1부. 임베디드 시스템

01장. 임베디드 시스템 개요
02장. 임베디드 시스템 설계

DIGNSYS
www.dignsys.com

교 육 목 표

- 임베디드 시스템을 이해한다.
- 임베디드 시스템의 구조를 이해한다.
- 프로세서에 대하여 이해한다.
- 시스템 버스에 대하여 이해한다.
- 메모리 장치에 대하여 이해한다.
- 입출력 장치의 구조 및 특징을 이해한다.
- 임베디드 시스템의 설계 절차와 방법을 이해한다.
- 임베디드 시스템의 개발 환경을 이해한다.

목 차

임베디드 시스템

- 하드웨어와 소프트웨어가 조합되어 특정한 목적을 수행하는 시스템
- 특정한 기능을 수행하도록 마이크로 프로세서와 입출력 장치를 내장하며, 이를 제어하기 위한 프로그램이 내장되어 있는 우리의 일상 생활에서 사용되는 각종 전자기기, 가전제품, 제어장치 등

임베디드 시스템 응용분야

- 정보 가전 : 세탁기, 오디오, 인터넷 냉장고, **HDTV** 등
- 제어분야 : 공장자동화, 가정자동화, 로봇 제어, 공정제어 등
- 정보 단말 : 핸드폰, **PDA**, 스마트 폰, 네비게이션, **MP3, PMP, DivX** 플레이어, 디지털 카메라 등
- 네트워크기기 : 교환기, **Router**, 공유기, 홈 게이트웨이 등
- 게임기기 : 가정용 게임기**(PS2, XBox)**, 지능형 장난감 등
- 항공**/**군용 : 비행기, 우주선, 로켓, 야전 이동단말**(GPS, GIS)**
- 물류**/**금융 : **ATM, RFID**, 물류단말, 영업단말 등
- 차량**/**교통 : 자동차, **ITS** 등
- 사무, 의료 : 전화기, 프린터, **Heart pacer**, 수술로봇, 증강현실장비

정보단말기

❑ 정보단말기

- ❖ 단순한 통화 중심의 이동 전화기에서 각종 정보검색, 오락, 메시징 등의 복합 기능이 수행되는 디지털 정보단말기기로 발전
- ❖ 단말기기 각각의 기능에 맞는 마이크로프로세서, 메모리, 운영체제, 응용 프로그램 등으로 구성
- ❖ 앞으로는 다양한 단말기기가 하나의 기기로 통합될 것으로 예상됨
- ❖ 핸드폰, PDA, 스마트 폰, MP3 플레이어, PMP, 게임기기 등

게임기

- 고성능 프로세서 탑재
- 마이크로소프트의 **Xbox**
- 소니의 **playstation 2**
- 닌텐도 게임보이 어드벤스 **(nintendo gameboy advance)**
  - 32-Bit ARM 프로세서
  - 2.9인치 TFT 스크린
  - 32,768 색상을 지원하는 휴대형 게임 장치

물류/금융/사무용기기

- 물류/금융
  - 물류 : POS 단말기
  - 금융 : 자동 현금 입출금기 혹은 ATM 단말기
- 사무용기기
  - 프린터, 스캐너, 팩스, 복사기, 이들의 기능을 하나로 모은 복합기 등

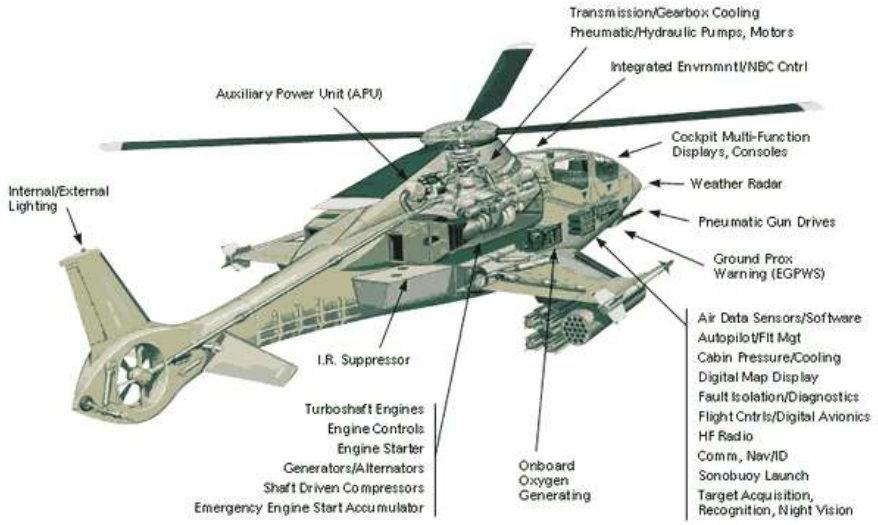
공장자동화

- **공장자동화 : FA (Factory Automation)**
  - 특정 기계나 장비를 통해 생산 과정을 자동적으로 관리하는 시스템
  - 센서와 제어 시스템, 로봇 등으로 구성하여 무인시스템을 구축
  - 공장 자동화 및 로봇은 실시간 시스템과 임베디드 시스템 발전의 원동력
  - 생산성증대: 인건비감소, 오류감소, 품질의 균일화, 생산기간단축
  - 로봇, conveyor belt

우주/항공

- 항공기
  - 보통 수 백 개의 프로세서 탑재
- 우주왕복선
  - Pathfinder -실시간 운영체제인 VxWorks가 탑재된 것으로 유명
  - 대표적인 실시간 시스템의 하나
  - 영상처리, 통신 등 모든 처리기능을 복합적으로 가짐

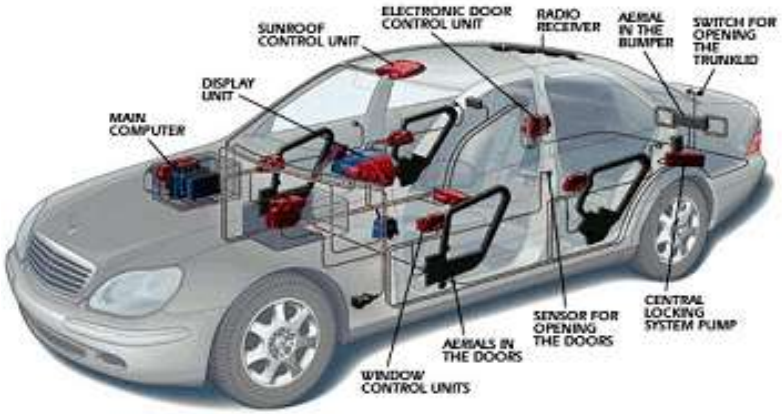
NASA Pathfinder
(mission to MAR 1997)



교통

❑ 교통

- ❖ 자동차의 엔진 및 각종 제어 시스템, 무인 자동화 시스템
- ❖ 지능형 교통시스템(ITS : Intelligent Transport Systems) 등



지능형 장난감

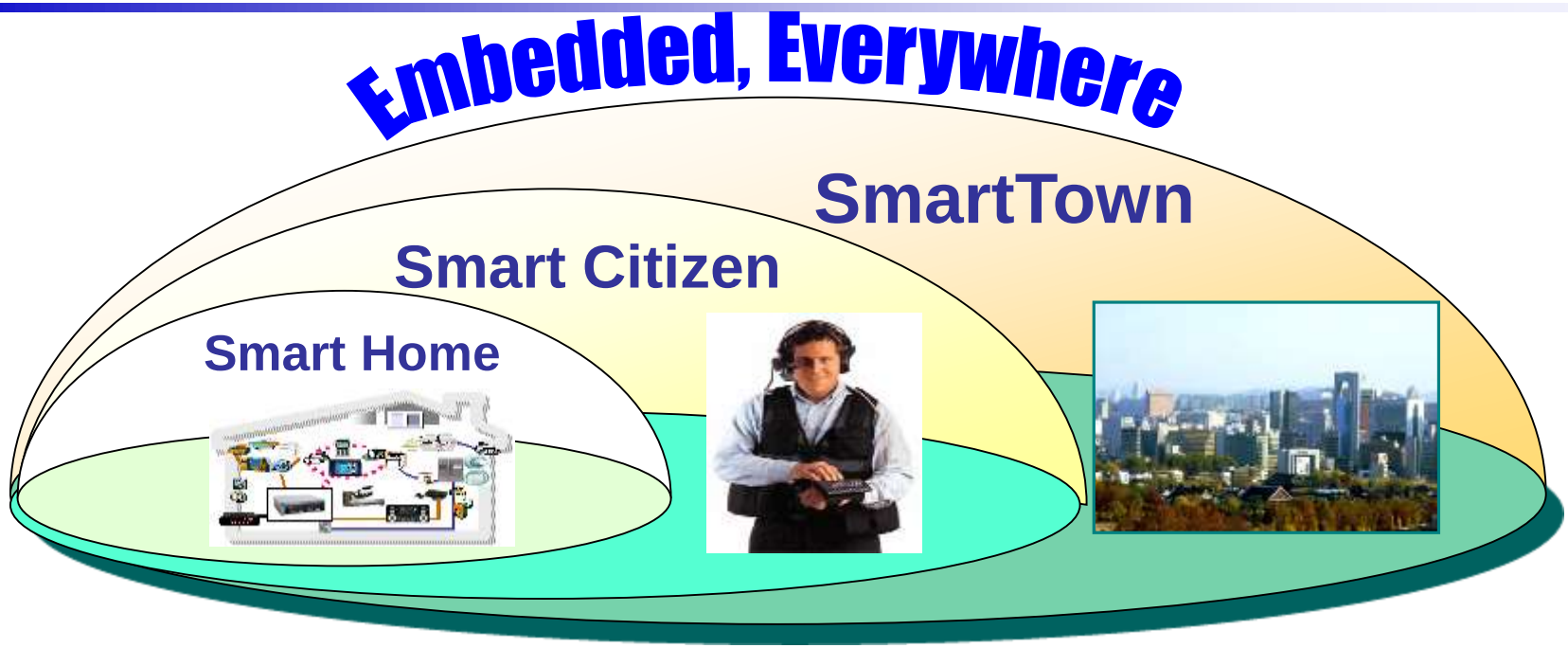
❑ 지능형 장난감

- ❖ 단순한 장난감의 형태에서 지능성을 갖는 형태로 변화

통신기기

- 디지털 교환기, **PABX (private automatic branch exchange)** 등의 음성 서비스 통신기기
- 라우터, 게이트웨이, 공유기 등의 유무선 데이터 통신 장비
- **Set-top box**

임베디드 시스템 전망



서버	홈서버	퍼스널 서버	웹서비스 서버
클라이언트	정보가전	웨어러블 단말	임베디드 시스템
통신망	홈 네트워크	Personal Area Network	Ubiquitous Network
서비스유형	홈 서비스	모바일 서비스	유비쿼터스 서비스

정보통신 시스템, 단말기, 자동제어 시스템에 수요 증가에 따른 임베디드 시스템 수요 급증

Ubiquitous Computing 이란?

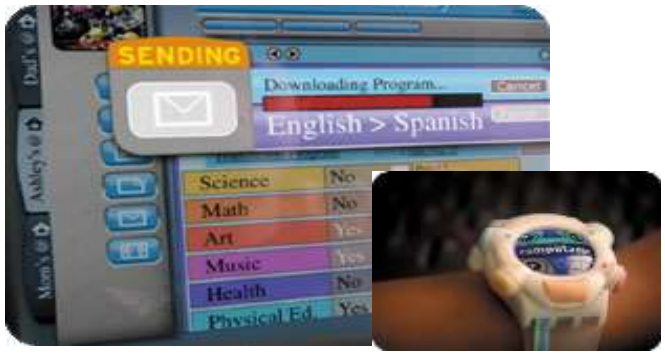
- **“Ubiquitous” 는「편재:보편적으로 존재한다(라틴어)」**
  - ‘(신은) 어디에나 널리 존재한다’ 라는 의미의 라틴어에서 유래
- **“Ubiquitous Computing” 은**

  사용자가 네트워크나 컴퓨터를 의식하지 않고 장소에 상관없이

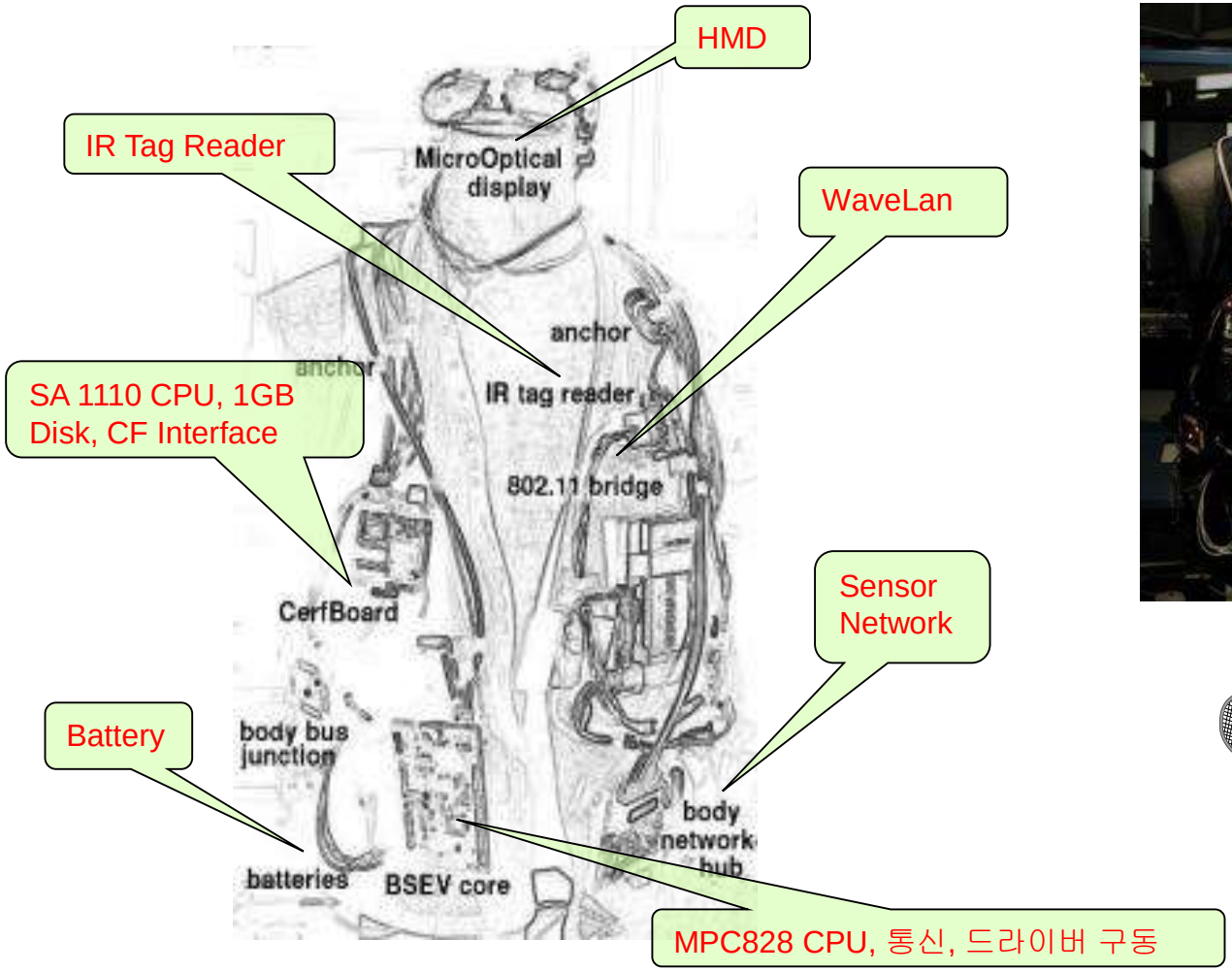
  자유롭게 네트워크에 접속할 수 있는 정보통신 환경

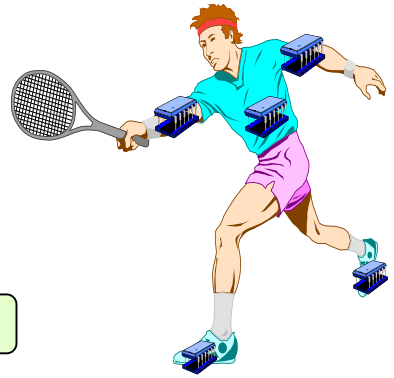
CoolTown 프로젝트 (Hewlett-Packrd)

- 유/무선 통신 네트워크 기술과 웹 기반의 미래 도시 모델 구현
  - 전자태그 및 내장형 웹서버(200KB), 근거리 무선통신이 가능한 PDA, 그리고 기존의 웹 인프라를 기반으로 하는 전자 공간에서 현실세계의 사람과 사물이 연동되는 시나리오 제시
- 현실세계의 사람・장소・사물이 가상세계에서도 연동되는 환경 구축
  - 쿨타운의 채택 기술
    - 표준 바코드 기술 , 무선 송수신 기술, 적외선 및 블루투스(Bluetooth) 기술 등
- 쿨타운 응용 사례
  - 쿨타운 미술관, 쿨타운 회의실, 쿨타운 버스
  - 커스터머 서비스, e-비즈니스, 원격교육 및 원격의료, 화재 및 방재에 대응한 서비스



Wearable Computer





Smart Wear(의류)

- 섬유(직물)나 의복 자체가 외부 자극을 감지하고 스스로 반응

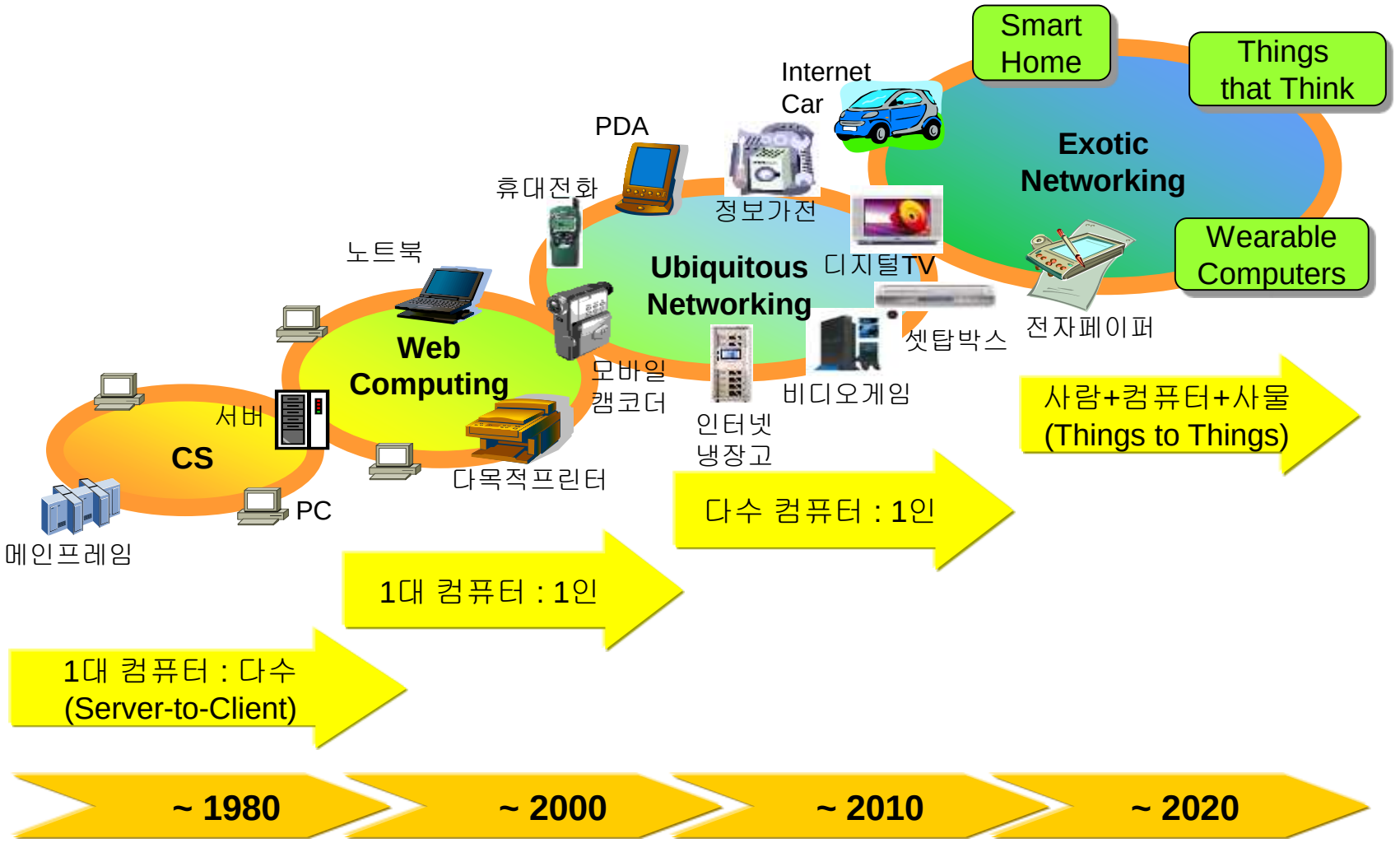
  (Assisted Cognition System)

- 센사텍스(Sensatex)사의 스마트 셔츠
  - 심장박동, 호흡, 혈압, 체온, 칼로리 소모량 등을 센싱
  - 원격 의료용, 스포츠용, 유아용
- 인피니온(Infineon)사의 스마트 의류
  - MP3플레이어와 이어폰을 내장한 의류, 음량은 소매단추로 조절

심박수 및 소모칼로리 계산용 시계 (핀란드)

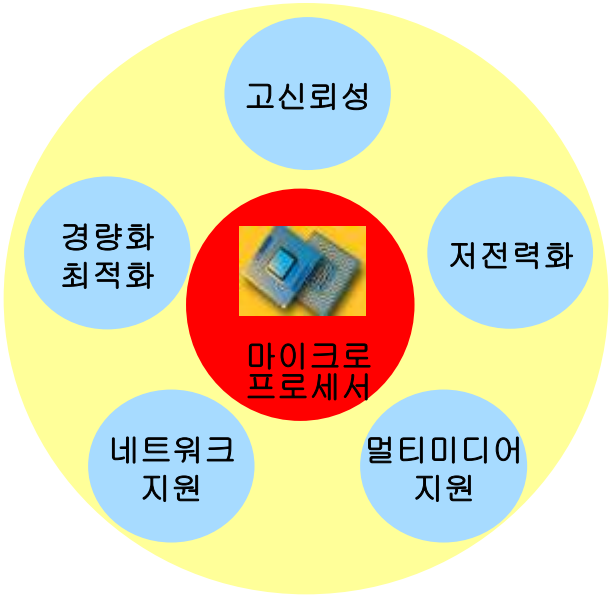
전화기에 내장된 혈류량 센서(일본)

새로운 시대의 가능성 - 임베디드



임베디드 시스템의 특징

- 마이크로 프로세세/컨트롤러를 비롯한 하드웨어와 소프트웨어를 내장**(embedded)**하여 특정한 기능을 수행
- 수행하는 기능은 미리 정해 진다.
- 소형, 경량, 저전력
- 가격에 민감하다
- 안정성이 뛰어나야 한다.
- **Real-time** 기능을 필요로 하는 시스템이 많다.

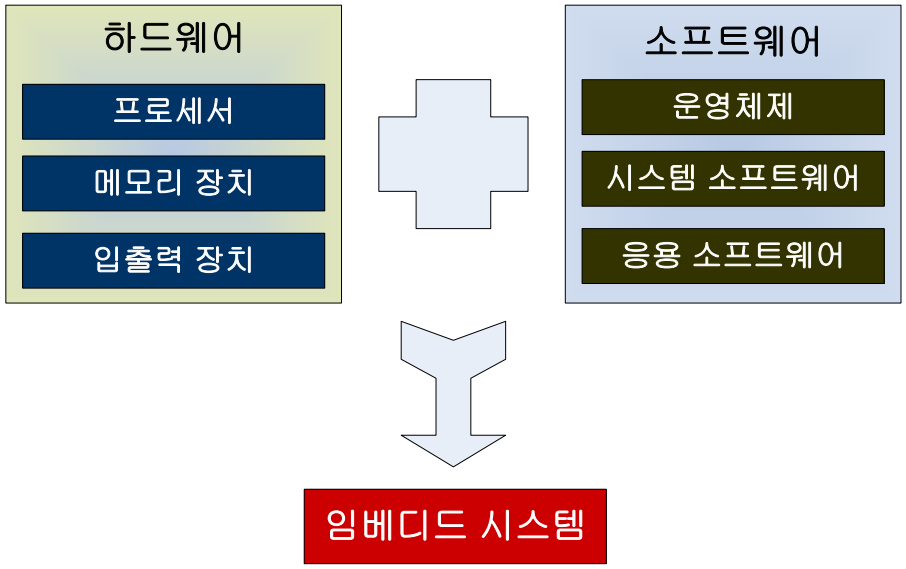


리얼타임 시스템

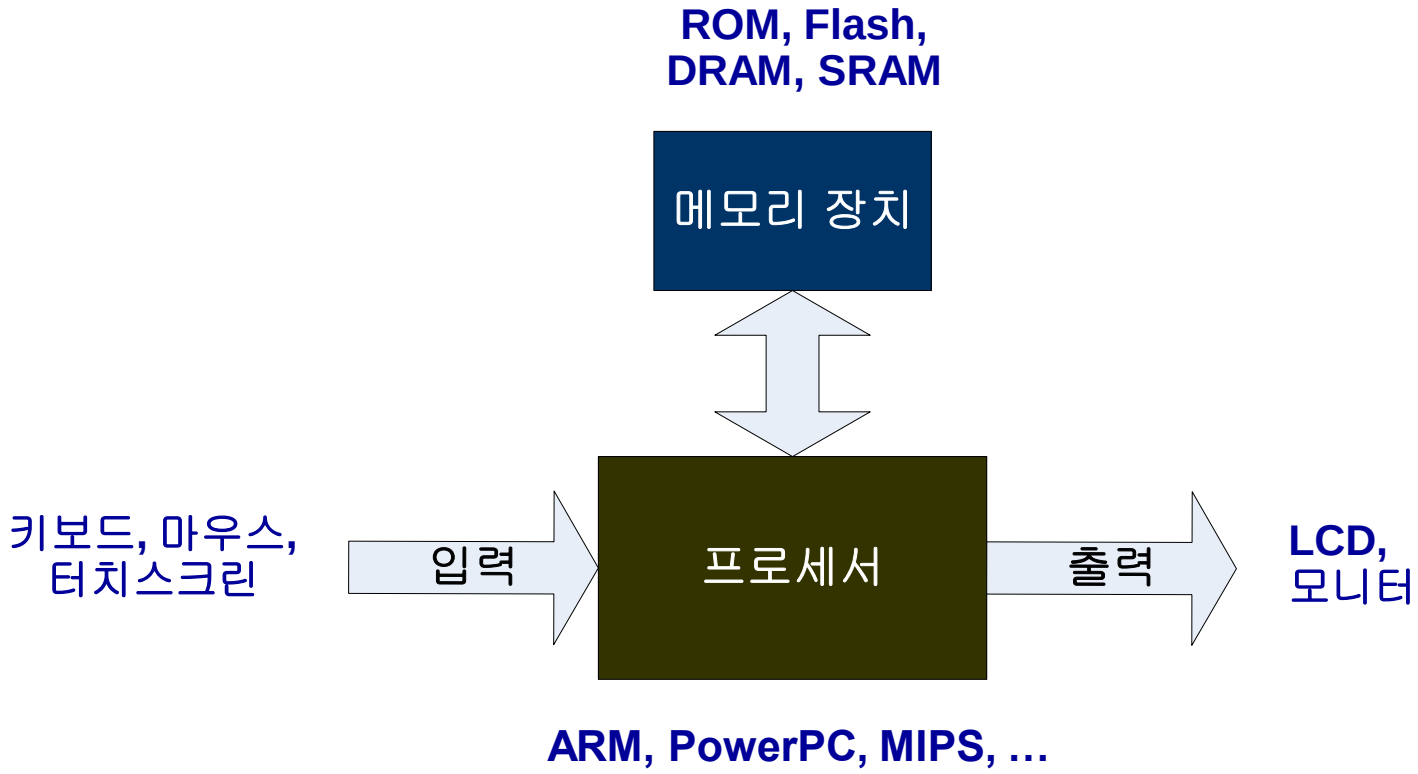
- 주어진 입력 조건을 주어진 시간 내에 처리하는 시스템
- 리얼타임 시스템의 종류
  - 하드 리얼타임 시스템
    - 리얼타임이 보장되지 않으면 시스템에 치명적인 오류를 유발
    - 대부분의 제어용 기기
    - 예) 공장 자동화 등
  - 소프트 리얼타임 시스템
    - 주어진 시간 내에 결과를 출력하지 않아도 시스템 전반에 큰 영향이 없는 시스템
    - 예) 네트웍 장비

임베디드 시스템의 구성

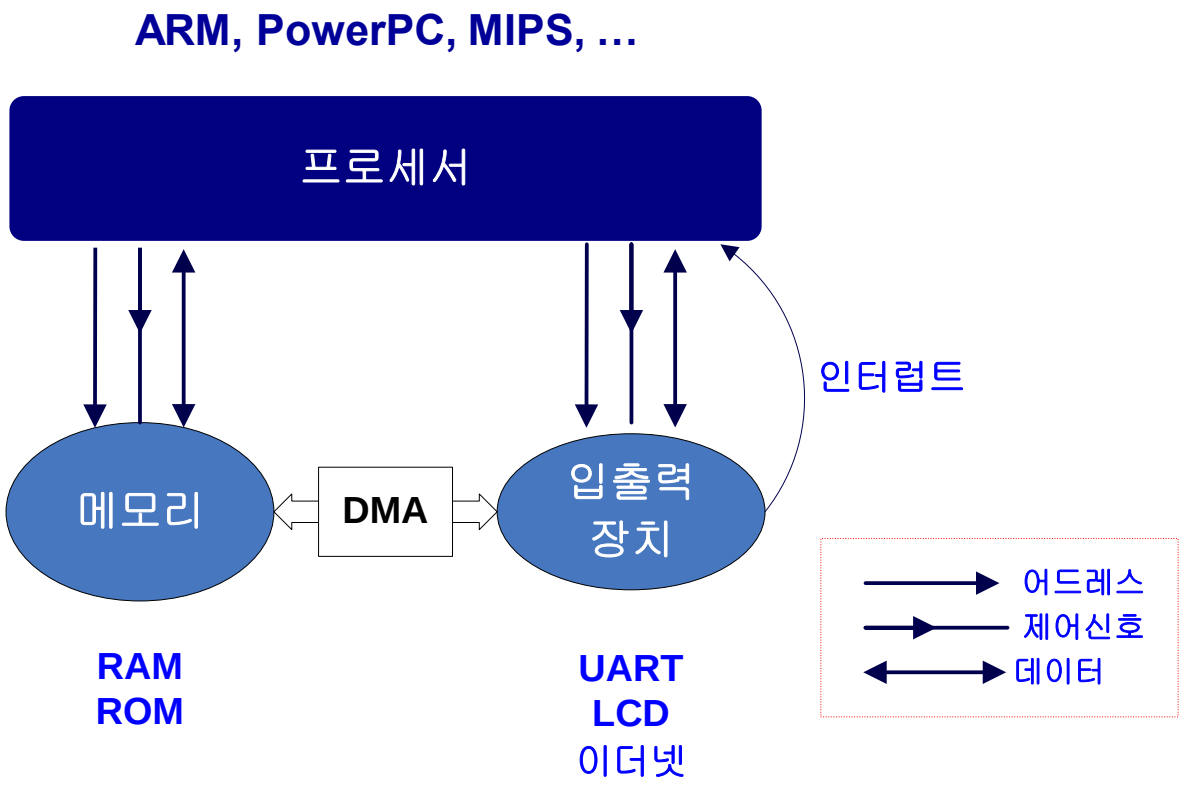
- 하드웨어
  - 프로세서(컨트롤러), 메모리 장치(ROM,RAM),
    입출력 장치(네트워크 장치, 센서, 구동기 등)
- 소프트웨어
  - 운영체제(OS), 시스템 S/W, 응용 S/W



하드웨어 구조


ROM, Flash,
DRAM, SRAM
메모리 장치
키보드, 마우스,
터치스크린
입력
프로세서
출력
LCD,
모니터
ARM, PowerPC, MIPS, …

하드웨어 동작


ARM, PowerPC, MIPS, …
프로세서
인터럽트
메모리
DMA
입출력
장치
어드레스
제어신호
데이터
RAM
ROM
UART
LCD
이더넷

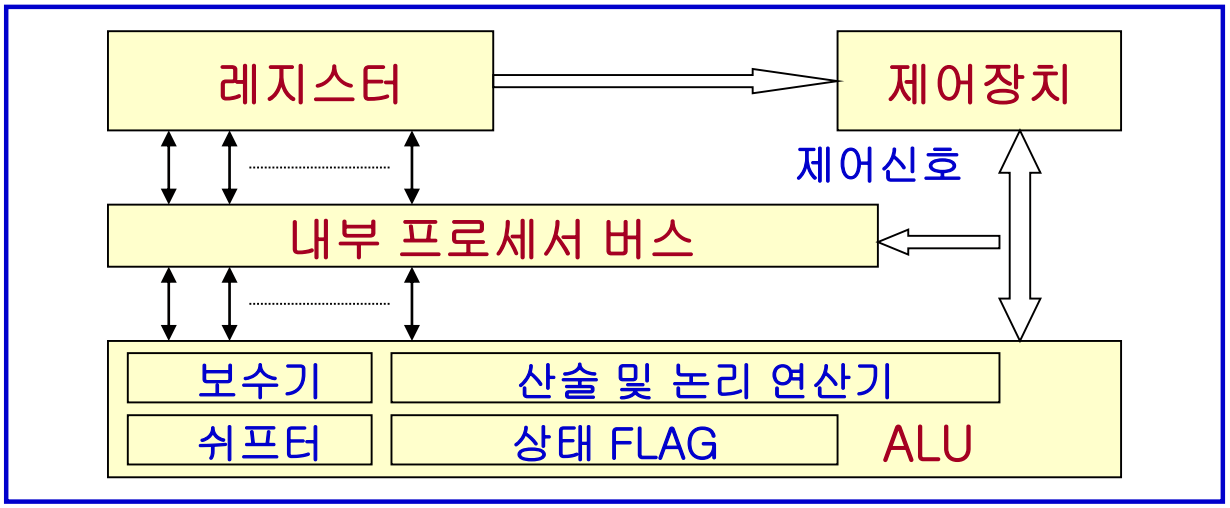
목 차

프로세서

- 프로세서 디지털 시스템의 핵심 부분
  - CPU(Central Processor Unit)라고도 한다.
- 프로그램을 메모리 장치에서 읽어 연산처리, 비교처리, 데이터 전송, 편집, 변환, 테스트와 분기 등의 데이터 처리와, 각종 입출력 장치 구동
- 제어장치, 연산장치, 레지스터와 데이터 버스로 구성된다.

프로세서의 구조

- 레지스터(**Register**)
- 산술 논리 연산 장치 (**ALU : Arithmetic Logic Unit**)
- 제어 장치 (**CU : Control Unit**)
- 버스(**BUS**)



레지스터 (Register)

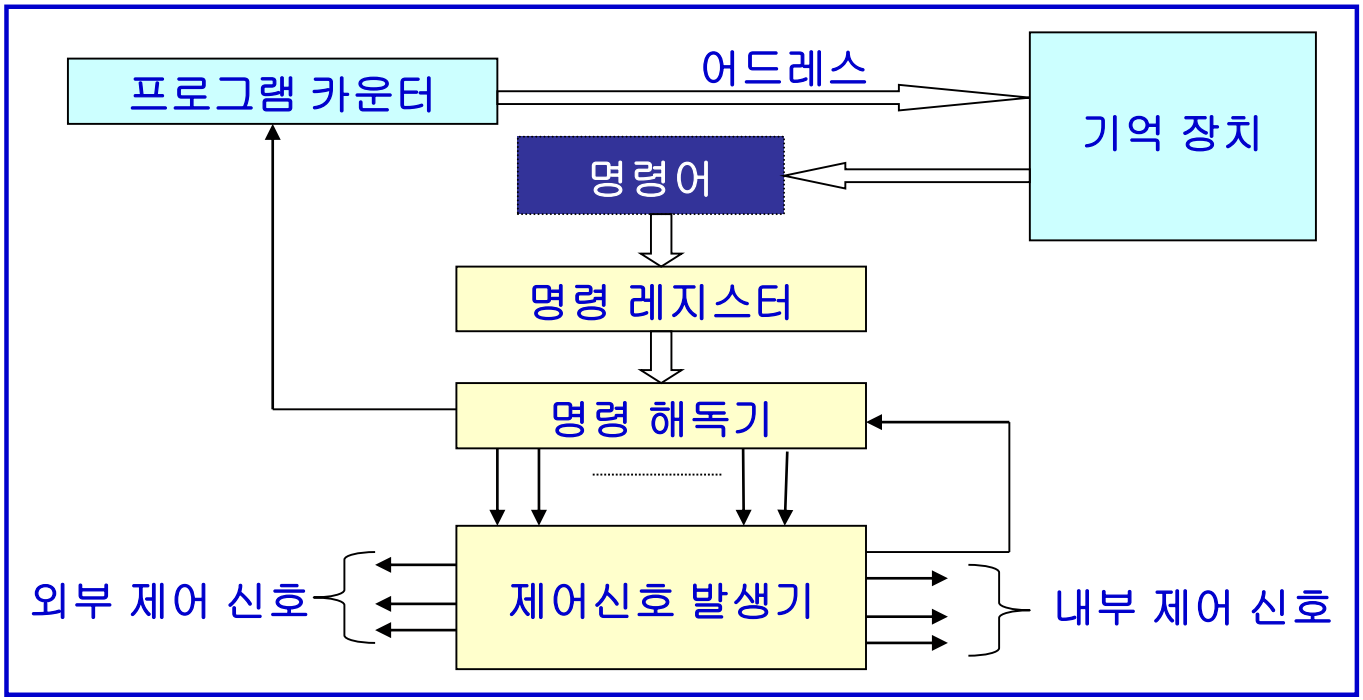
- 프로세서 내부에서 데이터를 일시적으로 보관하는 기억 장치
  - Flip-flop와 Latch로 구성되어 있다
- 프로세서 레지스터의 종류
  - 법용 레지스터
    - 프로그램 또는 데이터 처리에 필요한 작업을 수행하기 위해서 사용
  - 제어용 레지스터
    - 프로그램이나 프로세서를 제어
    - 프로그램 카운터(PC : Program Counter) 등
  - 상태 레지스터
    - 프로세서의 상태를 나타낸다.

산술 논리 연산장치 (ALU)

- 산술 연산 수행
  - 덧셈, 뺄셈 등
- 논리 연산 수행
  - AND, OR 등
- 상태 레지스터 또는 **flag** 레지스터에 연산 결과 기록
  - Carry 발생, overflow 발생 등

제어 장치 (CU)

- 명령을 해석하고 실행
- 명령을 읽고 실행하기 위한 내부 데이터 흐름 제어



프로세서 버스

- **버스**
  - 디지털 회로에서 시스템의 여러 장치들을 연결하는 경로
- **내부 버스 (Internal Bus)**
  - 프로세서 내부에서 레지스터와 ALU 사이의 신호를 교환하고, 그 결과를 다시 레지스터에 전달하는 경로
- **외부 버스 (external bus)**
  - 프로세서와 외부의 기억장치 사이, 그리고 프로세서와 I/O 장치 사이에 존재하는 버스
  - 외부버스 구성
    - 데이터 버스(data bus)
      - ✓ *데이터를 외부 장치에 전달하거나 외부 장치로부터 읽어오는 경로*
    - 어드레스 버스(address bus)
      - ✓ *프로세서에서 기억장치나 I/O 장치의 주소 정보 전송 경로*
    - 제어 버스(control bus)
      - ✓ *프로세서에서 기억장치나 I/O 장치에 입출력 동작을 지시하는 제어신호를 전송하는 경로*

프로세서의 종류

- **i386**
  - 오랜 기간의 사용으로 안정성 확보
  - PC와 동일한 개발 환경 구성
- **ARM**
  - 간단한 명령어 사용하고, 개발 환경이 간단하다.
  - 전력 소모가 작아서 휴대폰이나 PDA같은 휴대 단말기에 많이 사용
- **PowerPC**
  - 강력한 네트워크 기능을 포함한 SoC로 널리 알려짐
- **M68K**
  - 네트워크 장비 및 휴대 단말기에서 많이 사용
- **MIPS**
  - 고속의 처리 능력
  - 고속 네트워크 장비등에 많이 사용

마이크로 프로세서와 SoC

- 마이크로 프로세서(Micro-processor)는 한 개의 조그만 IC칩 속에 CPU의 모든 내용을 내장한 칩을 말한다.
  - 레지스터, 산술 논리 연산 장치, 제어 장치를 하나의 IC 칩에 구현
- 근래에는 한 개의 IC칩 속에 CPU뿐만 아니라 다양한 입출력 장치를 포함하는 SoC 형태로 발전되고 있다.

MCU, MPU와 SoC

- **SoC (System on Chip)**
  - 여러 개의 반도체 부품이 하나로 집적되는 기술 및 제품
  - 근래의 프로세서는 메모리, I/O 장치를 포함한 시스템 기능을 칩 하나에 구성하는 SoC 형태를 가지고 있다.
  - 프로세서(CPU), 메모리, DSP, 로직 IC 등 반도체부터 소프트웨어에 이르기까지 단일 칩으로 구현
- **MCU, MPU**
  - 프로세서를 내장하고 있는 SoC를 말한다.
  - 제조회사 및 사용자에 따라 MCU(Micro Controller Unit) 또는 MPU(Micro Processor Unit)라 부른다.

프로그램

- 프로세서를 통하여 어떤 결과를 얻기 위하여 프로세서가 받아 들일 수 있는 형태로 구성된 명령**(instruction)**을 나열하여 구성된 문장
- 문장은 단어를 나열하여 구성
  - 명령은 단어이고, 프로그램은 언어를 구사하는 것

기계어 와 어셈블리어

- 기계어 **(machine language)**
  - 프로세서가 이해할 수 있도록 '0' 과 '1'로 표현되는 2진수로 구성된 명령
  - 프로세서가 이해하기는 편리하지만 작성자가 프로그램 하기에는 불편
- 니모닉**(Mnemonic)** 코드
  - 기계어를 프로그램 작성자가 이해하기 편리한 기호로 표시한 명령
- 어셈블리어 **(Assembly language)**
  - 니모닉(Mnemonic) 코드에 보다 편리하게 프로그램 작성자가 이해 할 수 있도록 pseudo 명령(Instruction)을 첨부한 것

어셈블리어 의 특징

- 장점
  - 기계어에 비해 일고 이해하기 쉽다.
  - 기계어에 비해 프로그램의 오류 수정과 보관이 쉽다.

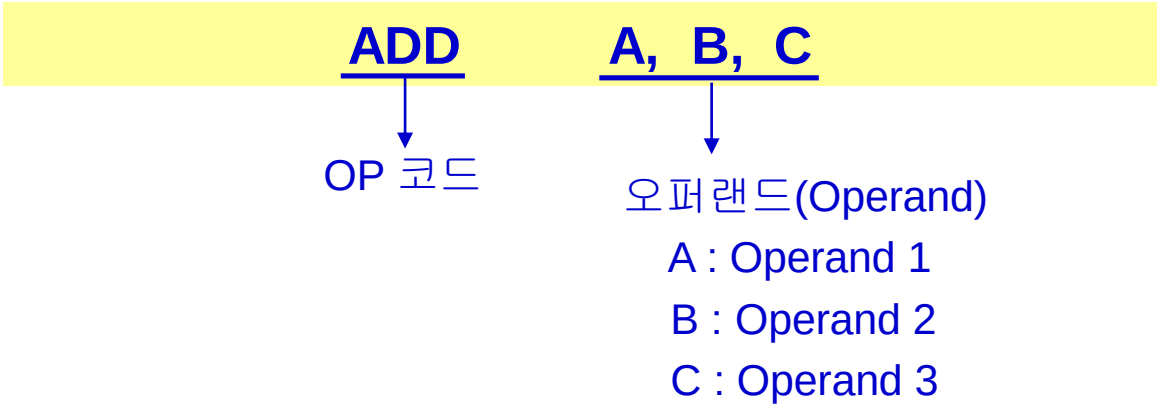
- 단점
  - 프로세서의 내부 구조 및 하드웨어를 자세히 알아야 한다.
  - 프로세서마다 어셈블리어가 서로 다르다.

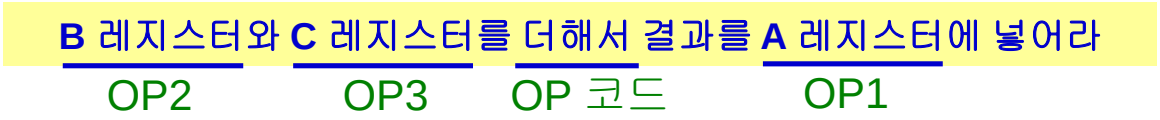
어셈블러와 어셈블

- **어셈블러 (Assembler)**
  - 어셈블리어를 '0'과 '1'의 2진수로 구성된 기계어로 자동 변환 하는 프로그램
- **어셈블 (Assemble)**
  - 어셈블리어 프로그램을 기계어로 바꾸는 과정

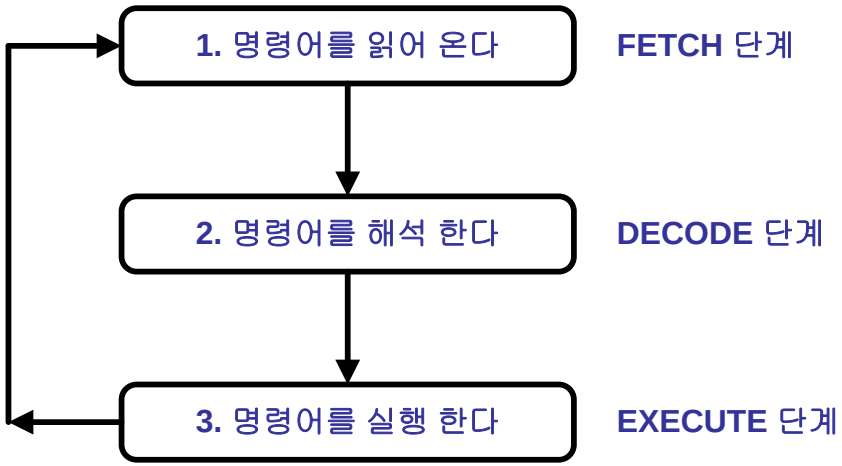
명령어 (Instruction)

- 명령어(Instruction)의 구성
  - OP 코드(Operation code)
    - 프로세서가 실제로 취해야 하는 동작
  - 오퍼랜드 (Operand)
    - OP 코드가 명령을 수행하기 위한 대상





명령어 수행 과정


1. 명령어를 읽어 온다
FETCH 단계
2. 명령어를 해석 한다
DECODE 단계
3. 명령어를 실행 한다
EXECUTE 단계

파이프라인(pipeline)

- 파이프라인
  - 프로세서로 가는 명령어들의 움직임, 또는 명령어를 수행하기 위해 프로세서에 의해 취해진 산술적인 단계가 연속적이고 겹치는 것을 말한다.
  - 파이프라인이 없으면 하나의 명령을 읽어와서 요구하는 연산을 수행하고, 다음 명령을 메모리에서 가져온다.
  - 파이프라인을 사용하면 프로세서가 산술 연산을 수행하는 동안에 다음 명령어를 메모리에서 가져온다.
- 파이프라인의 장점
  - 명령어를 가져오고 실행하는 단계가 끊임없이 계속 된다.
  - 주어진 시간동안에 수행될 수 있는 명령어의 수가 증가한다.

명령어 **Set**에 따른 프로세서 종류

- 명령어**(Instruction) Set**에 따른 프로세서 종류
  - CISC (Complex Instruction Set Computer)
  - RISC (Reduced Instruction Set Computer)

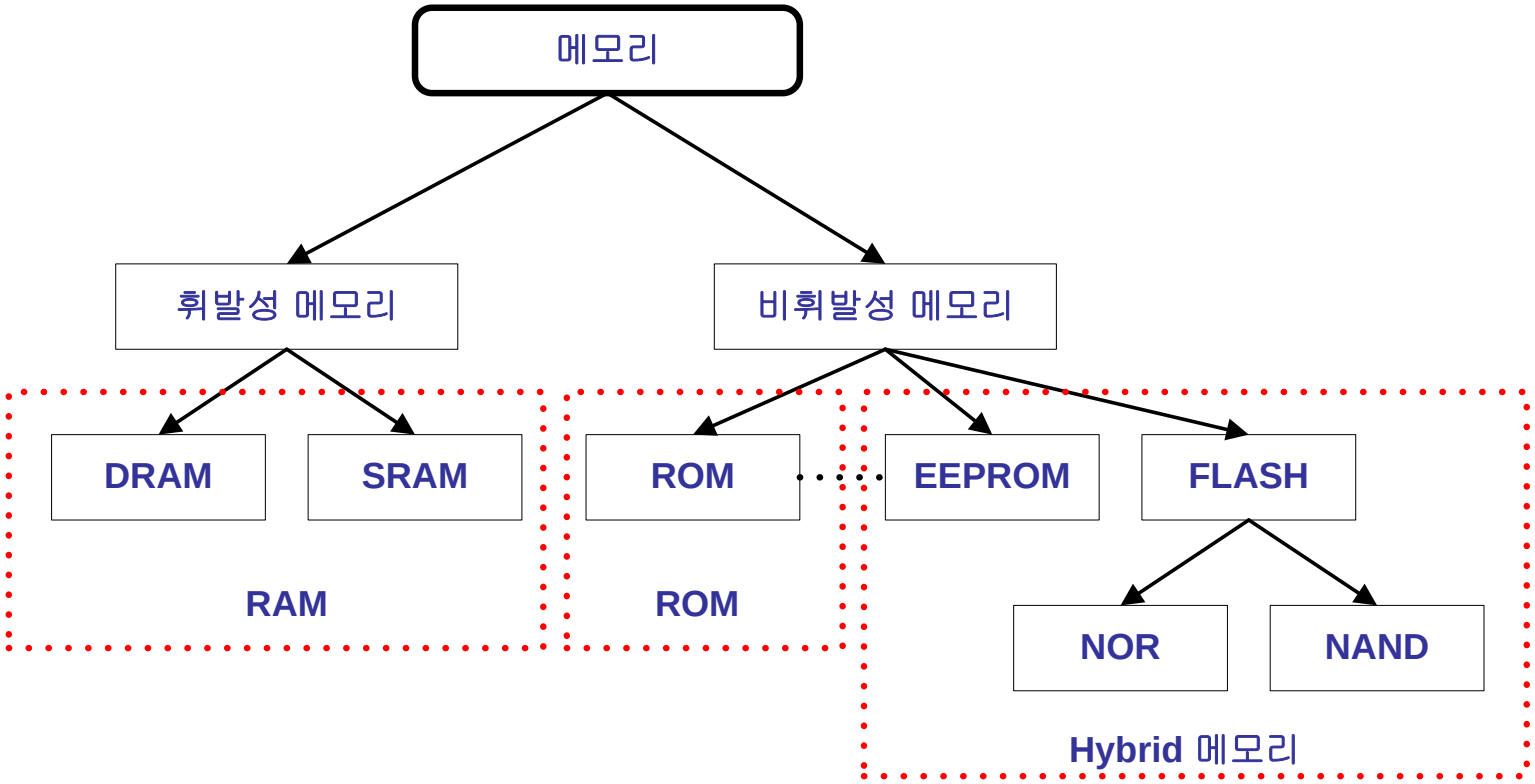
구분	**CISC**	**RISC**
역사	**1960** 년대	**1950** 년대
명령어	복잡	단순
특징	메모리 참조 연산	**Load/Store** 구조
성능	낮다	높다

목 차

메모리 장치

- 메모리 장치의 용도
  - 프로그램과 데이터를 저장하기 위한 공간
- 메모리 장치의 종류
  - 주 기억 장치 (main memory)
    - 프로그램이 실행되는 동안 프로그램과 데이터 저장
    - DRAM이 많이 사용된다
  - 보조 기억 장치(secondary memory)
    - 주 기억장치보다 빈번하게 사용하지 않는 프로그램과 데이터 저장
    - HDD, SD, MMC 등이 사용된다
  - 캐시 (cache)
    - 주 기억장치의 접근 속도를 빠르게 하기 위해서 프로세서 주변에 배치된 소 용량의 메모리
    - SRAM이 사용된다.

임베디드 시스템에서 사용되는 메모리



메모리장치의 종류

구 분			속 도	가 격	용 도	특 징
휘발성 (Volatile Memory)	SRAM (Static)		수ns,고속	비싸다	캐시 등	
	DRAM (Dynamic)		수십 ns	저렴	주기억 장치	
비휘발성(Non-Volatile Memory)	EEPROM (Electrically Erasable)		수십 ns	비싸다	소용량 데이터나 프로그램 저장용	
	Flash	NAND	수십 ns	저렴	대용량 데이터 저장	블록 단위 읽기 쓰기
		NOR	수십 ns	비싸다	프로그램 저장 데이터 저장	

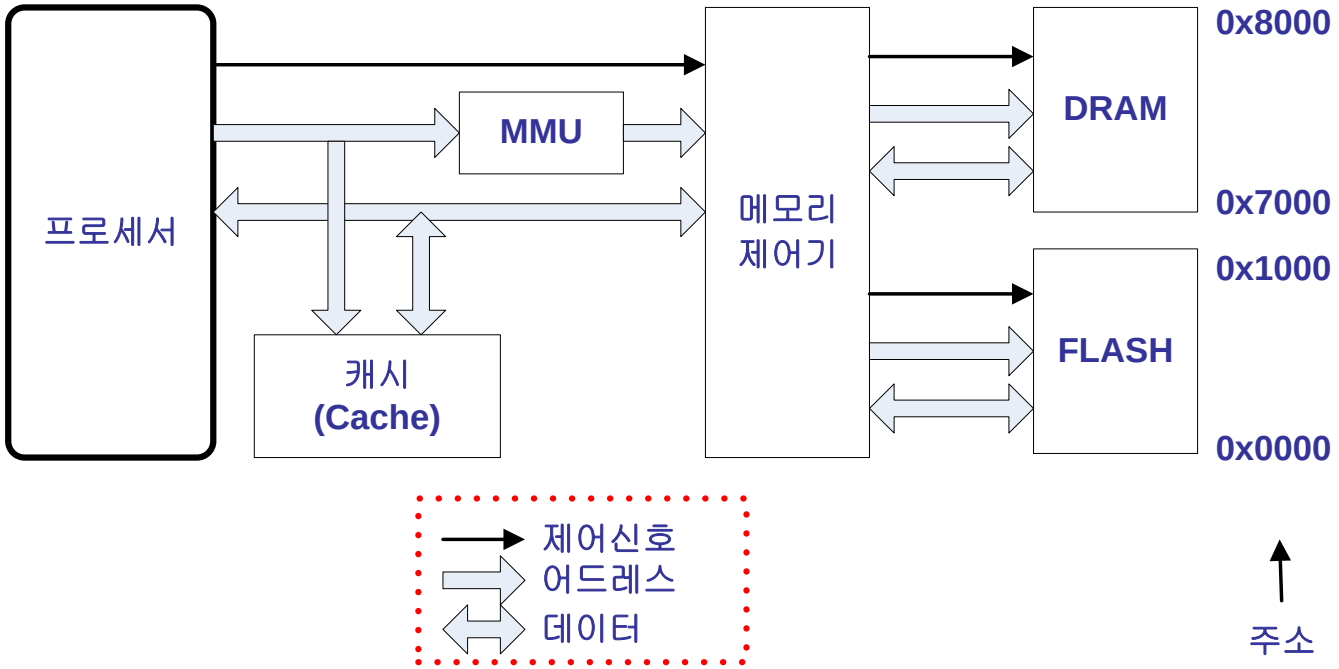
RAM (Random Access Memory)

- 전원이 인가되는 상태에서만 데이터를 유지
- 크게 **2** 종류의 **RAM**이 존재
  - Static RAM (SRAM) and Dynamic RAM (DRAM)
  - 비트가 저장되는 방법 상에 차이점이 존재
  - Static RAM
    - 빠르다 (active drive)
    - Less dense (4-6 transistors/bit)
    - Stable (holds value as long as power applied)
  - Dynamic RAM
    - SRAM에 비해 늦다
    - High density (1 transistor/bit)
    - Unstable (refresh가 필요)
  - 기타
    - SDRAM, Video RAM, FERAM 등

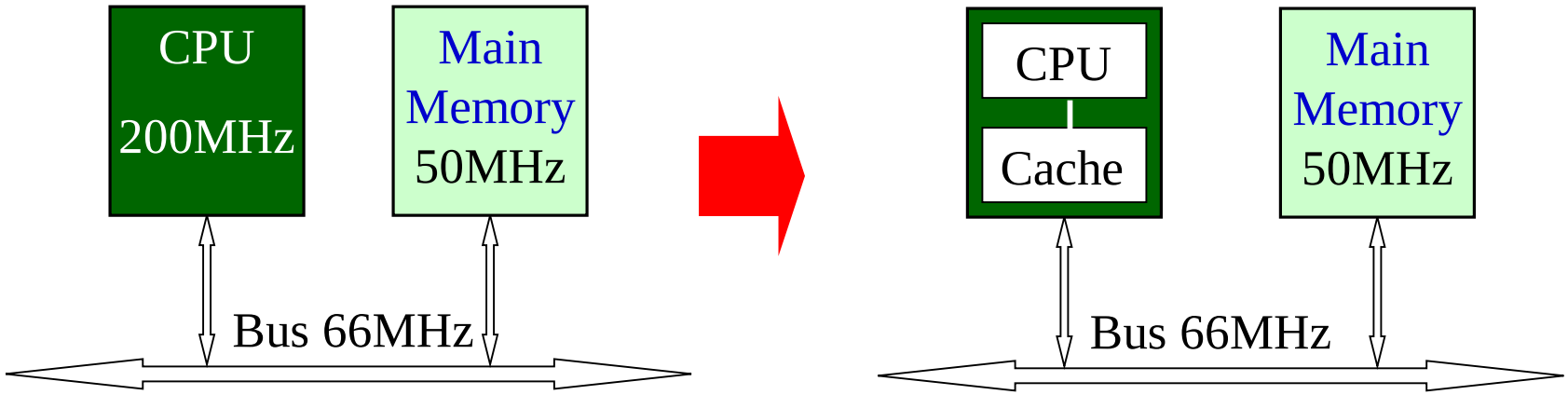
DRAM과 SRAM

구분	DRAM	SRAM
리플래시와 충전	주기적	필요 없다
억세스 주기	느리다	빠르다
회로구조	단순하다	복잡하다
칩크기	작다	크다
가격	싸다	비싸다
용도	일반 메모리	캐시 메모리

메모리 시스템 구조


프로세서
MMU
캐시
(Cache)
메모리
제어기
DRAM
FLASH
0x8000
0x7000
0x1000
0x0000
제어신호
어드레스
데이터
주소

Cache 메모리 시스템



고속의 CPU가 버스 및 메모리 속도에 의존적이며 늦다

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

Cache의 읽기 동작

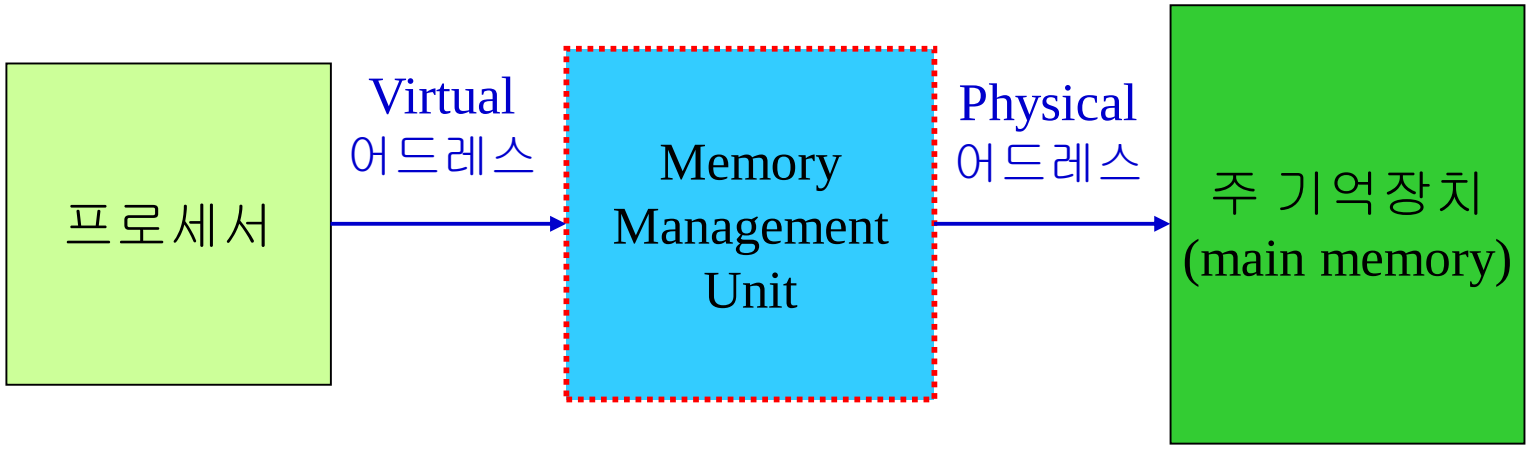
- **Cache의 목적**
  - 프로세서가 읽고자 하는 명령이나 데이터를 최대한 빨리 프로세스에 전달하는데 목적이 있다.
- **Cache의 성능**
  - CPU가 읽고자 하는 명령이나 데이터가 Cache 내에 존재(Cache Hit) 해야 할 회수가 많아야 Cache의 성능이 우수하다.
  - CPU가 데이터나 명령을 읽고자 하는데 Cache 내에 원하는 명령이나 데이터가 없으면(Cache Miss), Cache 제어기는 시스템 메모리 장치에서 line 크기 만큼 명령이나 데이터를 읽어 Cache 메모리에 저장(Line Fill)

Cache의 쓰기 동작

- **Write Through**
  - CPU가 특정 주소에 데이터를 write하는 경우, 해당하는 데이터가 Cache 메모리에 있을 때, Cache 메모리와 외부 메모리에 동시에 쓰기 동작을 한다.
- **Write Back**
  - CPU가 특정 주소에 데이터를 write하는 경우, 해당하는 데이터가 Cache 메모리에 있을 때, Cache 메모리에만 쓰기 동작을 하고, 외부 메모리는 나중에 기록 된다.

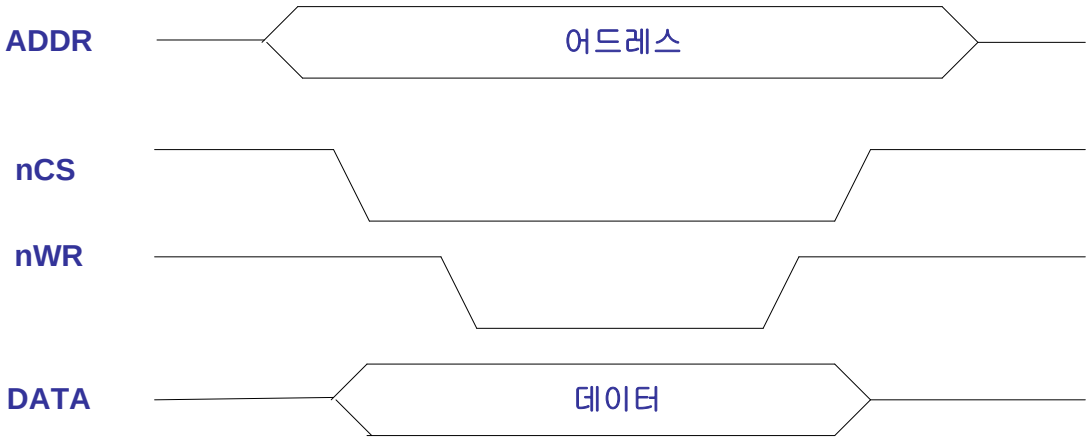
MMU (Memory Management Units)

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



메모리 제어기

- 메모리 제어기**(Memory Controller)** 주요 기능
  - 어드레스 디코드(Address Decode) 기능
  - 메모리 제어 신호 구동
- 메모리 제어기의 동작
  - 입력되는 주소 정보에 따라 어떤 메모리 장치를 억세스 할 것인지를 결정(Address Decode)하고 각각의 메모리 장치의 특징에 따라 제어 신호를 구동한다.
- 메모리 제어 신호 구동 예



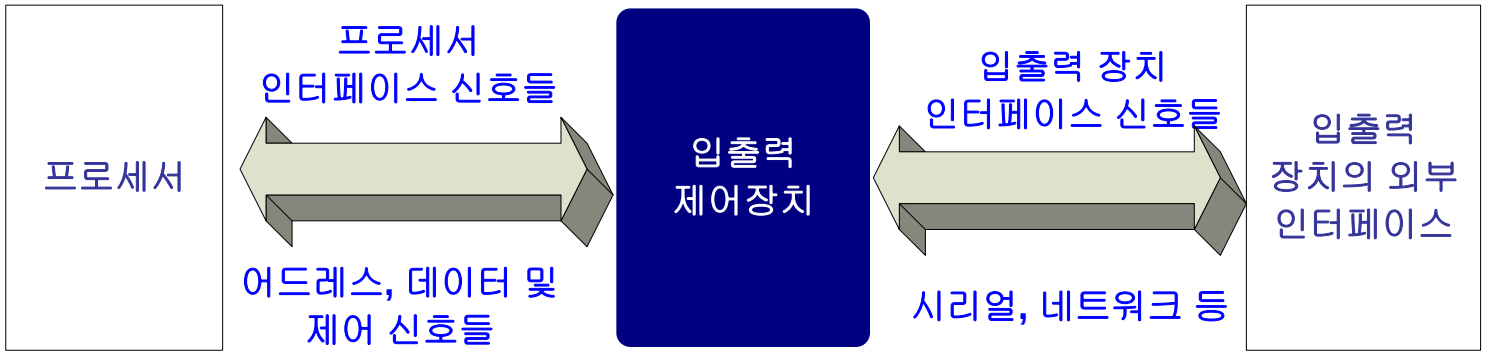
어드레스와 어드레스 버스

- 어드레스
  - 메모리 장치의 어느 영역을 읽어 올 것인가를 구분
- 어드레스 버스
  - 메모리 장치에 접근할 때 영역을 구분하기 위해서 사용하는 신호
  - 1K 바이트 메모리를 구분하기 위해서는 10개의 어드레스 신호선 필요
    - 1K 바이트 = 2^{10} = 1024
    - A0, A1, A2, A3 … A9 등으로 표기
  - 1M 바이트 메모리를 구분하기 위해서는 20개의 어드레스 신호선 필요
    - 1M 바이트 = 2^{20}

목 차

입출력 장치

- 프로세서와 정보를 교환하는 장치
- 디지털 신호 또는 아날로그 신호를 포함 한다.
- 프로세서와는 메모리 장치와 같이 디지털 신호인 어드레스, 데이터 및 제어 신호를 통해서 연결된다.



입출력 장치 제어

- 입출력 장치를 제어하기 위해서는 어드레스 할당이 필요하고 데이터를 교환하기 위한 데이타 버스와 제어 신호 사용
- **표준 I/O 맵 방식(I/O-mapped peripheral)**
  - 전용의 입출력 장치 주소 공간을 할당하여 사용
  - 인텔의 x86 CPU 계열이 대표적
- **메모리 맵 방식(Memory mapped peripheral)**
  - 메모리 주소 공간의 일부를 활용하여 사용
  - 대부분의 임베디드 프로세서에서 사용 됨

메모리 맵 방식과 I/O 맵 방식

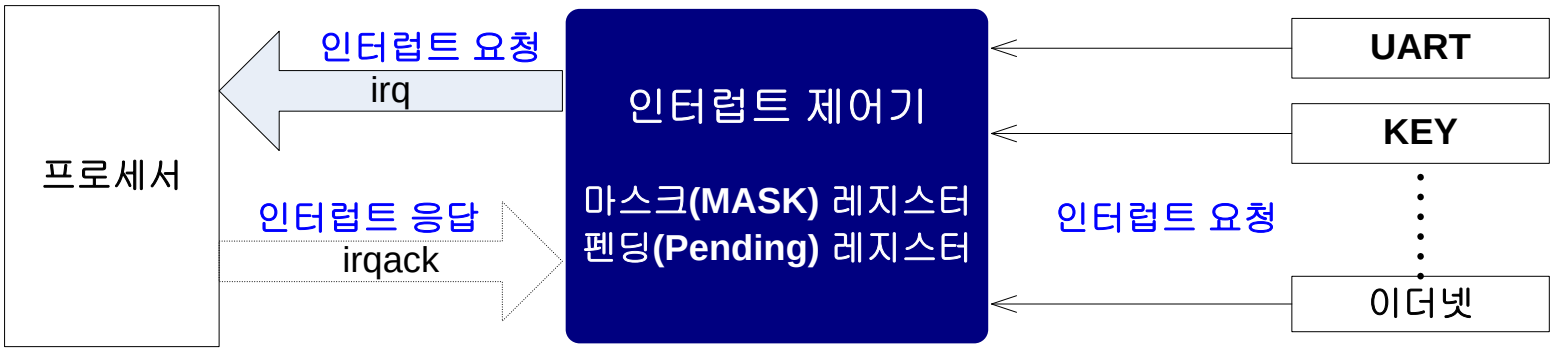
구분	메모리 맵 방식	I/O 맵 방식
대표적인 프로세서	**ARM, MIPS, PowerPC, M68K …**	**x86** 계열
입출력 장치의 영역	메모리의 일부를 **I/O** 장치로 사용	메모리 영역과는 별도의 **I/O** 번지 영역이 존재
명령어	메모리와 **I/O** 장치 모두 메모리 동작 명령으로 억세스 하며, 각 영역의 구분은 어드레스로 한다.	메모리 억세스 명령과 **I/O** 억세스 명령**(in/out)**이 구분
하드웨어	어드레스를 해석하는 디코더 회로에 따라 메모리 혹은 **I/O** 장치가 선택	메모리 번지와 **I/O** 번지를 구분하는 신호가 존재.
주의 사항	- **I/O** 영역은 **Non-cacheable**로 설정해야 한다 - **I/O** 영역 변수는 **volatile type**으로 선언해야 한다.	

입출력 장치의 자원 관리

- **폴링 방식**
  - 한 프로그램이나 장치에서 다른 프로그램이나 장치들이 어떤 상태에 있는지를 지속적으로 검사하는 전송 제어 방식
  - 입출력 장치의 접속 여부 및 데이터 전송의 요청과 종료를 검사한다.
- **인터럽트**
  - 프로세서는 일련의 처리를 수행하고, 주변장치에서 입출력 처리 동작이 필요한 경우 프로세서에게 진행 중 이던 명령을 멈추고 새로운 동작을 할 수 있도록 한다.
  - 프로세서는 한번에 한 개의 명령만을 수행할 수 있다.
  - 인터럽트를 이용하면 멀티태스킹을 지원할 수 있도록 한다.
  - 사용자는 모든 작업이 동시에 수행되는 것처럼 보이게 동작한다.
- **DMA 방식**
  - DMA(Direct Memory Access) 방식은 CPU의 개입 없이 입출력 장치와 기억장치 사이에 데이터를 전송하는 방식

인터럽트 인터페이스

- 인터럽트 제어기
  - 입출력 장치에서 발생되는 인터럽트의 요청을 제어 한다.
  - 하드웨어에 따라 인터럽트 응답을 위한 신호도 제공된다.



인터럽트의 발생

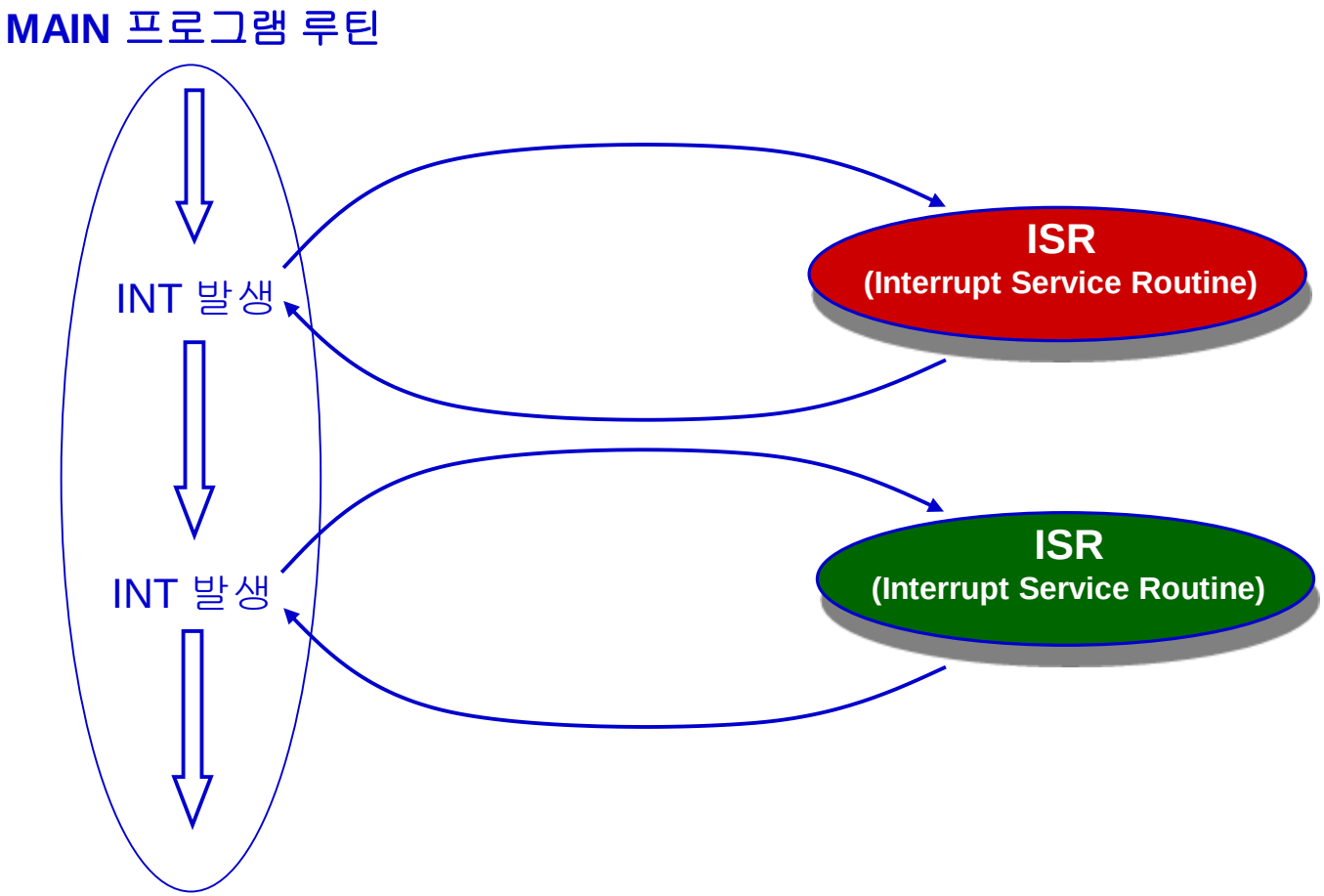
❑ 인터럽트 요청 (Interrupt Request)

- ❖ 외부 장치에서 입출력 동작에 대한 처리를 프로세서에 요청
- ❖ 인터럽트의 발생은 하드웨어적으로 이루어 진다. 따라서 인터럽트가 발생하면 프로세서가 스스로 프로그램의 개입 없이 일련의 동작을 수행해야 한다.
- ❖ 인터럽트 Vector
  - ➢ 인터럽트 서비스 루틴을 처리하기위한 명령 또는 위치가 저장된 메모리 공간
- ❖ 인터럽트의 요청에 따라 프로세서는 정해진 절차에 의하여 발생 된 인터럽트의 처리 여부를 결정하고 인터럽트를 서비스하는 절차(ISR : Interrupt Service Routine)를 수행한다.

❑ 인터럽트 발생의 예

- ❖ 시리얼로 데이터 입력 완료
- ❖ 시리얼 데이터 전송 준비 완료, 또는 전송 에러 발생
- ❖ 이더넷 데이터 수신 완료, 이더넷 전송에러 등

인터럽트 발생에 의한 프로세서 흐름제어

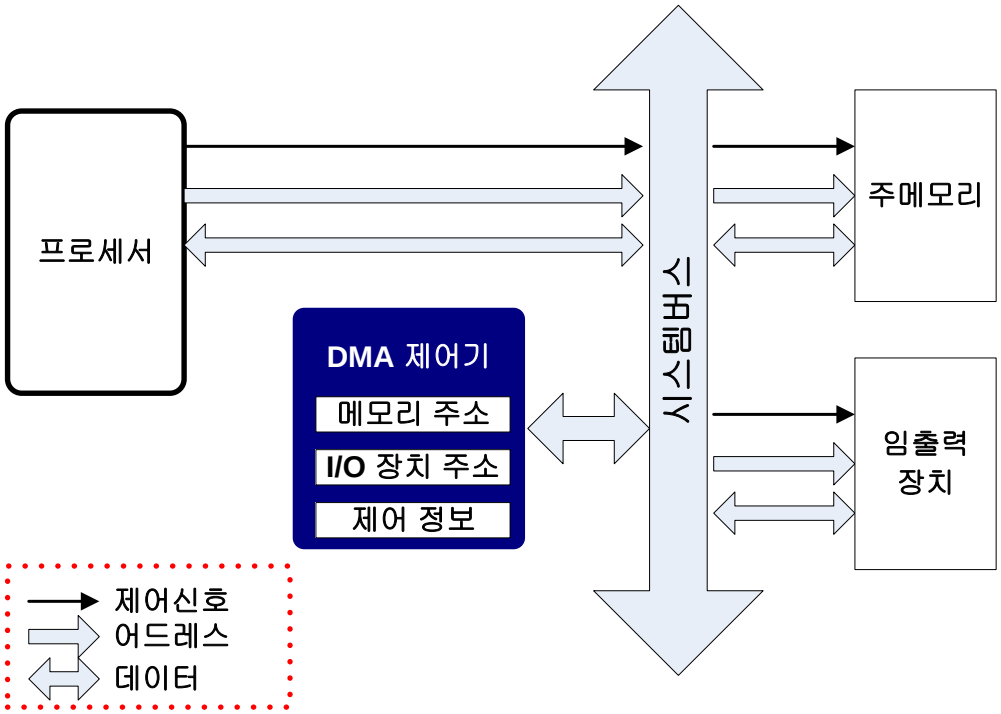


인터럽트 벡터(Vector)

- 인터럽트 벡터
  - 인터럽트 서비스 루틴을 처리하기 위한 명령 또는 위치가 저장된 메모리 공간
- 인터럽트 벡터 주소 지정 방식
  - 고정 인터럽트(Fixed interrupt)
    - 인터럽트가 발생시 처리할 어드레스가 지정되어 변경이 않 된다.
    - 지정된 어드레스에 인터럽트를 처리하기위한 명령 또는 위치가 저장되어 있다.
  - 벡터 인터럽트(Vectored interrupt)
    - 일반적인 마이크로프로세서 장치에서 여러 개의 주변 장치가 시스템 버스에 연결되어 사용되는 경우
    - 주변장치가 인터럽트를 처리할 주소를 제공

DMA 방식

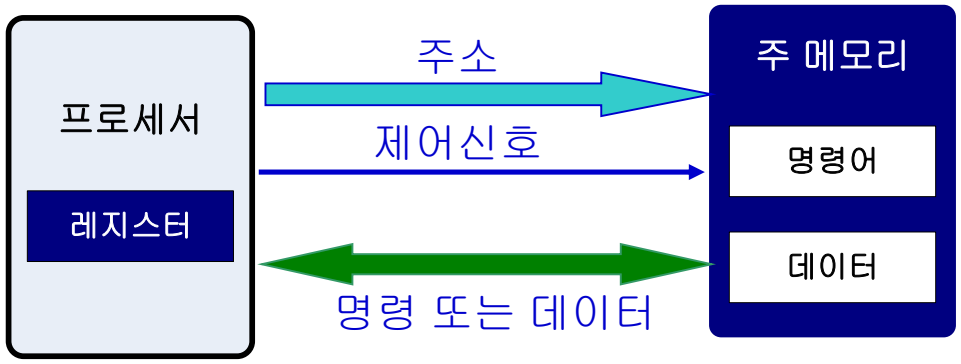
- **DMA**
  - 프로세서 개입 없이 입출력 장치와 기억 장치 사이의 데이터를 전송하는 방식으로 다른 버스 마스터가 주기억 장치를 억세스 하지 않는 동안에 데이터를 교환



목 차

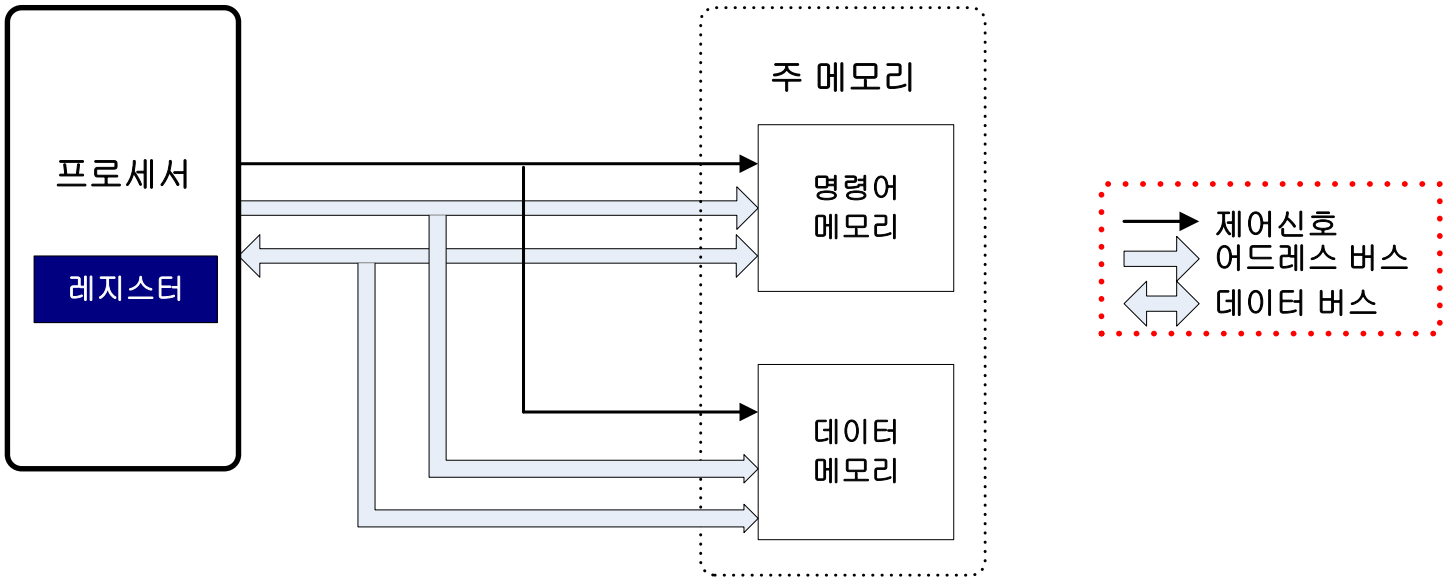
버스 (BUS)

- **버스(BUS)란 ?**
  - 컴퓨팅 시스템의 각 모듈에서 발생한 신호를 공유해서 사용할 수 있도록 만든 신호의 집합
  - 구동 주체(CPU 등)에 의해서 해당 소자에 데이터를 읽거나 쓸 수 있도록 구성된다.
  - 어드레스 버스(address bus), 제어버스(control bus), 그리고 데이터 버스(data bus)로 구성된다.



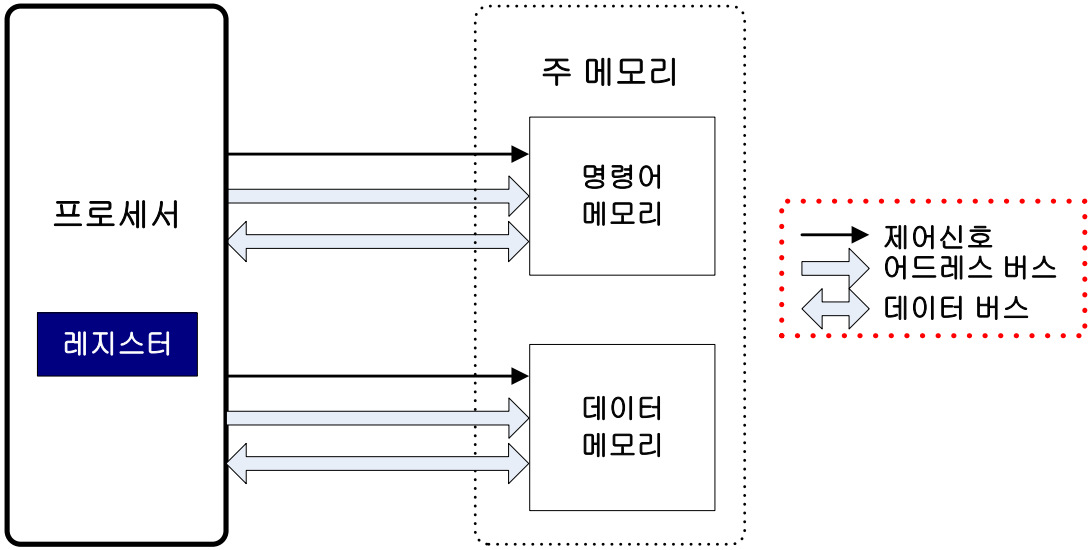
폰 노이만 아키텍쳐

- **폰 노이만(Von-Neumann) 아키텍쳐**
  - 명령어와 데이터를 위한 메모리 인터페이스가 하나이다.
  - 명령어를 읽을 때 데이터를 읽거나 쓸 수 없다.
  - IBM 계열 PC(개인용 PC), ARM7 등



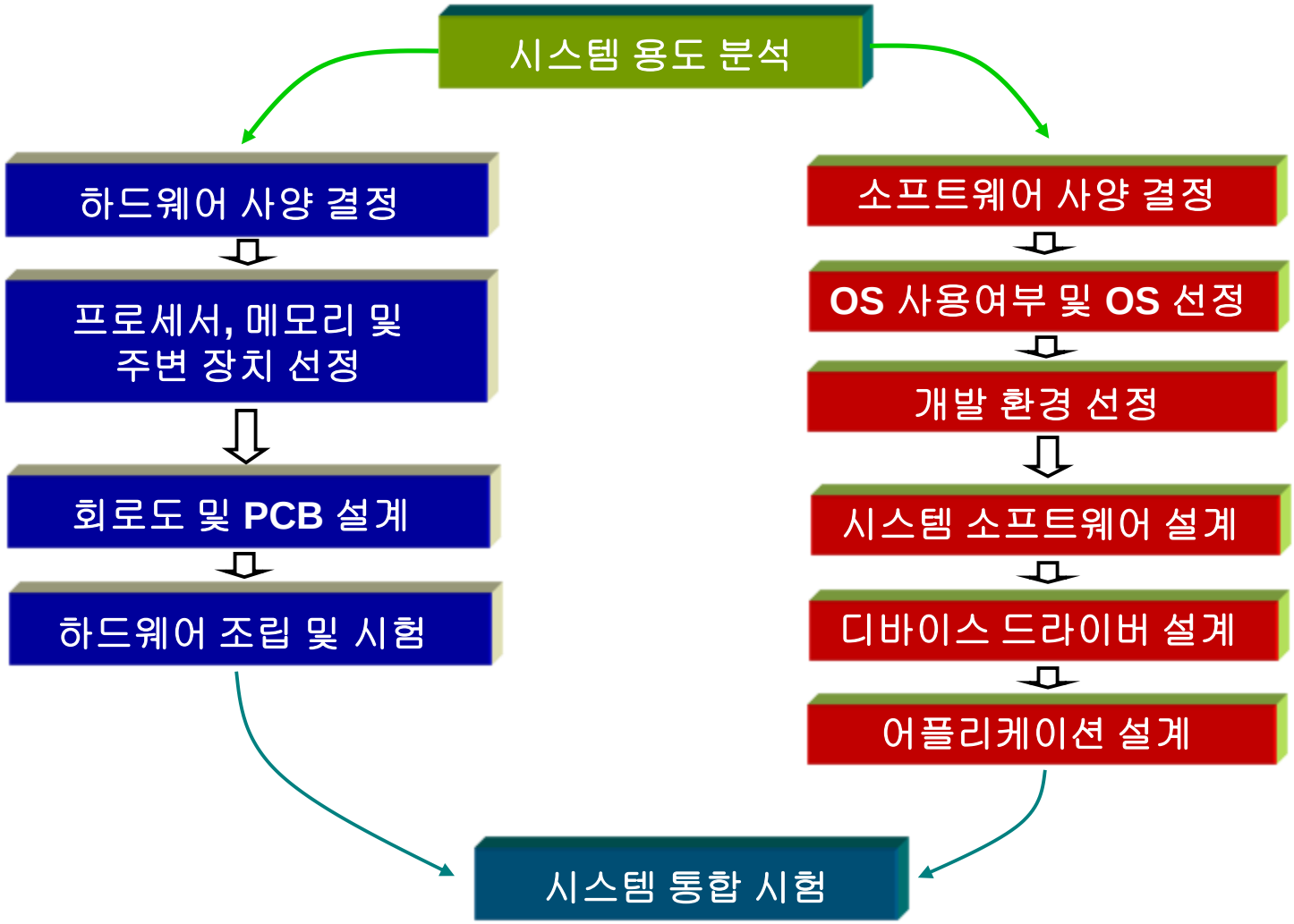
하버드 아키텍쳐

- 하버드(Havard) 아키텍쳐
  - 명령어를 위한 메모리 인터페이스와 데이터를 위한 메모리 인터페이스가 분리되어 있다.
  - 명령어를 읽을 때 데이터를 읽거나 쓸 수 있어 성능이 우수하다.
  - 버스 시스템이 복잡하여 설계가 복잡하다
  - ARM9, ARM10, XScale 등



목 차

임베디드 시스템 설계 절차



목 차

프로세서 선정

- 프로세서 선정 시 고려사항
  - 개발 하려는 제품의 특성
  - 개발의 용이성, 가격, 안정성
  - 제조회사의 지원 능력
- 프로세서 선정

CPU	특 징
ARM	간단한 명령어 사용하고, 개발 환경이 간단하다. 전력 소모가 작아서 휴대폰이나 PDA같은 휴대 단말기에 많이 사용
MIPS	고속의 처리 능력, 고속 네트워크 장비등에 많이 사용
i386	오랜 기간의 사용으로 안정성 확보, PC와 동일한 개발 환경 구성
Power PC	강력한 네트워크 기능을 포함한 SoC

메모리 선정

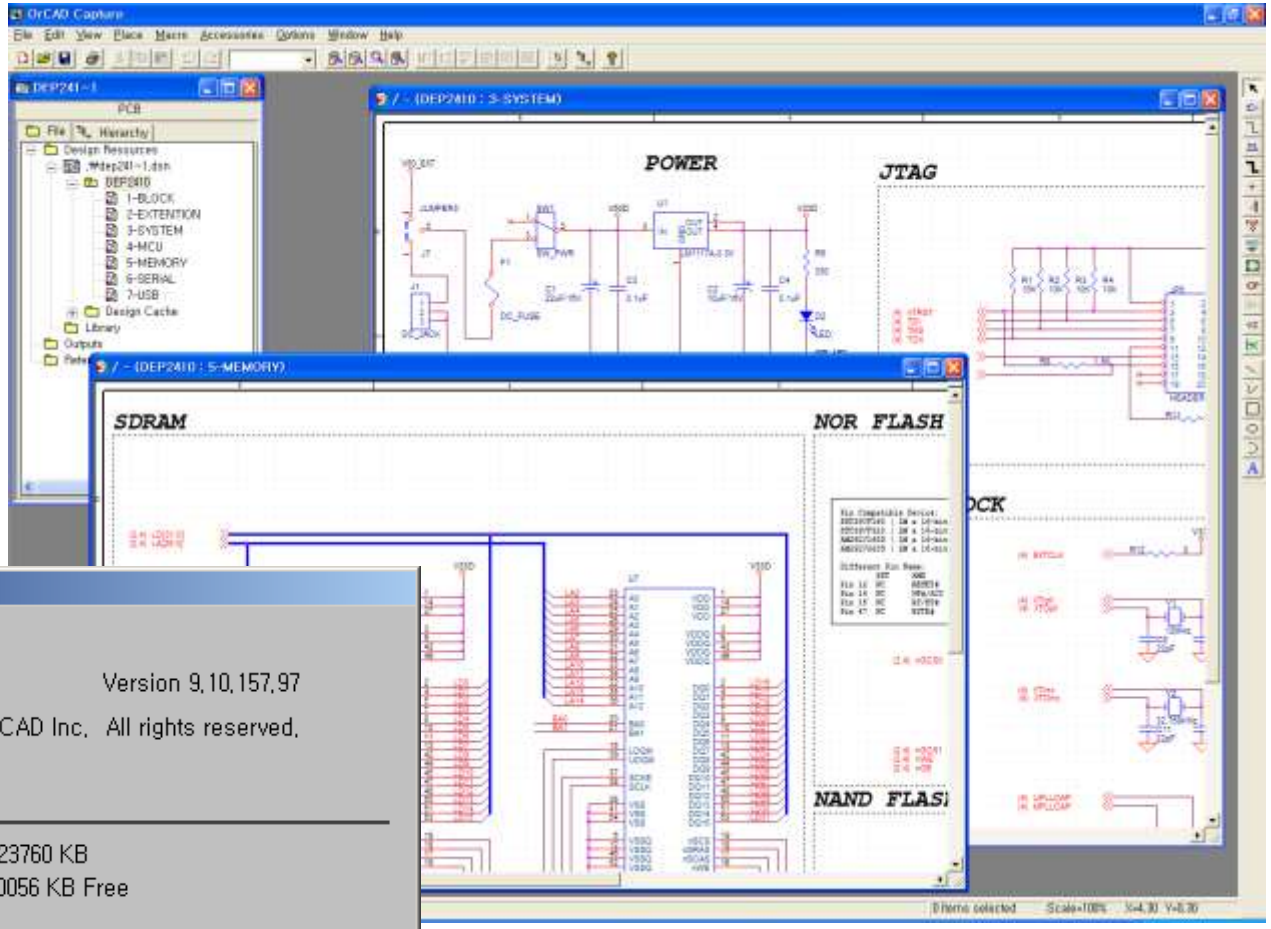
- **코드 및 데이터 저장용 메모리 (ROM)**
  - 저장할 코드 및 데이터의 크기, 속도 등에 따라 선정
  - EEPROM
    - 소용량, 시스템 구동 중 코드나 데이터의 업데이트 거의 없는 경우
  - NOR Flash
    - 저용량, 시스템 구동 중 코드나 데이터의 업데이트가 발생 하는 경우
  - NAND Flash
    - 대용량, 시스템 구동 중 코드나 데이터의 업데이트가 빈번하게 발생하고 대용량의 데이터를 저장하는 경우에 사용
- **프로그램 구동 중에 사용되는 메모리 (RAM)**
  - SDRAM
    - 시스템의 성능을 좌우하므로 빨라야 한다.
    - 시스템의 기능에 따라 가격 대비 적합한 크기의 메모리를 선정한다.

주변 장치 선정

- 시스템의 용도 및 특성에 맞는 주변 장치 선정
  - LCD, Touch
  - LAN, 무선 LAN
- **SoC 선정**
  - SoC는 일반적인 제품을 구성하는데 필요한 모든 주변 장치를 가지고 있다.
  - PDA 전용 SoC
    - PXA255, S3C2410, S3C24A0, …
  - IP 공유기 용 SoC
    - KS8695, RTL8650B, S3C2510, …

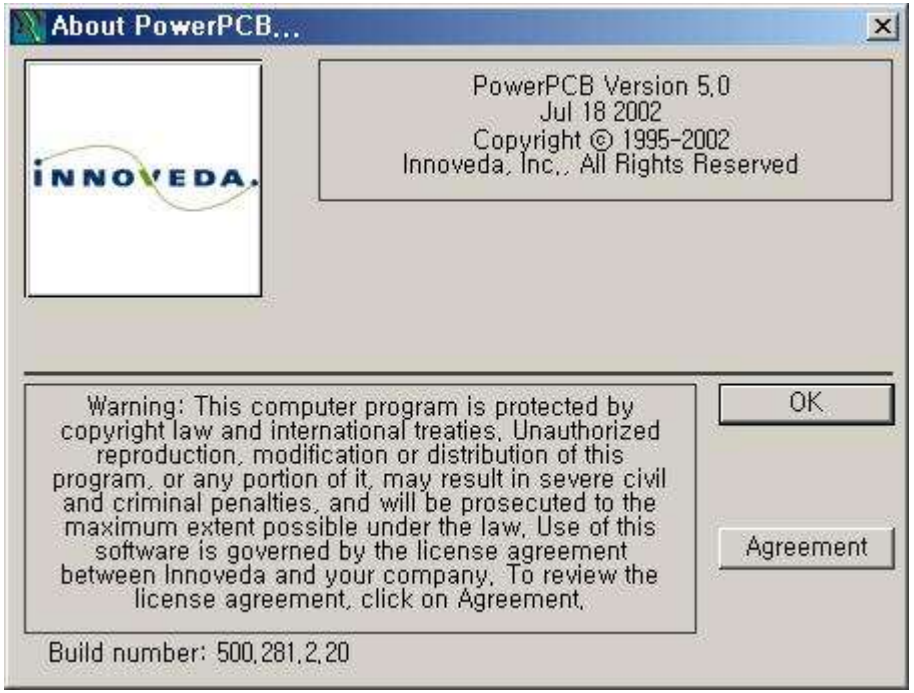
회로도 설계

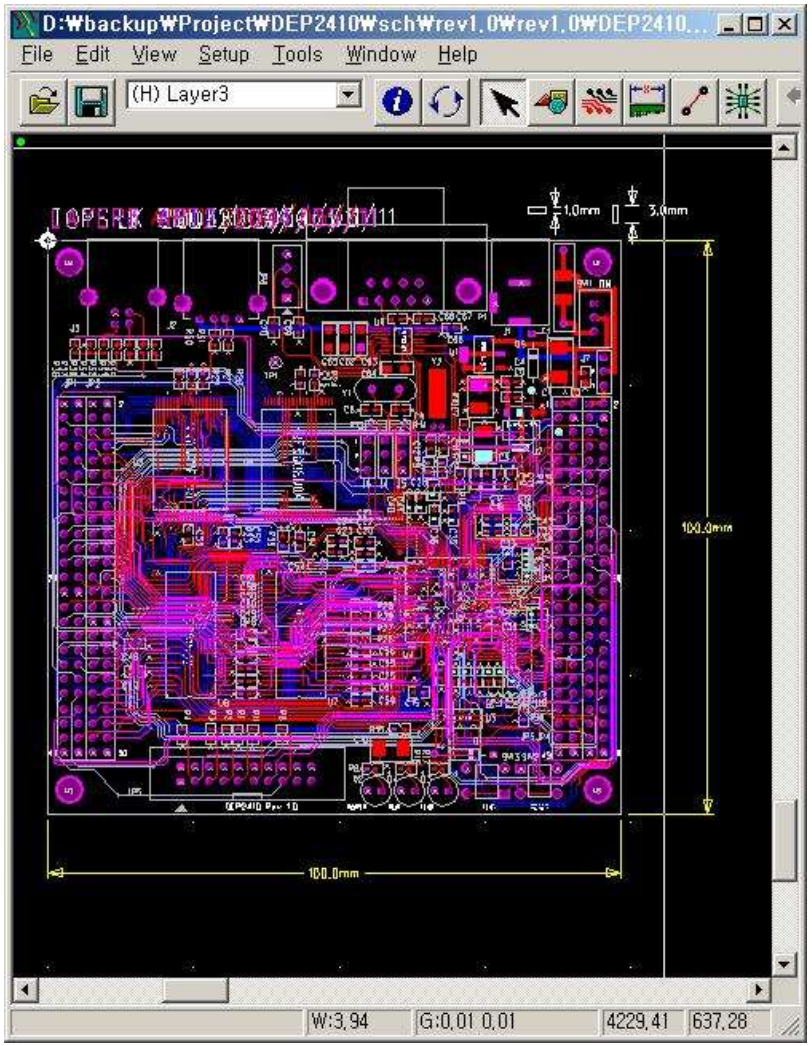
- **OrCAD**를 이용한 회로도 설계



PCB 설계

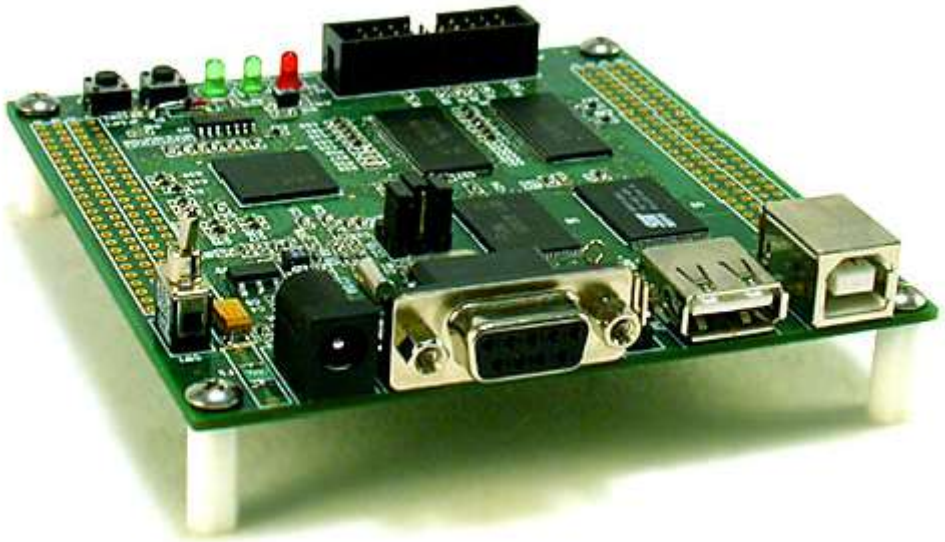
- **PowerPCB를 이용한 PCB 설계**





하드웨어 조립 및 시험

- **PCB**가 완료되면 각 부품을 조립하고 시험한다.



목 차

소프트웨어 사양 결정

- 시스템의 사양 및 성능에 따른 소프트웨어 선정
- **OS** 사용 여부 결정
  - OS의 필요성 결정
  - Real-time의 필요성 및 시스템 메모리의 크기 등에 따른 OS 선정
- 소프트웨어 개발
  - 개발 시간, 난이도 및 비용에 따라 자체 개발 또는 외주 개발 결정
- 라이슨스**(License)** 조건
  - 소프트웨어의 사용 권한 및 제한 사항 확인
  - MPEG, MP3 등

OS 사용 여부 결정

- 임베디드 시스템에서의 운영체제
  - 시스템의 규모가 커짐에 따른 멀티 태스킹(Multi Tasking) 기능 요구
  - 네트워크나 멀티미디어 기능이 시스템의 기본 요소가 됨
    - Networking, GUI, Audio, Video
  - 리얼타임의 필요성이 부각됨
  - 지능성이 부가되고, 기능이 많아지고, 복잡해짐
  - 순차적인 프로그램 작성이 불가능하여 운영체제가 도입됨
- 임베디드 운영체제
  - 상용 RTOS(Real-Time OS)
  - 윈도우 CE
  - 임베디드 Linux
  - 임베디드 JAVA

임베디드 OS 선정

- **상용 RTOS : Hard RealTime/Multi-thread/ Preemptive**
  - pSOS, VxWorks, VRTX 등 다수
  - 일반 운영체제와 거의 같은 기능을 수행
  - 시간 제약성, 신뢰성 등을 일반 운영체제 보다 중요시 함
  - 일반적으로 한가지 목적에 최적화 되어있음
- **임베디드 OS : Soft RealTime/Multi-process/ non preemptive**
  - Windows CE
  - 임베디드 리눅스
  - 임베디드 자바
- **최근 동향**
  - 임베디드 OS 세계시장 :
    - WinCE, 임베디드 리눅스가 기존의 RTOS 보다 시장 점유율이 높아지는 추세
- **OS 선정**
  - 시스템의 특성에 적합한 OS 선정

커널 포팅

❑ 커널 포팅 이란 ?

- ❖ 선정된 OS 커널이 개발하려는 시스템에서 동작 할 수 있도록, 프로세서, 메모리, 트랩(Trap or Exception), 인터럽트와 타이머 등을 맞추어 주는 작업
- ❖ OS의 Scheduler를 정상적으로 동작 할 수 있도록 한다.

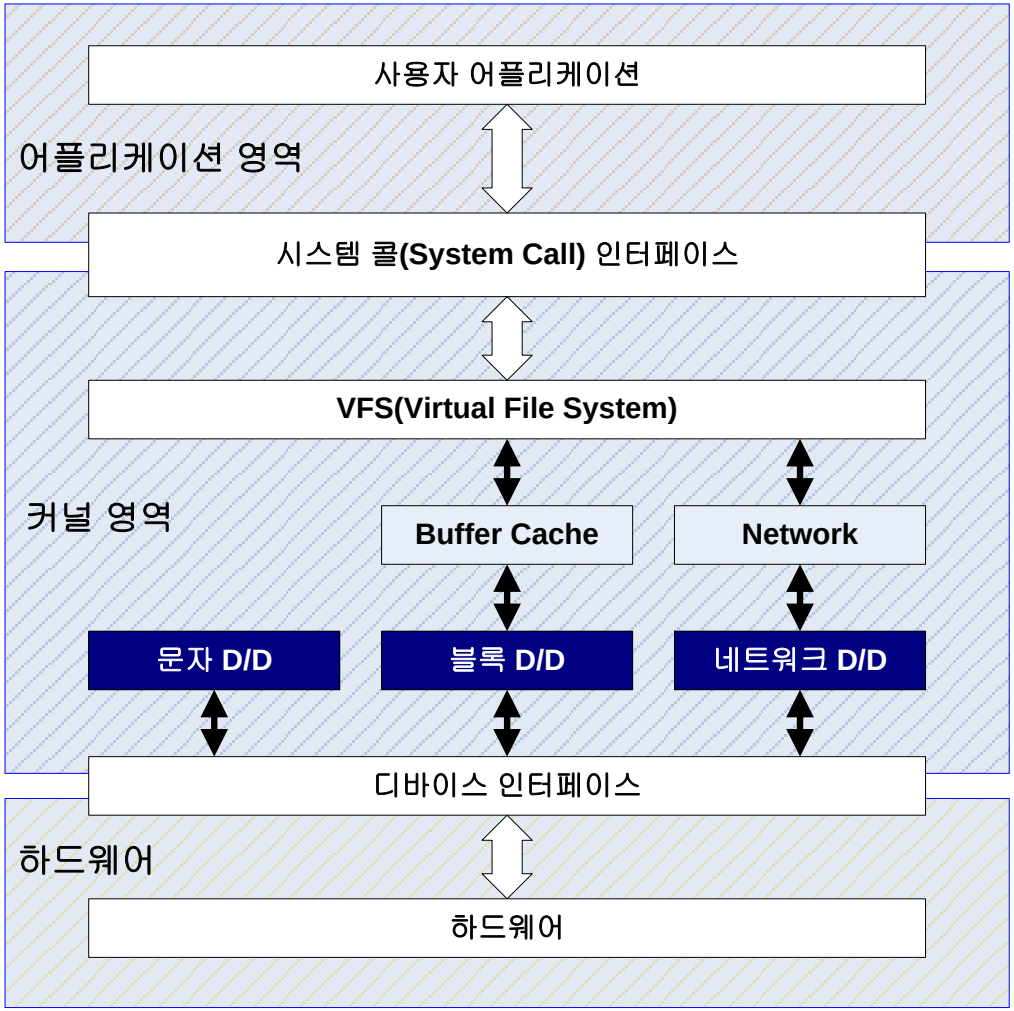
❑ 커널 포팅

- ❖ 커널을 포팅하는 개발자는 하드웨어 및 소프트웨어에 대하여 알고 있어야 한다.
- ❖ 근래에는 반도체 또는 SoC 개발 회사에서 커널을 포팅하여 제공하는 추세

디바이스 드라이버 포팅

- **디바이스(Device)**
  - 네트워크 어댑터, LCD 디스플레이, Audio, 터미널, 키보드, 하드디스크, 플로피디스크, 프린터 등과 같은 입출력 장치들을 말함
  - 디바이스의 구동에 필요한 프로그램, 즉 디바이스 드라이버가 필수적으로 요구됨
- **디바이스 드라이버**
  - 실제 장치 부분을 추상화 시켜 사용자 프로그램이 정형화된 인터페이스를 통해 디바이스를 접근할 수 있도록 해주는 프로그램
  - 디바이스 관리에 필요한 정형화된 인터페이스 구현에 요구되는 함수와 자료구조의 집합체
  - 응용프로그램이 하드웨어를 제어할 수 있도록 인터페이스 제공
  - 하드웨어와 독립적인 프로그램 작성을 가능하게 함

디바이스 드라이버 구조

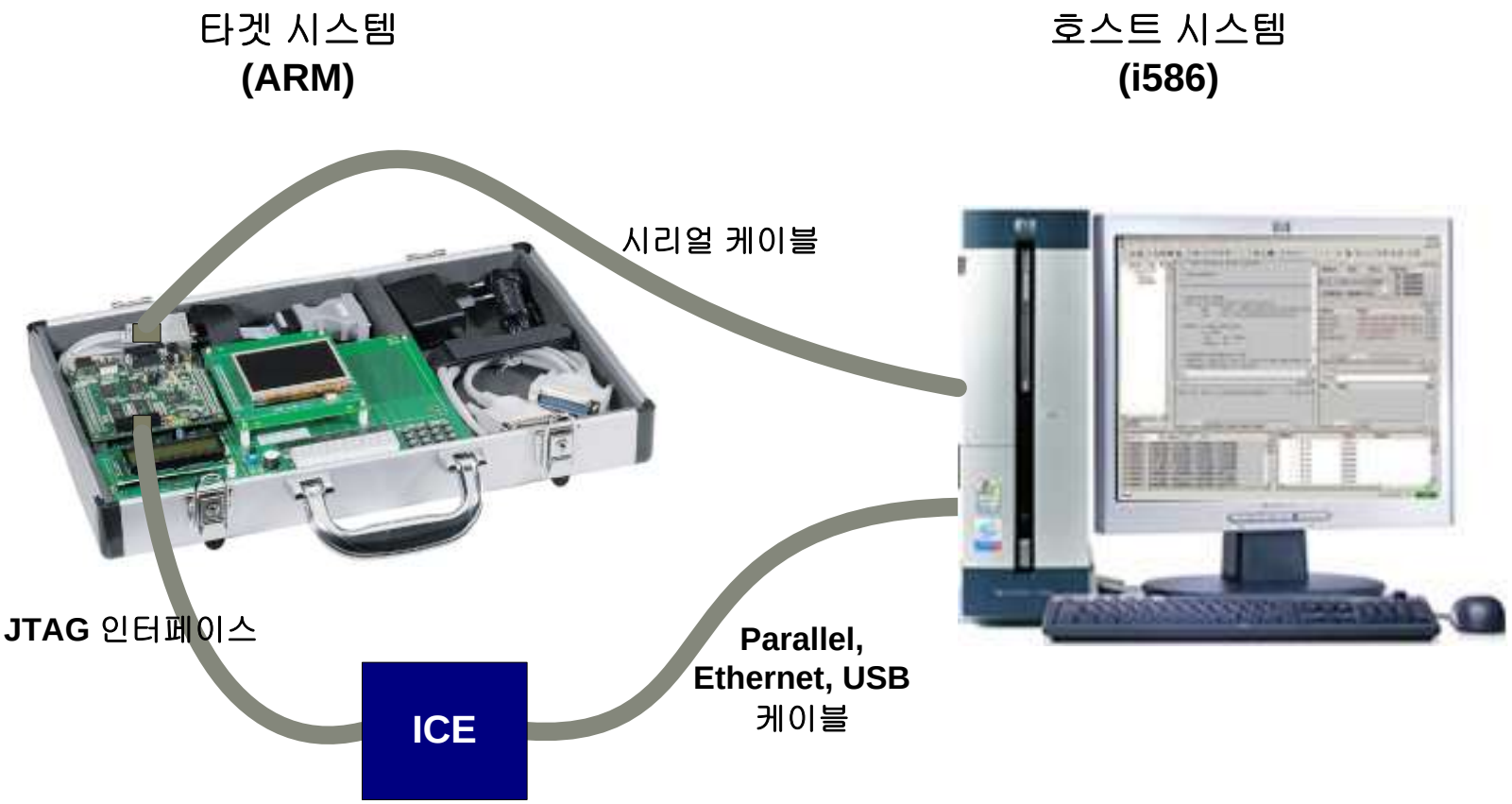

사용자 어플리케이션
어플리케이션 영역
시스템 콜(System Call) 인터페이스
VFS(Virtual File System)
커널 영역
Buffer Cache
Network
문자 D/D
블록 D/D
네트워크 D/D
디바이스 인터페이스
하드웨어
하드웨어

어플리케이션 개발

- **어플리케이션(응용 프로그램)**
  - 시스템(하드웨어와 OS)를 이용해서 어떠한 작업을 하는 것
  - 워드, 게임 등의 모든 응용 프로그램
- **임베디드 시스템에서의 어플리케이션**
  - 특정한 작업을 처리하기 위해서 작성 된다.
  - 근래에는 컨버전스 제품으로 변모, 다양한 어플리케이션 탑재

목 차

임베디드 소프트웨어 개발 환경



윈도우 기반의 호스트 시스템 설정

- **ARM 명령어 실습 및 독립형 프로그램 작성할 때 사용 가능**
- **MinGW**
  - 윈도우 환경에서 사용가능한 GNU 툴
- **MSYS**
  - 윈도우 환경에서 POSIX/Bourne 쉘 환경 제공
  - Makefile과 각종 스크립트를 수행 할 수 있도록 하는 툴
- **ARM용 교차 개발 툴**
  - Win32에서 임베디드 ARM 프로세서용으로 코드를 생성하는 GNU 툴
  - 어셈블러, 링커, 컴파일러 등 다양한 툴을 포함
- **OCD Commander**
  - 위글러와 호환되는 JTAG 동글과 함께 ELF 형식의 프로그램 다운로드, 싱글 스텝, 메모리 읽기 쓰기, 플래시 프로그래밍 등을 할 수 있는 툴
- **DevC++**
  - 무료로 사용 가능한 편집기
- 하이퍼 터미널
  - 윈도우에서 제공하는 시리얼 통신 에뮬레이터
  - 시리얼(UART)는 콘솔 및 디버깅용으로 사용 됨

리눅스 기반의 호스트 시스템 설정

- 임베디드 리눅스 개발 용
- 호스트 시스템 리눅스 설치
  - RedHat 리눅스 9.0 또는 이후 버전
  - 개발자 용으로 설치
- 교차 개발 툴 설치
  - ARM 용 교차 개발 툴 (arm-linux 타깃)
- 시리얼 통신 에뮬레이터 **minicom**
  - 리눅스에서 제공하는 시리얼 통신 에뮬레이터
  - 시리얼(UART)는 콘솔 및 디버깅용으로 사용 됨
- 네트워킹 설정
  - 고정 IP 주소를 사용하는 것이 유리
  - TFTP 서버 및 NFS 서버
- **USB** 로더
  - USB를 이용한 프로그램 탑재용으로 사용

질의 응답