시스템 프로그래밍을 위한 C언어

struct 이용하여 구조적인 multiple bytes 패턴을 메모리에 할당

현대자동차 입문교육

박대진 교수

What is Structure ? (struct)

- A compound data type which combines variables with different data types.

Data Type

```
int
```

기 정의된 타입

```
int i;
```

Data Type 이름을 내가 정의할 수 있음

struct+이름을 합쳐서, 하나의 Type을 결정

```
struct tag_name {
    type1   member1;
    type2   member2;
    …
    typeN   memberN;
};
```

members of structure

User-defubed Data Type의 구조 (structure)를 단순히 정의만 한 상태.

```
struct student {
    unsigned int studentID;
    char name[MAX];
    unsigned int score;
    char grade;
};
```

Data Type Variable

```
struct student stds[N]
```

Be aware of struct

- Operations on structure
  - Comparison of two structures in equality is not permitted
    - Equality function for member-by-member should be used
  - &(address of) is permitted for structure and a member of the struct
    - Pointer to the structure and pointer to the component
- Members in struct
  - Structure may not contain instances of themselves, but may contai
    pointers to instances of themselves.

```
struct S {int a; struct S next; };        /* illegal
struct S {int a; struct S* next; };        /* ok
```

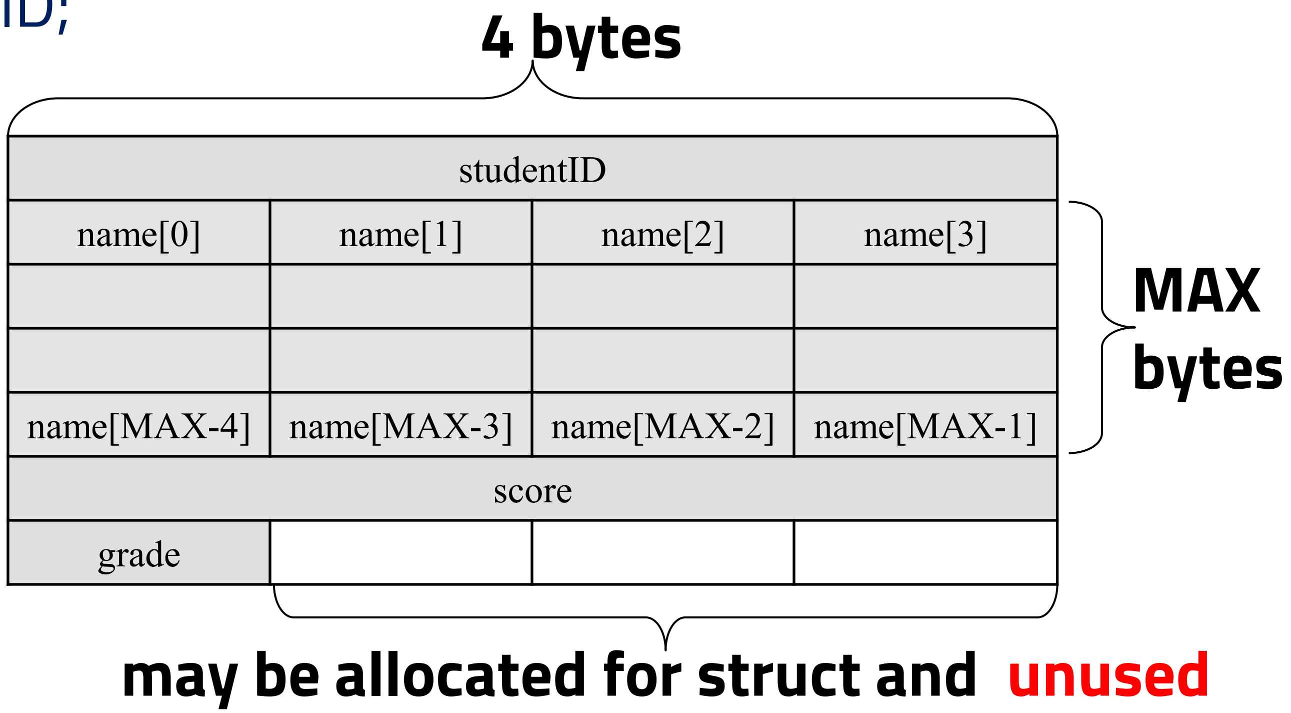
  - Names within a single struct must be distinct.

```
int x;
struct A {int x; double y;} y;   /* the seq of def. is ok */
```

Memory Allocation of struct

◆Consecutive memory space for members

```
struct student{
    unsigned int studentID;
    char name[MAX];
    unsigned int score;
    char grade;
} mydata ;
```



★ size of struct ≥ sum of size of members → address alignment by compiler

struct member initialization

int i = 1 와 비슷하게 생각하면 된다

1

User-defined Data Type의 구조 (structure)를 정의

```
struct student{
    unsigned int studentID;
    char name[MAX];
    unsigned int score;
    char grade;
} ;
```

2 **변수 선언, 데이터 채우기**

```
struct student mydata = { 20090001, “KIM J”, 100, ‘A’ };
```

=

```
struct student mydata;
mydata = { 20090001, “KIM J”, 100, ‘A’ };
```

=

```
mydata.studentID = 20090001;
mydata.name = “KIM J”;
mydata.score = 100;
mydata.grade = ‘A’;
```

Accessing struct member variables

```
struct student{
   unsigned int studentID;
   char name[MAX];
   unsigned int score;
   char grade;
} ;
```

```
mystudent.studentID
mystudent.name
mystudent.score
mystudent.grade
```

```
yourstudent -> studentID
yourstudent -> name
yourstudent -> score
yourstudent -> grade
```

```
struct student mystudent
 = { 20090101, “KIM JH”, 100, ‘A’ };  // scalar var
```

```
struct student* yourstudent
= &mystudent;   // pointer var
```

Structured Data Allocation on Memory

```c
struct ADC_CONFIG {
    unsigned char CNFG1;
    unsigned char CNFG2;
    unsigned short MODE;
};
```

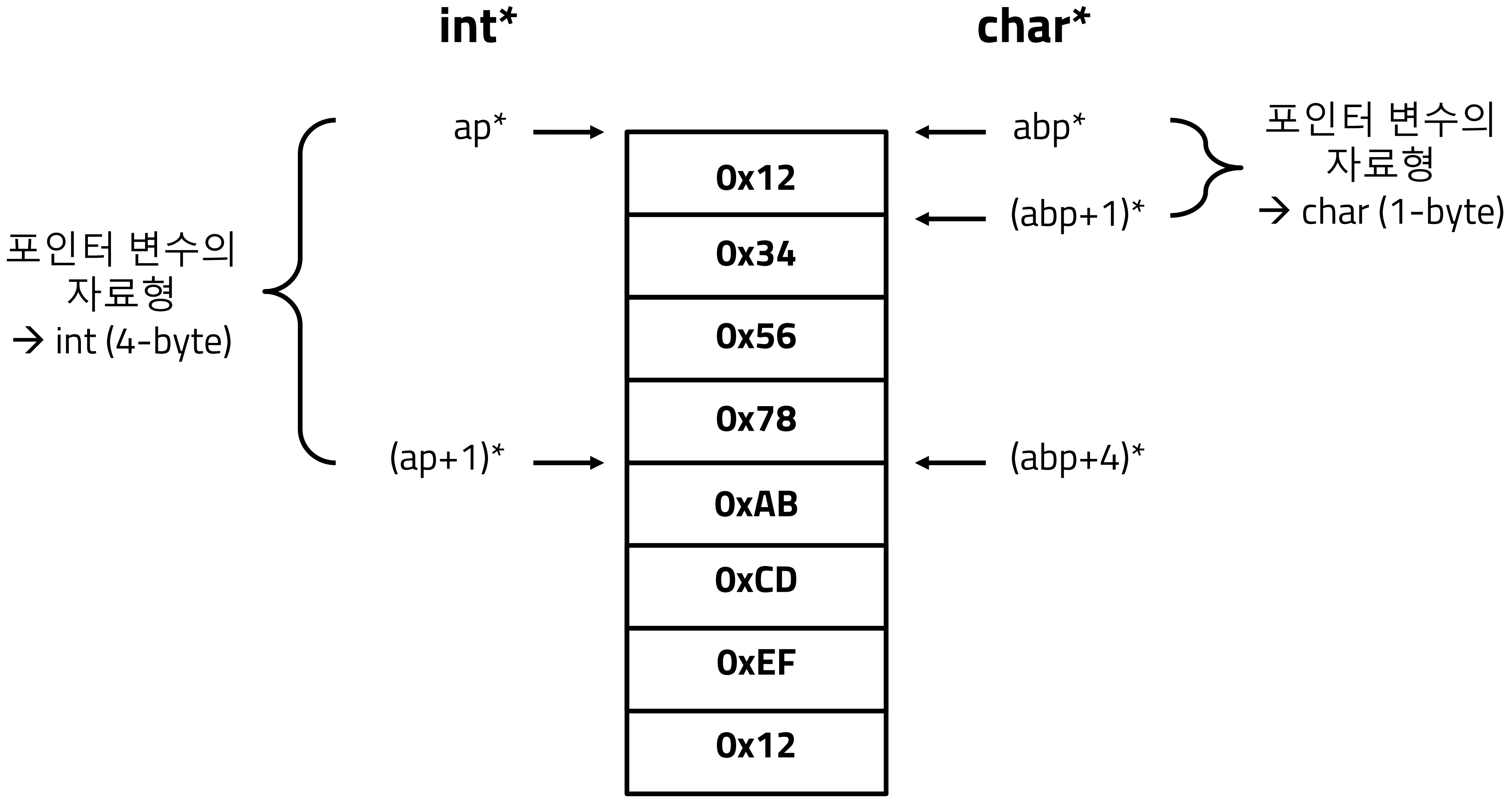
```c
struct ADC_CONFIG adc1 = { 0x01, 0x80, 0xF000 };
printf("CNFG1: 0x%02X\n", adc1.CNFG1);
printf("CNFG2: 0x%02X\n", adc1.CNFG2);
printf("MODE: 0x%04X\n", adc1.MODE);
```

```c
unsigned char* adc_p = (unsigned char*)&adc1;
mem_inspection(adc_p, sizeof(adc1));
```

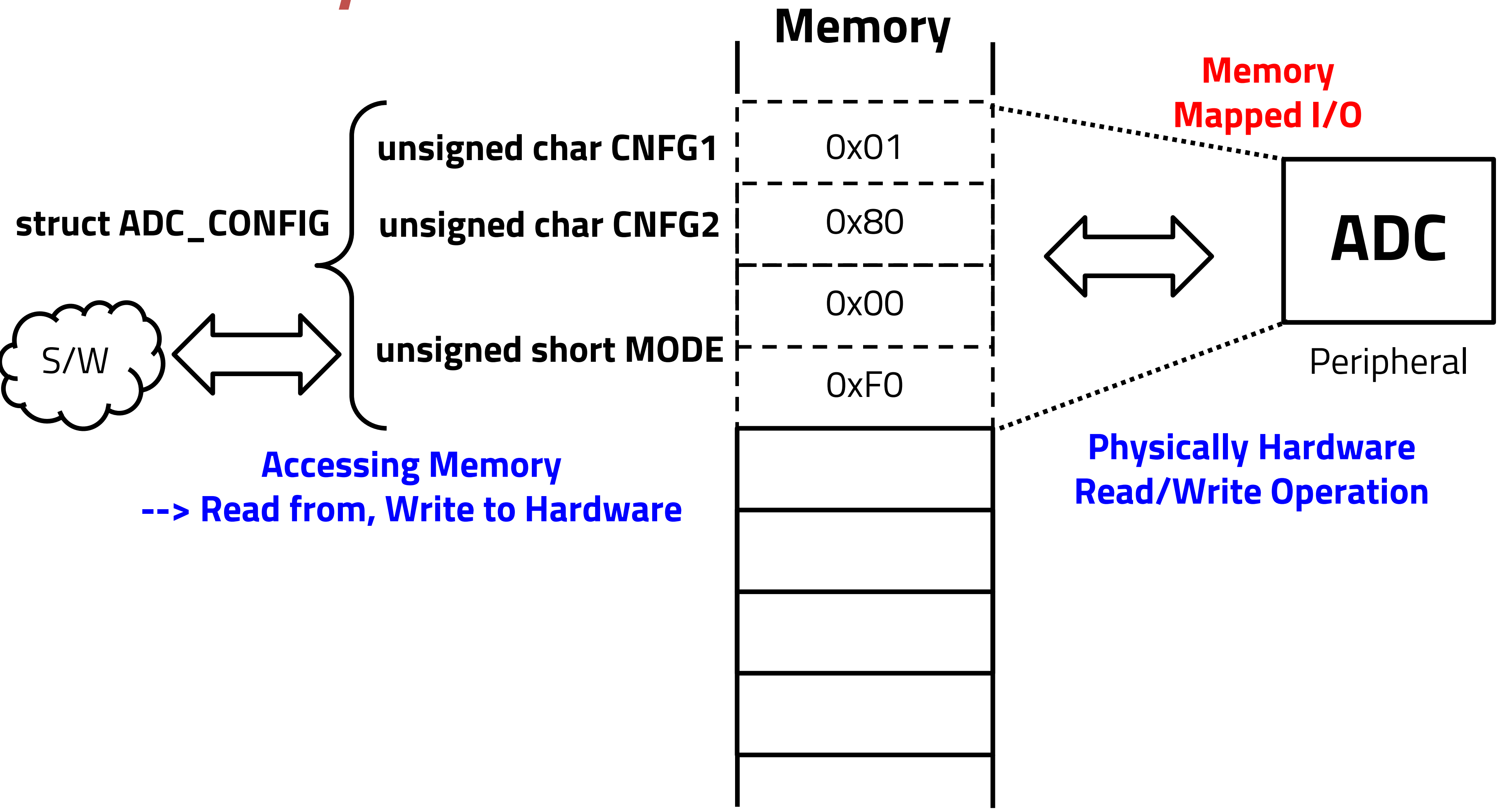
```c
void mem_inspection(unsigned char* p, int N) {
    for(int i=0; i<N; i++)
        printf("mem[%d] is 0x%02X at %p\n", i, *(p+i), p+i);
}
```

```c
*(adc_p+1) = 0x5A; // write CNFG2;
printf("CNFG2: 0x%02X\n", adc1.CNFG2);
```

Accessing Structured Memory via Displacement



Representing Hardware with Structured Memory Allocation


Memory
struct ADC_CONFIG
unsigned char CNFG1
unsigned char CNFG2
unsigned short MODE
0x01
0x80
0x00
0xF0
S/W
Memory Mapped I/O
ADC
Peripheral
Accessing Memory
--> Read from, Write to Hardware
Physically Hardware Read/Write Operation