

-Part1-

제5장 연산자란 무엇인가

(교재 89페이지 ~ 132페이지)

학습목차

5.1 변수와 변수의 시작 주소

-교재 91페이지 -

5.2 연산자의 종류

-교재 93페이지 -

5.3 비트 연산자

-교재 114페이지 -

5.1 변수와 변수의 시작 주소

-교재 91페이지 -

5.1 변수와 변수의 시작 주소 [5-1.c 실습]

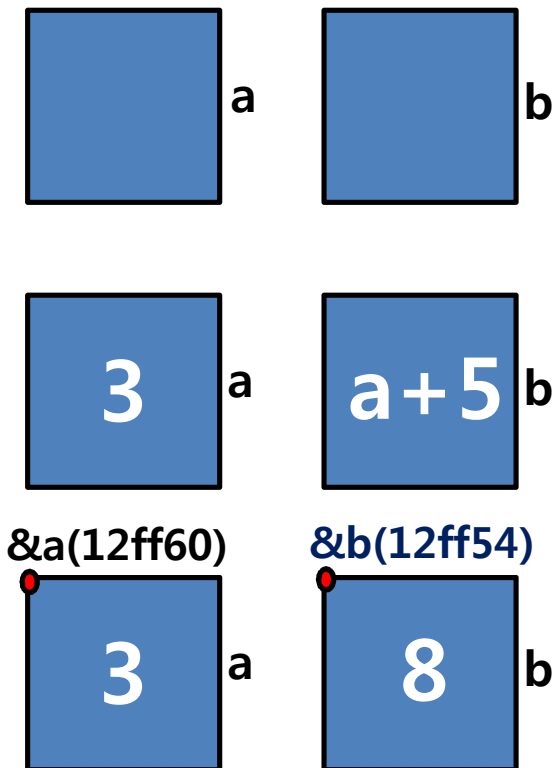
```
#include <stdio.h>
int main(void)
{
    int a;
    int b;

    a=3;
    b=a+5;

    printf("a의 값: %d \n", a);
    printf("b의 값: %d \n", b);

    printf("변수a의 주소: %x \n", &a);
    printf("변수b의 주소: %x \n", &b);

    return 0;
}
```



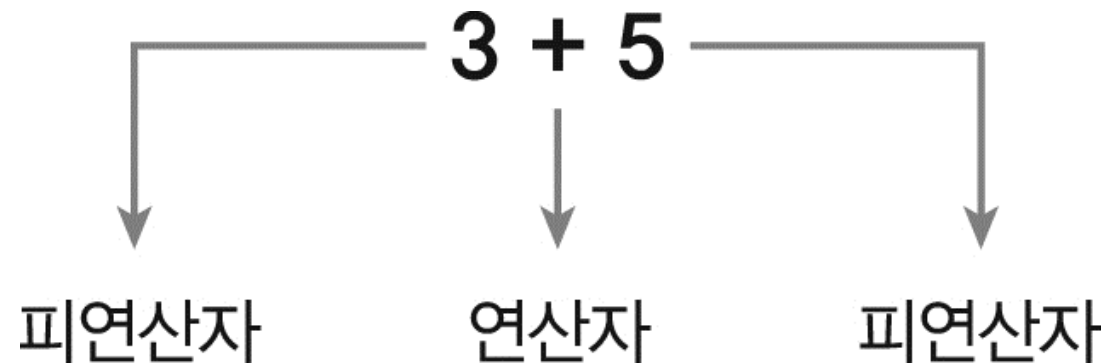
5.2 연산자의 종류

-교재 91페이지 -

5.2 연산자의 종류 (1/23)

▶ 연산자와 피연산자

- ✓ 연산자(Operator): 연산을 수행하는 기호
- ✓ 피연산자(Operand) : 연산에 참여하는 변수나 상수 값



5.2 연산자의 종류 (2/23)

▶ 연산자의 종류

분류	연산자
① 대입 연산자	=
② 산술 연산자	+, -, *, /, %
③ 복합 대입 연산자	+=, -=, *=, /=, %=
④ 증감연산자	++, --
⑤ 관계 연산자	>, <, ==, !=, >=, <=
⑥ 논리 연산자	&&, , !
⑦ 조건 연산자	? :
⑧ 비트 논리 연산자	&, !, ^, ~
⑨ 비트 이동 연산자	>>, <<

5.2 연산자의 종류 (3/23)

① 대입 연산자(=)

- ✓ 데이터를 저장하는 연산자
- ✓ 오른쪽에 있는 값을 왼쪽에 있는 변수 i에 대입(저장)하라는 의미

$i = 3$  i에 3을 저장한다는 의미

5.2 연산자의 종류 (4/23)---[5-2.c 실습]

```
#include <stdio.h>
int main(void)
{
    int i=0, j=0, k=0;

    printf("i = %d, j = %d, k = %d \n", i, j, k);

    i = 1;
    j = 5;
    k = 7;

    printf("i = %d, j = %d, k = %d\n", i, j, k);

    return 0;
}
```

5.2 연산자의 종류 (5/23)

② 산술 연산자

✓ 덧셈(+), 뺄셈(-), 곱셈(*), 나눗셈(/), 나머지(%)

산술 연산자	예	설명
+(덧셈 연산자)	$a = 6 + 2$	피연산자 6과 피연산자 2의 덧셈 연산
-(뺄셈 연산자)	$a = 6 - 2$	피연산자 6과 피연산자 2의 뺄셈 연산
*(곱하기 연산자)	$a = 6 * 2$	피연산자 6과 피연산자 2의 곱셈 연산
/(나누기 연산자)	$a = 6 / 2$	피연산자 6과 피연산자 2의 나눗셈 연산
%(나머지 연산자)	$a = 6 \% 2$	피연산자 6과 피연산자 2를 나눈 나머지 연산

5.2 연산자의 종류 (6/23)---[5-3.c 실습]

```
#include <stdio.h>
int main(void)
{
    int a, b;
    a = 6;
    b = 2;

    printf("  덧셈 연산 결과: %d \\\n", a+b);
    printf("  뺄셈 연산 결과: %d \\\n", a-b);
    printf("  곱셈 연산 결과: %d \\\n", a*b);
    printf("나누기 연산 결과: %d \\\n", a/b);
    printf("나머지 연산 결과: %d \\\n", a%b);

    return 0;
}
```

5.2 연산자의 종류 (7/23)---[5-4.c 실습]

```
#include <stdio.h>
int main(void)
{
    int num1, num2;

    num1 = 10/3;           // '몫' 출력
    num2 = 10%3;           // '나머지' 출력

    printf("    몫: %d \\\n", num1);
    printf("나머지: %d \\\n", num2);

    return 0;
}
```

5.2 연산자의 종류 (8/23)

③ 복합 대입 연산자

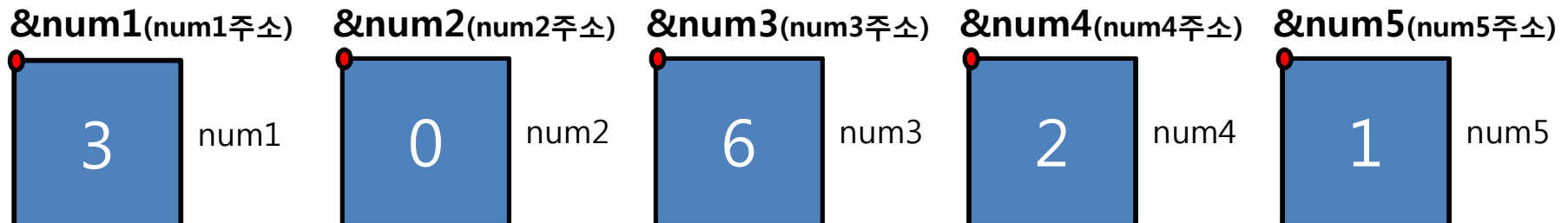
✓ 산술 연산자와 대입 연산자를 하나로 나타내는 기호

복합 대입 연산자	같은 표현	설명
$a = a + b$	$a += b$	$a + b$ 를 먼저 수행한 후에 a 에 값을 저장
$a = a - b$	$a -= b$	$a - b$ 를 먼저 수행한 후에 a 에 값을 저장
$a = a * b$	$a *= b$	$a * b$ 를 먼저 수행한 후에 a 에 값을 저장
$a = a / b$	$a /= b$	a / b 를 먼저 수행한 후에 a 에 값을 저장
$a = a \% b$	$a \% = b$	$a \% b$ 를 먼저 수행한 후에 a 에 값을 저장

5.2 연산자의 종류 (9/23)---[5-5.c 실습]

```
#include <stdio.h>
int main(void)
{
    int num1=1, num2=2, num3=3, num4=4, num5=5;
    num1 = num1 + num2;          // num1 += num2;
    num2 = num2 - 2;             // num2 -= 2;
    num3 = num3 * 2;             // num3 *= 2;
    num4 = num4 / 2;             // num4 /= 2;
    num5 = num5 % 2;             // num5 %= 2;
    printf("%d, %d, %d, %d, %d\n", num1, num2, num3, num4, num5);

    return 0;
}
```



5.2 연산자의 종류 (10/23)

④ 증감 연산자

✓ ++, -- 기호를 이용하는 연산자 (1증가 또는 1감소 시키는 연산자)

증감 연산자	설명
++a	선 증가, 후 연산 (먼저 증가하고 그 다음 연산)
a++	선 연산, 후 증가 (먼저 연산하고 그 다음 증가)
--a	선 감소, 후 연산 (먼저 감소하고 그 다음 연산)
a--	선 연산, 후 감소 (먼저 연산하고 그 다음 감소)

5.2 연산자의 종류 (11/23)---[5-6.c 실습]

```
#include <stdio.h>
int main(void)
{
    int num1=10;
    printf("%d \n", num1); // 결과는 10

    num1++;                // num1 = num1 + 1;
    printf("%d \n", num1); // 결과는 11

    ++num1;                // num1 = num1 + 1;
    printf("%d \n", num1); // 결과는 12

    --num1;                // num1 = num1 - 1;
    printf("%d \n", num1); // 결과는 11

    num1--;                // num1 = num1 - 1;
    printf("%d \n", num1); // 결과는 10
    return 0;
}
```


5.2 연산자의 종류 (12/23)---[5-7.c 실습]

```
#include <stdio.h>
int main(void)
{
    int num1=10, num2=10;
    int a, b;

    a = ++num1;           // 전위 방식, 선 증가 후 연산
    printf("%d, %d \n", a, num1); // 결과는 11, 11

    b = num2++;           // 후위 방식, 선 연산 후 증가
    printf("%d, %d \n", b, num2); // 결과는 10, 11

    return 0;
}
```

5.2 연산자의 종류 (13/23)---[5-8.c 실습]

```
#include <stdio.h>
int main(void)
{
    int num1=10, num2=10;
    int a, b;

    num1 = num1 + 1;           // 선 증가
    a = num1;                  // 후 연산
    printf("%d, %d \n", a, num1); // 결과는 11, 11

    b = num2;                  // 선 연산
    num2 = num2 + 1;           // 후 증가
    printf("%d, %d \n", b, num2); // 결과는 10, 11

    return 0;
}
```

5.2 연산자의 종류 (14/23)---[5-9.c 실습]

```
#include <stdio.h>
int main(void)
{
    int num1=10, num2=10;

    printf("%d \n", ++num1);    // 결과는 11
    printf("%d \n", num1);      // 결과는 11

    printf("%d \n", num2++);    // 결과는 10
    printf("%d \n", num2);      // 결과는 11

    return 0;
}
```

5.2 연산자의 종류 (15/23)

⑤ 관계 연산자

✓ 관계를 비교하여 참(True)과 거짓(False)으로 결론짓는 연산자

관계 연산자	예	설명	결과
>	$a > b$	a가 b보다 클지를 비교	1(참), 0(거짓)
<	$a < b$	a가 b보다 작을지를 비교	1(참), 0(거짓)
>=	$a \geq b$	a가 b보다 크거나 같을지를 비교	1(참), 0(거짓)
<=	$a \leq b$	a가 b보다 작거나 같을지를 비교	1(참), 0(거짓)
==	$a == b$	a가 b보다 같을지를 비교	1(참), 0(거짓)
!=	$a != b$	a가 b보다 같지 않을지를 비교	1(참), 0(거짓)

5.2 연산자의 종류 (16/23)---[5-10.c 실습]

```
#include <stdio.h>
int main(void)
{
    int num1=2, num2=4;
    int result1, result2, result3, result4;

    result1 = (num1 > num2);
    result2 = (num1 <= num2);
    result3 = (num1 == num2);
    result4 = (num1 != num2);

    printf("result1에 저장된 값 %d \n", result1); // 0(거짓)
    printf("result2에 저장된 값 %d \n", result2); // 1(참)
    printf("result3에 저장된 값 %d \n", result3); // 0(거짓)
    printf("result4에 저장된 값 %d \n", result4); // 1(참)

    return 0;
}
```

5.2 연산자의 종류 (17/23)

⑥ 논리 연산자

- ✓ && : AND 연산자 (논리곱)
- ✓ || : OR 연산자 (논리합)
- ✓ ! : NOT 연산자 (논리 부정)

&&연산자

피연산자	연산자	피연산자	결과
0	&&	0	0(거짓)
0	&&	1	0(거짓)
1	&&	0	0(거짓)
1	&&	1	1(참)

5.2 연산자의 종류 (18/23)

|| 연산자

피연산자	연산자	피연산자	결과
0		0	0(거짓)
0		1	1(참)
1		0	1(참)
1		1	1(참)

! 연산자

연산자	피연산자	결과
!	0	1(참)
!	1	0(거짓)

5.2 연산자의 종류 (19/23)---[5-11.c 실습]

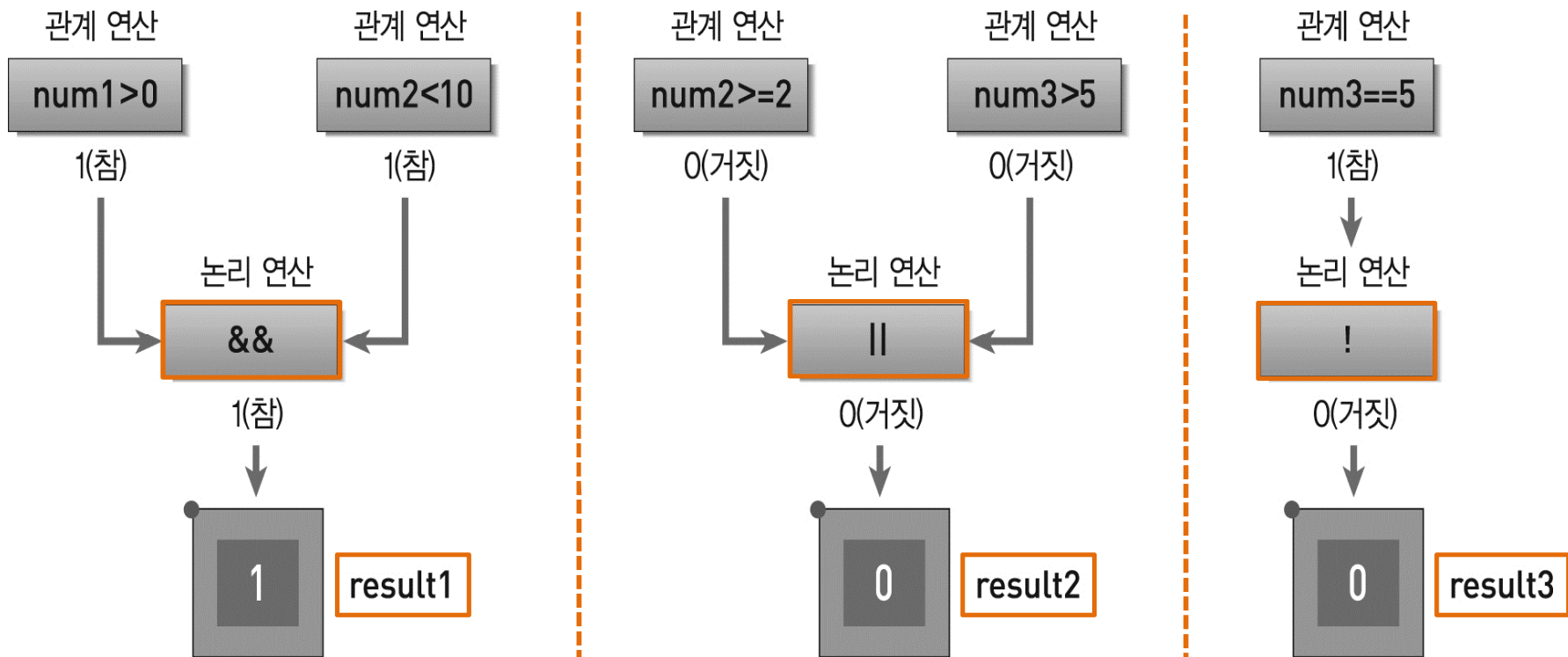
```
#include <stdio.h>
int main(void)
{
    int num1=2, num2=3, num3=5;
    int result1, result2, result3;

    result1 = (num1>0) && (num2<10);
    result2 = (num2<=2) || (num3>5);
    result3 = !num3;

    printf("result1에 저장된 값 %d \n", result1); // 1(참)
    printf("result2에 저장된 값 %d \n", result2); // 0(거짓)
    printf("result3에 저장된 값 %d \n", result3); // 0(거짓)

    return 0;
}
```


5.2 연산자의 종류 (20/23)---[5-11.c 분석]



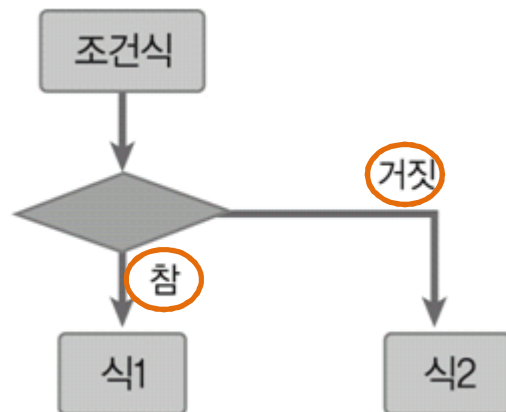
5.2 연산자의 종류 (21/23)

⑦ 조건 연산자

✓ '?' 와 ':' 로 이루어진 연산자



✓ 조건식 판단의 결과로 참과 거짓을 수행



5.2 연산자의 종류 (22/23)---[5-12.c 실습]

```
#include<stdio.h>
int main(void)
{
    int num1=2, num2=3;
    int result1;

    result1 = (num1>num2) ? num1 : num2;
    printf("result1에 저장된 값 %d \n",result1);

    return 0;
}
```

5.2 연산자의 종류 (23/23)---[5-12.c 분석]

7행

(num1>num2)가 거짓인 경우 'result1=num2'를 수행

result1 = (num1 > num2) ? num1 : num2 ;

(num1>num2)가 참인 경우 'result1=num1'을 수행

5.3 비트 연산자

-교재 114페이지 -

5.3 비트 연산자 (1/19)

▶ 비트와 바이트

✓ 비트(Bit) : 2진수 값 하나(0 또는 1)를 저장할 수 있는 최소 메모리 공간

1비트	2비트	3비트	...	n비트
$2^1 = 2$ 개	$2^2 = 4$ 개	$2^3 = 8$ 개 □	...	2^n 개

✓ 1바이트는 8비트



5.3 비트 연산자 (2/19)

▶ 2진수, 10진수, 16진수, 8진수

- ✓ 2진수: 0~1까지의 숫자를 사용 (컴퓨팅 세계 표현법)
- ✓ 10진수: 0~9까지의 숫자를 사용 (일상적인 표현법)
- ✓ 16진수: 0~9까지의 숫자를 사용하고,
9 이후부터 a, b, c, d, e, f 문자 사용 (컴퓨팅 세계 표현법)
- ✓ 8진수: 0~7까지의 숫자를 사용 (컴퓨팅 세계 표현법)

데이터 표현 방법 (2진수, 10진수, 16진수, 8진수)

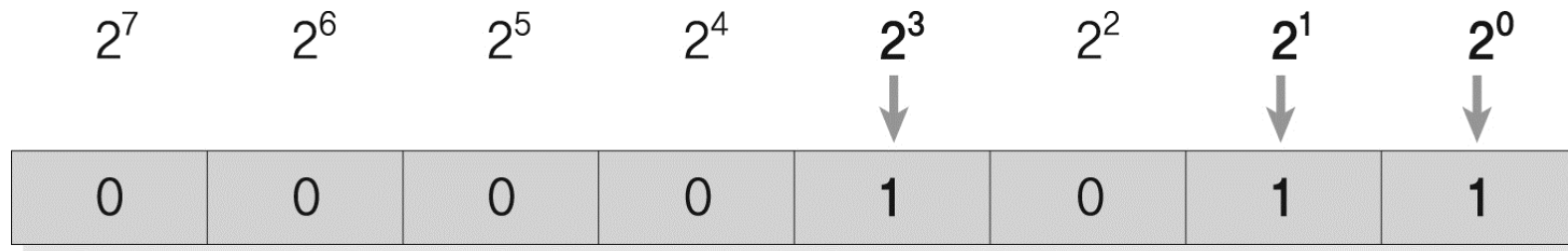
2진수	10진수	16진수	8진수
0000 0000	0	0	0
0000 0001	1	1	1
0000 0010	2	2	2
0000 0011	3	3	3
0000 0100	4	4	4
0000 0101	5	5	5
0000 0110	6	6	6
0000 0111	7	7	7
0000 1000	8	8	10

2진수	10진수	16진수	8진수
0000 1001	9	9	11
0000 1010	10	a	12
0000 1011	11	b	13
0000 1100	12	c	14
0000 1101	13	d	15
0000 1110	14	e	16
0000 1111	15	f	17
0001 0000	16	10	20
0001 0001	17	11	21

10 == 0xa == 012

5.3 비트 연산자 (3/19)

▶ 2진수를 10진수로 표현하는 방법

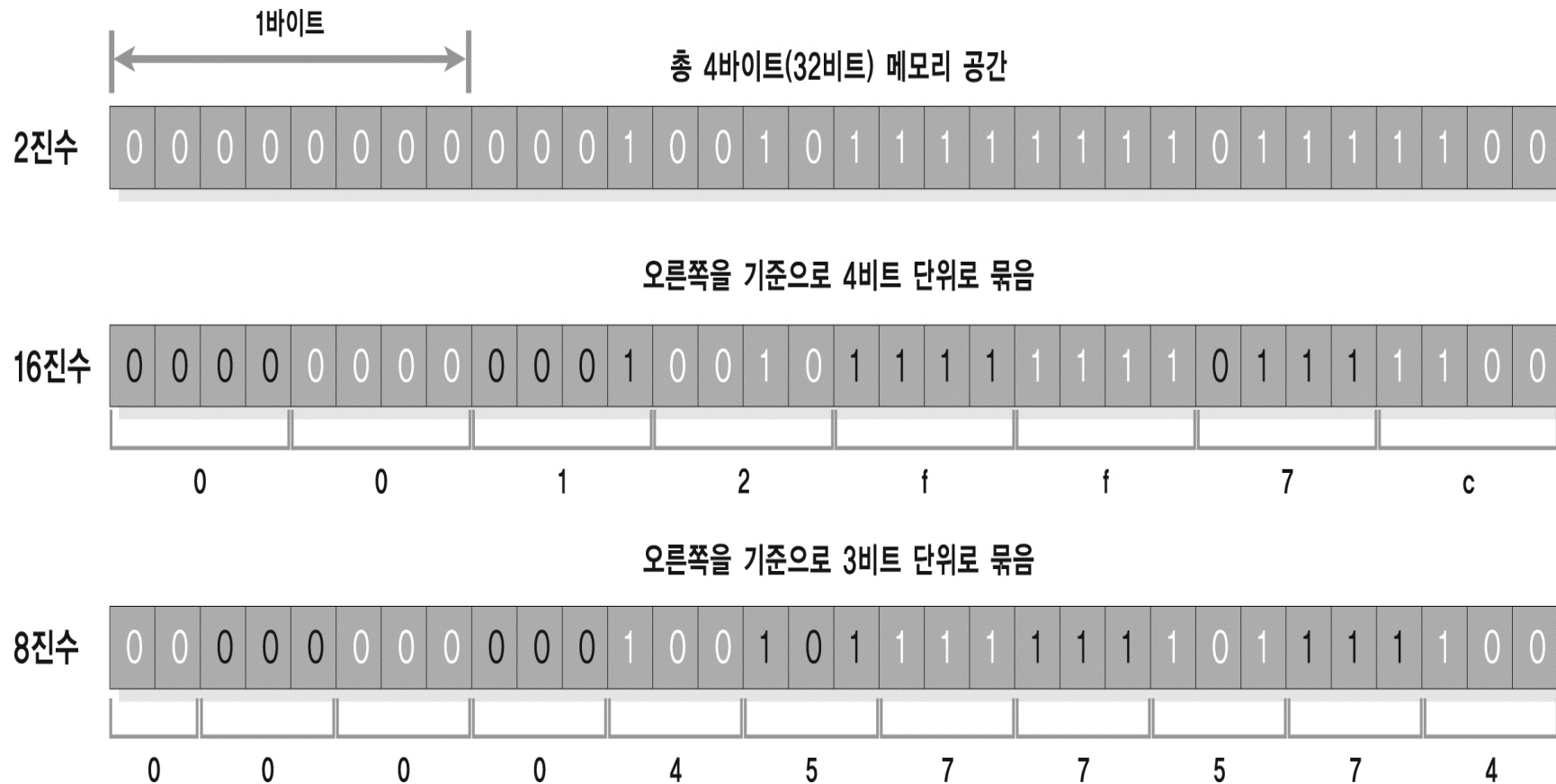


$$2^3 + 2^1 + 2^0 == \text{10진수로 11}$$

2진수	10진수	2진수	10진수	2진수	10진수
0000 0000		0000 0110		0000 1100	
0000 0001		0000 0111		0000 1101	
0000 0010		0000 1000		0000 1110	
0000 0011		0000 1001		0000 1111	
0000 0100		0000 1010		0001 0000	
0000 0101		0000 1011		0001 0001	

5.3 비트 연산자 (4/19)

▶ 2진수를 16진수와 8진수로 표현하는 방법



5.3 비트 연산자 (5/19)

⑧ 비트 연산자

✓ 데이터를 비트 단위로 처리하는 연산자

비트 연산자	연산식	설명
&	$a \& b$	비트 단위 AND 연산
	$a b$	비트 단위 OR 연산
^	$a \wedge b$	비트 단위 XOR 연산
~	$\sim a$	비트 단위 NOT 연산
<<	$a << 3$	왼쪽으로 세 칸 이동
>>	$a >> 1$	오른쪽으로 한 칸 이동

5.3 비트 연산자 (6/19)

▶ & 연산자

- ✓ 비트 단위로 AND 연산을 수행
- ✓ 두 개의 비트가 모두 1일 때 1을 반환

피연산자	연산자	피연산자	결과
0	&	0	0
0	&	1	0
1	&	0	0
1	&	1	1

5.3 비트 연산자 (7/19)---[5-13.c 실습]

```
#include <stdio.h>
int main(void)
{
    int num1=20, num2=16;
    int result1;

    result1 = num1 & num2;
    printf("비트단위 & 연산의 결과 %d \\\n", result1); // 결과는 16

    return 0;
}
```

&	0000 0000 0000 0000 0000 0000 0001 0100 (20)							
	0000 0000 0000 0000 0000 0000 0001 0000 (16)							
	<hr/>							
	0000 0000		0000 0000		0000 0000		0001 0000	(16)

5.3 비트 연산자 (8/19)

▶ | 연산자

- ✓ 비트 단위로 OR 연산을 수행
- ✓ 두 개의 비트 중의 하나가 1일 때 1을 반환

피연산자	연산자	피연산자	결과
0		0	0
0		1	1
1		0	1
1		1	1

5.3 비트 연산자 (9/19)---[5-14.c 실습]

```
#include <stdio.h>
int main(void)
{
    int num1=20, num2=16;
    int result1;

    result1 = num1 | num2;
    printf("비트단위 | 연산의 결과 %d \\\n",result1);

    return 0;
}
```

	0000	0000	0000	0000	0000	0000	0001	0100	(20)
	0000	0000	0000	0000	0000	0000	0001	0000	(16)
	0000	0000	0000	0000	0000	0000	0001	0100	(20)

5.3 비트 연산자 (10/19)

▶ ^ 연산자

- ✓ 비트 단위로 XOR 연산을 수행
- ✓ 두 개의 비트가 서로 같지 않을 경우 1을 반환

피연산자	연산자	피연산자	결과
0	^	0	0
0	^	1	1
1	^	0	1
1	^	1	0

5.3 비트 연산자 (11/19)---[5-15.c 실습]

```
#include <stdio.h>
int main(void)
{
    int num1=20, num2=16;
    int result1;

    result1 = num1 ^ num2;
    printf("비트단위 ^ 연산의 결과 %d \n",result1);

    return 0;
}
```

^	0000	0000	0000	0000	0000	0000	0001	0100	(20)
	0000	0000	0000	0000	0000	0000	0001	0000	(16)
<hr/>									
	0000	0000	0000	0000	0000	0000	0000	0100	(4)

5.3 비트 연산자 (12/19)

▶ ~ 연산자

- ✓ 비트 단위로 NOT 연산을 수행
- ✓ 보수 연산으로 비트를 반전 시킴

연산자	피연산자	결과
~	0	1
~	1	0

5.3 비트 연산자 (13/19)---[5-16.c 실습]

```
#include <stdio.h>
int main(void)
{
    int num1=20;
    int result1;

    result1 = ~num1;
    printf("비트단위 ~ 연산의 결과 %d \n",result1); // 결과는 -21

    return 0;
}
```

~		0000	0000	0000	0000	0000	0000	0001	0100	(20)
		<hr/>								
		1111	1111	1111	1111	1111	1111	1110	1011	(-21)

5.3 비트 연산자 (14/19)---[5-16.c 분석]

~

0000	0000	0000	0000	0000	0000	0001	0100	(20)
1111	1111	1111	1111	1111	1111	1110	1011	(-21)



1의 보수

1의 더함

0000	0000	0000	0000	0000	0000	0001	0101	(21)
1111	1111	1111	1111	1111	1111	1110	1010	
0000	0000	0000	0000	0000	0000	0000	0001	
1111	1111	1111	1111	1111	1111	1110	1011	(-21)

5.3 비트 연산자 (15/19)

▶ ⑨ 비트 이동 연산자

- ✓ <<연산자 (왼쪽 시프트 연산자)
 - 비트를 왼쪽으로 이동시키는 연산자
- ✓ >>연산자 (오른쪽 시프트 연산자)
 - 비트를 오른쪽으로 이동시키는 연산자

5.3 비트 연산자 (16/19)---[5-17.c 실습]

```
#include <stdio.h>
int main(void)
{
    int num1=10;
    int result1;

    result1 = num1 << 2;
    printf("비트단위 << 연산의 결과 %d \n", result1);

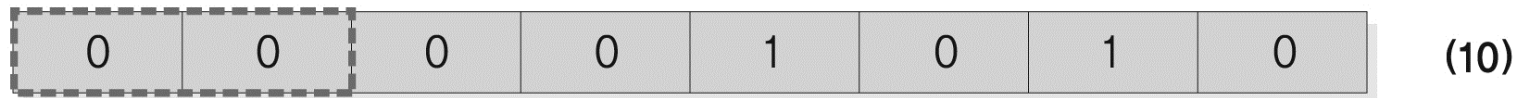
    return 0;
}
```

5.3 비트 연산자 (17/19)---[5-17.c 분석]

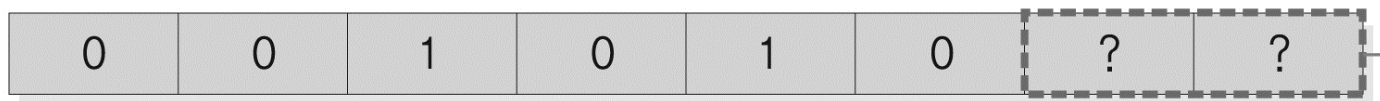
```
result1 = num1 << 2;
```



Step 1 : 왼쪽 2비트 메모리 공간의 데이터가 사라짐



Step 2 : 오른쪽에 2비트 빈 메모리 공간이 생김



Step 3 : 빈 메모리 공간을 0으로 채움



왼쪽으로 1비트씩 이동할 때 마다 2씩 곱해진다

5.3 비트 연산자 (18/19)---[5-18.c 실습]

```
#include <stdio.h>
int main(void)
{
    int num1=10;
    int num2=-10;

    int result1;
    int result2;

    result1 = num1 >> 1;
    result2 = num2 >> 1;

    printf("비트단위 >> 연산의 결과 %d \n", result1);
    printf("비트단위 >> 연산의 결과 %d \n", result2);

    return 0;
}
```


5.3 비트 연산자 (19/19)---[5-18.c 분석]

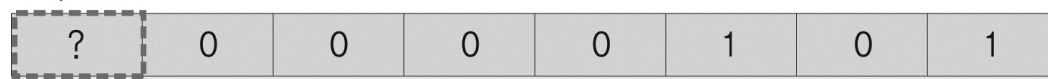
```
result1 = num1 >> 1;
```



Step 1 : 오른쪽 1비트 메모리 공간의 데이터가 사라짐 ←



Step 2 : 왼쪽에 1비트 빈 메모리 공간이 생김



Step 3 : 빈 메모리 공간을 0으로 채움 ←



오른쪽으로 1비트씩 이동할 때 마다 2씩 나누어진다



```
result2 = num2 >> 1;
```

Step 1 : 오른쪽 1비트 메모리 공간의 데이터가 사라짐 ←



Step 2 : 왼쪽에 1비트 빈 메모리 공간이 생김



Step 3 : 빈 메모리 공간을 1로 채움 ←



5.4 연산자 우선순위

-교재 127페이지 -

5.4 연산자 우선순위

우선 순위	연산자	연산 방향	우선 순위	연산자	연산방향
1	() [] -> .	왼쪽에서 오른쪽	8	&	왼쪽에서 오른쪽
2	~ ~ ++ -- + - * &	오른쪽에서 왼쪽	9	^	왼쪽에서 오른쪽
3	* / %	왼쪽에서 오른쪽	10		왼쪽에서 오른쪽
4	+ -	왼쪽에서 오른쪽	11	&&	왼쪽에서 오른쪽
5	<< >>	왼쪽에서 오른쪽	12		왼쪽에서 오른쪽
6	< <= > >=	왼쪽에서 오른쪽	13	?:	오른쪽에서 왼쪽
7	== !=	왼쪽에서 오른쪽	14	= += -= *= /= %= &= ^= != <<= >>=	오른쪽에서 왼쪽
			15	,	왼쪽에서 오른쪽

공부한 내용 떠올리기

▶ 변수와 변수의 시작 주소

▶ 다양한 연산자

- ✓ 대입 연산자, 산술 연산자, 복합 대입 연산자, 증감 연산자
- ✓ 관계 연산자, 논리 연산자, 조건 연산자

▶ 비트, 바이트, 2진수, 10진수, 16진수

▶ 비트 연산자

- ✓ & 연산자, | 연산자, ^ 연산자, ~ 연산자, << 연산자, >> 연산자

▶ 연산자 우선순위

소중한 것을 잃게 하는 분노(출처: 사랑과 지혜의 탈무드)

