REPORT

[어셈블리 실습 01]

과 목 :
시스템소프트웨어 03
담당교수 : 석문기 교수님
학 과 : 컴퓨터공학과
학 번 : 2021111971
이 름 : 이재혁

문제 1

ex01_01.asm	실행결과

```
jaehyuk — 2021111971@linuxserver1: ~ —
  GNU nano 4.8                          ex01_01.asm
section .data
    msg db 'Hello, NASM!', 0
section .text
    global _start
_start:
    mov rax, 1
    mov rdi, 1
    mov rsi, msg
    mov rdx, 13
    syscall
    mov rax, 60
    xor rdi, rdi
    syscall
```

```
jaehyuk — 2021111971@linuxserver1: ~ — ssh 2021111971@cs.
[2021111971@linuxserver1:~$ nano ex01_01.asm
[2021111971@linuxserver1:~$ nasm -f elf64 ex01_01.asm -o ex01_01.o
[2021111971@linuxserver1:~$ ld ex01_01.o -o ex01_01
[2021111971@linuxserver1:~$ ./ex01_01
Hello, NASM!2021111971@linuxserver1:~$
```

문제 2

ex01_main.cpp	ex01_addsub.asm

```
jaehyuk — 2021111971@linuxserver1: ~ — ssh 2021111971@cs
// 2021111971 이재혁

#include "ex01.h"
#include <iostream>
using namespace std;

void DisplayResults(int a, int b, int c, int d, int r1, int r2) {
    cout << "----- Results for ex01 -----\n";
    cout << "a = " << a << '\n';
    cout << "b = " << b << '\n';
    cout << "c = " << c << '\n';
    cout << "d = " << d << '\n';
    cout << "r1 = " << r1 << '\n';
    cout << "r2 = " << r2 << '\n';
    cout << '\n';
}

extern "C" int AddSubI32_a(int a, int b, int c, int d);
// (a + b) - (c + d) + 7
// asm과 link할 함수

int main() {
    int a = 10;
    int b = 40;
    int c = 9;
    int d = 6;
    int r1 = (a + b) - (c + d) + 7;
    int r2 = AddSubI32_a(a, b, c, d);
    DisplayResults(a, b, c, d, r1, r2);
    return 0;
}
```

```
jaehyuk — 2021111971@linuxserver1: ~ —
section .text
    global AddSubI32_a
AddSubI32_a:
    ; int AddSubI32_a(int a, int b, int c, int d)
    ; Calculate (a + b) - (c + d) + 7
    ; rdi = a, rsi = b, rdx = c, rcx = d
    mov eax, edi ; eax = a
    add eax, esi ; eax = a + b
    sub eax, edx ; eax = a + b - c
    sub eax, ecx ; eax = a + b - c - d
    add eax, 7 ; eax = a + b - c - d + 7
    ret
```

std namespace 사용했습니다.

nl 변수 대신 바로 '₩n'문자를 출력했습니다.

ex01.h	실행결과

```
jaehyuk — 2021111971@linuxserver1: ~ — ssh 2021111971@
#pragma once
void DisplayResults(int a, int b, int c, int d, int r1, int r2);
extern "C" int AddSubI32_a(int a, int b, int c, int d);
```

```
[2021111971@linuxserver1:~$ nasm -f elf64 ex01_addsub.asm -o ex01_addsub.o
[2021111971@linuxserver1:~$ g++ -no-pie -o ex01_main ex01_addsub.o ex01_main.cpp
[2021111971@linuxserver1:~$ ./ex01_main
----- Results for ex01 -----
a = 10
b = 40
c = 9
d = 6
r1 = 42
r2 = 42
```

문제 3

ex02_main.cpp

jaehyuk — 2021111971@linuxserver1: ~ — ssh 20211119

```cpp
#include <iostream>
using namespace std;

// 어셈블리 함수 선언
extern "C" int BitwiseAnd(int a, int b);
extern "C" int BitwiseOr(int a, int b);
extern "C" int BitwiseXor(int a, int b);
extern "C" int BitwiseNot(int a);

int main() {
    int a = 0x55; // 01010101
    int b = 0xAA; // 10101010
    // 비트 연산 결과 출력
    cout << "a AND b = " << hex << BitwiseAnd(a, b) << endl;
    cout << "a OR b = " << hex << BitwiseOr(a, b) << endl;
    cout << "a XOR b = " << hex << BitwiseXor(a, b) << endl;
    cout << "NOT a = " << hex << BitwiseNot(a) << endl;
    return 0;
}
```

namespace std 사용

ex02_bitwise.asm

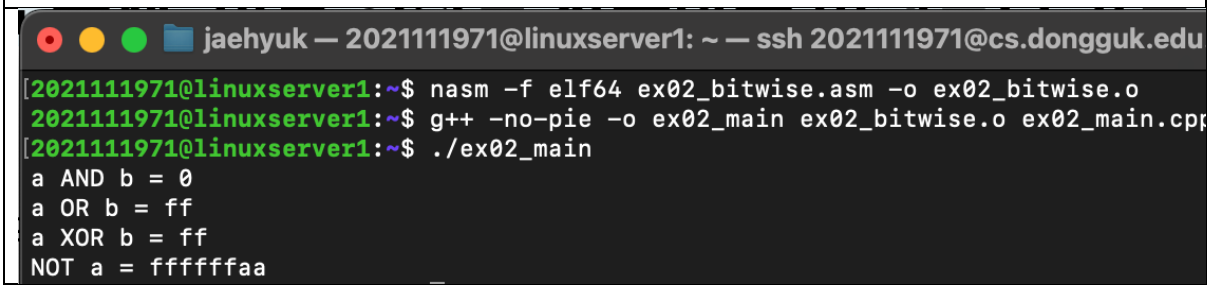
jaehyuk — 2

```
section .text
    global BitwiseAnd
    global BitwiseOr
    global BitwiseXor
    global BitwiseNot

BitwiseAnd:
    mov eax, edi
    and eax, esi
    ret

BitwiseOr:
    mov eax, edi
    or  eax, esi
    ret

BitwiseXor:
    mov eax, edi
    xor eax, esi
    ret

BitwiseNot:
    mov eax, edi
    not eax
    ret
```

작성코드 text

```
section .text
    global BitwiseAnd
    global BitwiseOr
    global BitwiseXor
    global BitwiseNot

BitwiseAnd:
    mov eax, edi
    and eax, esi
    ret

BitwiseOr:
    mov eax, edi
    or  eax, esi
    ret

BitwiseXor:
    mov eax, edi
    xor eax, esi
    ret

BitwiseNot:
    mov eax, edi
    not eax
    ret
```

실행결과



결과 분석

a : 0000 0000 0000 0000 0000 0000 0101 0101 (32 비트)

b : 0000 0000 0000 0000 0000 0000 1010 1010 (32 비트)

a & b : 0000 0000 0000 0000 0000 0000 0000 0000

➔ 0

a | b : 0000 0000 0000 0000 0000 0000 1111 1111

➔ ff

a ^ b : 0000 0000 0000 0000 0000 0000 1111 1111

➔ ff

~a : 1111 1111 1111 1111 1111 11111 1010 1010

➔ ffffffaa