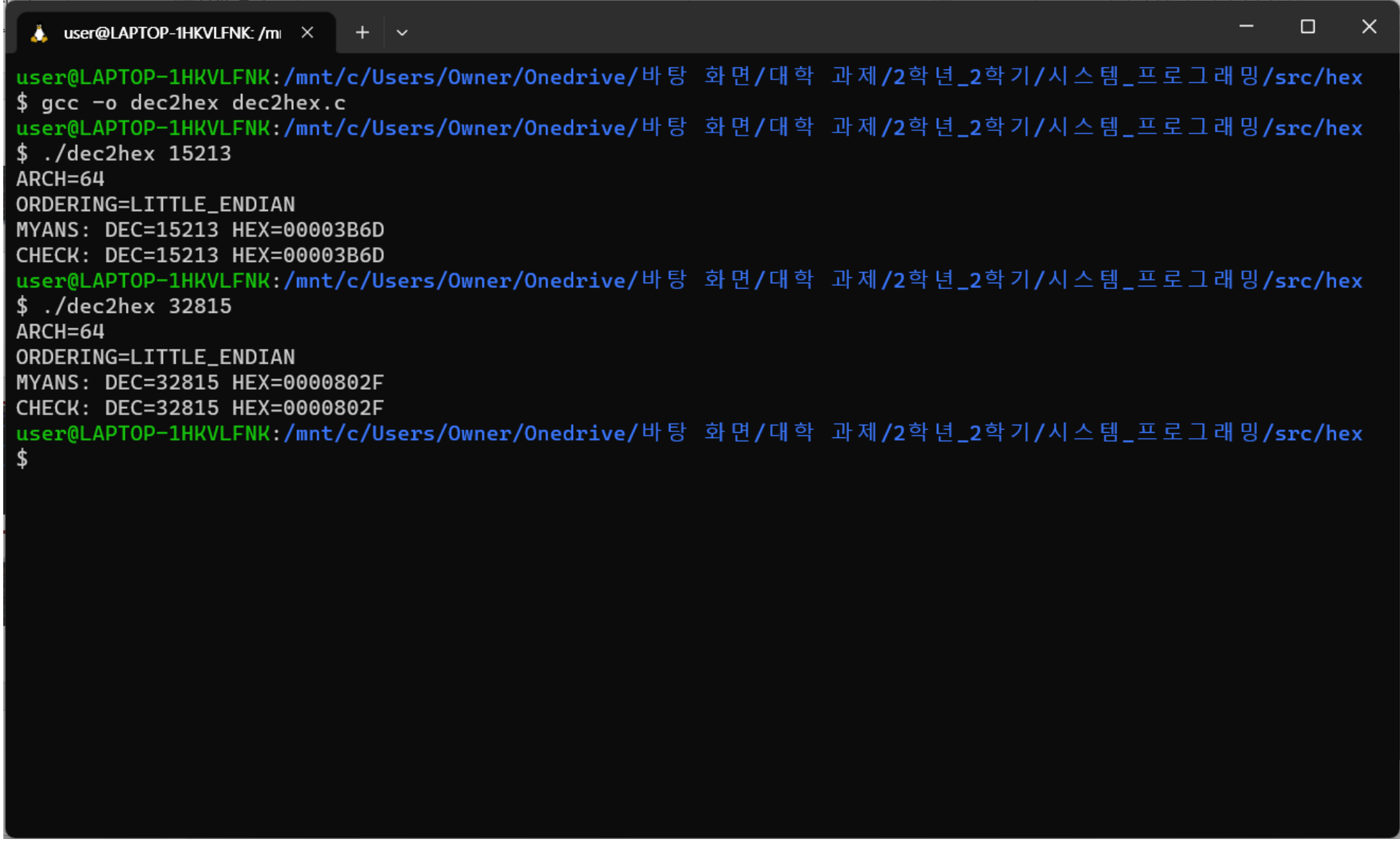
dec2hex.c

```c
#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>

typedef unsigned char* pointer;

bool is64bit() {
    return sizeof(void*) == 8; //void 포인터 크기가 64비트인지 32비트인지 판별
}

bool isBigEndian() {
    int a = 0x01234567;
    if (*((char*)&a) == 0x01) //0x01234567의 메모리 첫 번째 값이 01이면 be 67이면 le
        return true; //big endian
    else
        return false; //little endian
}

void be_show_bytes(unsigned int a) {
    pointer ptr = (pointer)&a; //a의 메모리 주소를 char 포인터로 변환
    for (int i = 0; i < sizeof(unsigned int); i++) { //be이므로 낮은 자리부터 읽어옴
        unsigned char byte = ptr[i];
        //char 포인터이므로 1바이트씩 접근할 수 있음
        printf("%c%c", "0123456789ABCDEF"[byte / 16], "0123456789ABCDEF"[byte % 16]);
        //1바이트씩 2자리 16진수로 변환
    }
    printf("\n");
}
```

dec2hex.c

```c
void le_show_bytes(unsigned int a) {
    pointer ptr = (pointer)&a; //be_show_bytes와 동일
    for (int i = sizeof(unsigned int) - 1; i >= 0; i--) { //le이므로 큰 자리부터 읽어옴
        unsigned char byte = ptr[i];
        printf("%c%c", "0123456789ABCDEF"[byte / 16], "0123456789ABCDEF"[byte % 16]);
    }
    printf("\n");
}

int main(int argc, char* argv[]) {
    if (argc < 2) {
        printf("Usage: ./a.out number\n");
        exit(0);
    }

    unsigned int a = atoi(argv[1]);

    printf("ARCH=%d\n", is64bit() ? 64 : 32);
    printf("ORDERING=%s\n", isBigEndian() ? "BIG_ENDIAN" : "LITTLE_ENDIAN");

    printf("MYANS: DEC=%d HEX=", a);
    isBigEndian() ? be_show_bytes(a) : le_show_bytes(a);

    printf("CHECK: DEC=%d HEX=%.8X\n", a, a);
    return 0;
}
```

실행화면


user@LAPTOP-1HKVLFNK: /mi
user@LAPTOP-1HKVLFNK:/mnt/c/Users/Owner/Onedrive/바탕 화면/대학 과제/2학년_2학기/시스템_프로그래밍/src/hex
$ gcc -o dec2hex dec2hex.c
user@LAPTOP-1HKVLFNK:/mnt/c/Users/Owner/Onedrive/바탕 화면/대학 과제/2학년_2학기/시스템_프로그래밍/src/hex
$ ./dec2hex 15213
ARCH=64
ORDERING=LITTLE_ENDIAN
MYANS: DEC=15213 HEX=00003B6D
CHECK: DEC=15213 HEX=00003B6D
user@LAPTOP-1HKVLFNK:/mnt/c/Users/Owner/Onedrive/바탕 화면/대학 과제/2학년_2학기/시스템_프로그래밍/src/hex
$ ./dec2hex 32815
ARCH=64
ORDERING=LITTLE_ENDIAN
MYANS: DEC=32815 HEX=0000802F
CHECK: DEC=32815 HEX=0000802F
user@LAPTOP-1HKVLFNK:/mnt/c/Users/Owner/Onedrive/바탕 화면/대학 과제/2학년_2학기/시스템_프로그래밍/src/hex
$

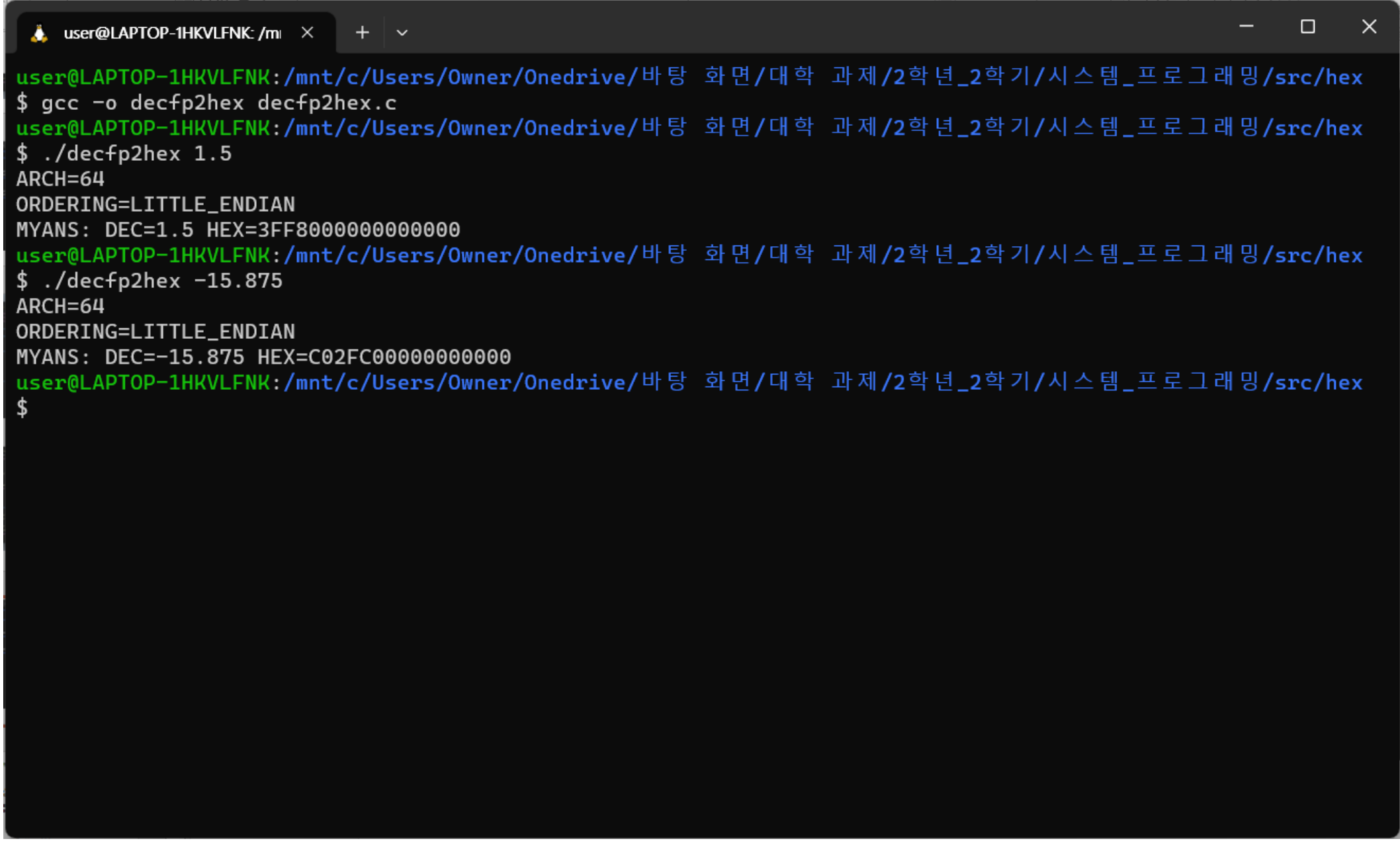
decfp2hex.c

```c
#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>

typedef unsigned char* pointer;

bool is64bit() {
    return sizeof(void*) == 8; //void 포인터 크기가 64비트인지 32비트인지 판별
}

bool isBigEndian() {
    int a = 0x01234567;
    if (*((char*)&a) == 0x01) //0x01234567의 메모리 첫 번째 값이 01이면 be 67이면 le
        return true; //big endian
    else
        return false; //little endian
}

void be_show_bytes(double a) {
    pointer ptr = (pointer)&a; //a의 메모리 주소를 char 포인터로 변환
    for (int i = 0; i < sizeof(double); i++) { //be이므로 낮은 자리부터 읽어옴
        unsigned char byte = ptr[i];
        //char 포인터이므로 1바이트씩 접근할 수 있음
        printf("%c%c", "0123456789ABCDEF"[byte / 16], "0123456789ABCDEF"[byte % 16]);
        //1바이트씩 2자리 16진수로 변환
    }
    printf("\n");
}
```

decfp2hex.c

```c
void le_show_bytes(double a) {
    pointer ptr = (pointer)&a; //le_show_bytes와 동일
    for (int i = sizeof(double) - 1; i >= 0; i--) { //le이므로 큰 자리부터 읽어옴
        unsigned char byte = ptr[i];
        printf("%c%c", "0123456789ABCDEF"[byte / 16], "0123456789ABCDEF"[byte % 16]);
    }
    printf("\n");
}

int main(int argc, char* argv[]) {
    if (argc < 2) {
        printf("Usage: ./a.out number\n");
        exit(0);
    }

    double a = atof(argv[1]);

    printf("ARCH=%d\n", is64bit() ? 64 : 32);
    printf("ORDERING=%s\n", isBigEndian() ? "BIG_ENDIAN" : "LITTLE_ENDIAN");

    printf("MYANS: DEC=%g HEX=", a);
    isBigEndian() ? be_show_bytes(a) : le_show_bytes(a);
    return 0;
}
```

실행화면


user@LAPTOP-1HKVLFNK: /mi
user@LAPTOP-1HKVLFNK:/mnt/c/Users/Owner/Onedrive/바탕 화면/대학 과제/2학년_2학기/시스템_프로그래밍/src/hex
$ gcc -o decfp2hex decfp2hex.c
user@LAPTOP-1HKVLFNK:/mnt/c/Users/Owner/Onedrive/바탕 화면/대학 과제/2학년_2학기/시스템_프로그래밍/src/hex
$ ./decfp2hex 1.5
ARCH=64
ORDERING=LITTLE_ENDIAN
MYANS: DEC=1.5 HEX=3FF8000000000000
user@LAPTOP-1HKVLFNK:/mnt/c/Users/Owner/Onedrive/바탕 화면/대학 과제/2학년_2학기/시스템_프로그래밍/src/hex
$ ./decfp2hex -15.875
ARCH=64
ORDERING=LITTLE_ENDIAN
MYANS: DEC=-15.875 HEX=C02FC00000000000
user@LAPTOP-1HKVLFNK:/mnt/c/Users/Owner/Onedrive/바탕 화면/대학 과제/2학년_2학기/시스템_프로그래밍/src/hex
$

decfp2hex.c 결과 설명

double에서 exp는 11비트 frac는 52비트로 표현한다, exp가 11비트이므로 2048/2 – 1인 1023이 Bias가 된다

1.5를 2진수로 바꾸면 1.1이다

sign (S): 1.5는 양수이므로 부호 비트는 0
exp (E): 소수점 이동은 없으므로 0+Bias인 1023, 이를 2진수로 표현하면 01111111111
Frac (M): 소수 부분이 1이므로 나머지 비트를 0으로 채우면 100…0(총 52비트)

0011111111110000000000000000000000000000000000000000000000000000를 16진수로 변환하면
0011 1111 1111 1000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 ->
3FF8 0000 0000 0000

-15.875를 2진수로 표현하면 -1111.111이다

sign (S): -15.875는 음수이므로 부호 비트는 1
exp (E): 소수점 이동이 3번 있으므로 3+Bias인 1026, 이를 2진수로 표현하면 10000000010
frac (M): 소수 부분이 111111이므로 나머지 비트를 0으로 채우면 111111000…0(총 52비트)

1100000000101111110000000000000000000000000000000000000000000000를 16진수로 변환하면
1100 0000 0010 1111 1100 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 ->
C02F C000 0000 0000

Little_Endian 인 경우엔 메모리에 1바이트씩 거꾸로 저장되지만 (3F와 C0이 맨 오른쪽에 있음)
le_show_bytes 함수에서 읽어올 때 역순으로 읽어오도록 코딩했기 때문에 결과는 동일함