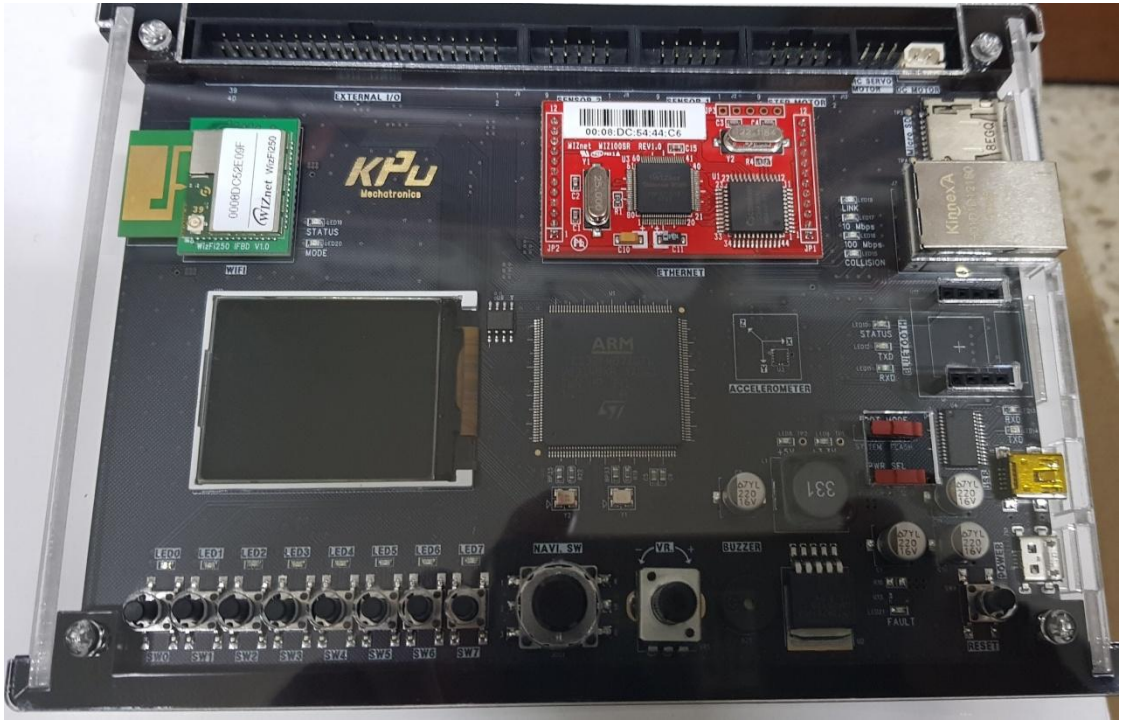
# 마이크로컴퓨터의 구조와 동작



# 목 차

# 1. 마이크로컴퓨터 구조

## 1.1 마이크로프로세서와 마이크로컴퓨터

**(1) 정의**

- **컴퓨터(Computer): S/W에 의해 H/W를 변화 (H/W 회로 또는 H/W 동작 변환) 시키는 전자 기기**
- **컴퓨터의 3가지 구성요소: CPU, Memory, IO-peripherls**
- **CPU(Central Processing Unit):**

  **사용자가 입력한 명령어(프로그램)를 메모리로부터 읽어(Fetch), 해독하고 연산 및 입출력 제어명령디지털신호를 발생/출력하는 동작실행(Execute)을 하는 컴퓨터의 핵심모듈. ALU, Registers, Controller 세 부분으로 구성됨**
- **프로세서 (Processor): CPU**

- 마이크로프로세서(Micro(u)-processor) : One-chip CPU
- 마이크로컴퓨터(Micro-computer):
  마이크로프로세서에 메모리장치가 연결되고 또 외부와 데이터를 송수신할 수 있는 장치(I/O 포트)가 장착된 장치
- 원칩 마이크로컴퓨터: 원칩으로 된 마이크로컴퓨터
- MCU(Micro Controller Unit): 임베디드 제어를 목적으로 제조된 원칩 마이크로컴퓨터 (예:ATMega128, Cortex-M4)

✓ CPU core, 메모리, 프로그램 가능한 입/출력모듈을 가지고 있음

✓ NOR Flash 메모리 기계어 코드를 써 넣음

✓ 기계어 코드가 실행되기 위한 변수나 데이터 저장을 위해 적은 용량의 SRAM을 가지고 있음

✓ MCU는 임베디드 애플리케이션을 위해 디자인되었으며 임베디드시스템에 널리 사용되며, PC가 다양한 요구에 따라 동작하는 일반적인 일에 사용된다면, MCU는 기능을 설정하고 정해진 일을 수행하도록 프로그래밍되어 장치 등에 장착되어 동작함

## * 임베디드 시스템(Embedded system)

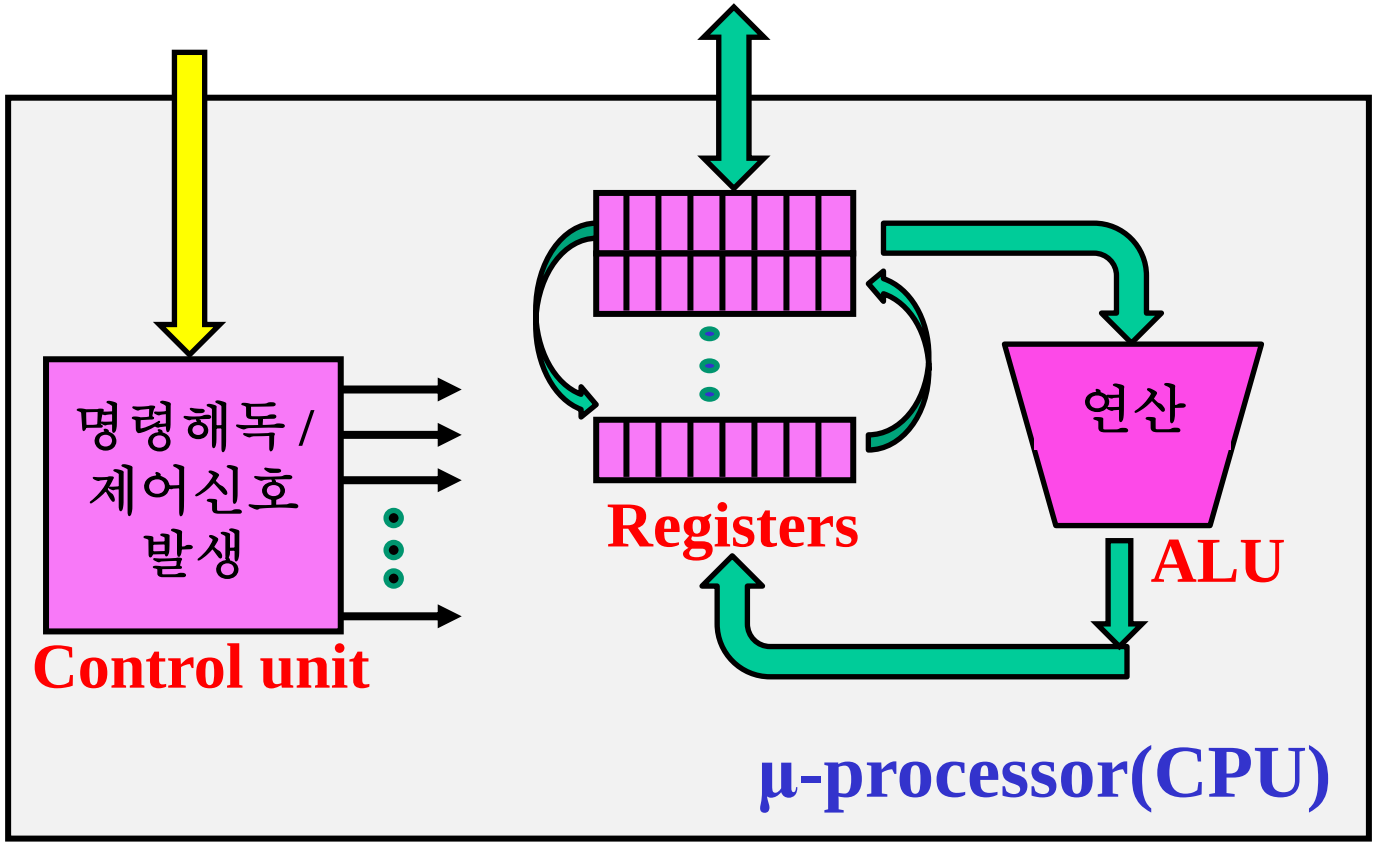
- 임베디드시스템 정의: 기계나 기타 제어가 필요한 시스템에 대해, 제어를 위한 특정 기능을 수행하는 컴퓨터시스템으로 장치 내에 존재하는 전자 시스템임. 즉, 임베디드 시스템은 전체 장치의 일부분으로 구성되며 제어가 필요한 시스템을 위한 두뇌 역할을 하는 특정 목적의 컴퓨터 시스템임
- 임베디드시스템 구성: 대부분은 MCU나 DSP를 메인 프로세서로 장착함
- 임베디드시스템 예: Smart phone, Home network, Digital TV, DVR, GPS, Industrial controller, Digital camera, USB memory, Electric rice cooker, Refrigerator, MP3 player, DMB, Satellite, Measuring instrument(Logic Analyzer, Oscilloscope, Spectrum Analyzer, Network Analyzer, Protocol Analyzer), Router, Network system, Server, Game machine, Micro mouse, Robot controller, Car ECU etc.

## (2) 마이크로프로세서(CPU)의 3가지 구성 요소

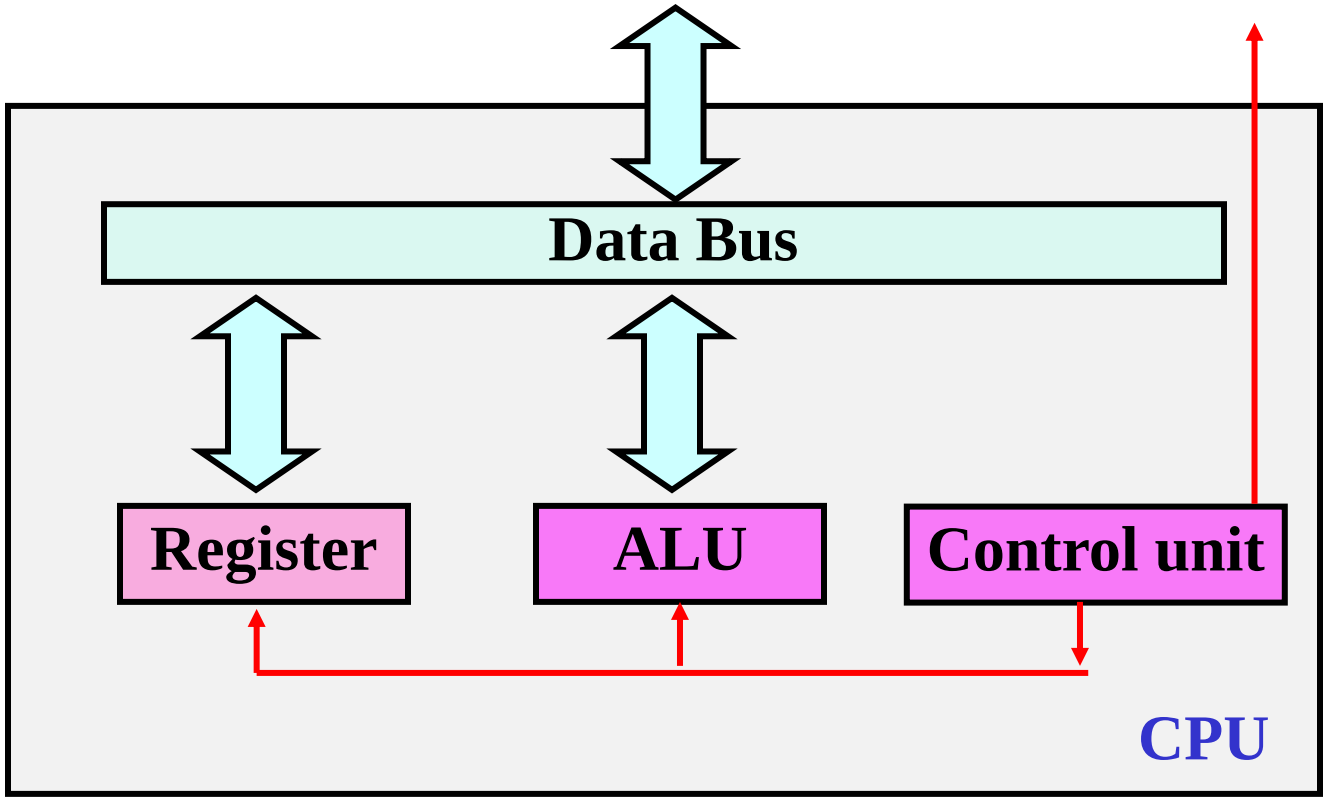
- 산술논리장치(Arithmetic & Logic Unit: ALU)
  산술연산(사칙연산 등)과 논리연산(AND, OR 등)을 수행하는 unit
- 레지스터(register)
  마이크로프로세서에서 프로그램이 수행되는 동안에 데이터를 일시적으로 또는 장기간(주로 일시적) 저장하는 unit
- 제어장치(control unit: Instruction decoder & Control signal generator/Sequencer)

  명령어(프로그램)를 해독하여 컴퓨터를 동작시키는 각종 신호를 발생하는 unit. 즉, 데이터가 마이크로프로세서에 들어오고 나가는 것, 프로그램 명령수행, 그 밖의 모든 동작을 위한 타이밍과 제어신호를 제공하는 unit.

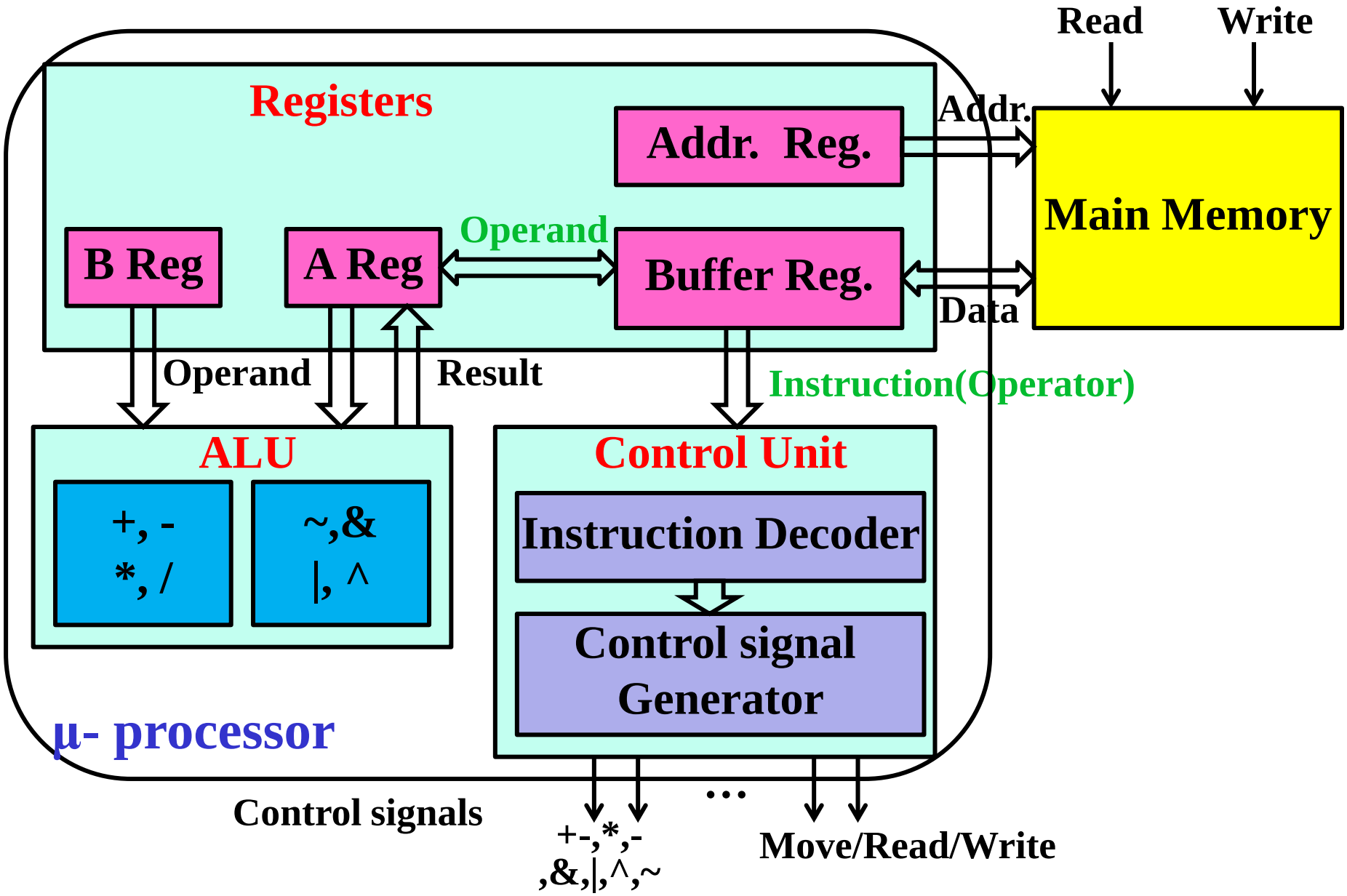
# 마이크로프로세서 내부 구조 및 기본요소-1



# 마이크로프로세서 내부 구조 및 기본요소-2



# 마이크로프로세서의 주요 동작



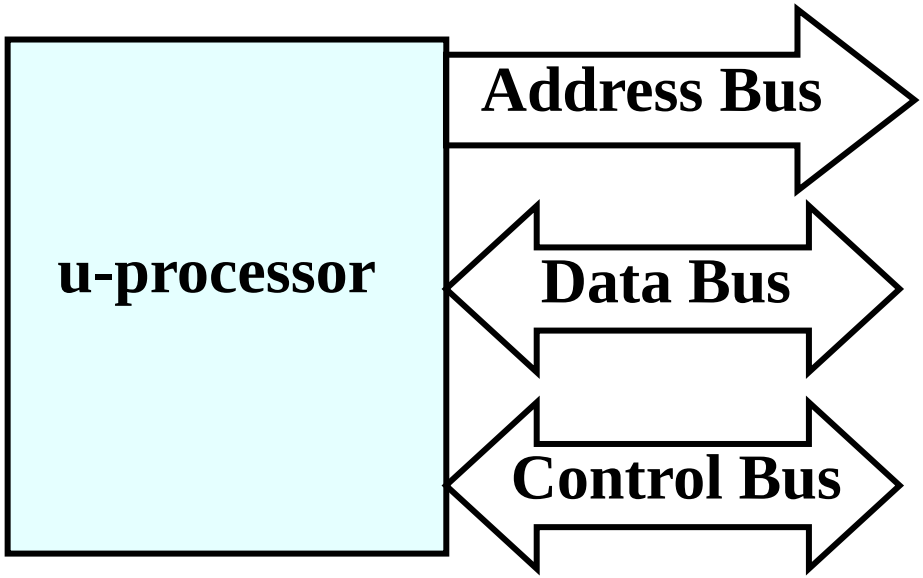
## (3) 마이크로프로세서 버스(BUS)

- 디지털회로의 배선 종류: 전원선(power line), 정보선(data line)
- 정보선: 디지털신호(High(1) 또는 Low(0)) 전송용 선
- 정보선의 종류: 단선(single wire), 버스(group of wires)
- 단선: 1bit의 독립적인 정보(High(1)나 Low(0))를 전달하는 선
- 버스: 같은 성질을 갖는 정보들을 묶어서 함께 송수신하는 선

  * 8,16,32,64개의 선들이 동시에 같은 성격의 신호를 발생
  * Big data를 빠른 시간에 전송할 수 있는 장점있음

  * 버스의 의미: 여러 데이터의 동시전송, 공통으로 사용

- 마이크로프로세서의 버스:
  어드레스 버스, 데이터 버스, 컨트롤 버스

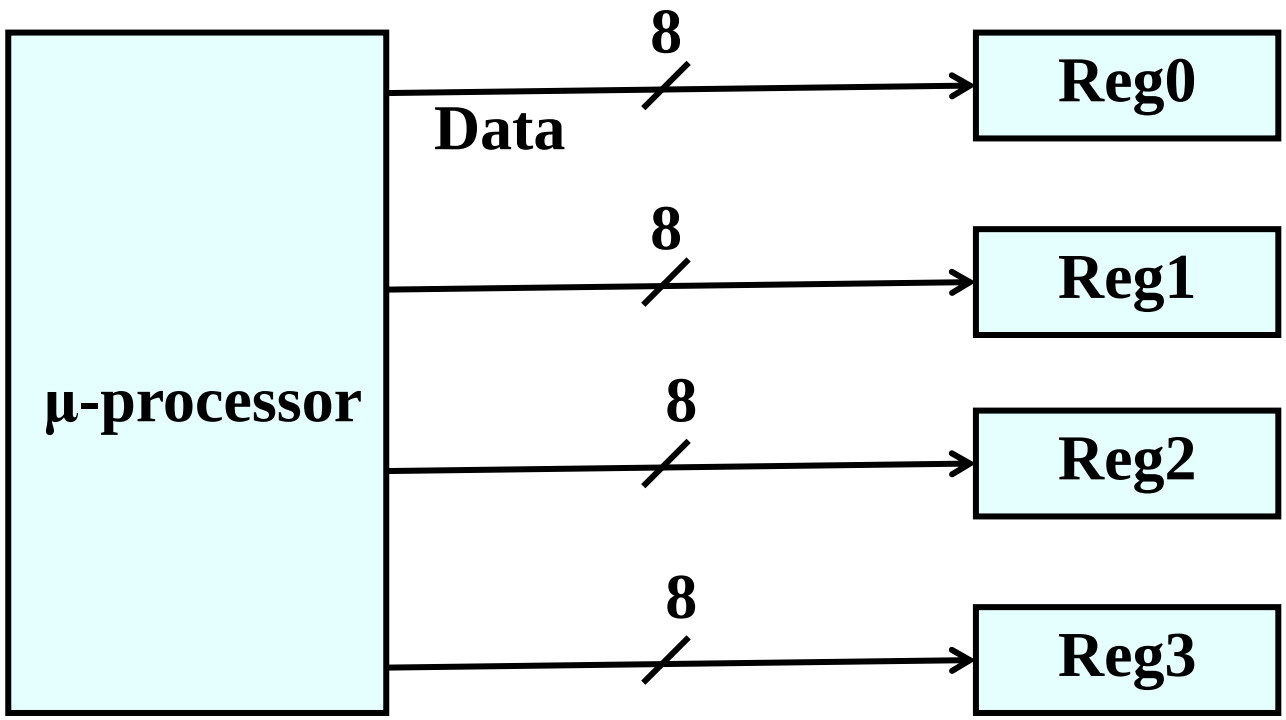



(가) 데이터 버스

- 데이터나 명령 코드(기계어코드)가 마이크로프로세서 내부로 들어오거나 혹은 명령 수행이나 계산의 결과를 마이크로프로세서로부터 밖으로 내보낼 때 사용되는 양방향성 버스
- 컴퓨터의 의미 있는 동작을 가능케 하는 정보전달 버스
- 마이크로프로세서에 따라 **8, 16, 32, 64 bit** 데이터를 취급

- 데이터 버스 개념 도입

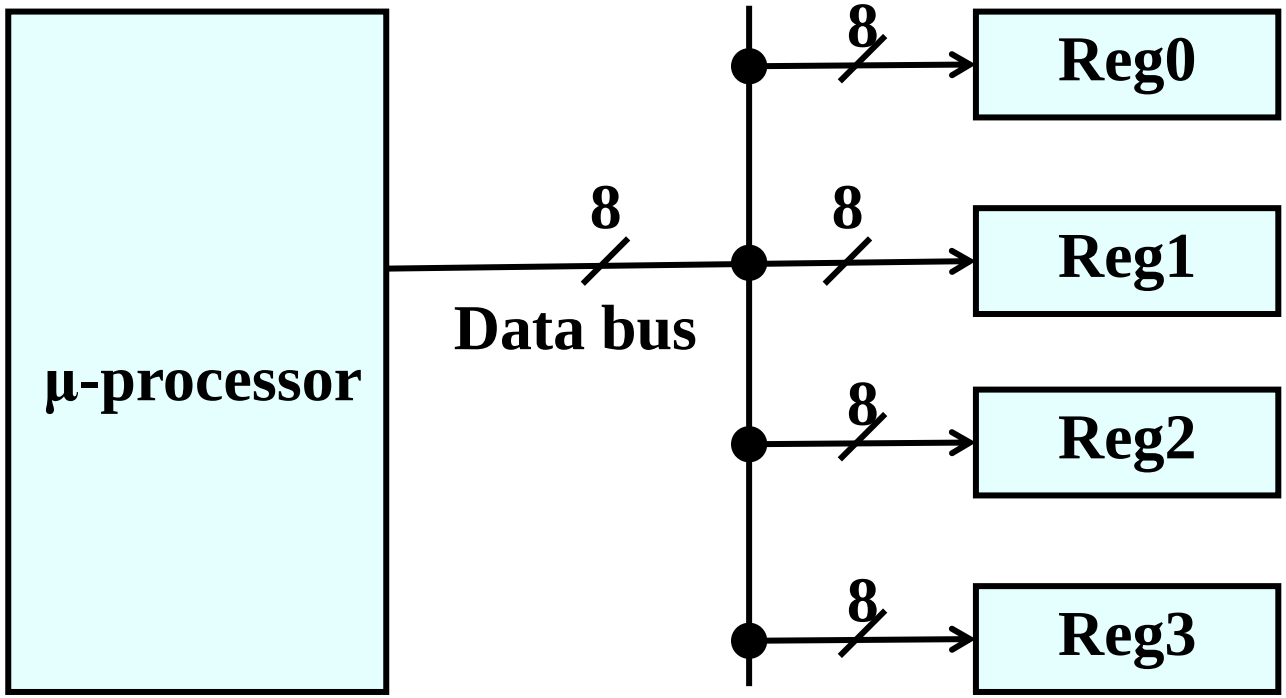
* 데이터버스: 주로 메모리나 레지스터 사이의 데이터 전달




<전용 data lines>

✓ 문제점: 메모리수에 비례하여 마이크로프로세서의 핀 수 증가

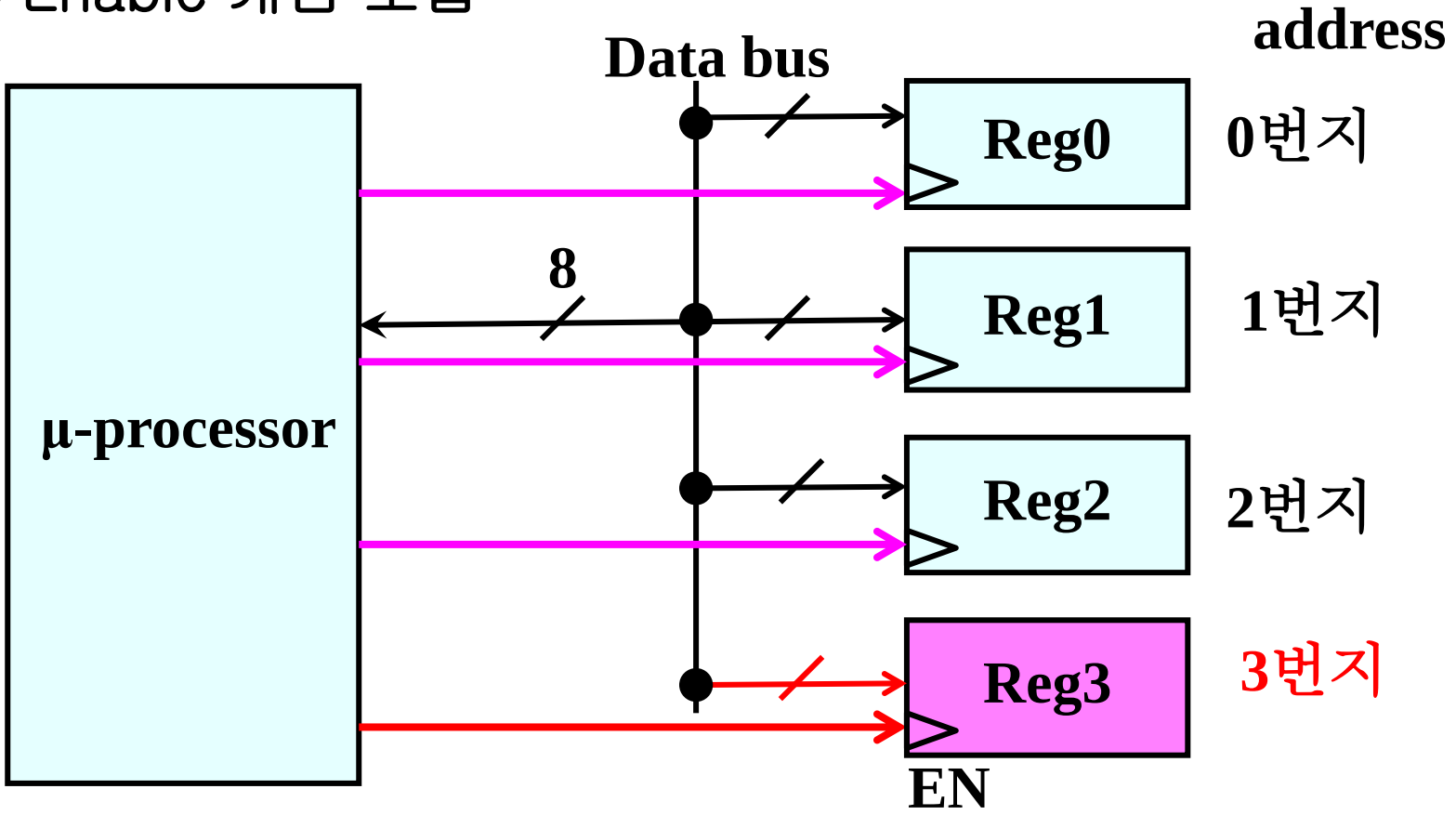
- 데이터 버스 개념 도입




<공용 data bus>

✓ 문제점: 모든 메모리가 같은 데이터 저장하거나 데이터 충돌

- Chip Enable 개념 도입



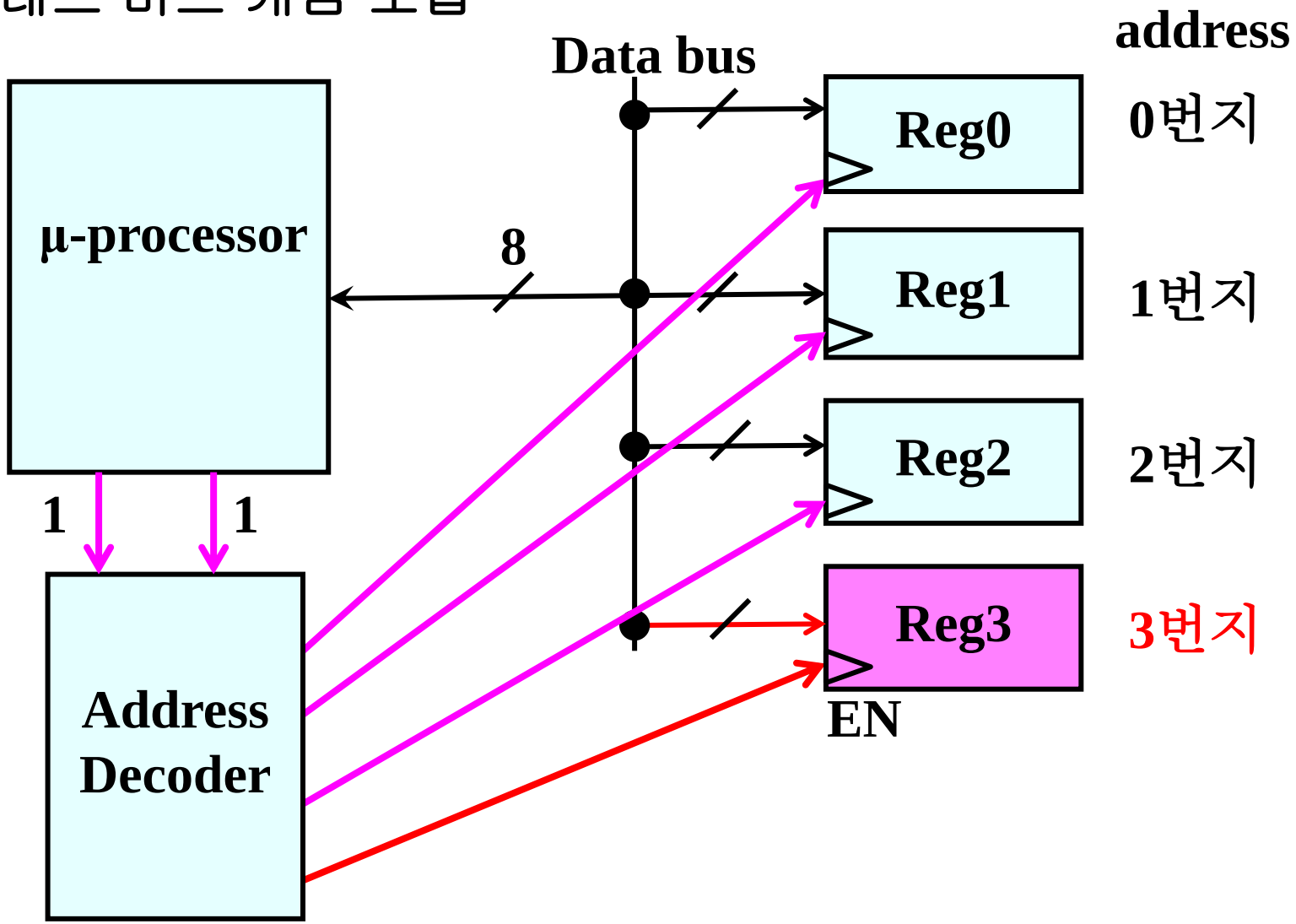

<3번지에 데이터 저장 >

✓Chip/Memory Enable 도입: Enable되면 chip이 데이터 loading

✓ 주소코드에 해당하는 chip만이 Enable되도록 회로 디자인

✓ 문제점: 메모리수(주소갯수)에 비례한 마이크로프로세서의 chip enable 핀 증가

- 어드레스 버스 개념 도입




✓Address code 발생

✓Address decoder 도입

**(나) 어드레스 버스**

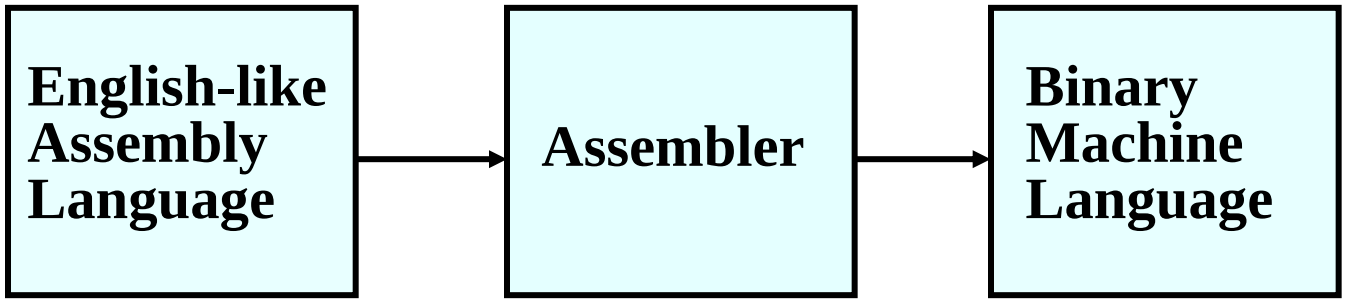
- 마이크로프로세서가 주소 코드를 메모리 혹은 다른 외부 장치로 한 방향으로 보낼 때 사용되는 버스
- 어드레스 버스의 크기는 취급할 수 있는 비트 수에 의해 정해짐 **(4, 8, 16, 20, 24, 32 등)**
- 어드레스 비트 수가 클수록 마이크로프로세서가 액세스**(access)** 할 수 있는 메모리 영역도 증가 **(8 bit: 256, 16 bit: 65,536 메모리 영역)**
- **Q: 32bit ?**

**(다) 제어버스**

- 마이크로프로세서 동작을 조정하기 위한 제반 신호나 외부 장치와의 통신을 위하여 사용되는 버스 **(/Read, /Write 등)**

# (4) 마이크로프로세서 프로그래밍

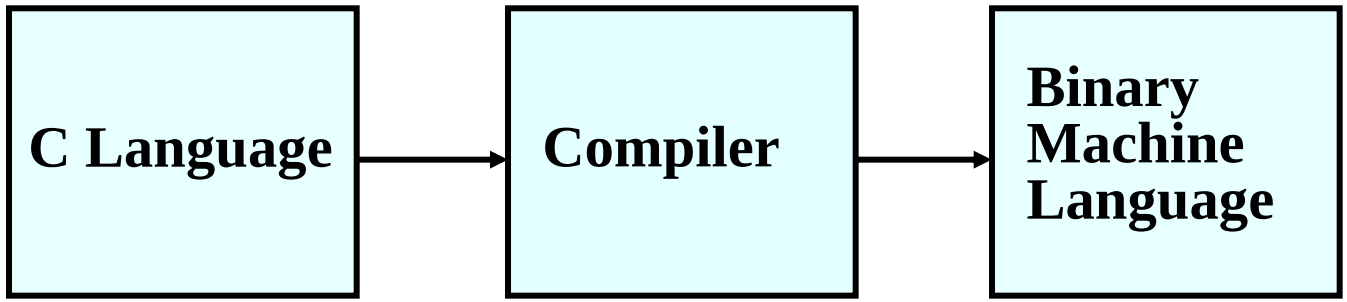
- 마이크로프로세서 프로그래밍: 마이크로프로세서를 동작하기 위한 일련의 명령어를 작성하고 마이크로프로세서에 제공하는 작업



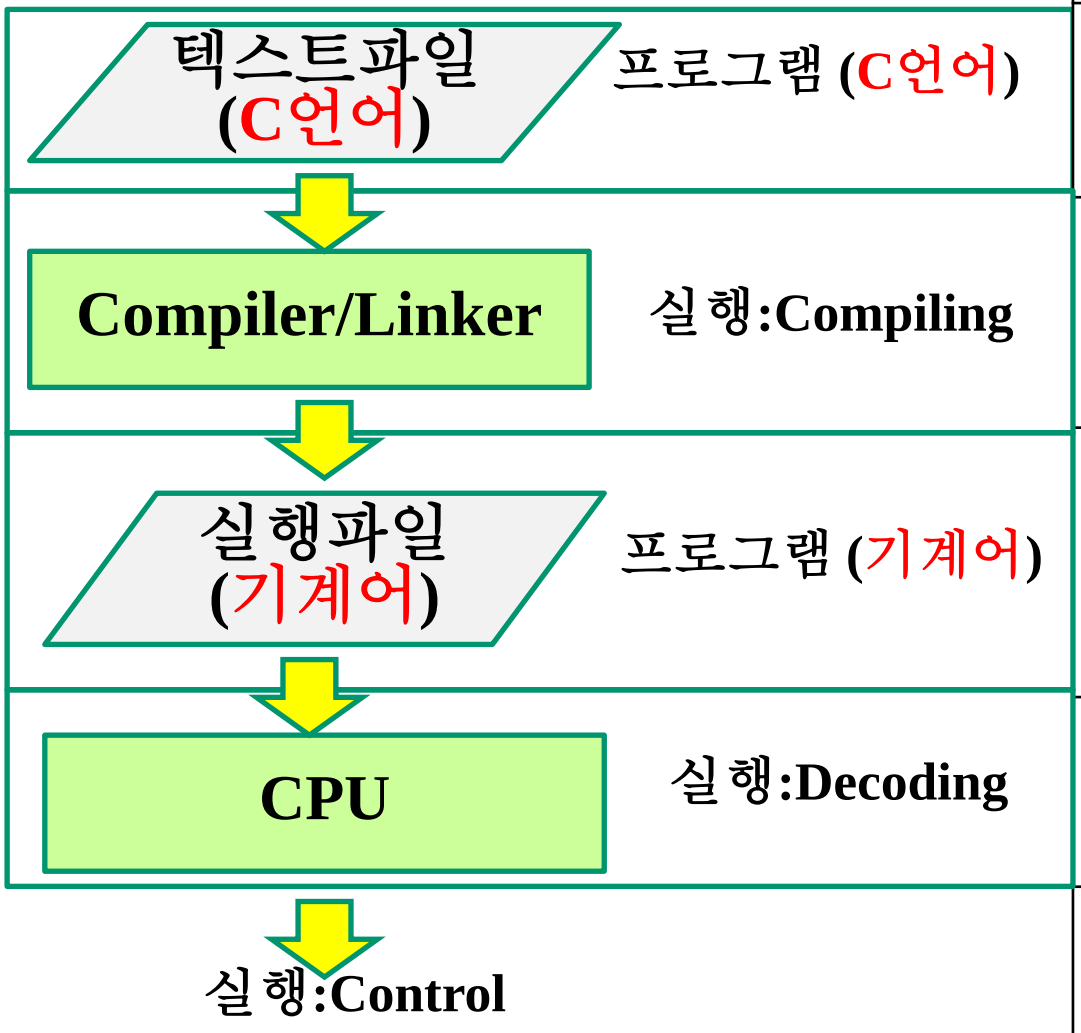

- 어셈블리언어(**assembly language**): 마이크로프로세서를 직접 제어하는데 쓰이는 2진 코드를 의미하는 영어로 된 하위-레벨 프로그래밍 명령어(니모닉:**mnemonics**)
- 기계어(**machine language**): 마이크로프로세서만이 인식하는 2진 코드로 표현된 명령어
- 어셈블러(**assembler**): 니모닉을 기계어로 번역하는 프로그램

- **컴파일러(compiler): 고급 프로그래밍 언어(BASIC, Fortran, C, Pascal 등)를 기계어로 번역하는 프로그램**

> 어셈블리언어/기계어, 어셈블러/컴파일러:
마이크로프로세서마다 다름
> 고급 프로그래밍 언어: 마이크로프로세서에 무관



# u-Processor 프로그래밍




| | PC 프로그래밍 | uC 프로그래밍 |
|---|---|---|
| 파일 위치 | DRAM | DRAM(PC) |
| 파일 위치 | DRAM | DRAM(PC) |
| 파일 위치 | DRAM | DRAM(PC) →Flash(uC) |
| CPU 위치 | PC | uC |
| 제어 대상 | PC | uC |

*uC: 마이크로컴퓨터

## ▪ u-Processor 프로그래밍

- C 언어

(예) char A,B,C;  A=A+B;  C=A;

- 컴파일러 종류

-(Self) compiler: C 파일이 존재하는 컴퓨터의 CPU가 인식(해석) 가능한 기계어 파일로 변환하는 컴파일러 (예: PC에서 실행하는 파일을 생성하는 컴파일러, Visual studio내장 compiler)

-Cross compiler: 타 컴퓨터의 CPU가 인식(해석) 가능한 기계어 파일로 변환하는 컴파일러 (예: 모든 임베디드 컴퓨터용 실행파일을 생성하는 컴파일러, IAR workbench내장 compiler (Cortex용 컴파일러))

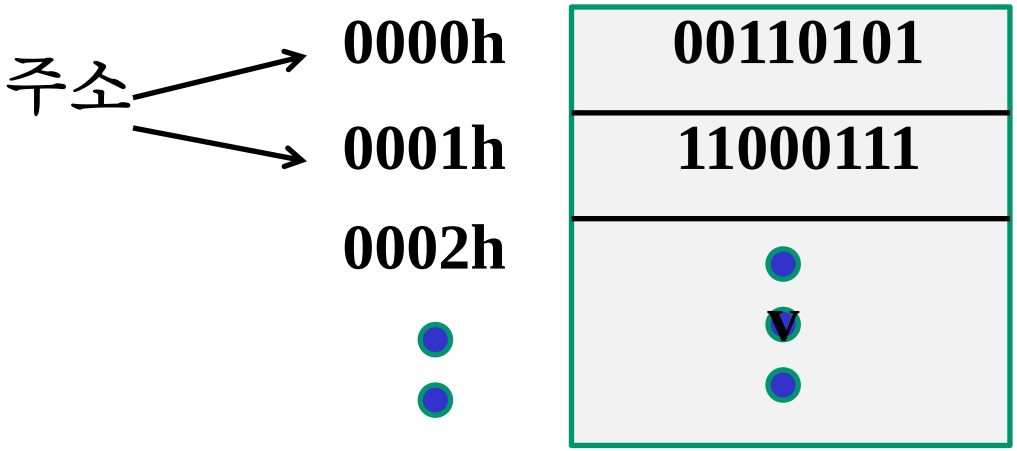
- 기계어(Machine language) : CPU가 인식하는 이진수 조합으로 된 약속된 코드(8,16,32,64 bit)

(예)0b00110101(or 00110101B): “ADD” (A=A+B, A,B: Register)

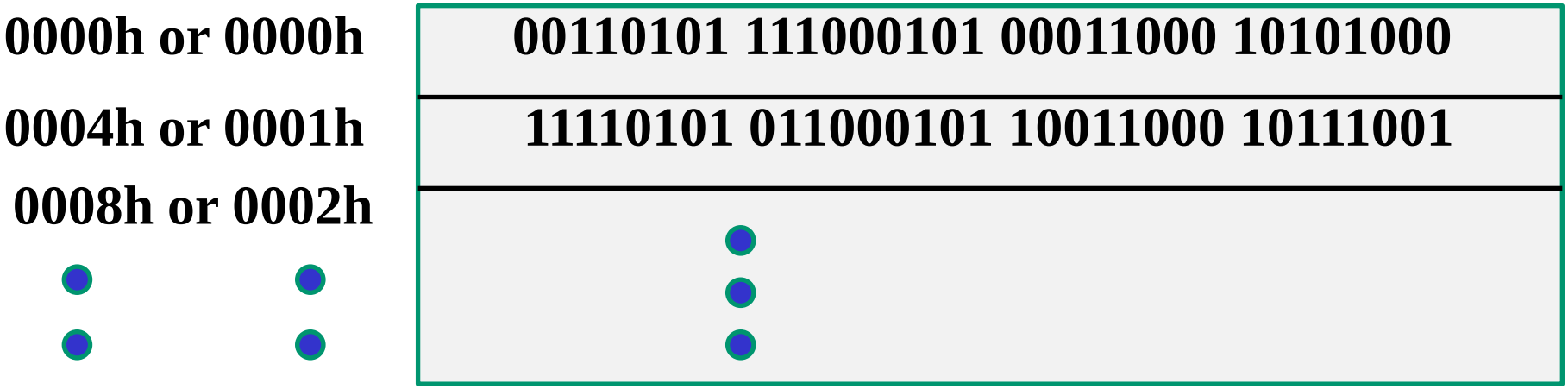
0b11000111(or 11000111B): “MOV” (C←A, A,C: Register)

## ▪ u-Processor 프로그래밍

- 기계어 파일의 저장 형태
  - 8bit 경우 : 명령어 2^8=256가지




- 32bit 경우 : 명령어 2^32= 4G



## ▪ u-Processor 프로그래밍

- 기계어 파일의 저장 매체 : RAM, ROM, Flash(반도체 메모리)

 * PC 경우: HD/SSD에 저장했다가 셀행할 때 DRAM에 load된 후 실행됨

 * 임베디드컴퓨터 경우: Flash/ROM에 저장되었다가 바로 실행됨

- 메모리 특성

 -기본단위: 1-byte(8bit)

 -기본저장단위: 1-byte(8),2-bytes(16),4-bytes(32),8-bytes(64)

 -각 저장단위 마다 주소(Address) 배정되어 관리

 *진법표기

 2진수: 0b…. …B

 10진수: …

 16진수: 0x…, …h

## u-Processor 프로그래밍

- 메모리용량과 주소사이 관계

(Q) 0x0000~0xFFFF의 주소를 가진 메모리 용량은?

(A) 하나의 16진수는 4개의 2진수(4bit)를 가진다. 경우의 수는 2^4=16. 10bit, 즉 0x000~0x3FF(0b0011 1111 1111)는 2^10=1024=1K

그러므로 0x0000~0xFFFF 주소영역의 메모리용량은 2^16=64K (즉, 2^16= 2^6 * 2^10 = 64K)

(Q) 0x0000~0xFFFFFF : 24bit

(A) 2^4 * 2^10 * 2^10 = 16 * K * K = 16M

(Q) 64M의 용량을 가진 메모리의 가능한 주소영역은?

(A) 64 * M = 2^6 * 2^20

= 0b 0011 1111 1111 1111 1111 1111 1111= 0x3FFFFFF

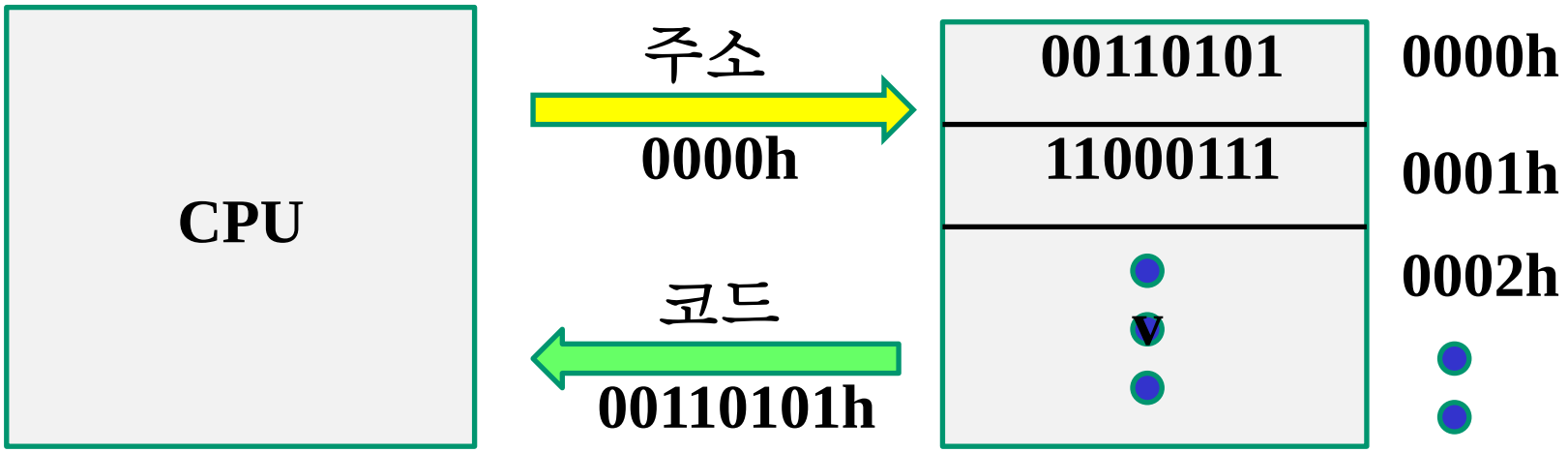
즉 주소영역은 0x0000000 ~ 0x3FFFFFF

(Q) 4G?

## ▪ u-Processor 프로그래밍

- 기계어 코드(문장)을 CPU로 읽어오기

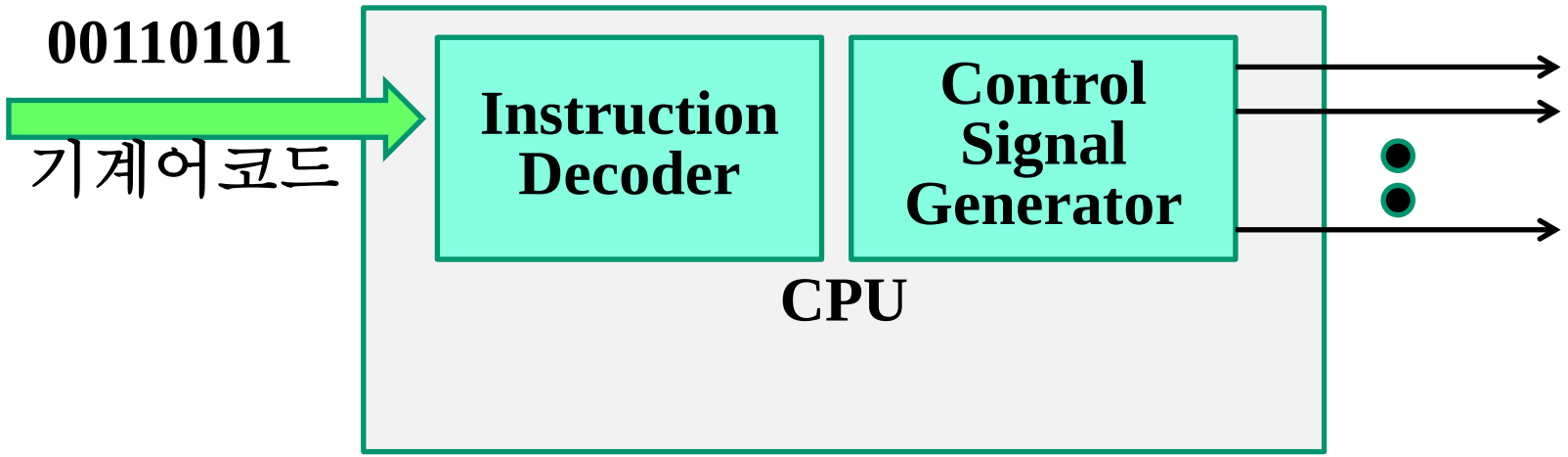
: CPU는 읽어야할 기계어코드가 있는 주소를 메모리에 전송하고 메모리는 주소를 받고 해당 주소의 내용물(기계어 코드)를 CPU에 전송



## ▪ u-Processor 프로그래밍

• 기계어 파일의 해독 및 실행(제어신호 발생)

: CPU 내부의 해독기(Instruction Decoder)에서 기계어코드를 해독(decoding)하고 해독된 명령어에 따라 실행해야할 동작을 위해 순차적으로 제어신호를 CPU는 컴퓨터 내부의 각 모듈(회로)로 전송.

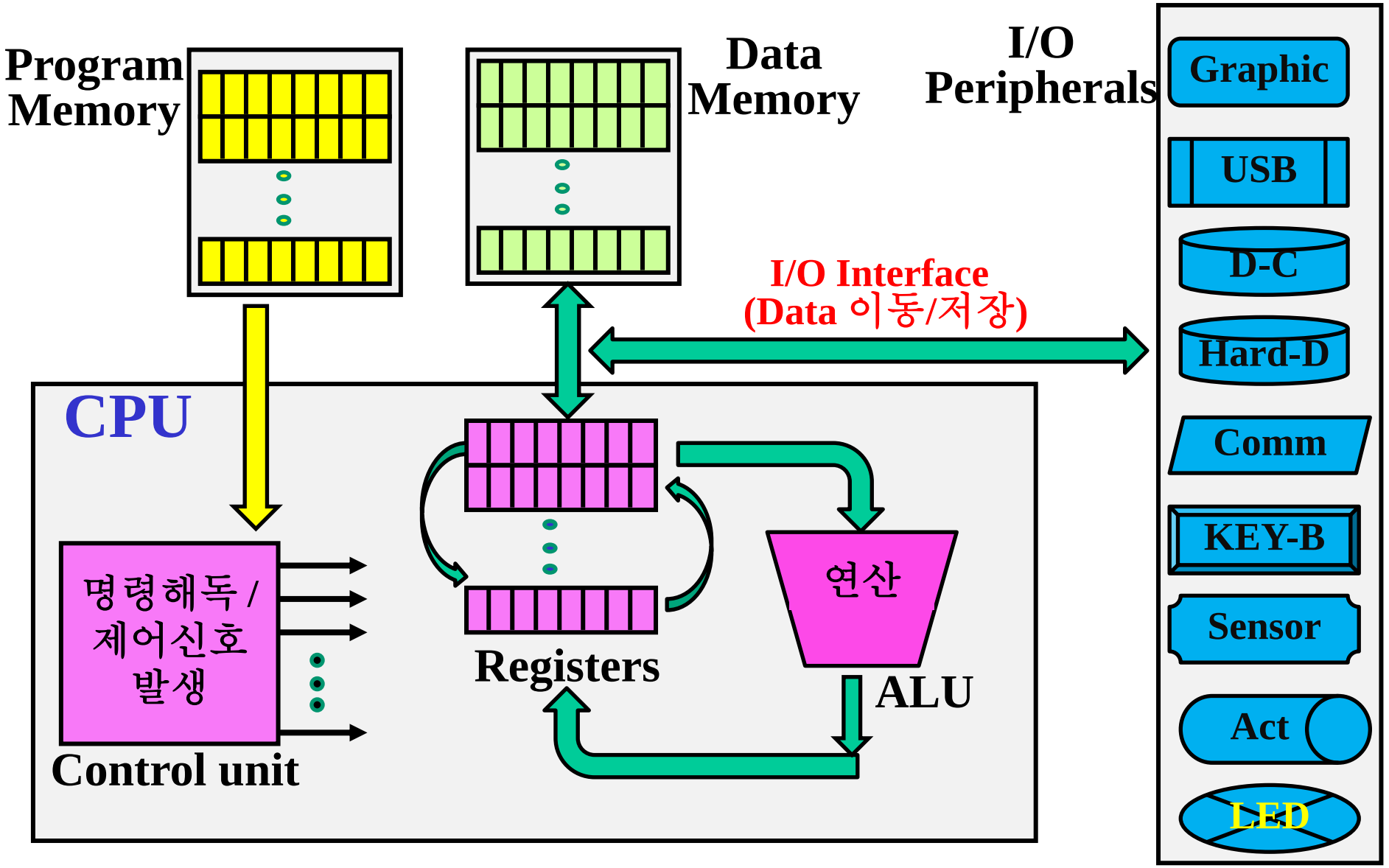


# (5) 마이크로컴퓨터

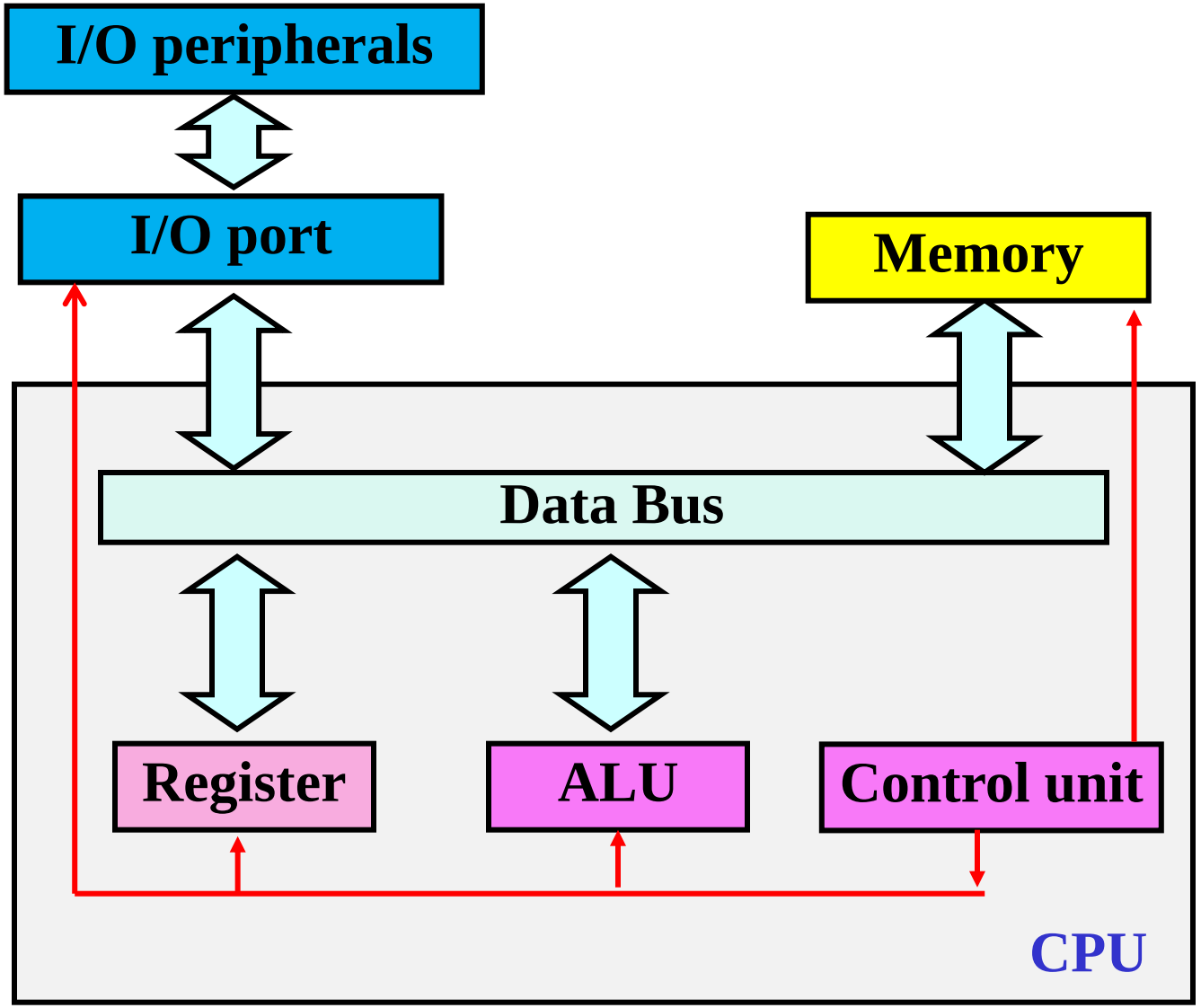
- 마이크로컴퓨터(마이컴)
  = 마이크로프로세서(CPU)+기억장치+입출력장치
- 마이크로프로세서 : 메모리로부터 프로그램 읽어와 해석하고 처리, 입력 장치로부터 외부 데이터 읽어오고, 출력장치를 통해 외부에 데이터 출력.
- 기억장치: 프로그램(마이컴이 수행해야할 명령어들) 저장, 프로그램 수행시 발생된 데이터, 외부에서 들어온 데이터 보관
- 입출력장치: 마이컴과 사용자 및 외부와의 인터페이스(연결)를 수행하는 장치(ADC, 통신(USB, TCP/IP, 마우스, 모니터 인터페이스 회로 등)

- 마이크로프로세서가 버스를 통해 기억장치 및 입출력장치와 정보와 신호를 교환
- 주소는 어드레스 버스를 통해 메모리나 입출력장치에 보내지고, 명령 또는 데이터는 데이터 버스를 통해서 CPU와 기억장치 및 입출력 장치 사이에 전송

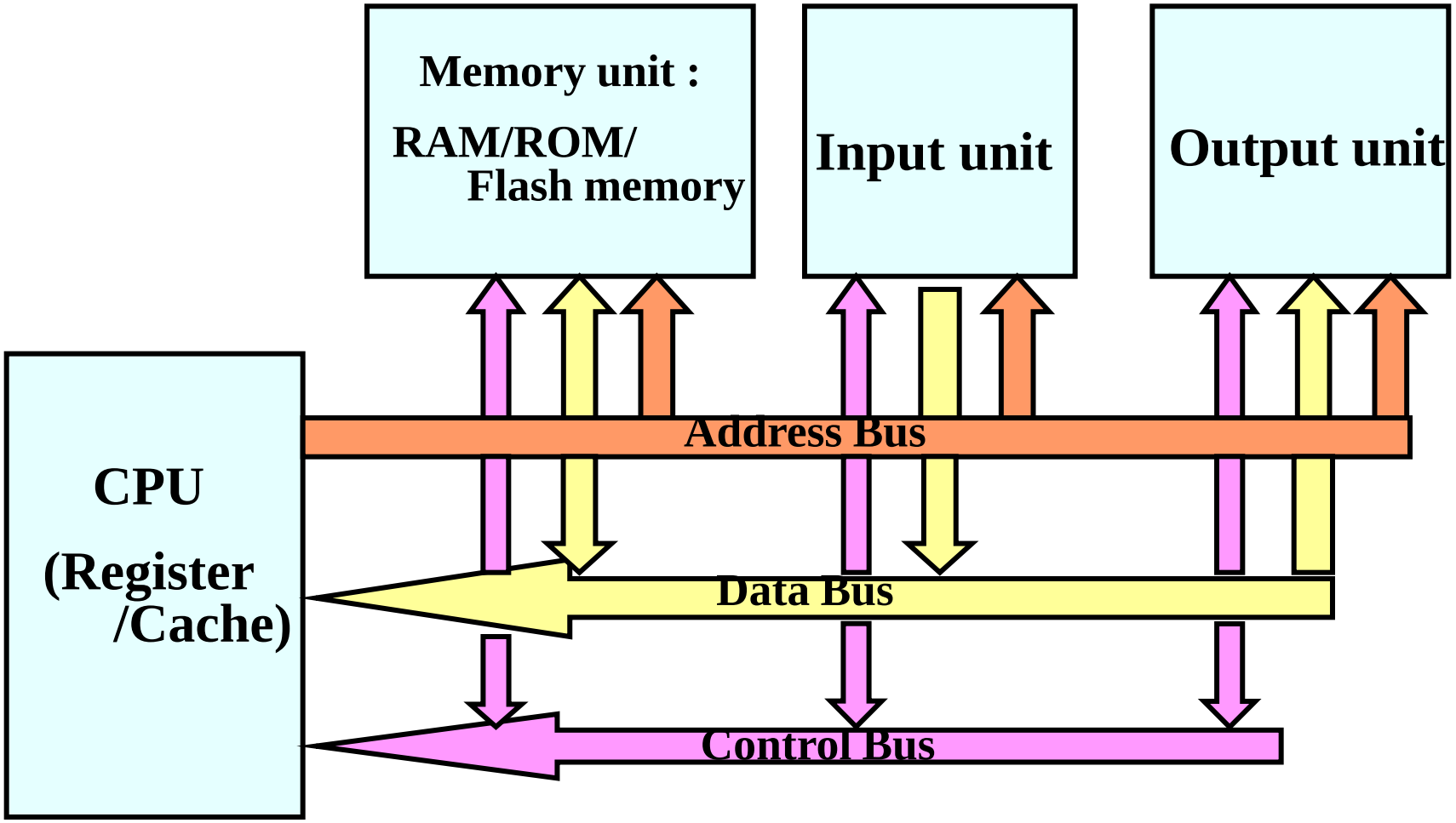
# 마이크로컴퓨터 구조 및 기본 요소 -1 (CPU, Memory and I/O peripherals)



# 마이크로컴퓨터 구조 및 기본요소 -2 (블럭다이아그램)



# 마이크로컴퓨터 구조-3 (BUS based 블록다이아그램)



- 범용 컴퓨터(PC)의 소프트웨어 구조

| Application S/W (아래한글, Visual Studio etc) |
| --- |
| System S/W (Windows, Android etc) |
| H/W dependent S/W (F/W, BIOS etc) |

- 임베디드(전용) 컴퓨터의 소프트웨어 구조

| H/W dependent S/W (F/W, BIOS etc) |
| --- |

# ➢ BIOS(Basic Input Output System)

: 컴퓨터의 가장 기본적인 기능을 처리해 주는 프로그램들의 집합

▪운영체제(OS)에서 I/O 주변장치를 구동하기 위한 루틴들의 집합체로서 운영체제 가장 하위에 있는 부분으로 다른 S/W는 모두 이 층을 기반으로 하여 실행

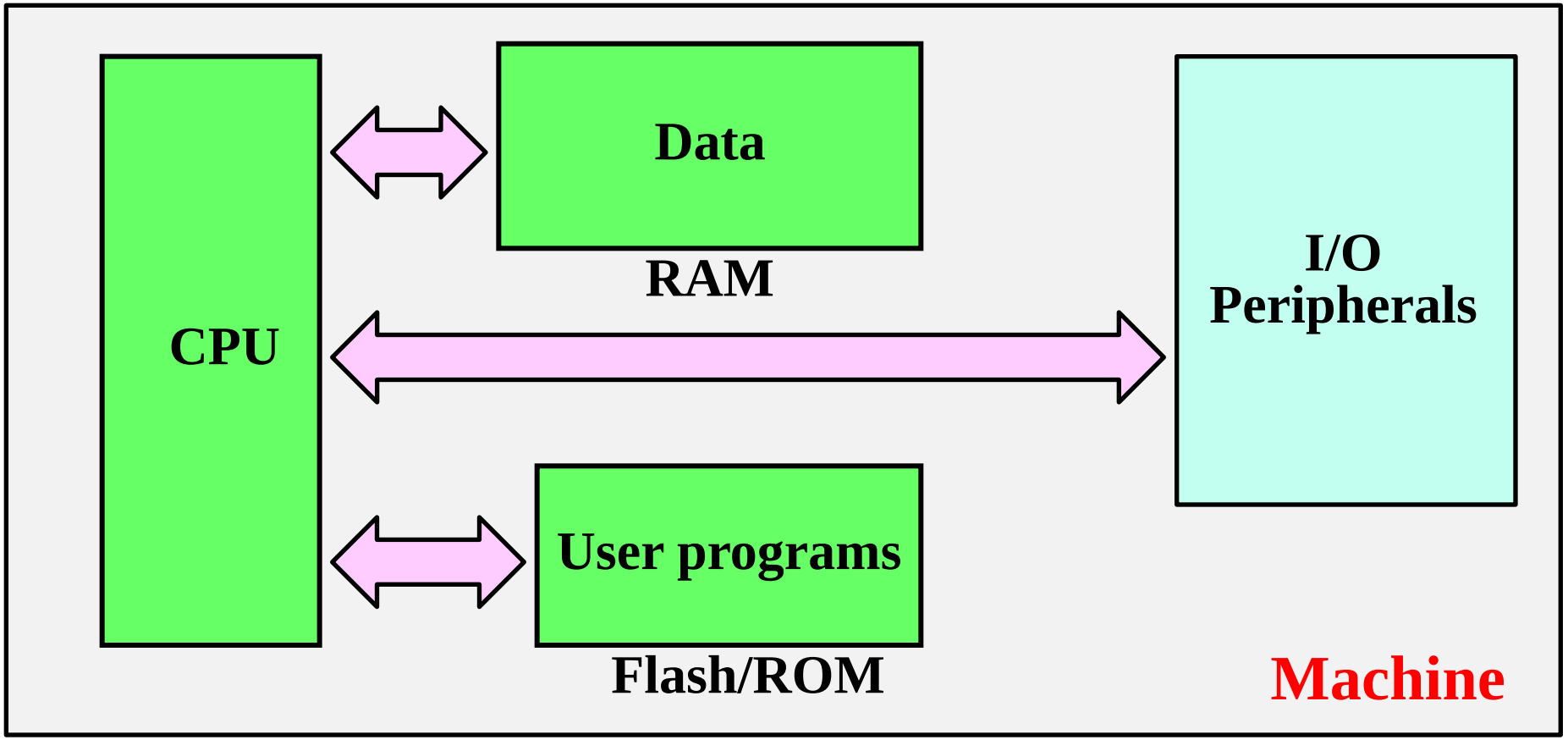
▪역할: 컴퓨터 H/W의 '즉시 응답 요구'를 처리, 프로그램을 H/W와 관계 없이 수행시켜 줌. 말하자면 BIOS는 H/W와 S/W사이의 연결과 번역 기능을 담당하는 인터페이스

▪BIOS를 구성하는 루틴: 스타트업(start-up) 루틴과 서비스 처리 루틴, H/W 인터럽트 처리 부분이 대표적

-스타트업 루틴: 컴퓨터가 켜질 때 자동으로 실행되어 컴퓨터의 상태를 검사하고(POST:Power-On Self Test) 시스템을 초기화(initialize)하는 작업을 하며, 초기화 작업 중에 어떤 I/O 주변장치가 연결되어 있는지 확인

-서비스 처리루틴은 사용자 프로그램 또는 OS가 요구하는 일을 처리. 예를 들어 화면 내용을 지우거나 화면을 텍스트 모드에서 그래픽 모드로 바꾸는 일, 디스크 데이터를 읽거나 프린터로 출력하는 등의 일(모니터, 키보드, 디스크, 프린터 등과 같은 H/W 장치와 관련된 기본적인 입출력 서비스) [네이버 지식백과]

# Embedded Microcomputer의 구조




-Small user program: with BIOS

- Big user program: with small O/S(Linux/Android etc)

## (가) 중앙처리장치(Central processing unit: CPU)

- **CPU**는 메모리 주소를 지정하고 거기에 저장된 명령어를 가져와서(페치:**fetch**) 그 명령을 실행(**execute**)
- 한 명령어를 완료한 후에 **CPU**는 다음 명령어로 이동하며 하나의 프로그램 내에 있는 모든 명령들이 완전히 처리될 때까지 페치와 실행(**fetch-and-execute**) 과정을 반복

* 모든 동작은 **CPU**의 외부에 부착된 오실레이터(또는 **clock**)에 동기(**synchronization**)되어 이루어짐

## (나) 기억장치

- **RAM(random access memory):** 프로그램이 수행되는 동안 데이터나 프로그램을 일시적으로 저장

  **(종류: SRAM, DRAM, SDRAM, DDR SDRAM 등)**

  ***프로그램: μ-processor가 수행해야 할 명령어들의 집합**

- **ROM(read only memory): CPU**를 포함한 시스템 하드웨어 구동에 필요한 기본적인 프로그램(예: **BIOS,** 비디오 디스플레이 그래픽스, 프린터와의 통신, 자기진단 프로그램, 주변장치에 대한 구동 및 서비스 프로그램)을 주로 저장하는 반영구적인 저장장치

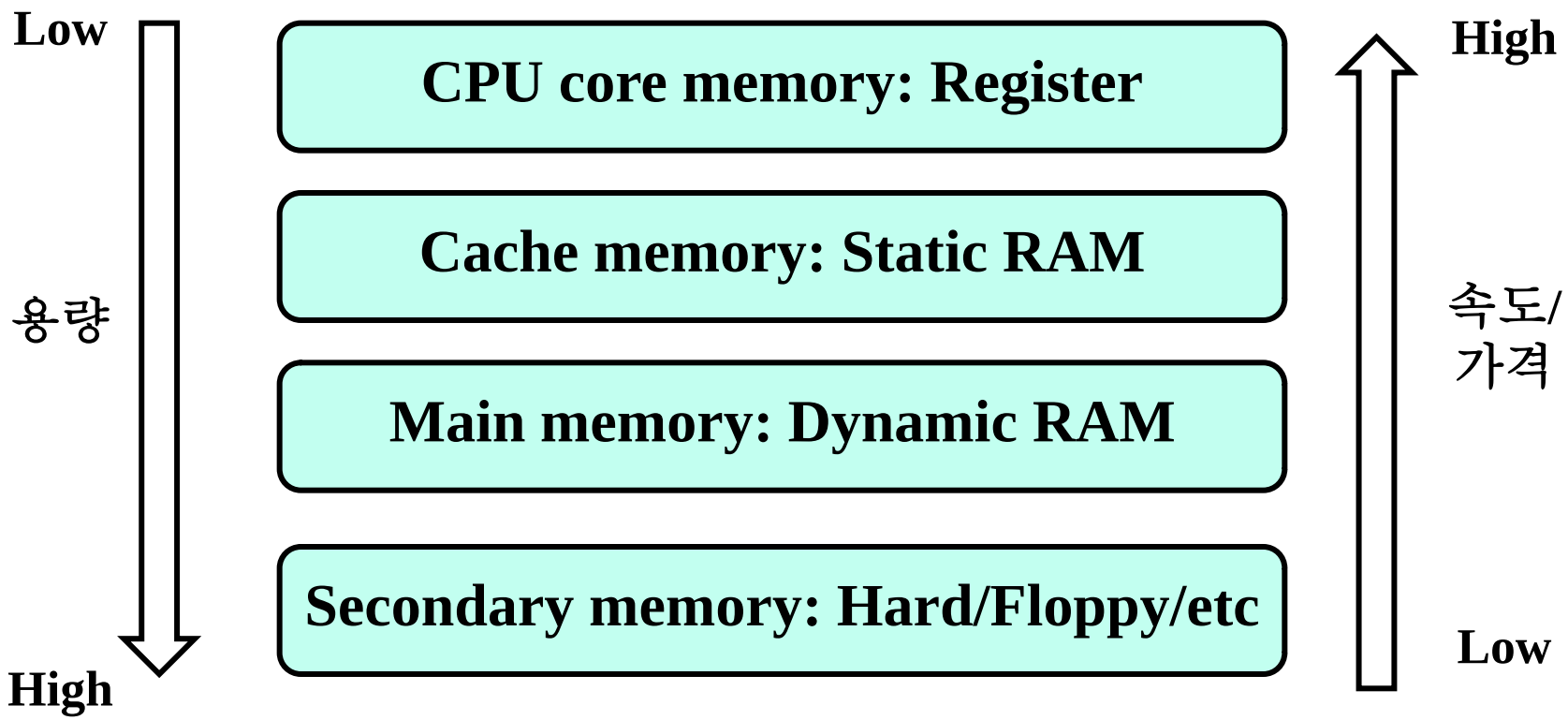
  *** ROM의 종류: Mask-ROM, PROM, EPROM, EEPROM 등**

- **Flash Memory:** 시스템에 있는 상태**(on board)**에서 읽고 쓰기가 가능한 메모리. 데이터 쓰기는 영역지우기를 우선 수행해야함. 프로그램 저장용**(NOR**형**)**과 데이터 저장용**(NAND**형**)**이 있음.

- 디스크: 데이터나 프로그램을 반영구적으로 보관하는 기억장치

  *** 종류: H/D, F/D, C/D**

# 기억장치의 계층구조


Low
용량
High
CPU core memory: Register
Cache memory: Static RAM
Main memory: Dynamic RAM
Secondary memory: Hard/Floppy/etc
High
속도/
가격
Low

(다) 입력장치

외부 정보를 받는 장치: 마우스, 키보드 등

(라) 출력장치

내부 정보를 외부로 내보내는 장치: 비디오 모니터, 프린터 등

* 입출력 겸용장치: 디스크 드라이버, 모뎀 등

# 1.2 마이크로프로세서 역사 (Intel사)

**(1) 초기의 Intel 마이크로프로세서**

- **80X86 마이크로프로세서 이전의 마이크로프로세서**

**: 4004(4 bit, 1971)**
**8008(8 bit)**
**8080, 8085(8bit, 16bit address, 1970 중반)**

**(2) 80X86 마이크로프로세서 계열**

- **X86 아키텍쳐(architecture): 1978년 소개된 이후 5단계로 발달**
- **최초 모델: 8086, 8088, 80186(16bit)**

**두 번째 모델: 80286**
**세 번째 모델: 80386**
**네 번째 모델: 80486**
**다섯 번째 모델: 펜티엄**

**(가) 8086/8088 과 80186 (1978)**

- **8086: 16/20, 10MHz,**
- **8088: 16(8)/20, 8 bit data bus를 가지고 IBM-PC와 호환성 유지**
- **80186: 클럭 발생기 등을 내장, 12.5MHz**

**(나) 80286 (1982)**

- **16/24, 12.5MHz, XT**

**(다) 80386 (1985)**

- **32/32, Pipeline 사용으로 처리속도 향상, co-processor, 33MHz**

**(라) 80486 (1989)**

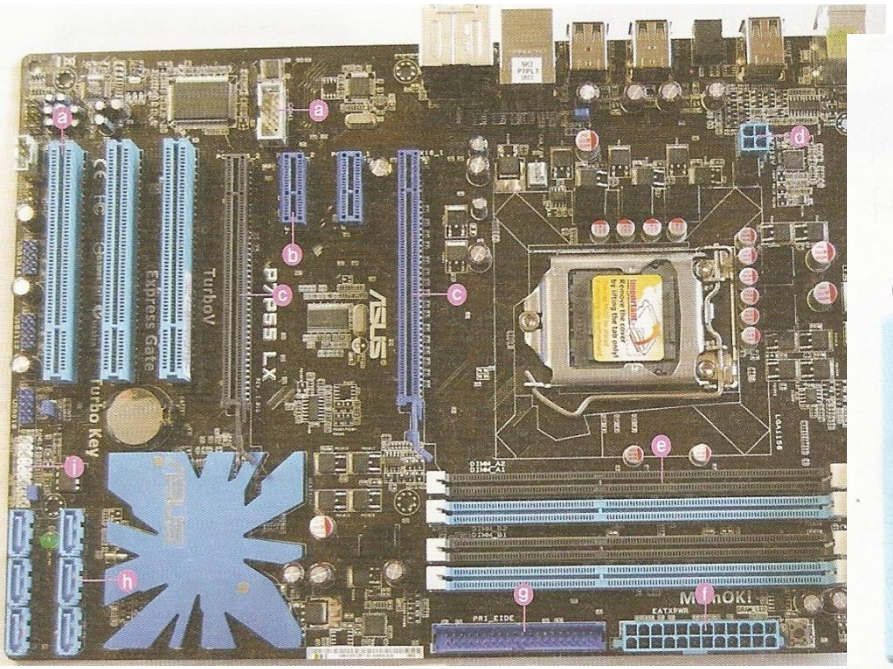
- **32/32, cache memory 사용, 66MHz**

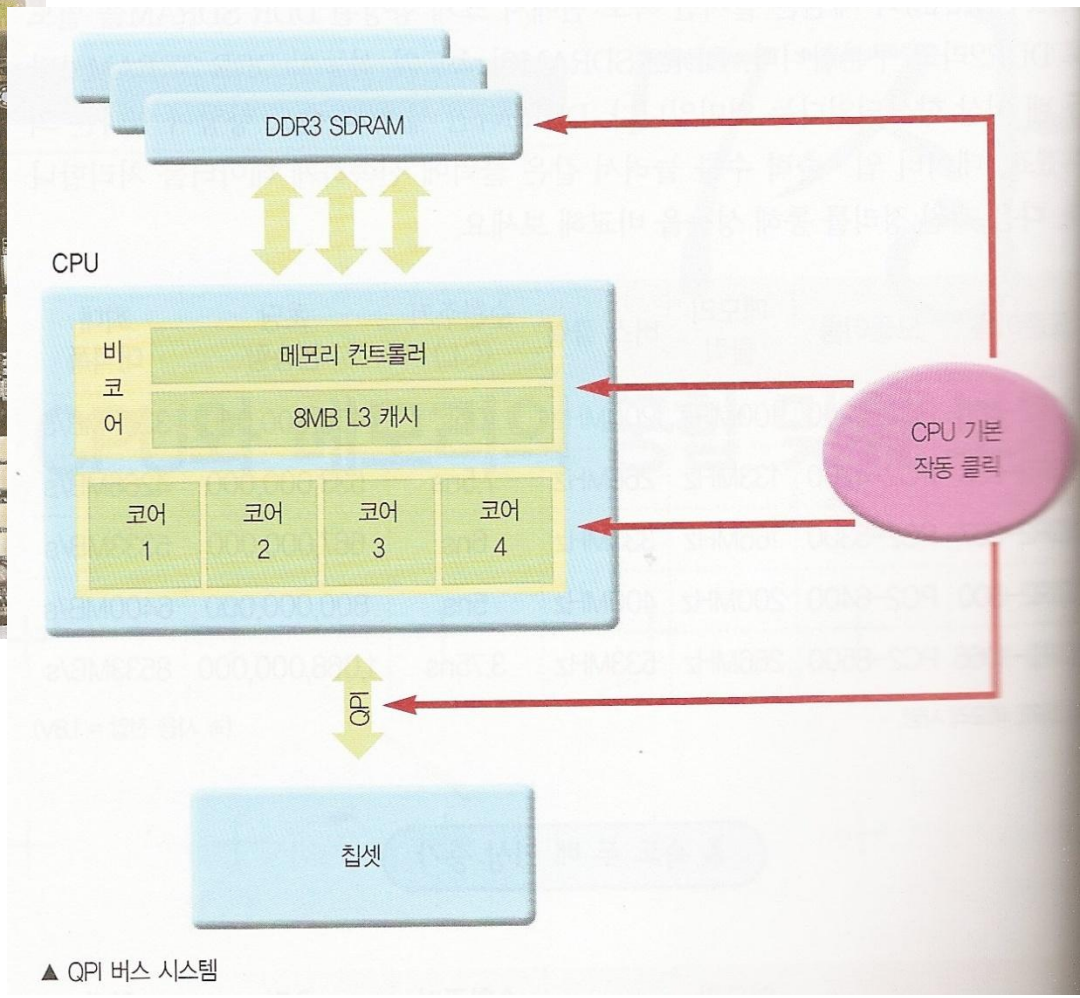
**(마) Pentium (1993)**

- **64/32, cache 증가, 이중 pipeline 구조, 166MHz ~**

# (바) Pentium2~4, Pentium D, Core_Duo, Core2_Duo, Core2-Quad, i3, i5, i7 (1995~현재)

- ~64 data bus, Multi-core(~10개), 3-layer cache, ~ 4GHz






▲ QPI 버스 시스템

## ➢CPU 성능(속도)을 결정하는 요소들

- **Clock frequency :**
- **Word size(Data bus size) : 8, 16, 32, 64 bits**
- **Cache size :**
- **Command structure : CISC, RISC**
- **Pipelining :**
- **Parallel processing : dual-core, quad-core**

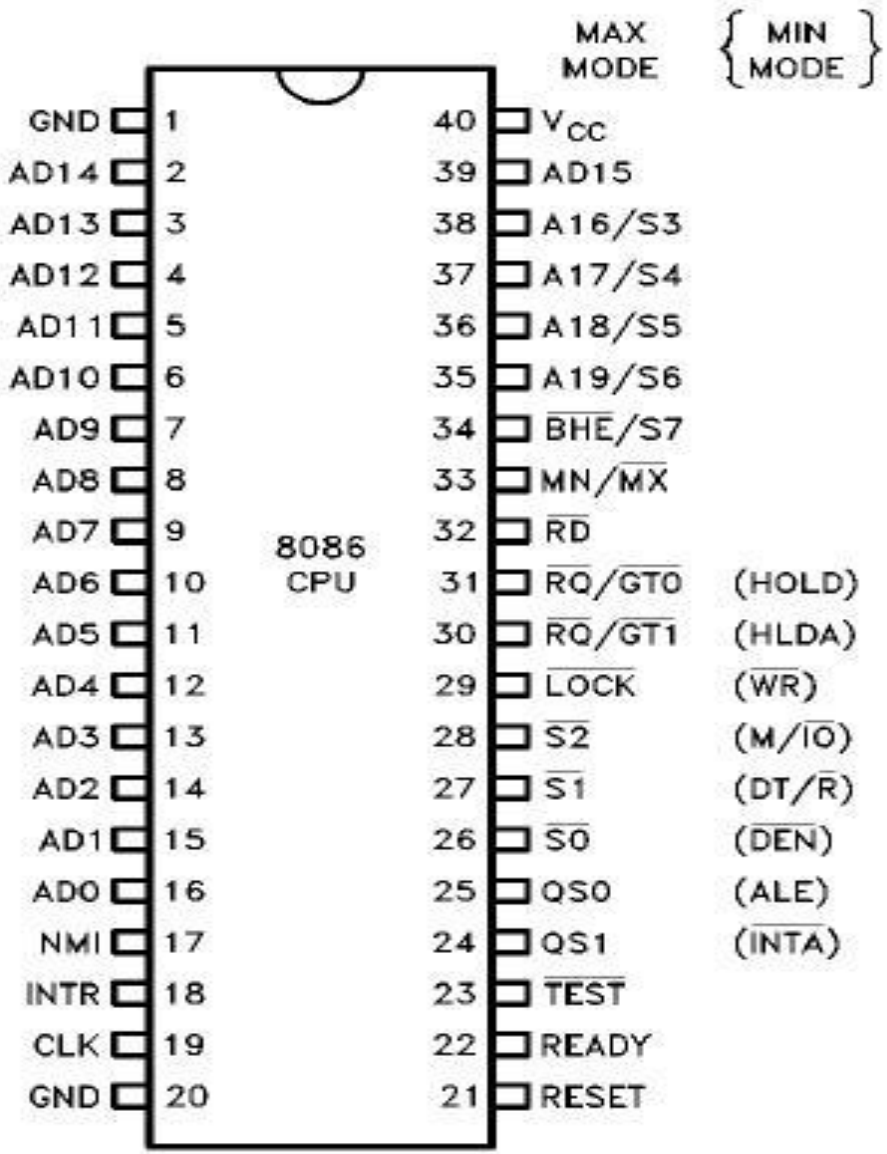
# 1.3 8086/8088 마이크로프로세서

## (1) 기본 동작

① **Fetch instruction**
② **Read operand**
③ **Execute**
④ **Write result**

**Memory**

| |
|---|
| Operator(E4H) |
| Operand(02H) |
| Operator(04H) |
| Operand(05H) |
| Operator(E6H) |
| Operator(03H) |
| |



- 마이크로프로세서는 프로그램메모리로부터 읽어온 **operator (instruction)** 에 따라 다음 네가지 동작사이클중 하나의 동작을 수행

  **-사이클 0 : ① Fetch instruction => ③ Execute**

  **-사이클 1 : ① Fetch instruction => ② Read operand => ③ Execute**

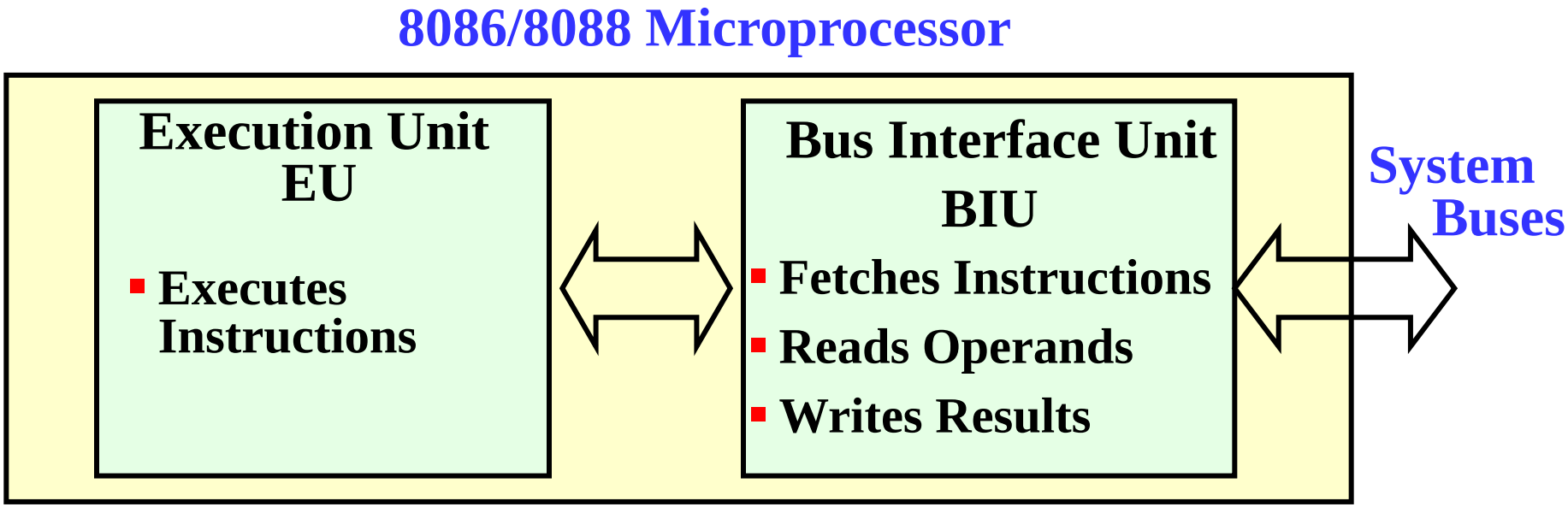
  **-사이클 2 : ① Fetch instruction => ③ Execute => ④ Write result**

  **-사이클 3 : ① Fetch instruction => ② Read operand => ③ Execute => ④ Write result**

*** Instruction Decoding는 ① Fetch instruction 동작에 포함**

## (가) 특징1: 2개의 분리된 내부장치에 의해 수행

**: EU(execute unit), BIU(bus interface unit)**



# (나) 특징2: Pipelining 구조(속도 증가 효과 )

## : Prefetching, Instruction queue

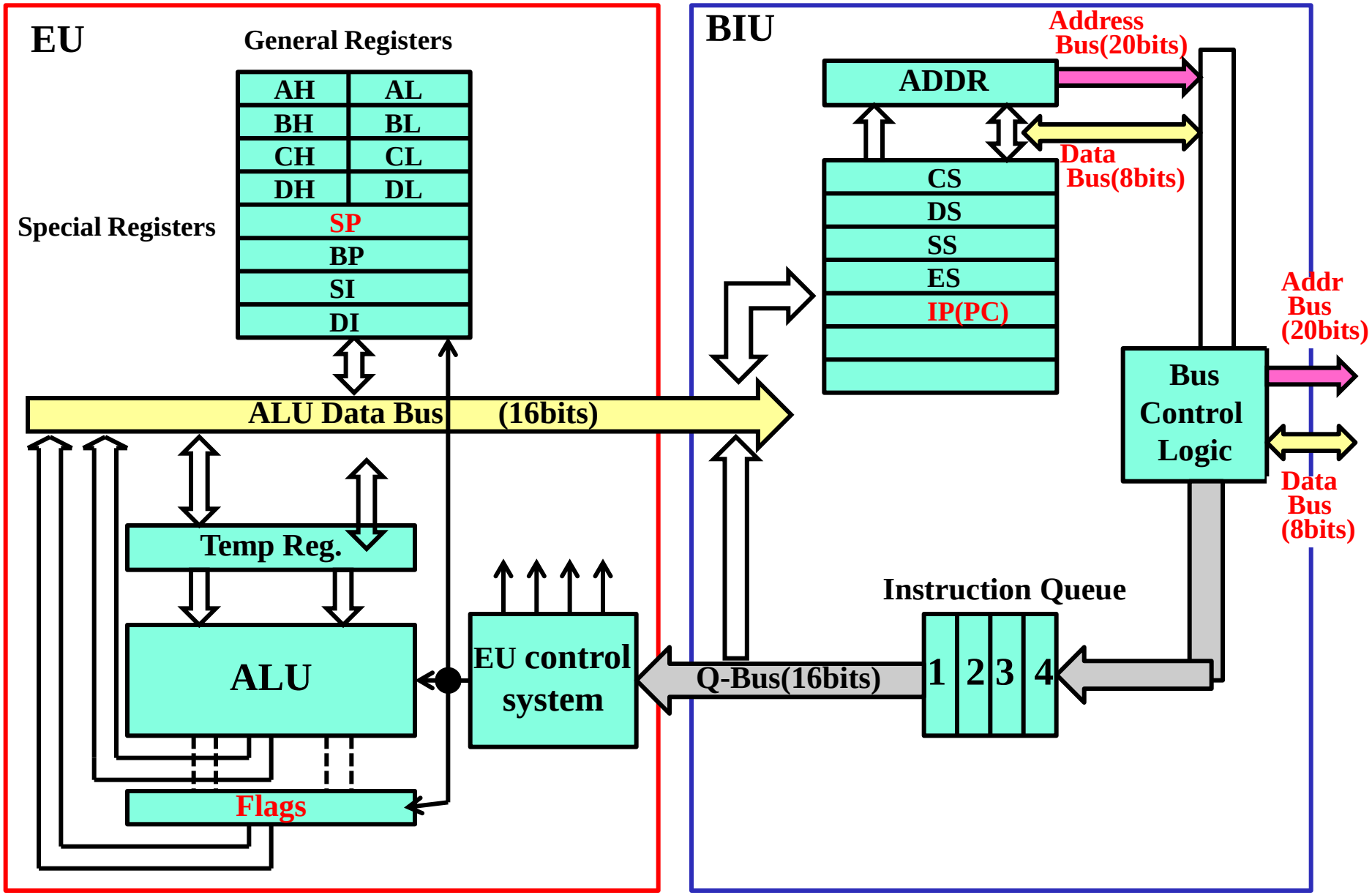
- **Serial fetch/execute (8 unit time)**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Fetch 1st inst | Execute 1st inst | Write result | Fetch 2nd inst | Execute 2nd inst | Fetch 3rd inst | Read operand | Execute 3rd inst |

- **Overlpped fetch/execute (6 unit time)**

| | | | | | |
|---|---|---|---|---|---|
| | Execute 1st inst | Execute 2nd inst | | | Execute 3rd inst |
| Fetch 1st inst | Fetch 2nd inst | Write result | Fetch 3nd inst | Read operand | Fetch 4th inst |

# (2) 8086/8088 H/W 구조



# <주요 용어 >

- **CPU: 프로그램 저장장치로부터 프로그램을 읽어와서 자동적으로 수행할 수 있도록 ALU와 제어장치를 갖춘 디지털시스템**
- **마이크로프로세서: 하나의 IC칩으로 된 CPU**
- **마이크로프로세서(CPU)의 내부 주요구성요소**

**1) 레지스터: 범용과 특수 레지스터**

**가) 범용레지스터: 프로그래머가 프로그램 도중 사용할 수 있는 메모리로서 외부 메모리보다 빠르게 데이터 access.**

**나) 특수레지스터: CPU가 전용으로 사용하는 CPU내 특수 용도 메모리.**

**2) PC(Program Counter): 다음에 읽어 들일 명령어의 위치를 저장하는 역할을 하는 특수 레지스터. Memory address 레지스터를 통해서 주소 값을 Address Bus에 전달.**

**-보통 8비트 CPU인 경우 16비트 PC**

**- 32/64 비트 CPU인 경우는 32비트 PC**

**3) Status Reg(Flag Reg):** 연산결과가 양수인지, 음수인지, 자리올림 발생하였는지 등을 표시하는 정보를 가지는 특수 레지스터.

**4) Stack Pointer: LIFO (Last In First Out)** 형태로 운영하는 **stack** 메모리 공간을 활용할 때 마지막에 입력된 데이터가 들어 있는 곳을 가리키도록 사용하는 레지스터. 인터럽트/함수 **call** 등에 사용되는 중요한 특수 레지스터.

**5) 제어장치: Instruction Decoder/Control signal Sequencer**를 포함. 명령을 해석하고 실행하는데 필요한 **CPU** 내부의 각 **unit** 사이의 데이터 흐름을 제어.

**6) Bus:** 마이크로프로세서 내의 각 **unit** 간, 마이크로프로세서와 메모리 및 주변 **unit** 들과의 정보교환을 위한 전송로**(Data /Address /Control Bus).**

**7) ALU:** 덧셈/뺄셈, 자리이동**(shift),** 비트논리연산**(AND/OR/NOT)** 등의 산술/논리연산장치 .

## (3) 버스 인터페이스 장치 (BIU)

: 프로그램메모리에서 **EU(core)**와 독립적으로 **command**를 가지고 오도록 버스를 관리

- 구성 요소
  - **Instruction Queue: Pre-fetch, pipeline**기능 가능,
    **EU**와 **BIU**동시 동작가능.
  - **Segment register (ES, CS, DS)**
  - **Instruction pointer(IP) = PC(Program Counter)**
  - 주소 가산 블록

- 내부 데이터 버스와 **Q-bus** 가 **BIU**와 **EU**를 연결

**(4) 실행 장치 (EU)**

**: BIU가 가져온 명령어를 decoding하여 해당 제어신호를 발생하고 명령을 실행**

**(가) ALU**

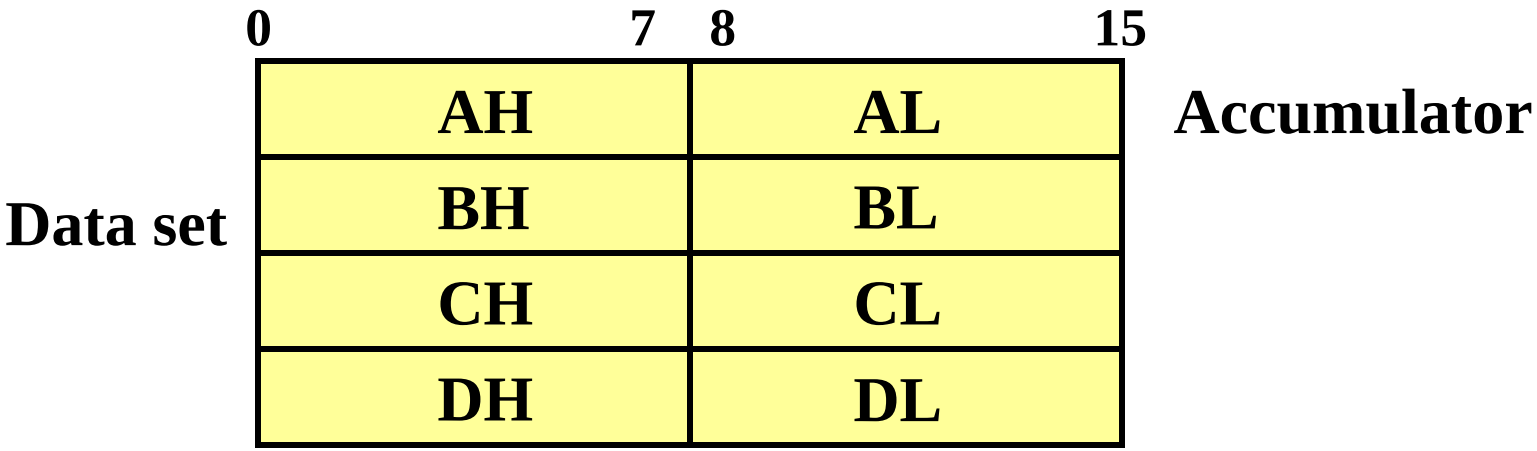
- **8bit or 16bit 연산**

**(나) 범용 레지스터**

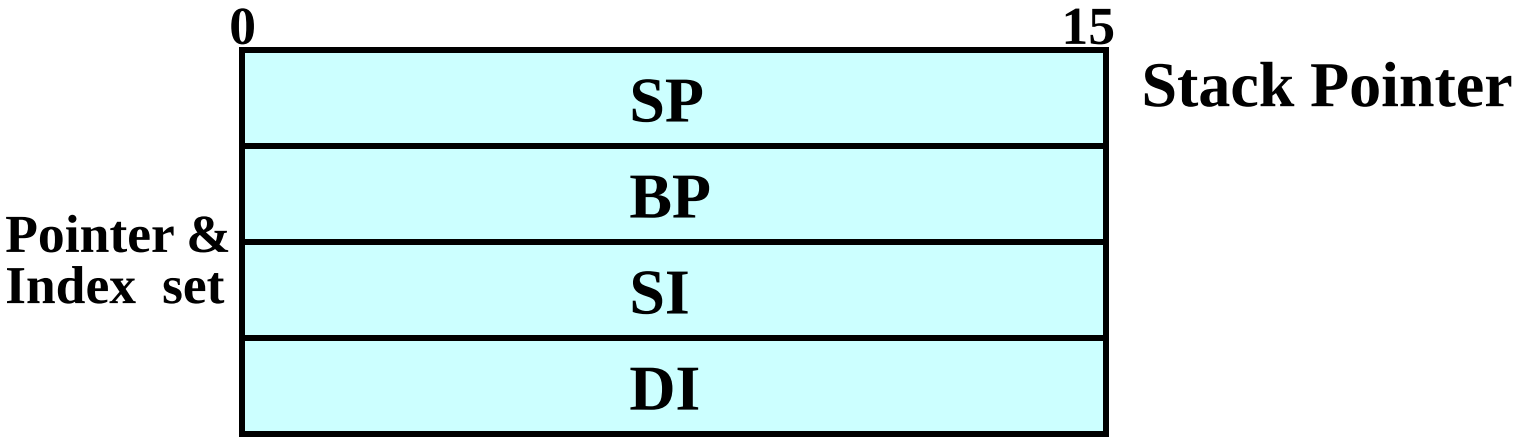
- **Data reg.: 프로그램의 실행 중 데이터나 처리 후의 결과를 저장하기 위하여 사용**
- **Pointer & Index reg.: EU의 제어 하에 메모리의 주소를 지정**
  **SP(stack pointer), BP(base pointer)**
  **SI(source index), DI(destination index)**

**(다) EU control system: 명령어 해석, 제어신호 발생 등**

- 범용 레지스터




- 특수 레지스터



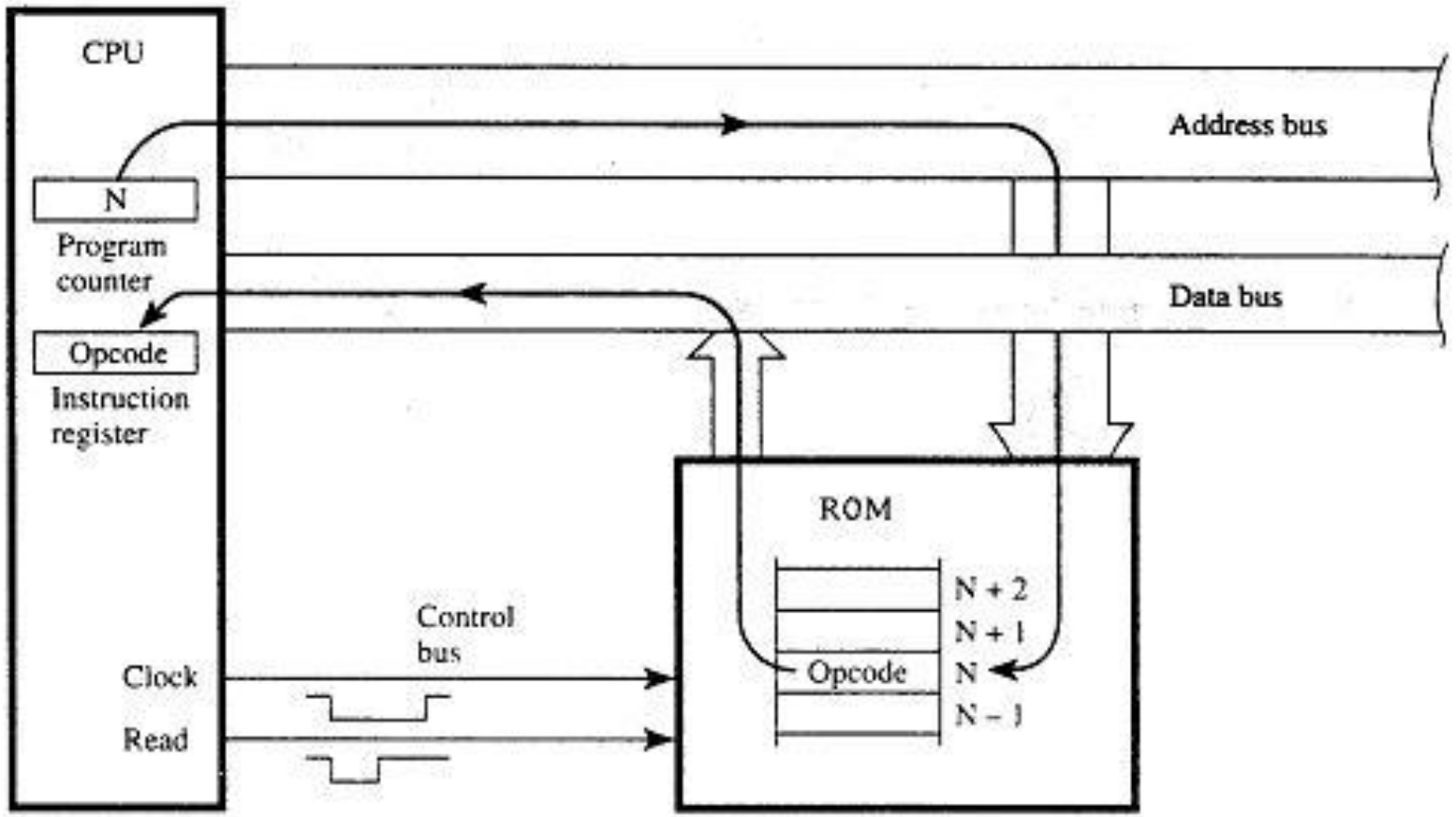
## (라) Flag

- **Status flag: ALU에서 수치 계산이나 논리연산 후의 상태 (carry 발생(CF), 결과 값 0(ZF), 결과값의 부호(SF) 등, ...)를 표시**

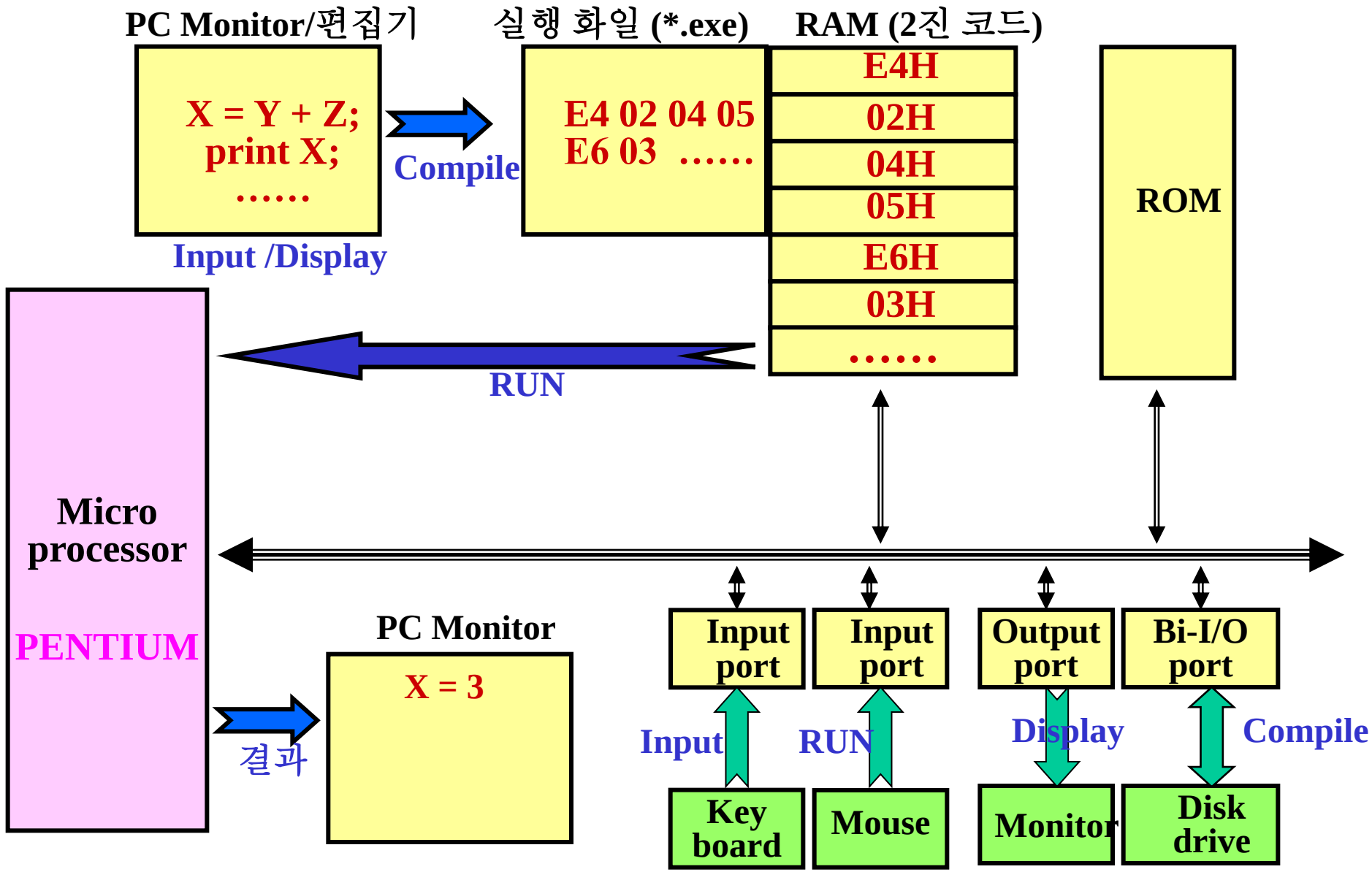
| TF | DF | IF | OF | SF | ZF | AF | PF | CF |
|---|---|---|---|---|---|---|---|---|

# 1.4 마이크로프로세서 프로그래밍

## CPU의 동작



# CPU의 동작


PC Monitor/편집기
X = Y + Z;
print X;
……
Input /Display
Compile
실행 화일 (*.exe)
E4 02 04 05
E6 03 ……
RAM (2진 코드)
E4H
02H
04H
05H
E6H
03H
……
ROM
RUN
Micro processor
PENTIUM
결과
PC Monitor
X = 3
Input port
Input port
Output port
Bi-I/O port
Input
RUN
Display
Compile
Key board
Mouse
Monitor
Disk drive

(1) 프로그램 예제

1. 포트 2로부터 숫자를 입력시킨다.
2. 이 숫자에 5를 더한다.
3. 포트 3으로 더한 값을 출력시킨다.

(단계1) IN 02H (기계어: E4H, 02H)

* IN: operand에 해당하는 포트(입력장치의 입력정보를 가지고 있는 reg.)로부터 데이터(1 byte)를 읽어와 AL reg.에 저장하는 명령어
* E4H : 특정 명령어에 대한 OP-code

(단계2) ADD 05H (기계어: 04H, 05H)

* ADD: AL reg.에 저장한 데이터와 operand를 더하고, 그 결과를 다시 AL reg.에 저장하는 명령어

(단계3) OUT 03H (기계어: E6H, 03H)

* OUT: AL reg.에 저장된 데이터(1 byte)를 operand에 해당하는 포트(출력장치의 출력정보를 내보내는 reg.)에 출력하는 명령어

- 위의 명령어들을 실행하기 위해 작업자는 명령어들을 memory에 다음과 같이 미리 저장해야 한다.

| | Memory | Address |
|---|---|---|
| IN | E4H | 00000 |
| Port 02H | 02H | 00001 |
| ADD | 04H | 00002 |
| 05H | 05H | 00003 |
| OUT | E6H | 00004 |
| Port 03H | 03H | 00005 |
| | | |

- 프로그램 실행

1. Fetch: IN op-code를 fetch
2. Decode: IN decode (-> 다음에 포트번호가 있다는 것을 인식 -> 해당 control 신호 발생)
3. Read Operand: Input port(02H)를 read
4. Execute: 입력포트에 포트 address와 'read' 신호를 전송하여 입력데이터를 읽어와 AL reg.에 저장한다. (예:07H)
5. Fetch: ADD op-code를 fetch
6. Decode: ADD decode (-> 다음에 operand가 있고, 이 operand를 읽어와 AL reg의 데이터와 더해 결과를 다시 AL reg에 저장해야 한다는 것을 인식 -> 해당 control 신호 발생)
7. Read Operand:Operand(예:05H)를 read
8. Execute: AL reg(예:07H)와 더한 후 결과(예:0CH)를 AL reg.에 overwrite

9. Fetch: OUT op-code를 fetch
10. Decode: OUT decode (-> 다음에 포트번호가 있다는 것을 인식 -> 해당 control 신호 발생)
11. Read Operand: Output port(03H)를 read
12. Execute: 출력포트에 포트 address와 'write' 신호를 전송하고 출력 데이터(예:0CH)를 출력한다.

(2) 명령어의 종류(8086/8088 의 경우)

(가) 데이터 전송: IN, OUT, MOV(reg.와 reg. 또는 memory 사이의 데이터 전송명령), PUSH(stack에 입력), POP(stack에서 출력), XCHG(exchange) 등

(나) 산술연산: ADD, SUB, MUL, DIV, INC, DEC, CMP(compare) 등

(다) 비트 조작

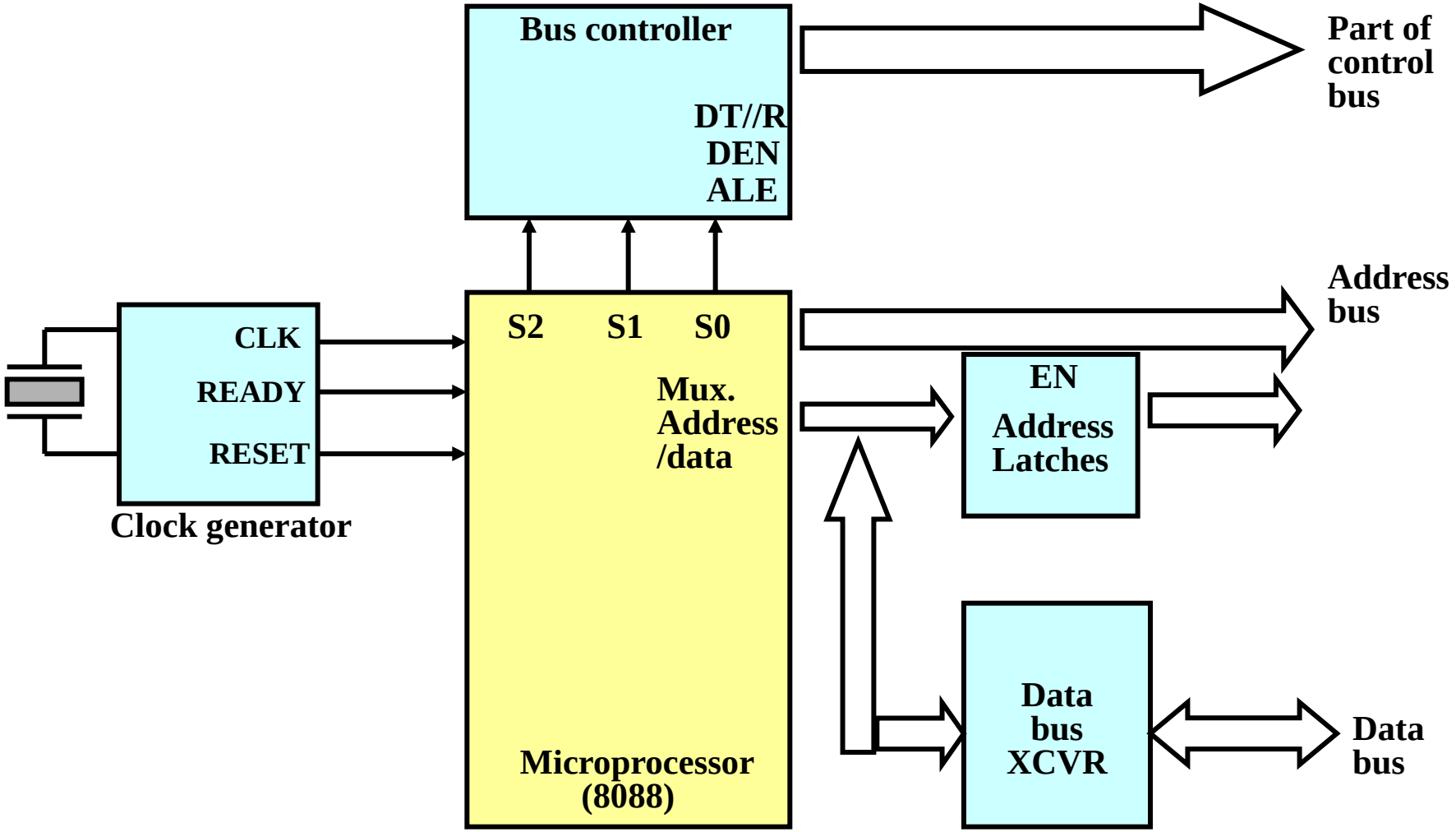
논리 연산: NOT, AND, OR, XOR, TEST 등

Shift : SAR(shift arithmetic right) 등

Rotate: ROL(rotate left) 등

# 1.5 CPU(중앙처리장치)

## (1) 8088 CPU을 사용한 컴퓨터



## (2) 8088 마이크로프로세서

- **Data bus:** 내부**(16bit),** 외부**(8bit)**
- **Address bus: 20bit(1Mbyte** 지정**)**
- 외부 어드레스 버스와 데이터 버스가 결합된 형태

**: multiplexing (bus controller**와 **address latch** 사용**)**

**(A8~A19** 는 어드레스 전용**, AD0~AD7**은 어드레스와 데이터 겸용**)**

## (3) 클럭 발생기

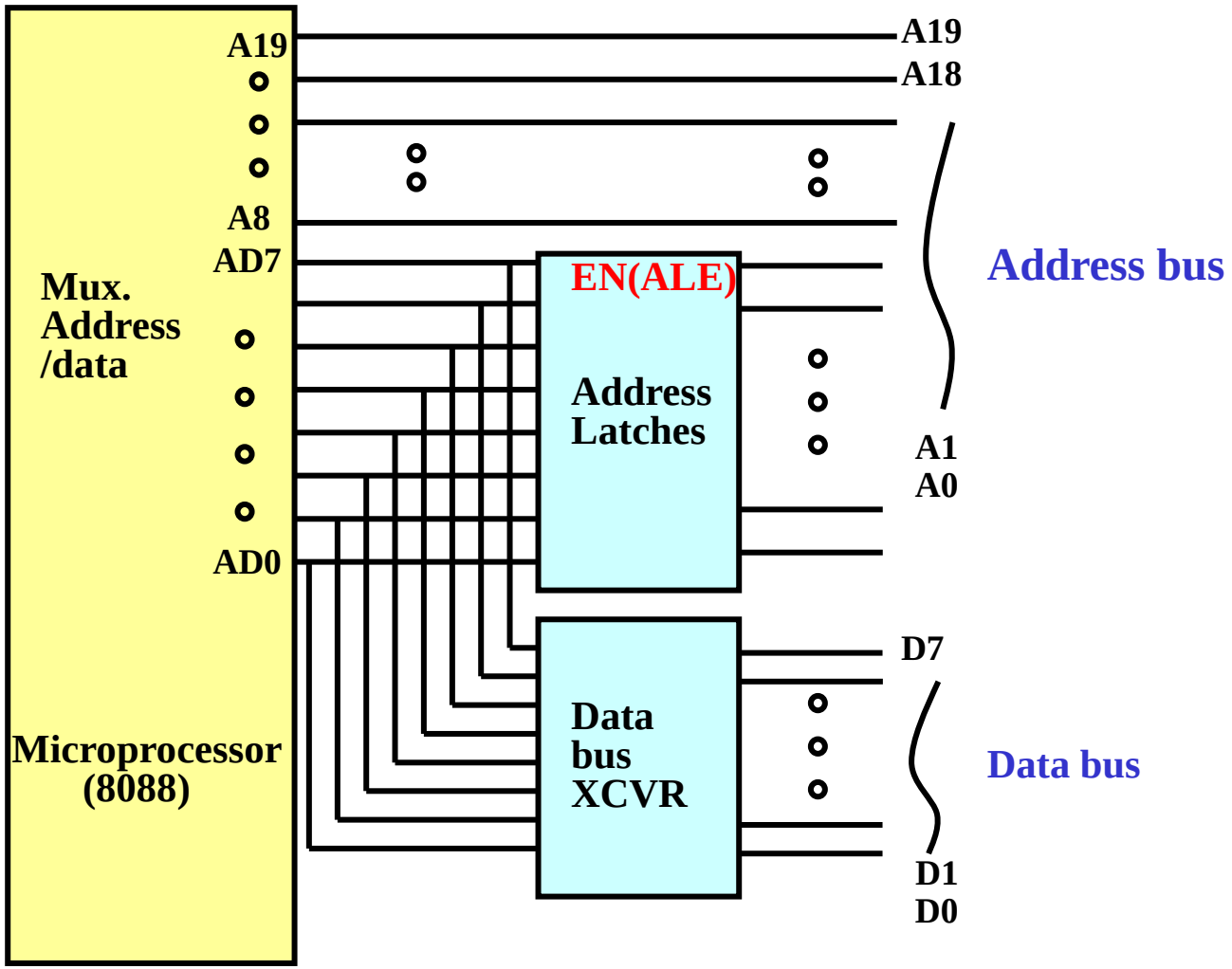
- **8088**에 **clock** 공급
- **RESET** 신호 발생

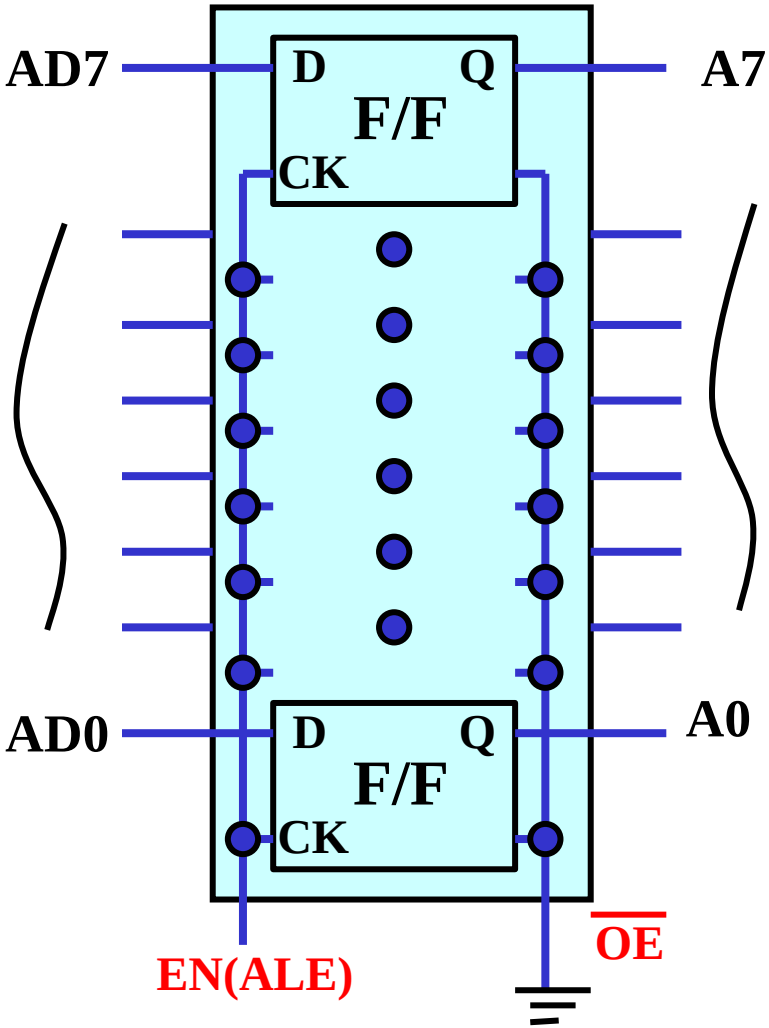
## (4) Address latch

- 필요성: **Address bus**와 **Data bus**가 **Multiplexing** 되어 두 버스의 충돌을 예방
- 동작: **Address** 신호가 먼저 발생하면 **Address latch**가 **ALE (Address Latch Enable)**신호를 받아 어드레스를 저장(통과) 하고, 데이터 신호가 발생하기 전에 **disable** 하여 **latch** 입구를 막음.

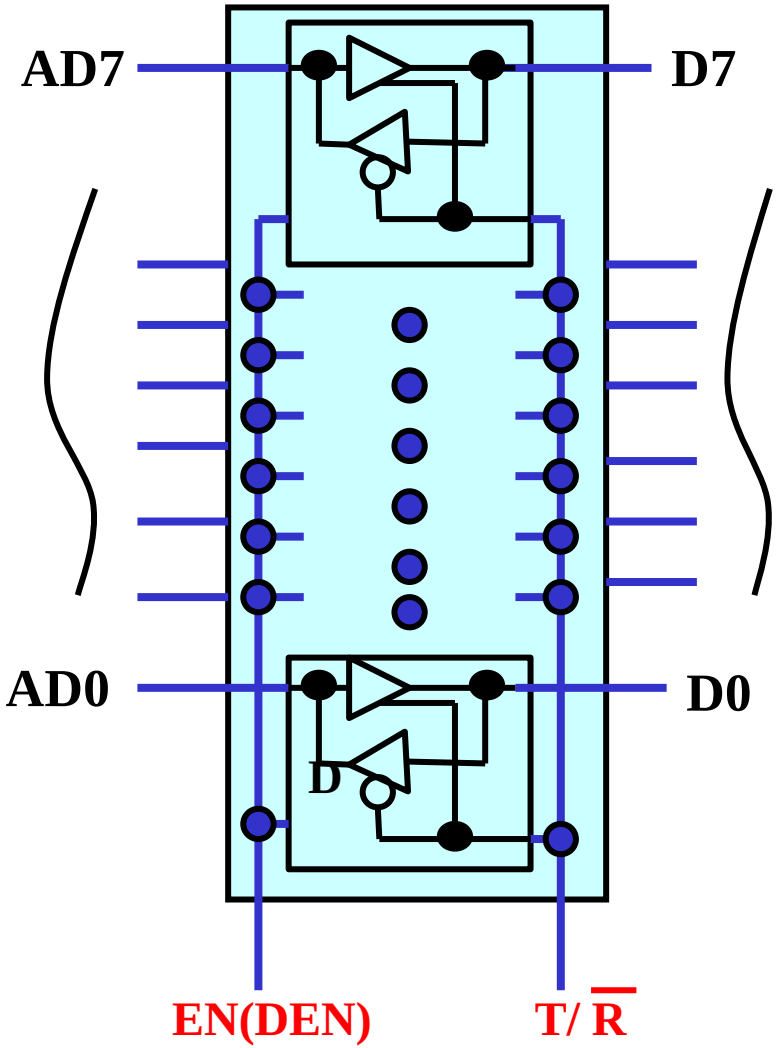
## (5) Bus transceiver

- 기능: **Data bus** 방향 제어
- 동작: 데이터 **Read**시 데이터버스 방향을 **Receive**
  데이터 **Write**시 데이터 버스 방향을 **Transmit**


A19
A8
AD7
AD0
Mux.
Address
/data
Microprocessor
(8088)
EN(ALE)
Address
Latches
Data
bus
XCVR
A19
A18
A1
A0
D7
D1
D0
Address bus
Data bus




<Address latch>




<Bus transceiver>

## (6) Bus controller(버스 제어기)

### (가) 기능

- 마이크로프로세서로부터 들어온 **AD0~AD7** 버스 선을 **latch (어드레스 latch)**
- 메모리나 **I/O**의 **READ/WRITE** 신호 발생
- 데이터 버스 제어 신호 발생

### (나) 작동

- 마이크로프로세서로부터 들어온 **/S0, /S1, /S2** 상태선의 코드에 따라 제어신호를 발생

**(다) 어드레스 latch**

- 마이크로프로세서가 메모리나 **I/O**를 **access**하고자 할 때 버스 제어기에서 **ALE** 신호(**8**개의 어드레스 **bit**들을 **latch** 하기 위한 신호)를 발생시키는 상태 코드를 발생 (**latch IC**는 현재의 유효한 어드레스를 어드레스 버스를 통해 유지)

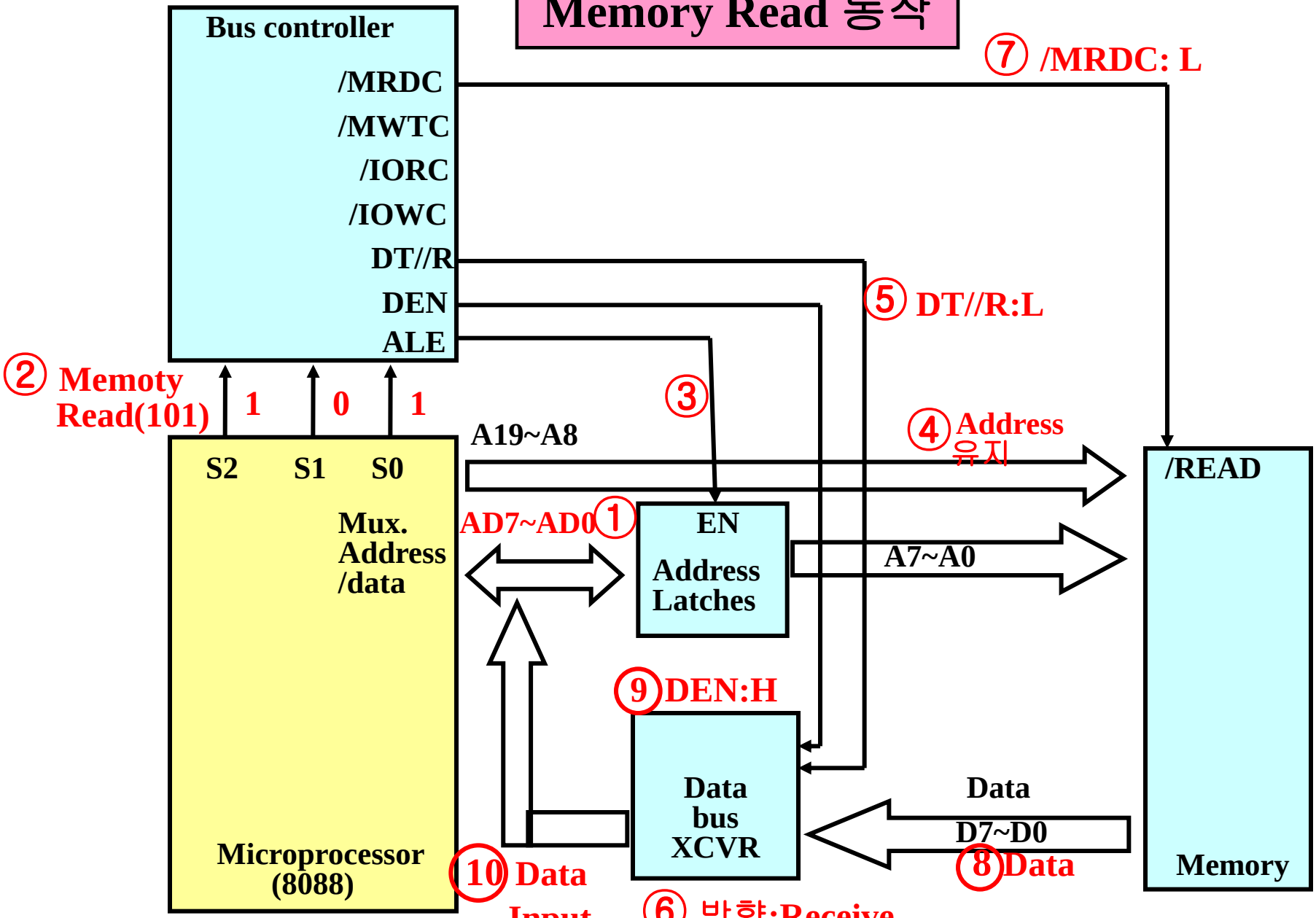
**(라) READ/WRITE 신호 발생**

- 어드레스가 **latch**에 저장되면 **READ**나 **WRITE** 신호를 메모리(**/MRDC, /MWTC**)나 **I/O**(**/IORC, /IOWC**)에 보내기 위해 상태 신호를 전송

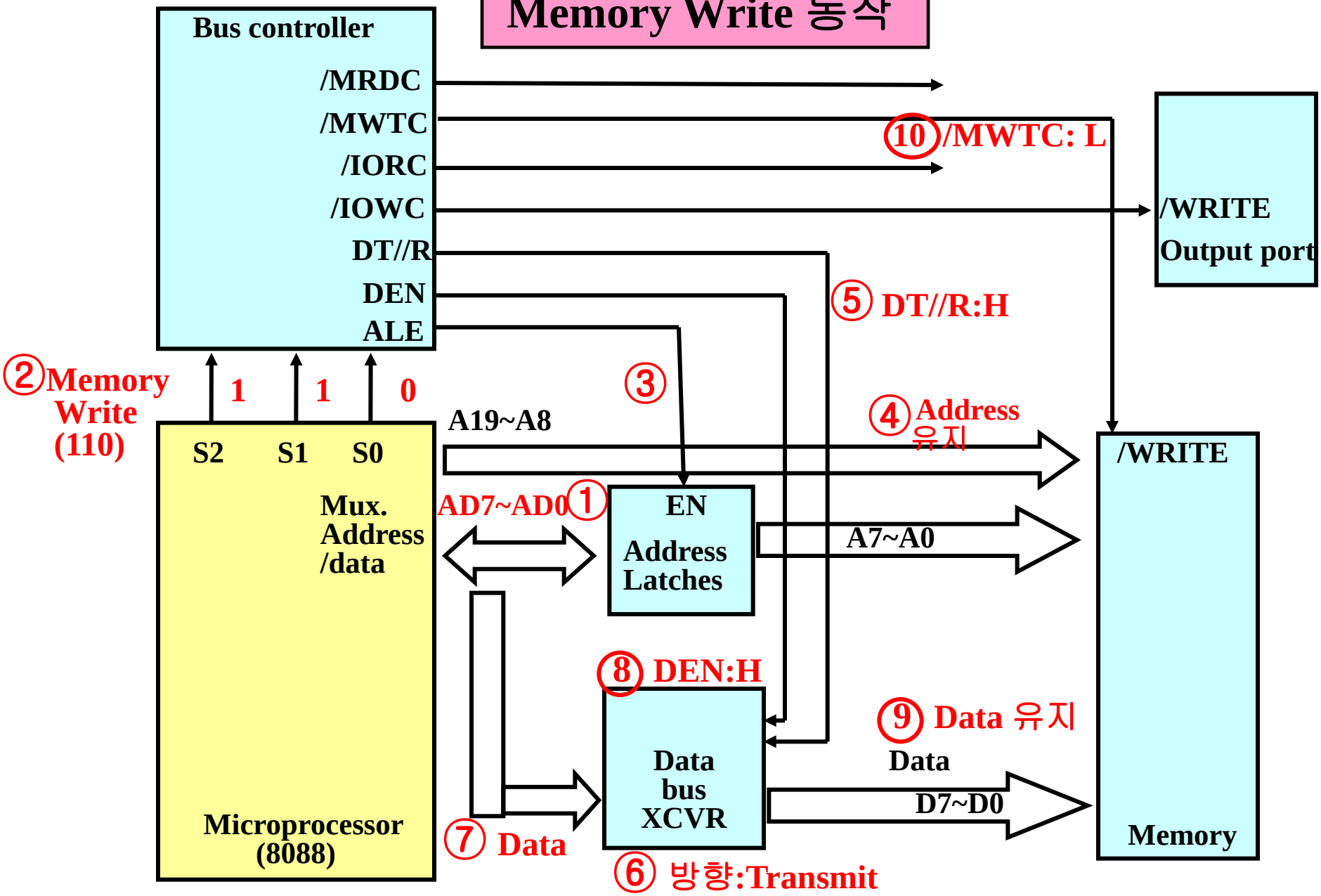
**(마) 데이터 버스 송수신기(XCVR) 제어 신호 발생**

- **DEN(Data Enable):** 송수신기 **enable**
- **DT//R (Data Transmit/Receive):** 데이터 방향 설정

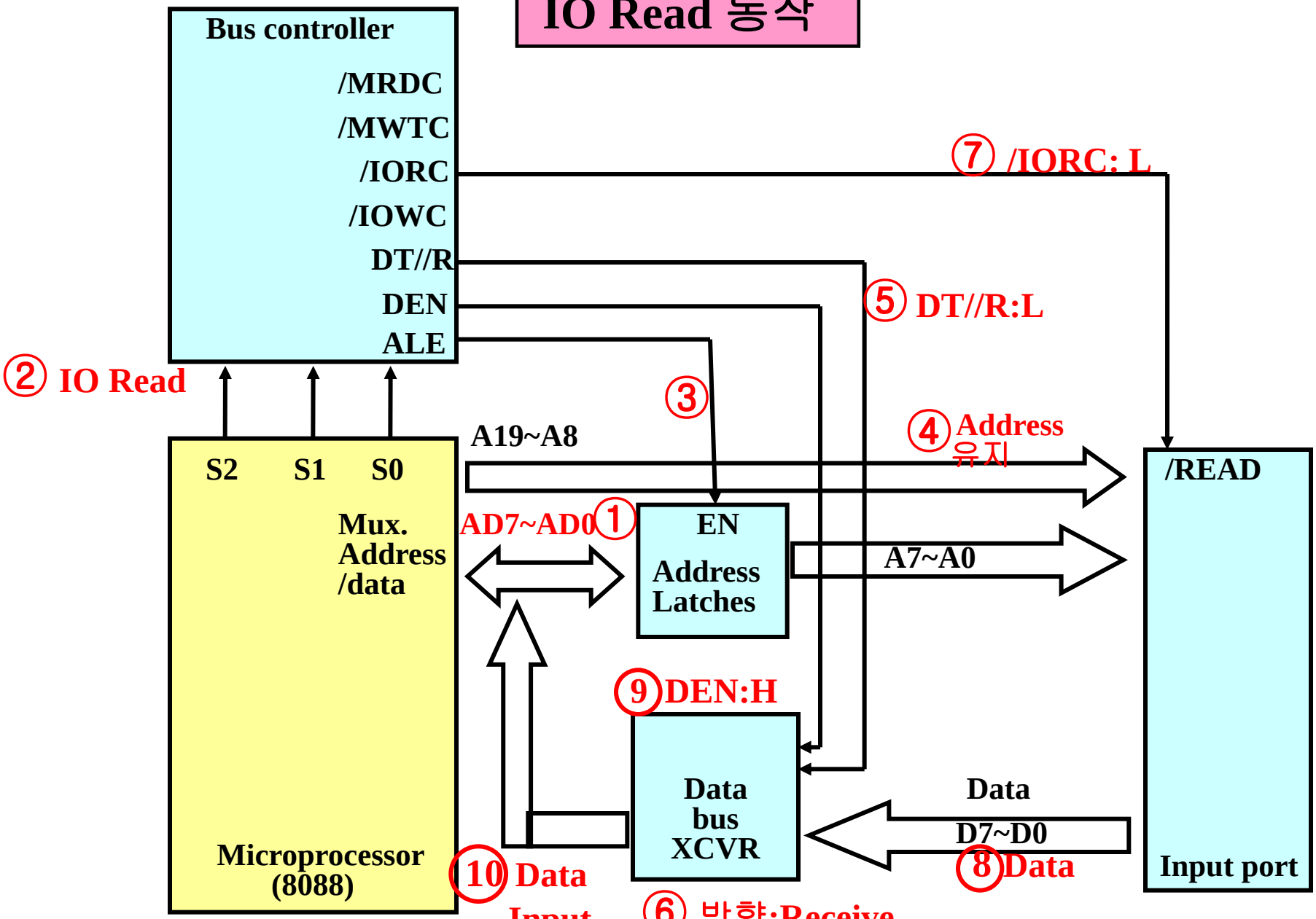
# Memory Read 동작



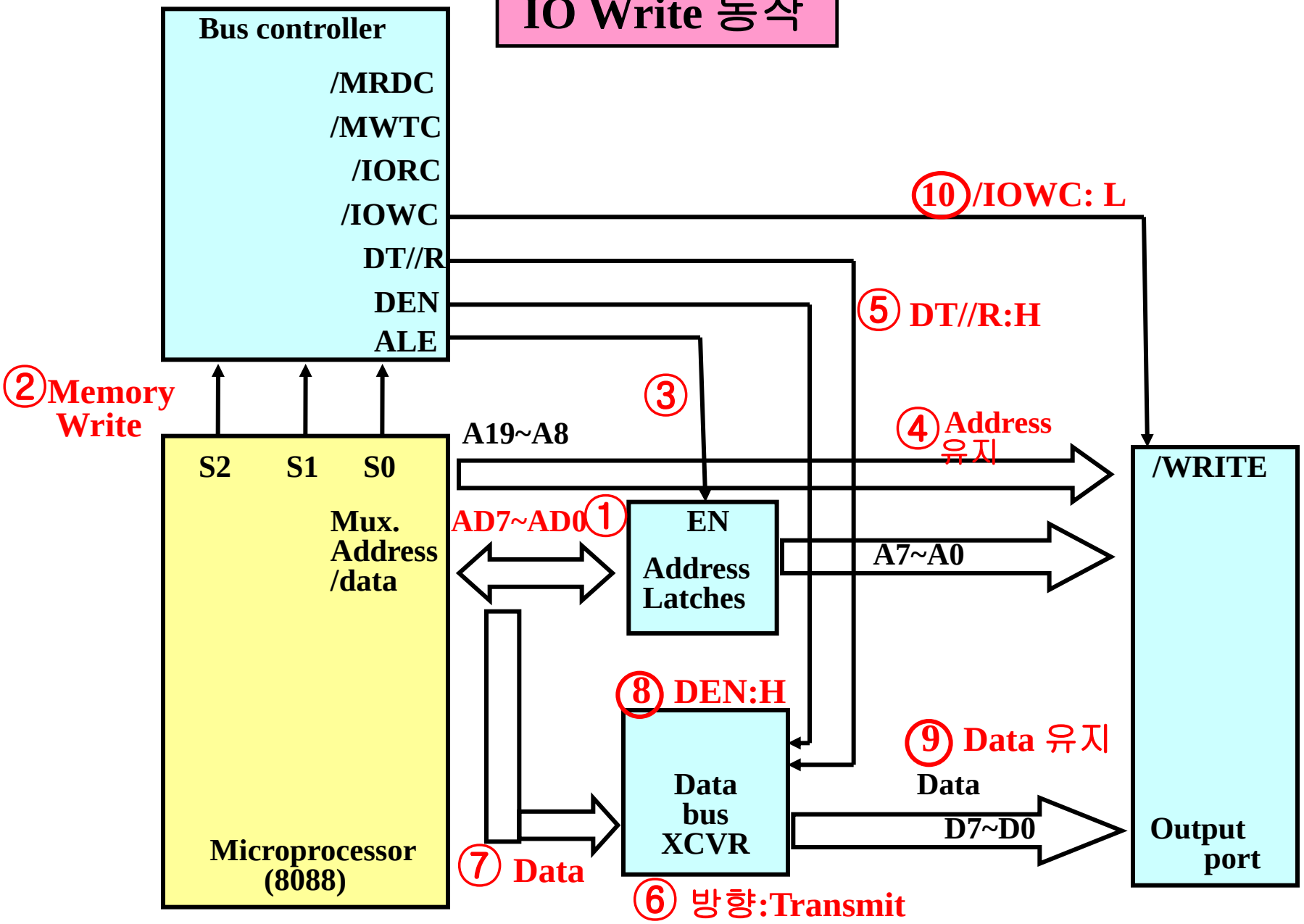
# Memory Write 동작



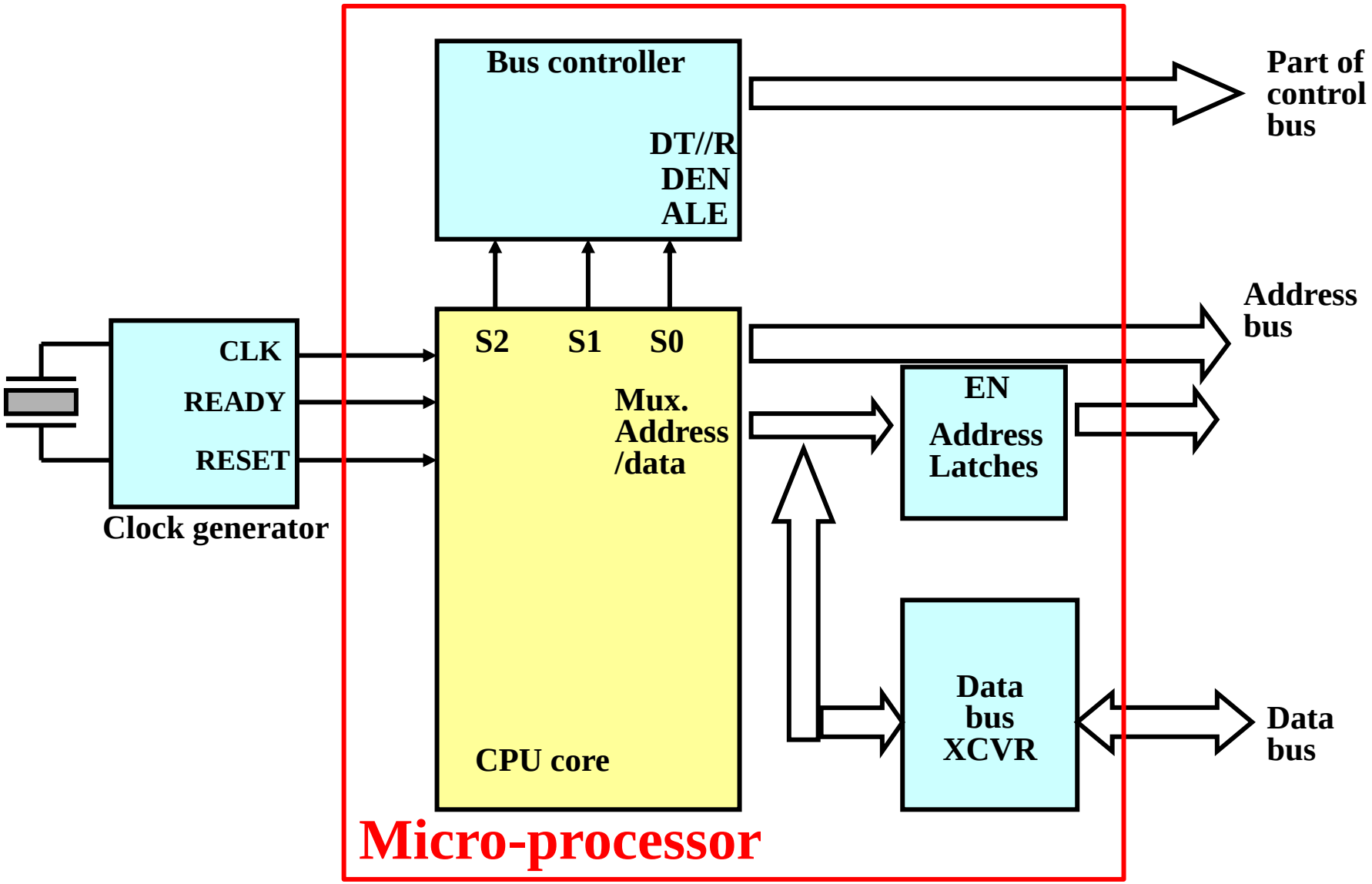
# IO Read 동작


Bus controller
/MRDC
/MWTC
/IORC
/IOWC
DT//R
DEN
ALE
⑦ /IORC: L
⑤ DT//R:L
② IO Read
③
④ Address 유지
A19~A8
S2
S1
S0
Mux. Address /data
AD7~AD0
①
EN
Address Latches
A7~A0
/READ
⑨ DEN:H
Data bus XCVR
Data
D7~D0
⑧ Data
Input port
Microprocessor (8088)
⑩ Data Input
⑥ 방향:Receive

# IO Write 동작

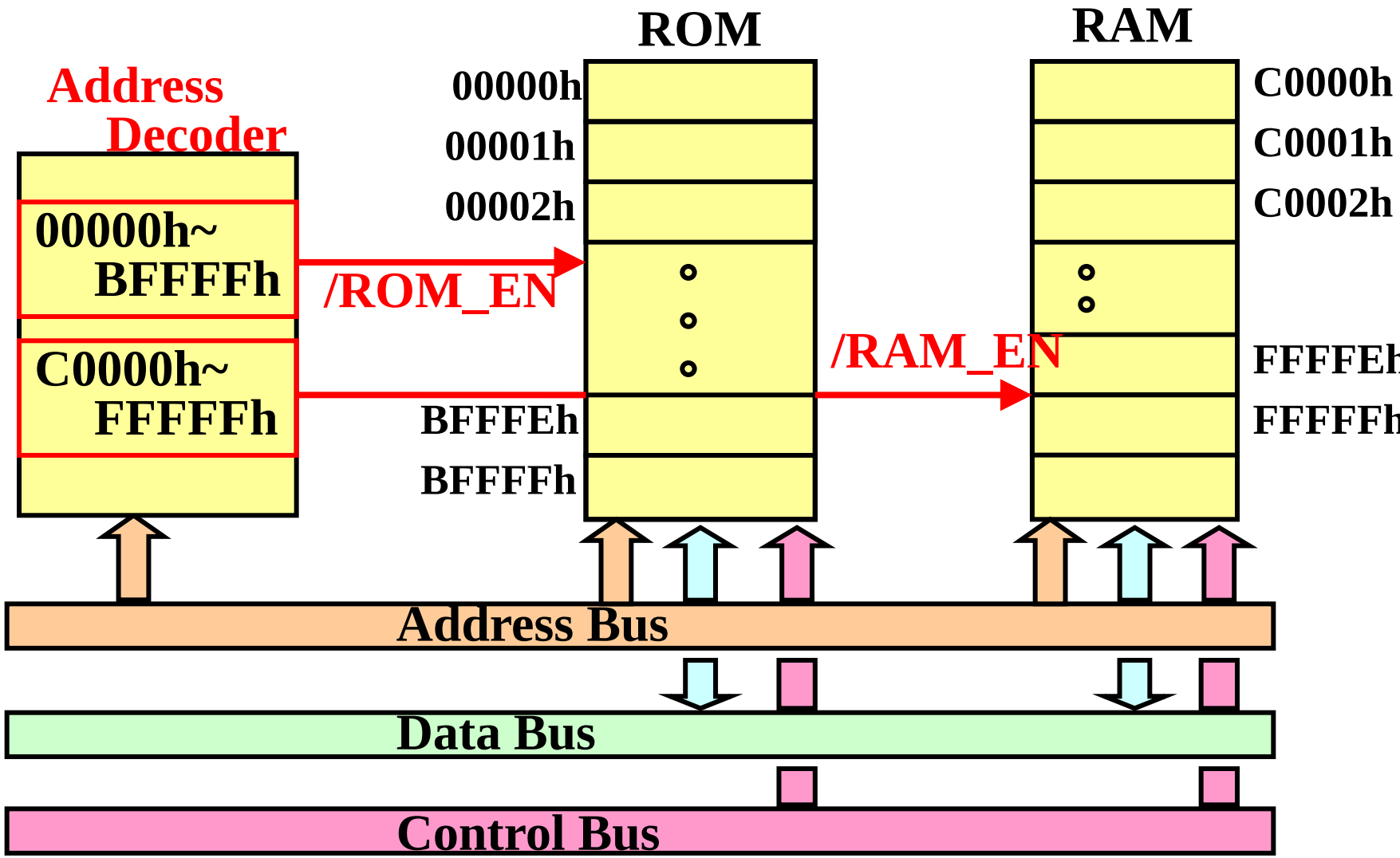

Bus controller
/MRDC
/MWTC
/IORC
/IOWC
DT//R
DEN
ALE
10 /IOWC: L
5 DT//R:H
2 Memory Write
3
4 Address 유지
A19~A8
S2
S1
S0
Mux. Address /data
AD7~AD0
1
EN
Address Latches
A7~A0
/WRITE
8 DEN:H
9 Data 유지
Data
Data bus XCVR
D7~D0
Output port
7 Data
6 방향:Transmit
Microprocessor (8088)

# * 최근 CPU 구조

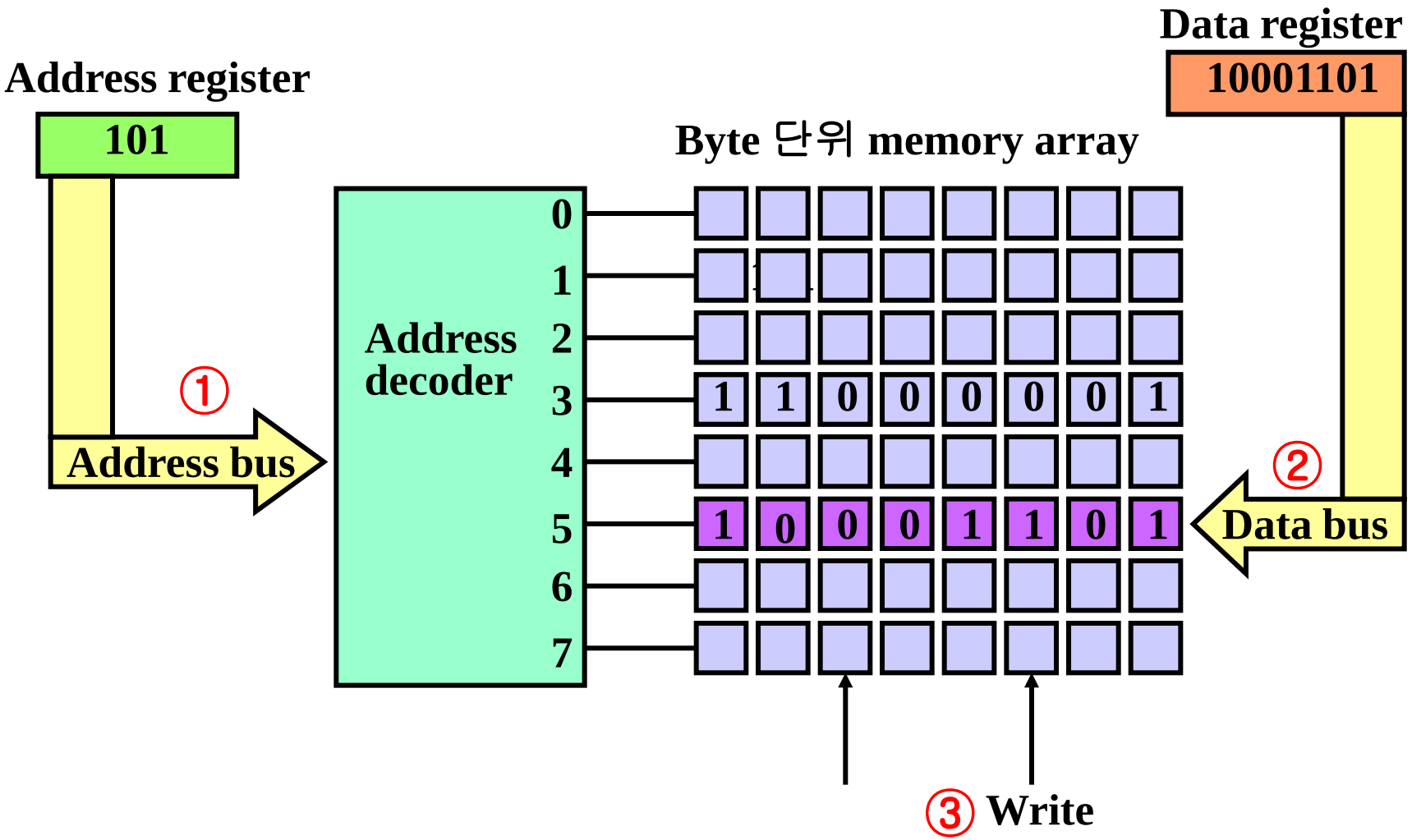

Bus controller
DT//R
DEN
ALE
Part of control bus
Address bus
CLK
READY
RESET
Clock generator
S2
S1
S0
Mux. Address /data
EN
Address Latches
CPU core
Data bus XCVR
Data bus
Micro-processor

# 1.6 메모리 장치

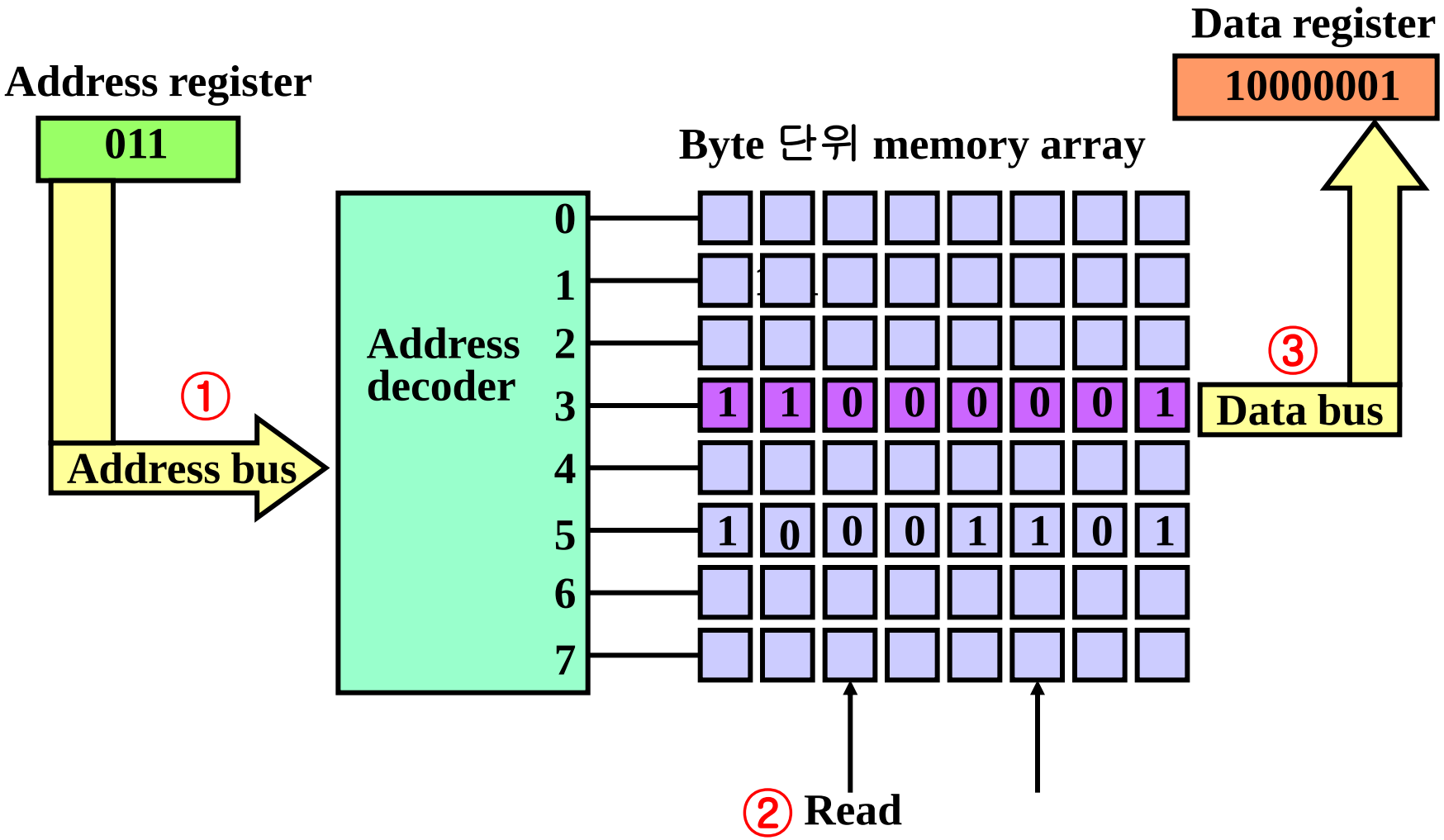
## (1) 주소 할당



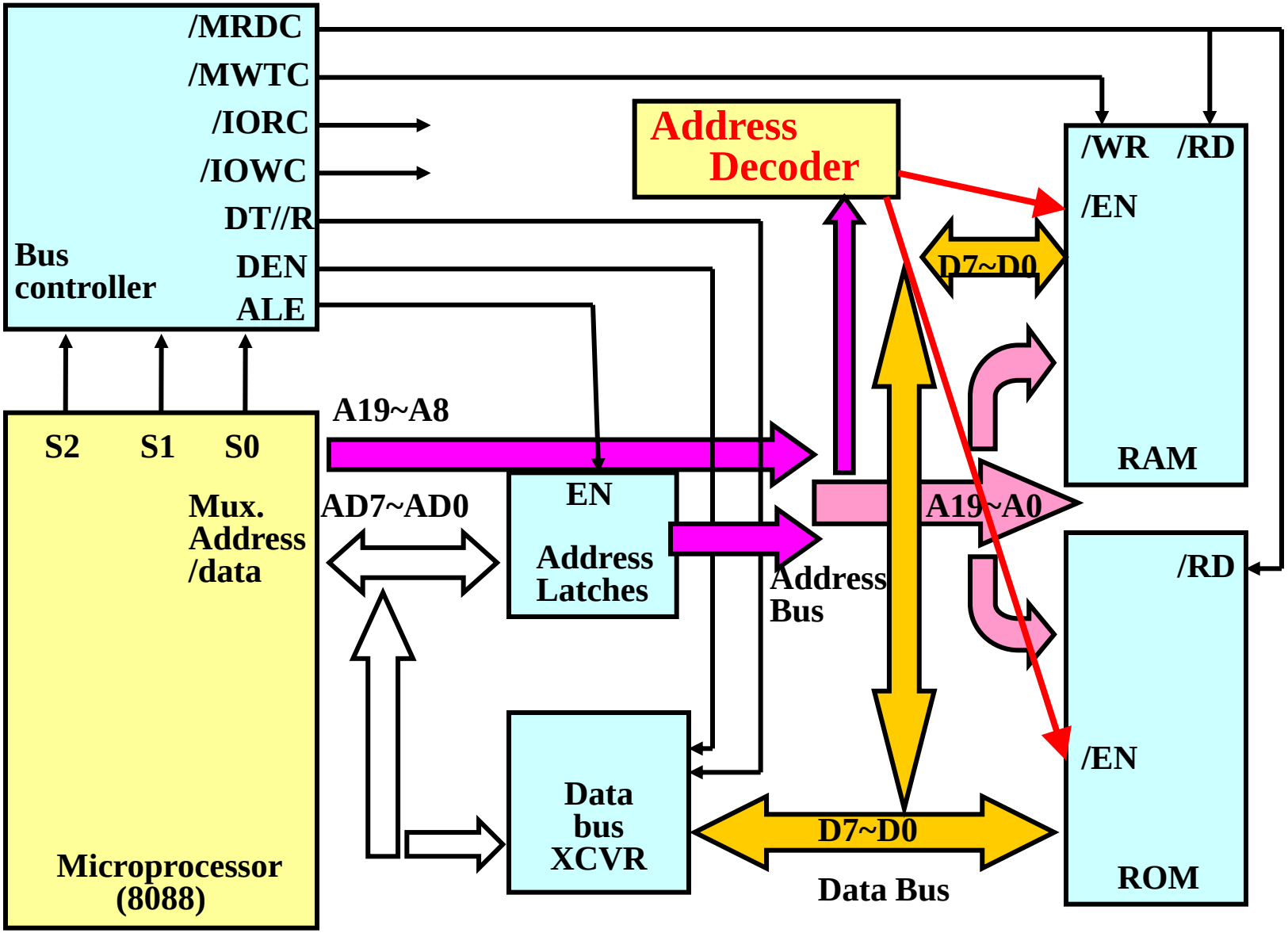
# (5) WRITE 동작


Data register
10001101
Address register
101
Byte 단위 memory array
Address decoder
0
1
2
3
4
5
6
7
1 1 0 0 0 0 0 1
1 0 0 0 1 1 0 1
①
Address bus
②
Data bus
③ Write

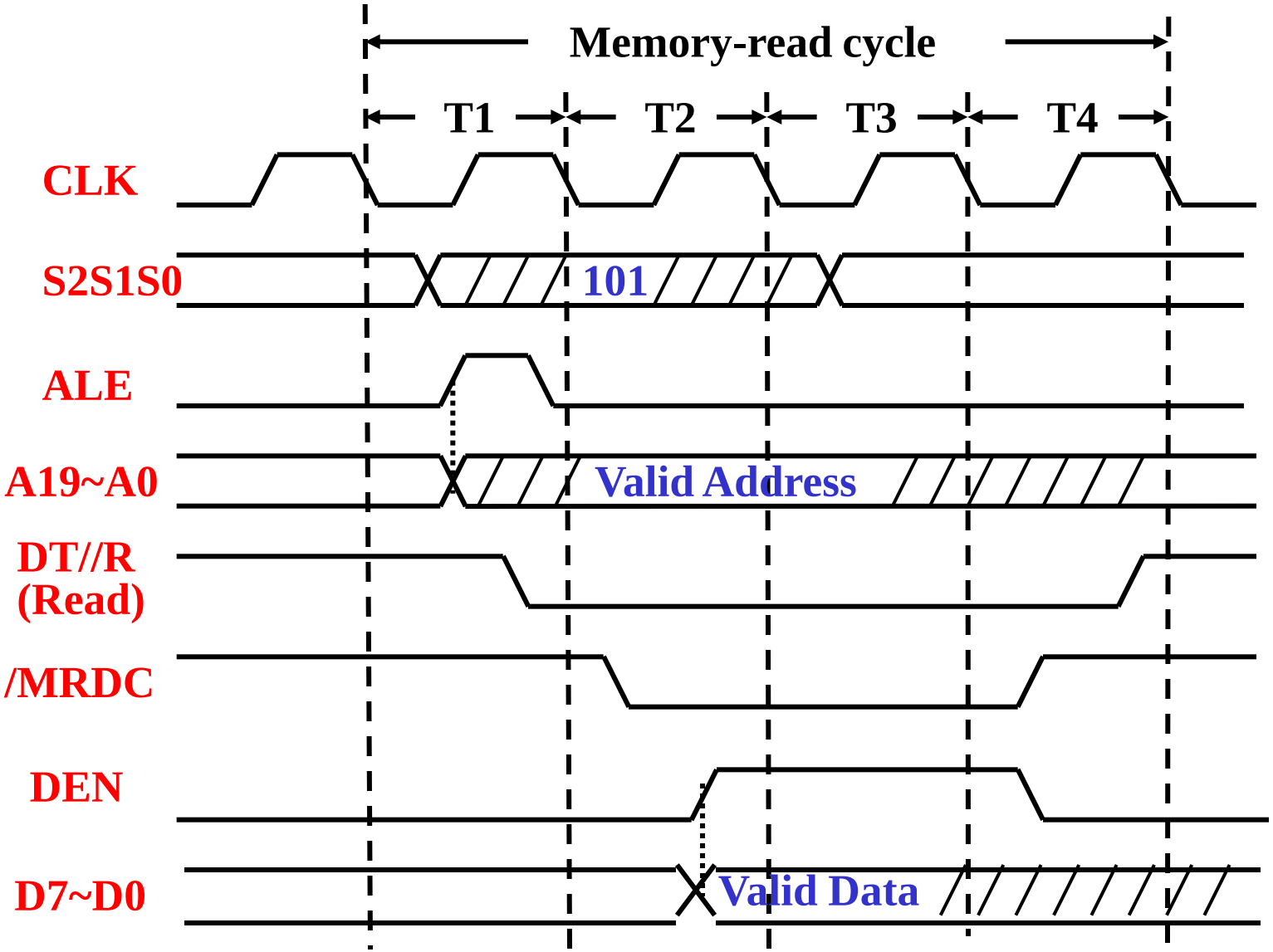
# (6) READ 동작


Data register
10000001
Address register
011
Byte 단위 memory array
Address decoder
0
1
2
3
4
5
6
7
1 1 0 0 0 0 0 1
1 0 0 0 1 1 0 1
③
Data bus
①
Address bus
② Read

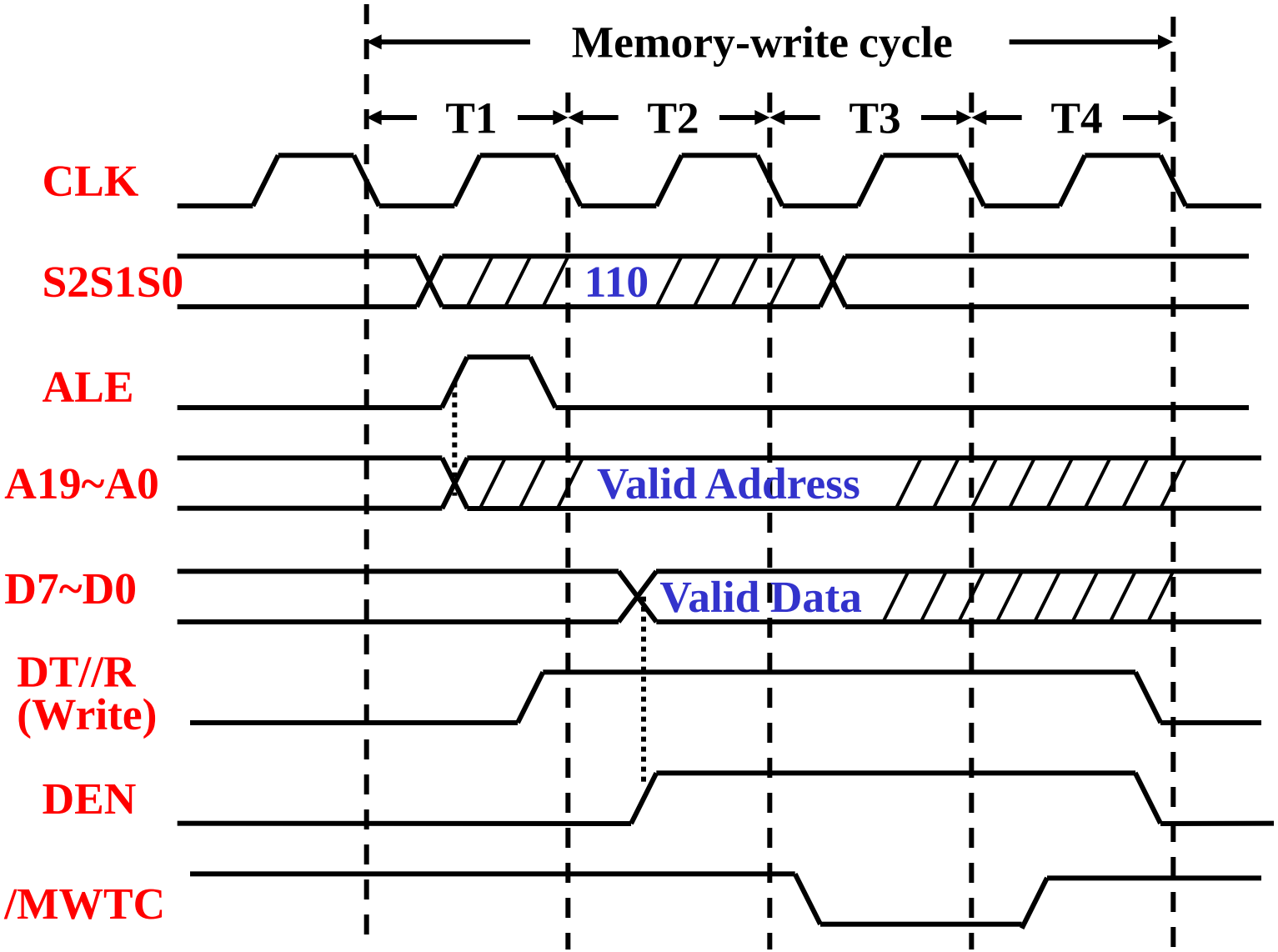
# (2) CPU 메모리 동작


/MRDC
/MWTC
/IORC
/IOWC
DT//R
DEN
ALE
Bus controller
Address Decoder
/WR
/RD
/EN
D7~D0
RAM
A19~A8
S2
S1
S0
Mux. Address /data
AD7~AD0
EN
Address Latches
A19~A0
Address Bus
/RD
/EN
Data bus XCVR
D7~D0
Data Bus
ROM
Microprocessor (8088)

# (가) 메모리 READ 사이클(Timing Diagram)



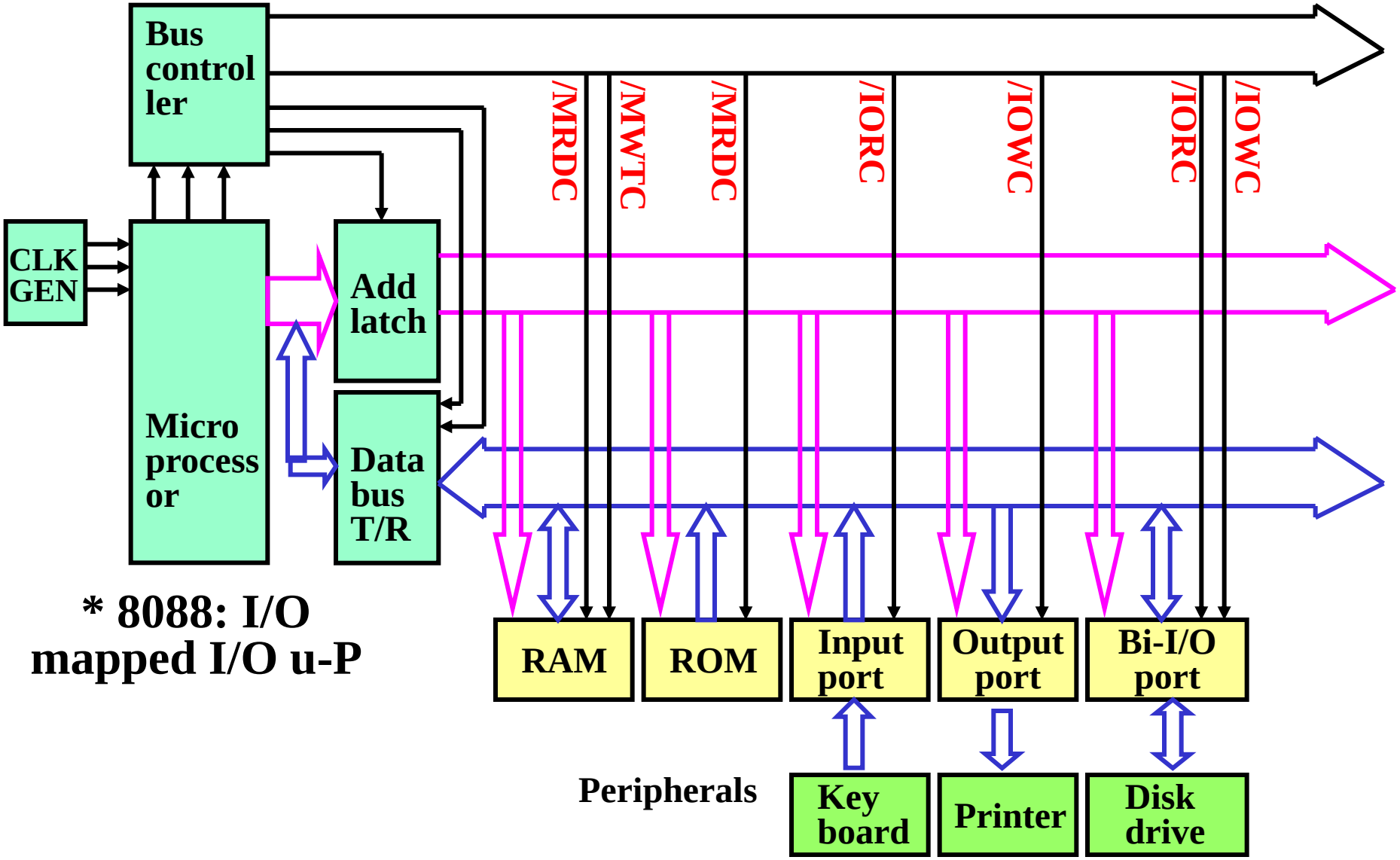
# (나) 메모리 WRITE 사이클



# 1.7 입출력 포트

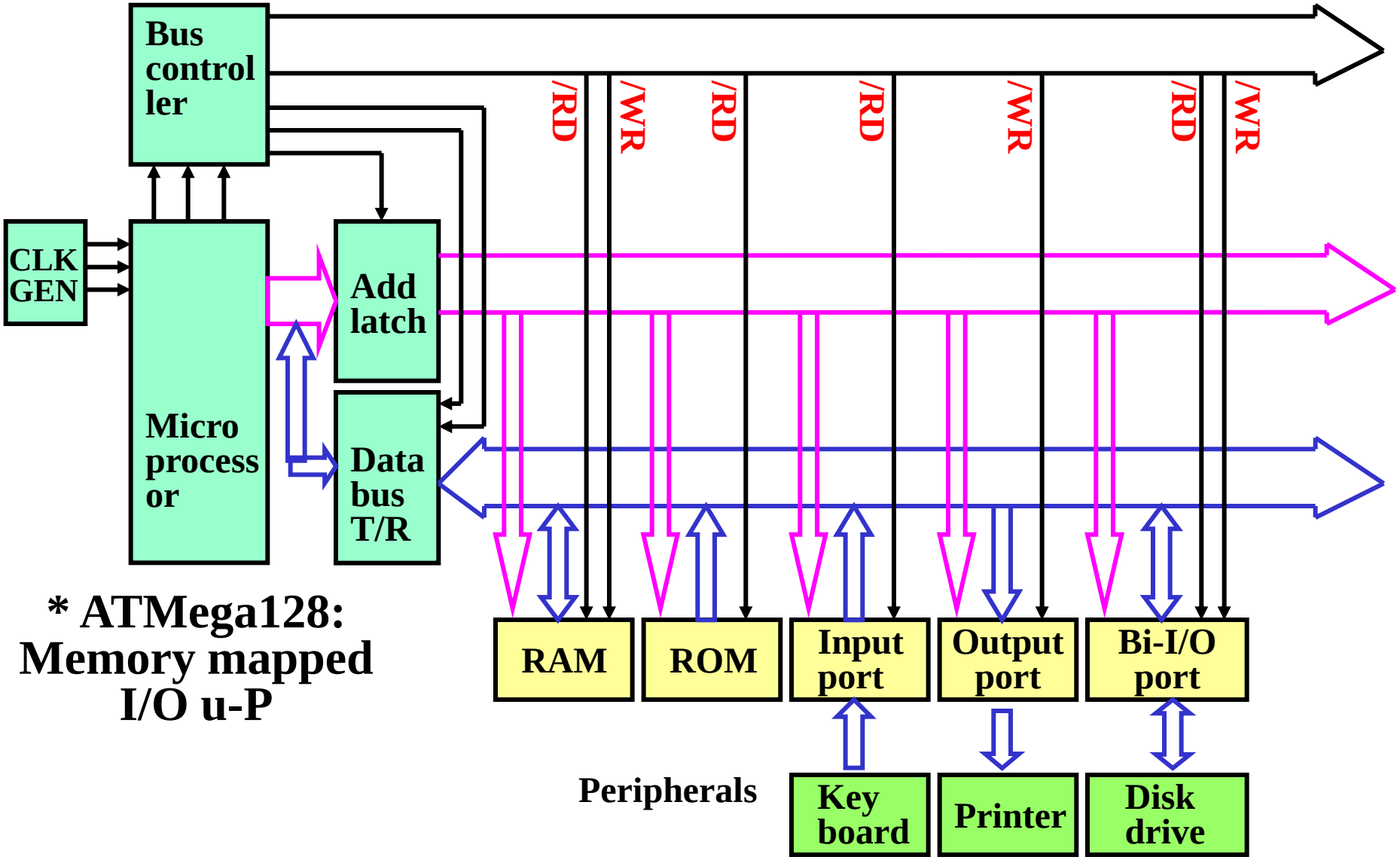
- **I/O 포트 정의: 컴퓨터와 외부 세계사이에서 정보를 전달(interface)하는 창문(window)**
- **종류: 입력전용, 출력전용, 입출력겸용**
- **주소 지정방식:**
  **전용주소 I/O 주소지정(I/O mapped I/O)**
  **메모리-맵 I/O 주소지정(Memory mapped I/O)**

# <Microcomputer>



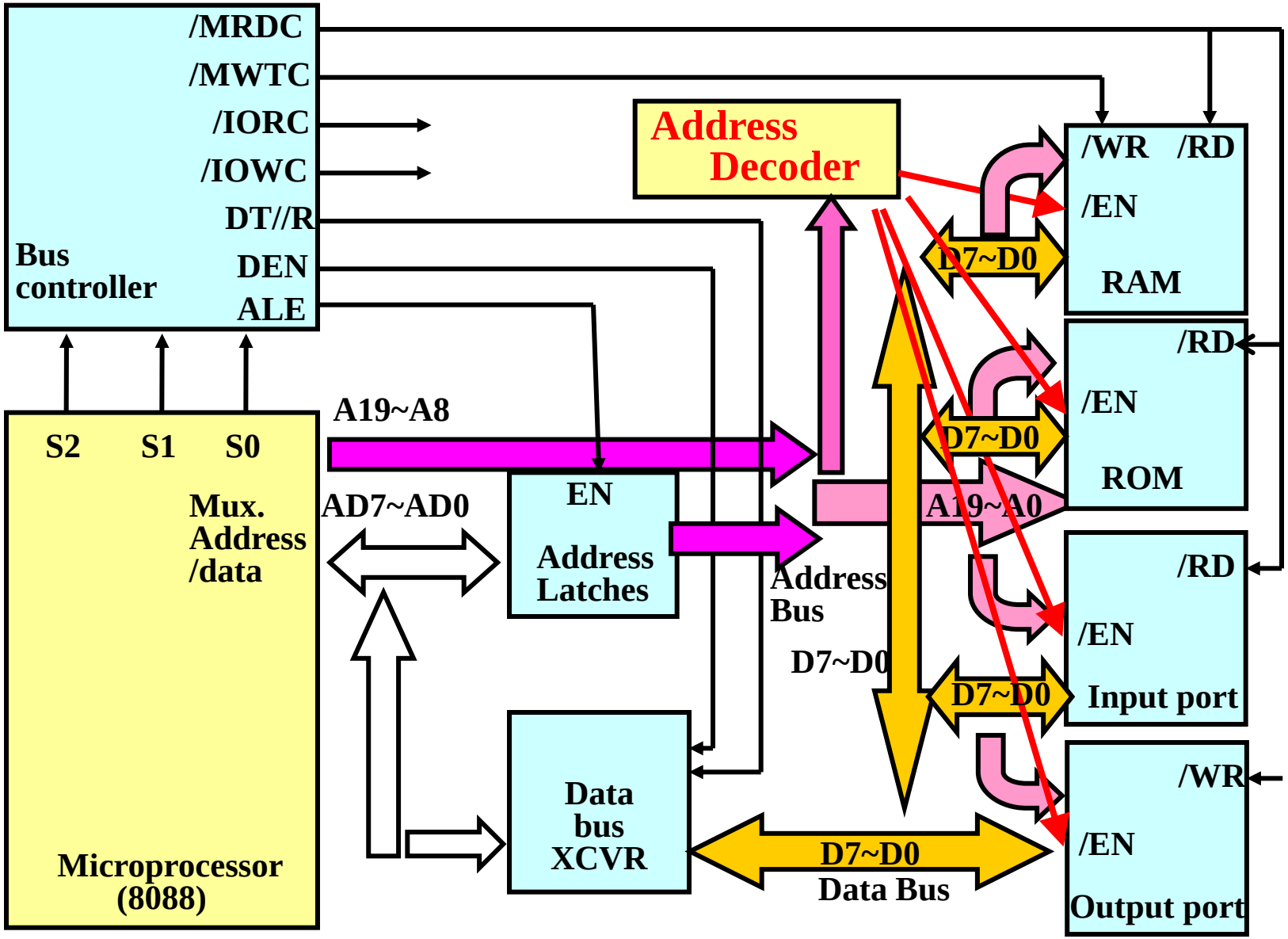

*** 8088: I/O mapped I/O u-P**

# <Microcomputer>



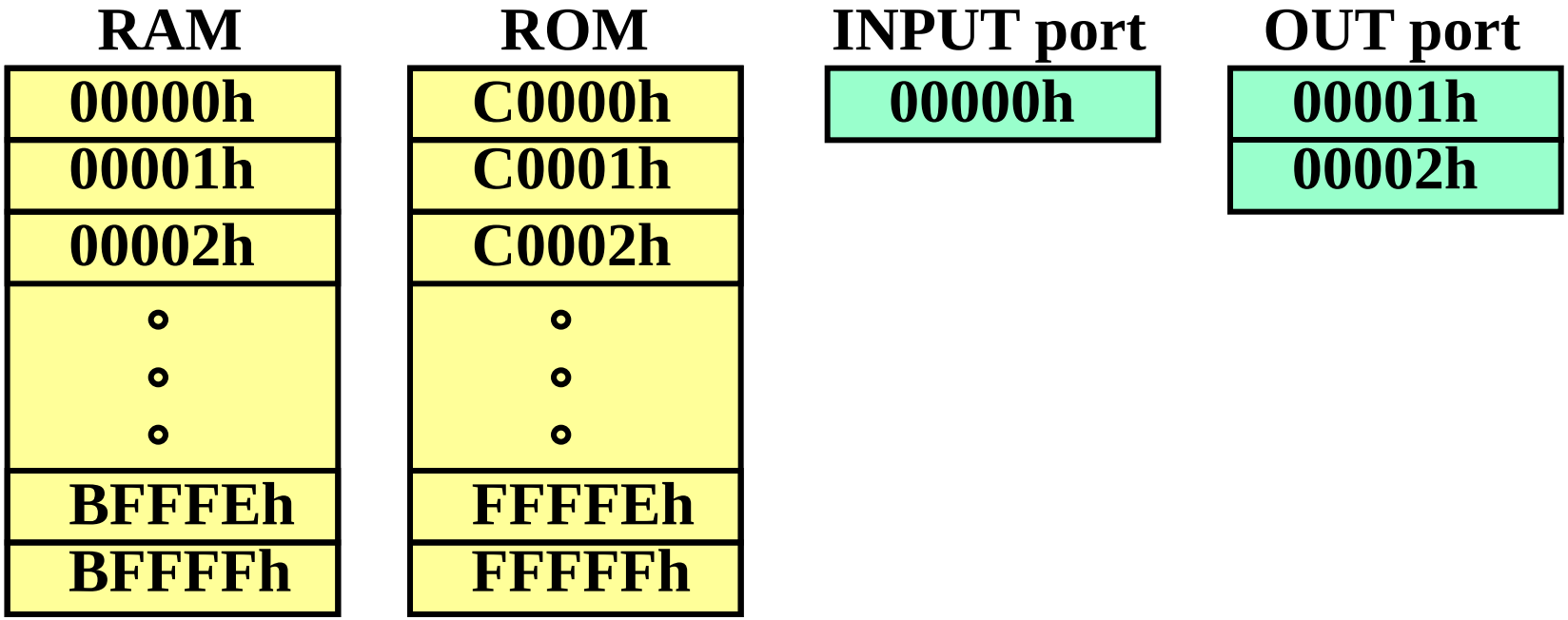

*** ATMega128: Memory mapped I/O u-P**

# (2) IO포트 및 Memory와Address decoder 연결

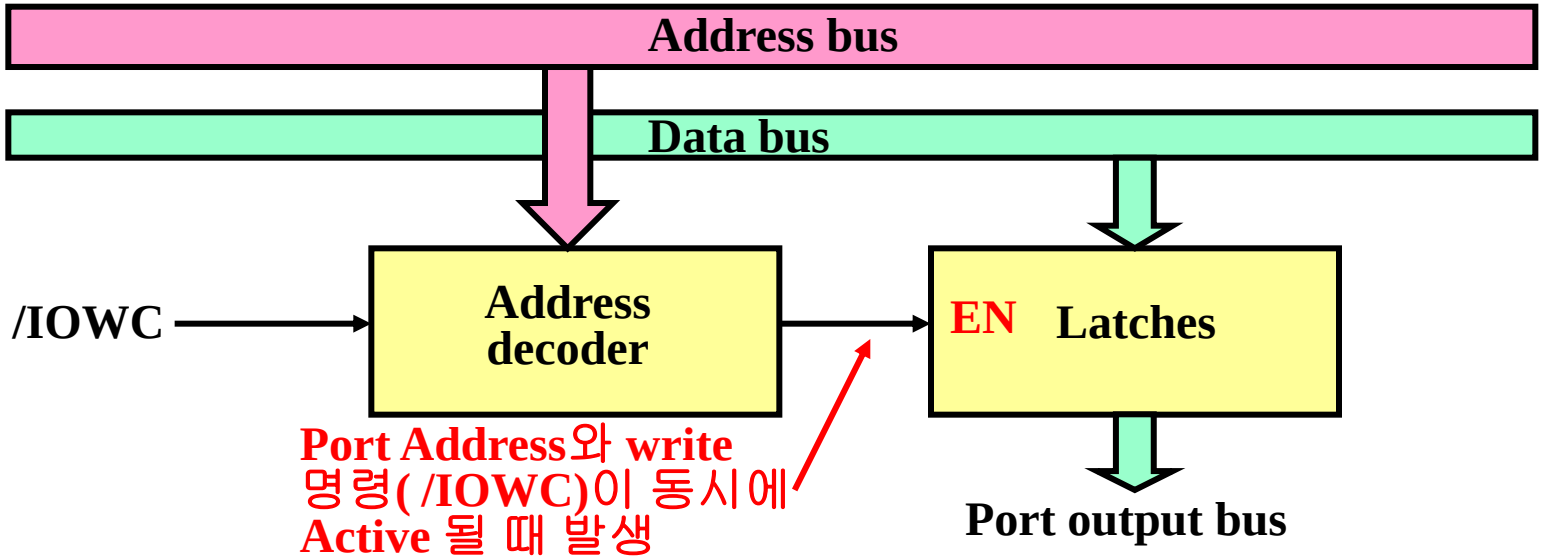


## (1) 전용주소 I/O 포트(예: INTEL u-P)

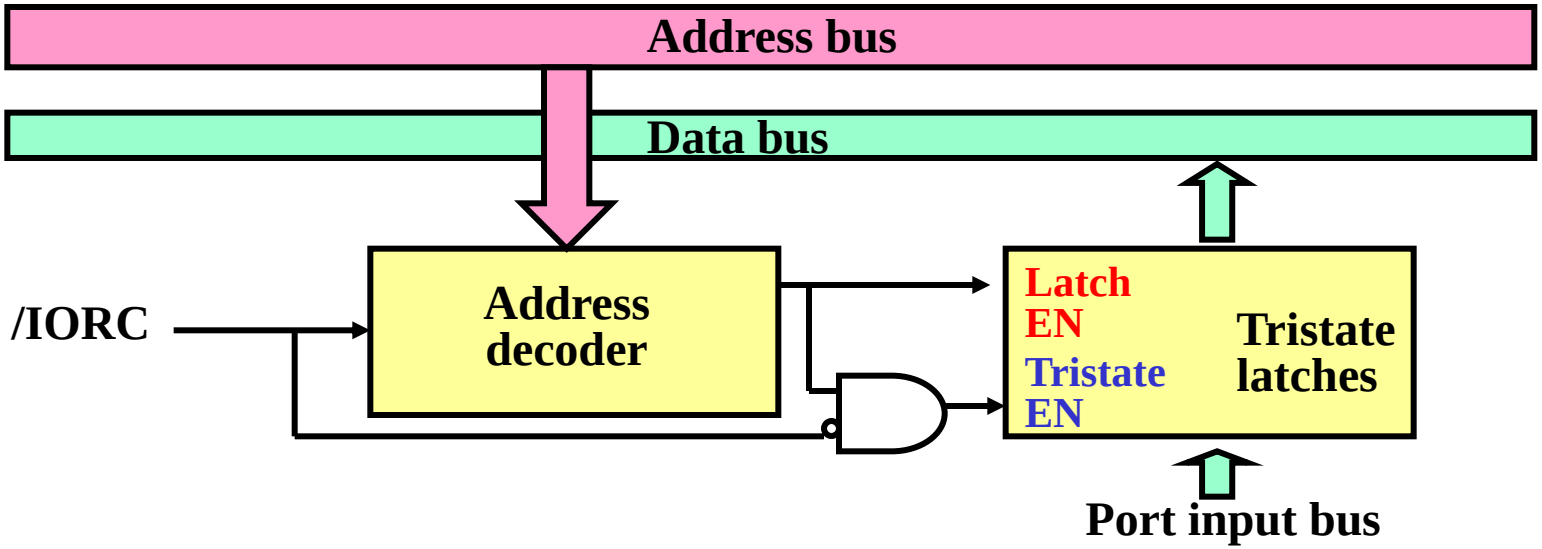
- I/O 포트에 전용주소를 할당.
- 메모리(RAM,ROM)와 주소 체계가 다름.
  즉, 메모리에 지정된 번지와 I/O 포트에 지정된 번지는 중첩 가능
- 통신(read나 write) 할 때 I/O 전용 명령어
  (IN:/IORC 발생, OUT:/IOWC 발생)를 사용. (메모리:MOV)
- 예시 (장점: 주소 2배 할당 가능, 단점: 제어신호선증가, 회로 복잡)



## (가) 출력 포트

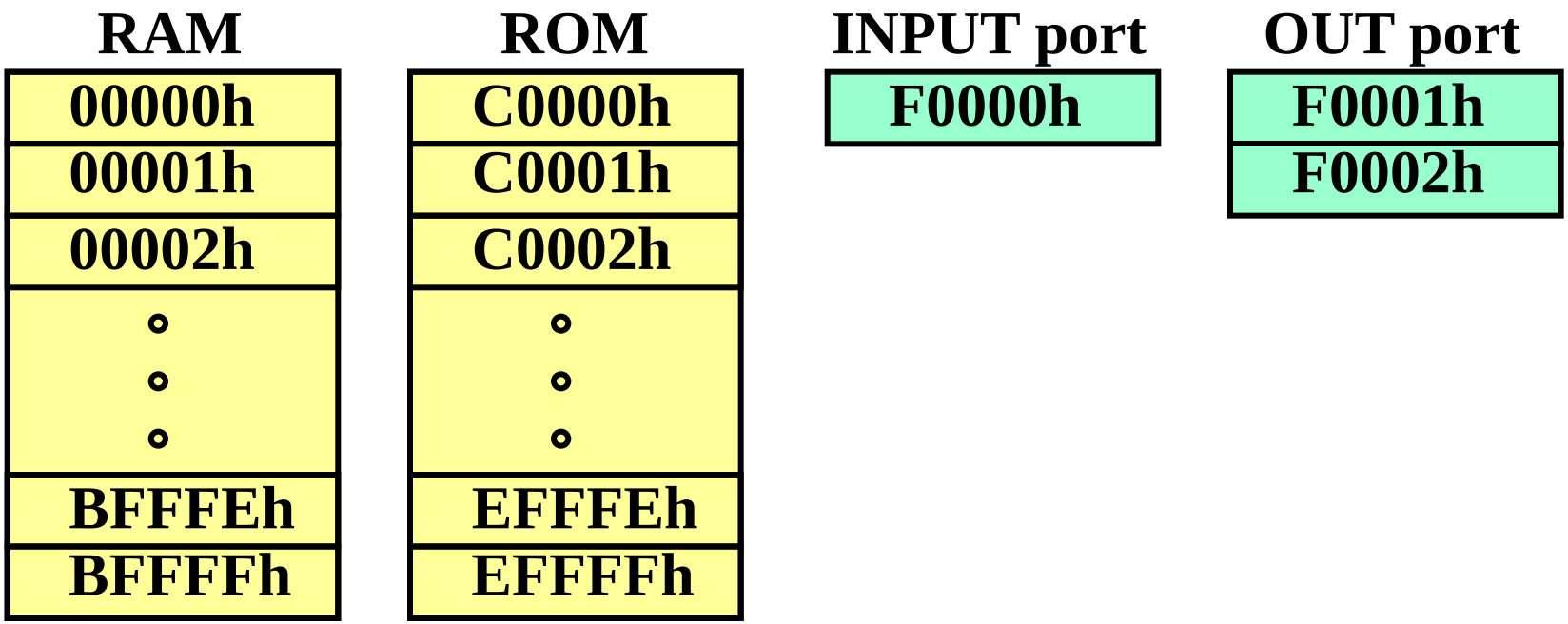

Address bus
Data bus
/IOWC
Address decoder
EN
Latches
Port Address와 write 명령( /IOWC)이 동시에 Active 될 때 발생
Port output bus

## (나) 입력 포트


Address bus
Data bus
/IORC
Address decoder
Latch EN
Tristate EN
Tristate latches
Port input bus

## (2) 메모리-맵(memory-map) I/O 포트 (예: ATMega128, 대부분 MCU)

- I/O 포트를 메모리와 같이 취급하여 메모리 할당.
- 메모리(RAM,ROM)와 주소 체계가 같음.
  즉, 메모리에 지정된 번지와 I/O 포트에 지정된 번지는 중첩 불가
- 통신(read나 write) 할 때 메모리와 같은 명령어 (예:MOV:/MRDC, /MWTC)를 사용.
- 예시 (장점: 제어신호감소, 회로 단순)



# 1.8 I/O 인터럽트

- 주변장치(I/O 장치)가 CPU에 작업을 요청할 때 사용하는 방법
  - **Polling**
  - **Interrupt**

- 마이크로컴퓨터에서 모든 작업은 CPU(master)가 주도하기 때문에
- 주변장치(slave)가 임의의 시간에 임의의 작업 을 하기 위해서는 위의 2가지 방식 중 한가지를 이용

## (1) Polled I/O

- **CPU**가 일정 시간 간격으로 각 주변장치의 상태를 순차적으로 확인하여 주변장치가 서비스를 필요로 하는지 또는 서비스 받을 준비가 되어 있는지를 확인하여, 서비스(해당 주변장치 관련 프로그램 수행)를 하는 방식
- 어떤 주변장치는 서비스 요구가 불규칙적이고, 예측 불가능한 특성이 있으므로 **CPU**는 가장 높은 속도로 주변장치를 자주 **polling**해야 함
- **Examples**

**1) if (key_input==1) // Any key input?**

**then key_no= KEY_BOARD;**

**2) while(!key_input); // wait until any key input**

**key_no= KEY_BOARD;**

- 단점

  - 시간 낭비: 주변장치가 요구하지 않더라도 일정 시간마다 수행중인 작업을 멈추고 **polling**

  - 우선순위 문제: 두 개 이상의 장치에서 동시에 서비스를 요청하게 되면 서비스의 우선순위나 긴급한 정도에 관계없이 가장 먼저 **polling**된 장치에 서비스

- **Polling** 방식은 정기적이고, 예측 가능한 시간 간격으로, 우선순위 고려가 필요하지 않은 서비스를 받는 장치에 적합

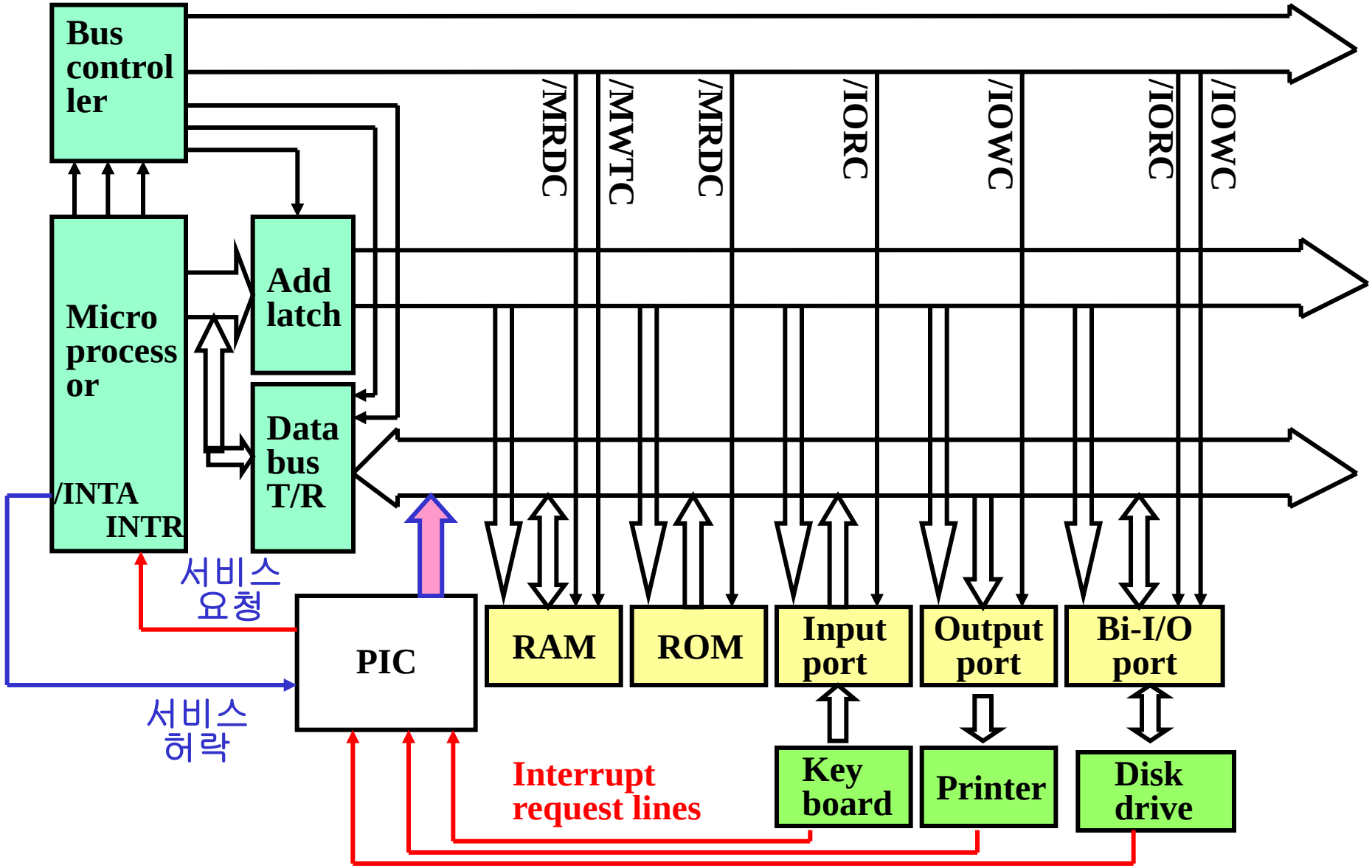
# (2) Interrupt-driven I/O

- 주변장치가 서비스를 요청할 때만 **CPU**가 서비스를 제공하고, 평상시는 처리하고 있던 프로그램에 집중하는 방식
- 인터럽트**:** 주변장치로부터 서비스를 요구 받는 것

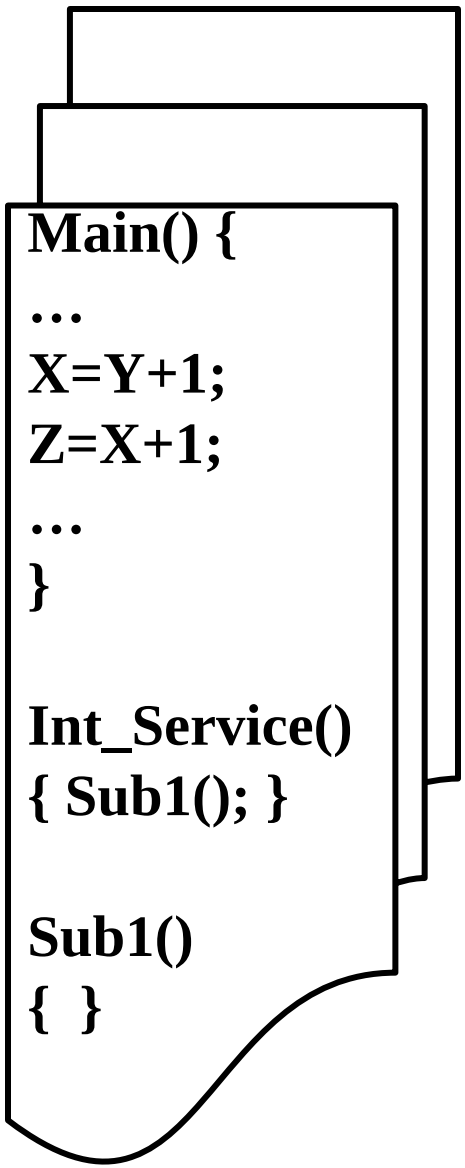
- **CPU**가 **I/O** 인터럽트를 받으면 현재 진행중인 프로그램을 일시적으로 멈추고 인터럽트를 요청한 장치에 해당하는 프로그램**(service routine)**을 메모리에서 가져와 수행
- 인터럽트 루틴이 끝나면 다시 원래의 프로그램으로 복귀하여 중단되었던 지점부터 다시 실행

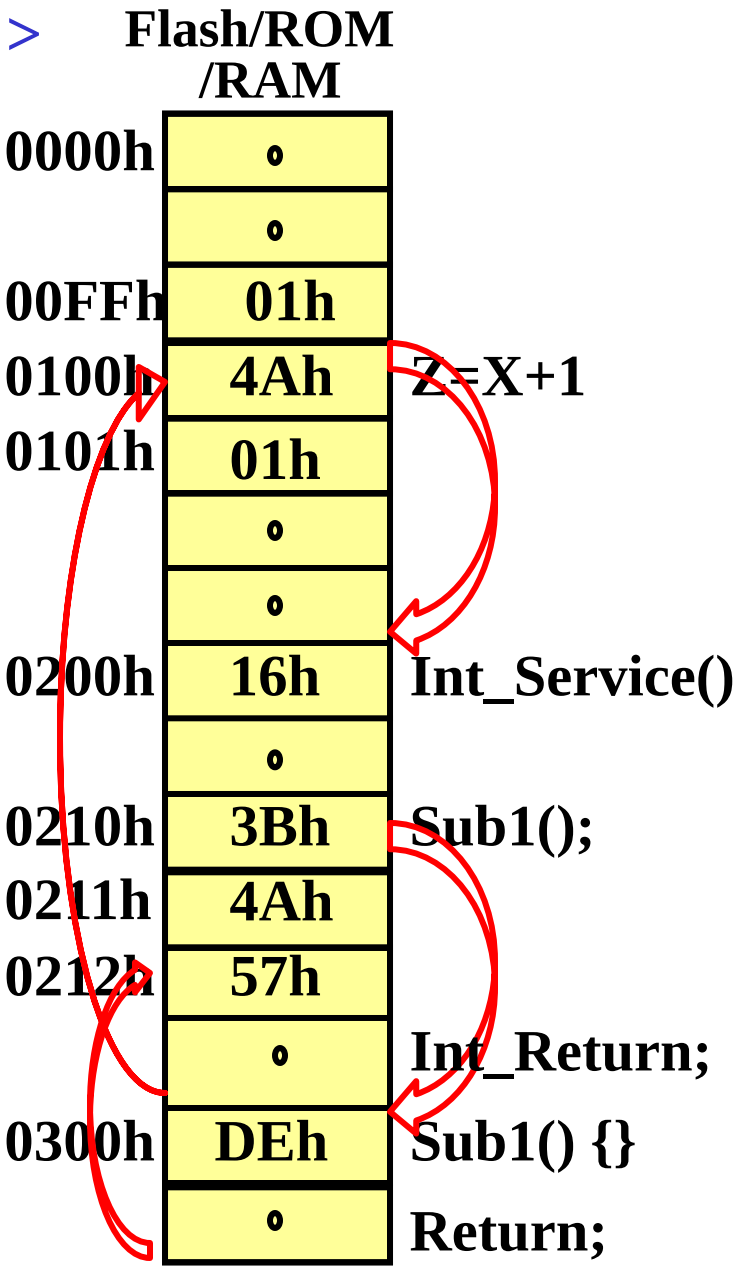
- 프로그램 가능 인터럽트 제어기**(Programmable interrupt controller: PIC) :** 우선순위에 입각하여 인터럽트를 처리

# < Interrupt-driven I/O 방식을 사용한 시스템 예 >



# < 인터럽트를 사용한 프로그램 예 >



- 마이크로컴퓨터 개발(H/W 설계 / 프로그래밍) 과정