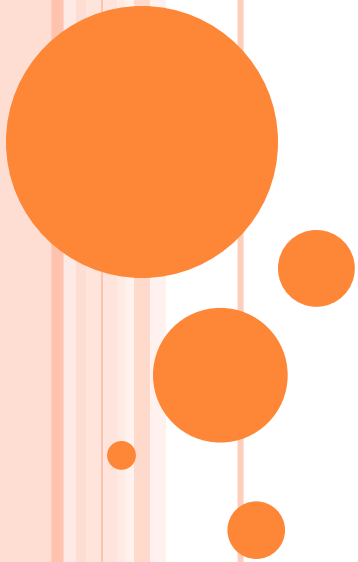


----- LAB 1 -----

MIPS ASSEMBLY PROGRAMMING

洪銘佑

2018/10/8



實驗內容

- 實驗目的
- MIPS Assembly
- MARS (MIPS Assembler and Runtime Simulator)
- 課堂實作
- 課後作業



實驗目的

- 熟悉 MIPS assembly programming
- 認識 MIPS編譯工具MARS
- 實作 MIPS assembly programming



MIPS ASSEMBLY

- 程式由data section與text section構成，data section包含global data而text section包含組語指令。

```
.data
input : .asciiz "pls enter a number : "
output : .asciiz "\nearest prime number is: "
```

Data section

```
.text
.globl main
main :
```

Text section

```
    #read the first number
    li $v0, 4           # load syscall code (4 : print string)
    la $a0, input       # load address of string to be printed into $a0
    syscall             # print str
```

```
    .
    .
    .
```

```
down:
    add $t1, $zero, $t2

loop_ob :
    add $t3, $zero, $t0 #reset $t3
    sub $t1, $t1, $t0 # $t0 down
    j loop_i
```



INSTRUCTION

Basic Instructions	Extended (pseudo) Instructions	Directives	Syscalls	Exceptions	Macros
abs.d \$f2,\$f4	Floating point absolute value double precision : Set \$f2 to absolute value of \$f4, double precision				
abs.s \$f0,\$f1	Floating point absolute value single precision : Set \$f0 to absolute value of \$f1, single precision				
add \$t1,\$t2,\$t3	Addition with overflow : set \$t1 to (\$t2 plus \$t3)				
add.d \$f2,\$f4,\$f6	Floating point addition double precision : Set \$f2 to double-precision floating point value of \$f4 plus \$f6				
add.s \$f0,\$f1,\$f3	Floating point addition single precision : Set \$f0 to single-precision floating point value of \$f1 plus \$f3				
addi \$t1,\$t2,-100	Addition immediate with overflow : set \$t1 to (\$t2 plus signed 16-bit immediate)				
addiu \$t1,\$t2,-100	Addition immediate unsigned without overflow : set \$t1 to (\$t2 plus signed 16-bit immediate), no overflow				
addu \$t1,\$t2,\$t3	Addition unsigned without overflow : set \$t1 to (\$t2 plus \$t3), no overflow				
and \$t1,\$t2,\$t3	Bitwise AND : Set \$t1 to bitwise AND of \$t2 and \$t3				
andi \$t1,\$t2,100	Bitwise AND immediate : Set \$t1 to bitwise AND of \$t2 and zero-extended 16-bit immediate				
bc1f 1,label	Branch if specified FP condition flag false (BC1F, not BCLF) : If Coprocessor 1 condition flag specified is false, branch to statement at label's address				
bc1f label	Branch if FP condition flag 0 false (BC1F, not BCLF) : If Coprocessor 1 condition flag 0 is false, branch to statement at label's address				
bc1t 1,label	Branch if specified FP condition flag true (BC1T, not BCLT) : If Coprocessor 1 condition flag specified is true, branch to statement at label's address				
bc1t label	Branch if FP condition flag 0 true (BC1T, not BCLT) : If Coprocessor 1 condition flag 0 is true, branch to statement at label's address				
beq \$t1,\$t2,label	Branch if equal : Branch to statement at label's address if \$t1 and \$t2 are equal				
bgez \$t1,label	Branch if greater than or equal to zero : Branch to statement at label's address if \$t1 is greater than or equal to zero				
bgezal \$t1,label	Branch if greater than or equal to zero and link : If \$t1 is greater than or equal to zero, then set \$ra to the PC plus 4 and branch to statement at label's address				
bgtz \$t1,label	Branch if greater than zero : Branch to statement at label's address if \$t1 is greater than zero				
blez \$t1,label	Branch if less than or equal to zero : Branch to statement at label's address if \$t1 is less than or equal to zero				
bltz \$t1,label	Branch if less than zero : Branch to statement at label's address if \$t1 is less than zero				
bltzal \$t1,label	Branch if less than zero and link : If \$t1 is less than or equal to zero, then set \$ra to the PC plus 4 and branch to statement at label's address				
bne \$t1,\$t2,label	Branch if not equal : Branch to statement at label's address if \$t1 and \$t2 are not equal				
break	Break execution : Terminate program execution with exception				
break 100	Break execution with code : Terminate program execution with specified exception code				



SYSTEM CALL

Basic Instructions	Extended (pseudo) Instructions	Directives	Syscalls	Exceptions	Macros
--------------------	--------------------------------	------------	-----------------	------------	--------

Table of Available Services

Service	Code in \$v0	Arguments	Result
print integer	1	\$a0 = integer to print	
print float	2	\$f12 = float to print	
print double	3	\$f12 = double to print	
print string	4	\$a0 = address of null-terminated string to print	
read integer	5		\$v0 contains integer read
read float	6		\$f0 contains float read
read double	7		\$f0 contains double read
read string	8	\$a0 = address of input buffer \$a1 = maximum number of characters to read	<i>See note below table</i>
sbrk (allocate heap memory)	9	\$a0 = number of bytes to allocate	\$v0 contains address of allocated memory
exit (terminate execution)	10		

MARS(MIPS ASSEMBLER AND RUNTIME SIMULATOR)

- MARS is MIPS assembler and runtime simulator
- GUI with point-and-click control and integrated editor
- Easily editable register and memory values, similar to a spreadsheet
- Display values in hexadecimal or decimal
- Command line mode for instructors to test and evaluate many programs easily
- MARS website <http://courses.missouristate.edu/kenvollmar/mars/>
- Java https://www.java.com/zh_TW/



MARS介面

C:\Users\Wang\Google Drive\106-1課程\CO\LAB\lab1\near_prime_num.s - MARS 4.5

File Edit Run Settings Help

Run speed at max (no interaction)

near_prime_num.s*

Code

Registers

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x10010000
\$v0	2	0x0000000a
\$v1	3	0x00000000
\$a0	4	0x00000094
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000001
\$t1	9	0x00000094
\$t2	10	0x0000009f
\$t3	11	0x00000094
\$t4	12	0x00000000
\$t5	13	0x00000002
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x00000000
\$s1	17	0x00000094
\$s2	18	0x00000000
\$s3	19	0x00000094
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$t8	24	0x00000000
\$t9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10008000
\$sp	29	0x7fffffc0
\$fp	30	0x00000000
\$ra	31	0x00000000
\$pc		0x00400000
hi		0x00000000
lo		0x00000000

Mars Messages

Run I/O

Clear

pls enter a number : 159

earrest prime number is: 163

earrest prime number is: 157

-- program is finished running --

暫存器訊息

組譯與執行

1. 開啟舊檔

3. 執行檔案

2. 組譯

4. 輸入指令並閱讀資訊

Registers Coproc 1 Coproc 0

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x10010000
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000094
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000001
\$t1	9	0x00000094
\$t2	10	0x0000009f
\$t3	11	0x00000094
\$t4	12	0x00000000
\$t5	13	0x00000002
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x00000000
\$s1	17	0x00000094
\$s2	18	0x00000000
\$s3	19	0x00000094
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$t8	24	0x00000000
\$t9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10000000
\$sp	29	0x7fffffc0
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00400000
hi		0x00000000
lo		0x00000000

Mars Messages Run I/O

pls enter a number : 159

nearest prime number is: 163

nearest prime number is: 157

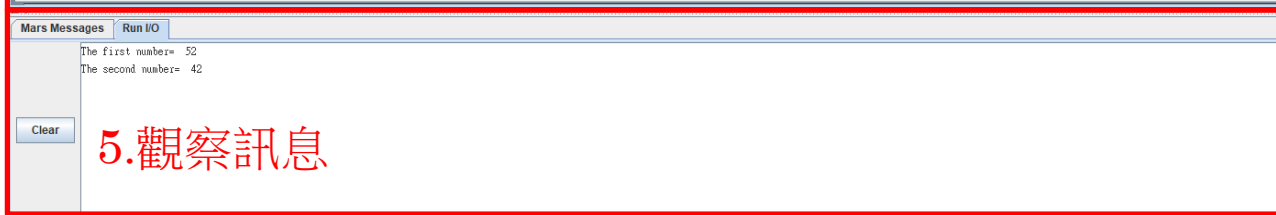
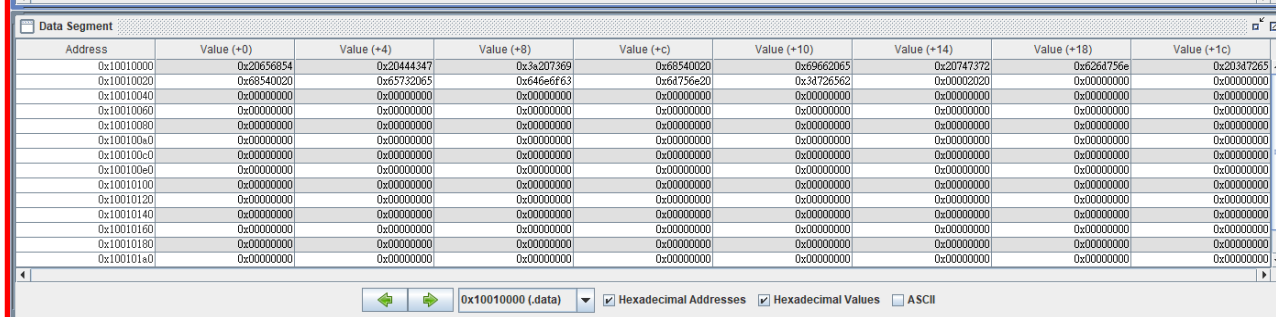
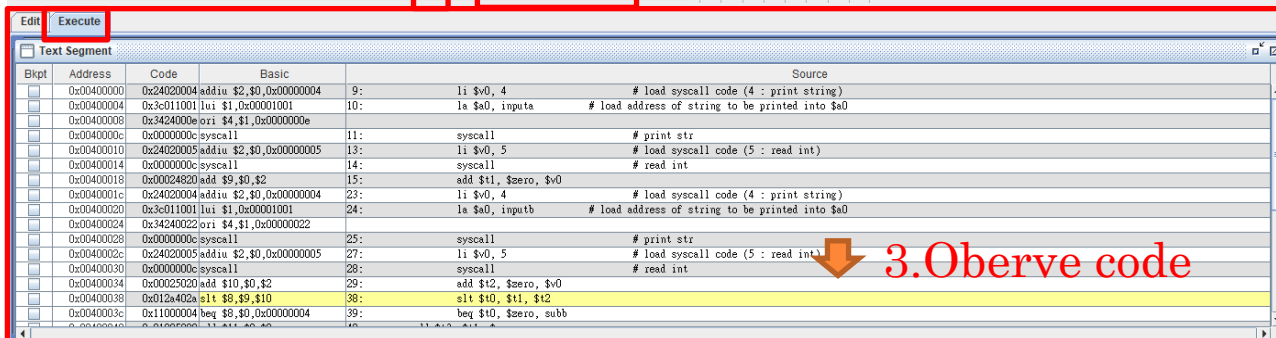
-- program is finished running --

啟用 Windows
移至 [設定] 以啟用 Windows

逐步執行

1. 選擇execute tab 2. Assemble and run one step at a time

C:\Users\Wang\Google Drive\106-1課程\CO\LAB\lab1\LAB1_code\GCD20170821.s - MARS 4.5



4. 觀察暫存器

Register	Register 4	Register 5
Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x10010000
\$v0	2	0x0000002a
\$v1	3	0x00000000
\$a0	4	0x10010022
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000
\$t1	9	0x00000034
\$t2	10	0x0000002a
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$t8	16	0x00000000
\$t9	17	0x00000000
\$s0	18	0x00000000
\$s1	19	0x00000000
\$s2	20	0x00000000
\$s3	21	0x00000000
\$s4	22	0x00000000
\$s5	23	0x00000000
\$s6	24	0x00000000
\$s7	25	0x00000000
\$s8	26	0x00000000
\$s9	27	0x00000000
\$sp	28	0x10008000
\$fp	29	0x7ffffcfc
\$ra	30	0x00000000
pc	31	0x00000000
hi		0x00000000
lo		0x00000000

3. Observe code

5. 觀察訊息

啟用 Windows
移至 [設定] 以啟用 Windows。

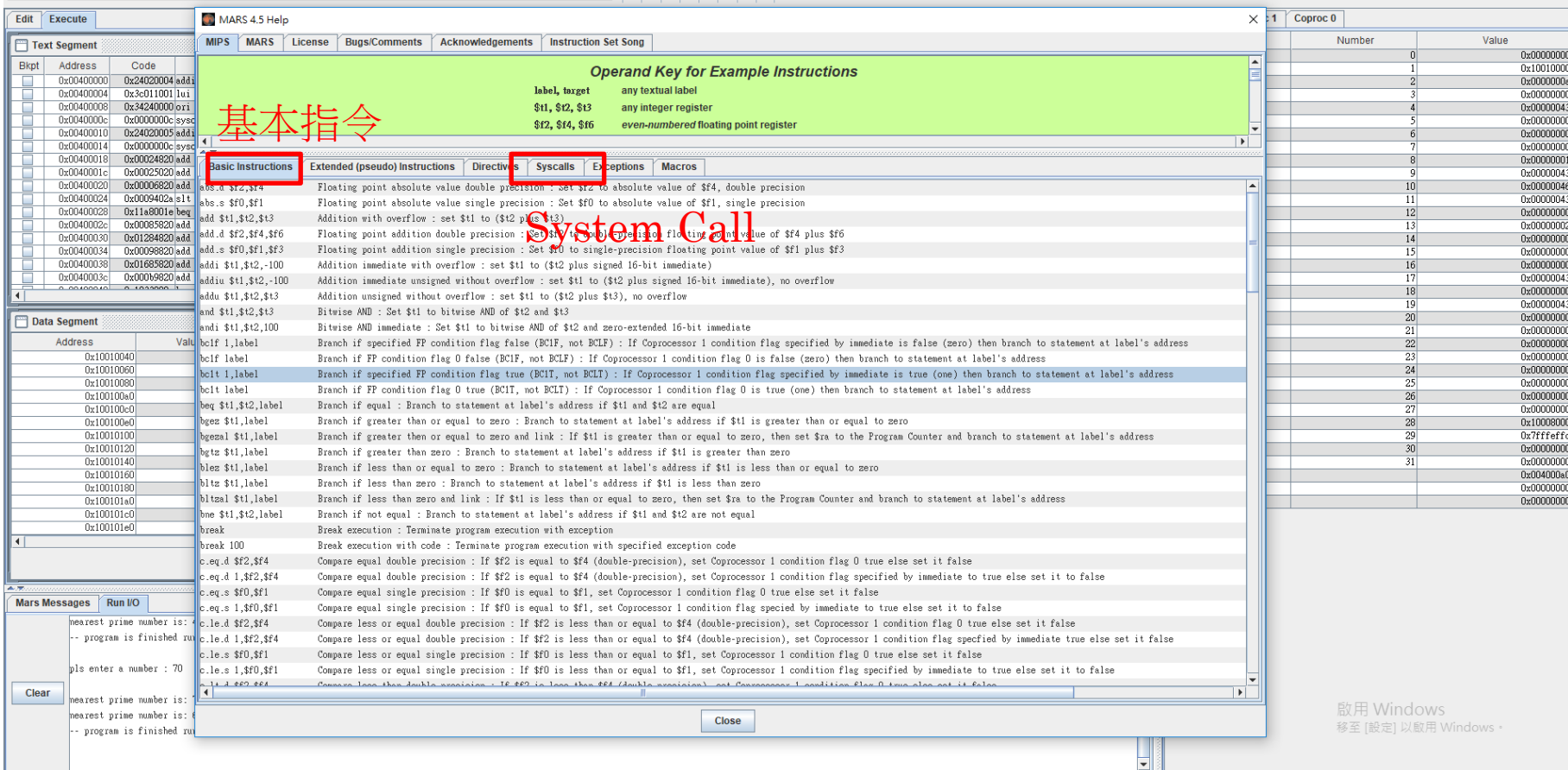
查詢組語指令

點擊help



File Edit Run Settings Tools Help

Run speed at max (no interaction)



範例程式解說 – 最大公因數


```
.data
msgb : .asciiz "The GCD is : "
inputa : .asciiz "The first number= "
inputb : .asciiz "The second number= "
.text
.globl main
main :

    #讀第一個數到t1
    li $v0, 4
    la $a0, inputa
    syscall
    li $v0, 5
    syscall
    add $t1, $zero, $v0

    #讀第二個數到t2
    li $v0, 4
    la $a0, inputb
    syscall
    li $v0, 5
    syscall
    add $t2, $zero, $v0

    # load syscall code (4 : print string)
    # load address of string to be printed into $a0
    # print str
    # load syscall code (5 : read int)
    # read int

    # load syscall code (4 : print string)
    # load address of string to be printed into $a0
    # print str
    # load syscall code (5 : read int)
    # read int
```



#比對 t1 t2 大小，若t1大則做減法，若t1較小則與t2做對調

```
comp :    slt $t0, $t1, $t2          #判斷t1<t2 成立t1=1 else t1=0
          beq $t0, $zero, subb       #如果相等跳到subb(標籤)
          add $t3, $t1, $zero
          add $t1, $t2, $zero
          add $t2, $t3, $zero
subb :    sub $t1, $t1, $t2
          bne $t1, $zero, comp
```

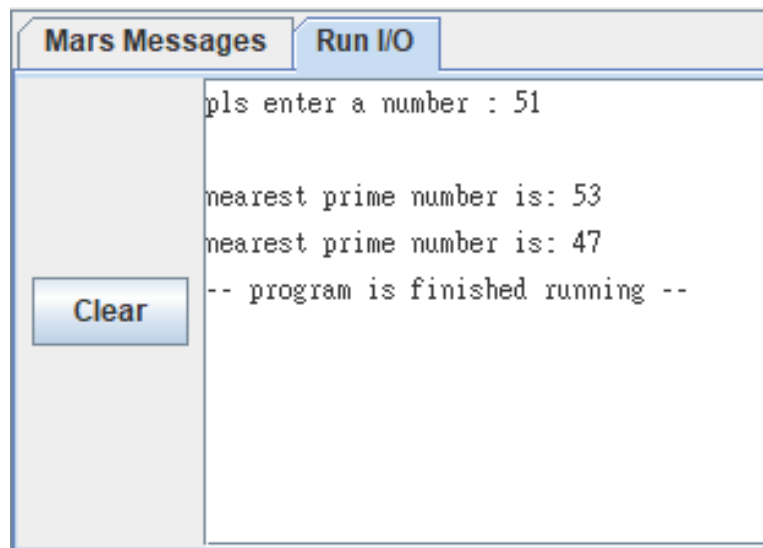
#顯示最大公因數

```
li $v0, 4
la $a0, msgb
syscall
add $a0,$zero $t2
li $v0, 1
syscall
li $v0, 10
syscall
```



課堂實作

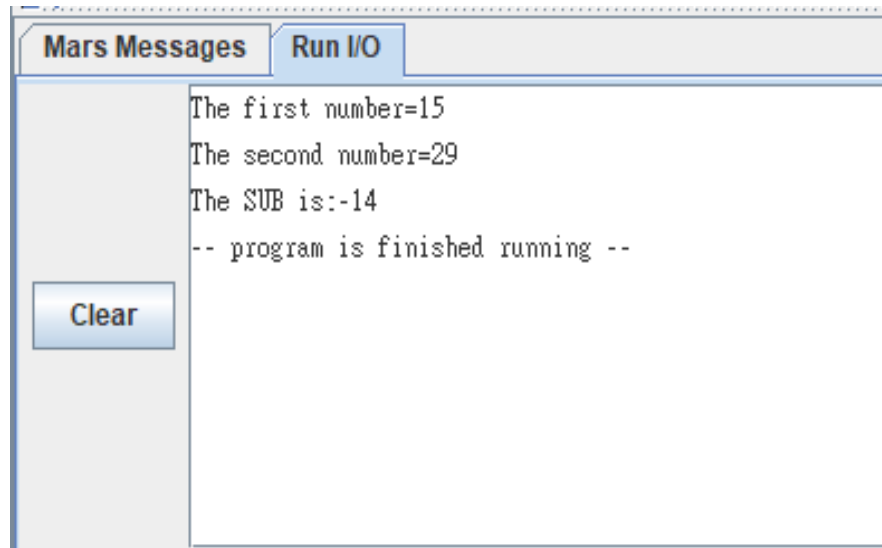
- 題目一：執行範例程式並輸入兩自然數找出最大公因數
- 題目二：參考範例程式與PPT輸入兩自然數做減法
- 期限：Lab1課堂結束前



Mars Messages Run I/O

```
pls enter a number : 51  
  
nearest prime number is: 53  
nearest prime number is: 47  
-- program is finished running --
```

Clear



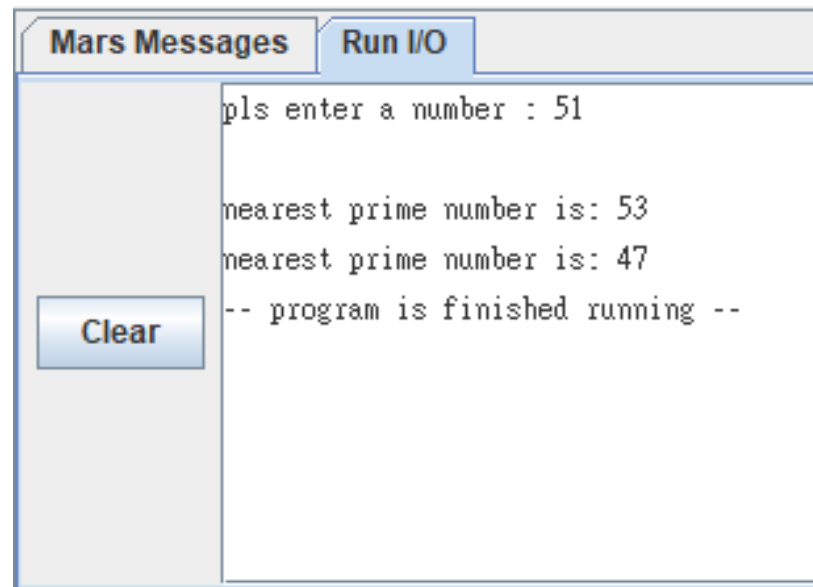
Mars Messages Run I/O

```
The first number=15  
The second number=29  
The SUB is:-14  
-- program is finished running --
```

Clear

課後作業

- 題目：輸入一自然數 X ，找出最接近的兩質數 A 與 B ($X > A$, $X < B$)
- 期限：2018/10/15
- 繳交：上傳程式碼至ECOURSE



The screenshot shows a window titled "Mars Messages" with a "Run I/O" button. The output text is as follows:

```
pls enter a number : 51  
  
nearest prime number is: 53  
nearest prime number is: 47  
-- program is finished running --
```

A "Clear" button is visible on the left side of the output area.



作業限制指令

- lw:把資料從記憶體搬到暫存器
- sw:把資料從暫存器搬到記憶體
- add:加法
- sub:減法
- slt:判斷小於
- beq:若相等跳躍
- bne:若不相等跳躍
- j:跳躍
- or:或
- and:和



作業計分方式

- 課堂練習：50%
- 課後作業：50%
 - 功能正常：35%
 - 降低執行指令數：1~5 名 15%
6~20名 5%
- 遲交分數打九折，未交者不予計分。

