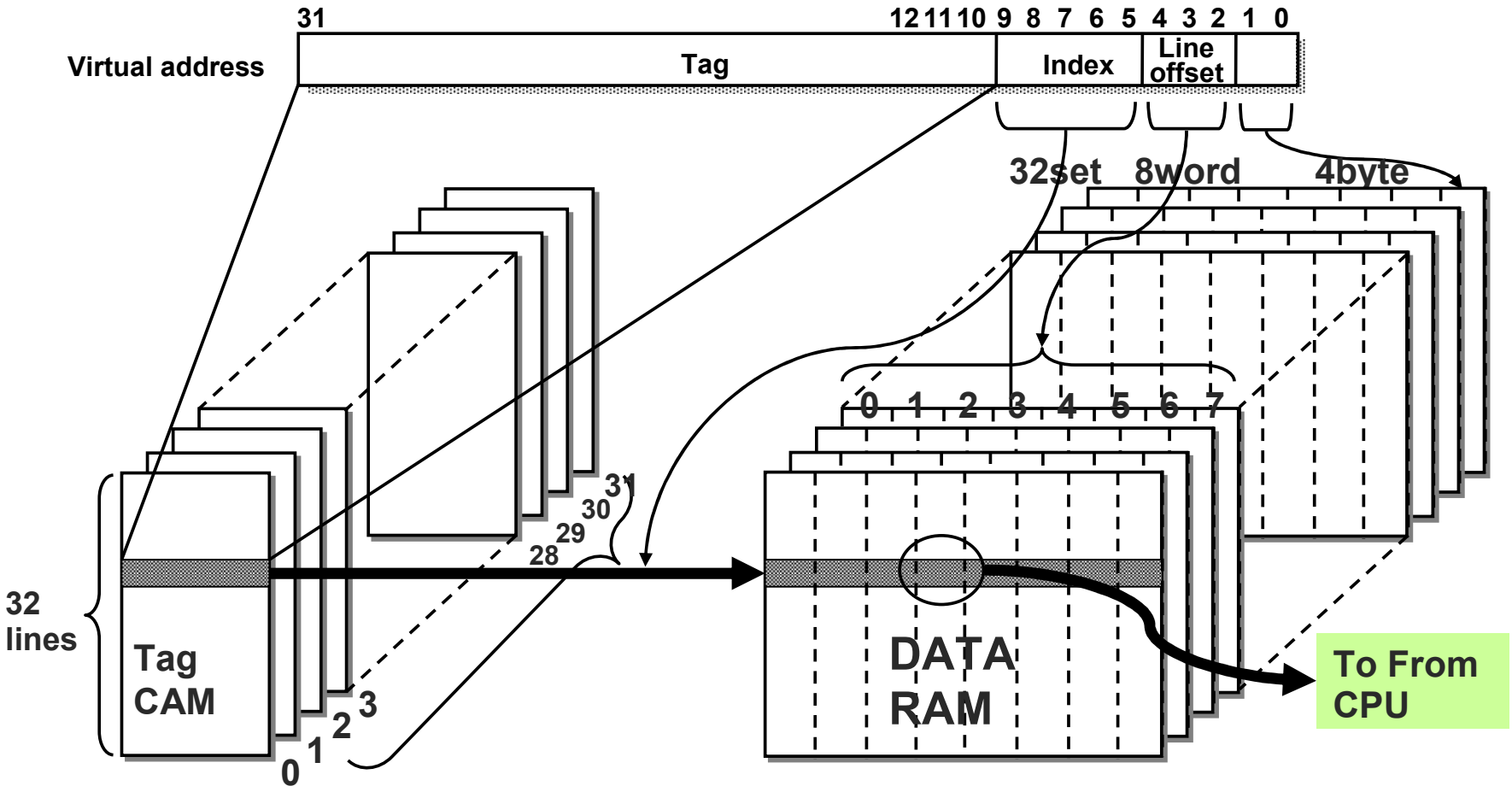


Write Buffer for Write Through



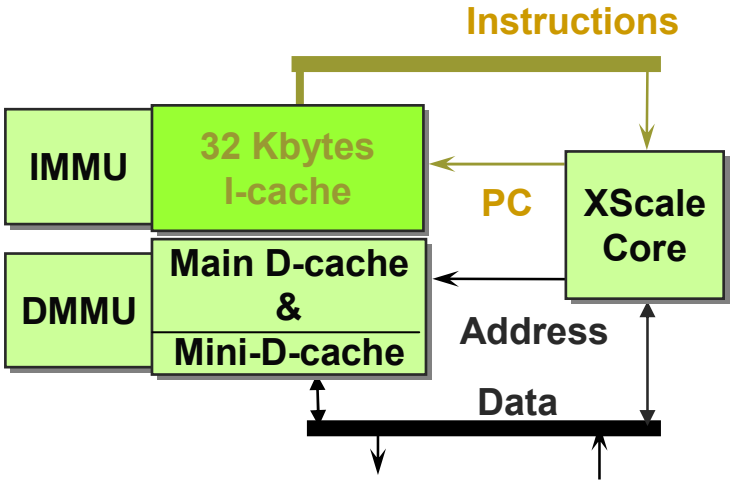
- A Write Buffer is needed between the Cache and Memory
  - Processor: writes data into the cache and the write buffer
  - Memory controller: write contents of the buffer to memory
- Write buffer is just a FIFO:
- Write Buffer는 쓰는 경우의 성능 향상을 위해 존재
  - 캐시 메모리는 명령어, 데이터를 읽을 경우의 성능 향상
- CPU가 쓰기 동작을 하는 동안에도 다른 처리를 계속할 수 있도록, 주소와 데이터가 write buffer에 저장
- 버스의 사용권한이 write buffer에 주어지면 외부장치에 write

Cache organization



[PXA255 - 명령어 캐시]

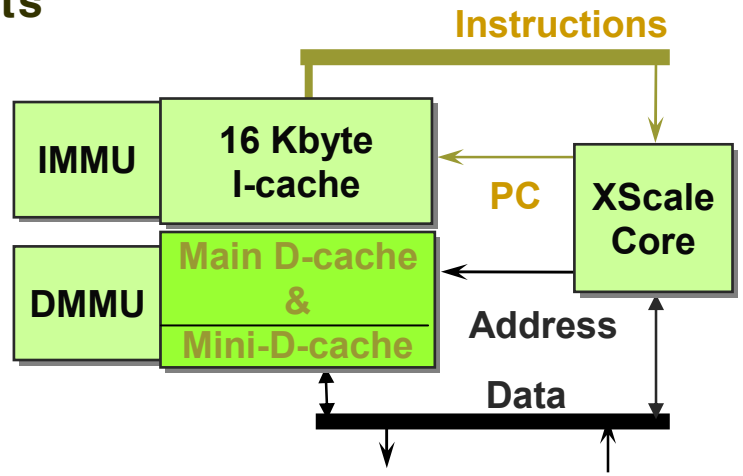
- 명령어 캐시(Instruction Cache)
  - 32KB Instruction Cache
    - 1024 lines of 32bytes(8words)
    - Uses the virtual address
    - 32-way 32-set associative
    - Round-Robin replacement
    - Mapped via MMU page C bits



  - MMU가 enable 되었을 경우에는 memory management table에 있는 C비트에 의해서 제어된다.
  - MMU가 disable 되었을 경우에는 모든 어드레스에 대하여 C=1인 된다.
  - C=1 또는 MMU가 disable 상태인 경우 miss인 경우
    - 8word의 line fetch가 수행이 되어 Round-robin replacement에 의해서 Cache bank가 대치된다.
  - MMU가 enable되고 C=0인 경우에는 virtual address에 해당하는 외부 메모리로부터 single word를 읽어오고, cache에 쓰여지지 않는다.

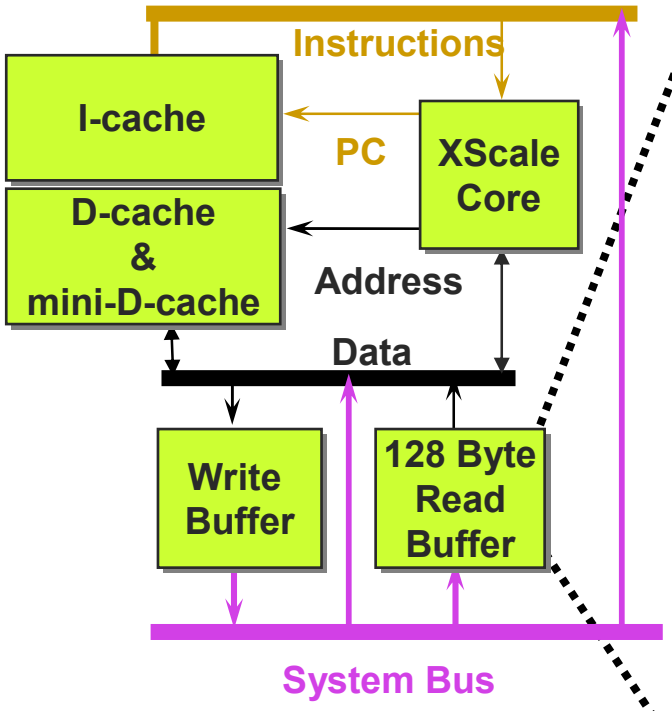
[PXA255 - 데이터 캐시]

- 데이터 캐시(Data Caches)
  - Two Data Caches(Main Data Cache, Mini Data Cache)
    - Both: writeback, read allocate, virtual
    - Mapped via MMU page B, C bits
  - Main Data Cache, 32KB
    - 32-way 32-set associative
    - Round-Robin replacement
    - B=1 & C=1
  - Mini Data Cache, 2KB
    - 2-way set associative
    - Least Recently Used(LRU) replacement
    - B=0 & C=1



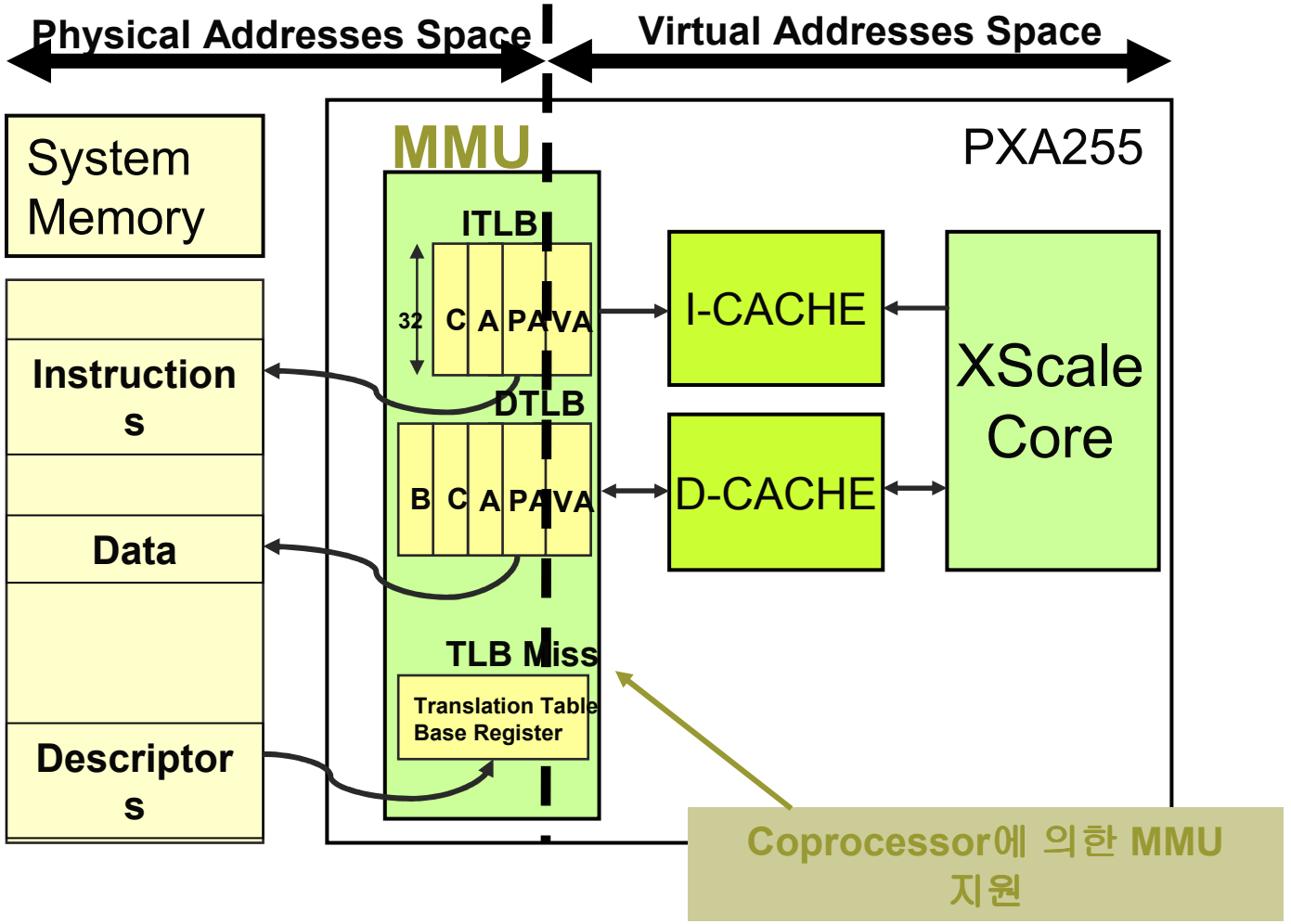
[PXA255 – Read Buffer]

- **PXA255 Read Buffer**



- **Data prefetcher**
  - saves processor waiting
  - load & calculate in parallel
  - for *Read-Only* data
  - supplements the data cache
- **Under software control**
  - Coprocessor 15, register #9
  - 4 entries, 32 bytes each
  - Loads of 1, 4, 8 words
  - Replace or invalidate data

PXA255 Memory Management



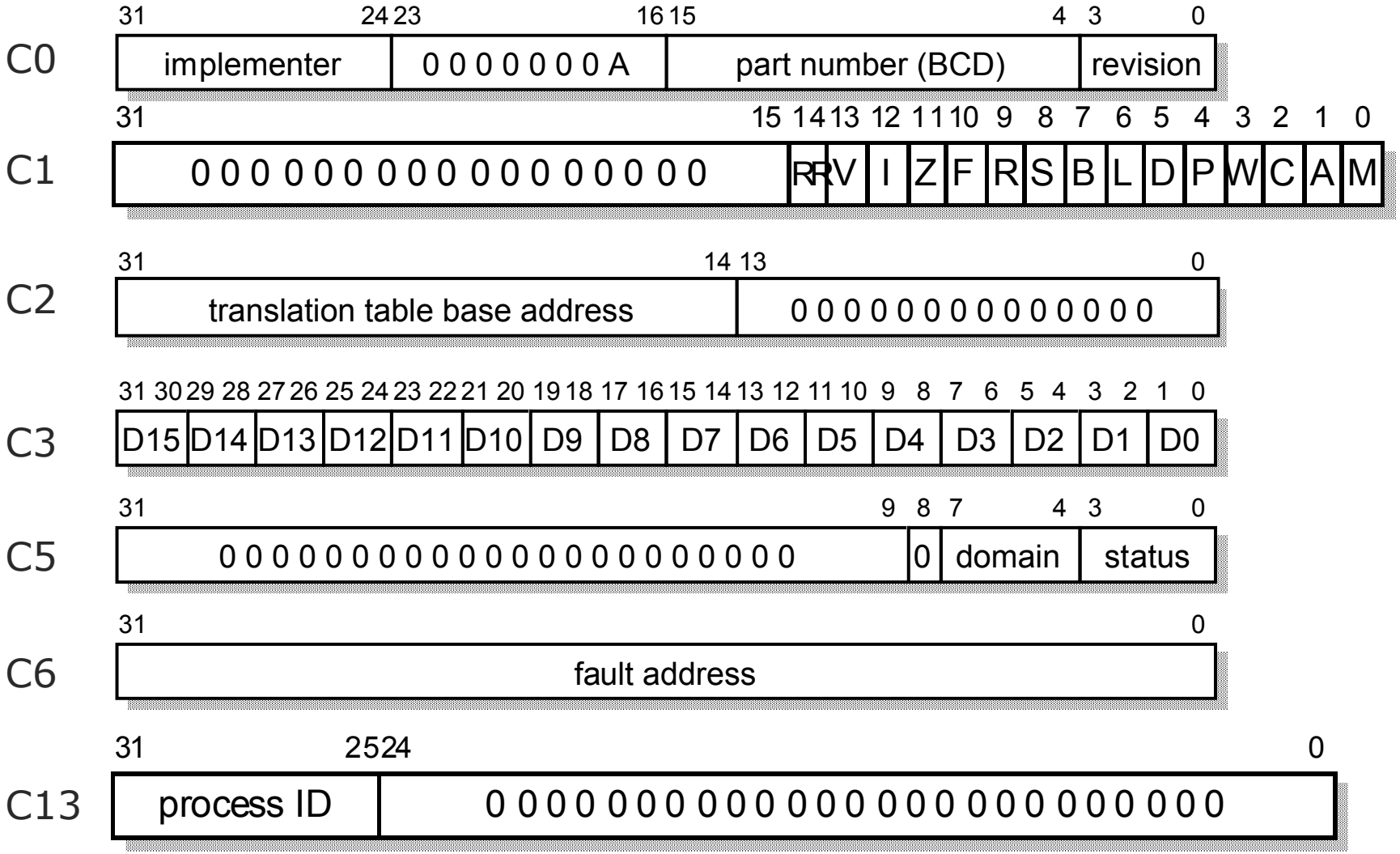
[PXA255 Processor – CP15]

- **CP15 Register structure**

Register	Purpose
0	ID Register
1	Control
2	Translation Table Base
3	Domain Access Control
5	Fault Staus
6	Fault Address
7	Cache Operations
8	TLB Operations
9	Read Buffer Operations
10	TLB lockdown
13	Process ID Mapping
14	Debug Support
15	Test & Clock Control
4,11~12	UNUSED

[PXA255 CoProcessor]

- **CP15 register**



[RISC – PowerPC]

- **Power-PC 코어의 특징**

1) 정수 값을 가질 수 있는 **32**개의 범용 레지스터 **(GPRs)**

2) 레지스터와 메모리 사이의 자료를 읽고 쓸 수 있는 명령어

3) 파이프라인과 병렬적인 명령어 추출을 쉽게 하기 위한 단일 크기 명령어 구성

4) 정수연산 명령어는 결과를 저장할 레지스터를 정할 수 있음

5) 정확한 예외처리 구조

6) 각 기능을 하드웨어나 소프트웨어로 구현 할 수 있는 유연한 아키텍처

7) 캐시 제어를 위한 사용자 레벨의 명령어 지원

8) 메모리에 대한 약한 일관성 모델 적용

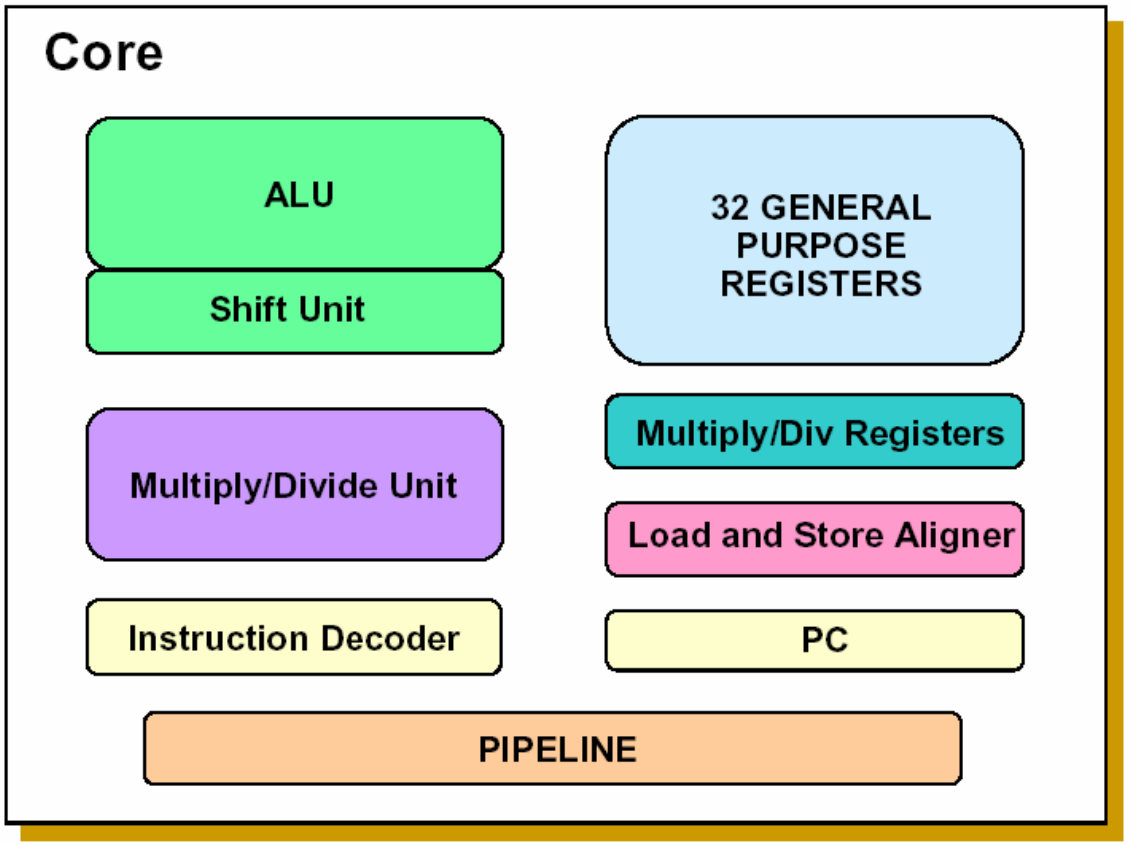
9) 단일 캐시 뿐 만 아니라 명령어와 자료의 분리된 형태의 **Harvard** 아키텍처 캐시도 지원

10) big-endian뿐만 아니라 **little-endian**도 지원

11) 32-bit 주소 뿐 만 아니라 **64-bit** 주소도 지원

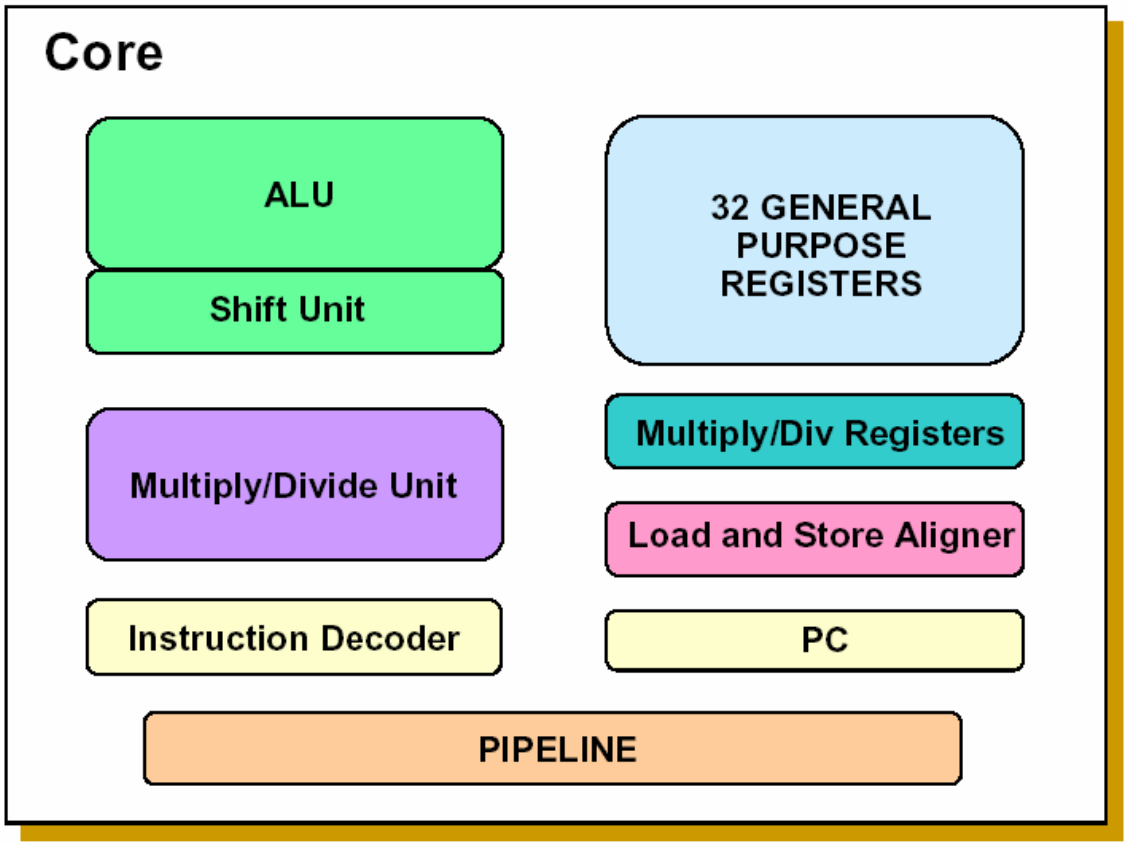
MIPS 코어 구조

- **MIPS 코어 구조도**



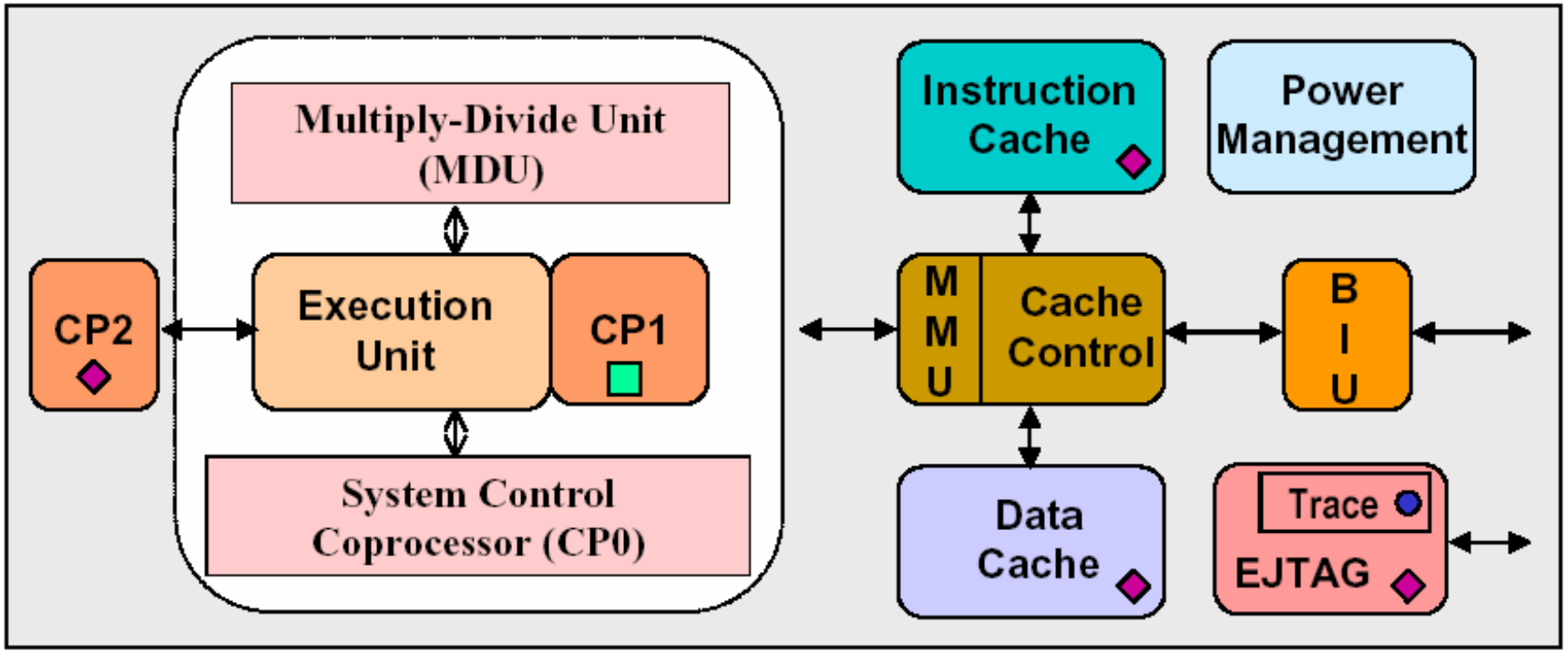
[MIPS 코어 구조]

- **MIPS 코어 구조도**



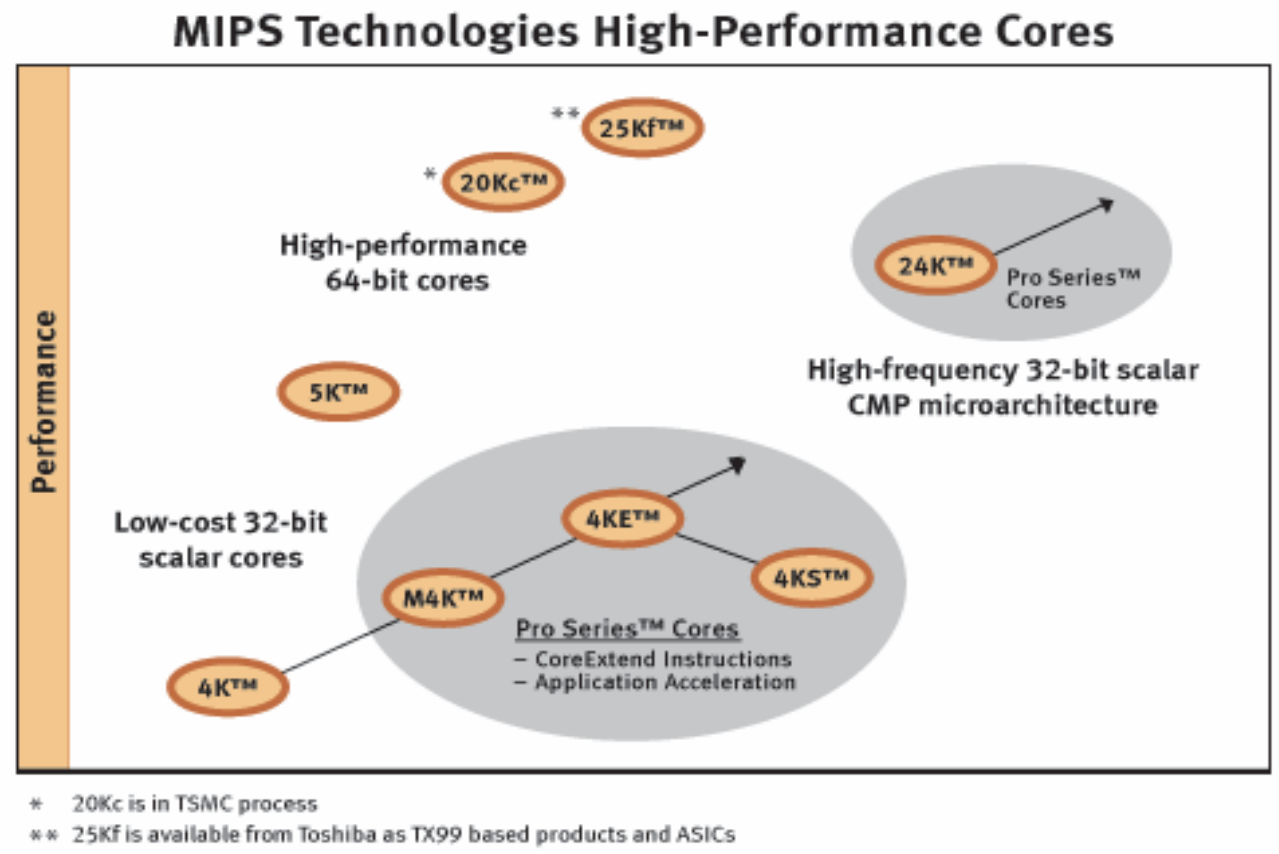
[MIPS 코어 구조]

- **MIPS 코어 구조도**



[MIPS 코어 로드맵]

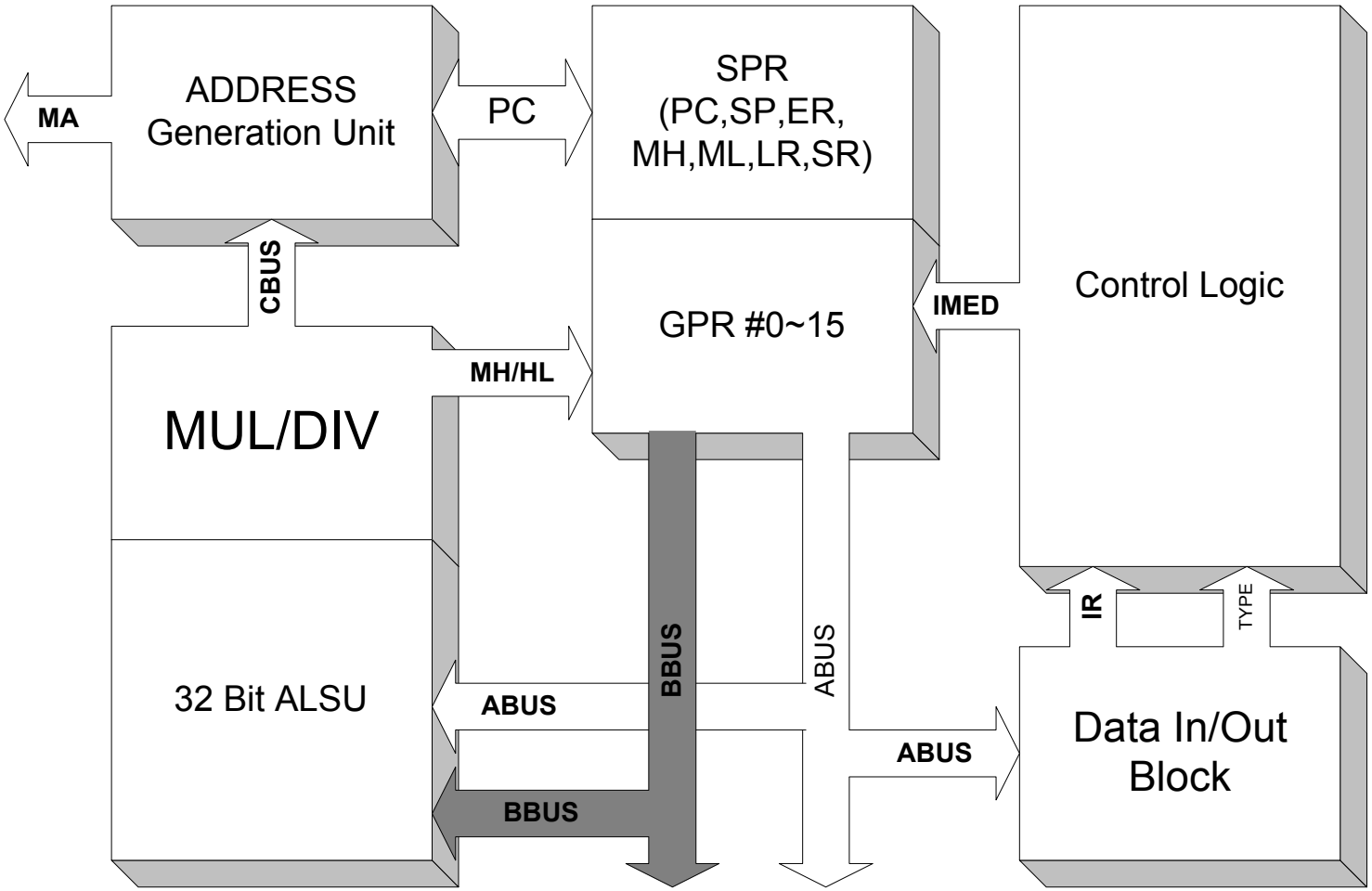
- **MIPS 코어 로드맵**



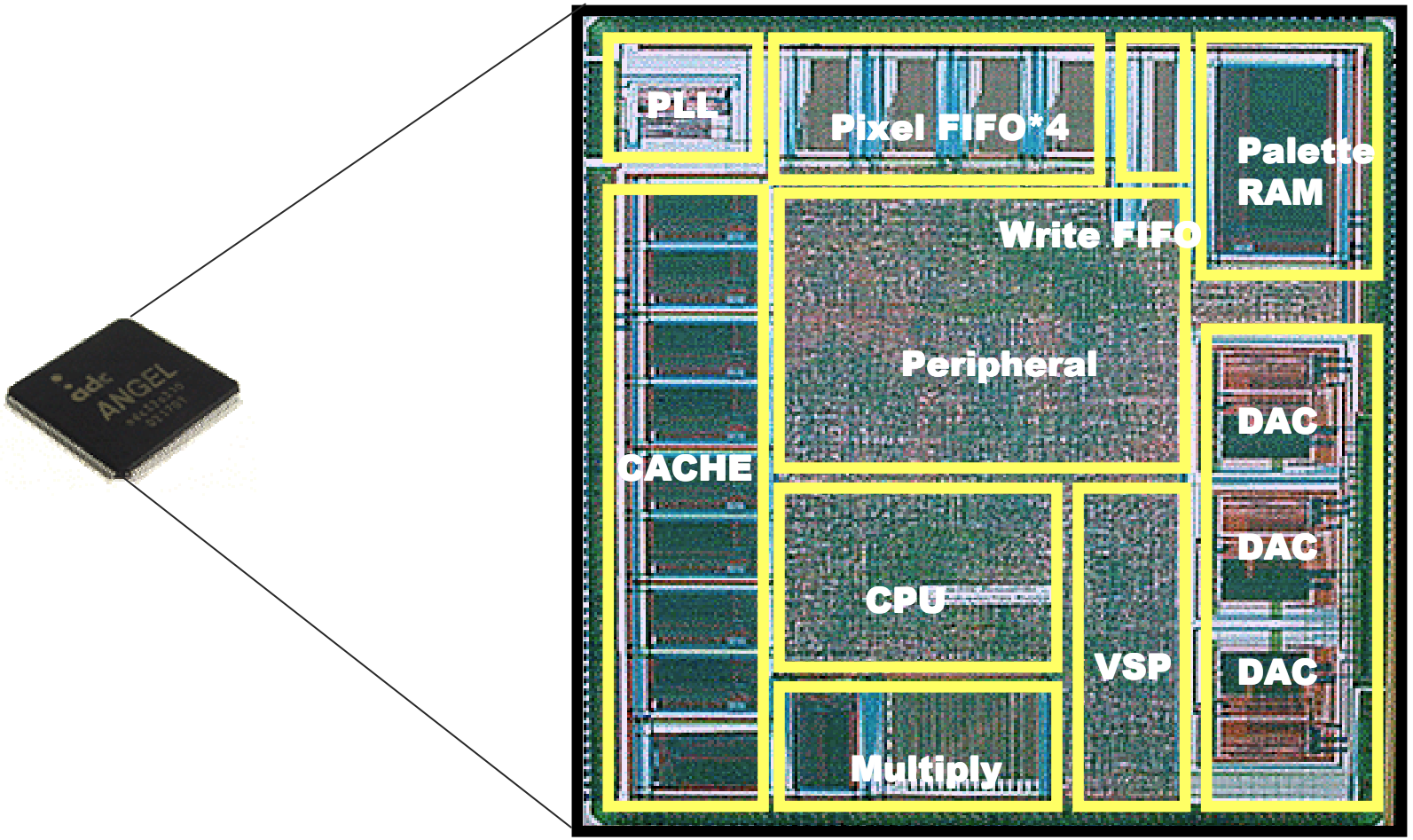
[EISC - 개요]

- EISC 프로세서의 종류
  - 16비트
  - 32비트: AE32000
  - 64비트
- AE32000의 특징
  - 16비트 명령어 구조
  - 하바드 구조
  - 5단계 파이프라인
  - 32비트 연산을 지원하는 ALU 와 Barrel Shifter
  - 32 × 32-bit 곱셈기
  - CNT명령 : 허프만(Huffman) 디코딩을 효율적으로 지원
  - MAC (Multiply and ACcumulate) 연산기: DSP 응용

[EISC – AE32000 프로세서 블록도]



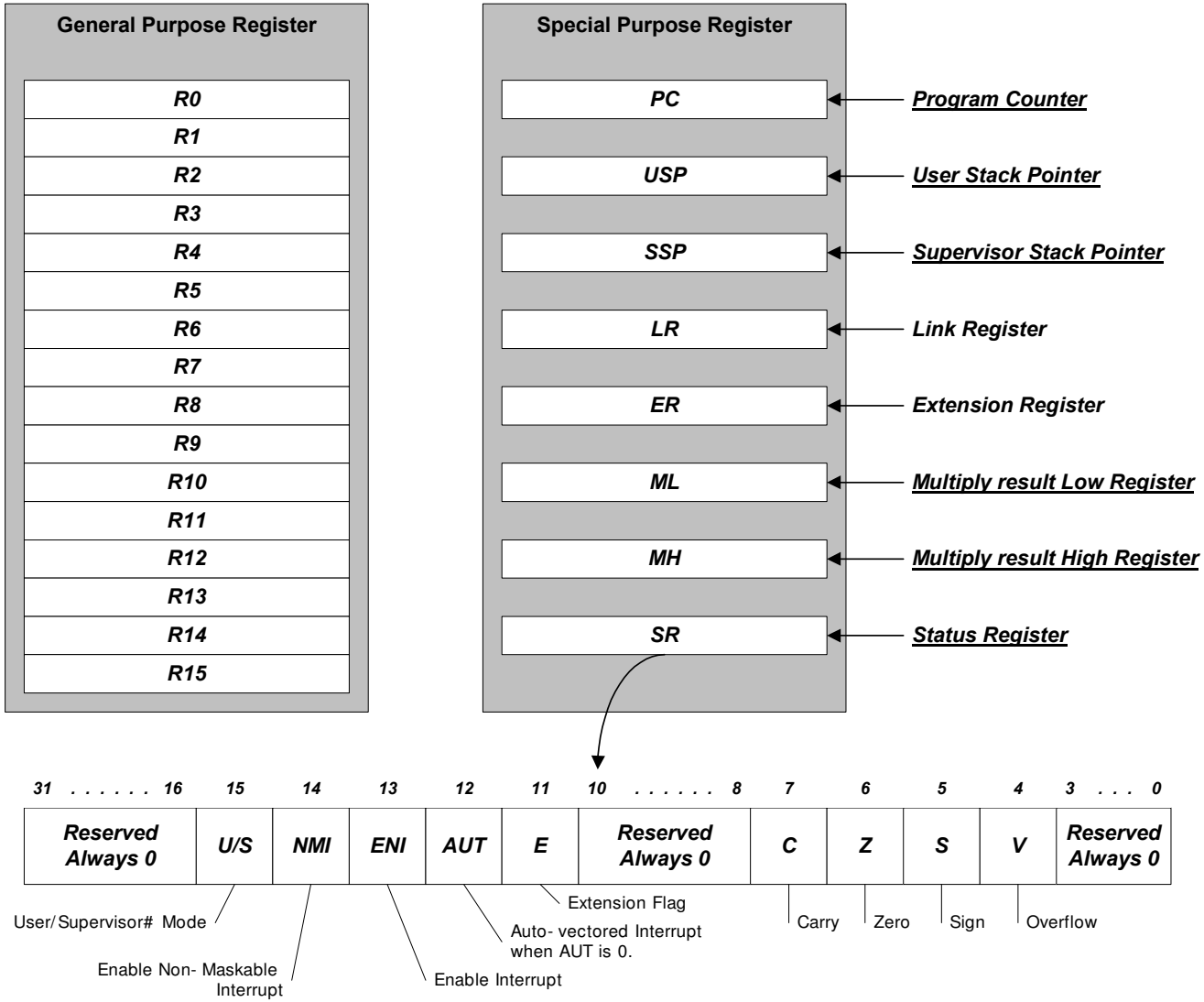
[EISC – ANGEL 칩/내부 사진]



[EISC – 프로세서 모드 : 3 가지]

- 관리자모드
  - OS 사용
  - 사용자 모드에서 SWI로 진입; POP SR로 복귀
- 사용자모드
  - 응용 S/W 모드
  - 사용자에게 할당된 자원만 사용가능
- OSI 모드
  - 프로세서의 동작을 디버깅하기 위한 모드
  - 관리자 모드와 동일한 권한을 가짐
    - 관리자 모드와 구분되는 스택을 가짐

[EISC – 프로그래밍 모델]



[EISC – AE32000 레지스터]

- 범용 레지스터
  - 16개
  - 할당 순서: 0, 1, 2, 3, 4, 6, 10, 11, 12, 13, 14, 15
  - 5번: 프레임 포인터
  - 7번: 인덱스 레지스터
  - 8, 9번: Argument Passing 용도.

- 특수 목적 레지스터
  - 9개
  - PC: Program Counter
  - USP: User Stack Pointer
  - SSP: Supervisor Stack Pointer
  - LR: Link Register
  - ER: Extension Register
  - ML/MH: Multiply result Low/High Register
  - SR: Status Register

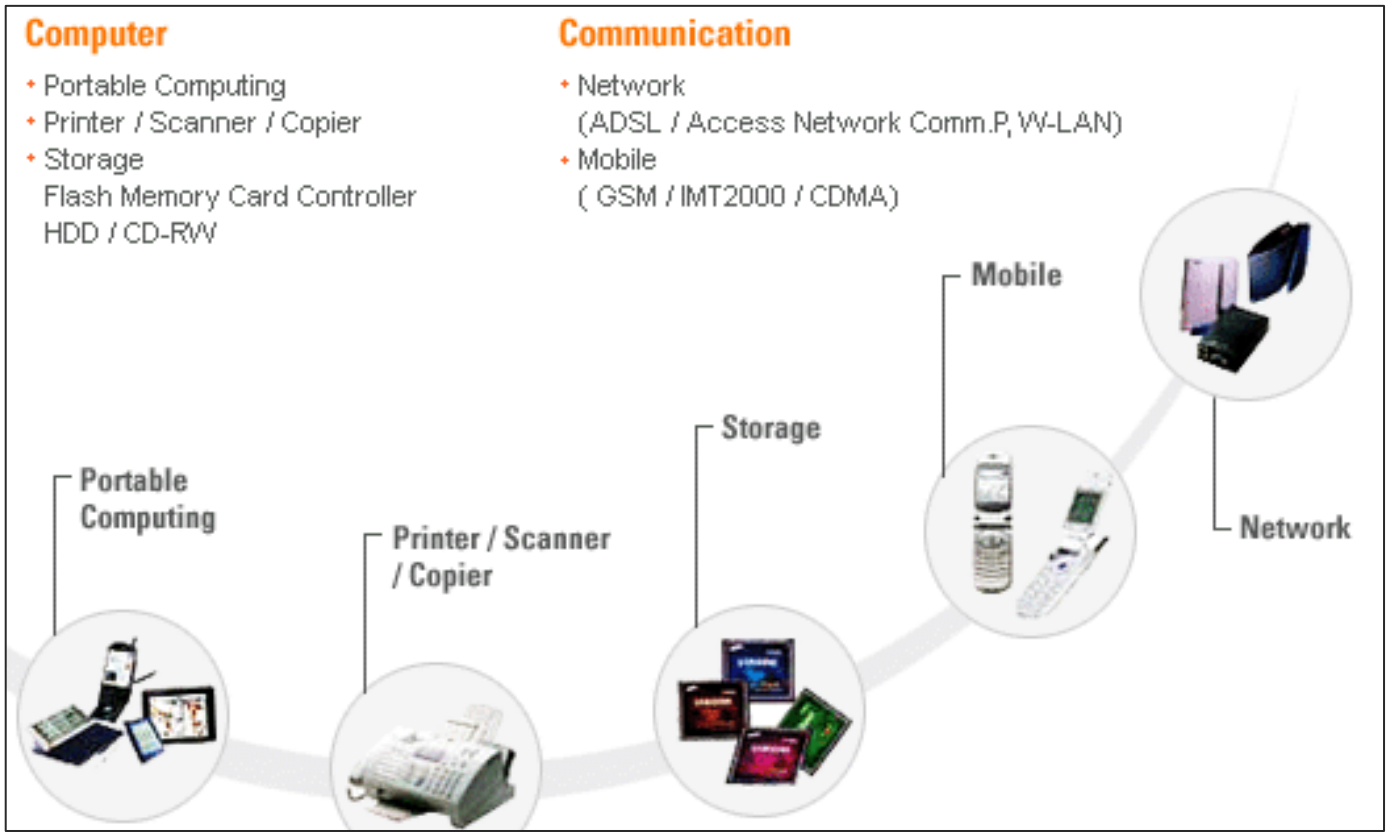
[EISC – AE32000 레지스터]

- SR : 15~8 번은 관리자 모드에서만 사용가능
- LR : 호출 되었던 원래 위치로 복기 주소
- PC : Next Program Address
  - PC = (address of current instruction) + 2
- ER : LERI 사용시 부족한 상수 값을 저장
- MH/ML : MAC 연산 결과 저장
- SP : 3 가지 동작 모드마다 1개의 SP

[EISC – SR]

- 프로세서 모드 (bit.15, bit9) : 프로세서의 동작 모드
  - 00 : 슈퍼바이저 모드 / 01 : OSI mode1x : 사용자 모드
- NMI (bit.14) : Non-maskable 인터럽트
  - 0 : NMI disable / 1 : NMI enable
- ENI (bit.13) : 인터럽트
  - 0 : INT disable / 1 : INT enable
- AUT (bit.12) : 외부 vectored interrupt
  - 0 : Auto-vectored interrupt / 1 : Vectored interrupt
- Extend (bit.11) : ER 사용 여부
- Endianess (bit.10) : 엔디안(endian) 지정
- Lock (bit.8) : 임계 구역(critical section)에 대한 제어
- Carry (bit.7) : 캐리(carry)
- Zero (bit.6) : 연산의 결과가 '0'.
- Sign (bit.5) : 연산의 결과에서 MSB
- oVerflow (bit.4) : 연산의 결과에서 오버플로우가 발생

[SoC – 개요 & Applications]

- **SoC 목적: Multi-chip solution => single chip solution**
  - 

[SoC – 구현방법]

- **ASIC, ASSP, PLD (엑스칼리버)**

비교항목	ASIC	ASSP	엑스칼리버
집적도	High	Moderate	High
개발 비용	High	Low	Low
제품 단가	Low	Low	Moderate
설계 용이성	Low	Low	High
출시 시기	Long	Moderate	Short

[SoC - ARM]

- **AMBA(Advanced Micro-controller Bus Architecture) 이용하여 IP를 연결**
  - **AHB (Advanced High-performance Bus): 고속, burst, split 전송**
  - **ASB (Advanced System Bus) : 이전 규격**
  - **APB (Advanced Peripheral Bus) : 저속**