Embedded C-Programming: Pointer, Type Casting

Young-Keun Kim

mod .2023.08.19

Pointer

What are Pointers?

- A **pointer** is a variable whose value is the address of another variable
- i.e., direct address of the memory location.
- Pointers are the basis for data structures

(a) Define a pointer variable

(b) Assign the address of a variable to a pointer and

(c) Access the value at the address available in the pointer variable

```
int  *ptr;
ptr = &x;
int value = *ptr
```

```
int x =100;          // 변수의 선언
int *ptr = &x;       // 포인터의 선언
int **pptr = &ptr; // 포인터의 참조
```

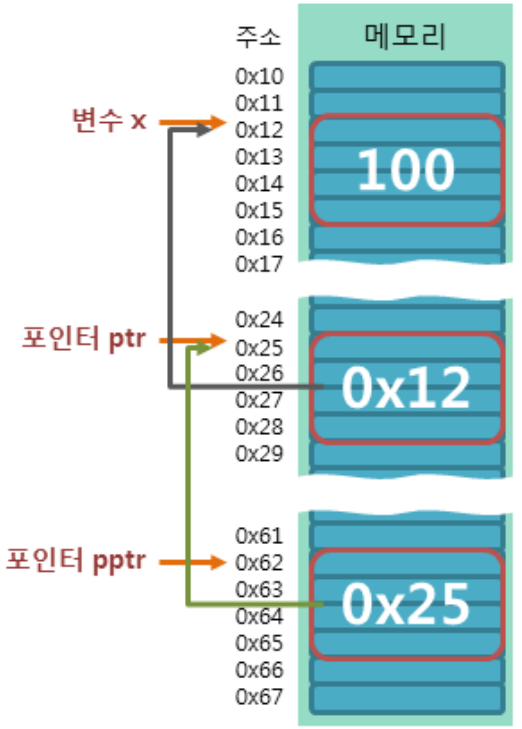
```
int main() {
      int  x = 100;

      // Pointer Define
      int  *ptr;

      // Pointer Assignment
      ptr = &x;// store address of var in pointer variable

      // Pointer Access
      printf("Value of *ptr: %d”, *ptr);

      return 0;
}
```



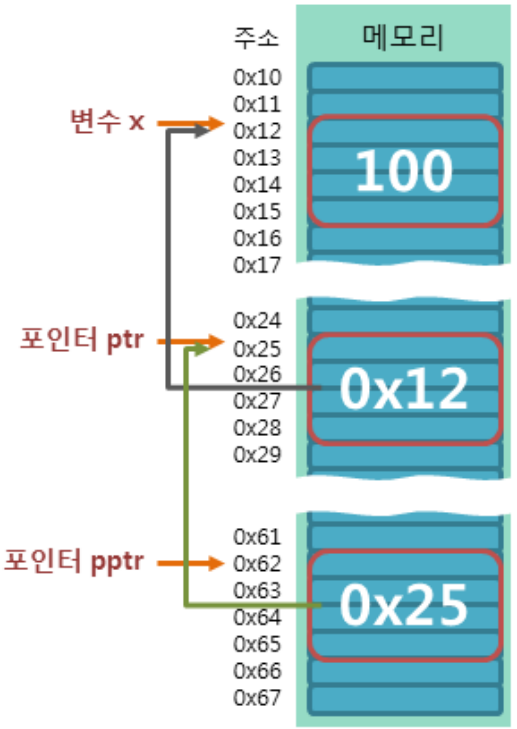
[Image from TCPschool]

Pointer

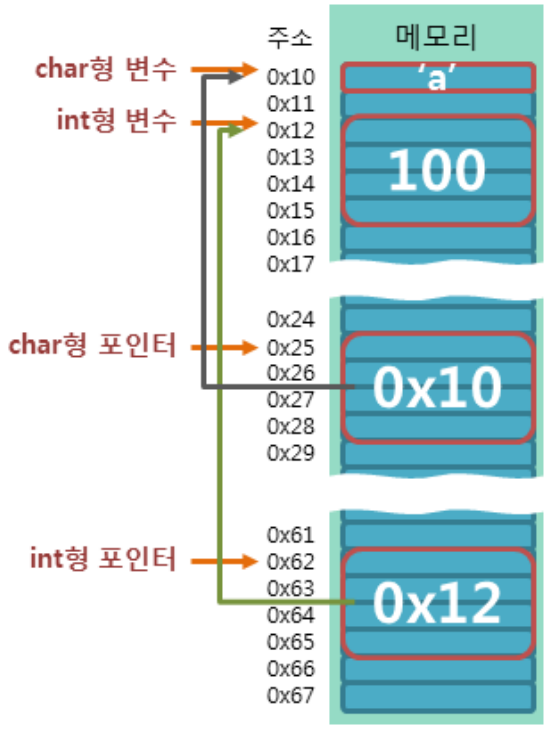
- 32-CPU: 1 word = 4 bytes (32bit)
  - Memory: byte units
  - Int : 4 byte
  - Pointer variable: 4 byte

```
int n = 100;  // 변수의 선언
int *ptr = &n; // 포인터의 선언
```


```
int x =100;        // 변수의 선언
int *ptr = &x;     // 포인터의 선언
int **pptr = &ptr; // 포인터의 참조
```




```
int x =100;        // 변수의 선언
char y='a';
int *ptr = &x;     // 포인터의 선언
int *pptr = &y;
```



Pointer

- **Examples:**

Example 1

```
uint32_t a, b;
int main(void) {
    a = 3; b = 4;
    swap(&a, &b);
}

void swap(uint32_t* a, uint32_t* b) {
    uint32_t t;
    t = *a;
    *a = *b;
    *b = t;
}
```

Example 2

```
char str1[13] = "ARM Assembly";
char str2[13];
strcpy(str2, str1);

void strcpy(char* dest, char* source) {
        while (*source) {
                *dest = *source;
                dest++;
                source++;
        }
*dest = *source;   // termination 0
}
```

Example 3

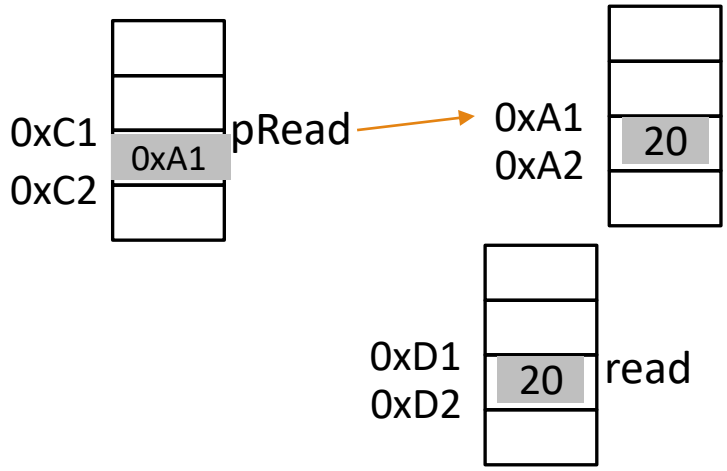
```
uint16_t const Prime[3] = { 1,2,3 };
uint16_t const* Pt;// const does NOT indicate pointer is fixed. Refers to const data
Pt = Prime;  //or  &Pt=Prime[0]
```

Type Casting of Pointer Variable

- **Examples in Firmware programming**
  - Since we are working with memory mapped I/O, we must use pointers to access data of I/O

(uint8_t*) casting: address of 0xA1
points 8-bit unsigned int data

```
uint8_t* pRead;
pRead = (uint8_t*)0xA1
//uint8_t *pRead= (uint8_t*) 0xA1;
uint8_t read = *pRead;  // read=5
```




```
// Casting memory address to a pointer
#define GPIOA ((GPIO_TypeDef *) 0x40020000)
…
void GPIO_Initialize(GPIO_TypeDef* Port);
…
GPIO_Initialize(GPIOA);
```

Type Casting of Pointer Variable

- **Examples in Firmware programming**
  - Since we are working with memory mapped I/O, we must use pointers to access data of I/O

```
#define  P1IN (* ((volatile uint8_t *) 0x40004C00))          // what does it mean?
```

Let address of Port 1 input data (P1IN) is 0x4000 4C00

data=0x4000 4C00; // does not read values @ that address
data=(*0x4000 4C00); // read the content @ that address BUT does not know 8bit?16bit?32bit?

→ Use typecast (uint8_t *) force conversion to 8-bit type pointer
→ *Volatile* because data value of port can be changed by beyond SW action (Interrupt etc)

data= (* ((volatile uint8_t *)0x4000 4C00))
→ get the data(uint8_t) pointed by that address

Thus,
#define P1IN (* ((volatile uint8_t *) 0x4000 4C00))

Exercise

Exercise: Pointer and Pointer Type Casting

Pointer_exercise

```
double y=2.5;
int *ptrY = &y;
```

- Print the address of variable ‘y’

- Print the value of pointer ‘ptrY ‘
- Print the address of pointer ‘ptrY ‘
- Print the size of pointer ‘ptrY ‘

- **Typecast pointer ‘ptrY’ to as (double *)