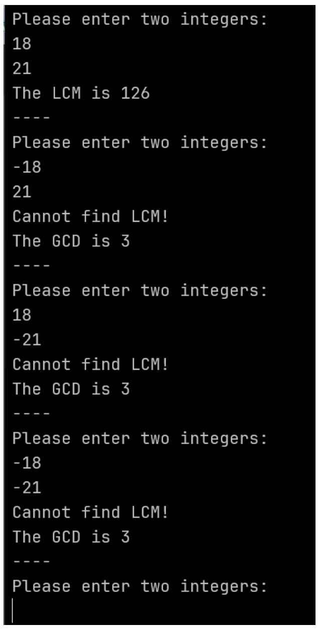
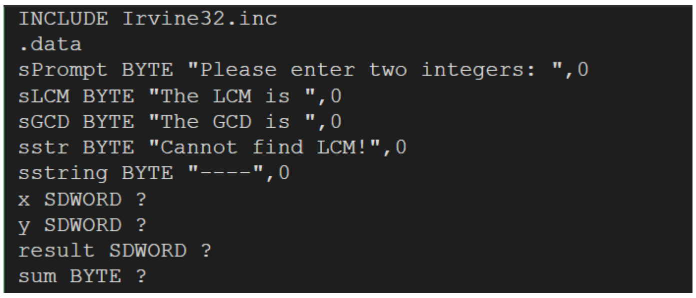
HW2

輸出結果



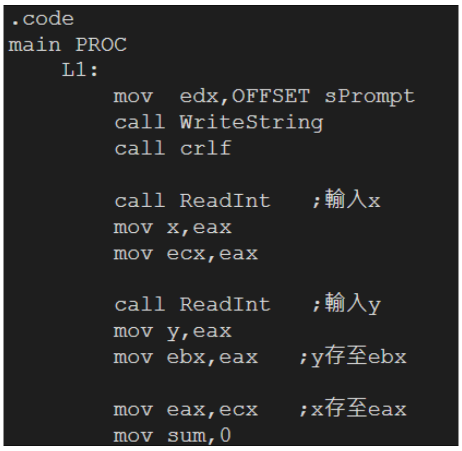
```
Please enter two integers:
18
21
The LCM is 126
----
Please enter two integers:
-18
21
Cannot find LCM!
The GCD is 3
----
Please enter two integers:
18
-21
Cannot find LCM!
The GCD is 3
----
Please enter two integers:
-18
-21
Cannot find LCM!
The GCD is 3
----
Please enter two integers:
```

Data 設定



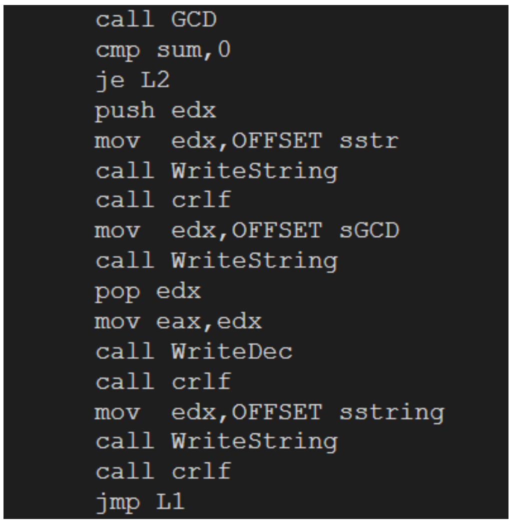
```
INCLUDE Irvine32.inc
.data
sPrompt BYTE "Please enter two integers: ",0
sLCM BYTE "The LCM is ",0
sGCD BYTE "The GCD is ",0
sstr BYTE "Cannot find LCM!",0
sstring BYTE "----",0
x SDWORD ?
y SDWORD ?
result SDWORD ?
sum BYTE ?
```

主函式



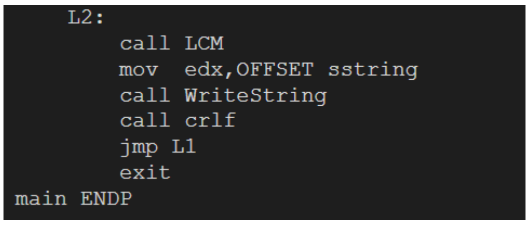
```
.code
main PROC
    L1:
        mov  edx,OFFSET sPrompt
        call WriteString
        call crlf

        call ReadInt    ;輸入x
        mov x,eax
        mov ecx,eax

        call ReadInt    ;輸入y
        mov y,eax
        mov ebx,eax    ;y存至ebx

        mov eax,ecx    ;x存至eax
        mov sum,0
```

輸入 x,y 的值，以及將 sum 設為 0。



```
    call GCD
    cmp sum,0
    je L2
    push edx
    mov  edx,OFFSET sstr
    call WriteString
    call crlf
    mov  edx,OFFSET sGCD
    call WriteString
    pop edx
    mov eax,edx
    call WriteDec
    call crlf
    mov  edx,OFFSET sstring
    call WriteString
    call crlf
    jmp L1
```

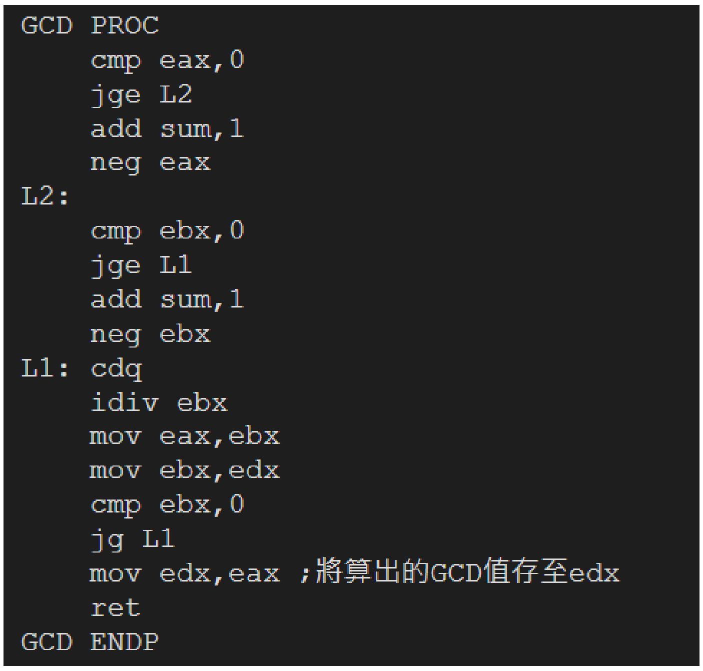
呼叫 GCD 函式，如果 sum 等於 0 就跳至 L2，否則就把算出的 GCD 值印出並跳回 L1。



```
     L2:
         call LCM
         mov  edx,OFFSET sstring
         call WriteString
         call crlf
         jmp L1
         exit
main ENDP
```

呼叫 LCM，並跳回至 L1。

GCD 函式

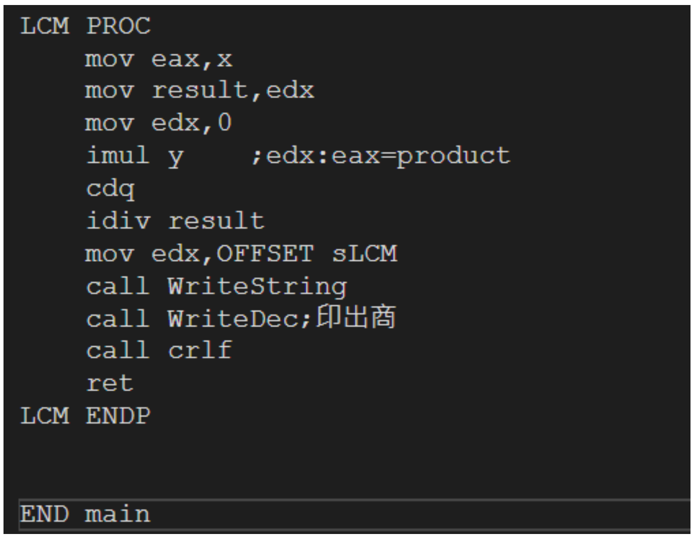


```
GCD PROC
    cmp eax,0
    jge L2
    add sum,1
    neg eax
L2:
    cmp ebx,0
    jge L1
    add sum,1
    neg ebx
L1: cdq
    idiv ebx
    mov eax,ebx
    mov ebx,edx
    cmp ebx,0
    jg L1
    mov edx,eax ;將算出的GCD值存至edx
    ret
GCD ENDP
```

判斷 eax 和 ebx 是否為正數，如果是的話就跳至 L1 迴圈，否則就將他轉成正數並將 sum+1。

L1 迴圈是進行除法的部分，除完之後判斷 ebx 是否等於 0，若等於 0 則將 eax 的值存入 edx 並返回主函式。

LCM 函式



```
LCM PROC
    mov eax,x
    mov result,edx
    mov edx,0
    imul y     ;edx:eax=product
    cdq
    idiv result
    mov edx,OFFSET sLCM
    call WriteString
    call WriteDec;印出商
    call crlf
    ret
LCM ENDP


END main
```

進行 x*y 的動作並除以前面

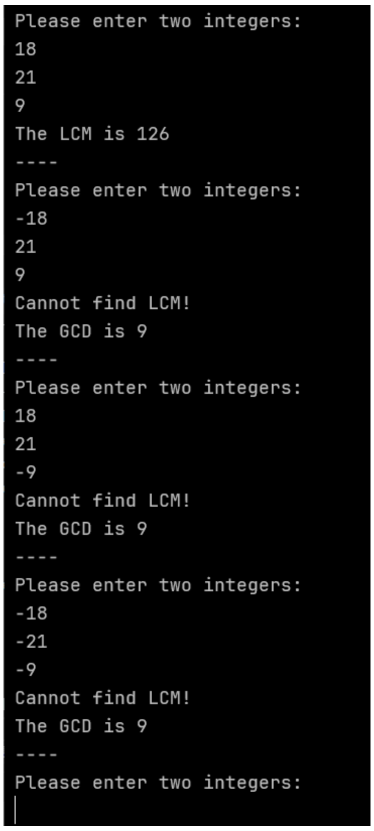
算出的 gcd 值，要先把 edx

存至 result 是因為在 cdq

時會對 edx 進行擴展。

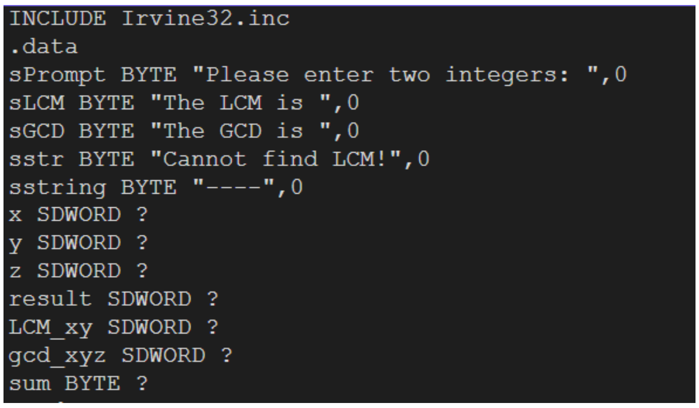
Bonus!!!!!

輸出結果



```
Please enter two integers:
18
21
9
The LCM is 126
----
Please enter two integers:
-18
21
9
Cannot find LCM!
The GCD is 9
----
Please enter two integers:
18
21
-9
Cannot find LCM!
The GCD is 9
----
Please enter two integers:
-18
-21
-9
Cannot find LCM!
The GCD is 9
----
Please enter two integers:
```

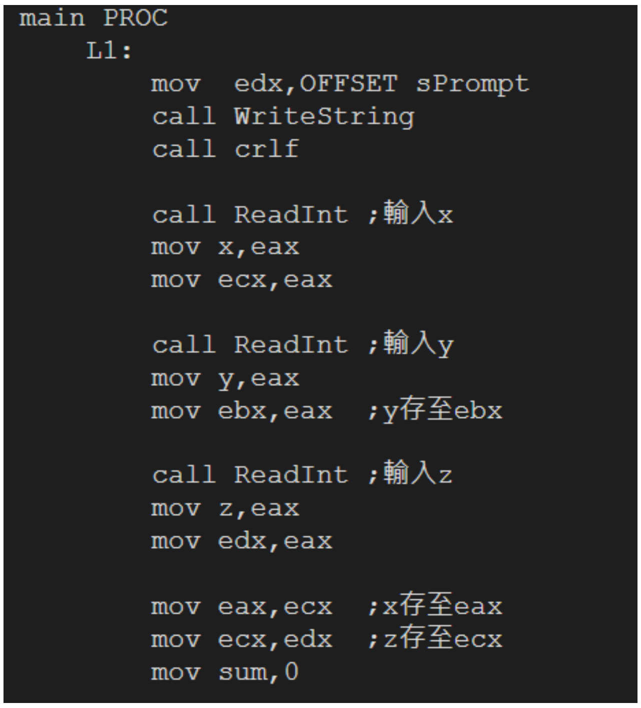
Data 值



```
INCLUDE Irvine32.inc
.data
sPrompt BYTE "Please enter two integers: ",0
sLCM BYTE "The LCM is ",0
sGCD BYTE "The GCD is ",0
sstr BYTE "Cannot find LCM!",0
sstring BYTE "----",0
x SDWORD ?
y SDWORD ?
z SDWORD ?
result SDWORD ?
LCM_xy SDWORD ?
gcd_xyz SDWORD ?
sum BYTE ?
```

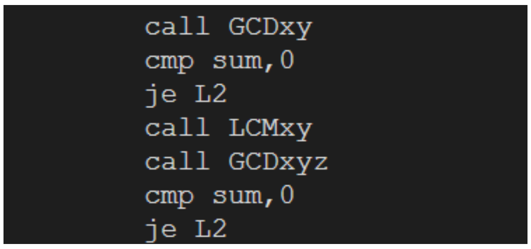
和前面差不多，多了幾個變數拿來儲存。

主函式



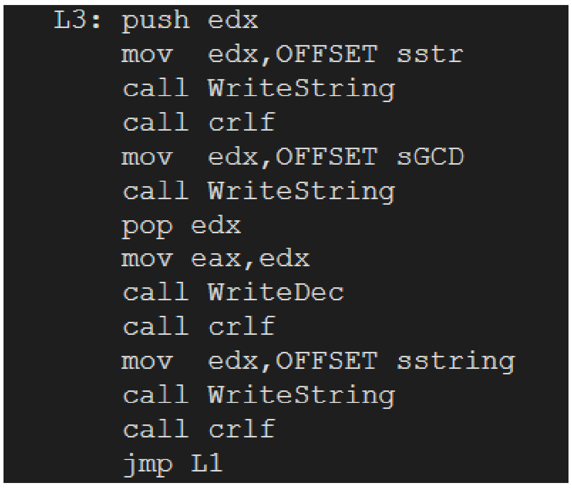
```
main PROC
    L1:
        mov  edx,OFFSET sPrompt
        call WriteString
        call crlf

        call ReadInt ;輸入x
        mov x,eax
        mov ecx,eax

        call ReadInt ;輸入y
        mov y,eax
        mov ebx,eax  ;y存至ebx

        call ReadInt ;輸入z
        mov z,eax
        mov edx,eax

        mov eax,ecx  ;x存至eax
        mov ecx,edx  ;z存至ecx
        mov sum,0
```

輸入 x,y,z 的值，以及將 sum 設為 0。



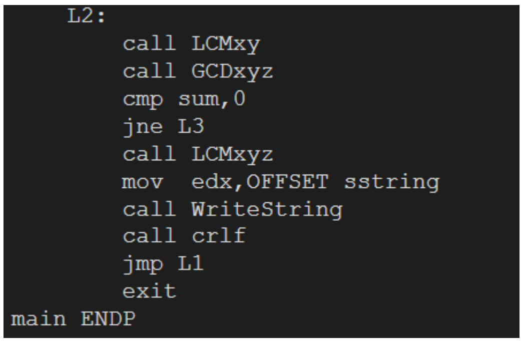
```
        call GCDxy
        cmp sum,0
        je L2
        call LCMxy
        call GCDxyz
        cmp sum,0
        je L2
```

呼叫 GCDxy，如果 sum 等於 0(代表 x,y 都為正數)則跳至 L2，否則即呼叫 LCMxy 再呼叫 GCDxyz，再判斷 sum 是否等於 0(代表 x,y,z 都為正數)，則跳至 L2。



```
L3: push edx
    mov  edx,OFFSET sstr
    call WriteString
    call crlf
    mov  edx,OFFSET sGCD
    call WriteString
    pop edx
    mov eax,edx
    call WriteDec
    call crlf
    mov  edx,OFFSET sstring
    call WriteString
    call crlf
    jmp L1
```

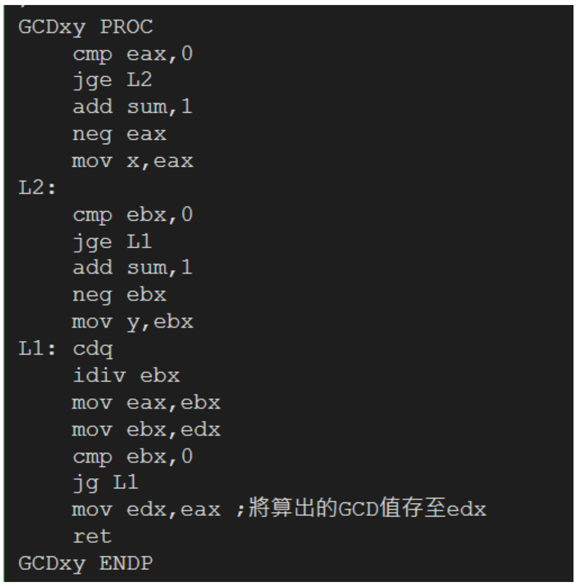
把算出的 GCD 值印出並跳回 L1。



```
    L2:
        call LCMxy
        call GCDxyz
        cmp sum,0
        jne L3
        call LCMxyz
        mov  edx,OFFSET sstring
        call WriteString
        call crlf
        jmp L1
        exit
main ENDP
```

呼叫 LCMxy 再呼叫 GCDxyz，並判斷 sum 是否等於 0，如果不等於 0 則代表 z 為負數，所以要跳回 L2 印出 GCD 值，如果等於 0 則呼叫 LCMxyz 最後跳回 L1。

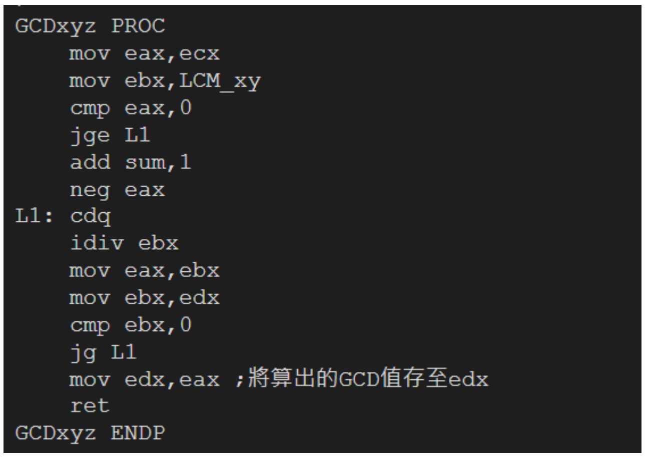
GCDxy 函式



```
GCDxy PROC
    cmp eax,0
    jge L2
    add sum,1
    neg eax
    mov x,eax
L2:
    cmp ebx,0
    jge L1
    add sum,1
    neg ebx
    mov y,ebx
L1: cdq
    idiv ebx
    mov eax,ebx
    mov ebx,edx
    cmp ebx,0
    jg L1
    mov edx,eax ;將算出的GCD值存至edx
    ret
GCDxy ENDP
```

和上面的相同。

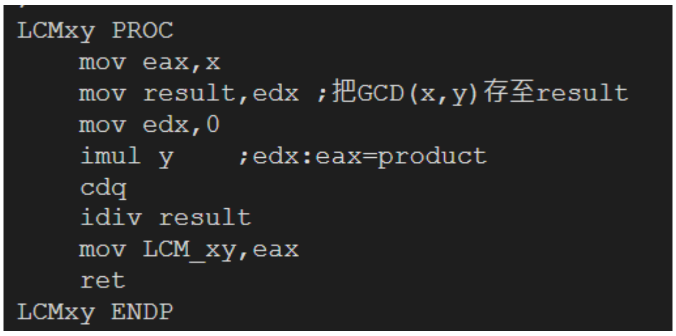
GCDxyz 函式



```
GCDxyz PROC
    mov eax,ecx
    mov ebx,LCM_xy
    cmp eax,0
    jge L1
    add sum,1
    neg eax
L1: cdq
    idiv ebx
    mov eax,ebx
    mov ebx,edx
    cmp ebx,0
    jg L1
    mov edx,eax ;將算出的GCD值存至edx
    ret
GCDxyz ENDP
```

和 GCDxy 差不多，只是在這裡的 eax 的值是存入 LCM(x,y)的值。

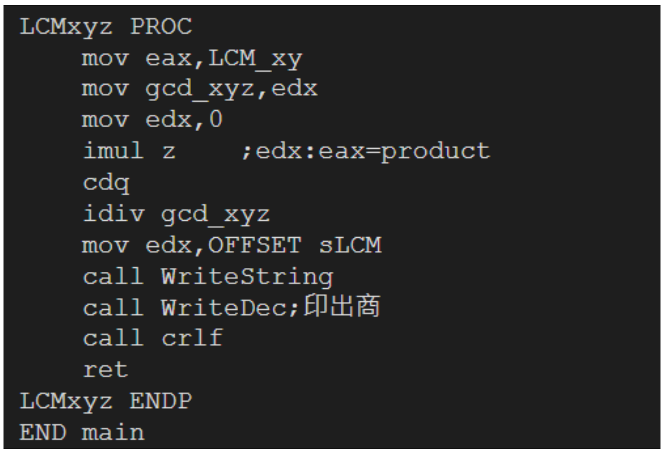
LCMxy 函式



```
LCMxy PROC
    mov eax,x
    mov result,edx ;把GCD(x,y)存至result
    mov edx,0
    imul y    ;edx:eax=product
    cdq
    idiv result
    mov LCM_xy,eax
    ret
LCMxy ENDP
```

和上面的差不多，只是最後把算出的值存至 LCM_xy 這個變數裡。

LCMxyz 函式



```
LCMxyz PROC
    mov eax,LCM_xy
    mov gcd_xyz,edx
    mov edx,0
    imul z    ;edx:eax=product
    cdq
    idiv gcd_xyz
    mov edx,OFFSET sLCM
    call WriteString
    call WriteDec;印出商
    call crlf
    ret
LCMxyz ENDP
END main
```

計算 LCM(x,y)*z 的值並除以 z，最後印出最終結果。