

**2016 시스템 프로그래밍**  
**- Lab 05 -**

제출일자	2016.10.18
분 반	02
이 름	신종욱
학 번	201302423

## 실습 1    Test.s

.section .data//데이터 섹션이 시작된다.

msg :

.string "Number = %d Age = %d Name = %s \n"

Number:

.int 201302423

Age:

.int 24

Name :

.string "ShinJongWook\0"

.section .text//텍스트 섹션이 시작

.global main// main을 전역으로 선언

main:

movq \$msg, %rdi// msg의 주소를 1st argument인 rdi에 저장한다.

movq Number, %rsi// Number의 값을 2nd argument인 rsi에 저장한다.

movq Age, %rdx// Age의 값을 3rd argument인 rdx에 저장한다.

movq \$Name, %rcx// Name의 주소를 4th argument인 rcx에 저장한다.

movq \$0,%rax// return value인 rax에 0을 저장

call printf 프린터 함수를 호출한다.

movq \$0,%rax// return value인 rax에 0을 저장

ret// call함수를 끝내고 돌아간다.

movq \$0,%rax가 두 개 나오는데 차이가 궁금하다.

첫 번째의 경우 \$0을 바꾸면 프로그램이 안되는데

두 번째 경우 \$0대신 다른 주소값을 넣어도 잘 실행된다.

## 실습 2

### 1 swap에 break point 설정후 스택디버깅

```
c201302423@localhost: ~/04
(gdb) b Test.c :swap
Breakpoint 1 at 0x40077d: file Test.c, line 30.
(gdb) b Test.c : 38
Breakpoint 2 at 0x4007e7: file Test.c, line 38.
(gdb) r
Starting program: /home/sys02/c201302423/04/Test.out
2 IHS 201302423 JUS

Breakpoint 1, swap (a=0x602010, b=0x602030) at Test.c:30
30 void swap(struct student *a,struct student *b){
(gdb) bt full
#0 swap (a=0x602010, b=0x602030) at Test.c:30
    temp = 0
    t = "\002\000\000"
#1 0x0000000000400716 in main () at Test.c:22
    A = 0x602010
    B = 0x602030
(gdb) c
Continuing.

Breakpoint 2, swap (a=0x602010, b=0x602030) at Test.c:38
38 }
(gdb) bt full
#0 swap (a=0x602010, b=0x602030) at Test.c:38
    temp = 2
    t = "IHS"
#1 0x0000000000400716 in main () at Test.c:22
    A = 0x602010
    B = 0x602030
(gdb)
```

swap 함수가 시작할때와 끝날 때 breakpoint를 걸어  
스택이 어떻게 바뀌는지 확인하였다.

시작때 bt full을 해보면 정수를 저장하는 Temp와 문자열을 저장하  
는 t가 비어있다는걸 확인할 수 있다.

그리고 swap이 끝난 후에는 A의 정수와 문자열을  
임시로 저장하였기 때문에  
2와 IHS가 저장되어있다

```
c201302423@localhost: ~/04
Reading symbols from Test.out...(no debugging symbols found)...done.
(gdb) x/20xb swap
0x40076d <swap>:      0x55    0x48    0x89    0xe5    0x48    0x83    0xec    0x30
0x400775 <swap+8>:    0x48    0x89    0x7d    0xd8    0x48    0x89    0x75    0xd0
0x40077d <swap+16>:   0x64    0x48    0x8b    0x04
(gdb)
```

x/20xb는

메모리상태를 검사하는데(x/) 16진수(x)로 1바이트(b) 단위로  
20바이트(20) 출력한다는 의미다.

```

c201302423@localhost: ~/04
(gdb) disas swap
Dump of assembler code for function swap:
0x000000000040076d <+0>:    push    %rbp
0x000000000040076e <+1>:    mov     %rsp,%rbp
0x0000000000400771 <+4>:    sub     $0x30,%rsp
0x0000000000400775 <+8>:    mov     %rdi,-0x28(%rbp)
0x0000000000400779 <+12>:   mov     %rsi,-0x30(%rbp)
0x000000000040077d <+16>:   mov     %fs:0x28,%rax
0x0000000000400786 <+25>:   mov     %rax,-0x8(%rbp)
0x000000000040078a <+29>:   xor     %eax,%eax
0x000000000040078c <+31>:   mov     -0x28(%rbp),%rax
0x0000000000400790 <+35>:   mov     0x4(%rax),%eax
0x0000000000400793 <+38>:   mov     %eax,-0x14(%rbp)
0x0000000000400796 <+41>:   mov     -0x30(%rbp),%rax
0x000000000040079a <+45>:   mov     0x4(%rax),%edx
0x000000000040079d <+48>:   mov     -0x28(%rbp),%rax
0x00000000004007a1 <+52>:   mov     %edx,0x4(%rax)
0x00000000004007a4 <+55>:   mov     -0x30(%rbp),%rax
0x00000000004007a8 <+59>:   mov     -0x14(%rbp),%edx
0x00000000004007ab <+62>:   mov     %edx,0x4(%rax)
0x00000000004007ae <+65>:   mov     -0x28(%rbp),%rdx
0x00000000004007b2 <+69>:   lea     -0x10(%rbp),%rax
0x00000000004007b6 <+73>:   mov     %rdx,%rsi
0x00000000004007b9 <+76>:   mov     %rax,%rdi
0x00000000004007bc <+79>:   callq   0x400530 <strcpy@plt>
0x00000000004007c1 <+84>:   mov     -0x30(%rbp),%rdx
0x00000000004007c5 <+88>:   mov     -0x28(%rbp),%rax
0x00000000004007c9 <+92>:   mov     %rdx,%rsi
0x00000000004007cc <+95>:   mov     %rax,%rdi
0x00000000004007cf <+98>:   callq   0x400530 <strcpy@plt>
---Type <return> to continue, or q <return> to quit---
```

disas swap를해서 swap함수를 어셈블리어로 변경해서 출력한다.