Assembly Language and System Programming
Homework#2
0016046 蔡佩珊

> Multiplication implement

```
multiplication:
                mov     eax,0       ; for product
                mov     ecx,n       ; m*n=m+...+m 共 n 次, n 存於 loop counter ecx
                cmp     m,0
                jz      mulout
                cmp     n,0         ; n<0 則 neg ecx  為求 counter 為正
                jz      mulout
                jg      mul1
                neg     ecx
mul1:           add     eax,m
                loop    mul1
                cmp     n,0         ; n<0 則 neg eax
                jge     mulout
                neg     eax
mulout:         call    WriteInt
                exit
```

> Division implement

```
division:
                cmp     n,0         ; check if divided by 0
                jz      div0
                mov     eax,0       ; for quotient
                mov     ebx,0       ; for accumulator
                cmp     m,0         ; 4 types division
                jg      dposem
                jmp     dnegm
dposem:         cmp     n,0
                jg      d1          ; + / +
                jmp     d2          ; + / -
dnegm:          cmp     n,0
                jg      d3          ; - / +
                jmp     d4          ; - / -
d1:             add     ebx,n       ; + / + = + ... +
                inc     eax         ; inc quotient
                cmp     ebx,m
                jbe     d1          ; when exit loop, ebx>m, eax=quotient+1
                dec     eax
                jmp     dout        ; done, jmp for output
```

```
d2:     sub   ebx,n     ; + / - = - ... +
        dec   eax       ; dec quotient
        cmp   ebx,m
        jbe   d2        ; when exit loop, ebx>m, eax=quotient-1
        inc   eax
        jmp   dout      ; done, jmp for output
d3:     sub   ebc,n     ; - / + = - ... -
        dec   eax       ; dec quotient
        cmp   ebx,m
        jge   d3        ; when exit loop, ebx<m, eax=quotient-1
        inc   eax
        jmp   dout      ; done, jmp for output
d4:     add   ebx,n     ; - / - = + ... -
        inc   eax       ; inc quotient
        cmp   ebx,m
        jge   d4        ; when exit loop, ebx<m, eax=quotient+1
        dec   eax
        jmp   dout      ; done, jmp for output
dout:   call  WriteInt  ; output
        exit
```

> Modulo implement

```
modulo:
        cmp   n,0       ; check if divided by 0
        jz    div0
        mov   eax,n     ; for remainder
        mov   ebx,0     ; for accumulator
        cmp   m,0       ; 4 types modulo
        jg    mposem
        jmp   mnegm
mposem: cmp   n,0
        jg    m1        ; + % +
        jmp   m2        ; + % -
mnegm:  cmp   n,0
        jg    m3        ; - % +
        jmp   m4        ; - % -
m1:     add   ebx,n     ; + / + = + ... +
        cmp   ebx,m
        jbe   m1        ; when exit loop, ebx>m
        sub   ebx,m     ; m+(ebx-m)=ebx
        sub   eax,ebx   ; remainder=n-(ebx-m)
        jmp   mout      ; done, jmp for output
m2:     sub   ebx,n     ; + / - = - ... +
        cmp   ebx,m
        jbe   m2        ; when exit loop, ebx>m
```

```
        sub     ebx,m       ; m+(ebx-m)=ebx
        neg     eax         ; n<0, neg eax
        sub     eax,ebx     ; remainder=-n-(ebx-m)
        jmp     mout        ; done, jmp for output
m3:     sub     ebx,n       ; - / + = - ... -
        cmp     ebx,m
        jge     m3          ; when exit loop, ebx<m
        sub     ebx,m       ; m+(ebx-m)=ebx
        neg     eax         ; n>0, neg eax
        sub     eax,ebx     ; remainder=-n-(ebx-m)
        jmp     mout        ; done, jmp for output
m4:     add     ebx,n       ; - / - = + ... -
        cmp     ebx,m
        jge     m4          ; when exit loop, ebx<m
        sub     ebx,m       ; m+(ebx-m)=ebx
        sub     eax,ebx     ; remainder=n-(ebx-m)
        jmp     mout        ; done, jmp for output
mout:   call    WriteInt    ; output
        exit
```