컴퓨터구조1 과제

> [19반] 정민석_7000

리눅스 개발환경 구축하기

기존에 구축해둔 Ubuntu 24.0.4 환경의 서버가 있습니다. 해당 서버에서 실습을 진행하겠습니다.

sizeof 연산 타이핑해보기

```
#include <stdio.h>

int main(void) {
    printf("char: %lu\n", sizeof(char));
    printf("short: %lu\n", sizeof(short));
    printf("int: %lu\n", sizeof(int));
    printf("long: %lu\n", sizeof(long));
    printf("long long: %lu\n", sizeof(long long));
    printf("float: %lu\n", sizeof(float));
    printf("double: %lu\n", sizeof(double));
    printf("long double: %lu\n", sizeof(long double));
    printf("pointer: %lu\n", sizeof(void *));
    return 0;
}
```

```
$ ./a.out
char: 1
short: 2
int: 4
long: 8
long long: 8
float: 4
```

```
double: 8
long double: 8
pointer: 8
```

오버플로 예제를 언더플로로 바꿔서 해보기

음의 오버플로와 언더플로를 각각 수행하였습니다.

```
#include <stdio.h>

int main(void) {
    // 음의 오버플로
    int n = -2147483648;
    n -= 1;
    printf("%d\n", n);

    // 언더플로
    float f = 1.0e-38;
    printf("%f\n", f);
}
```

```
(gdb) b 7
Breakpoint 1 at 0x1160: file overflow.c, line 7.
(gdb) b 14
Breakpoint 2 at 0x11b2: file overflow.c, line 14.
(gdb) r
Breakpoint 1, main () at overflow.c:7
7	    printf("%d\n", n);
(gdb) p/t n
$1 = 1111111111111111111111111111111
(gdb) c
Continuing.
2147483647

Breakpoint 2, main () at overflow.c:14
14	    printf("%f\n", f);
```

```
(gdb) p/t f
$2 = 0
(gdb) exit
```

비트 연산 프로그램 바꿔보기

- 특정 위치의 비트를 끄는 함수 구현

```
unsigned char clear_bit(unsigned char x, int n) {
    return x & ~(1 << n);
}
```

- 사용자의 입력(특정위치 - int값)을 받도록 수정

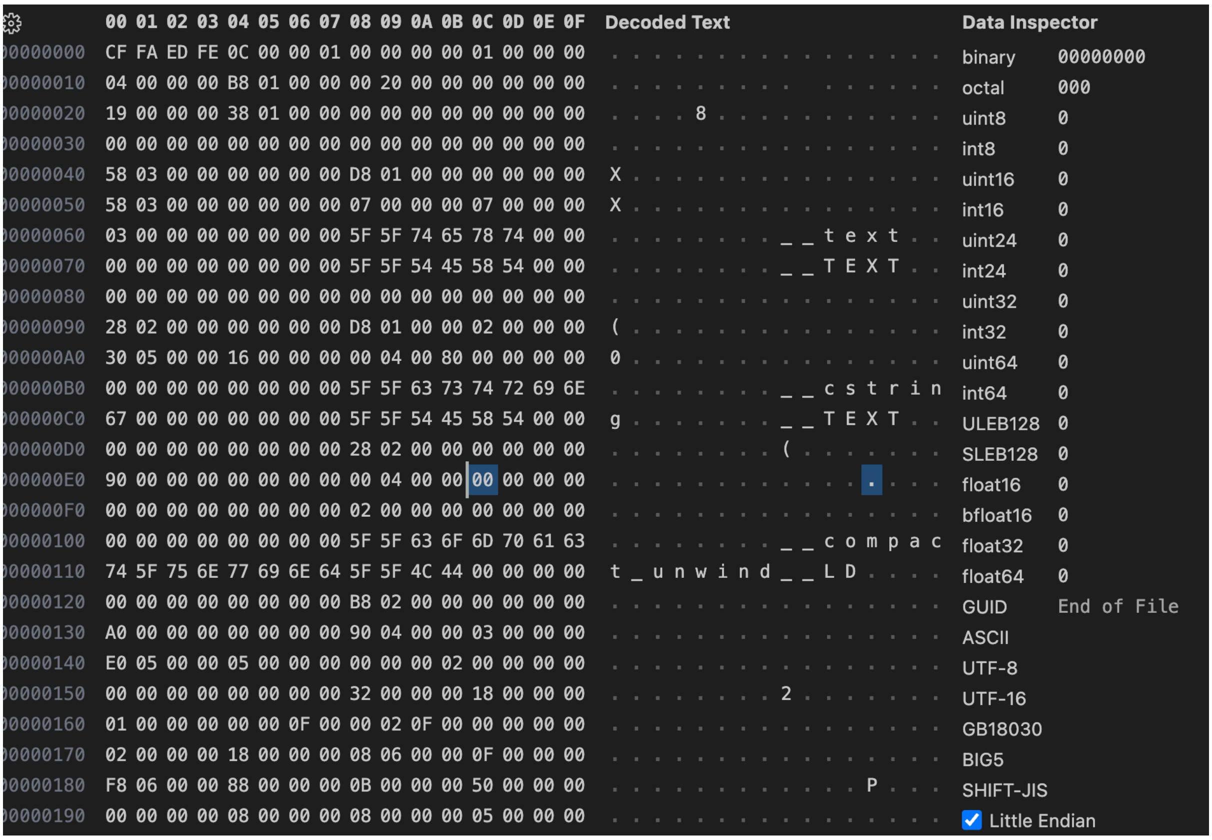
```
#include <stdio.h>

unsigned char clear_bit(unsigned char x, int n) {
    return x & ~(1 << n);
}

void print_bits(unsigned char x) {
    for (int i = 7; i >= 0; i--) {
            printf("%d", (x >> i) & 1);
    }
}

unsigned char input_bits() {
    unsigned int origin;
    unsigned char bits = 0;
    int tmp = 1e7;

    printf("비트열을 입력하세요: ");
    scanf("%u", &origin);
    for (int i = 0; i < 8; i++) {
          bits = (bits << 1) | ((origin / tmp) % 10);
          tmp /= 10;
    }
    return bits;
}

int input_index() {
    int index;
    printf("\n0부터 7사이의 초기화할 인덱스를 입력하세요: ");
    scanf("%d", &index);

    if (index < 0 || index > 7) {
        printf("0부터 7사이의 값을 입력하세요.");
        return 1;
    }   return index;
}

int main(void) {
    unsigned char srcBits;
    unsigned char distBits;
    int taregtIndex;

    srcBits = input_bits();
    taregtIndex = input_index();
    distBits = clear_bit(srcBits, taregtIndex);

    print_bits(distBits);
}
```

```
$ gcc bit_calc.c

$ ./a.out
비트열을 입력하세요: 11

0부터 7사이의 초기화할 인덱스를 입력하세요: 0
00000010
```

C언어가 기계어가 되는 과정 직접 해보기

```
$ gcc -E bit_calc.c -o bit_calc.i

$ ls | grep bit_calc
bit_calc.c
bit_calc.i

$ head -7 bit_calc.i
# 1 "bit_calc.c"
# 1 "<built-in>" 1
# 1 "<built-in>" 3
# 424 "<built-in>" 3
# 1 "<command line>" 1
# 1 "<built-in>" 2
# 1 "bit_calc.c" 2

$ gcc -S bit_calc.i -o bit_calc.s

$ ls | grep bit_calc
bit_calc.c
bit_calc.i
bit_calc.s

$ head  bit_calc.s
	.section	__TEXT,__text,regular,pure_instructions
	.build_version macos, 15, 0	sdk_version 15, 2
	.globl	_clear_bit                      ; -- Begin function clear_bit
	.p2align	2
_clear_bit:                             ; @clear_bit
	.cfi_startproc
; %bb.0:
	sub	sp, sp, #16
	.cfi_def_cfa_offset 16
	strb	w0, [sp, #15]

$ gcc -c bit_calc.s -o bit_calc.o
$ ls | grep bit_calc
bit_calc.c
bit_calc.i
bit_calc.o
bit_calc.s

$ head -3 bit_calc.o
����
   � 8X�X__text__TEXT(�0�__cstring__TEXT(�__compact_unwind__LD�
�C��?9�                                                P
      ��?@9�
           @�)�R)!)
�C��_������{��C���8��R�
                    ��
                     @q鮭(7��_8�
                               @)�
���

$ file bit_calc.*
bit_calc.c: c program text, Unicode text, UTF-8 text
bit_calc.i: c program text, Unicode text, UTF-8 text
bit_calc.o: Mach-O 64-bit object arm64
bit_calc.s: assembler source text, ASCII text
```

hex editer로 bit_calc.o를 출력한 결과



```
          00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded Text
00000000  CF FA ED FE 0C 00 00 01 00 00 00 00 01 00 00 00  ................
00000010  04 00 00 00 B8 01 00 00 00 20 00 00 00 00 00 00  ......... ......
00000020  19 00 00 00 38 01 00 00 00 00 00 00 00 00 00 00  ....8...........
00000030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000040  58 03 00 00 00 00 00 00 D8 01 00 00 00 00 00 00  X...............
00000050  58 03 00 00 00 00 00 00 07 00 00 00 07 00 00 00  X...............
00000060  03 00 00 00 00 00 00 00 5F 5F 74 65 78 74 00 00  ........__text..
00000070  00 00 00 00 00 00 00 00 5F 5F 54 45 58 54 00 00  ........__TEXT..
00000080  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000090  28 02 00 00 00 00 00 00 D8 01 00 00 02 00 00 00  (...............
000000A0  30 05 00 00 16 00 00 00 00 04 00 80 00 00 00 00  0...............
000000B0  00 00 00 00 00 00 00 00 5F 5F 63 73 74 72 69 6E  ........__cstrin
000000C0  67 00 00 00 00 00 00 00 5F 5F 54 45 58 54 00 00  g.......__TEXT..
000000D0  00 00 00 00 00 00 00 00 28 02 00 00 00 00 00 00  ........(.......
000000E0  90 00 00 00 00 00 00 00 00 04 00 00 00 00 00 00  ................
000000F0  00 00 00 00 00 00 00 00 02 00 00 00 00 00 00 00  ................
00000100  00 00 00 00 00 00 00 00 5F 5F 63 6F 6D 70 61 63  ........__compac
00000110  74 5F 75 6E 77 69 6E 64 5F 5F 4C 44 00 00 00 00  t_unwind__LD....
00000120  00 00 00 00 00 00 00 00 B8 02 00 00 00 00 00 00  ................
00000130  A0 00 00 00 00 00 00 00 90 04 00 00 03 00 00 00  ................
00000140  E0 05 00 00 05 00 00 00 00 00 00 02 00 00 00 00  ................
00000150  00 00 00 00 00 00 00 00 32 00 00 00 18 00 00 00  ........2.......
00000160  01 00 00 00 00 00 0F 00 00 02 0F 00 00 00 00 00  ................
00000170  02 00 00 00 18 00 00 00 08 06 00 00 0F 00 00 00  ................
00000180  F8 06 00 00 88 00 00 00 0B 00 00 00 50 00 00 00  ............P...
00000190  00 00 00 00 08 00 00 00 08 00 00 00 05 00 00 00  ................
```

Data Inspector

binary	00000000
octal	000
uint8	0
int8	0
uint16	0
int16	0
uint24	0
int24	0
uint32	0
int32	0
uint64	0
int64	0
ULEB128	0
SLEB128	0
float16	0
bfloat16	0
float32	0
float64	0
GUID	End of File
ASCII	
UTF-8	
UTF-16	
GB18030	
BIG5	
SHIFT-JIS	

☑ Little Endian