HBO *(host byte order)*

&

NBO *(network byte order)*

byte order

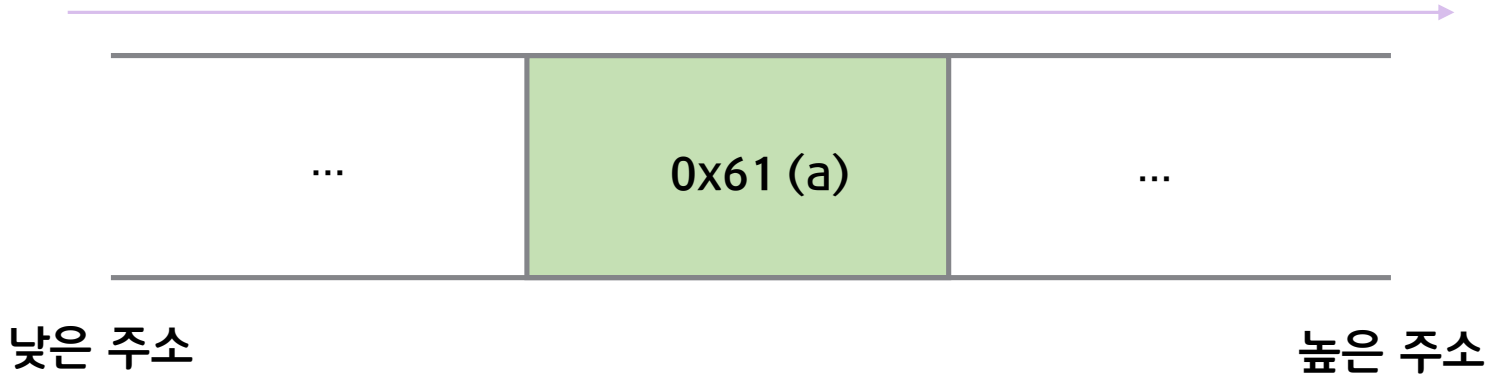
컴퓨터는 데이터를 저장할 때

바이트 단위로 저장한다.

byte order

1 byte

unsigned char data = 0x61; (0x61은 문자로 a를 의미)

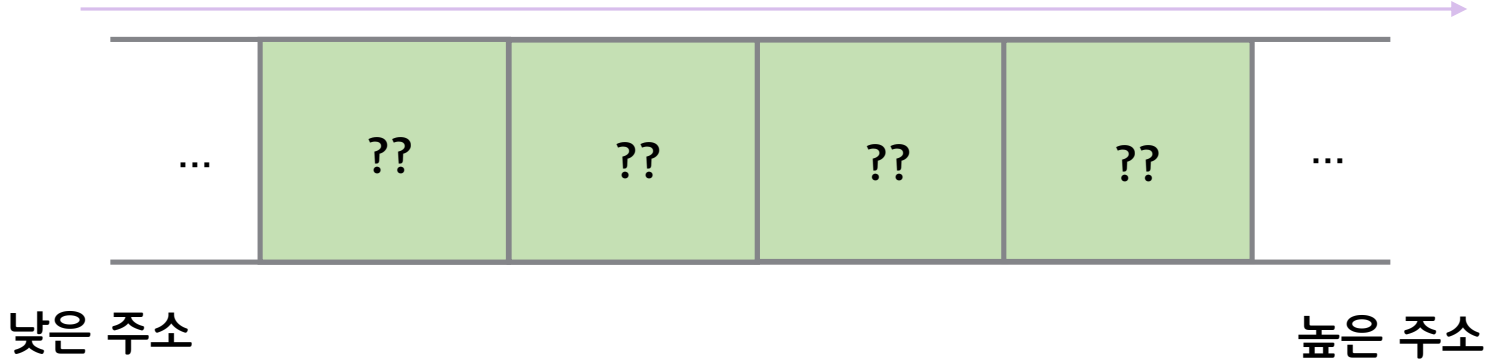


byte order

4 byte

unsigned int data = 0x12345678;

(0x12345678은 정수로 305419896을 의미)

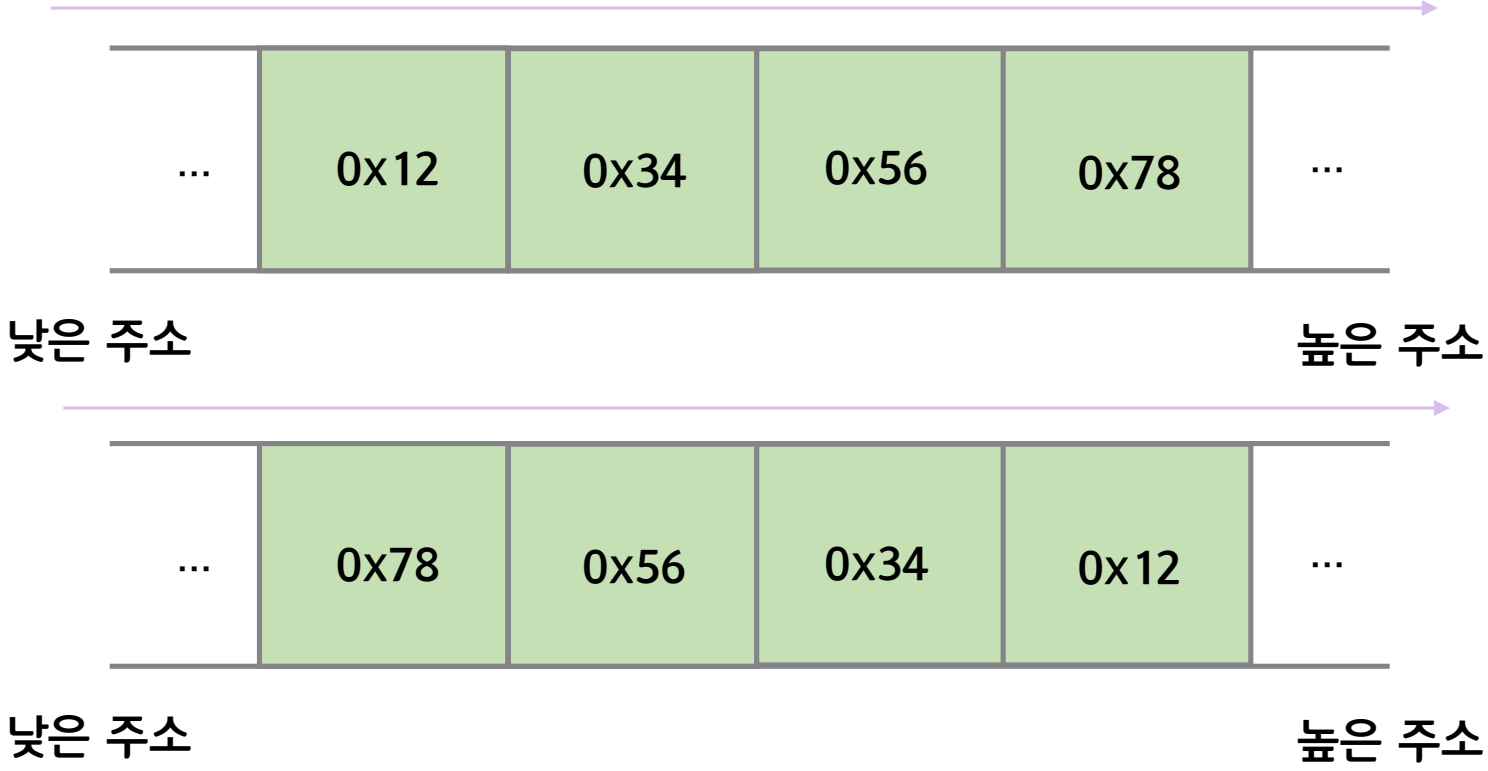


byte order

4 byte

unsigned int data = 0x12345678;

(0x12345678은 정수로 305419896을 의미)

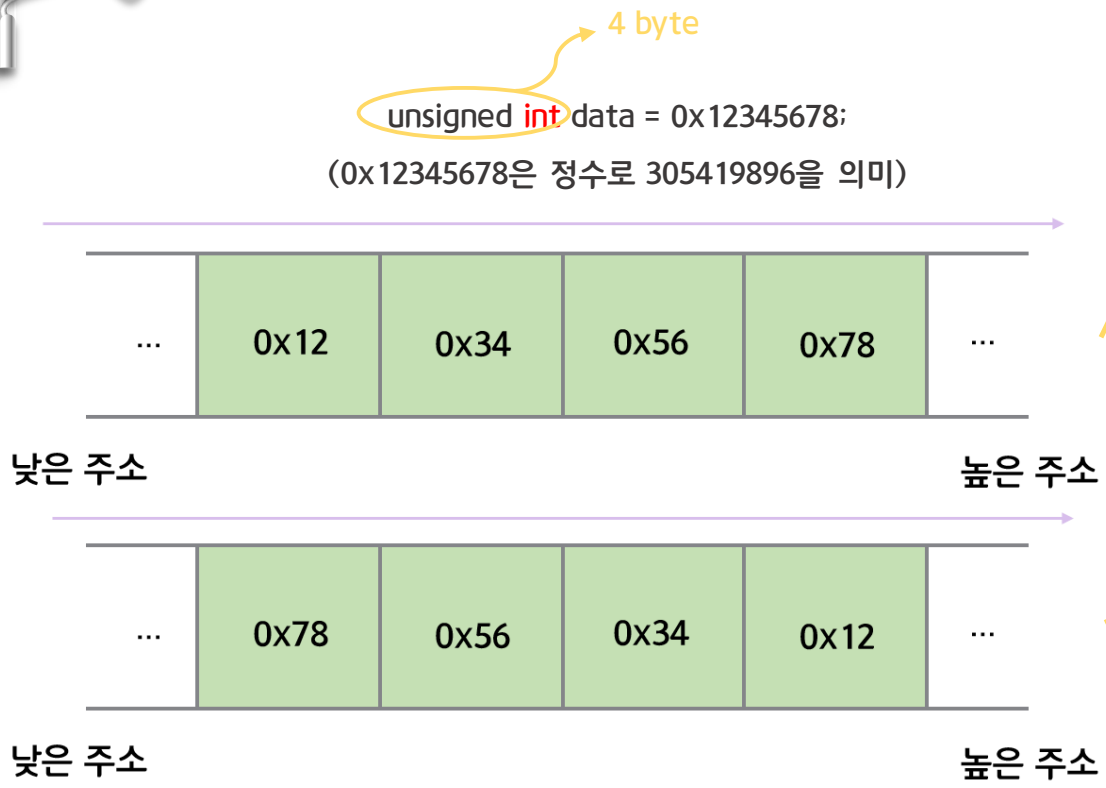


byte order

byte order란??

바이트를 배열하는 방법

byte order



Big-Endian

: 큰 단위의 바이트가 앞에 오는 방법

- sparc, arm, motorola 계열의 cpu

Little-Endian

: 작은 단위의 바이트가 앞에 오는 방법

- intel 계열의 cpu

HBO (host byte order)

HBO

(host byte order)란??

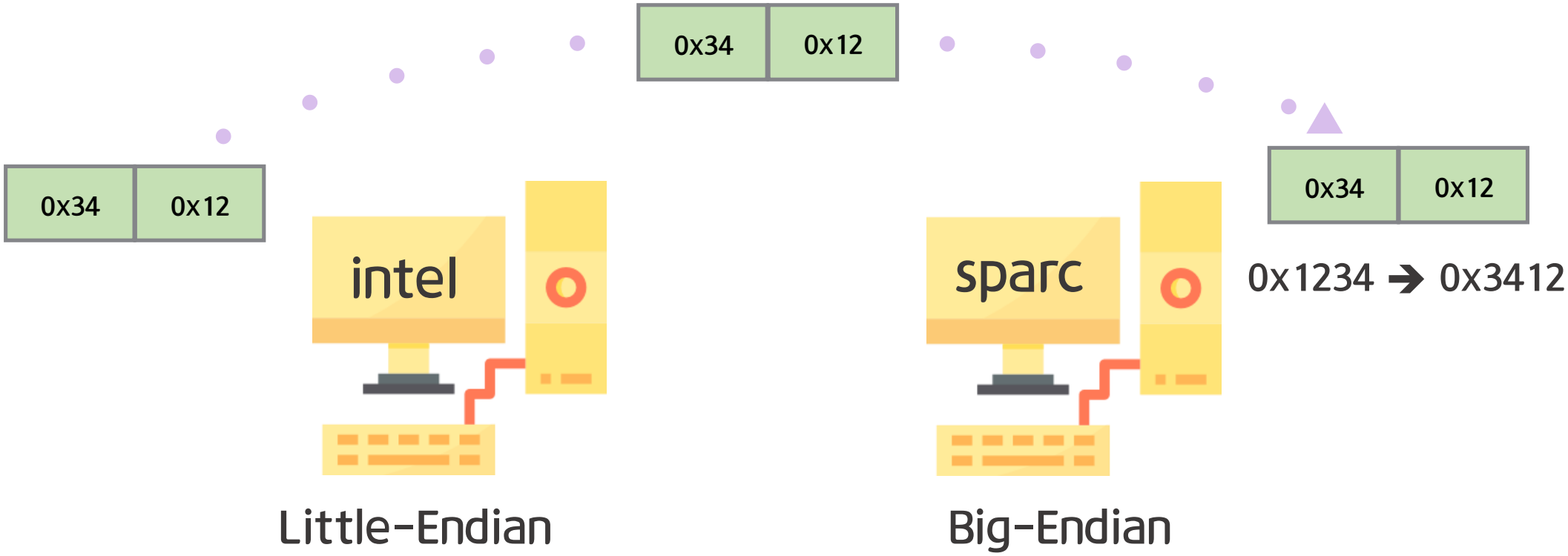
시스템이 사용하는 byte order

sparc 계열 cpu의 host byte order는? Big-Endian

intel 계열 cpu의 host byte order는? Little-Endian

NBO (network byte order)

- Intel 계열의 호스트가 0x1234를 sparc 계열의 호스트에게 보낸다고 가정



NBO (network byte order)

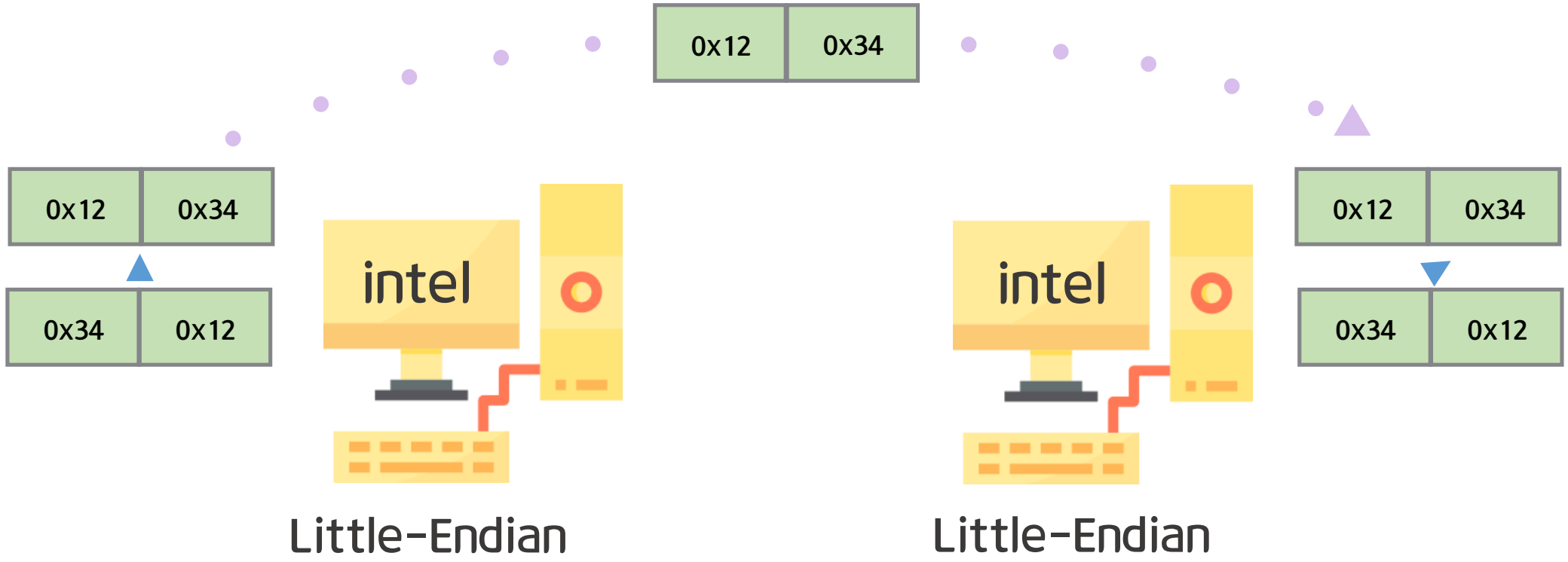
NBO

(network byte order)란??

데이터를 전송할 때 사용하는 byte order

NBO는 Big-Endian !!!

NBO (network byte order)



HBO <-> NBO

예제) 두 파일에 서로 다른 4바이트 데이터를 저장하고, 저장된 값을 더하여 출력하는 코드 작성

```
syntax : add_nbo <file1> <file2>
sample : add_nbo a.bin c.bin

# example
$ echo -n -e \\x00\\x00\\x03\\xe8 > thousand.bin      // thousand.bin 파일에 0x000003e8 저장
$ echo -n -e \\x00\\x00\\x01\\xf4 > five-hundred.bin // five-hundred.bin 파일에 0x000001f4 저장
$ ./add_nbo thousand.bin five-hundred.bin
1000(0x3e8) + 500(0x1f4) = 1500(0x5dc)
```

HBO <-> NBO

예제) 두 파일에 서로 다른 4바이트 데이터를 저장하고, 저장된 값을 더하여 출력하는 코드 작성

- thousand.bin 파일
- 0x000003e8(1000) 저장

- Five-hundred.bin 파일
- 0x000001f4(500) 저장

1000 (0x3e8) + 500 (0x1f4) = 1500 (0x5dc)

```
syntax : add_nbo <file1> <file2>
sample : add_nbo a.bin c.bin

# example
$ echo -n -e \\x00\\x00\\x03\\xe8 > thousand.bin      // thousand.bin 파일에 0x000003e8 저장
$ echo -n -e \\x00\\x00\\x01\\xf4 > five-hundred.bin // five-hundred.bin 파일에 0x000001f4 저장
$ ./add_nbo thousand.bin five-hundred.bin
1000(0x3e8) + 500(0x1f4) = 1500(0x5dc)
```

HBO <-> NBO

1) 두 파일 생성 및 데이터 저장

```
root@kali:~/temp# touch thousand.bin                                  // thousand.bin 파일 생성
root@kali:~/temp# echo -n -e \\x00\\x00\\x03\\xe8 > thousand.bin      // thousand.bin 파일에 0x000003e8 저장
root@kali:~/temp# touch five-hundred.bin                              // five-hundred.bin 파일 생성
root@kali:~/temp# echo -n -e \\x00\\x00\\x01\\xf4 > five-hundred.bin  // five-hundred.bin 파일에 0x000001f4 저장
root@kali:~/temp#
```

2) host byte order 확인

장치 사양

디바이스 이름	DESKTOP-M4O4D3M
프로세서	Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz 1.80 GHz
설치된 RAM	8.00GB(7.80GB 사용 가능)
장치 ID	4221BBC6-029A-4620-AF33-5D2CA602B35F
제품 ID	00326-10000-00000-AA380
시스템 종류	64비트 운영 체제, x64 기반 프로세서
펜 및 터치	이 디스플레이에 사용할 수 있는 펜 또는 터치식 입력이 없습니다.

```
root@kali:~/temp/add_nbo# ./add_nbo five-hundred.bin thousand.bin
-20126105(0xf4010000) + -402456576(0xe8030000) = -603717632(0xdc040000)
root@kali:~/temp/add_nbo#
```

HBO <-> NBO

3) HBO (Little-Endian) -> NBO (Big-Endian) 변환

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

void usage() {
    printf("syntax : add_nbo <file1> <file2>");
    printf("sample : add_nbo a.bin c.bin");
}

uint32_t my_ntohl(uint32_t n) {
    return ((n & 0xFF000000) >> 24 | (n & 0x00FF0000) >> 8 |
    (n & 0x0000FF00) <<8 | (n & 0x000000FF) << 24);
}

uint32_t add_nbo(char filename[]) {
    FILE *fp;
    uint32_t buffer;
    size_t num;

    if ((fp = fopen(filename, "rb")) == NULL) {
        printf("Can't open file.\n");
        exit(1);
    }
    if ((num = fread(&buffer, 4, 1, fp)) != 1) {
            printf("Can't read file.\n");
            exit(1);
    }
    fclose(fp);

    buffer = my_ntohl(buffer);

    return buffer;
}

int main(int argc, char* argv[]) {
    if (argc != 3) {
        usage();
        return -1;
    }

    int i;
    uint32_t rv[3];
    for (i=1; i<3; i++) {
        rv[i] = add_nbo(argv[i]);
    }

    printf("%d(%#x) + %d(%#x) = %d(%#x)\n", rv[1], rv[1],
    rv[2], rv[2], rv[1] + rv[2], rv[1] + rv[2]);
}
```

```c
uint32_t my_ntohl(uint32_t n) {
    return ((n & 0xFF000000) >> 24 | (n & 0x00FF0000) >> 8 |
    (n & 0x0000FF00) <<8 | (n & 0x000000FF) << 24);
}
```

& 연산자	시프트 연산자	| 연산자
0001	00000110 << 2	0001
&	↓↓	|
0011	00011000	0011
↓		↓
0001		0011

HBO <-> NBO

```
uint32_t my_ntohl(uint32_t n) {
    return ((n & 0xFF000000) >> 24 | (n & 0x00FF0000) >> 8 |
    (n & 0x0000FF00) <<8 | (n & 0x000000FF) << 24);
}
```

& 연산자	시프트 연산자	| 연산자
0001	00000110 << 2	0001
&		|
0011	00011000	0011
↓		↓
0001		0011

3) HBO (Little-Endian) -> NBO (Big-Endian) 변환

n : Little-Endian으로 저장된 데이터

(0xe8030000, 0xf4010000)

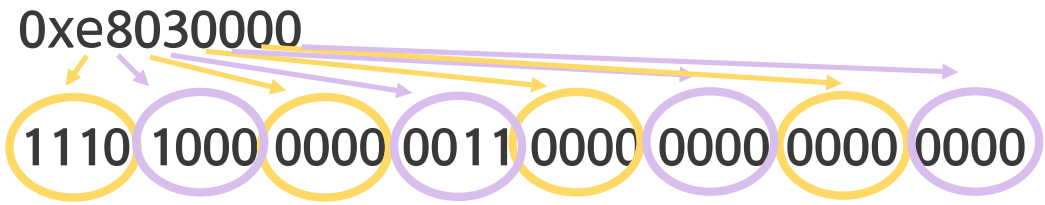
① n = 0xe8030000이라 할 때,

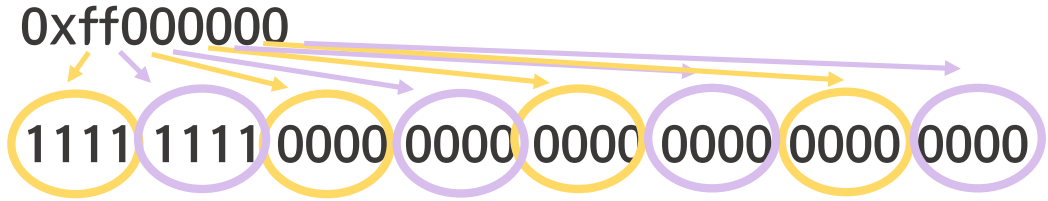
n & 0xFF000000 ?? 0xe8000000

n & 0x00FF0000 ?? 0x00030000

n & 0x0000FF00 ?? 0x00000000

n & 0x000000FF ?? 0x00000000





HBO <-> NBO

```
uint32_t my_ntohl(uint32_t n) {
    return ((n & 0xFF000000) >> 24 | (n & 0x00FF0000) >> 8 |
    (n & 0x0000FF00) <<8 | (n & 0x000000FF) << 24);
}
```

& 연산자	시프트 연산자	| 연산자
0001	00000110 << 2	0001
&	⇓	|
0011	00011000	0011
↓		↓
0001		0011

3) HBO (Little-Endian) -> NBO (Big-Endian) 변환

n : Little-Endian으로 저장된 데이터

(0xe8030000, 0xf4010000)

② n = 0xe8030000이라 할 때,

(n & 0xFF000000) >> 24 ??

(== 0xe8000000) >> 24?? 0x000000e8

0x00030000 >> 8 → 0x00000300

0x00000000 << 8 → 0x00000000

0x00000000 << 24 → 0x00000000

HBO <-> NBO

```
uint32_t my_ntohl(uint32_t n) {
    return ((n & 0xFF000000) >> 24 | (n & 0x00FF0000) >> 8 |
    (n & 0x0000FF00) <<8 | (n & 0x000000FF) << 24);
}
```

& 연산자	시프트 연산자	| 연산자
0001	00000110 << 2	0001
&	//	|
0011	00011000	0011
↓		↓
0001		0011

3) HBO (Little-Endian) -> NBO (Big-Endian) 변환

n : Little-Endian으로 저장된 데이터

(0xe8030000, 0xf4010000)

③ n = 0xe803000이라 할 때,

0x000000e8 | 0x00000300 |

0x00000000 | 0x00000000

➔ 0x000003e8

HBO <-> NBO

4) 더한 값 출력

```
41      int i;
42      uint32_t rv[3];
43      for (i=1; i<3; i++) {
44          rv[i] = add_nbo(argv[i]);
45      }
46
47      printf("%d(%#x) + %d(%#x) = %d(%#x)\n", rv[1], rv[1],
48      rv[2], rv[2], rv[1] + rv[2], rv[1] + rv[2]);
49  }
50
```

rv[1] 에는 0x000003e8

rv[2] 에는 0x000001f4

```
root@kali:~/temp/add_nbo# ./add_nbo five-hundred.bin thousand.bin
500(0x1f4) + 1000(0x3e8) = 1500(0x5dc)
root@kali:~/temp/add_nbo#
```

HBO <-> NBO

- HBO와 NBO간 변환시켜주는 함수

```
root@kali:~/temp/add_nbo# ./add_nbo five-hundred.bin thousand.bin
500(0×1f4) + 1000(0×3e8) = 1500(0×5dc)
root@kali:~/temp/add_nbo#
```

- netinet/in.h 헤더파일에 존재

Function	Size	conversion
uint16_t ntohs(uint16_t netshort);	2 byte	NBO to HBO
uint16_t htons(uint16_t hostshort);	2 byte	HBO to NBO
uint32_t ntohl(uint32_t netlong);	4 byte	NBO to HBO
uint32_t htonl(uint32_t hostlong);	4 byte	HBO to NBO

감사합니다 (: