

Chapter 02. 컴퓨터 구조

컴퓨터가 이해하는 정보

장동호

이것이 ^{취업을 위한} 컴퓨터 과학이다

with CS 기술 면접



취업과 이직을 결정하는 필수 CS 지식+기술 면접 가이드

강민철 지음



한빛미디어

020304

목차

1. 데이터

- 실수 표현하기
- 문자 표현하기

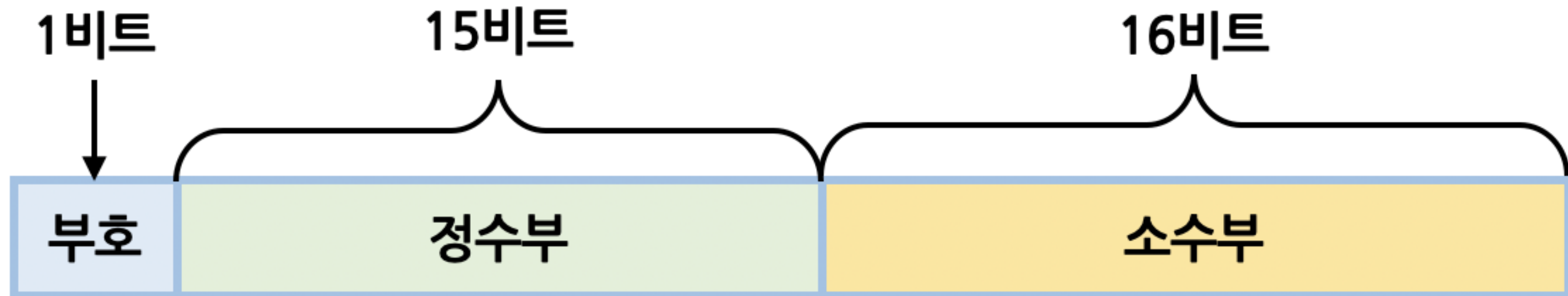
2. 명령어

- 명령어의 종류
- 명령어 주소 지정 방식

1. 데이터

실수 표현하기

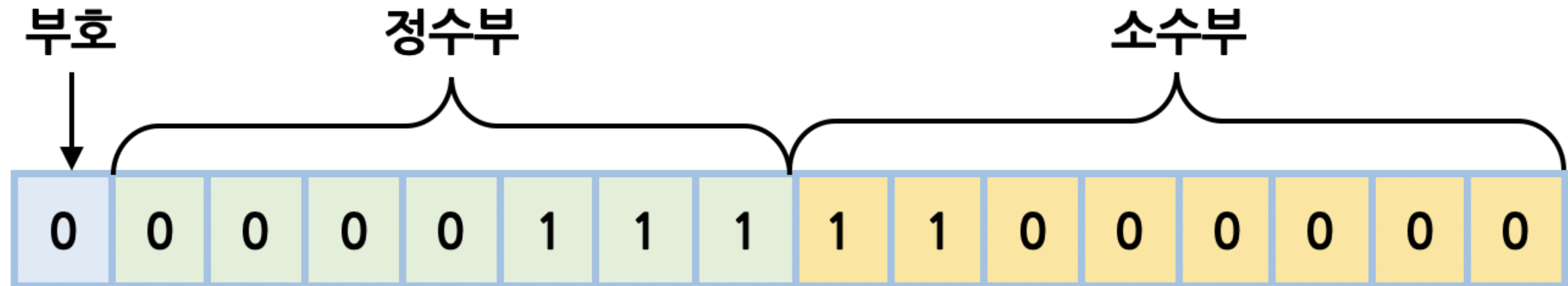
- 10진수 7.75 을 컴퓨터는 어떻게 2진수로 저장할까?



고정 소수점 방식

실수 표현하기

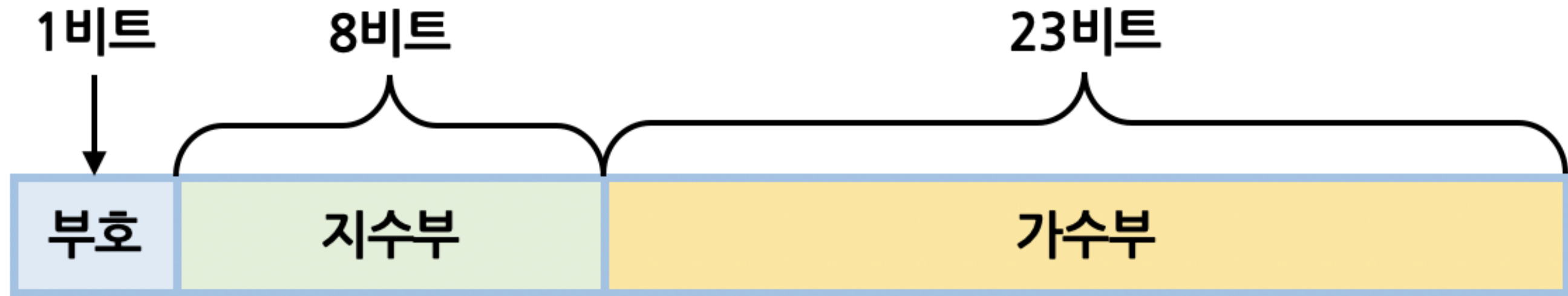
- 소수점의 위치를 고정시킨 후 표현하는 고정 소수점 방식
- 표현 가능한 실수의 범위가 제한적이고 아주 큰 수나 아주 작은 수를 정확하게 표현하기 어렵다는 한계가 존재



실수 7.75를 2진수로 변환 (고정 소수점 방식)

실수 표현하기

- 소수점의 위치를 자유롭게 조정할 수 있는 부동소수점 방식 도입
- 수의 크기에 따라 **가수**와 **지수**를 분리하여 표현

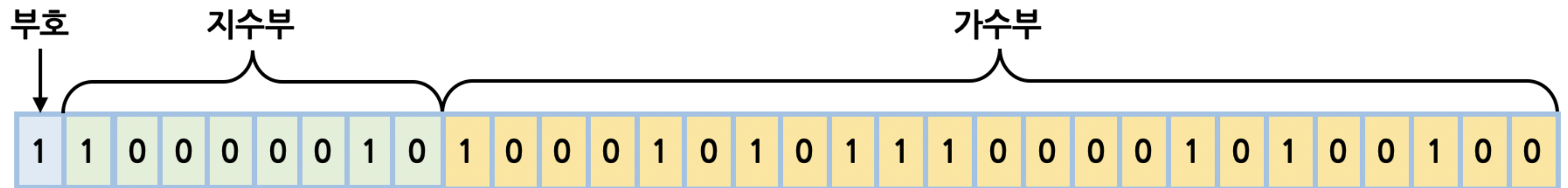


IEEE 부동 소수점 방식

실수 표현하기

- 10진수 -12.34를 부동 소수점 방식으로 2진수로 변환해보자.

$$\begin{aligned} &\overset{\text{가수}}{\underbrace{-12.34}} \times \overset{\text{밑수}}{\underbrace{10}}^{\text{지수 } 1} = -1100.01010\dots \\ &= -1.10001010\dots \times 2^3 \end{aligned}$$

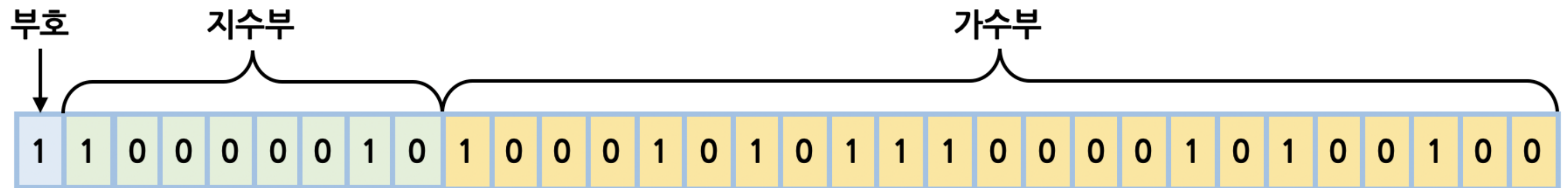


실수 -12.34를 2진수로 변환 (부동 소수점 방식)

실수 표현하기

- 10진수 -12.34를 부동 소수점 방식으로 2진수로 변환해보자.

$$\begin{aligned} &\overset{\text{가수}}{\underbrace{-12.34}} \times \overset{\text{밑수}}{\underbrace{10}}^{\text{지수 } 1} = -1100.01010\dots \\ &= -1.10001010\dots \times 2^3 \end{aligned}$$

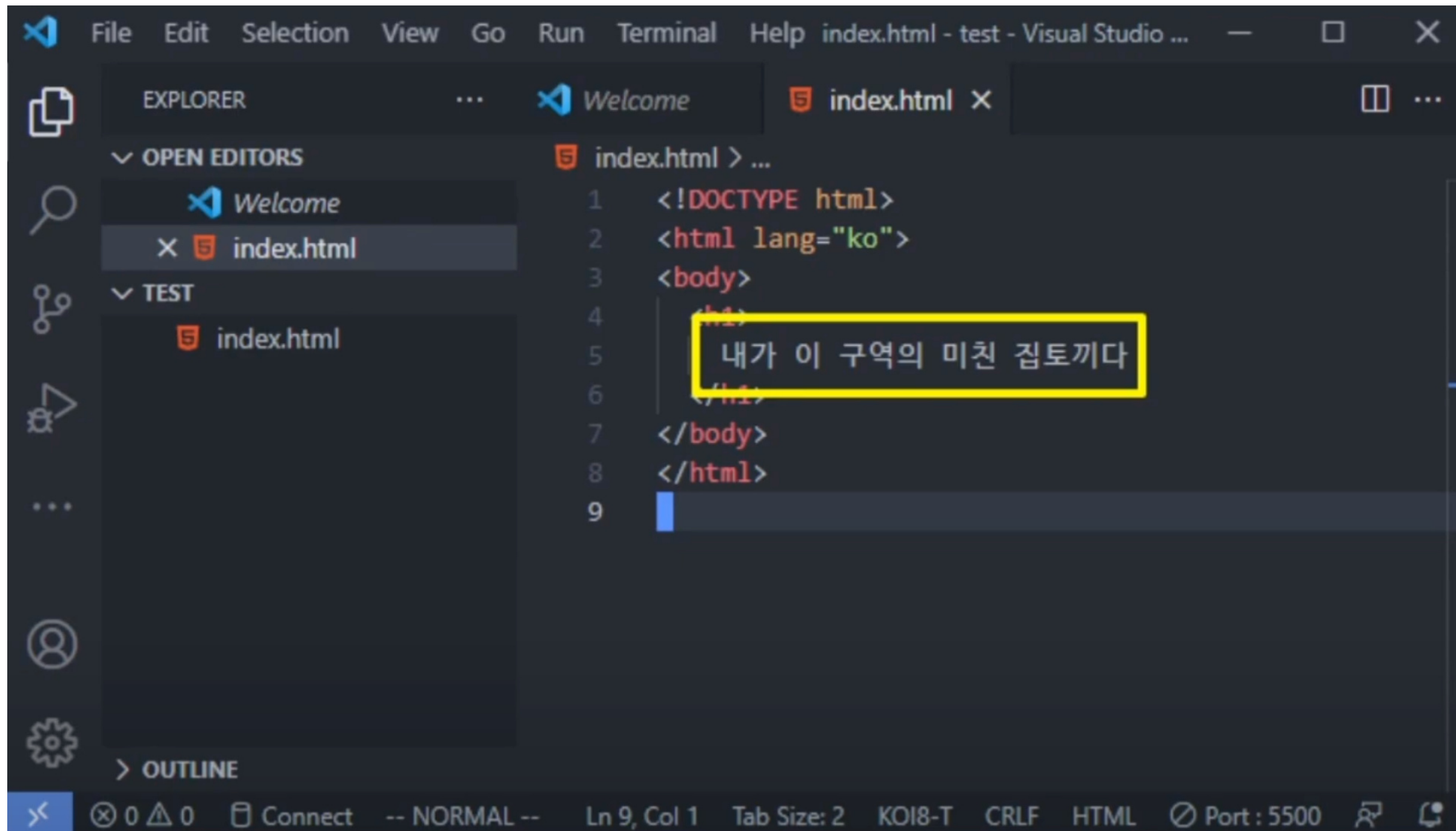


실수 -12.34를 2진수로 변환 (부동 소수점 방식)

실수 표현하기

- 가수가 23비트를 초과하는 경우 일부 정보가 손실
 - 10진수 $0.1 = 2$ 진수 $0.00011001100\dots$ (무한 반복)
- 은행 시스템에서는 0.01원 단위까지 정확히 계산해야 하므로 고정소수점 방식을 사용하기도 한다.
 - ex) Java의 BigDecimal

문자 표현하기



The screenshot shows the Visual Studio Code interface with the following details:

- Explorer Panel:** Shows 'OPEN EDITORS' with 'Welcome' and 'index.html' (selected). The 'TEST' panel also shows 'index.html'.
- Editor Panel:** Displays the content of 'index.html' with the following HTML code:

```
1 <!DOCTYPE html>
2 <html lang="ko">
3 <body>
4   <h1>
5     내가 이 구역의 미친 집토끼다
6   </h1>
7 </body>
8 </html>
9
```

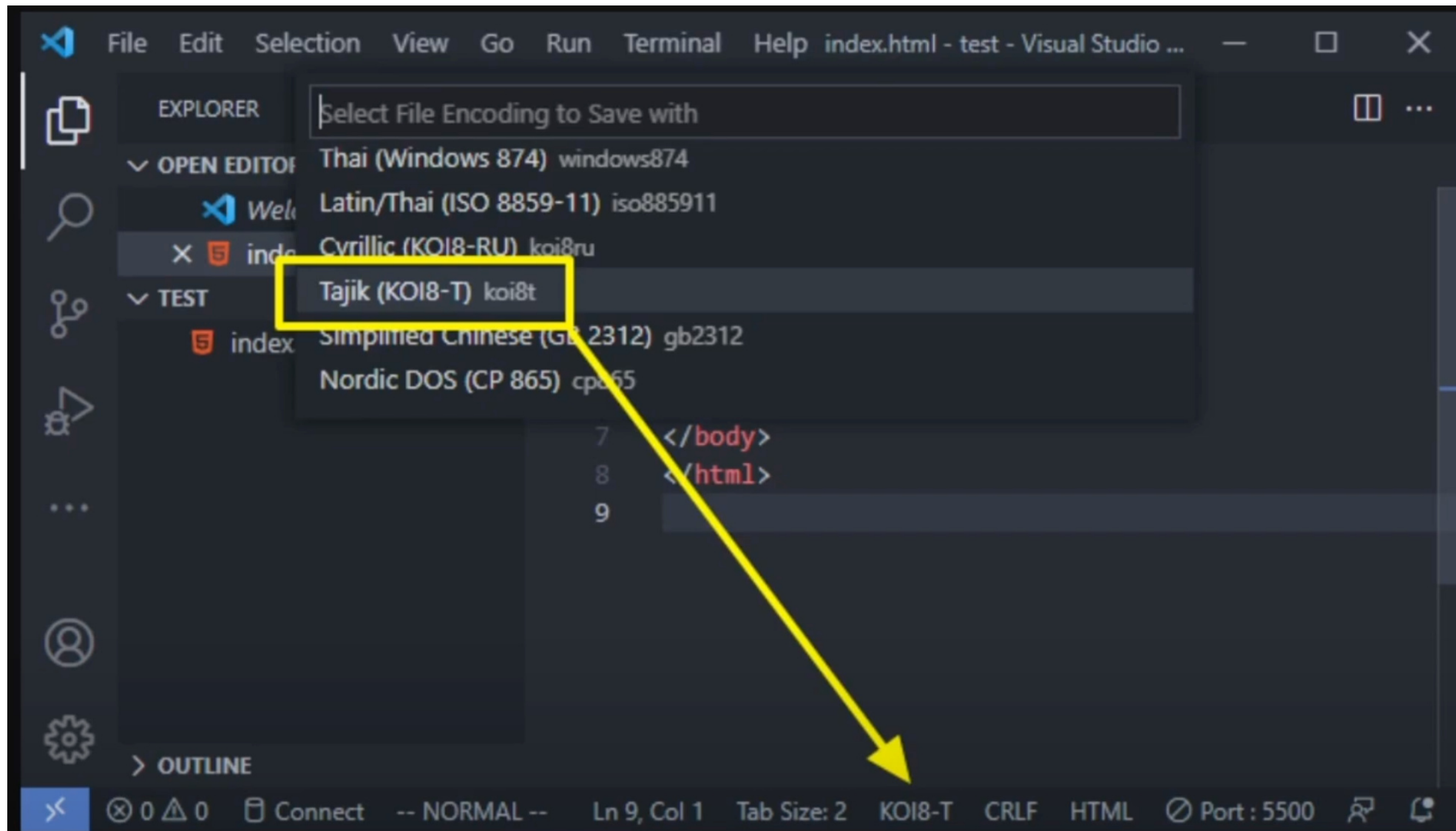
The text '내가 이 구역의 미친 집토끼다' on line 5 is highlighted with a yellow rectangular box.
- Status Bar:** At the bottom, it shows 'Ln 9, Col 1', 'Tab Size: 2', 'KOI8-T', 'CRLF', 'HTML', and 'Port : 5500'.

문자 표현하기

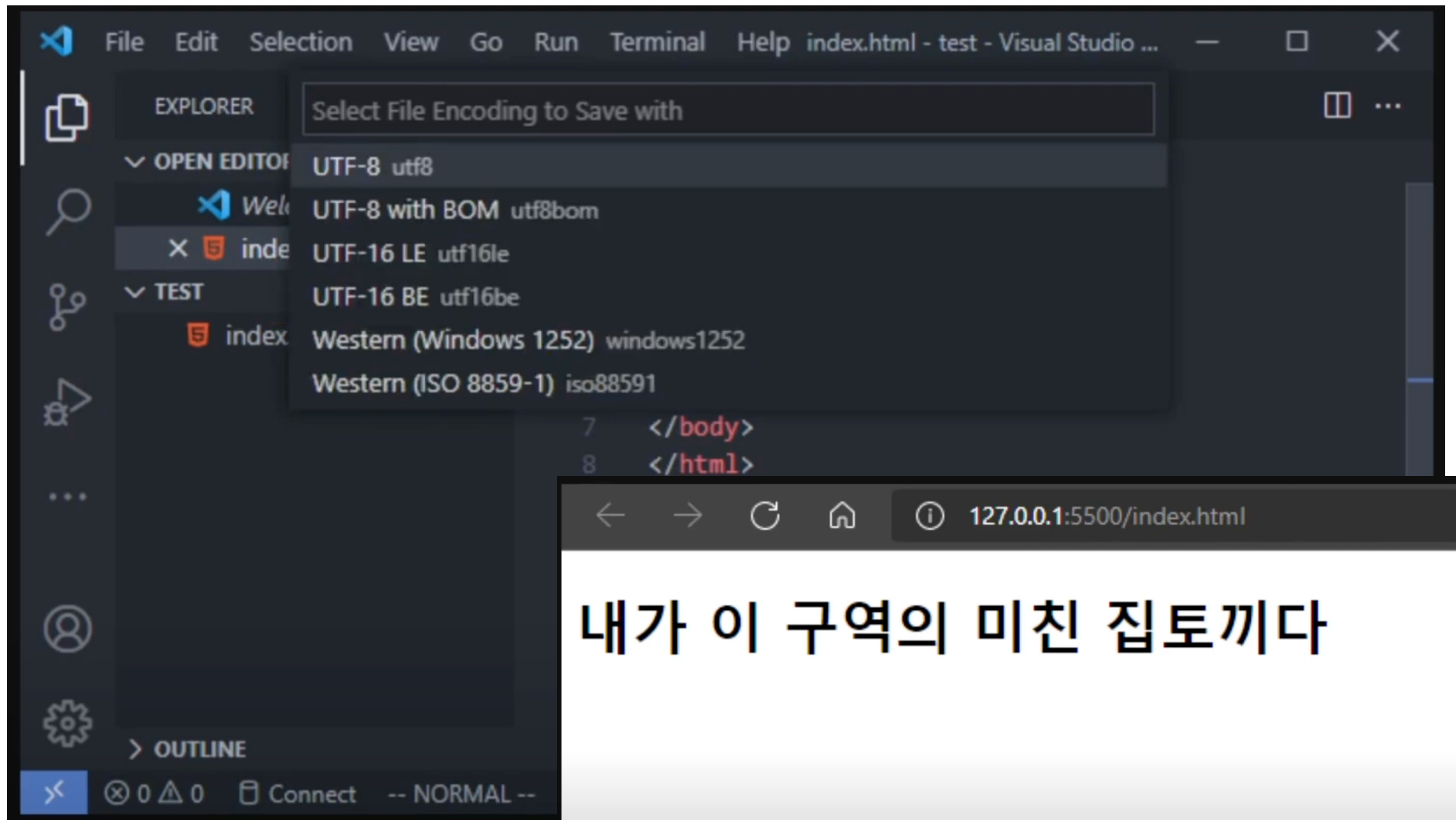


? ? ??? ?? ????

문자 표현하기



문자 표현하기



문자 표현하기

- **문자 인코딩:**

문자를 컴퓨터에 저장하기 위해 숫자(2진수)로 변환하는 것

ex) A -> 1000001

- **문자 디코딩:**

반대로 인코딩한 숫자를 읽기 위한 문자로 변환하는 것

ex) 1000001 -> A

문자 표현하기

제어 문자		공백 문자		구두점		숫자		알파벳			
10진	16진	문자	10진	16진	문자	10진	16진	문자	10진	16진	문자
0	0x00	NUL	32	0x20	SP	64	0x40	@	96	0x60	`
1	0x01	SOH	33	0x21	!	65	0x41	A	97	0x61	a
2	0x02	STX	34	0x22	"	66	0x42	B	98	0x62	b
3	0x03	ETX	35	0x23	#	67	0x43	C	99	0x63	c
4	0x04	EOT	36	0x24	\$	68	0x44	D	100	0x64	d
5	0x05	ENQ	37	0x25	%	69	0x45	E	101	0x65	e
6	0x06	ACK	38	0x26	&	70	0x46	F	102	0x66	f
7	0x07	BEL	39	0x27	'	71	0x47	G	103	0x67	g
8	0x08	BS	40	0x28	(72	0x48	H	104	0x68	h
9	0x09	HT	41	0x29)	73	0x49	I	105	0x69	i
10	0x0A	LF	42	0x2A	*	74	0x4A	J	106	0x6A	j
11	0x0B	VT	43	0x2B	+	75	0x4B	K	107	0x6B	k
12	0x0C	FF	44	0x2C	,	76	0x4C	L	108	0x6C	l
13	0x0D	CR	45	0x2D	-	77	0x4D	M	109	0x6D	m
14	0x0E	SO	46	0x2E	.	78	0x4E	N	110	0x6E	n
15	0x0F	SI	47	0x2F	/	79	0x4F	O	111	0x6F	o
16	0x10	DLE	48	0x30	0	80	0x50	P	112	0x70	p
17	0x11	DC1	49	0x31	1	81	0x51	Q	113	0x71	q
18	0x12	DC2	50	0x32	2	82	0x52	R	114	0x72	r
19	0x13	DC3	51	0x33	3	83	0x53	S	115	0x73	s
20	0x14	DC4	52	0x34	4	84	0x54	T	116	0x74	t
21	0x15	NAK	53	0x35	5	85	0x55	U	117	0x75	u
22	0x16	SYN	54	0x36	6	86	0x56	V	118	0x76	v
23	0x17	ETB	55	0x37	7	87	0x57	W	119	0x77	w
24	0x18	CAN	56	0x38	8	88	0x58	X	120	0x78	x
25	0x19	EM	57	0x39	9	89	0x59	Y	121	0x79	y
26	0x1A	SUB	58	0x3A	:	90	0x5A	Z	122	0x7A	z
27	0x1B	ESC	59	0x3B	;	91	0x5B	[123	0x7B	{
28	0x1C	FS	60	0x3C	<	92	0x5C	₩	124	0x7C	
29	0x1D	GS	61	0x3D	=	93	0x5D]	125	0x7D	}
30	0x1E	RS	62	0x3E	>	94	0x5E	^	126	0x7E	~
31	0x1F	US	63	0x3F	?	95	0x5F	_	127	0x7F	DEL

아스키 코드

문자 표현하기

	AC0	AC1	AC2	AC3	AC4	AC5	AC6	AC7	AC8	AC9	ACA	ACB	ACC	ACD	ACE	ACF
0	가	감	감	갇	갈	각	갸	거	검	겐	겉	결	격	겻	고	곰
	AC00	AC10	AC20	AC30	AC40	AC50	AC60	AC70	AC80	AC90	ACA0	ACB0	ACC0	ACD0	ACE0	ACF0
1	각	갑	갸	갱	갹	갈	갼	걱	겁	겉	겡	겨	결	겻	곡	굽
	AC01	AC11								AC91	ACA1	ACB1	ACC1	ACD1	ACE1	ACF1
2	갇	갸	갹	갼	갽	갾	갿	갻	갼	갽	갾	갿	갻	갼	갽	갾
	AC02	AC12	AC22	AC32	AC42	AC52	AC62	AC72	AC82	AC92	ACA2	ACB2	ACC2	ACD2	ACE2	ACF2
3	갸	갹	갼	갽	갾	갿	갻	갼	갽	갾	갿	갻	갼	갽	갾	갿
	AC03	AC13	AC23	AC33	AC43	AC53	AC63	AC73	AC83	AC93	ACA3	ACB3	ACC3	ACD3	ACE3	ACF3
4	간	갸	갹	갼	갽	갾	갿	건	겉	겡	겉	겉	계	겻	곤	곶
	AC04	AC14	AC24	AC34	AC44	AC54	AC64	AC74	AC84	AC94	ACA4	ACB4	ACC4	ACD4	ACE4	ACF4
5	갸	강	갸	갼	갽	갾	갿	겉	겡	겉	겉	겉	겉	겉	곶	공
	AC05	AC15	AC25	AC35	AC45	AC55	AC65	AC75	AC85	AC95	ACA5	ACB5	ACC5	ACD5	ACE5	ACF5
6	갸	갸	갸	갼	갽	갾	갿	겉	겉	겉	겉	겉	겉	겉	곶	곶
	AC06	AC16	AC26	AC36	AC46	AC56	AC66	AC76	AC86	AC96	ACA6	ACB6	ACC6	ACD6	ACE6	ACF6

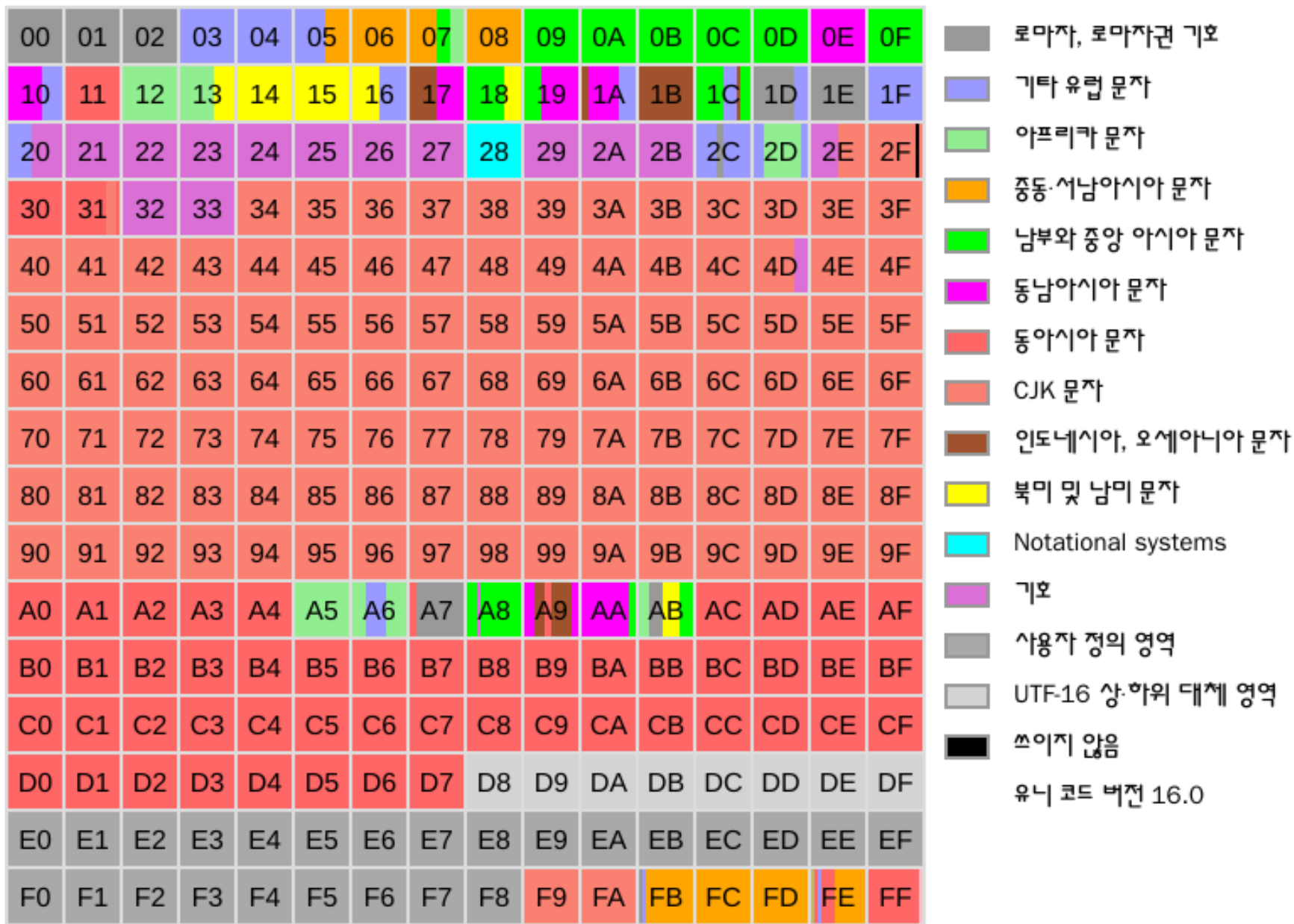
EUC-KR

문자 표현하기



문자열셋의 춘추전국시대

문자 표현하기



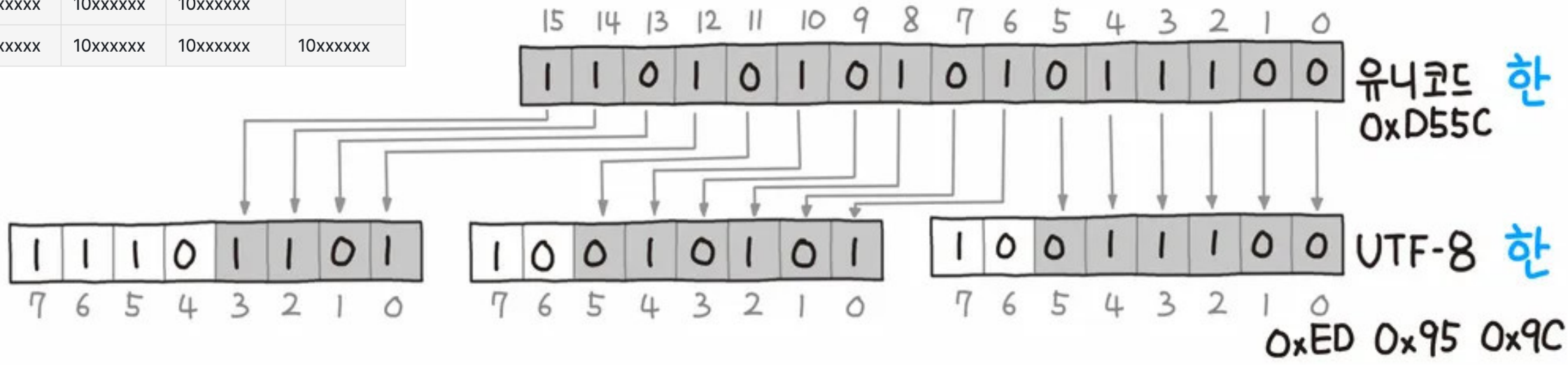
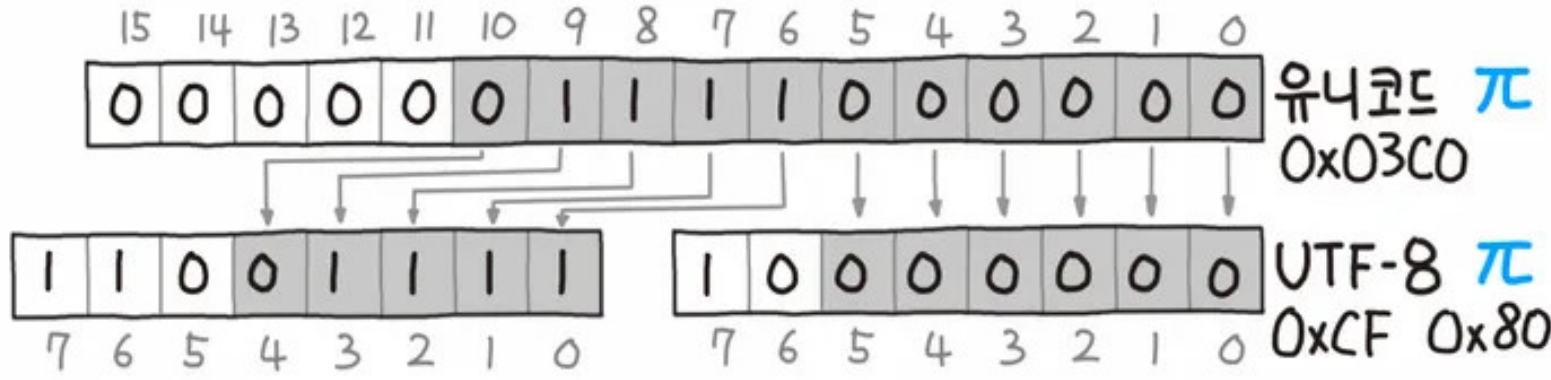
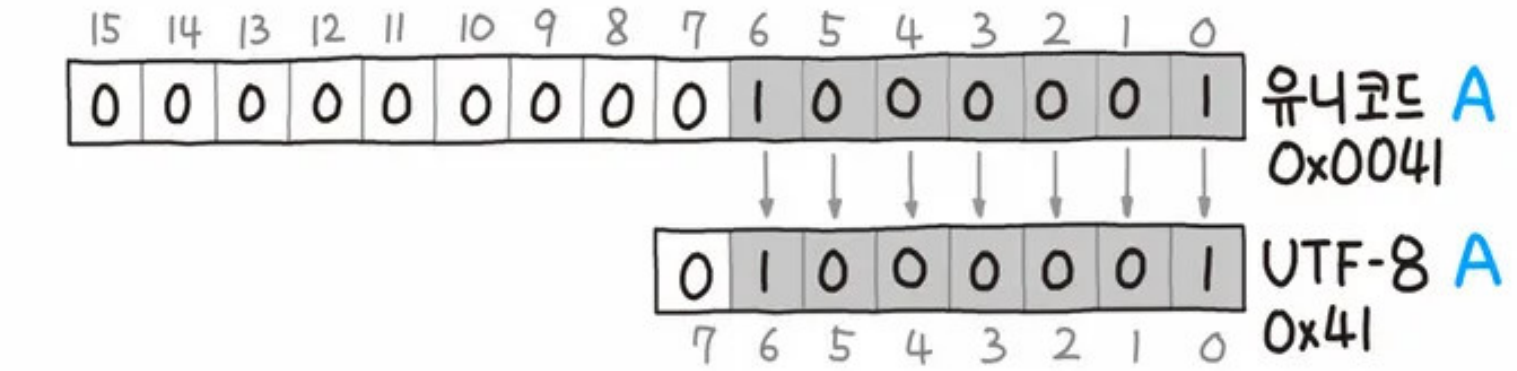
유니코드

문자 표현하기

1. 1개 바이트를 사용:
- 가장 큰 비트에 0을 할당하고, 나머지 7비트에 기존의 아스키 코드를 모두 할당한다. 0xxxxxxx값을 모두 사용하였다. (아직 1xxxxxxx는 사용 가능한 상황이다.)
2. 사용하려는 바이트가 2개가 넘을 때: 첫 바이트에는 몇 바이트를 사용하는지 알려주는 비트를 먼저 넣는다.
- 2바이트(110), 3바이트(1110), 4바이트(11110)

나머지 바이트에는 여러 바이트에서 연결되었음을 알리는 비트를 먼저 넣는다. 이때 2바이트 표식을 넣은 데이터와 겹치지 않도록, 10이라는 비트를 넣어준다.
3. 표식 비트가 아닌 나머지 비트들은 모두 데이터 비트로 사용한다.

유니코드		utf-8로 저장하는 값					
자릿수	코드값 범위	첫 바이트	둘째 바이트	셋째 바이트	넷째 바이트	다섯째 바이트	여섯째 바이트
00~07비트	0 ~ 0×7F ¹²⁷	0xxxxxxx					
08~11비트	0×80 ¹²⁸ ~ 0×7FF ^{2,047}	110xxxxx	10xxxxxx				
12~16비트	0×800 ^{2,048} ~ 0xFFFF ^{65,535}	1110xxxx	10xxxxxx	10xxxxxx			
17~21비트	0×10000 ^{65,536} ~ 0×1FFFFF ^{2,097,151} ^[5]	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx		
22~26비트	(미사용) ^[6]	111110xx	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx	
27~31비트	(미사용) ^[7]	1111110x	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx



2. 명령어

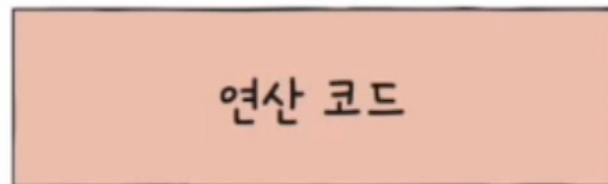
명령어 구조

- CPU가 이해할 수 있는 명령어들의 모음 (CPU의 언어)

더해라	100과	120을
빼라	메모리 32번지 안의 값과	메모리 33번지 안의 값을
저장해라	10을	메모리 128번지에

명령어 구조

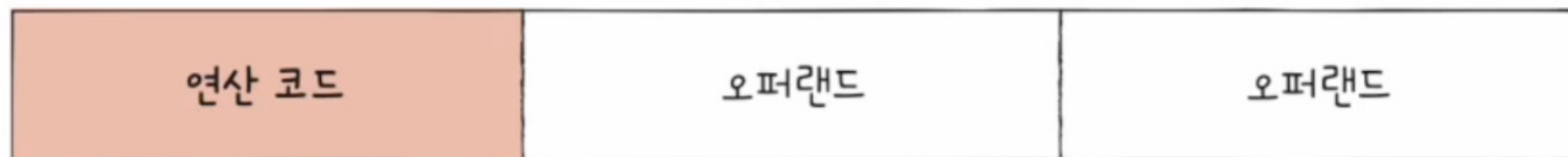
오퍼랜드가 없는 경우(0-주소 명령어)



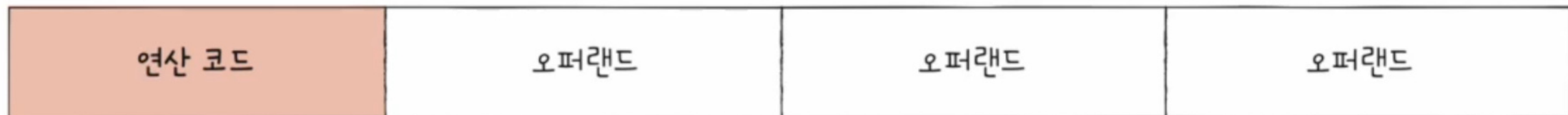
오퍼랜드가 한 개인 경우(1-주소 명령어)



오퍼랜드가 두 개인 경우(2-주소 명령어)



오퍼랜드가 세 개인 경우(3-주소 명령어)

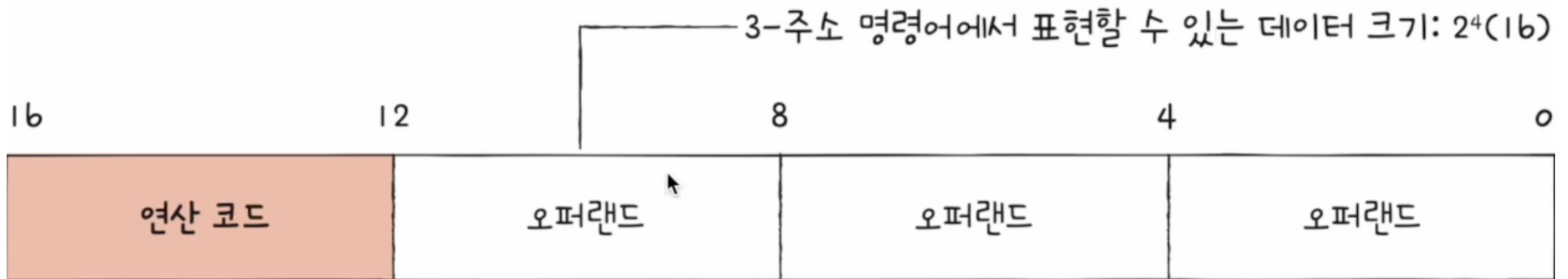


연산 코드

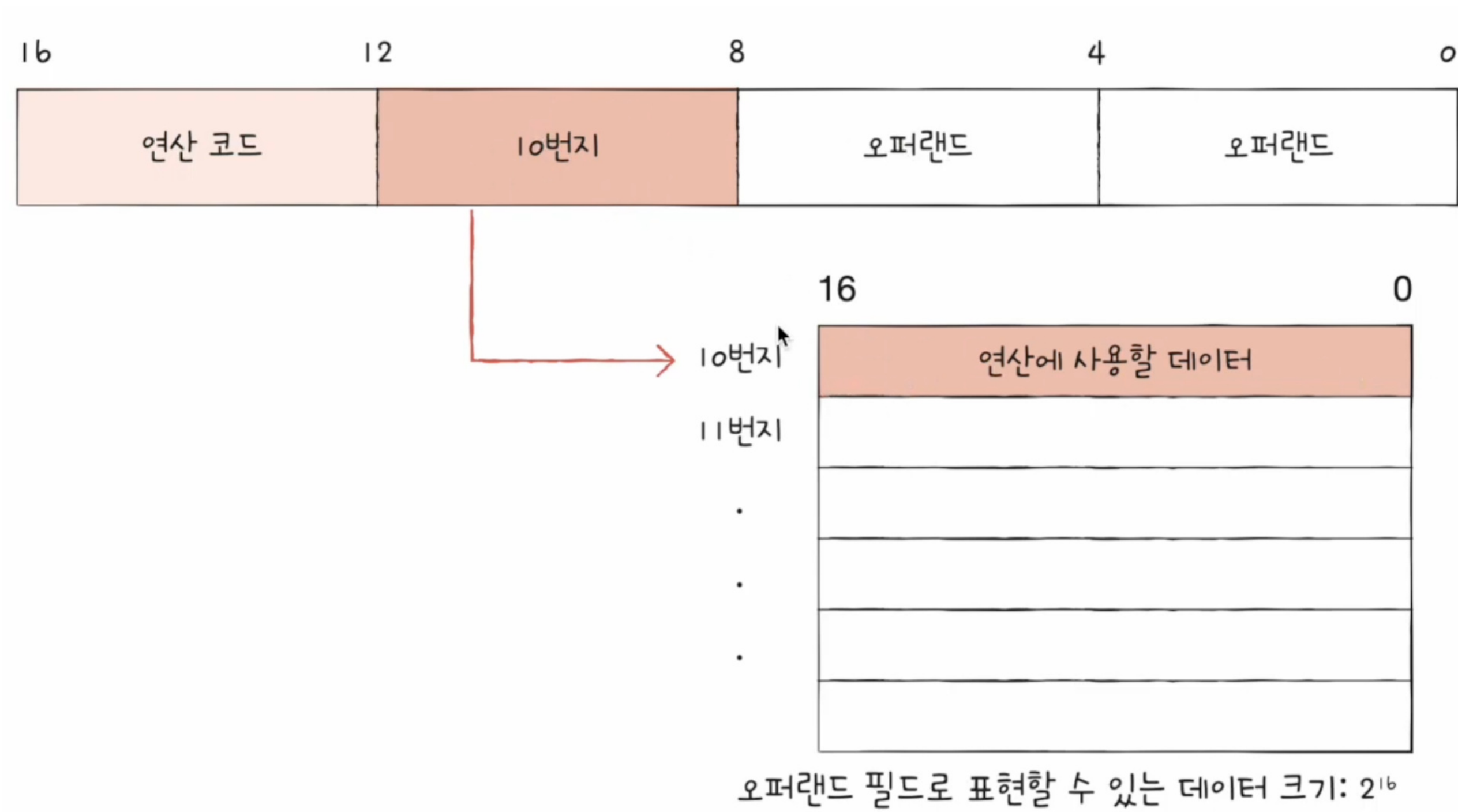
1. 데이터 전송
2. 산술/논리 연산
3. 제어 흐름 변경
4. 입출력 제어

연산 코드

왜 굳이 저장된 위치를 쓰는걸까?



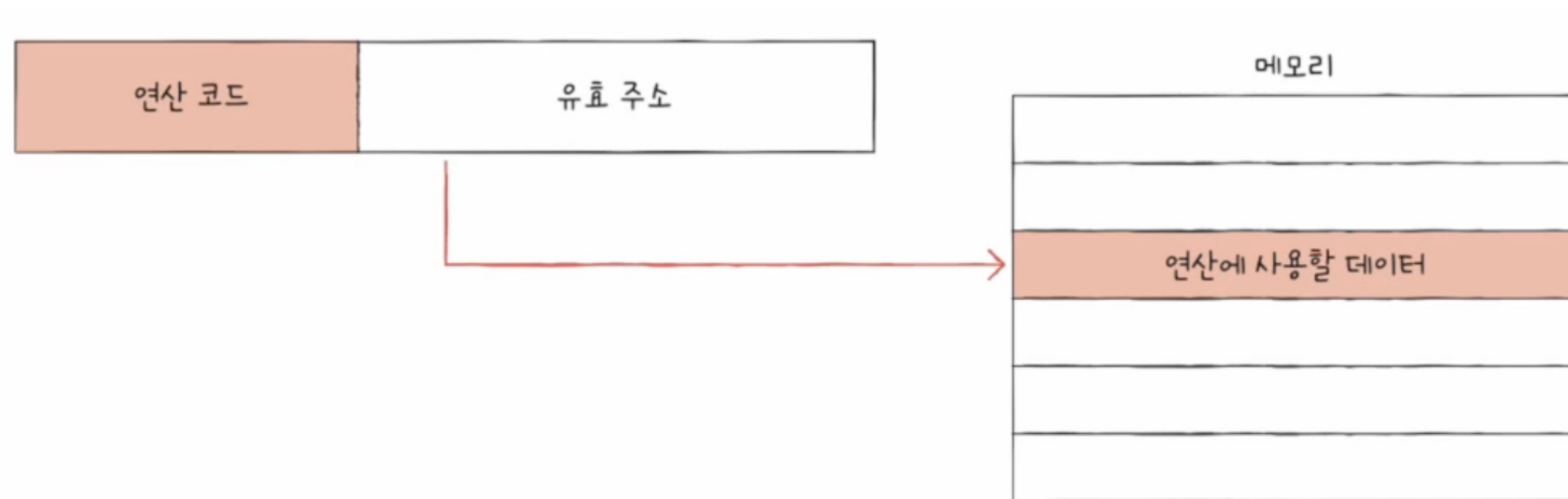
연산 코드



명령어 주소 지정 방식

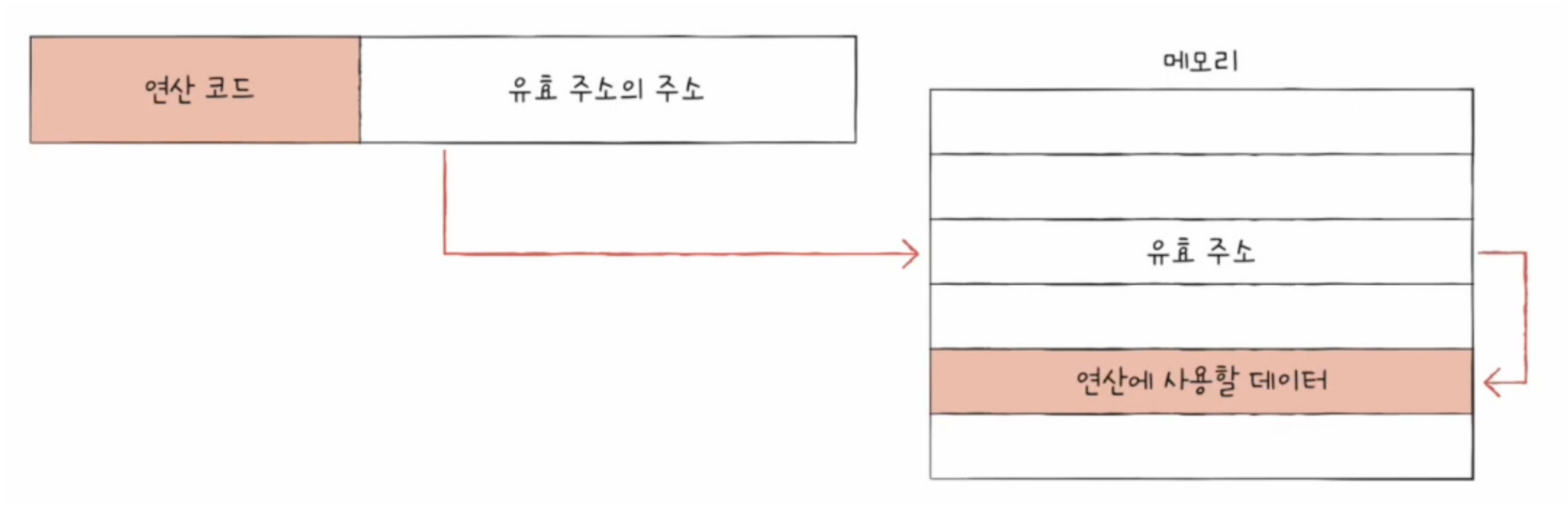


즉시 주소 지정 방식 (immediate addressing mode)



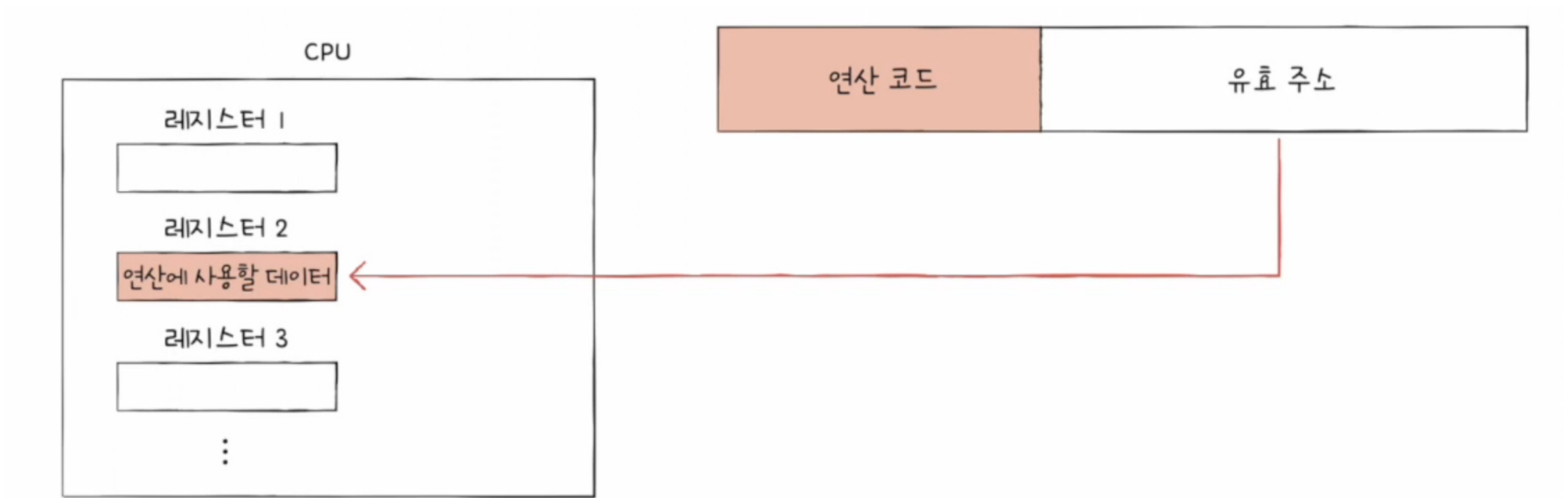
직접 주소 지정 방식 (direct addressing mode)

명령어 주소 지정 방식



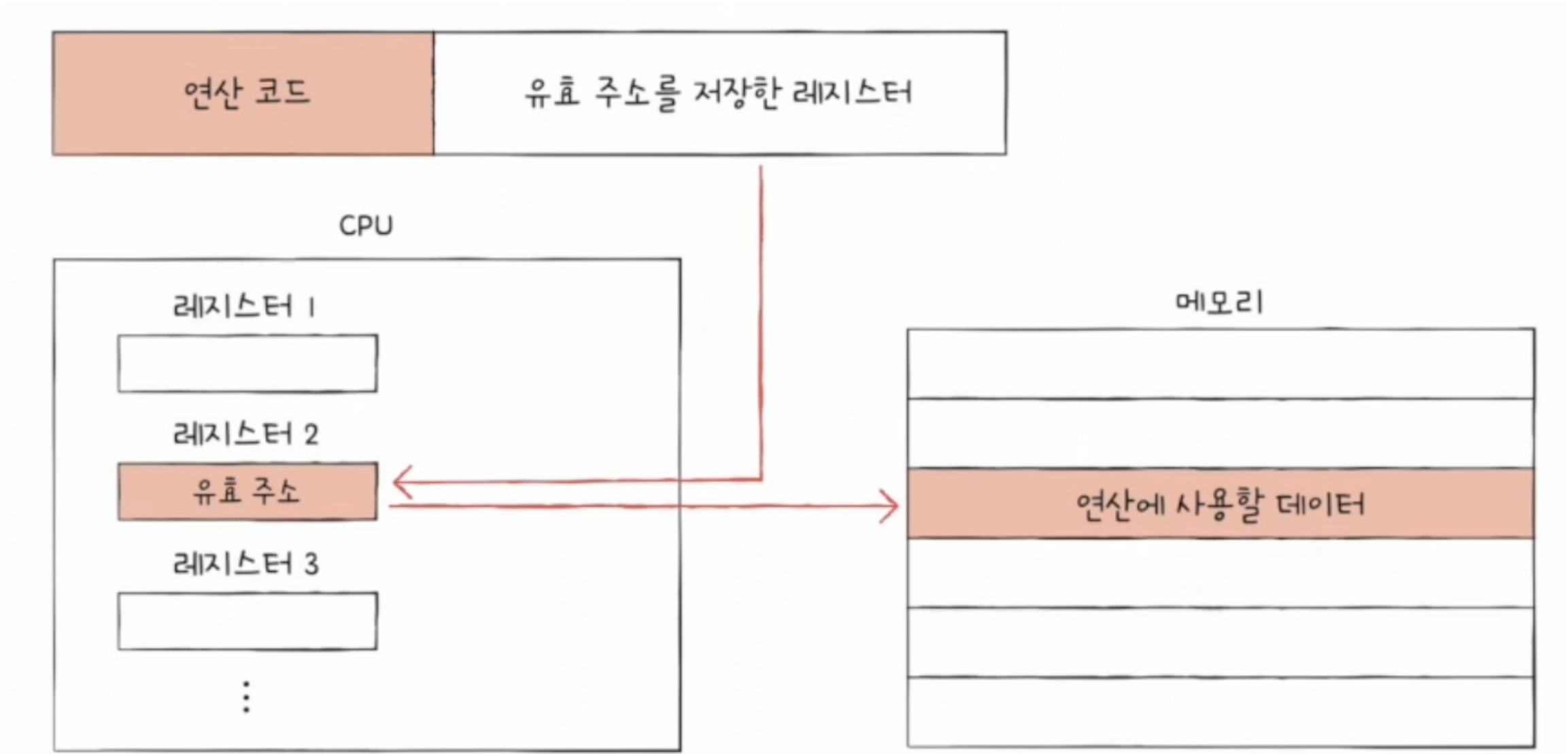
간접 주소 지정 방식 (indirect addressing mode)

명령어 주소 지정 방식



레지스터 주소 지정 방식 (register addressing mode)

명령어 주소 지정 방식



레지스터 간접 주소 지정 방식 (register indirect addressing mode)

감사합니다

