C 언어 기초 – 메모리관리 방식

작성자 : 홍효상
이메일 : hyomee@naver.com
소스 : https://github.com/hyomee/c_basic

Content

I. C 언어 기초 메모리관리 방식

1. 메모리 종류
2. 주소 지정 방식

1. 메모리 종류

“ Heap를 사용 하는 메모리는 개발자가 관리 하여야 하므로 할당 및 해제를 해야 함“

메모리 종류

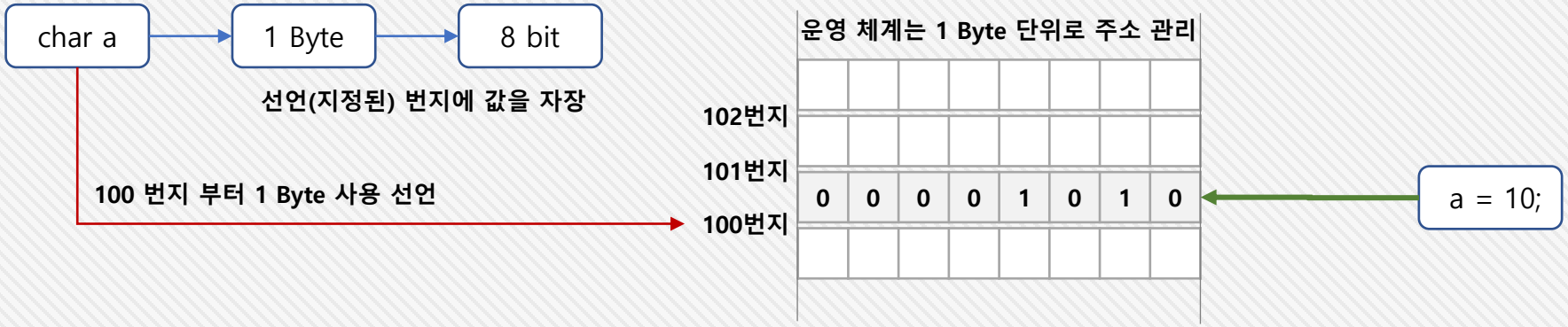
분 류		특 징
Stack		자동 변수 지역 변수인 변수가 사용하는 메모리 영역 / 크기가 작고 관리(할당 및 반환)가 자동으로 이루어짐
Heap		동적 할당할 수 있는 자유 메모리 영역 개발자 스스로 관리(수동) - 제일 큰 영역 할당 및 해제를 신경 써야 함
PE image (실행 영역	Text section	C언어의 소스코드가 번역된 기계어가 저장된 메모리 영역 기본적으로 읽기전용 메모리
	Data section(Read only)	상수 형태를 기술하는 문자열(예: "Hello")이 저장된 메모리 영역
	Data section(Read/Write)	정적변수나 전역변수들이 사용하는 메모리 영역 별도로 초기화하지 않아도 0으로 초기화 관리는 자동이기 때문에 Heap 영역 메모리처럼 할당 및 해제를 신경 쓸 필요 없음

1. 메모리 지정 방식

" 변수는 컴파일 이후 메모리 주소 "

메모리 지정 방식

- 운용 체제는 메모리 주소를 1Byte 단위로 관리
- 프로그래머는 메모리를 사용 하기 위해서는 사용 할 주소를 표기 해 주어야 한다.



2 진수	16 진수	2 진수	16 진수	2 진수	16 진수	2 진수	16 진수
0000	0	0100	4	1000	8	1100	C (12)
0001	1	0101	5	1001	9	1101	D (13)
0010	2	0110	6	1010	A (10)	1110	E (14)
0011	3	0111	7	1011	B (11)	1111	F (15)

- 직접 주소 방식과 간접 주소 방식 프로그래머는 사용 할 메모리를 지정 할 수 있음

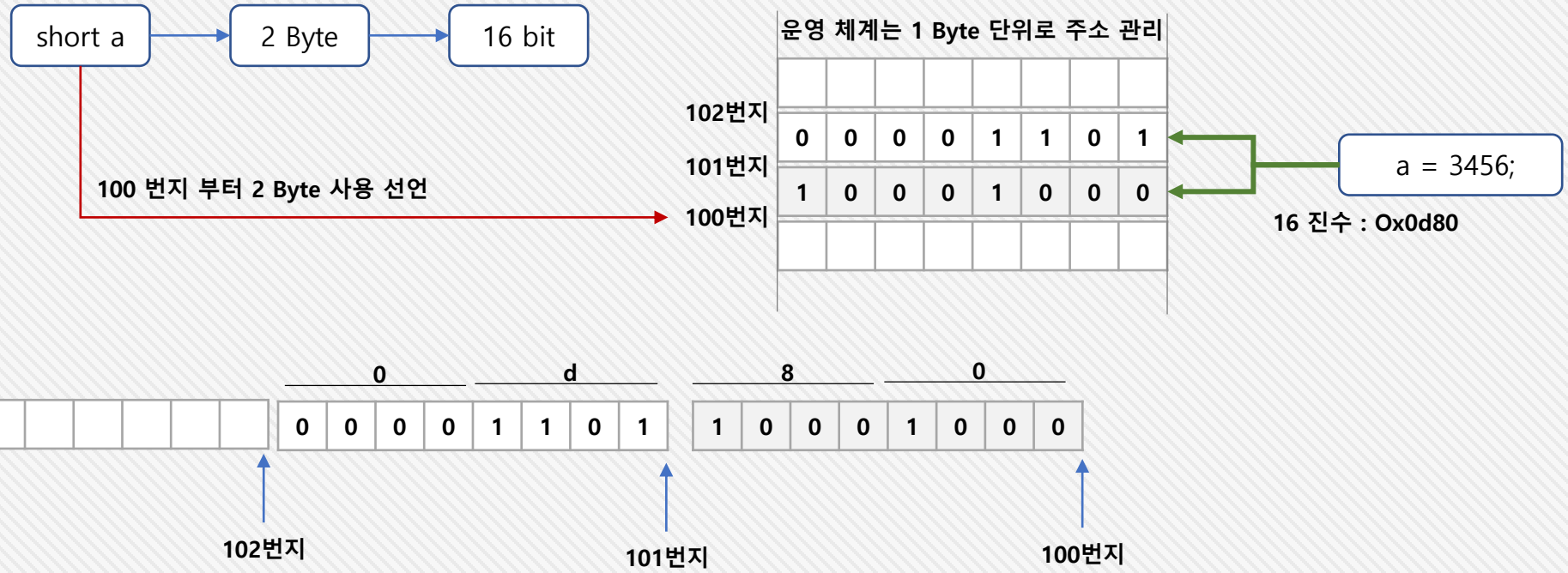
2. 직접 주소 지정 방식

" 변수 사용은 C 언어에서 직접 주소 지정 방식 "

직접 주소 지정 방식

- 변수를 사용 하는 문법을 의미함 .

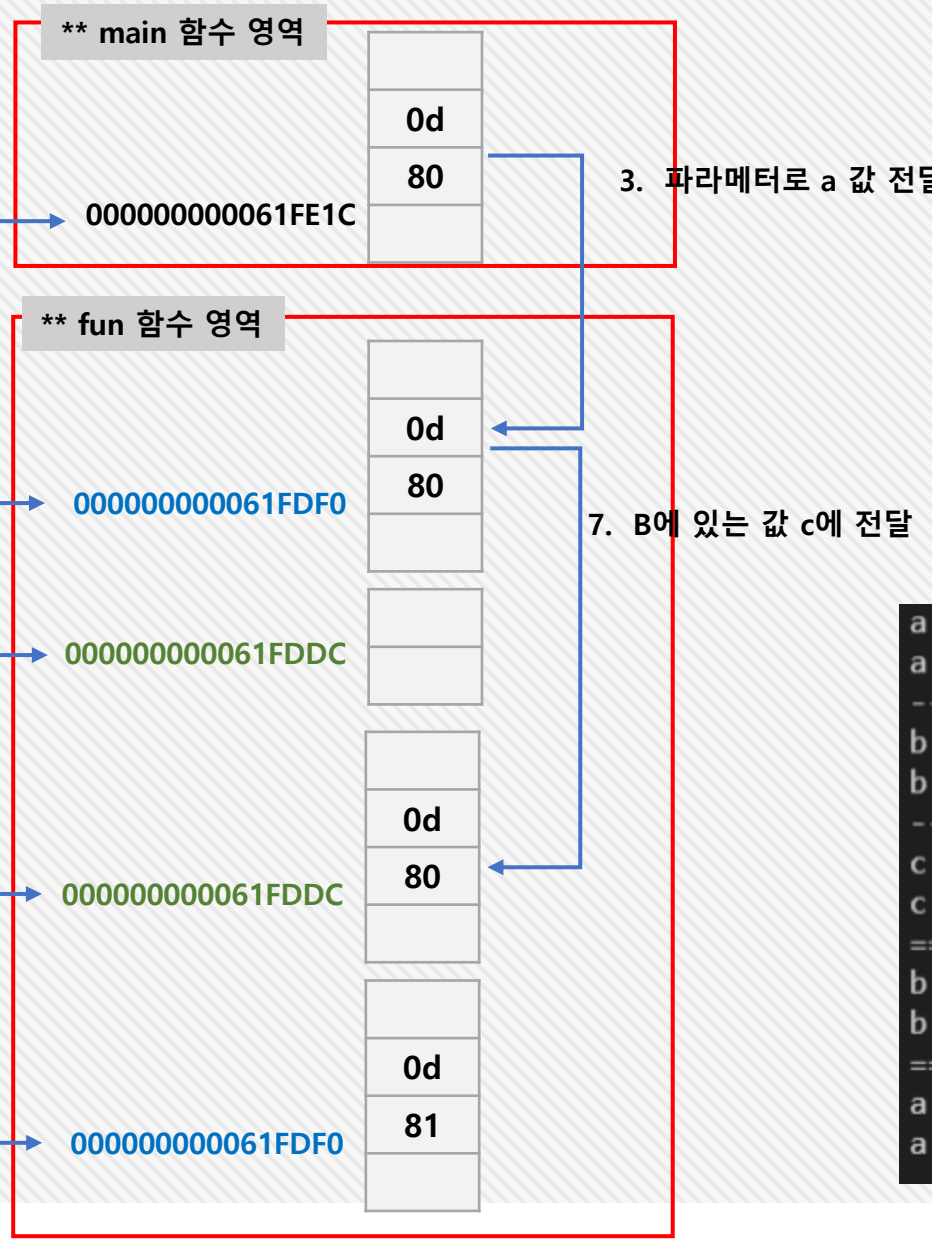
```c
#include <stdio.h>

int main() {
    short a = 3456;

    printf("a    : %#x₩n",  a);
    printf("size : %d",  sizeof(a));
}
```



2. 직접 주소 지정 방식


```c
#include <stdio.h>

void fun(b);

int main() {
    int a = 3456;

    printf("a          : 0x%X₩n", a);
    printf("a address : %p₩n",  &a);
    fun(a);
    printf("== after fun ==₩n");
    printf("a          : 0x%X₩n", a);
    printf("a address : %p₩n",  &a);
}

void fun(int b) {
    int c;
    c = b;
    printf("-------------₩n");
    printf("b          : 0x%X₩n", b);
    printf("b address : %p₩n",  &b);
    printf("-------------₩n");
    printf("c          : 0x%X₩n", c);
    printf("c address : %p₩n",  &c);
    b += 1;
    printf("== b operator =₩n");
    printf("b          : 0x%X₩n", b);
    printf("b address : %p₩n",  &b);
}
```

1. 정수형 a 선언 및 초기화
2. 함수 fun에 파라메터 전송
3. 파라메터로 a 값 전달
4. 파라메터 변수 b에 a 값 저장
5. 정수형 변수 c 선언
6. b에 있는 값 c에 저장
7. B에 있는 값 c에 전달
8. b에 있는 갑 증감 계산




```
a          : 0xD80
a address : 000000000061FE1C
-------------
b          : 0xD80
b address : 000000000061FDF0
-------------
c          : 0xD80
c address : 000000000061FDDC
== b operator =
b          : 0xD81
b address : 000000000061FDF0
== after fun ==
a          : 0xD80
a address : 000000000061FE1C
```

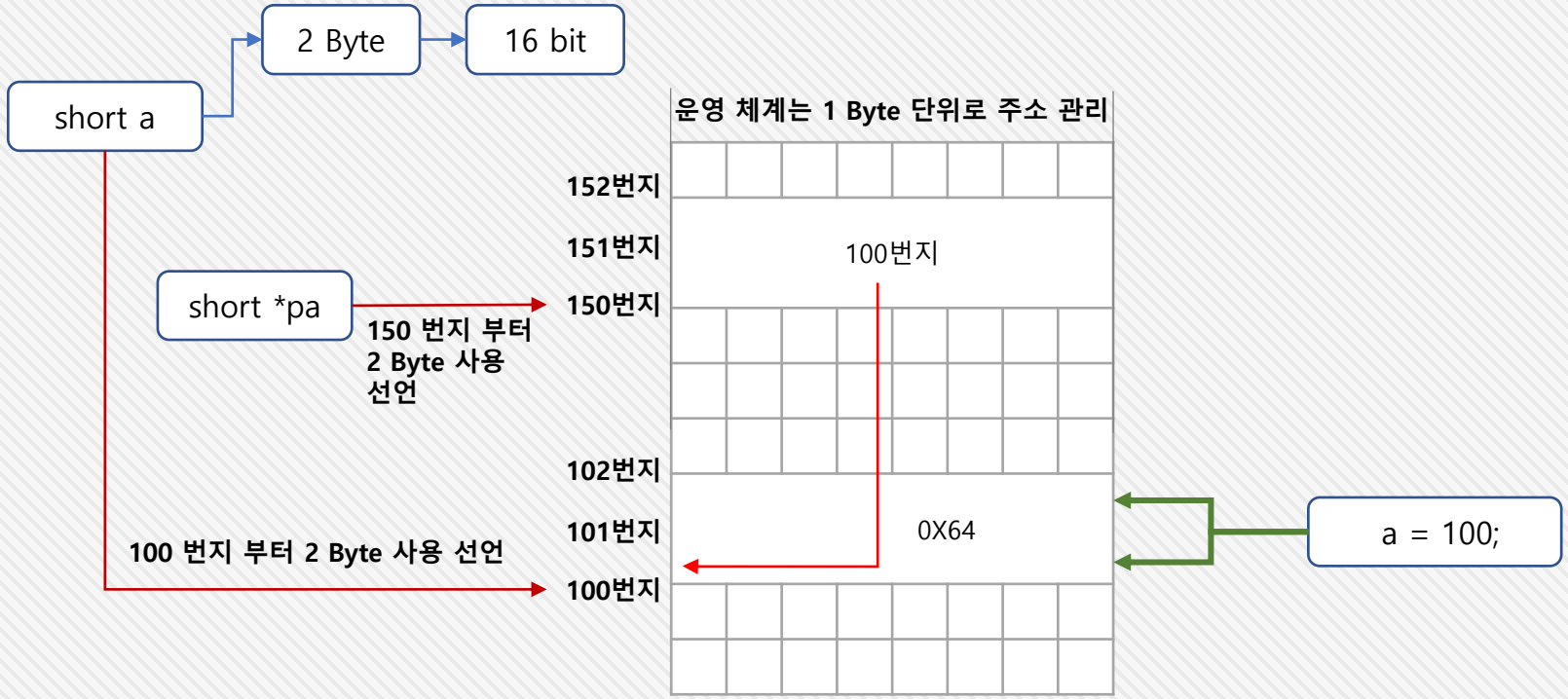
3. 간접 주소 지정 방식

" 참조 주소를 통해서 값은 관리 "

간접 주소 지정 방식

- 변수에 주소를 지정 하여 지정된 주소의 값을 관리.

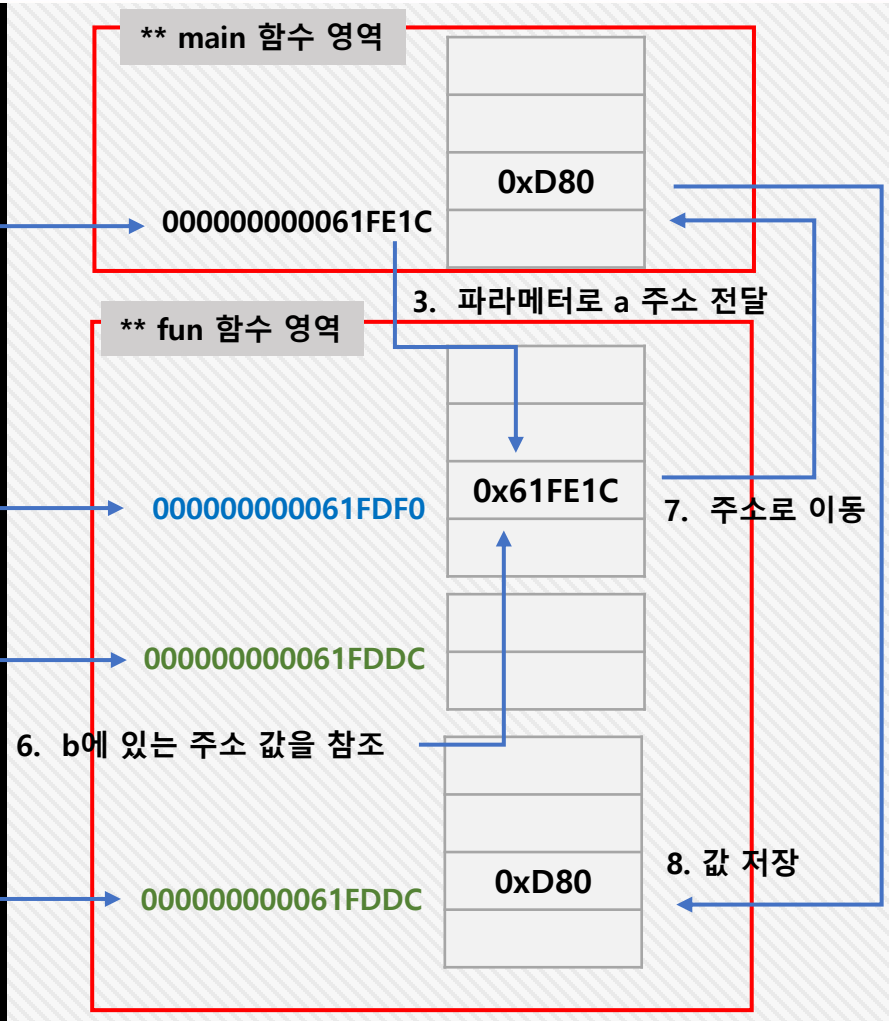
```
#include <stdio.h>

void main() {
    short a;
    short *pa;

    pa = &a;
    a = 100;

    printf("a data      : 0x%X₩n", a );
    printf("a address   : %p₩n",   &a);
    printf("pa data     : 0x%X₩n", pa );
    printf("pa address : %p₩n",   &pa);
    *pa += 1;
    printf("pa data     : 0x%X₩n", pa );
    printf("pa address : %p₩n",   &pa);
    printf("a data      : 0x%X₩n", a );
}
```



2. 간접 주소 지정 방식


```c
#include <stdio.h>

void fun(b);

int main() {
    int a = 3456;

    printf("a         : 0x%X\n", a);
    printf("a address : %p\n",  &a);
    fun(&a);
    printf("== after fun ==\n");
    printf("a         : 0x%x\n", a);
    printf("a address : %p\n",  &a);
}

void fun(int *b) {
    int c;
    c = *b;
    printf("-------------\n");
    printf("*b         : 0x%X\n", b);
    printf("*b address : %p\n",  &b);
    printf("-------------\n");
    printf("c         : 0x%X\n", c);
    printf("c address : %p\n",  &c);
    *b += 1;
    printf("== b operator =\n");
    printf("*b         : 0x%X\n", b);
    printf("*b address : %p\n",  &b);
}
```

1. 정수형 a 선언 및 초기화
2. 함수 fun 호출
3. 파라메터로 a 주소 전달
4. 파라메터 포인터 변수 b에 a 주소 저장
5. 정수형 변수 c 선언
6. b에 있는 주소 값을 참조
7. 주소로 이동
8. 값 저장
9. 포인터 p에 저장 되어 있는 주소에 있는 값을 읽어서 +1 후 포인터 p에 저장된 주소를 찾아가서 값을 변경 함




```
a         : 0xD80
a address : 000000000061FE1C
-------------
*b         : 0x61FE1C
*b address : 000000000061FDF0
-------------
c         : 0xD80
c address : 000000000061FDDC
== b operator =
*b         : 0x61FE1C
*b address : 000000000061FDF0
== after fun ==
a         : 0xd81
a address : 000000000061FE1C
```