어셈블리 프로그래밍

양경석(gaeng4@gmail.com)

<NASM 문법 사용>

(문제) hello, world!를 출력하는 어셈블리 프로그램을 제작하시오.
hello.asm

```
segment .data
msg:    db    'hello, world!', 10 // line feed에 해당하는 ASCII코드가 10임.

segment .rss

segment .text

global _start

_start
    ;write(1,"hello, world!\n",14);
    mov    eax,    4          // write의 system call번호는 4번임
    mov    ebx,    1          // 기본출력장치인 모니터가 1번임.
    mov    ecx,    msg          //msg라는 segment.data에 라벨되어있는 것을 호출
    mov    edx,    14          //글자수
    int    0x80              //int는 정수형이 아니라 interrupt임
                    //0x80인터럽트 호출. 모든 작업을 중단하고 system call호출.
                    //이 때 호출할 system call은 eax에 저장된 값을 찾음.
    ;exit(0);
    mov    eax,    1          //exit의 system call번호는 1번임.
    mov    ebx,    0
    int    0x80
```

(문제)아래와 같이 출력하는 어셈블리 프로그램을 작성하세요(rabbit.asm)

```
(0)_(0)(0)_(0)
(=^.^=)(*^.^*)
(_m_m_)(_m_m_)

segment.data
msg1:    db    '(0)_(0)(0)_(0)',10
msg1:    db    '(=^.^=)(*^.^*)',10
msg1:    db    '(_m_m_)(_m_m_)',10

segment.text

global _start

_start
    mov    eax,    4
    mov    ebx,    1
    mov    ecx,    msg1
    mov    edx,    15
    int    0x80

    mov    eax,    4
    mov    ebx,    1
    mov    ecx,    msg2
    mov    edx,    15
```

```
    int     0x80

    mov     eax,    4
    mov     ebx,    1
    mov     ecx,    msg3
    mov     edx,    15
    int     0x80

    mov     eax,    1
    mov     ebx,    0
    int     0x80
```

or

```
segment.data
msg:    db    '(0)_(0)(0)_(0)',10,'(=^.^=)(*^.^*)',10,'(_m_m_)(_m_m_)',10

segment. text

global _start

_start

    mov     eax,    4
    mov     ebx,    1
    mov     ecx,    msg
    mov     edx,    45
    int     0x80

    mov     eax,    1
    mov     ebx,    0
    int     0x80
```

```
#>nasm -f elf rabbit.asm //어셈블리어->기계어로 컴파일
#>ld -o rabbit rabbit.o //링크단계
#>./rabbit //실행
```

(문제) 사칙연산 프로그램을 제작하시오.

```
;;calc_a.asm
;;
;;compile
;;#>nasm -f elf calc_a.asm
;;
;;link
;;#>gcc -o math math.o asm_io.o
;;

%include "asm_io.inc"

segment.data
number1:    dd    100
number2:    dd    20

segment. text

global main

main
```

```
    ;;add
    mov     eax,    [number1]
    mov     ebx,    [number2]
    add     eax,    ebx
    int     0x80
    call    print_int
    call    print_nl

    ;;sub
    mov     eax,    [number1]
    mov     ebx,    [number2]
    sub     eax,    ebx
    int     0x80
    call    print_int
    call    print_nl

    ;;mul
    mov     ax,     [number1]
    mul     word[number2]
    call    print_int
    call    print_nl

    ;;imul #1
    mov     eax,    [number1]
    imul    eax,    [number2]
    call    print_int
    call    print_nl

    ;;imul #2
    imul    eax,    [number1],    20
    call    print_int
    call    print_nl

    ;;div
    cdq
    mov     eax,    [number1]
    div     dword[number2]
    call    print_int
    call    print_nl

    ;;idiv
    cdq
    mov     eax,    [number1]
    idiv    dword[number2]
    call    print_int
    call    print_nl

    ;;나머지
    mov     eax,    edx
    call    print_int

    ;;exit(0)
    mov     eax,    1
    mov     ebx,    0
```

(문제) 임의의 세 정수의 합과 평균을 출력하는 프로그램을 작성하시오 calc2a.asm

```
%include "asm_io.inc"

segment.data
```

```
number1:    dd    100
number2:    dd    90
number3:    dd    80
;;segment.bss
segment.text

global main

main
    mov     eax,    [number1] ;;eax=number1(100)
    add     eax,    [number2] ;;eax=eax(number1) + number2
    add     eax,    [number3] ;;eax=eax(number1+number2) + number3
    call    print_int
    call    print_nl

    mov     ebx,    3    ;;ebx=3
    cdq
    div     ebx         ;;eax=eax(number1+number2+number3) / ebx(3)
    call    print_int
    call    print_nl

    mov     eax,    1    ;;exit(0)
    mov     ebx,    0
    int     0x80
```

(문제) 3개의 정수를 입력받아서 세 정수의 합과 평균을 출력하기.

```
%include "asm_io.inc

segment .data
input_msg:     dd     "input number:",0 ;;마지막에 0을 추가해 주어야 됨
total_msg:     dd     "sum: "          ;;문자열은 마지막에 0을 만날때까지 출력하므로
avg_msg:     dd     "average: "       ;;0을 주지 않으면 마지막에 쓰레가값이 입력됨.

segment .bss ;;띄어쓰기 주의

num1     resd     1
num2     resd     1
num3     resd     1

sum:     resd     1  ;;tot뒤에 :를 붙여줘도 되고 안써줘도 됨.
avg:     resd     1

segment .text

global main

main
    mov     eax,     input_msg
    call    print_string
    call    read_int
    mov     dword[num1],     eax

    mov     eax,     input_msg
    call    print_string
    call    read_int
    mov     dword[num2],     eax

    mov     eax,     input_msg
    call    print_string
```

```
    call    read_int
    mov     dword[num3],    eax

    mov     eax,    [num1]
    add     eax,    [num2]
    add     eax,    [num3]
    mov     dword[sum],     eax

    mov     eax,    total_msg
    call    print_string
    mov     eax,    [sum]
    call    print_nl

    mov     ebx,    3
    div     ebx
    mov     [avg],     eax

    mov     eax,    avg_msg
    call    print_string
    mov     eax,    [avg]
    call    print_nl
```

(문제) 섭씨->화씨로 바꾸어 주는 프로그램(tem.asm)
<섭씨 = 9*섭씨/5+32>

```
%include "asm_io_inc"

segment .data
sup_input_msg:     dd     "input 'c:",0
hwa_input_msg:     dd     "input 'F:",0
sup_msg:     dd     "'c: ",0
hwa_msg:     dd     "'F: ",0

segment .bss
sup:     resd     1
hwa:     resd     1

segment .text

global     main

main
    mov     eax,    sup_input_msg
    call    print_string
    call    read_int
    mov     [sup],     eax

    mov     eax,    [sup]
    imul    eax,    [sup],     9

    cdq
    mov     ebx,    5
    div     ebx
    add     eax,    32
    mov     [hwa],     eax

    mov     eax,    hwa_msg
    call    print_string
    mov     eax,    [hwa]
```

```
    call    print_int
    call    print_nl

    mov     eax,    hwa_input_msg
    call    print_string
    call    read_int
    mov     [hwa],    eax

    mov     eax,    [hwa]
    sub     eax,    32

    cdq
    mov     ebx,    9
    div     ebx

    mov     [sup],    eax
    imul    eax,    [sup],    5
    mov     [sup],    eax

    mov     eax,    sup_msg
    call    print_string
    mov     eax,    [sup]
    call    print_int
    call    print_nl

    mov     eax,    1    ;;exit(0)
    mov     ebx,    0
    int     0x80
```

(문제)
키보드로부터 입력받은 2개의 정수 중에서 더 큰 수를 출력하는 프로그램을 작성하세요.<a12.asm>

```
segment .data
segment .bss
a     resd    1
b     resd    1

segment .text

global main
main
    call    read_int
    mov     [a],    eax
    call    print_string
    call    read_int
    mov     [b],    eax

    cmp     eax,    [a]
    jng     large_right

large_left:
    mov     eax,    [b]
    call    print_int
    jmp     end

large_right:
    mov     eax,    [a]
    call    print_int
```

```
end:
```

(문제)
키보드로부터 입력받은 3개의 정수 중에서 더 큰 수를 출력하는 프로그램을 작성하세요.<a13.asm>

```
segment .bss
a     resd     1
b     resd     1
c     resd     1

segment .text
global main

main
    ;input data
    call     read_int
    mov      [a],     eax
    call     read_int
    mov      [b],     eax
    call     read_int
    mov      [c],     eax

    ;compare
    mov      eax,     [a]
    cmp      eax,     [b]
    jg     a_b
    jmp      b_a

a_b:     ; i) a > b
    mov      eax,     [c]
    cmp      eax,     [a]
    jg     c_a
    jmp      a_c

    ;ii)
    c_a:     ; c > a > b  MAX: C
        mov      eax,     [c]
        call     print_int
        jmp      end

    a_c:     ; a > c,b    MAX: A
        mov      eax,     [a]
        call     print_int
        jmp      end


b_a:     ; i) b > a
    mov      eax,     [c]
    cmp      eax,     [b]
    jg     c_b
    jmp      b_c

    ;ii)
    c_b:     ; c > b > a   MAX: C
        mov      eax,     [c]
        call     print_int
        jmp      end

    b_c:     ; b> a,c   MAX: B
        mov      eax,     [b]
```

```
        call    print_int
        jmp     end
end:
```

(문제) 구구단 어셈블리어로 만들기 // agugu.asm

```
%include "asm_io.inc"
segment .data
dan     dd     2
soo     dd     1
gop     dd     "x",0
nun     dd     "=",0

segment .bss
segment .text
global main
main

dan_loop: ( dan*1, dan*2 ... dan*9까지 반복)
    mov     eax,     [dan]
    call    print_int
    mov     eax,     [gop]     //** print_char쓸때는 [gop] 이렇게 해야 하고
    call    print_char     //** print_string쓸때는 gop 이렇게 써야함.
    mov     eax,     [soo]     //** print_char는 한글자만 출력됨.
    call    print_int     //** print_string은 여러글자 출력됨. 끝에 ,0 까먹지 말기!
    mov     eax,     [nun]
    call    print_char
                ; o x o = 까지 출력
    mov     eax,     dword[dan]
    imul    eax,     dword[soo]
    call    print_int
    call    print_nl
    inc     dword[soo]
    cmp     dword[soo],     9
    jng     dan_loop
    jg      soo_loop

soo_loop: (soo =1로 초기화, dan ++)
    mov     dword[soo],     1
    inc     dword[dan]
    mov     eax,     [dan]
    imul    eax,     dword[soo]
    call    print_nl
    cmp     dword[dan],     9
    jng     dan_loop
    jg      end
end:
```

(문제) 키보드에서 읽어들인 문자열을 역순으로 출력하는 어셈블리 프로그램을 작성하세요<rev.asm>
** read, write 시스템콜 사용

```
read(0, void *buf, size_t count);     // read의 system call 번호는 3번임
write(1, void *buf, size_t count);     // write의 system call번호는 4번임

input: korea
output: aerok
```

```
segment .data
segment .bss

input_msg     resd     30
ptr           resd     30
cnt           resd     1

segment .text

global main

main
    mov     eax,     3
    mov     ebx,     0
    mov     ecx,     input_msg
    mov     edx,     30
    int     0x80

    mov     dword[ptr],     input_msg
    add     dword[ptr],     29
    mov     dword[cnt],     29

for:
    mov     eax,     4
    mov     ebx,     1
    mov     ecx,     [ptr]
    mov     edx,     1
    int     0x80

    cmp     dword[cnt],     0
    jg      minus_cnt
    jng     end

minus_cnt:
    dec     dword[ptr]
    dec     dword[cnt]
    jmp     for
end:
```

;뒤부터 주소값을 빼주면서 출력하는 방법, 다른 공간에 저장후 한번에 출력하는 방법 활용 가능.

1. 주소 지정 [주소+scale*4]
2. Lea [주소+scale]

<rev2.asm>

```
segment .data
segment .bss

data     resb     1024
rdata     resb     1024
size     resb     1

segment .text

global main

main
    mov     eax,     3
    mov     ebx,     0
    mov     ecx,     data
```

```
    mov     edx,    1024
    int     0x80

    mov     eax,    0

cond:
    mov     ebx,    [data+eax*1]
    cmp     ebx,    0x0a
    je      end

body:
    inc     eax
    mov     dword[size],    eax
    jmp     cond

end:
    sub     dword[size],    1

    mov     eax,    4
    mov     ebx,    1
    mov     ecx,    data        ;mov    ecx,    [size]
    add     ecx,    [size]      ;lea    ecx,    [data+ecx] 이렇게 표현할 수 있다.
    mov     edx,    1
    int     0x80

    cmp     dword[size],    0
    jne     end
    je      exit
exit:
```

(문제) 키보드에서 소문자 하나를 읽어들여 대문자로 변환하는 프로그램을 작성하시오. char1.asm

```
%include "asm_io.inc"

segment .data
segment .bss

input   resb    1

segment .text

global main

main
    mov     eax,    3
    mov     ebx,    0
    mov     ecx,    input
    mov     edx,    1
    int     0x80

    sub     dword[input],   0x20

    mov     eax,    4
    mov     ebx,    1
    mov     ecx,    input
    mov     edx,    1
    int     0x80
```

(문제) 키보드에서 문자 하나를 읽어들여 소문자이면 대문자로, 대문자면 소문자로 변환(char2.asm)
a->A, A->a

```
input    resd    1
global main
main
    mov     eax,    3
    mov     ebx,    0
    mov     ecx,    input
    mov     edx,    1
    int     0x80

    cmp     dword[input],    0x60
    jg     smj
    jng     dmj
smj:
    sub     dword[input],    0x20
    jmp     prn
dmj:
    add     dword[input],    0x20
    jmp     prn
prn:
    mov     eax,    4
    mov     ebx,    1
    mov     ecx,    input
    mov     edx,    1
    int     0x80
```

(문제) 키보드에서 문자열을 읽어들여 소문자이면 대문자로, 대문자면 소문자로 변환(char2.asm)

```
input    resb    1024
cnt    resb
global main
main
    mov     eax,    3
    mov     ebx,    0
    mov     ecx,    input
    mov     edx,    1
    int     0x80                ;키보드에서 입력받기

    mov     byte[cnt],    0         ;cnt=0 으로 초기화

output_loop
    mov     eax,    [cnt]             ;eax에 cnt값 입력.
    cmp     byte[input+eax],    10    ;입력값이 10(엔터값)일 경우
    je     exit                ;이면 exit로 이동.
                        ;엔터값 아닌 경우
    cmp     byte[input],    0x60         ;대문자와 소문자의 경계값이 0x60으로 비교
    jg     smj                 ;0x60보다 큰 경우(소문자)->smj로 이동
    jng     dmj                  ;0x60보다 작은 경우(대문자)->dmj로 이동
smj:
    mov     eax,    [cnt]             ;cnt값을 eax로 이동
    sub     byte[input+eax],    0x20    ;input(시작주소값)에서 현재cnt만큼 증가해준 주소의 값에서 0x20 빼줌(대
문자로 변환)
    jmp     prn                 ;출력단계로 이동
dmj:
    mov     eax,    [cnt]
    add     byte[input+eax],    0x20    ;input(시작주소값)에서 현재cnt만큼 증가해준 주소의 값에서 0x20 더해줌
(소대문자로 변환)
```

```
    jmp     prn                 ;출력단계로 이동
prn:
    mov     eax,    4
    mov     ebx,    1
    mov     ecx,    input
    add     ecx,    [cnt]           ;input(시작주소값)에서 현재cnt만큼 증가해준 주소의 값을 한글자씩 출력해줌
    mov     edx,    1
    int     0x80

    inc     byte[cnt]           ;cnt값을 1개 증가시켜준 후
    jmp     output_loop             ;output_loop로 이동시킴.
exit:
```

(문제)키보드에서 파일명을 입력받아 파일의 내용을 화면에 출력하는 어셈블리 프로그램 작성

시스템콜 사용(read,write)
리눅스 시스템의 cat명령어와 동일하게 작동하면 됨
ex) mycat /etc/passwd

소스코드: bmycat.asm
실행: ./bmycat /etc/passwd

<참고한 홈페이지>
*argv[1]를 ebx에 저장하는 방법
http://stackoverflow.com/questions/7854706/reading-filename-from-argv-via-x86-assembly

*시스템콜 레퍼런스 http://www.joinc.co.kr/modules/moniwiki/wiki.php/Site/Assembly/Documents/article_linux_systemcall_quick_reference

```
 xfs:x:43:43:X Font Server:/etc/X11/fs:/sbin/nologin
gdm:x:42:42::/var/gdm:/sbin/nologin
vboxadd:x:101:1::/var/run/vboxadd:/bin/false
oprofile:x:16:16:Special user account to be used by OProfile:/home/oprofile:/sbin/nologin
user01:x:500:500::/home/user01:/bin/bash
```

이런식으로 출력되도록 프로그램 작성.

#>man 2 open

```
OPEN(2)                 System calls                OPEN(2)

NAME
       open, creat - open and possibly create a file or device

       #include <sys/types.h>
       #include <sys/stat.h>
       #include <fcntl.h>

       int open(const char *pathname, int flags);
       int open(const char *pathname, int flags, mode_t mode);
       int creat(const char *pathname, mode_t mode);
```

flag: O_RDONLY
mode: 755

c언어로 짠 것: mycat.c
argv[1]을 이용해서 파일 입력은 가능
read(0,buffer,100)을 이용해서 파일명 입력받은 후 출력은 안됨

scanf()로 입력받은 후에는 출력 됨.
==>argv[1]을 이용하여 파일입력.

어셈블리로 짠 것: bmycat.asm(파일명 입력) , ./bmycat /etc/passwd(argv[1]을 이용해서도 해결)

**시스템콜 번호
open: 5
write: 4
read: 3
close: 6

	open	read	write	close
input	mov eax, 5 mov ebx, file name mov ecx, file access mode mov edx, file permissions	eax 3 ebx:filedescriptor ecx: pointer to input buffer edx: buffer size	eax: 4 ebx: file descriptor ebx: pointer to output buffer edx: buffer size	eax: 6 ebx: file descriptor
return	eax: file descriptor	eax: number of bytes read	eax: number of bytes written	eax:--
error	eax: error code	eax: error code	eax: error code	eax: error code

cf. exit(0) mov eax, 1
mov ebx, 0
int 0x80

```
%include "asm_io.inc"
;open->read->write->close

segment .data
file_name1      db      'hello.txt',0
file_name2      db      '/etc/passwd',0
fd_msg          db      'file descriptor: ',0
fd_err_msg      db      'file open error!',10,0

segment .bss
fd              resd    1
buffer          resb    65535
argv            resb    30

segment .text

global main

main

;open

mov eax,5

pop     ebx                 ;&ret
pop     ebx                 ;&argc
pop     ebx                 ;&argv

mov     ebx,    [ebx+4]         ;argv[1]
;push ebx

mov     ecx, 2
```

스택에 저장되어있는 인자값을 pop을통해 ebx에 저장.
첫번째 pop: ret가 저장되어 있는 곳의 주소
두번째 pop: argc가 저장되어 있는 곳의 주소
세번째 pop: argv가 저장되어 있는 곳의 주소

세번재 pop을 통해서 argv[0]의 주소를 구해서
ebx에 저장했으므로 ebx+4가 argv[1]의 주소.
따라서 argv[1]의 값은 [ebx+4]가 되므로 이를
open의 인자값으로 위치시켜줌.

```
mov     edx, 0700
int     0x80

mov     [fd],   eax
mov     eax,    fd_msg
call    print_string
mov     eax,    [fd]
call    print_int
call    print_nl
call    print_nl
cmp   dword[fd],   1
jl       fd_err

;read
mov     eax,    3
mov     ebx,    [fd]
mov     ecx,    buffer
mov     edx,    65535
int     0x80

;write
mov     eax,    4
mov     ebx,    1
mov     ecx,    buffer
mov     edx,    65535
int     0x80

;close
mov     eax,    6
mov     ebx,    [fd]
int     0x80

exit:
mov     eax,    1
mov     ebx,    0
int     0x80

fd_err:
mov    eax,     fd_err_msg
call     print_string
jmp     exit
```

fd가 1보다 작으면 fd_err로 이동하여 에러메시지 출력하고 종료하도록함.
file descriptor가 1보다 작다는 것은 제대로 open되지 않았다는 의미임.
제대로 open되면 1이상값이 나옴.

fd가 1보다 작다면 에러메시지 출력후 종료

(문제) mkdir() 시스템콜을 사용하여 현재 디렉터리 아래에 "mydir"이라는 디렉터리를 생성하는 어셈블리 프로그램 작성
소스코드: <cmymkdir.asm>
실행: ./cmymkdir mydir

system call번호 (39)

MKDIR(2) Linux Programmer's Manual MKDIR(2)

#>man 2 mkdir

NAME
    mkdir - create a directory

SYNOPSIS
    #include <sys/stat.h>

```
#include <sys/types.h>
#include <fcntl.h>
#include <unistd.h>

int mkdir(const char *pathname, mode_t mode);
```

%include "asm_io.inc"
 	
segment .data
;dirname db 'abc123',0
 	
segment .bss
 	
segment .text
 	
global main
 	
main
 	
mov eax, 39
;mov ebx, dirname ;make dirname
pop ebx
pop ebx
pop ebx
mov ebx, [ebx+4]
mov ecx, 0755
int 0x80
 	
mov eax, 1
mov ebx, 0
int 0x80
	ebx에 dirname을 mov하면 우측 문자열대로 디렉토리 생성
	argv[1]을 이용할 경우에는 필요없으므로 주석처리
	
	mkdir()의 시스템콜 번호는 39번
	
	ret가 저장된 곳의 주소 pop 하여 ebx에 저장
	argc가 저장된 곳의 주소 pop 하여 ebx에 저장
	argv[0]이 저장된 곳의 주소 pop 하여 ebx에 저장
	ebx+4가 argv[1]의 주소가 되므로 그 주소에 저장된 값인
	argc[1], 즉 [ebp+4]를 ebx에 저장하여 mkdir의 첫번째 인자로 채워줌
	
	mkdir의 두번째 인자는 mode임(생성될 디렉토리의 권한)-0755로 설정해주자.
	
	exit(0);