

# 1. 임베디드 시스템의 구조

- ❖ 임베디드 시스템의 구조를 이해한다.
- ❖ 프로세서에 대하여 이해한다.
- ❖ 시스템 버스에 대하여 이해한다.
- ❖ 메모리 장치에 대하여 이해한다.
- ❖ 주변장치의 구조 및 특징을 이해한다.

1. 임베디드 시스템의 구성
2. 프로세서
3. 시스템 버스
4. 메모리 장치
5. 주변 장치

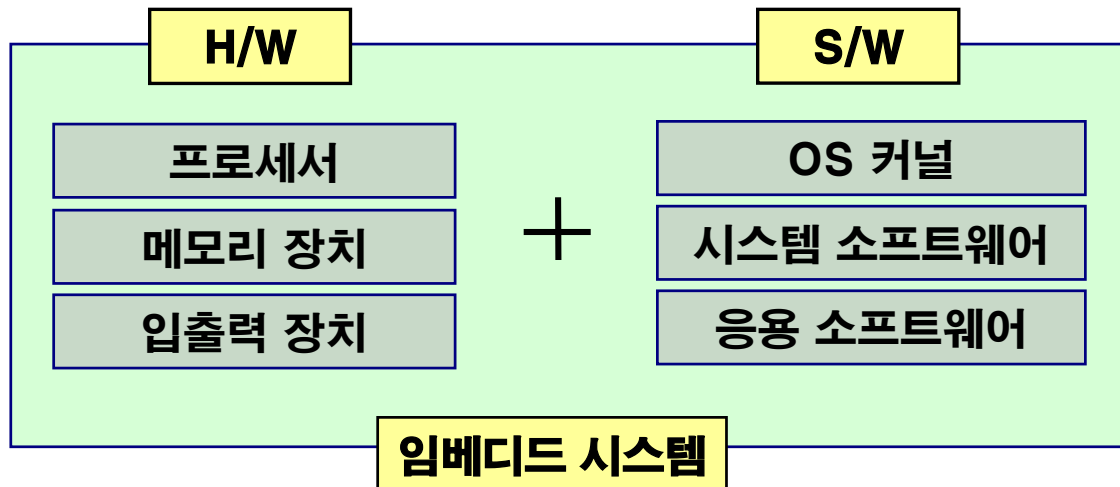
- ❖ 하드웨어와 소프트웨어가 조합되어 특정한 목적을 수행하는 시스템
- ❖ 특정한 기능을 수행하도록 마이크로 프로세서와 입출력 장치를 내장하며,  
이를 제어하기 위한 프로그램이 내장되어 있는 우리의 일상 생활에서 사용되는 각종 전자기기, 가전제품, 제어장치 등의 시스템을 말한다.

## ❖ 하드웨어

- 프로세서(컨트롤러), 메모리 장치(ROM, RAM),  
입출력 장치(네트워크 장치, 센서, 구동기 등)

## ❖ 소프트웨어

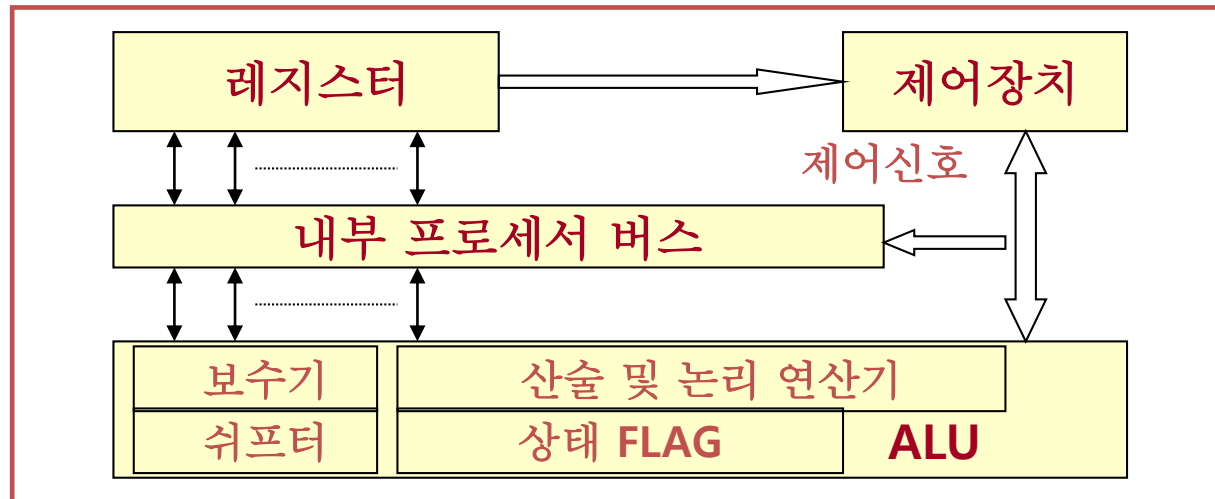
- 운영체제(OS), 시스템 S/W, 응용 S/W



1. 임베디드 시스템의 구성
- 2. 프로세서**
3. 시스템 버스
4. 메모리 장치
5. 주변 장치

- ❖ 프로세서 또는 CPU(Central Processor Unit) 는 디지털 시스템의 핵심 부분이다.
- ❖ 프로그램을 메모리 장치에서 읽어 연산처리, 비교처리, 데이터 전송, 편집, 변환, 테스트와 분기 등의 데이터 처리와, 각종 입출력 장치를 구동한다.
- ❖ 제어장치, 연산장치, 레지스터와 데이터 버스로 구성된다.
- ❖ 마이크로 프로세서(Micro-processor)는 한 개의 조그만 IC칩 속에 CPU의 모든 내용을 내장한 칩을 말한다.
- ❖ 근래에는 한 개의 IC칩 속에 CPU뿐만 아니라 다양한 입출력 장치를 포함하는 SoC 형태로 발전되고 있다.

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





- ❖ 프로세서 내부에서 데이터를 일시적으로 보관하는 기억 장치
  - Flip-flop와 Latch로 구성되어 있다.
- ❖ 범용 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 연산 수행

- 덧셈, 뺄셈 등

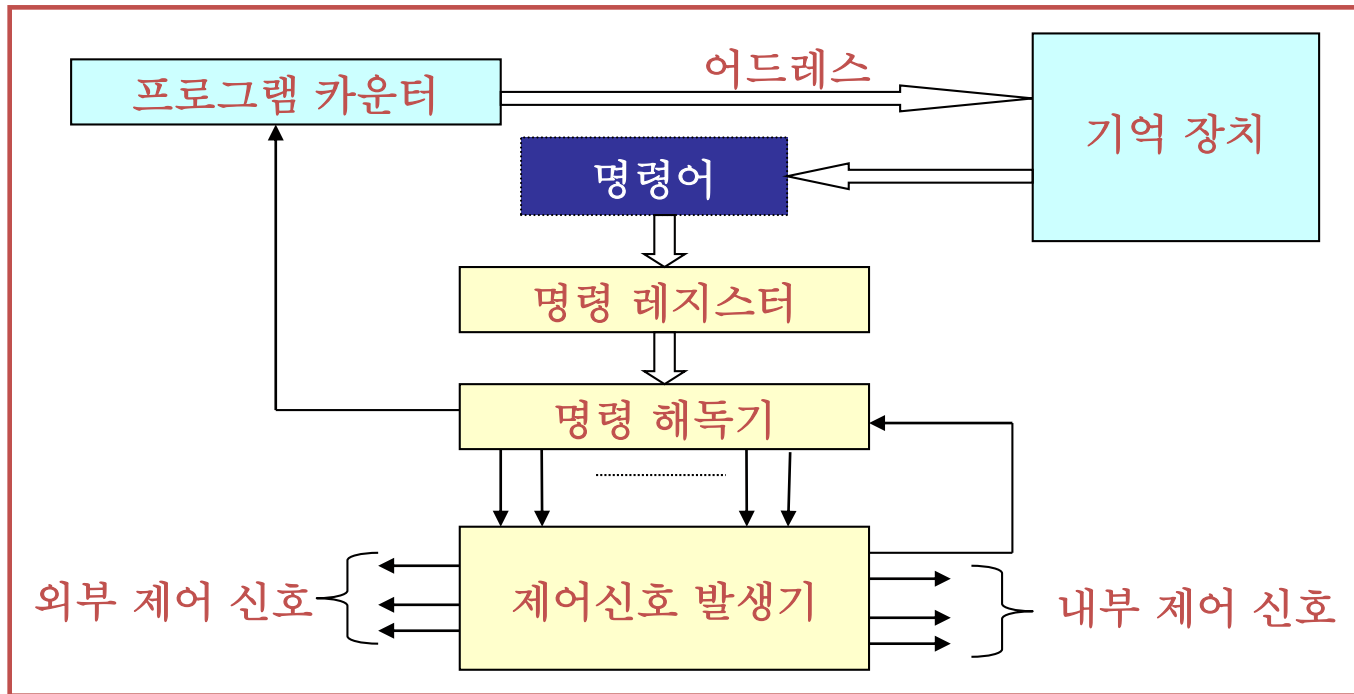
## ❖ 논리 연산 수행

- AND, OR 등

## ❖ 상태 레지스터 또는 flag 레지스터에 연산 결과 기록

- carry 발생, overflow 발생 등

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



❖ 버스는 시스템의 여러 장치들을 연결하는 경로이다.

## ❖ 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 입출력 버스
- 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

## ❖ 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 계속된다.
- 주어진 시간동안에 수행될 수 있는 명령어의 수가 증가한다.

## ❖ 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

## ❖ 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.

- ❖ 프로세서를 통하여 어떤 결과를 얻기 위하여 프로세서가 받아 들일 수 있는 형태로 구성된 명령(instruction)을 나열하여 구성된 문장



## ❖ 기계어 (machine language)

- 프로세서가 이해할 수 있도록 '0' 과 '1'로 표현되는 2진수로 구성된 명령
- 프로세서가 이해하기는 편리하지만 작성자가 프로그램하기에는 불편

## ❖ 니모닉 코드 (Mnemonic code)

- 기계어를 프로그램 작성자가 이해하기 편리한 기호로 표시한 명령

## ❖ 어셈블리어 (Assembly language)

- 니모닉 코드에 보다 편리하게 프로그램 작성자가 이해 할 수 있도록 pseudo 명령 (instruction)을 첨부한 것

## ❖ 장점

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

## ❖ 단점

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

## ❖ 어셈블러 (Assembler)

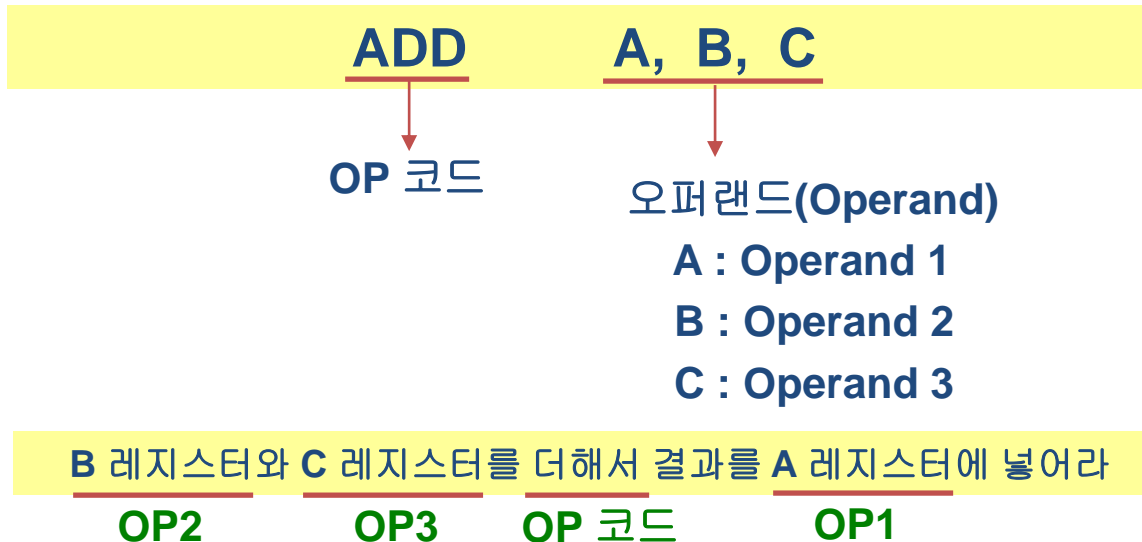
- 어셈블리어를 '0'과 '1'의 2진수로 구성된 기계어로 변환하는 프로그램

## ❖ 어셈블 (Assemble)

- 어셈블리어 프로그램을 기계어로 바꾸는 과정

## ❖ 명령어(Instruction)의 구성

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

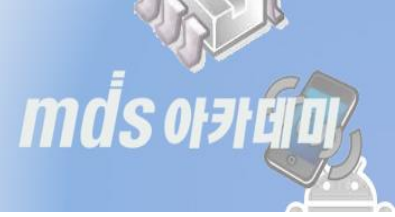


## ❖ 명령어(Instruction) Set에 따른 프로세서 종류

- CISC (Complex Instruction Set Computer)
- RISC (Reduced Instruction Set Computer)

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

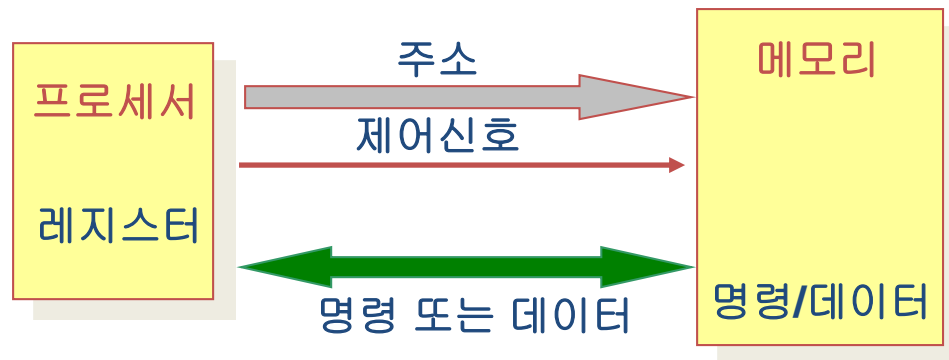
# 목 차



1. 임베디드 시스템의 구성
2. 프로세서
3. 시스템 버스
4. 메모리 장치
5. 주변 장치

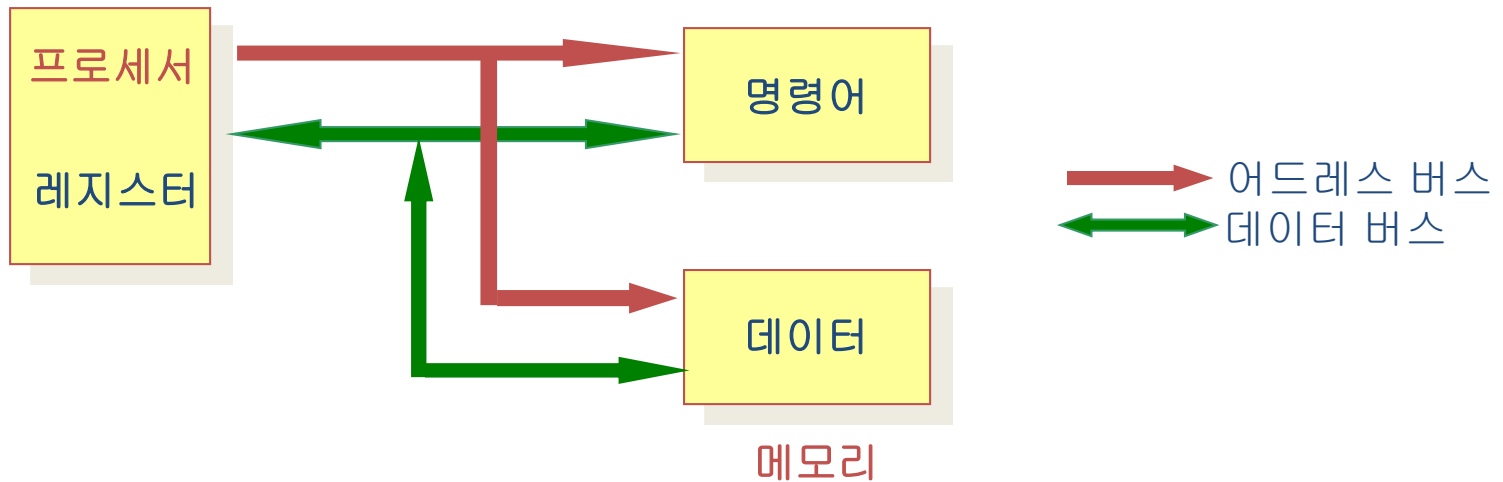
## ❖ 버스(BUS)란 ?

- 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(Von-Neumann) 아키텍처

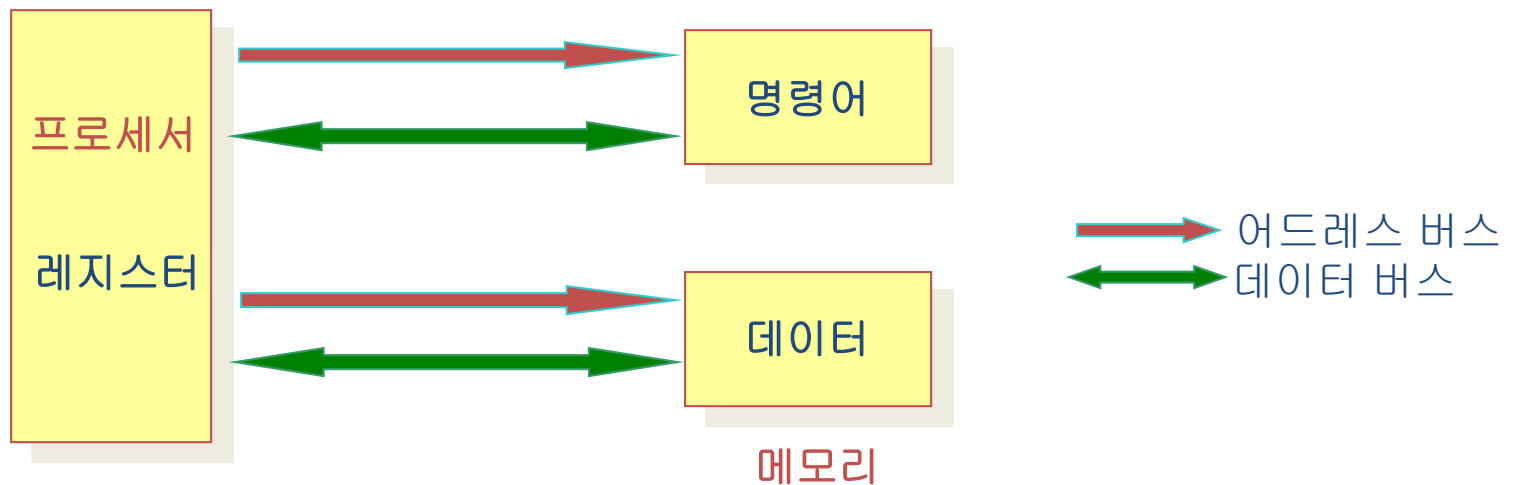
- 명령어와 데이터를 위한 메모리 인터페이스가 하나이다.
- 명령어를 읽을 때 데이터를 읽거나 쓸 수 없다.
- IBM 계열 PC(개인용 PC), ARM7 등





## ❖ 하버드(Harvard) 아키텍처

- 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, ARM11, XScale 등



1. 임베디드 시스템의 구성
2. 프로세서
3. 시스템 버스
4. 메모리 장치
5. 주변 장치

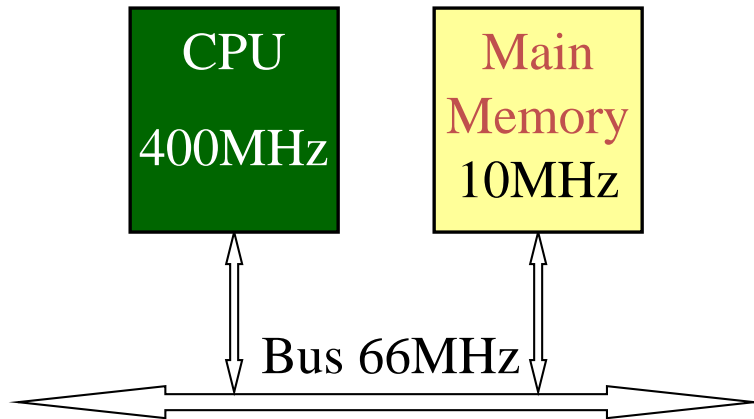
## ❖ 메모리의 종류

구 분			속 도	가 격	용 도	특 징
Volatile Memory	SRAM (Static)		수ns~수십ns 고속	비싸다	캐시 등	
	DRAM (Dynamic)		수십 ns	저렴	주기억 장치	
Non-Volatile Memory	EEPROM (Electrically Erasable)		수십 ns	비싸다	소용량 데이터나 프로그램 저장용	
	Flash	NAND	수십 ns	저렴	데이터 저장 MP3 등	블록 단위 읽기 쓰기 대용량
		NOR	수십 ns	비싸다	프로그램 저장 데이터 저장	

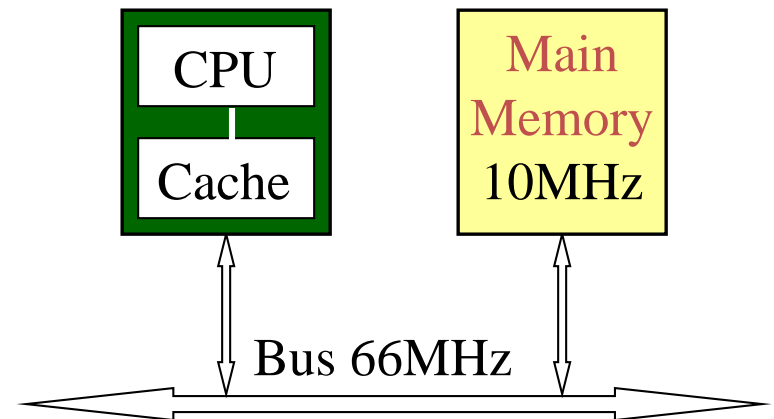
❖ 전원이 인가되는 상태에서만 데이터를 유지

❖ 크게 2 종류의 RAM이 존재

- Static RAM (SRAM) and Dynamic RAM (DRAM)
- 비트가 저장되는 방법이 다름
- 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



고속의 **CPU**가 버스 및 메모리 속도에 의존적이며 느다.



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

## ❖ Cache의 성능

- CPU가 읽고자 하는 명령이나 데이터가 Cache 내에 존재(Cache Hit) 하여야 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 크기만큼 명령이나 데이터를 읽어 Cache 메모리에 저장(Line Fill) 한다.

## ❖ Write Through

- CPU가 특정 주소에 명령이나 데이터를 write하는 경우, 해당하는 명령이나 데이터가 Cache 메모리에 있을 때, Cache 메모리와 외부 메모리에 모두 쓰기 동작을 한다.

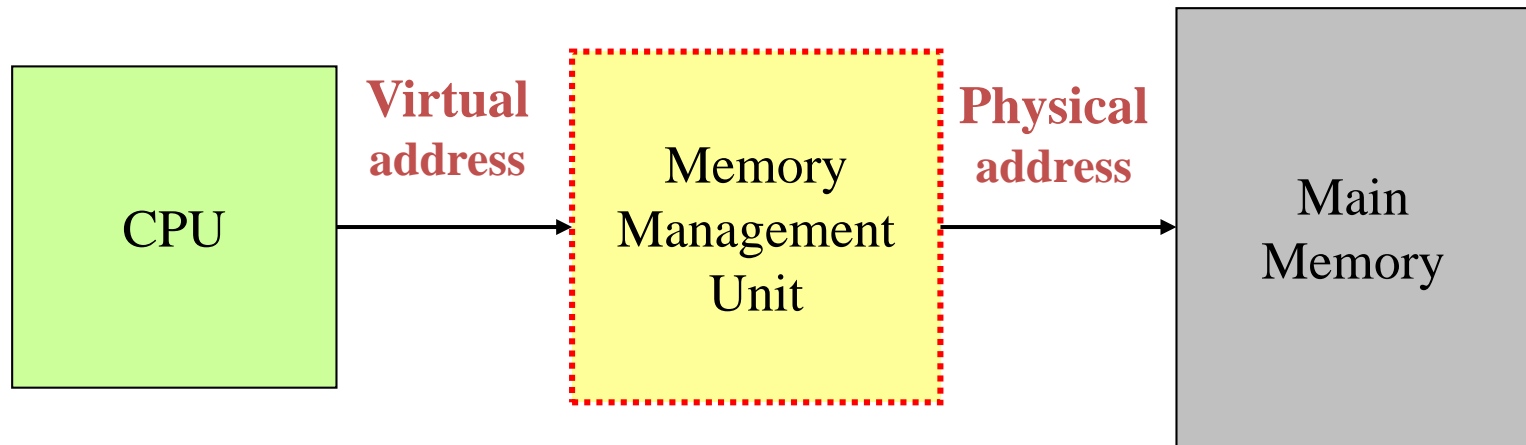
## ❖ Write Back

- CPU가 특정 주소에 명령이나 데이터를 write하는 경우, 해당하는 명령이나 데이터가 Cache 메모리에 있을 때, Cache 메모리에만 쓰기 동작을 하고, 외부의 메모리에 는 나중에 기록된다.

❖ 어드레스 변환(translation) 기능

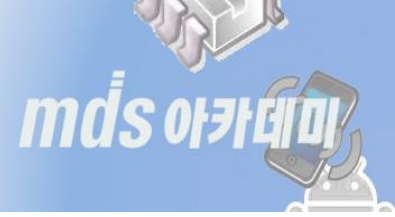
- CPU에서 사용되는 logical 한 virtual 어드레스를 physical 어드레스로 변환

❖ 메모리 보호(protection) 기능



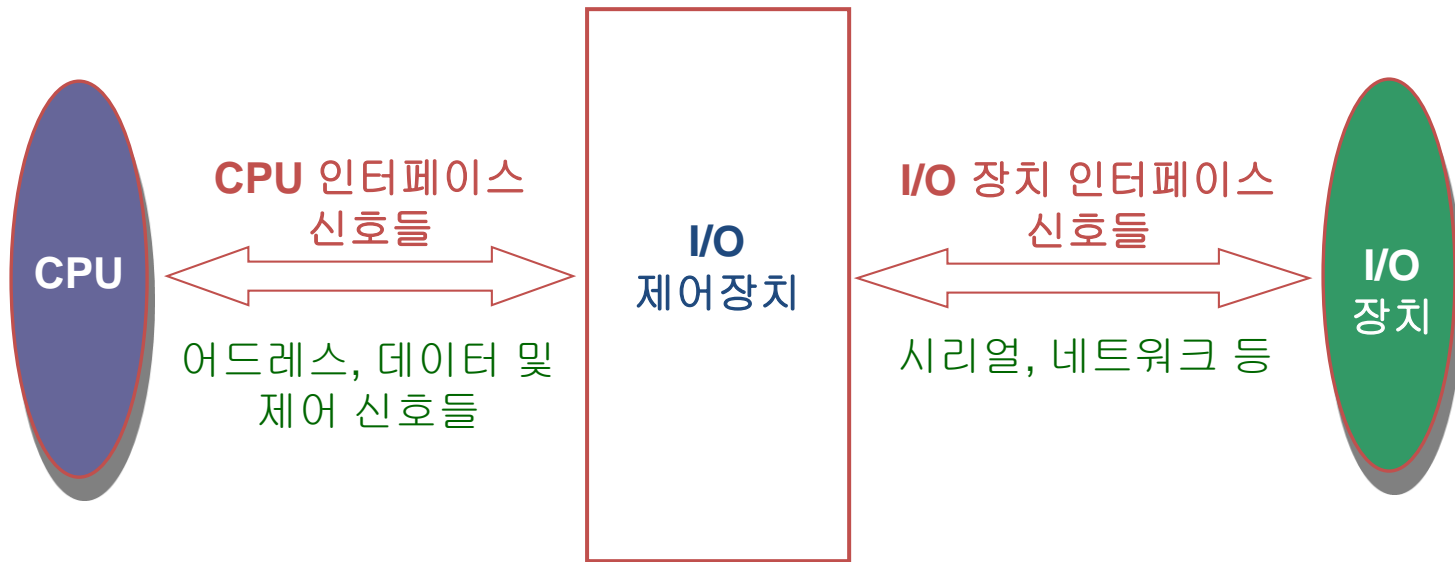


# 목 차



1. 임베디드 시스템의 구성
2. 프로세서
3. 시스템 버스
4. 메모리 장치
5. 주변 장치

- ❖ CPU와 정보를 교환하는 장치
- ❖ Digital 신호 또는 non-digital 신호를 포함 한다.
- ❖ CPU와는 digital 신호를 통해서 연결된다.



❖ 각각의 I/O 장치를 제어하기 위해서는 번지 할당이 필요

❖ 표준 I/O-mapped I/O

- 전용의 I/O 번지를 할당하여 사용
- 인텔의 x86 CPU 계열이 대표적

❖ 메모리 맵(Memory mapped) I/O

- 메모리 번지의 일부를 활용하여 사용
- 대부분의 임베디드 프로세서

# Memory Mapped I/O 와 I/O-mapped I/O

구분	Memory mapped I/O	I/O-mapped I/O
대표적인 CPU	ARM, PowerPC, M68K	x86 계열
I/O 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 액세스 하며, 각 영역의 구분은 어드레스로 한다.	메모리 액세스 명령과 I/O 액세스 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
기타	<ul style="list-style-type: none"><li>- I/O 영역은 <b>Non-cacheable</b>로 설정해야 한다</li><li>- I/O 영역 변수는 <b>volatile type</b>으로 선언해야 한다.</li></ul>	

## ❖ Polling 방식

- 한 프로그램이나 장치에서 다른 프로그램이나 장치들이 어떤 상태에 있는지를 지속적으로 검사하는 전송 제어 방식
- I/O 장치의 접속 여부 및 데이터 전송의 요청과 종료를 검사한다.

## ❖ Interrupt 방식

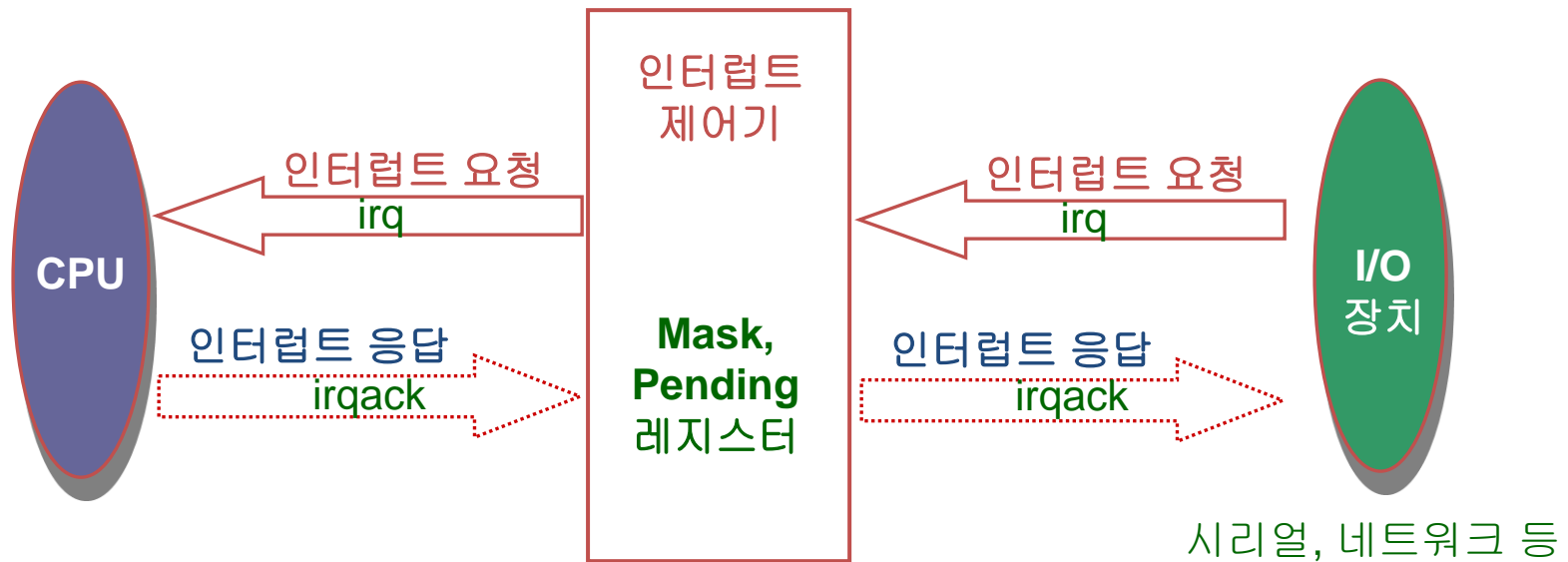
- 프로세서는 일련의 처리를 수행하고, 주변장치에서 입출력 처리 동작이 필요한 경우 프로세서에게 진행 중 이던 명령을 멈추고 새로운 동작을 할 수 있도록 한다.
- 프로세서는 한번에 한 개의 명령만을 수행할 수 있다.
- 인터럽트를 이용하면 멀티 태스킹을 지원할 수 있다.
- 사용자는 모든 작업이 동시에 수행되는 것처럼 보이게 동작한다.

## ❖ DMA 사용

- DMA(Direct Memory Access) 방식은 CPU의 개입없이 I/O장치와 기억장치 사이에 데이터를 전송하는 방식

## ❖ 인터럽트 제어기

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



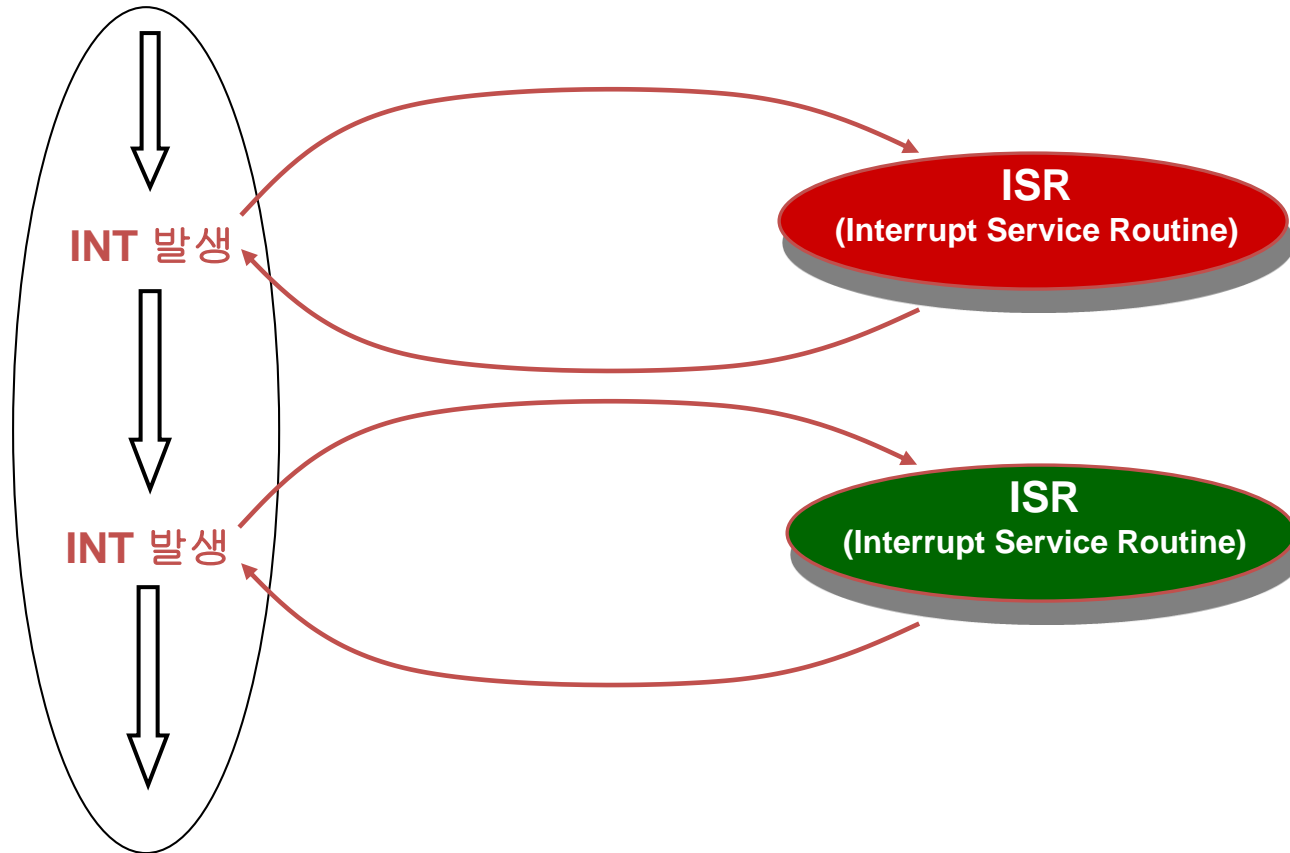
## ❖ 인터럽트 요청 (Interrupt Request)

- 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 처리하기 위한 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

MAIN 프로그램 루틴





## ❖ 인터럽트 Vector

- 인터럽트 서비스 루틴을 처리하기 위한 명령 또는 위치가 저장된 메모리 공간

## ❖ 인터럽트 Vector 주소 지정 방식

- Fixed interrupt
  - 인터럽트가 발생시 처리할 어드레스가 지정되어 변경이 않 된다.
  - 지정된 어드레스에 인터럽트를 처리하기 위한 명령 또는 위치가 저장되어 있다.
- Vectored interrupt
  - 일반적인 microprocessor 장치에서 여러 개의 주변 장치가 시스템 버스에 연결되어 사용 되는 경우
  - 주변장치가 인터럽트를 처리할 주소를 제공

- ❖ 임베디드 시스템의 특징에 대하여 설명하라.
- ❖ 임베디드 시스템의 구성에 대하여 설명하라.
- ❖ 프로세서 또는 CPU의 내부 구조에 대하여 설명하라.
- ❖ 프로세서 내의 레지스터가 가지는 역할과 용도는 무엇인가 ?
- ❖ 프로세서 내의 제어장치가 가지는 역할과 용도는 무엇인가 ?
- ❖ 프로세서 내의 연산장치가 가지는 역할과 용도는 무엇인가 ?
- ❖ 명령(Instruction)의 구조에 대하여 설명하라.
- ❖ CISC와 RISC는 무엇이며, 어떤 차이가 있는지 설명하라.

- ❖ Von-Neumann 아키텍처와 Harvard 아키텍처의 차이는 ?
- ❖ Cache란 무엇인가 ?
- ❖ MMU란 무엇인가 ?
- ❖ I/O 장치의 어드레스 할당 방법에 대하여 설명하라.
- ❖ I/O 장치의 자원을 관리하기 위한 방법에 대하여 설명하라.
- ❖ 인터럽트 Vector란 무엇인가 ?

# 질의 응답