메모리

1. 기본 반도체 메모리
2. READ 전용 메모리(ROM)
3. 랜덤 엑세스 메모리(RAM)
4. 플래쉬(Flash) 메모리
5. 특수 형태의 메모리

1. 기본 반도체 메모리

(1) 메모리: 오랜 기간 혹은 짧은 기간 동안 2진 데이터를 저장하는 장치

*** 1 cell에 1 bit 저장 (cell 은 메모리 종류에 따라 다름)**

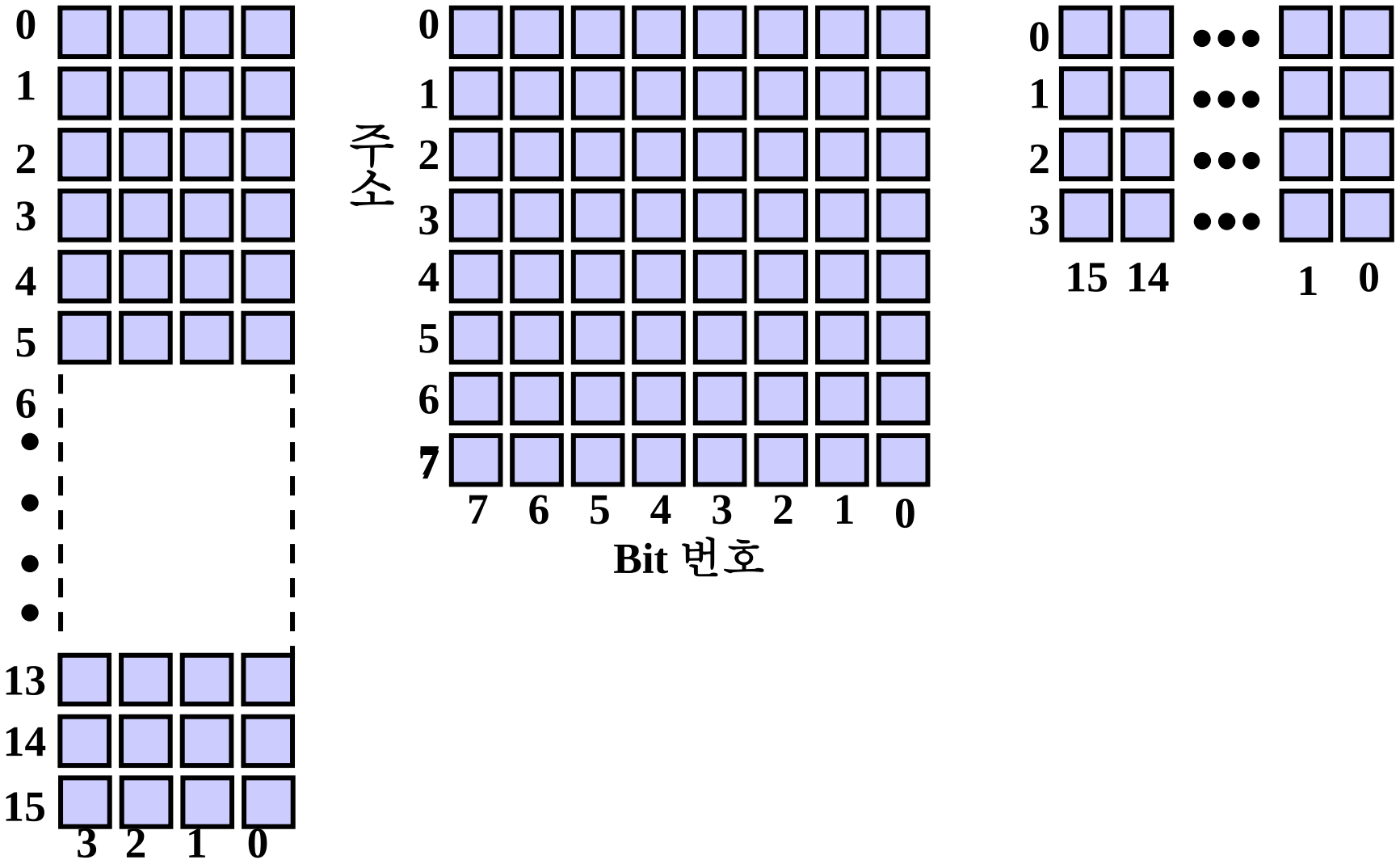
- **분류: 반도체 메모리, 자기 및 광학 메모리**
- **반도체 메모리 분류: 래치, 전하-저장 소자 등등**
- **반도체 메모리 역할: 프로그램(기계어문장들) 저장, 데이터 저장**

(2) 2진 데이터 저장 단위

- **보통 1byte(8 bit) 단위로 저장**
- **정보의 저장 단위: word**

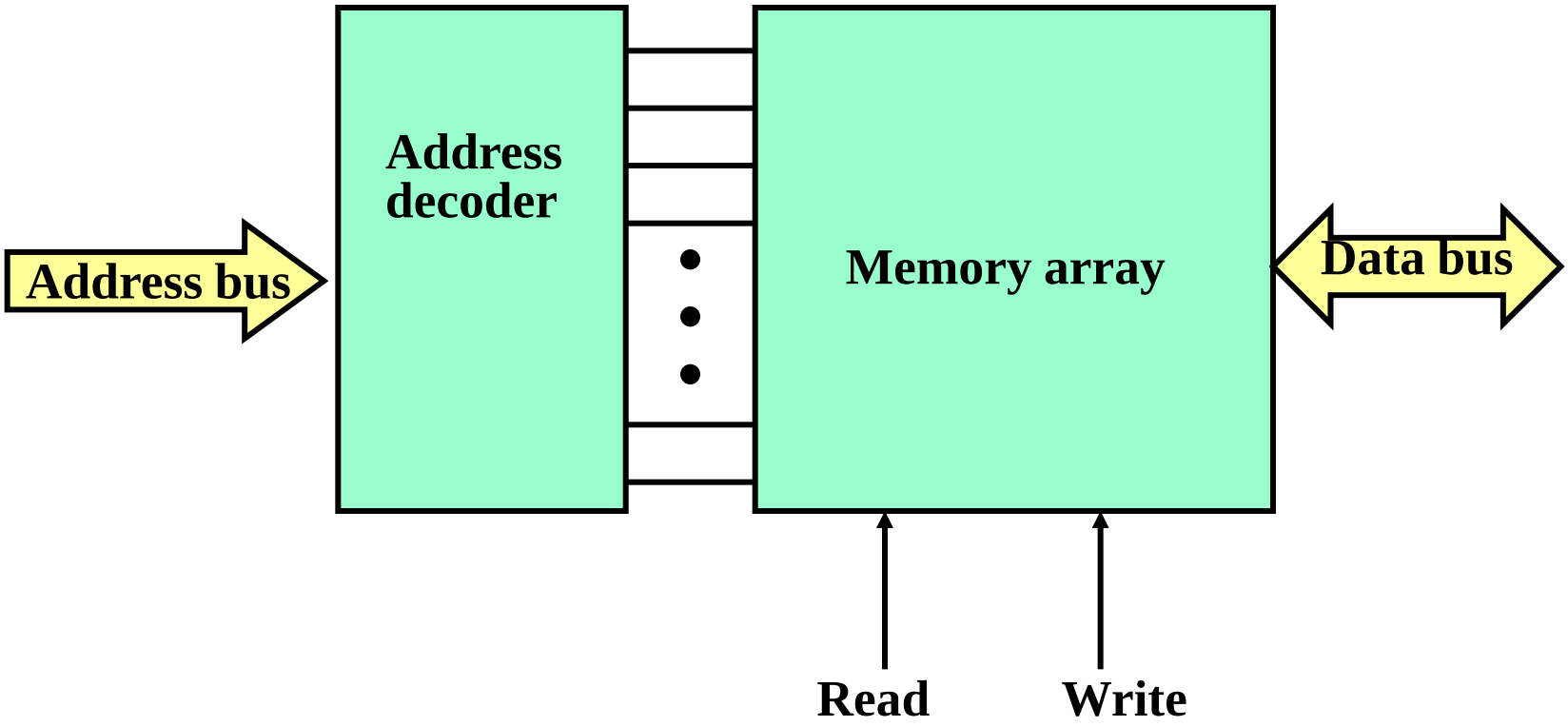
  1 word: 1bit, 1byte, 2byte, 4 byte 등 (보통은 2 byte)

(3) 기본 반도체 메모리 배열 (예: 64 cell의 여러가지 배치)

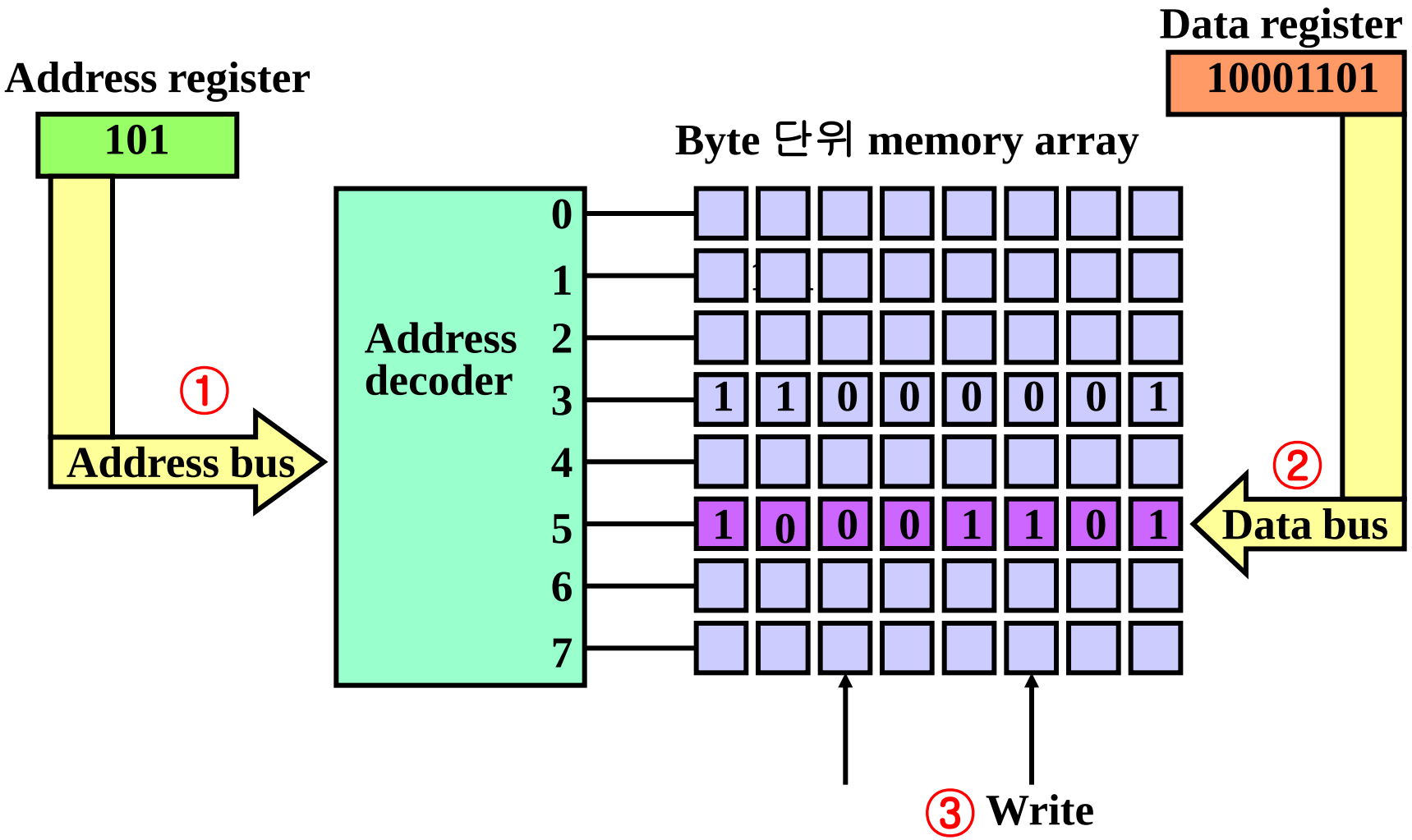


(a)16 x 4 array (b)8 x 8 array (c)4x 16 array

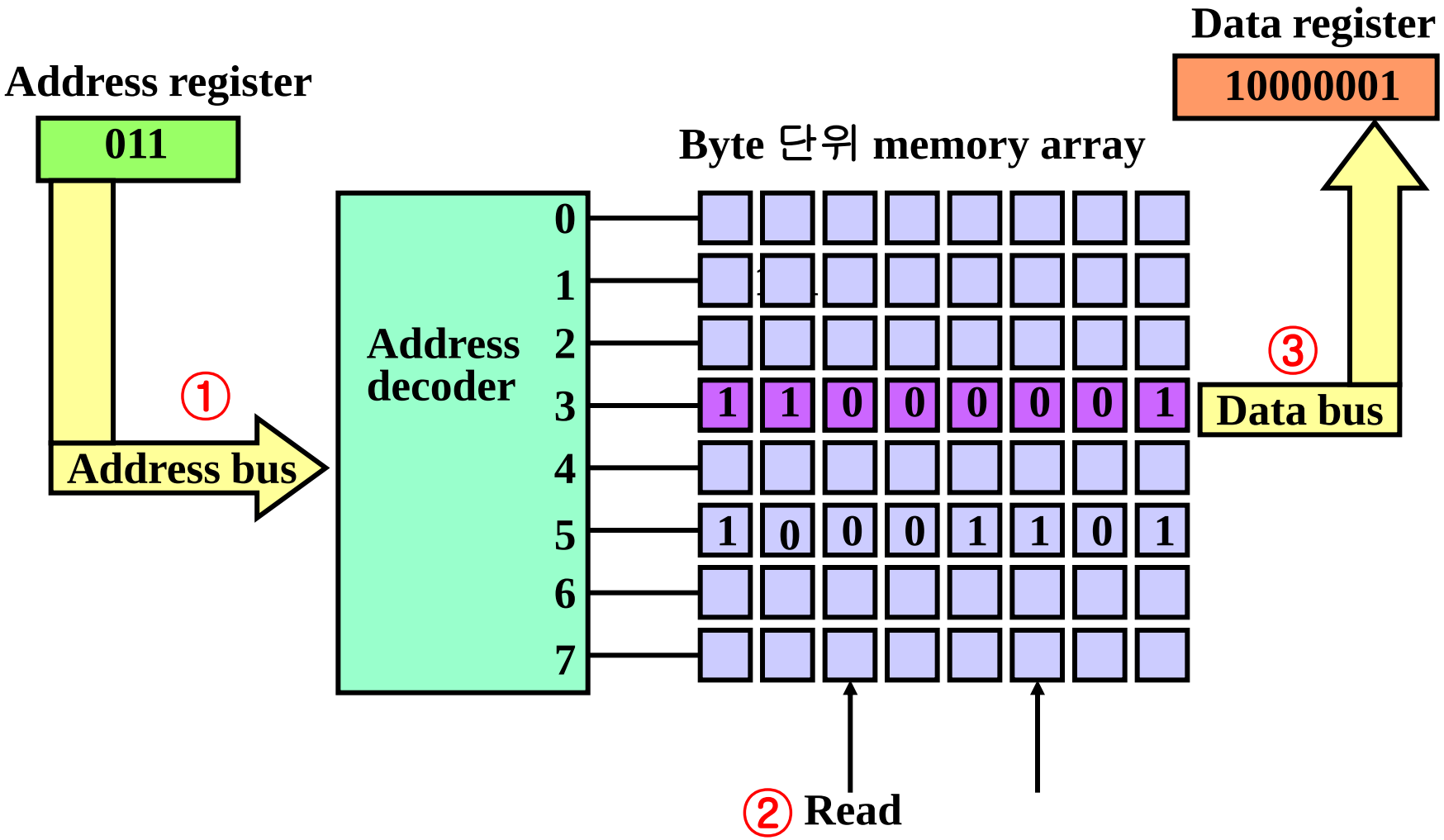
(4) 기본 메모리 동작


Address bus
Address decoder
Memory array
Data bus
Read
Write

(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

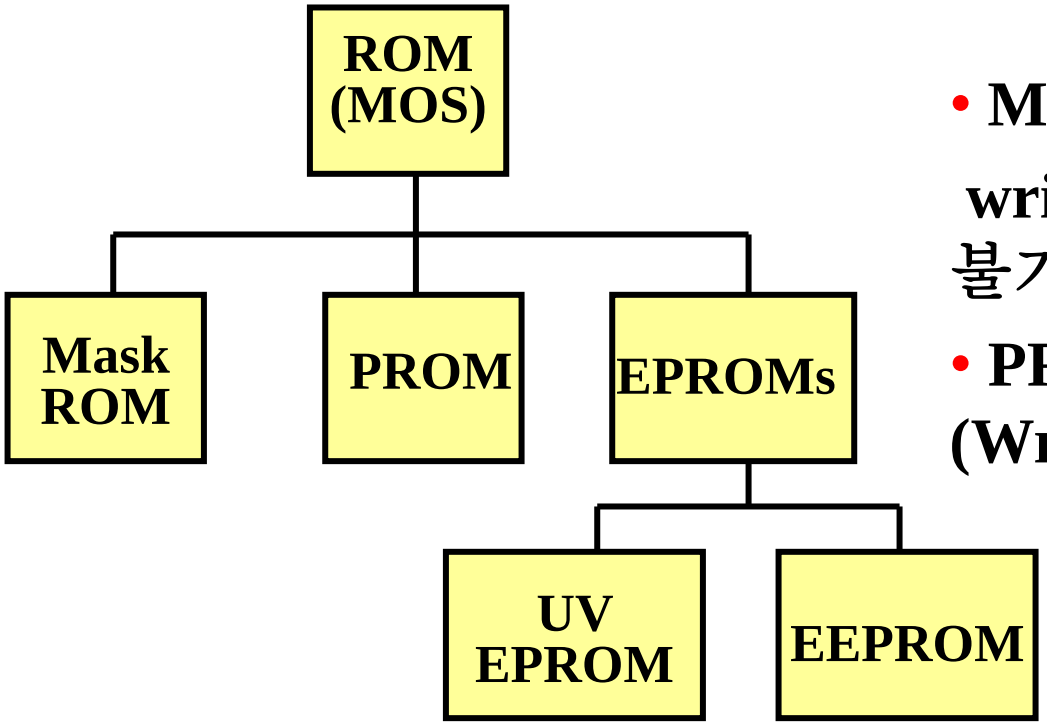
2. Read Only Memory(ROM)

- On-system(board)에서 Read 만 가능한 Memory

  * Writing은 Off –system에서만 가능(* 예외: EEPROM)

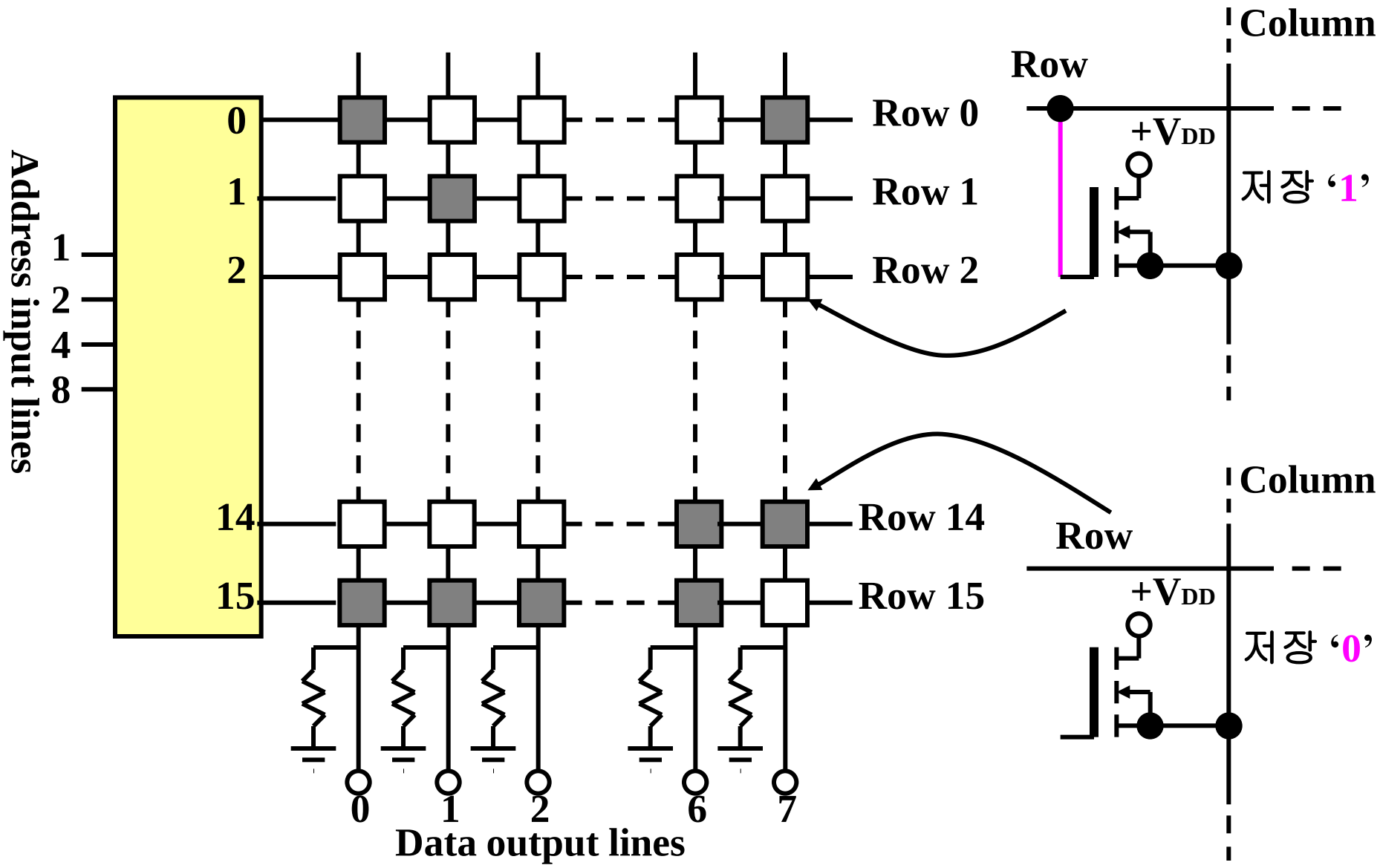
- 불휘발성(nonvolatile) memory: Power off시에도 저장된 데이터는 상태 유지

(1) ROM의 종류

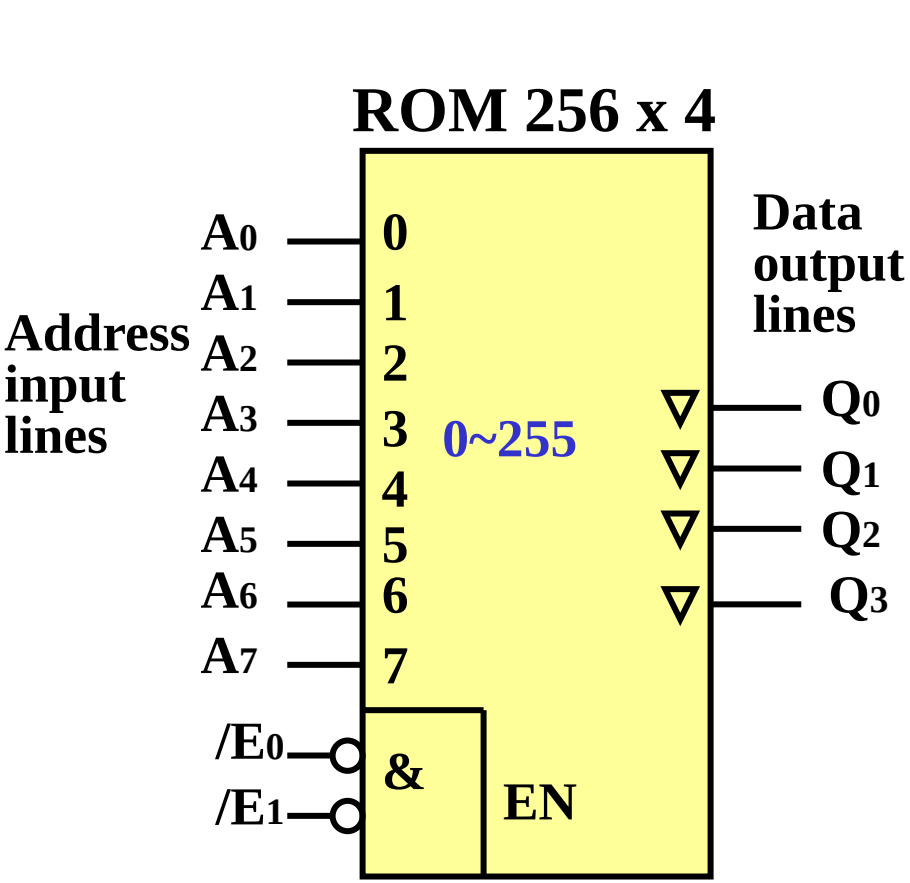


- Mask ROM: 제조시 데이터를 writing. ROM 구매자는 writing 불가.
- PROM: One time Progammable (Writable) ROM

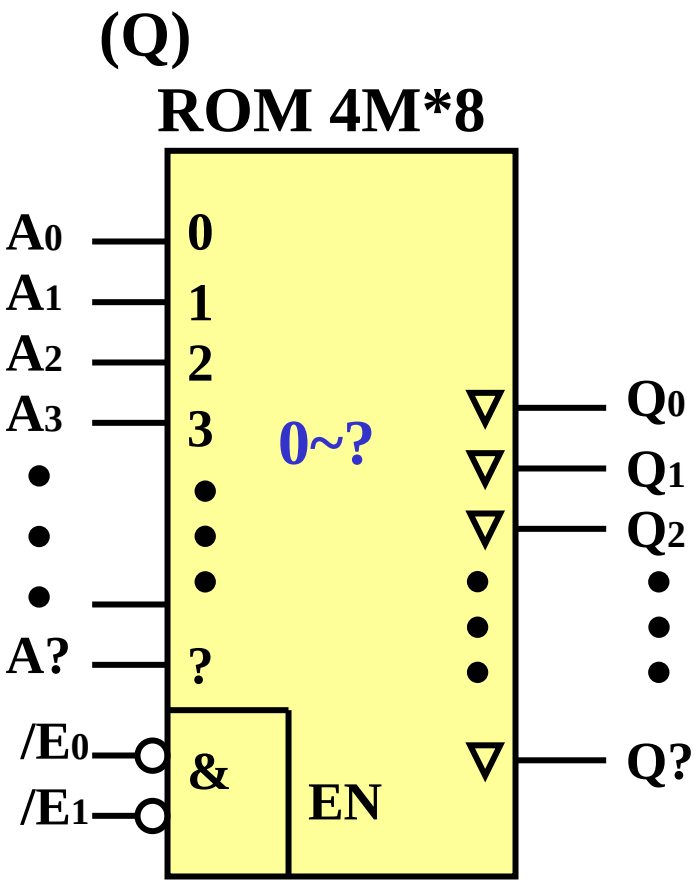
(2) 간단한 ROM IC 구조 (16 X 8 ROM)



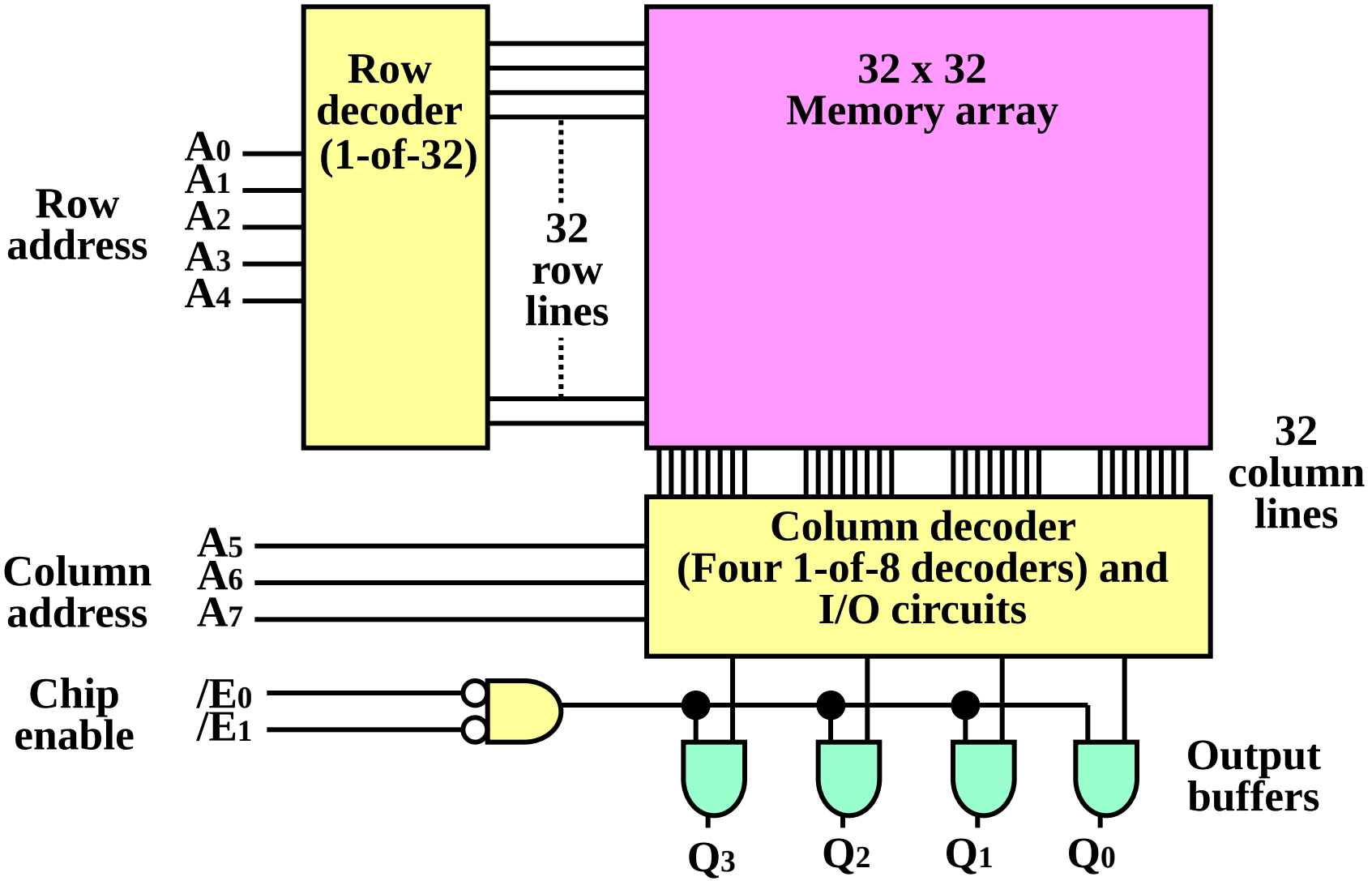
(3) 내부 ROM 구조


ROM 256 x 4
Address input lines
A0
A1
A2
A3
A4
A5
A6
A7
0
1
2
3
4
5
6
7
0~255
/E0
/E1
&
EN
Data output lines
Q0
Q1
Q2
Q3

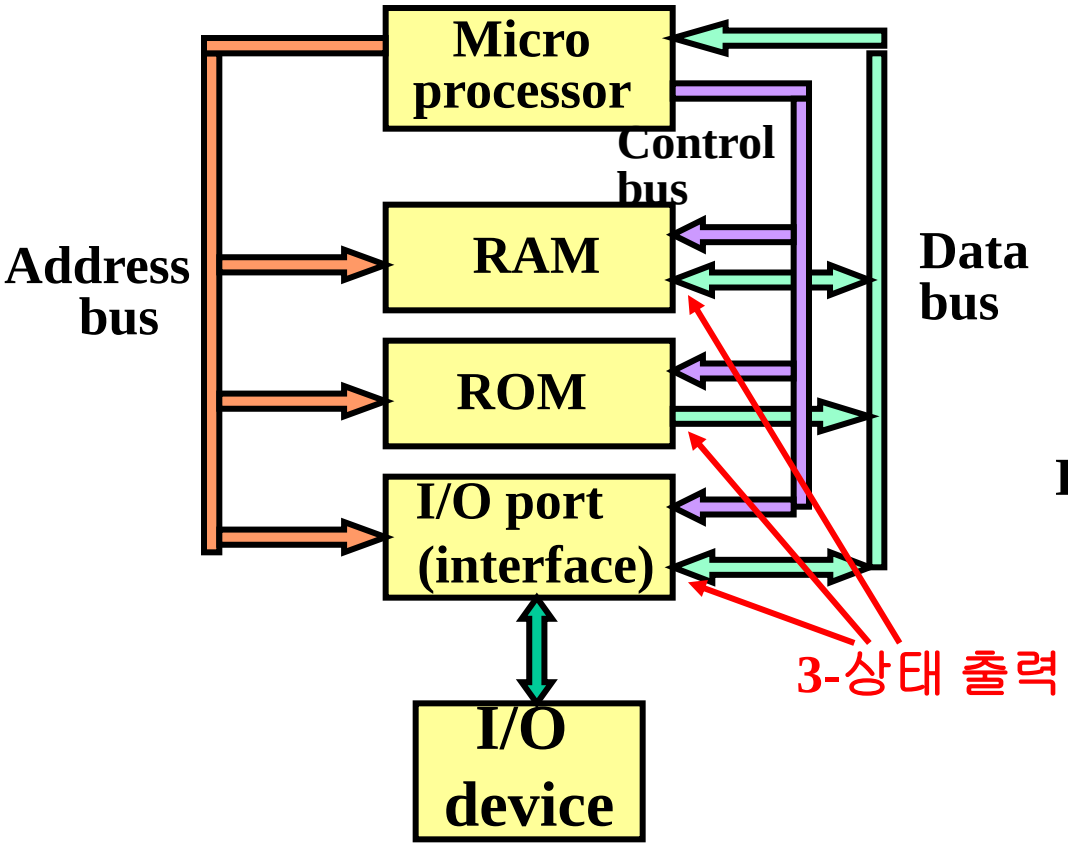
(Q)

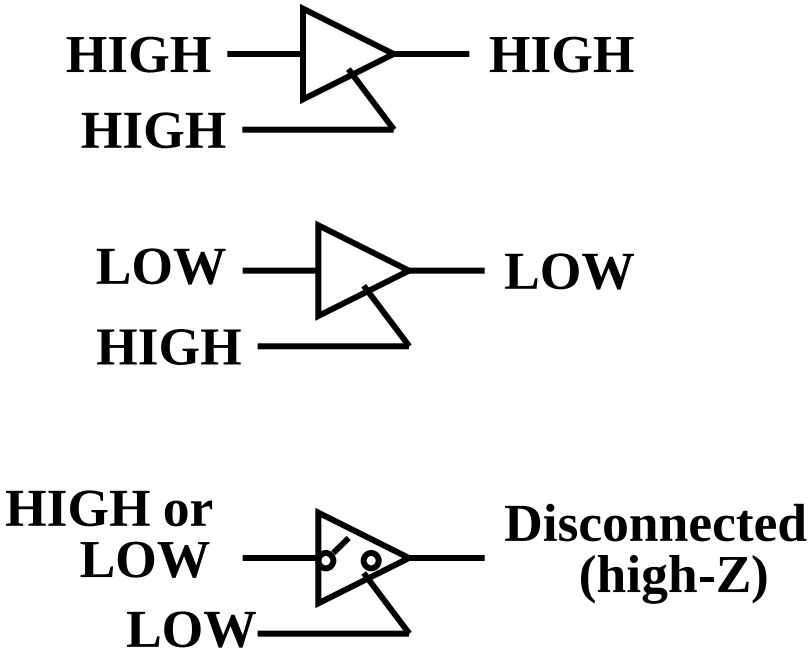

ROM 4M*8
A0
A1
A2
A3
A?
0
1
2
3
?
0~?
/E0
/E1
&
EN
Q0
Q1
Q2
Q?

ROM 256 x 4의 내부 구조



(4) 3-상태(Tri-state) 출력과 버스

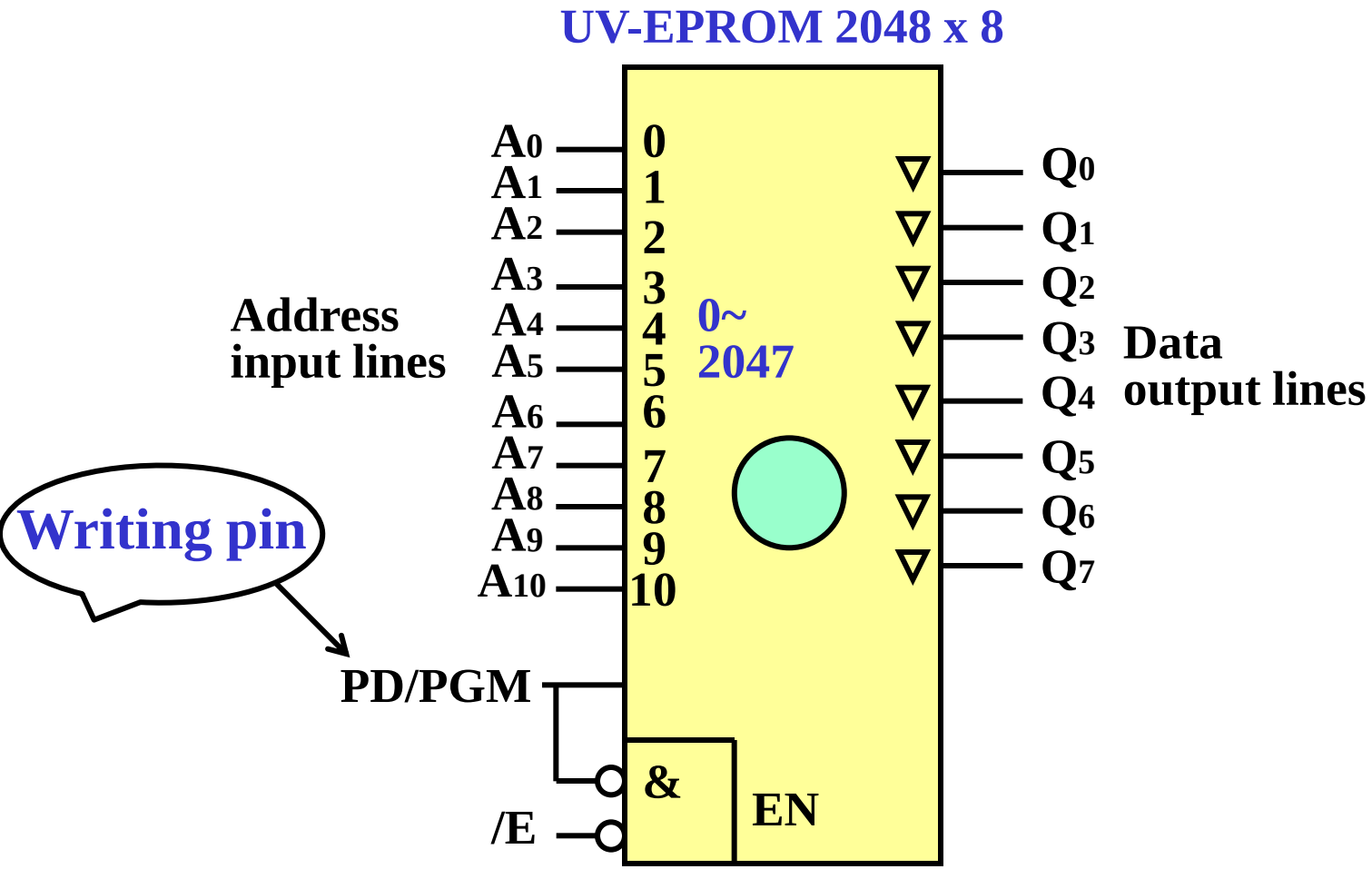




<Buffer의 3-상태>

(5) EPROM

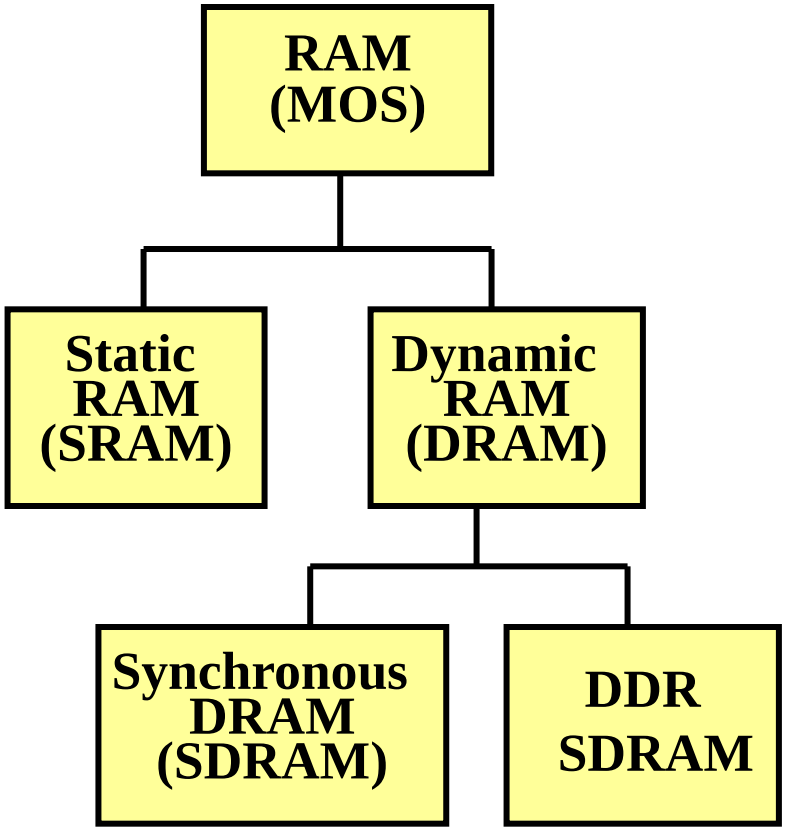
- **Multi-Times Programmable ROM**
- **UV EPROM: 전기적 write, 광학적 erase (Off-system Writing)**
- **EEPROM: 전기적인 펄스로 write, erase (On-system Writing)**



3. Random Access Memory (RAM)

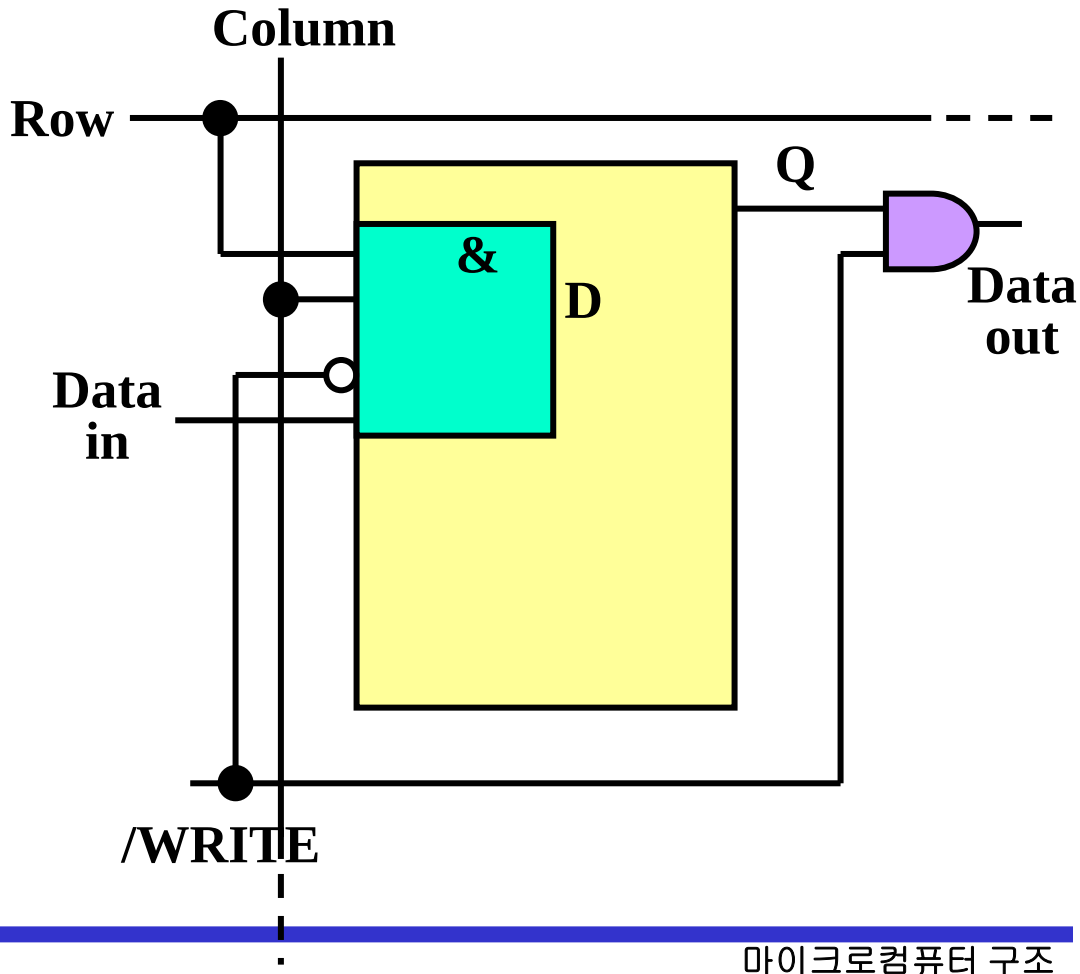
- On-system(board)에서 Read 및 Write가 가능한 Memory
- 휘발성(volatile) memory: Power off시에 저장된 데이터 사라짐

(1) RAM의 종류

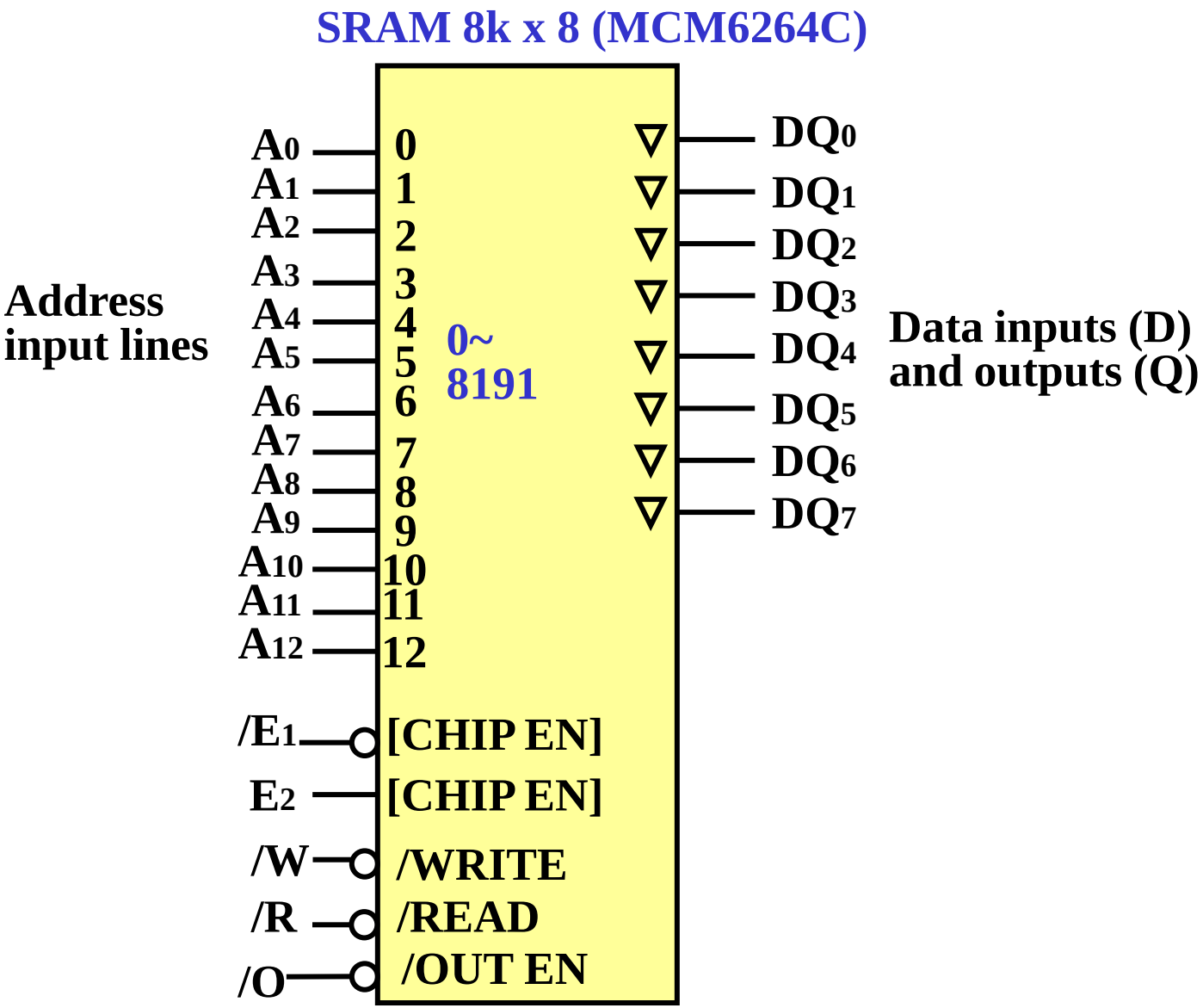


* DDR: Double Data Rate

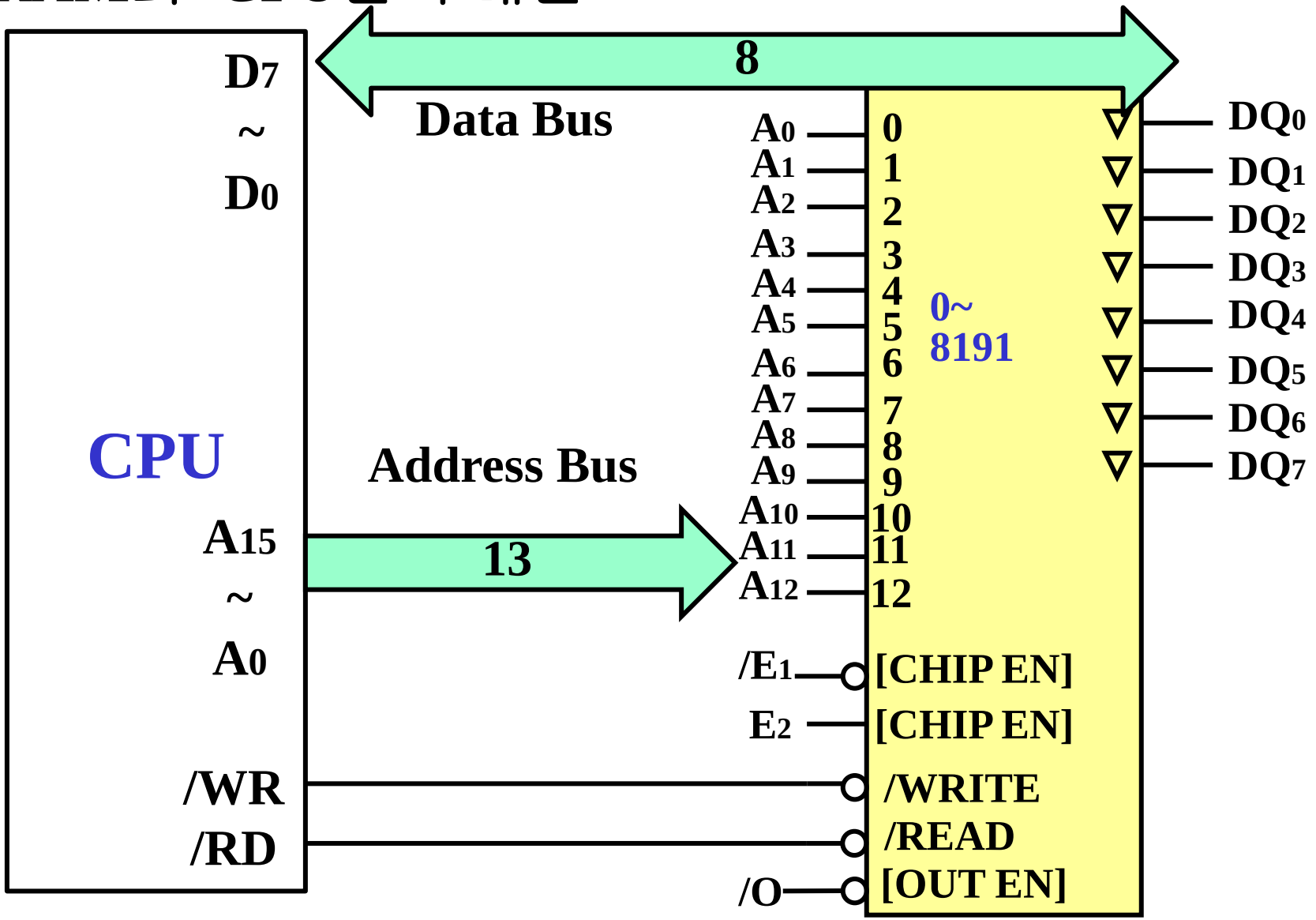
(2) SRAM cell : F/F



(3) SRAM의 구조

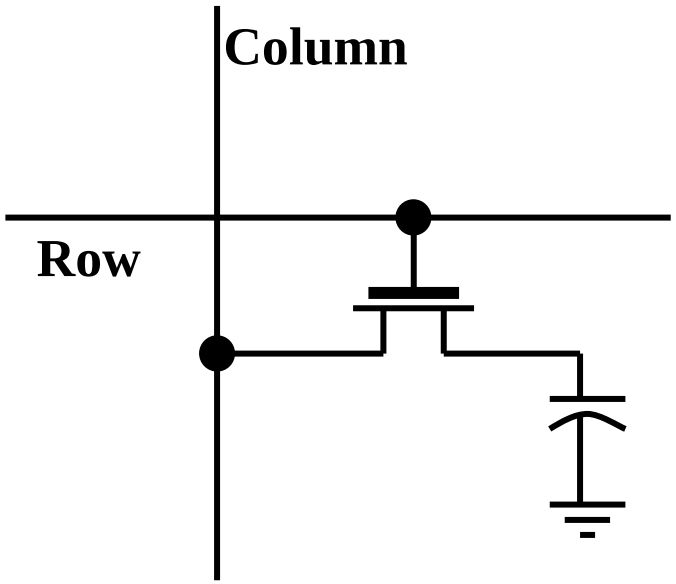

SRAM 8k x 8 (MCM6264C)
Address input lines
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
0
1
2
3
4
5
6
7
8
9
10
11
12
0~ 8191
/E1
[CHIP EN]
E2
[CHIP EN]
/W
/WRITE
/R
/READ
/O
/OUT EN
DQ0
DQ1
DQ2
DQ3
DQ4
DQ5
DQ6
DQ7
Data inputs (D) and outputs (Q)

(4) SRAM과 CPU간의 배선



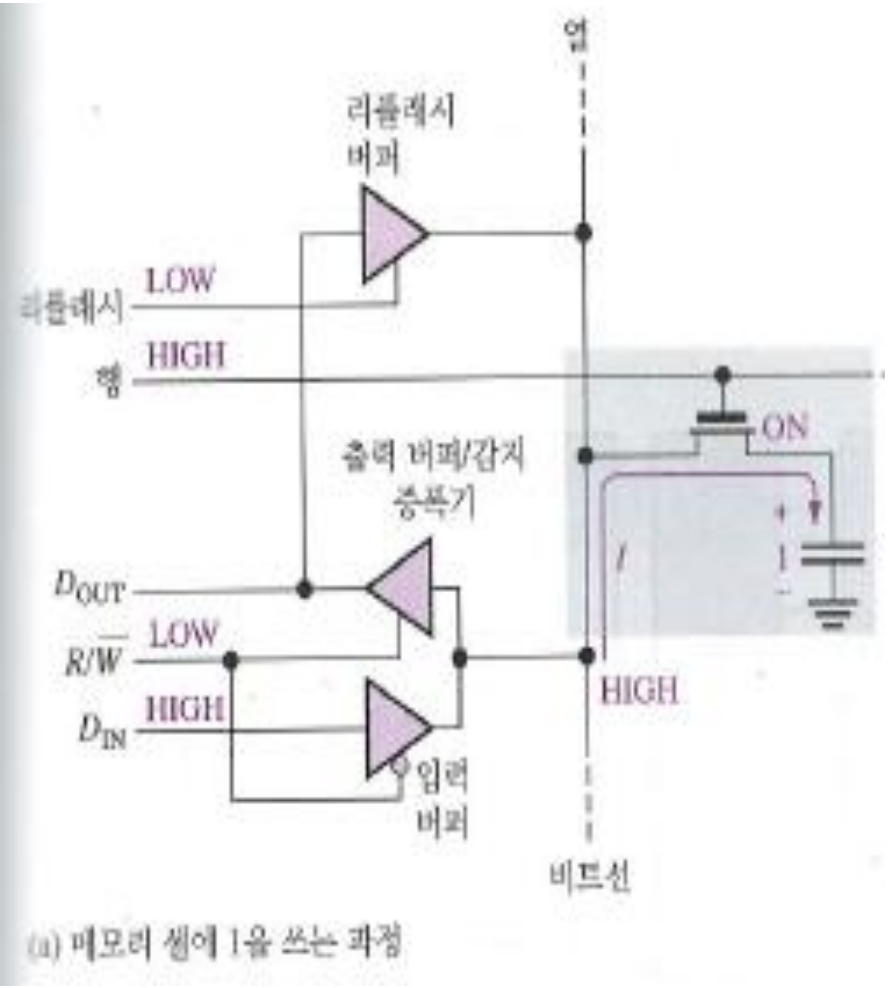
(4) DRAM의 구조

- **Cell의 구성: 1-TR, 1-CAP (**집적도 매우 높음**)**
- **CAP**의 충방전으로 **1-bit** 저장
- 오래 두면 방전하므로 재충전 필요 **(refresh** 회로**)**

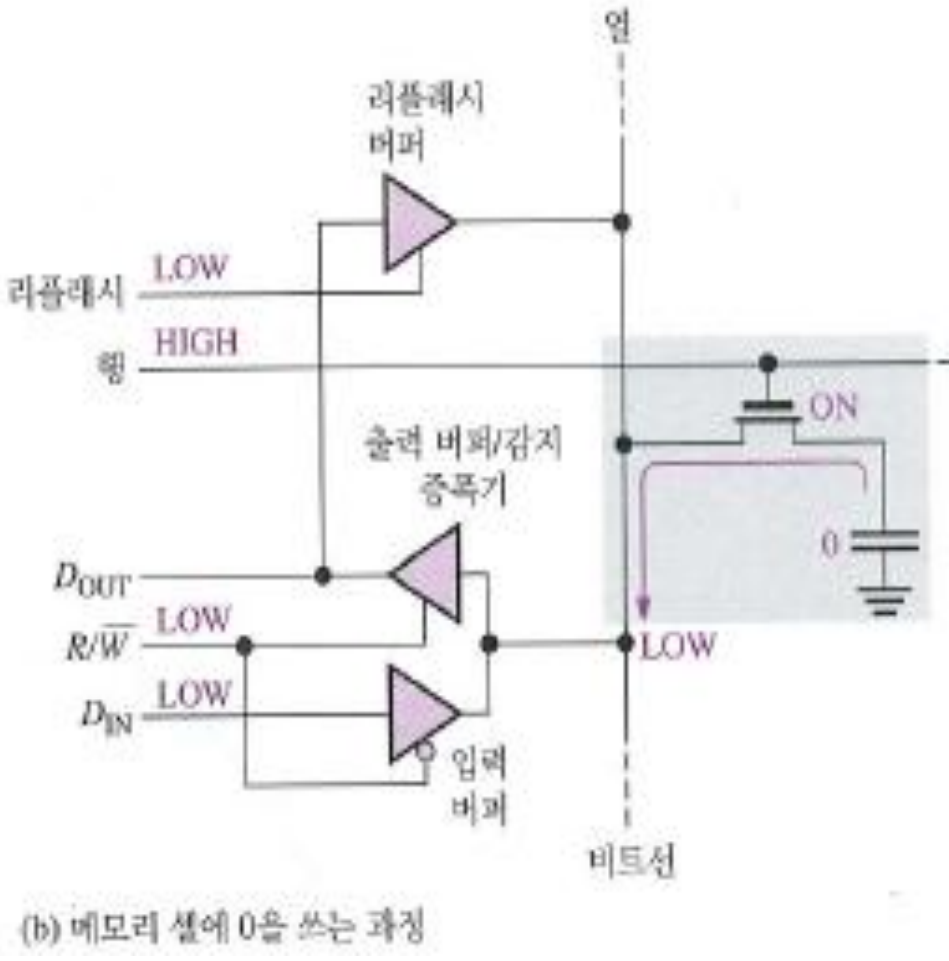


< DRAM cell >

DRAM cell의 기본동작(Write)

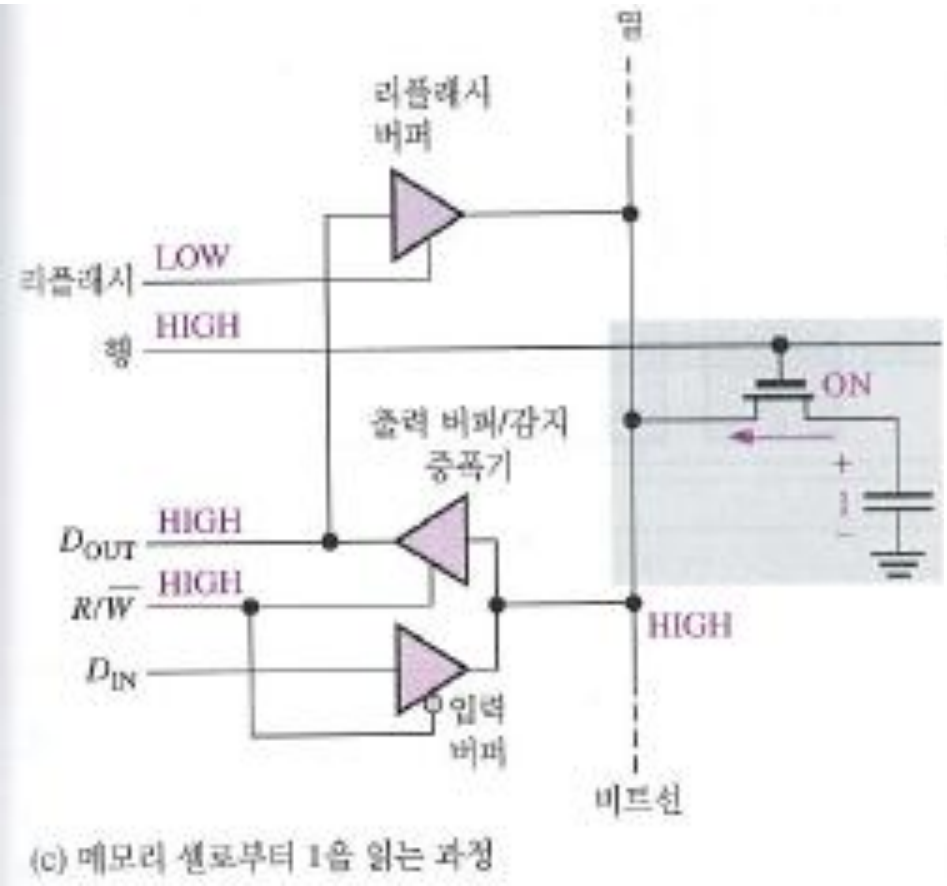


(a) 메모리 셀에 1을 쓰는 과정

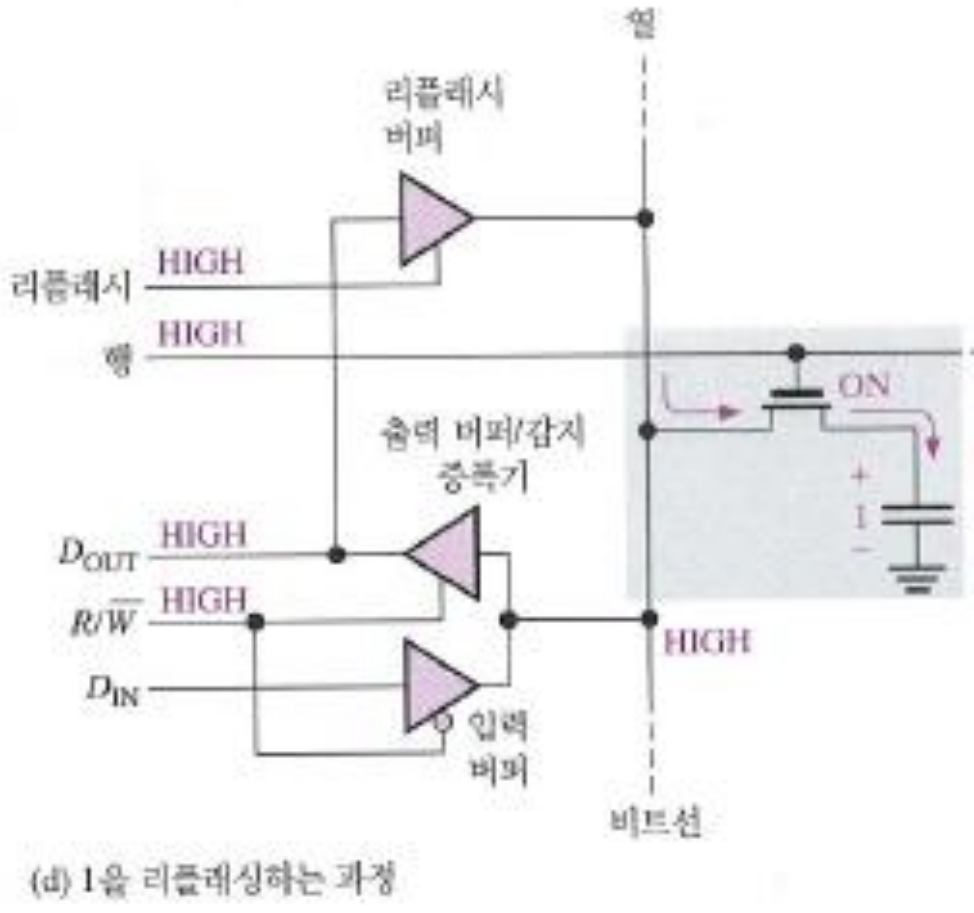


(b) 메모리 셀에 0을 쓰는 과정

DRAM cell의 기본동작(Read)



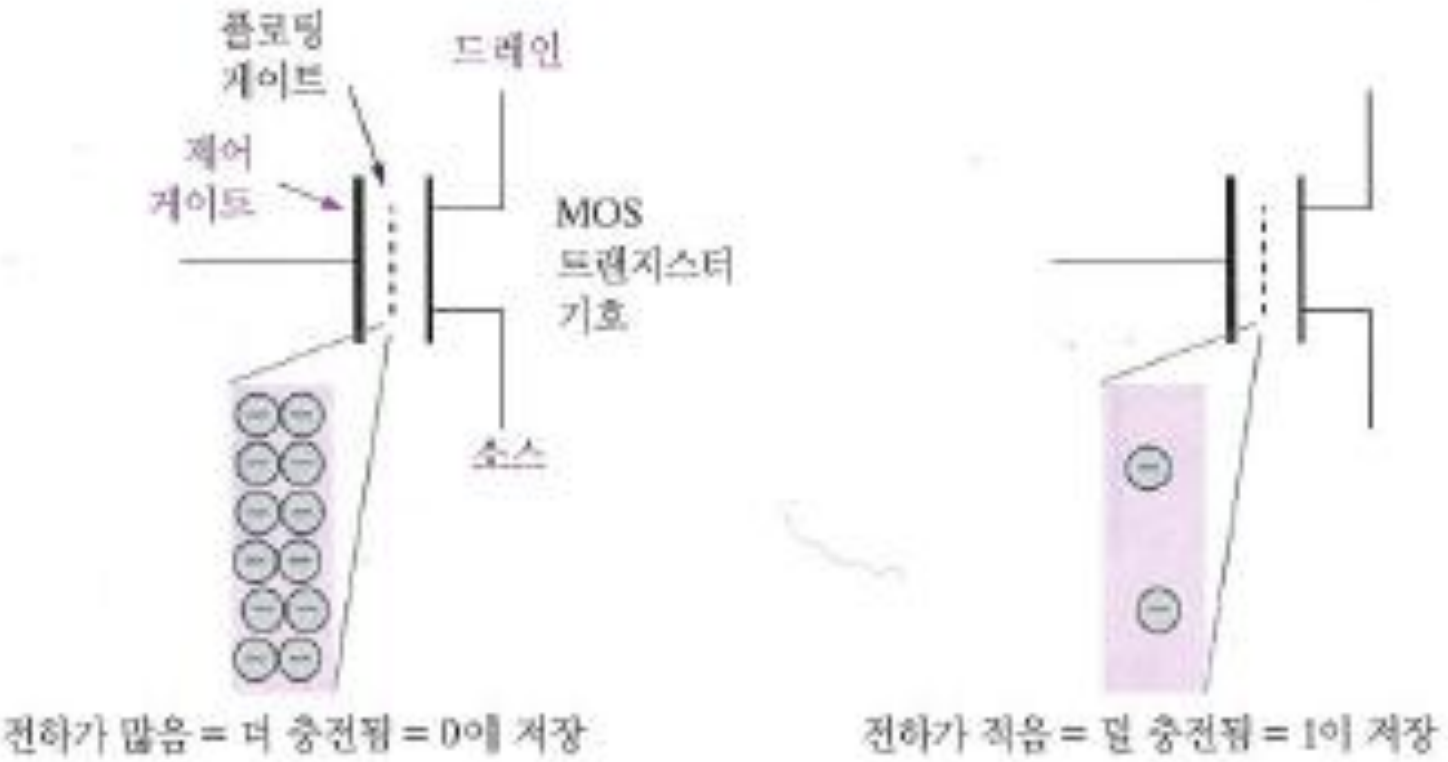
(c) 메모리 셀로부터 1을 읽는 과정



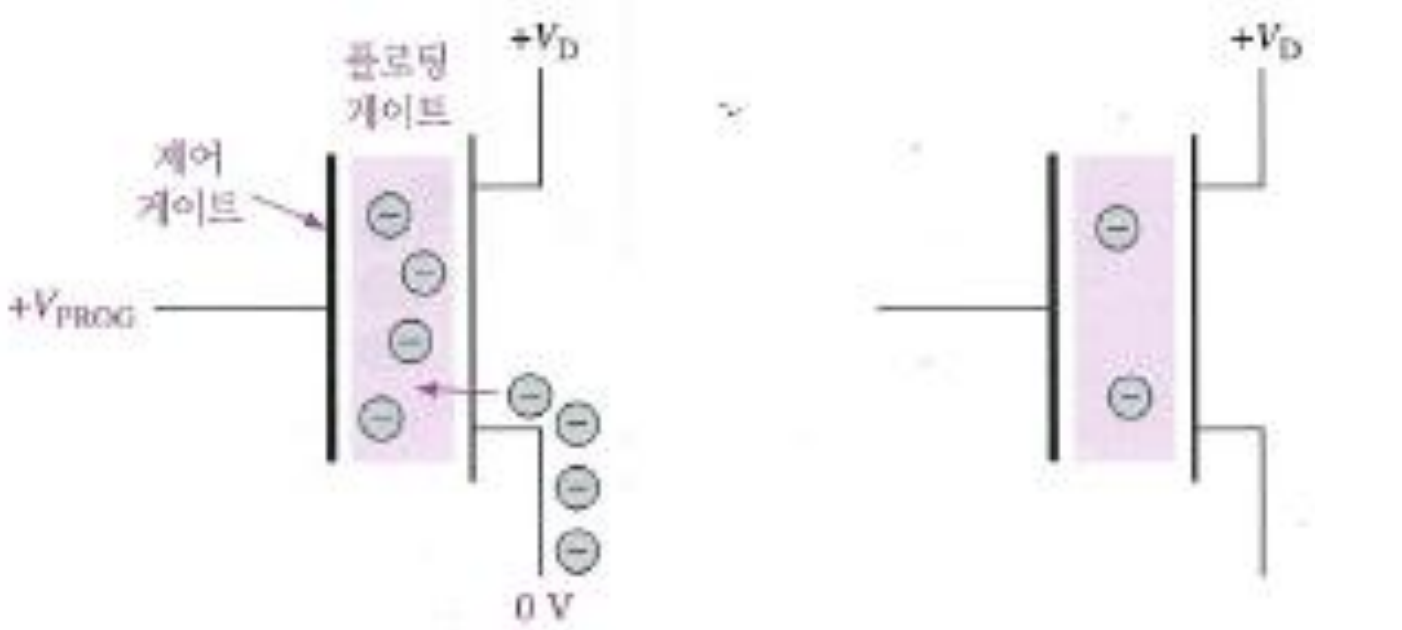
(d) 1을 리플레싱하는 과정

4. 플래쉬(Flash) 메모리

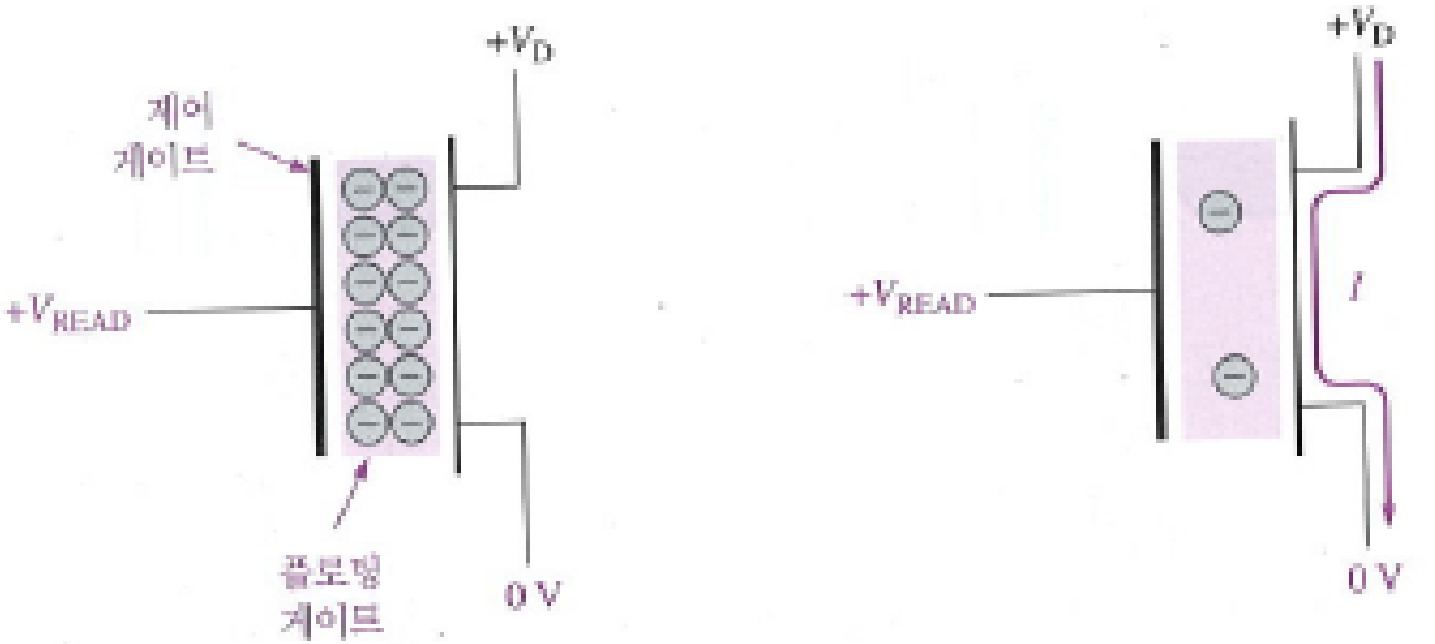
- 비휘발성인 고밀도의 **READ/WRITE** 메모리
- **floating-gate MOS TR**을 이용하여 충전 및 방전
- 재충전 필요 없음
- **NOR type :** 프로그램 저장용
  NAND type : 데이터 저장용



- **Write 동작**



- **Read 동작**



메모리 비교

기억형태	비휘발성	고집적도	하나의 TR cell	시스템내에서 쓰기 가능 여부
Flash	**YES**	**YES**	**YES**	**YES**
SRAM	**NO**	**NO**	**NO**	**YES**
DRAM	**NO**	**YES**	**YES**	**YES**
ROM	**YES**	**YES**	**YES**	**NO**
EPROM	**YES**	**YES**	**YES**	**NO**
EEPROM	**YES**	**NO**	**NO**	**YES**

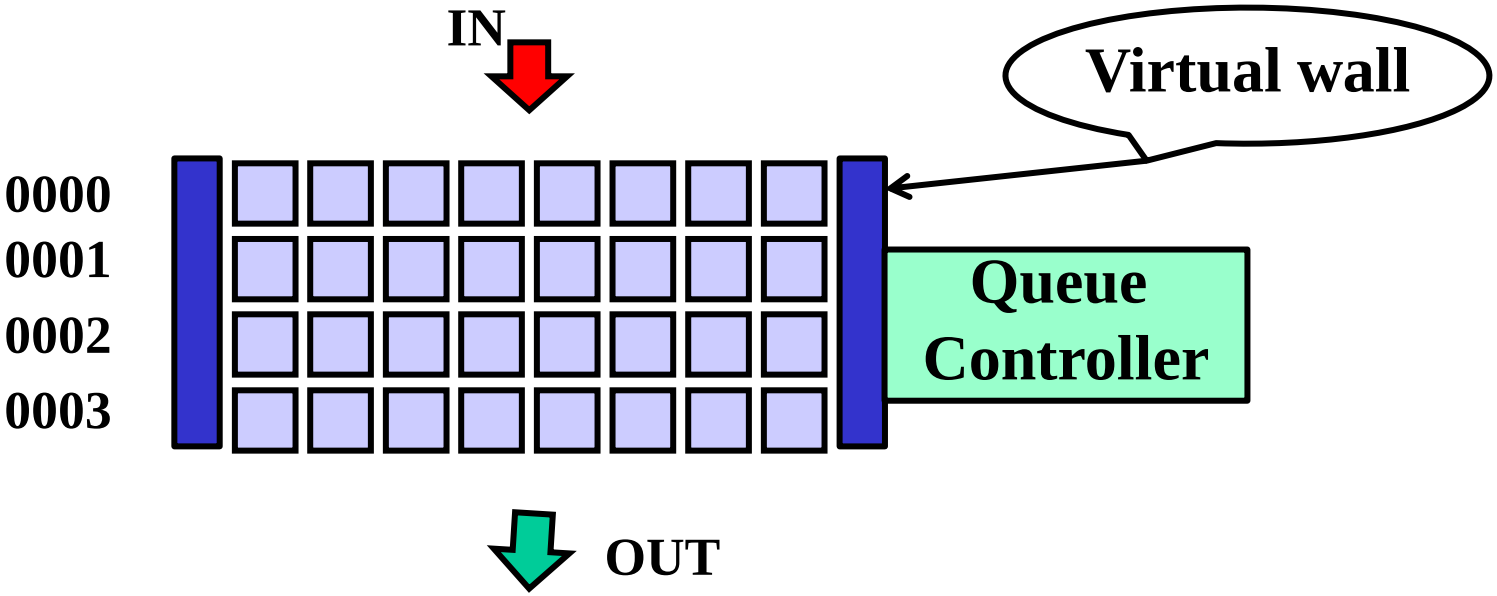
메모리 주요 용도

메모리종류	주요 용도
Flash(NOR)	프로그램 저장용
Flash(NAND)	데이터 저장용
DRAM	데이터/변수 저장용(주로 범용 컴퓨터)
SRAM	데이터/변수 저장용(주로 임베디드컴퓨터)
EPROM	프로그램 저장용
EEPROM	(작은)데이터 저장용

5. 특수 형태의 메모리

(1) FIFO (First In – First Out) 메모리 : Queue

- 구조: Standalone- type Queue (예:Queue size: 4, data size: 8bit)



- Queue 는 보통 RAM에 설정하지 않고 독립적인 모듈로 제작
- 용도: Instruction 처리, 컴퓨터내의 Job(Process)의 순서관리

- **Queue의 동작원리 (1 bit 데이터 예를 들어)**
  : Shift register 동작과 비교

Shift register

Input	X	X	X	X	Output
0	0	X	X	X	⟶
1	1	0	X	X	⟶
1	1	1	0	X	⟶
0	0	1	1	0	⟶

X: unknown data bits

Forced shift

FIFO (Queue)

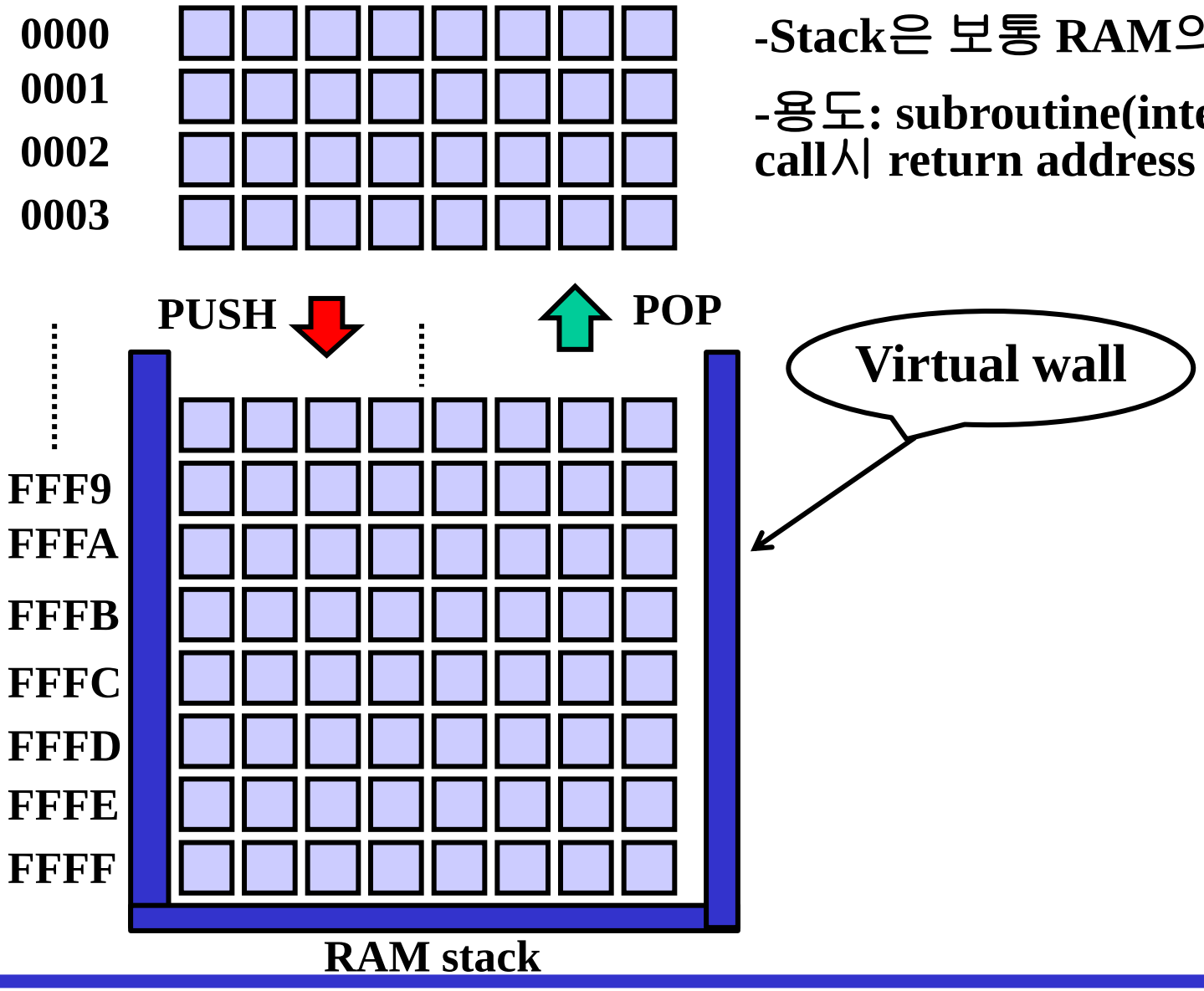
Input	-	-	-	-	Output
0	-	-	-	0	⟶
1	-	-	1	0	⟶
1	-	1	1	0	⟶
0	0	1	1	0	⟶

- : empty positions

Fall through

(2) LIFO (Last In – First Out) 메모리 : Stack

<구조>

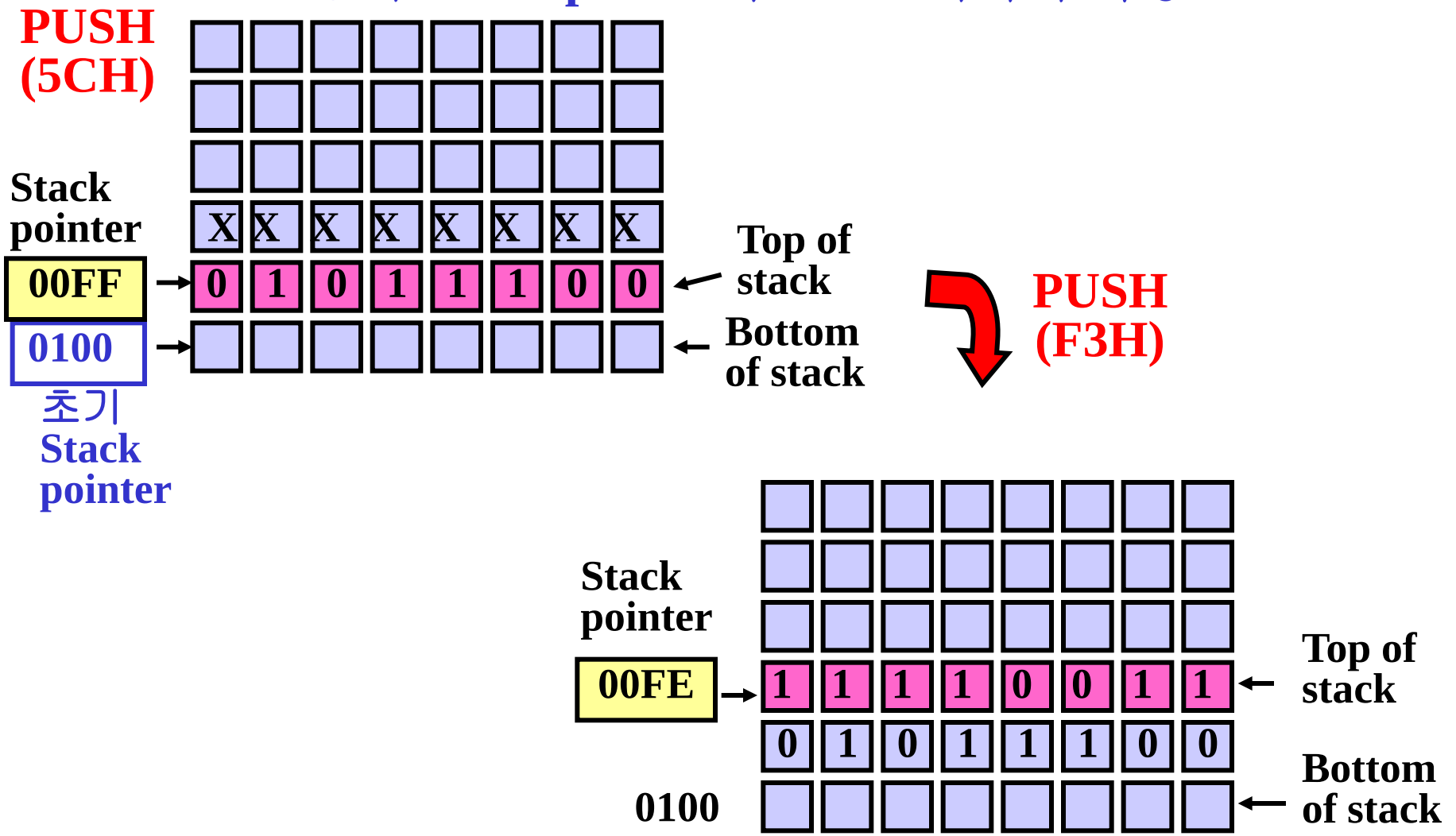


-Stack은 보통 RAM의 끝부분에 설정

-용도: subroutine(interrupt, general) call시 return address 저장

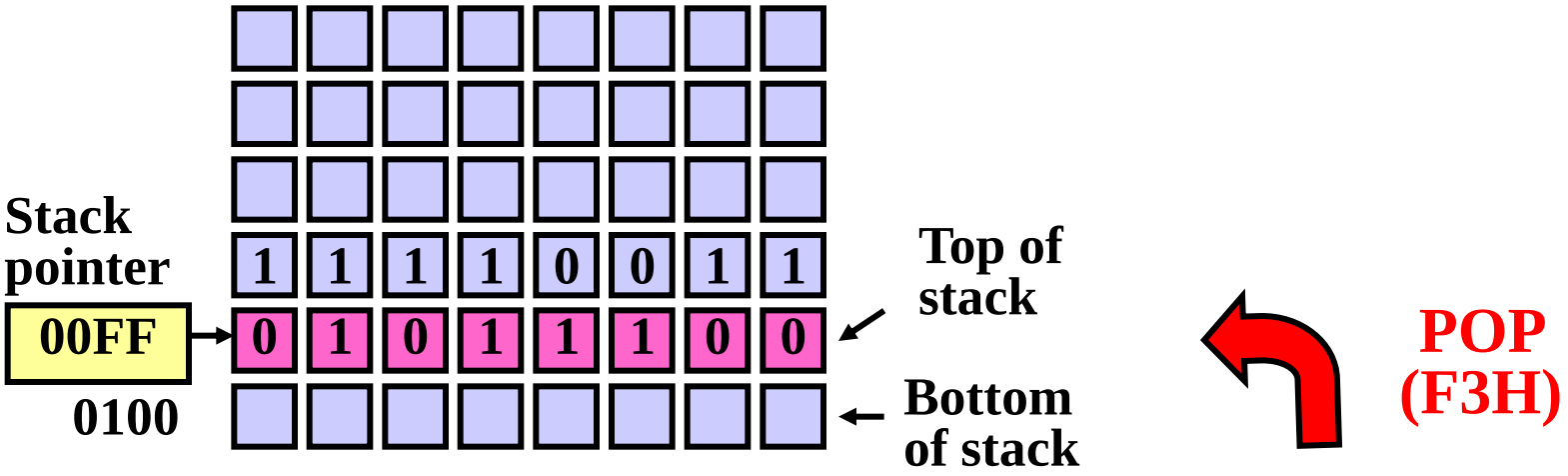
(2) LIFO (Last In – First Out) 메모리 동작원리

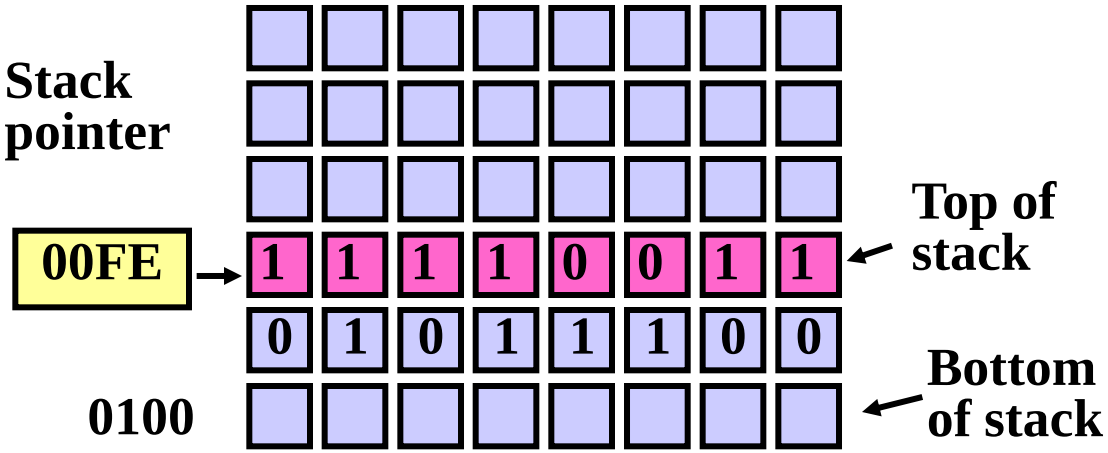
<PUSH 동작: stack pointer 감소 => 데이터 저장>



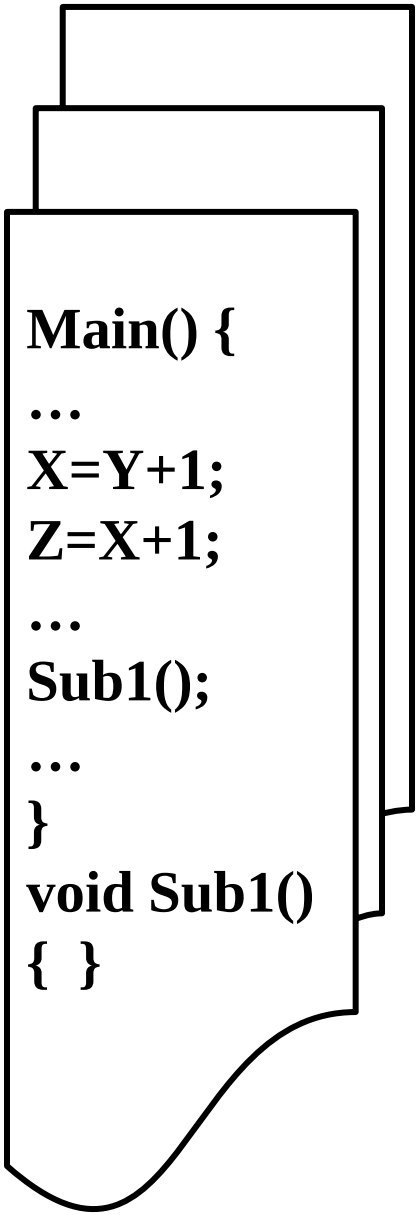
(2) LIFO (Last In – First Out) 메모리 동작원리

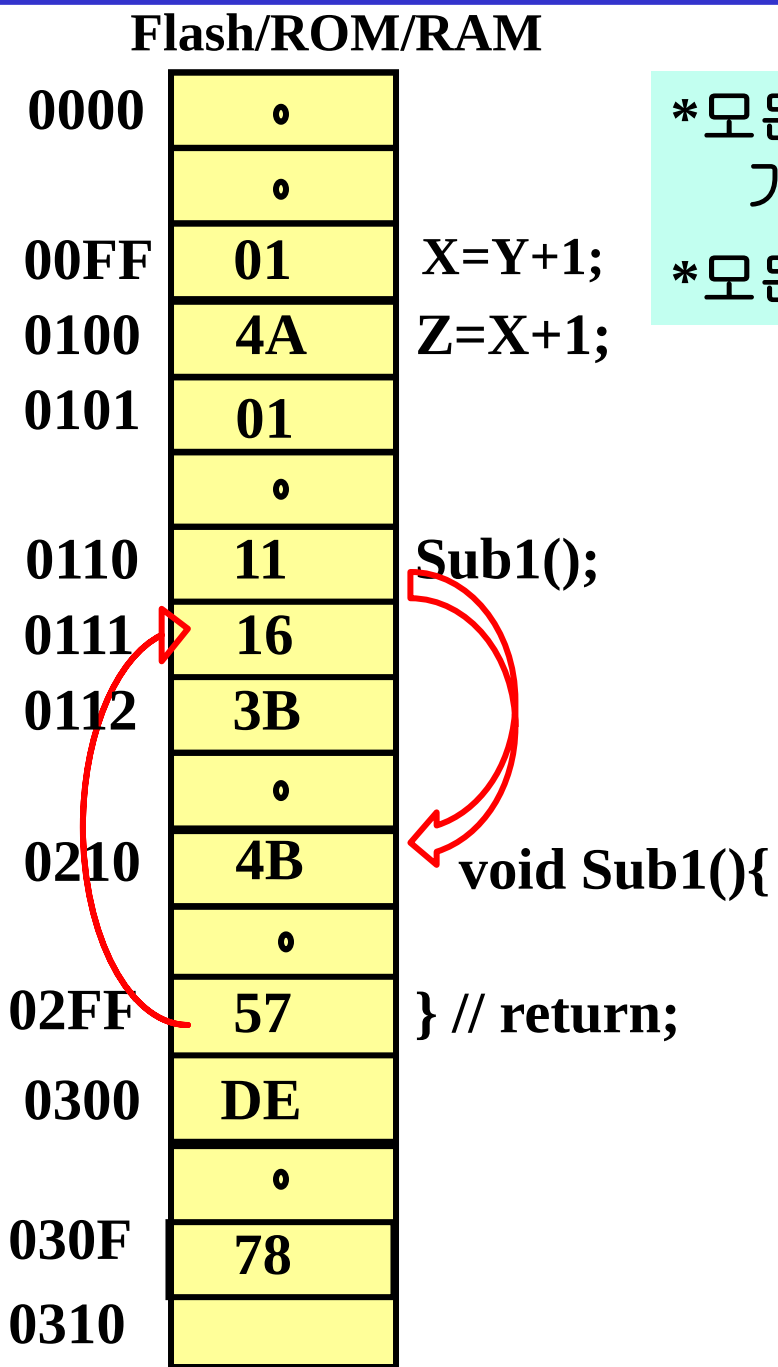
<POP 동작: 데이터 Read => stack pointer 증가>





< 서브루틴 Call 실행시 STACK 사용 예>

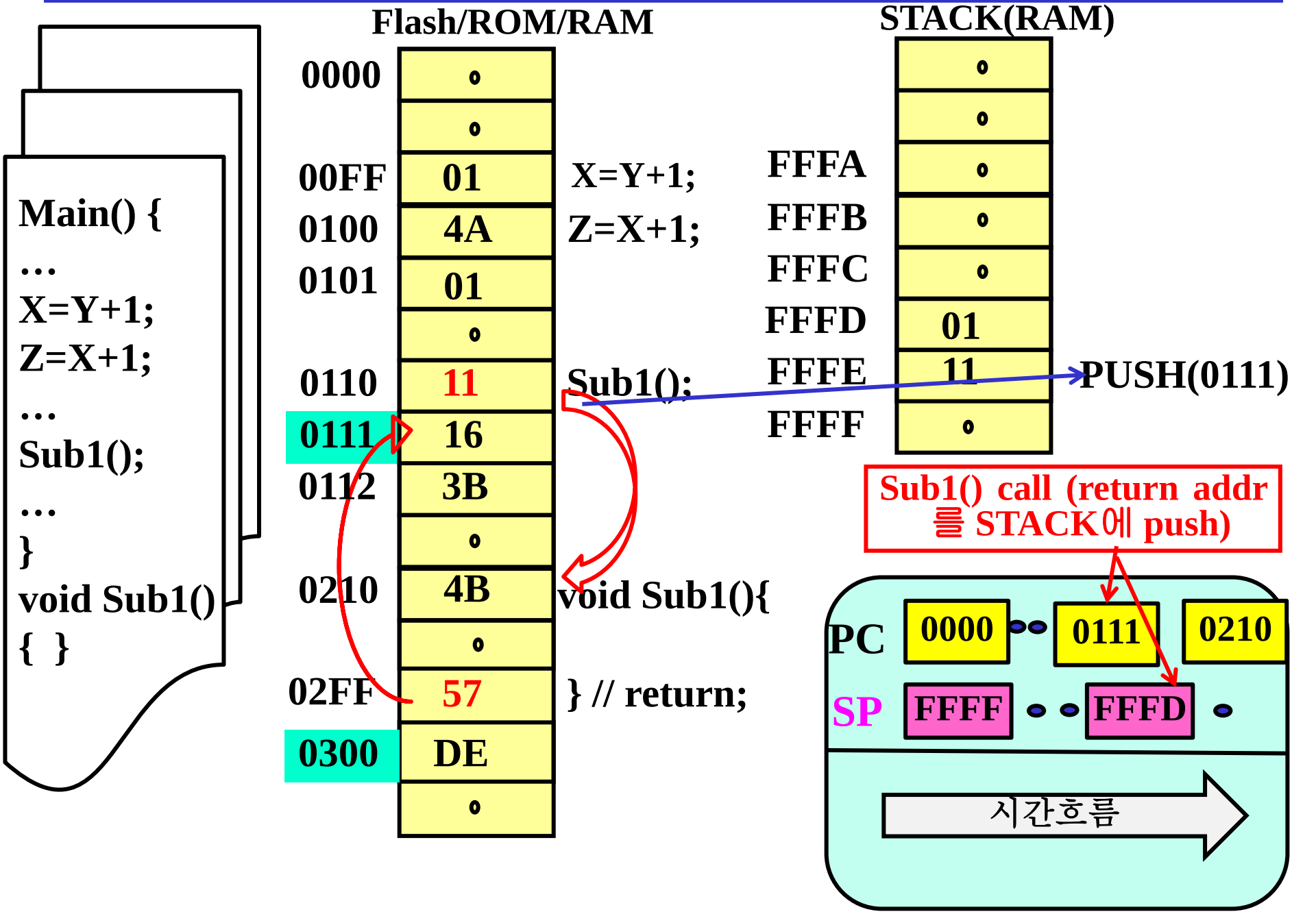




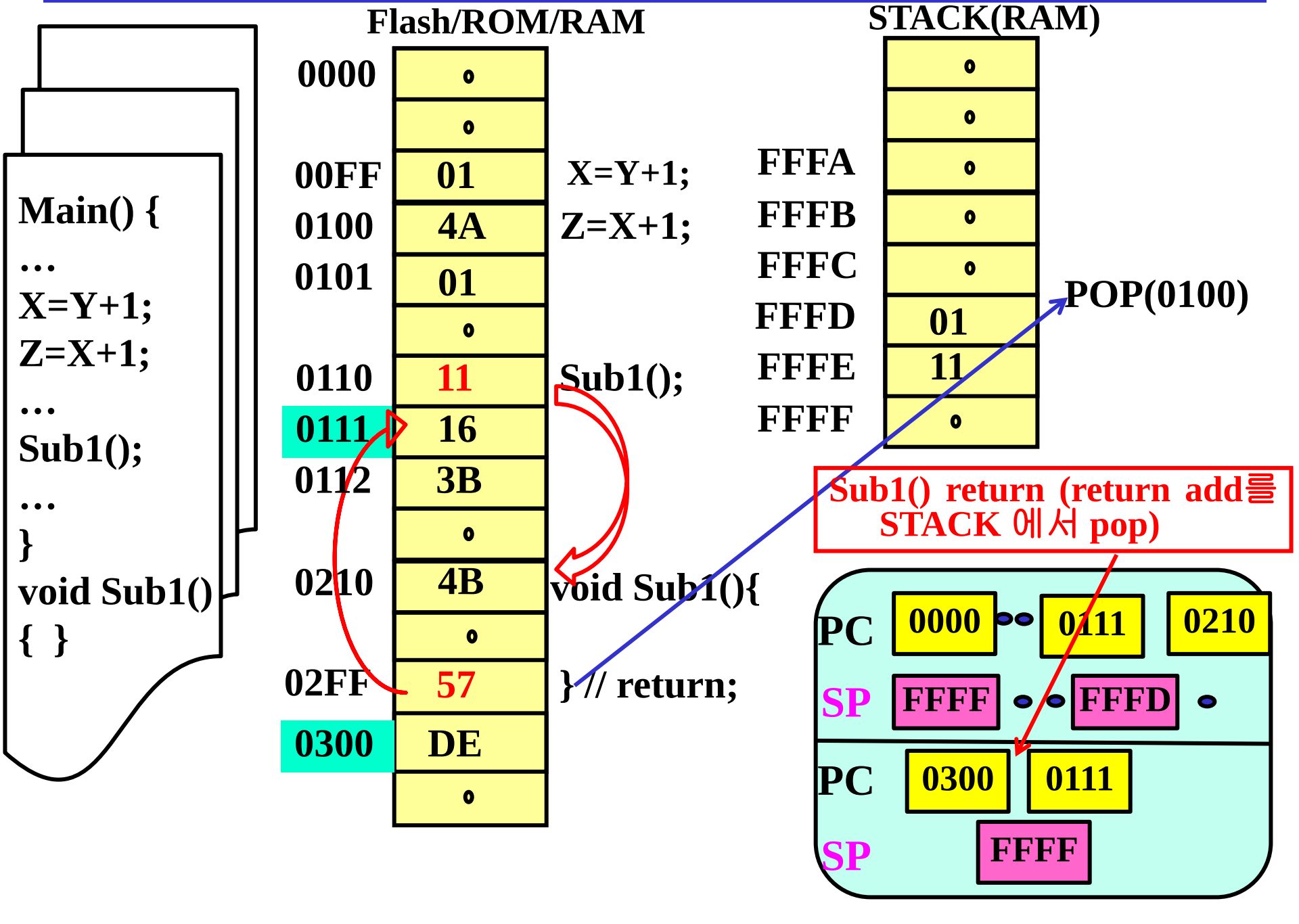
*모든 C 명령어는 1byte의 기계어로 번역된다고 가정

*모든 수는 16진수

< 서브루틴 Call 실행시 STACK 사용 예>



< 서브루틴 Call 실행시 STACK 사용 예>



< 인터럽트/서브루틴 Call 실행시 STACK 사용 예>

Interrupt (or 일반 subroutine call) 발생

(1) 현재PC 에 저장된주소(return addr)를 stack에 저장 (PUSH)

Stack ← PC

(2) Interrupt service routine(or 일반 subroutine)의 시작번지를 PC에 저장

(3) Interrupt service routine(or 일반 subroutine) 으로 jump

Interrupt (or 일반 subroutine call) Return

(1) Stack에서 return addr를 꺼내와(POP) PC에 저장

PC ← Stack

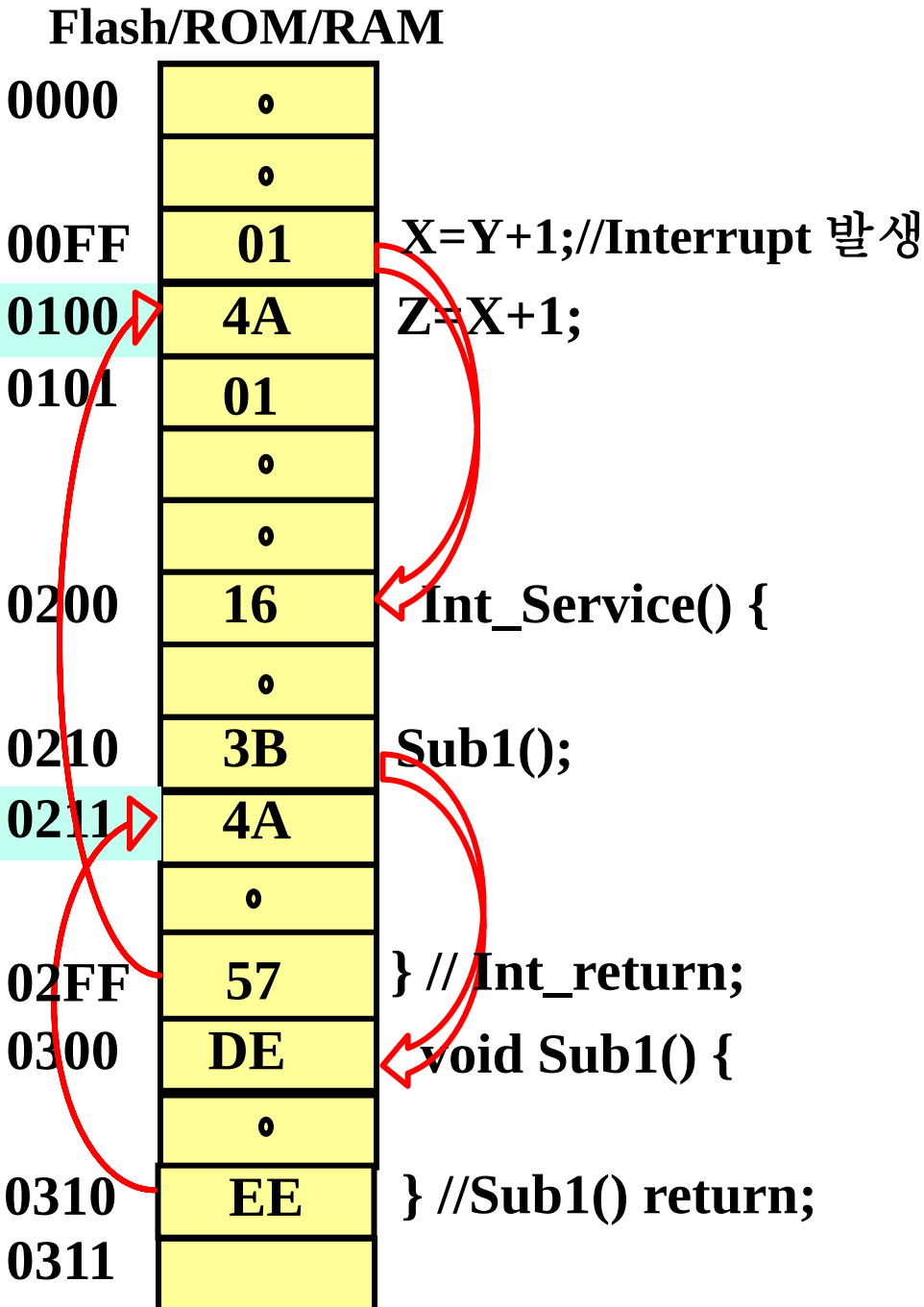
(2) Return addr (interrupt 발생 지점/subroutine call 지점) 지점으로 jump

< 인터럽트/서브루틴을 사용한 프로그램 예 >

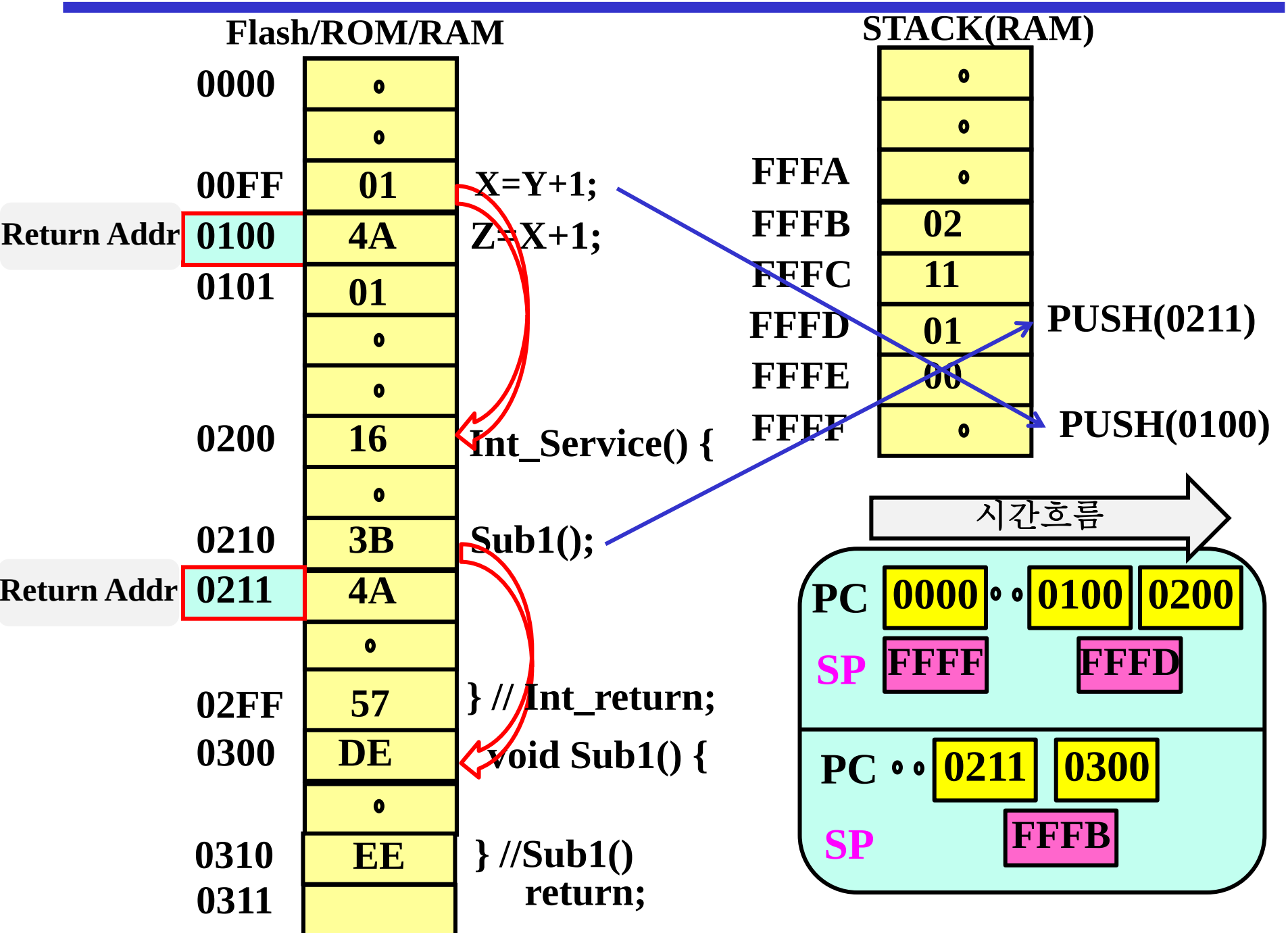
*모든 C 명령어는 1byte의 기계어로 번역된다고 가정

*모든 수는 16진수

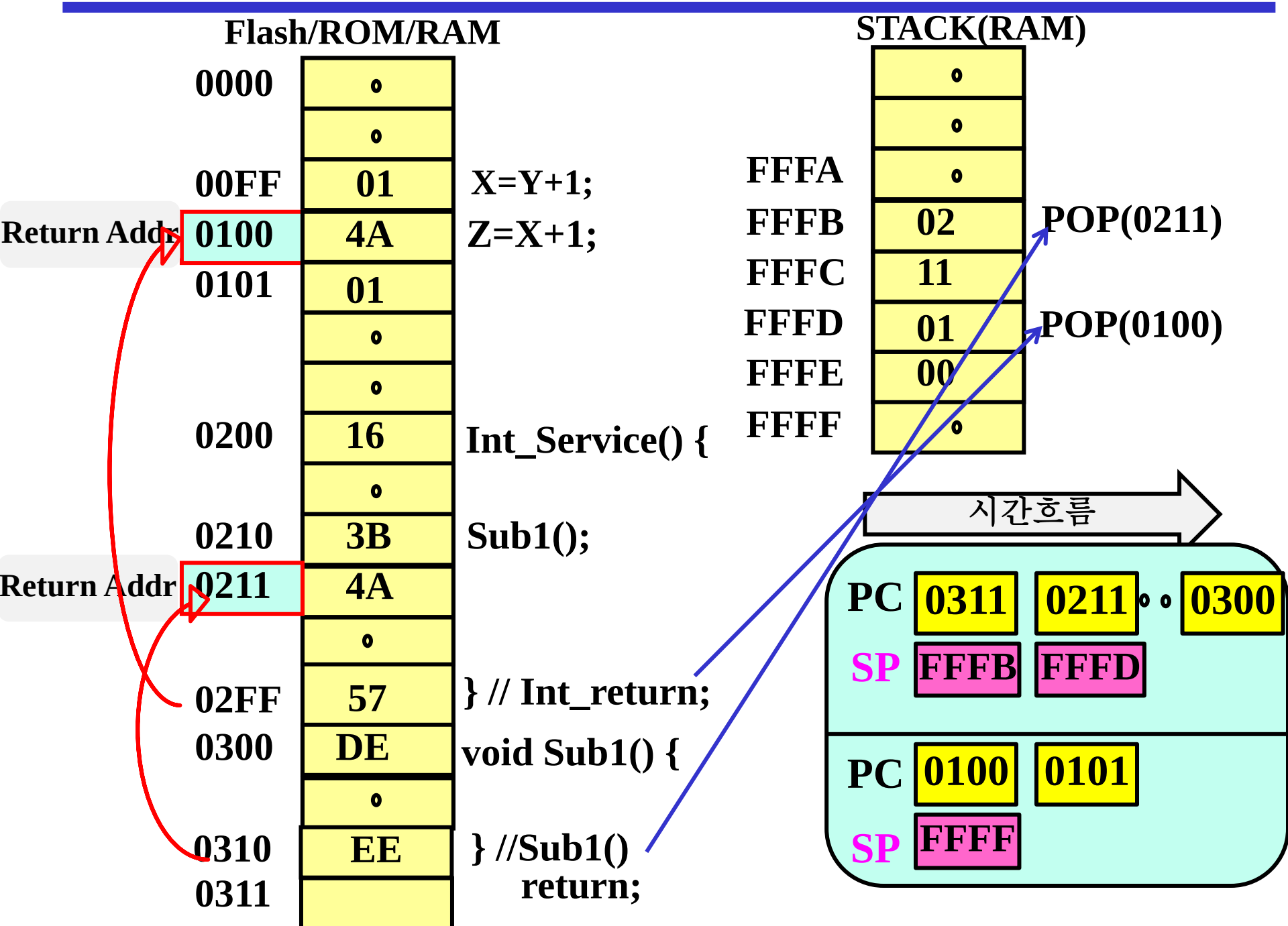
```
Main() {
…
X=Y+1;
Z=X+1;
…
}

Int_Service()
{ Sub1(); }

void Sub1()
{ }
```



< 인터럽트/서브루틴을 사용한 프로그램 예: PUSH >



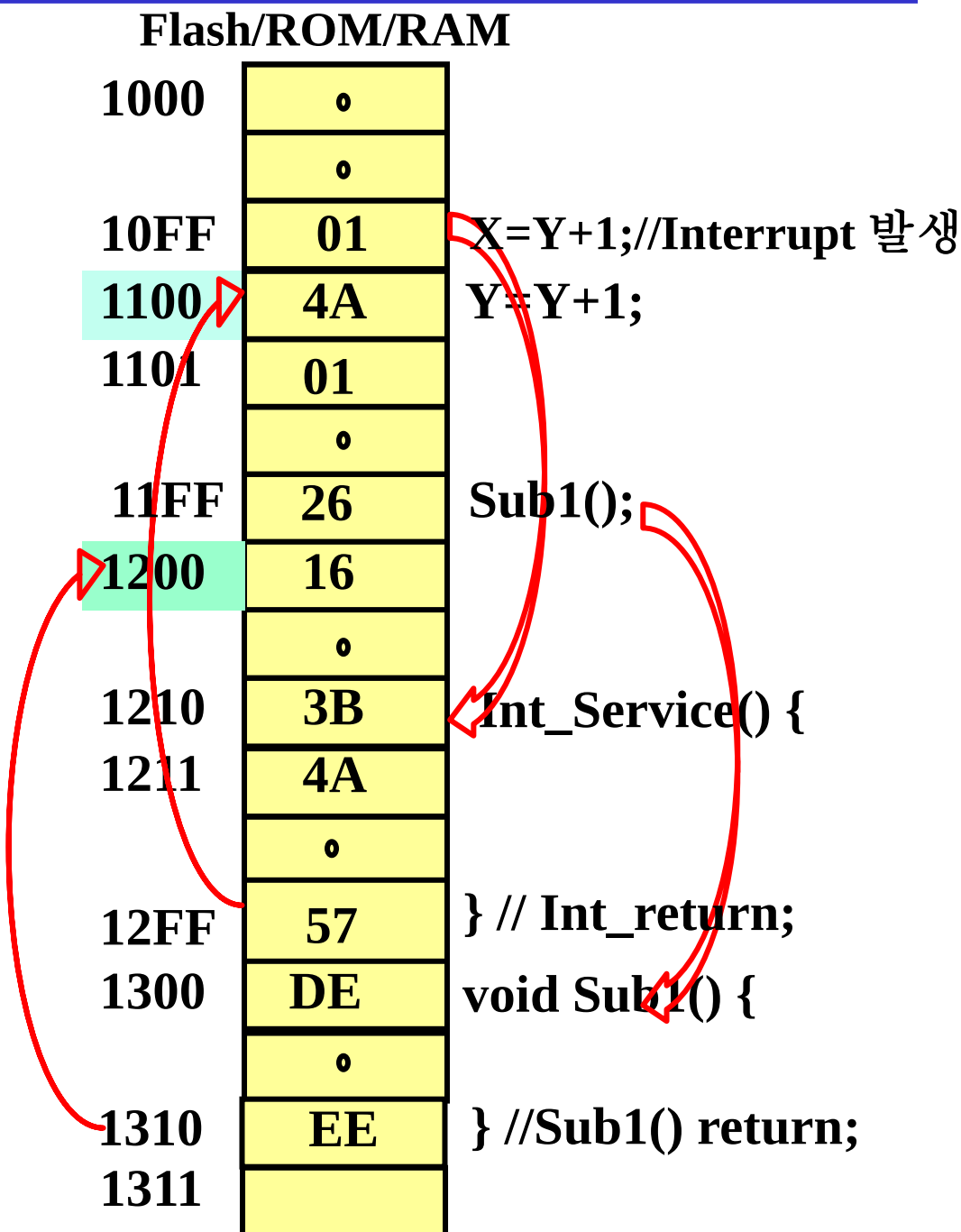
< 인터럽트/서브루틴을 사용한 프로그램 예: POP >

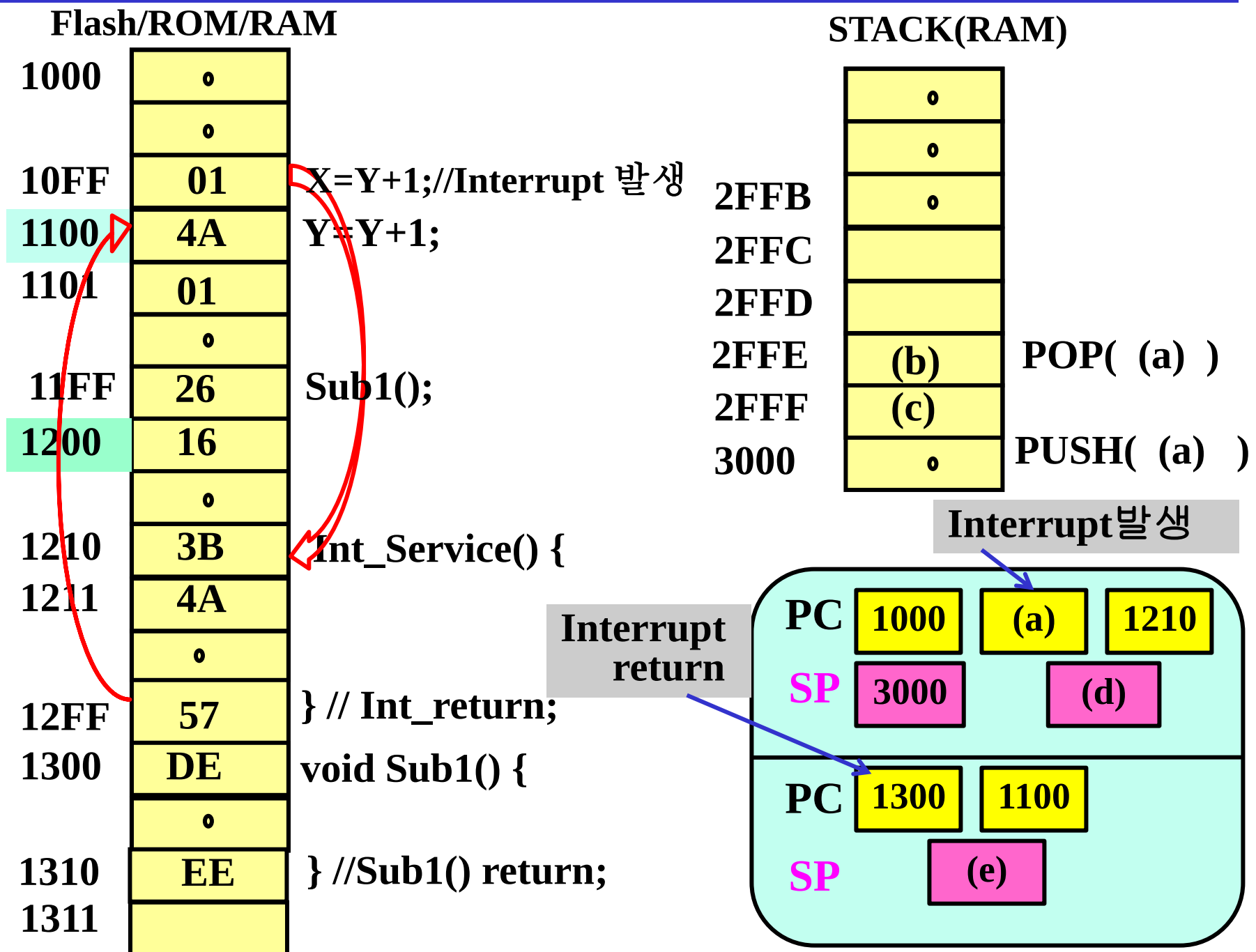


*모든 C 명령어는 1byte의 기계어로 번역된다고 가정

*모든 수는 16진수

```
Main() {
…
X=Y+1;
Y=Y+1;
…
Sub1();
…
}
Int_Service()
{ }
void Sub1()
{ }
```




Flash/ROM/RAM
1000
10FF
01
X=Y+1;//Interrupt 발생
1100
4A
Y=Y+1;
1101
01
11FF
26
Sub1();
1200
16
1210
3B
Int_Service() {
1211
4A
12FF
57
} // Int_return;
1300
DE
void Sub1() {
1310
EE
} //Sub1() return;
1311
STACK(RAM)
2FFB
2FFC
2FFD
2FFE
(b)
POP((a))
2FFF
(c)
3000
PUSH((a))
Interrupt발생
PC
1000
(a)
1210
SP
3000
(d)
Interrupt return
PC
1300
1100
SP
(e)

Flash/ROM/RAM

1000	o	
	o	
10FF	01	X=Y+1;//Interrupt 발생
1100	4A	Y=Y+1;
1101	01	
	o	
11FF	26	Sub1();
1200	16	
	o	
1210	3B	Int_Service() {
1211	4A	
	o	
12FF	57	} //Int_return;
1300	DE	void Sub1() {
	o	
1310	EE	} //Sub1() return;
1311		

STACK(RAM)

	o	
	o	
2FFB	o	
2FFC		
2FFD		
2FFE	(b)	POP((a))
2FFF	(c)	
3000	o	PUSH((a))