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

MIPS Assembly Programming

王宣仁

2017/10/25

實驗內容

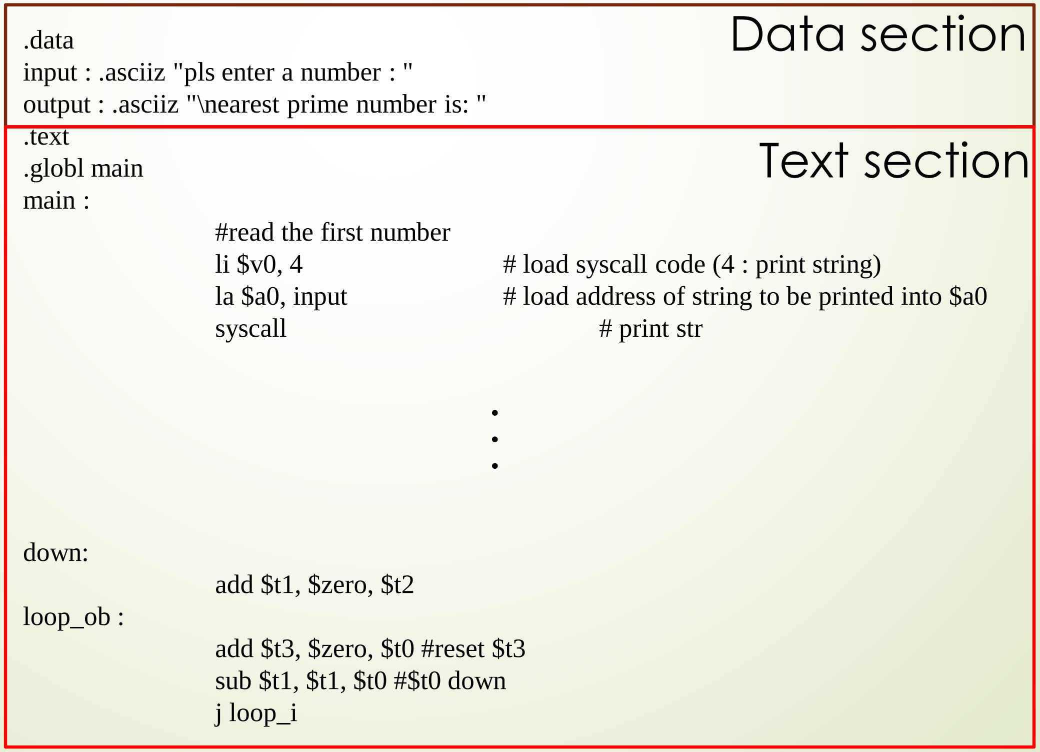
- 實驗目的
- MIPS Assembly Programming
- MARS (MIPS Assembler and Runtime Simulator)
- 課堂實作
- 課後作業
- 計分方式
- 附錄

實驗目的

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

MIPS Assembly Programming

- 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 section

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

Text section

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

範例程式解說 – 最大公因數

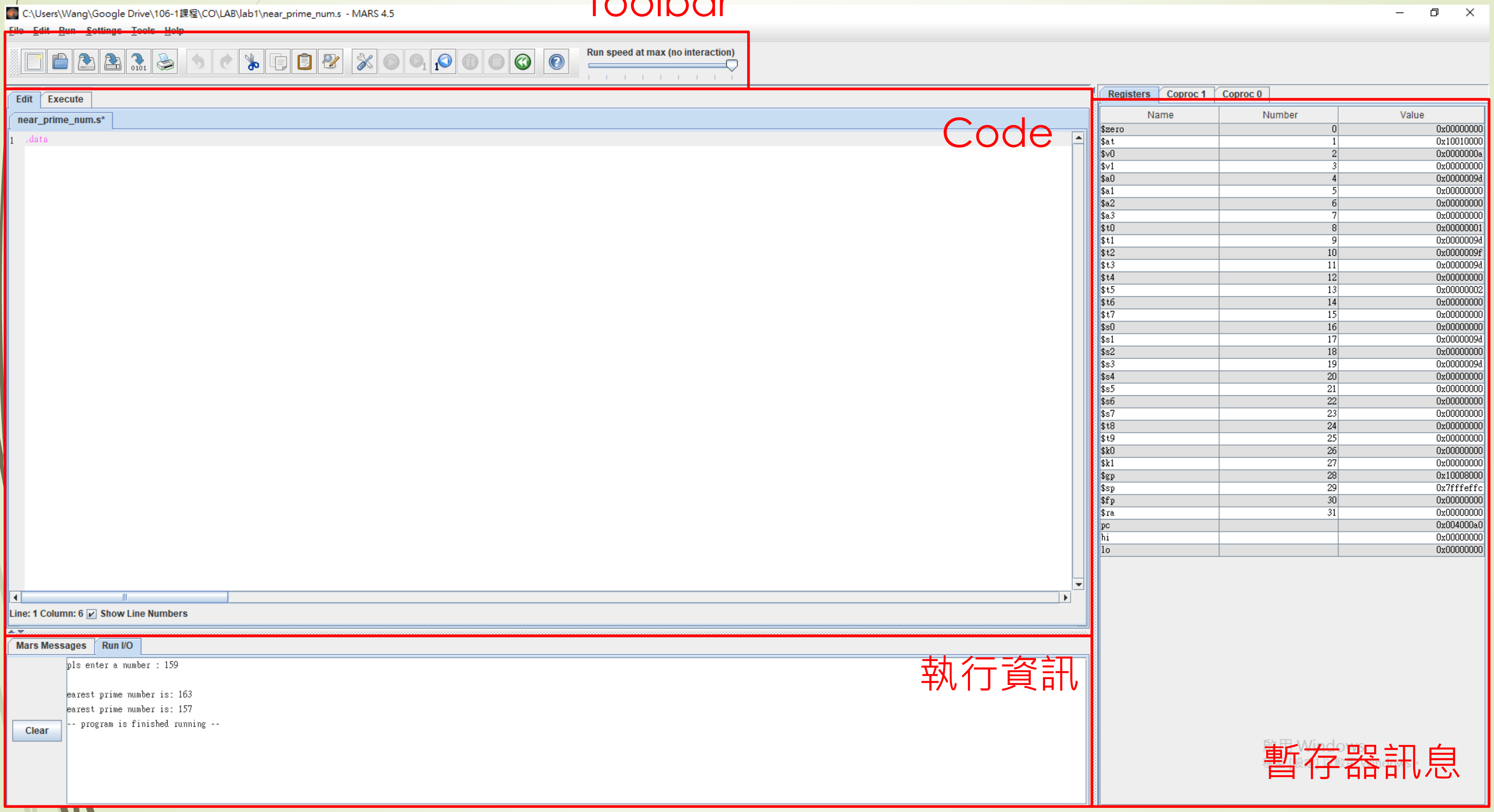
```
.data
msgb : .asciiz "The GCD is : "
inputa : .asciiz "The first number=  "
inputb : .asciiz "The second number=  "
.text
.globl main
main :
        #輸入兩自然數求最大公因數
        #讀第一個數到t1
        li $v0, 4               # load syscall code (4 : print string)
        la $a0, inputa          # load address of string to be printed into $a0
        syscall                 # print str
        li $v0, 5               # load syscall code (5 : read int)
        syscall                 # read int
        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
```

```
            #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
            beq $t0, $zero, subb
            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
```

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 websitehttp://courses.missouristate.edu/kenvollmar/mars/

MARS介面



```
.data
```

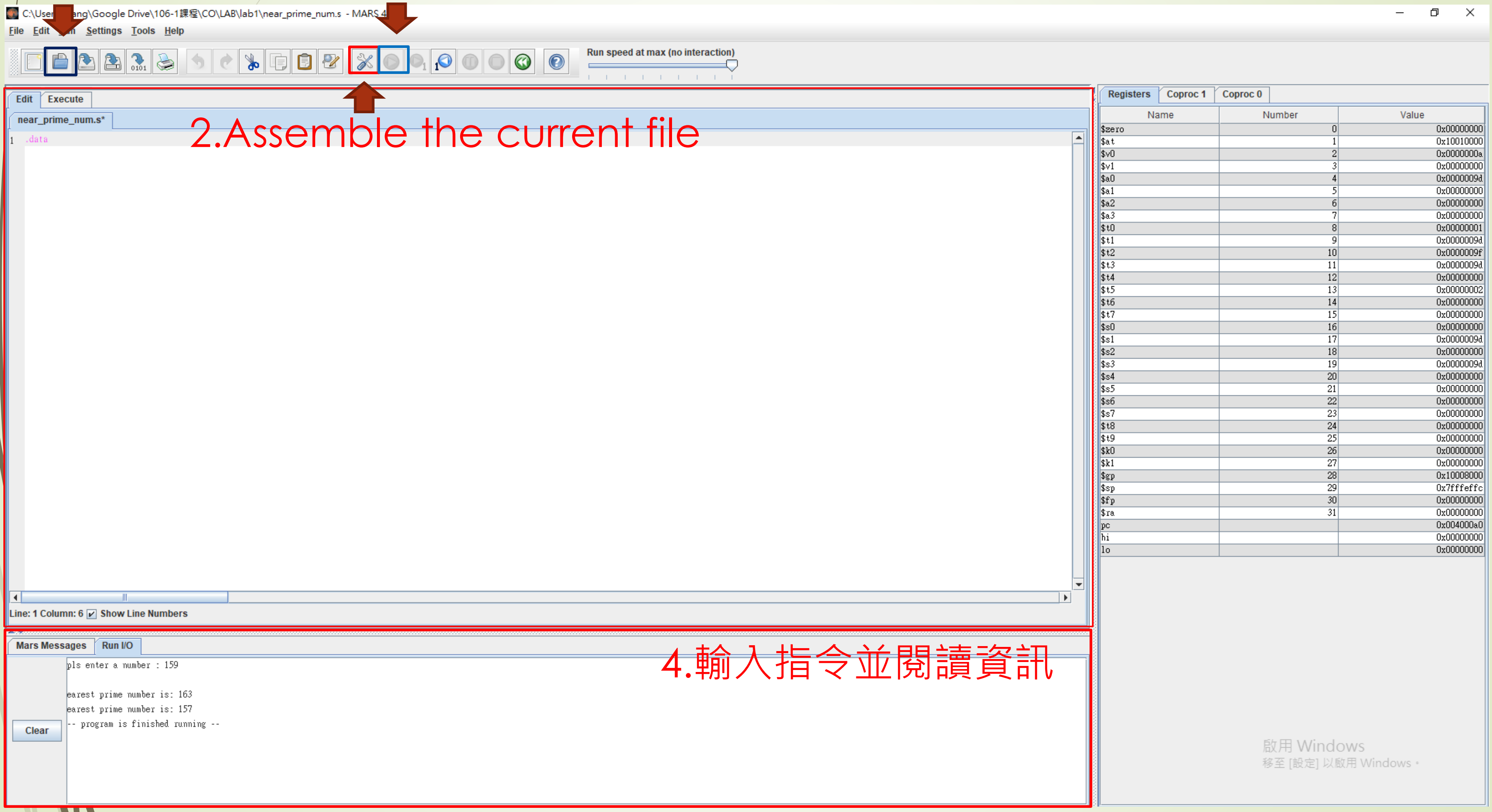
Name	Number	Value
$zero	0	0x00000000
$at	1	0x10010000
$v0	2	0x0000000a
$v1	3	0x00000000
$a0	4	0x0000009d
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000001
$t1	9	0x0000009d
$t2	10	0x0000009f
$t3	11	0x0000009d
$t4	12	0x00000000
$t5	13	0x00000002
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x0000009d
$s2	18	0x00000000
$s3	19	0x0000009d
$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	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x004000a0
hi		0x00000000
lo		0x00000000

```
pls enter a number : 159

earest prime number is: 163
earest prime number is: 157
-- program is finished running --
```

組譯與執行

1.Open code

3.Run the current file



C:\Users\ang\Google Drive\106-1課程\CO\LAB\lab1\near_prime_num.s - MARS 4
File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
near_prime_num.s*
Line: 1 Column: 6 ☑ Show Line Numbers
Mars Messages Run I/O
Clear
Registers Coproc 1 Coproc 0

2.Assemble the current file

```
1  .data
```

4.輸入指令並閱讀資訊

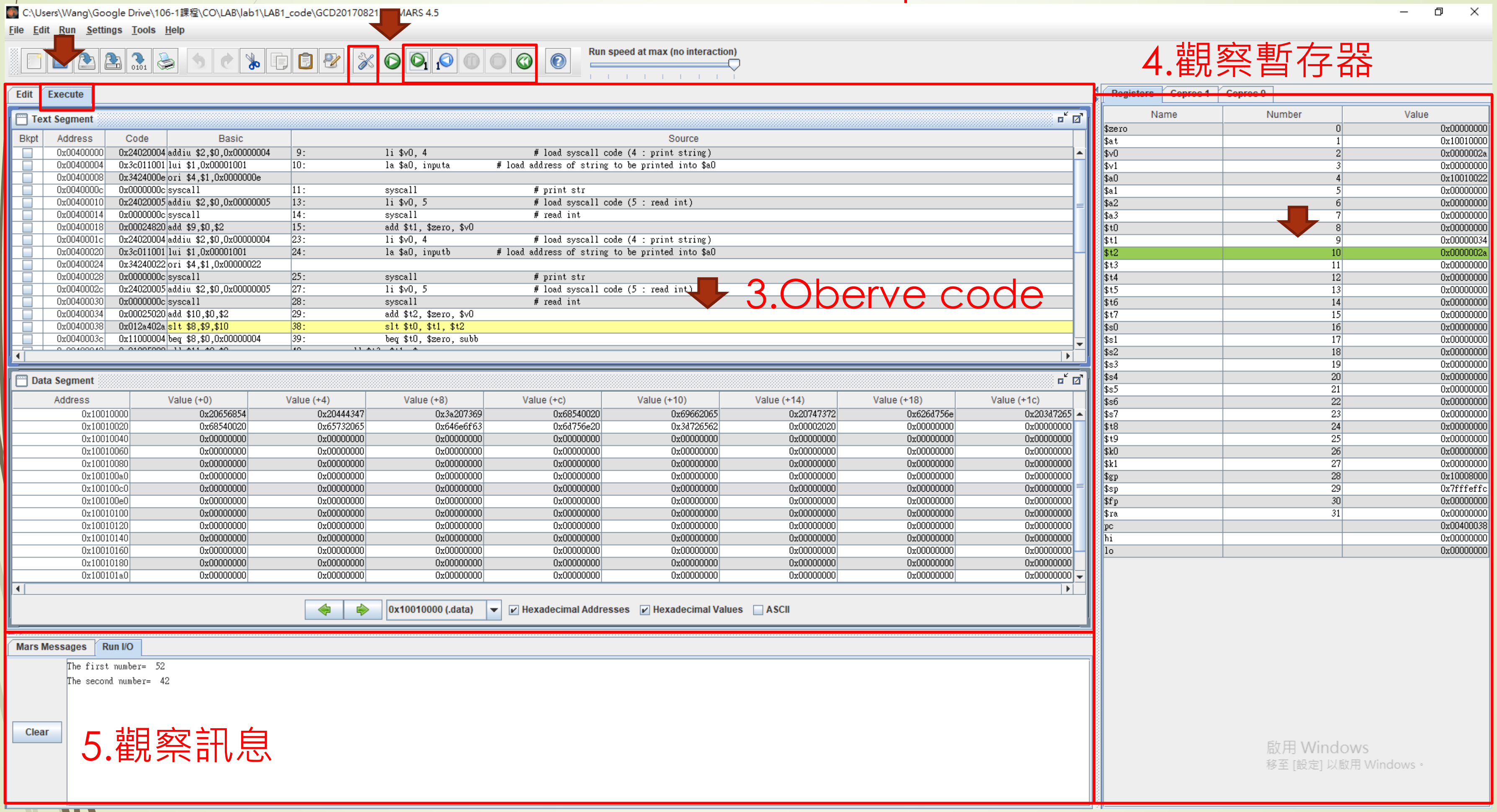
```
pls enter a number : 159

earest prime number is: 163
earest prime number is: 157
-- program is finished running --
```

Name	Number	Value
$zero	0	0x00000000
$at	1	0x10010000
$v0	2	0x0000000a
$v1	3	0x00000000
$a0	4	0x0000009d
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000001
$t1	9	0x0000009d
$t2	10	0x0000009f
$t3	11	0x0000009d
$t4	12	0x00000000
$t5	13	0x00000002
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x0000009d
$s2	18	0x00000000
$s3	19	0x0000009d
$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	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x004000a0
hi		0x00000000
lo		0x00000000

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 MARS 4.5
File
Edit
Run
Settings
Tools
Help
0101
Run speed at max (no interaction)
Edit
Execute

Text Segment

Bkpt	Address	Code	Basic	Source
☐	0x00400000	0x24020004	addiu $2,$0,0x00000004	9: li $v0, 4 # load syscall code (4 : print string)
☐	0x00400004	0x3c011001	lui $1,0x00001001	10: la $a0, inputa # load address of string to be printed into $a0
☐	0x00400008	0x3424000e	ori $4,$1,0x0000000e	
☐	0x0040000c	0x0000000c	syscall	11: syscall # print str
☐	0x00400010	0x24020005	addiu $2,$0,0x00000005	13: li $v0, 5 # load syscall code (5 : read int)
☐	0x00400014	0x0000000c	syscall	14: syscall # read int
☐	0x00400018	0x00024820	add $9,$0,$2	15: add $t1, $zero, $v0
☐	0x0040001c	0x24020004	addiu $2,$0,0x00000004	23: li $v0, 4 # load syscall code (4 : print string)
☐	0x00400020	0x3c011001	lui $1,0x00001001	24: la $a0, inputb # load address of string to be printed into $a0
☐	0x00400024	0x34240022	ori $4,$1,0x00000022	
☐	0x00400028	0x0000000c	syscall	25: syscall # print str
☐	0x0040002c	0x24020005	addiu $2,$0,0x00000005	27: li $v0, 5 # load syscall code (5 : read int)
☐	0x00400030	0x0000000c	syscall	28: syscall # read int
☐	0x00400034	0x00025020	add $10,$0,$2	29: add $t2, $zero, $v0
☐	0x00400038	0x012a402a	slt $8,$9,$10	38: slt $t0, $t1, $t2
☐	0x0040003c	0x11000004	beq $8,$0,0x00000004	39: beq $t0, $zero, subb

3.Oberve code

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x20656854	0x20444347	0x3a207369	0x68540020	0x69662065	0x20747372	0x626d756e	0x203d7265
0x10010020	0x68540020	0x65732065	0x646e6f63	0x6d756e20	0x3d726562	0x00002020	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010140	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010160	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010180	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values ☐ ASCII

4.觀察暫存器

Registers | Coproc 1 | Coproc 0

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
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$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	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00400038
hi		0x00000000
lo		0x00000000

Mars Messages | Run I/O

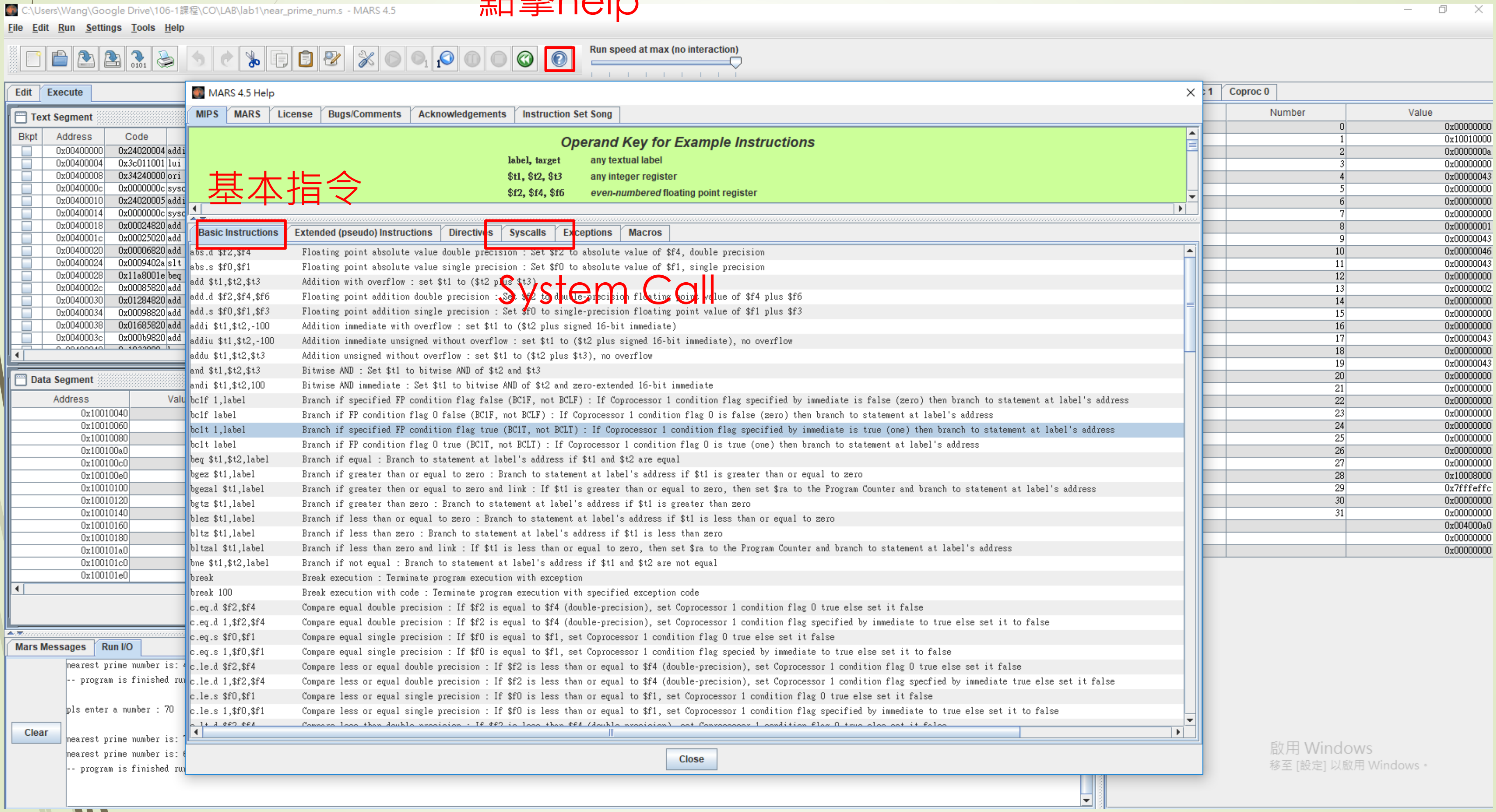
```
The first number= 52
The second number= 42
```

Clear

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

基本指令

System Call

Operand Key for Example Instructions

label, target	any textual label
$t1, $t2, $t3	any integer register
$f2, $f4, $f6	*even-numbered* floating point register

```
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 by immediate is false (zero) then 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 (zero) then 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 by immediate is true (one) then 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 (one) then 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 then or equal to zero and link : If $t1 is greater than or equal to zero, then set $ra to the Program Counter 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 Program Counter 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
c.eq.d $f2,$f4       Compare equal double precision : If $f2 is equal to $f4 (double-precision), set Coprocessor 1 condition flag 0 true else set it false
c.eq.d 1,$f2,$f4     Compare equal double precision : If $f2 is equal to $f4 (double-precision), set Coprocessor 1 condition flag specified by immediate to true else set it to false
c.eq.s $f0,$f1       Compare equal single precision : If $f0 is equal to $f1, set Coprocessor 1 condition flag 0 true else set it false
c.eq.s 1,$f0,$f1     Compare equal single precision : If $f0 is equal to $f1, set Coprocessor 1 condition flag specied by immediate to true else set it to false
c.le.d $f2,$f4       Compare less or equal double precision : If $f2 is less than or equal to $f4 (double-precision), set Coprocessor 1 condition flag 0 true else set it false
c.le.d 1,$f2,$f4     Compare less or equal double precision : If $f2 is less than or equal to $f4 (double-precision), set Coprocessor 1 condition flag specfied by immediate true else set it false
c.le.s $f0,$f1       Compare less or equal single precision : If $f0 is less than or equal to $f1, set Coprocessor 1 condition flag 0 true else set it false
c.le.s 1,$f0,$f1     Compare less or equal single precision : If $f0 is less than or equal to $f1, set Coprocessor 1 condition flag specified by immediate to true else set it to false
c.lt.d $f2,$f4       Compare less than double precision : If $f2 is less than $f4 (double-precision), set Coprocessor 1 condition flag 0 true else set it false
```

Bkpt	Address	Code	
□	0x00400000	0x24020004	addi
□	0x00400004	0x3c011001	lui
□	0x00400008	0x34240000	ori
□	0x0040000c	0x0000000c	sysc
□	0x00400010	0x24020005	addi
□	0x00400014	0x0000000c	sysc
□	0x00400018	0x00024820	add
□	0x0040001c	0x00025020	add
□	0x00400020	0x00006820	add
□	0x00400024	0x0009402a	slt
□	0x00400028	0x11a8001e	beq
□	0x0040002c	0x00085820	add
□	0x00400030	0x01284820	add
□	0x00400034	0x00098820	add
□	0x00400038	0x01685820	add
□	0x0040003c	0x000b9820	add

Address	Valu
0x10010040	
0x10010060	
0x10010080	
0x100100a0	
0x100100c0	
0x100100e0	
0x10010100	
0x10010120	
0x10010140	
0x10010160	
0x10010180	
0x100101a0	
0x100101c0	
0x100101e0	

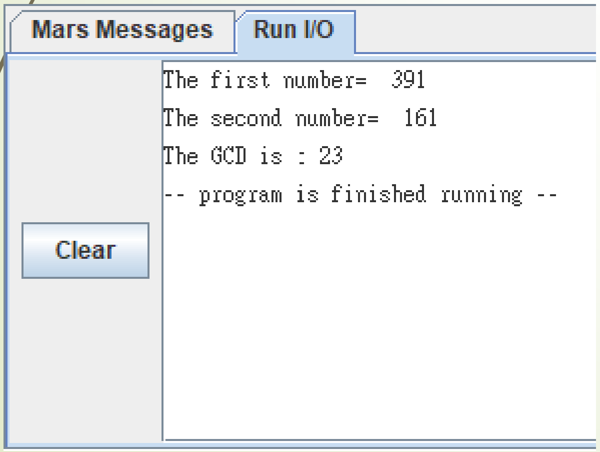
```
nearest prime number is:
-- program is finished ru

pls enter a number : 70

nearest prime number is:
nearest prime number is:
-- program is finished ru
```

Number	Value
0	0x00000000
1	0x10010000
2	0x0000000a
3	0x00000000
4	0x00000043
5	0x00000000
6	0x00000000
7	0x00000000
8	0x00000001
9	0x00000043
10	0x00000046
11	0x00000043
12	0x00000000
13	0x00000002
14	0x00000000
15	0x00000000
16	0x00000000
17	0x00000043
18	0x00000000
19	0x00000043
20	0x00000000
21	0x00000000
22	0x00000000
23	0x00000000
24	0x00000000
25	0x00000000
26	0x00000000
27	0x00000000
28	0x10008000
29	0x7fffeffc
30	0x00000000
31	0x00000000
	0x004000a0
	0x00000000
	0x00000000

課堂實作

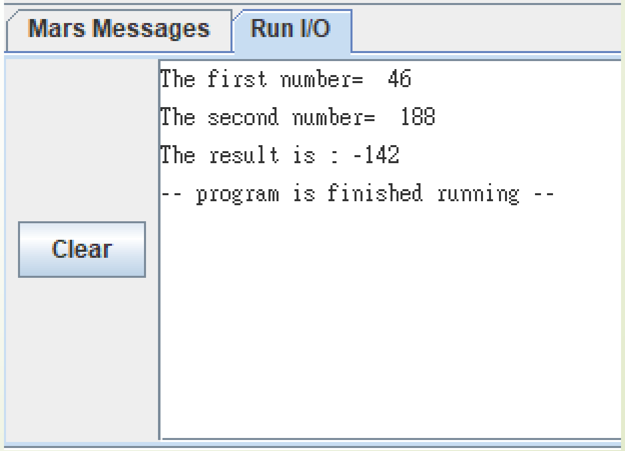
- 題目一：執行範例程式並輸入兩自然數找出最大公因數
- 題目二：參考範例程式與PPT輸入兩自然數做減法
- 期　限：Lab1課堂結束前

題目一示範



```
The first number=  391
The second number=  161
The GCD is : 23
-- program is finished running --
```

題目二示範



```
The first number=  46
The second number=  188
The result is : -142
-- program is finished running --
```

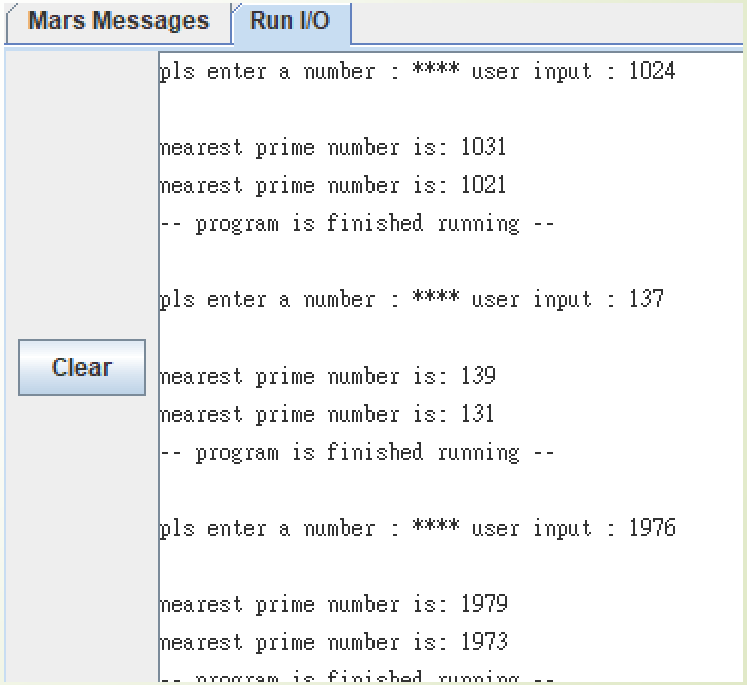
課後作業

- 題目：輸入X(X ∈ $N, X > 3$)，找出最接近的兩質數A與B(X>A, X<B)

  ，並能給予文字提示如附圖執行1024、137與1976。

- 期限：2017/11/01

- 繳交：上傳程式碼至ECOURSE



```
pls enter a number : **** user input : 1024

nearest prime number is: 1031
nearest prime number is: 1021
-- program is finished running --

pls enter a number : **** user input : 137

nearest prime number is: 139
nearest prime number is: 131
-- program is finished running --

pls enter a number : **** user input : 1976

nearest prime number is: 1979
nearest prime number is: 1973
-- program is finished running --
```

Lab1計分方式

- 課堂練習：50%

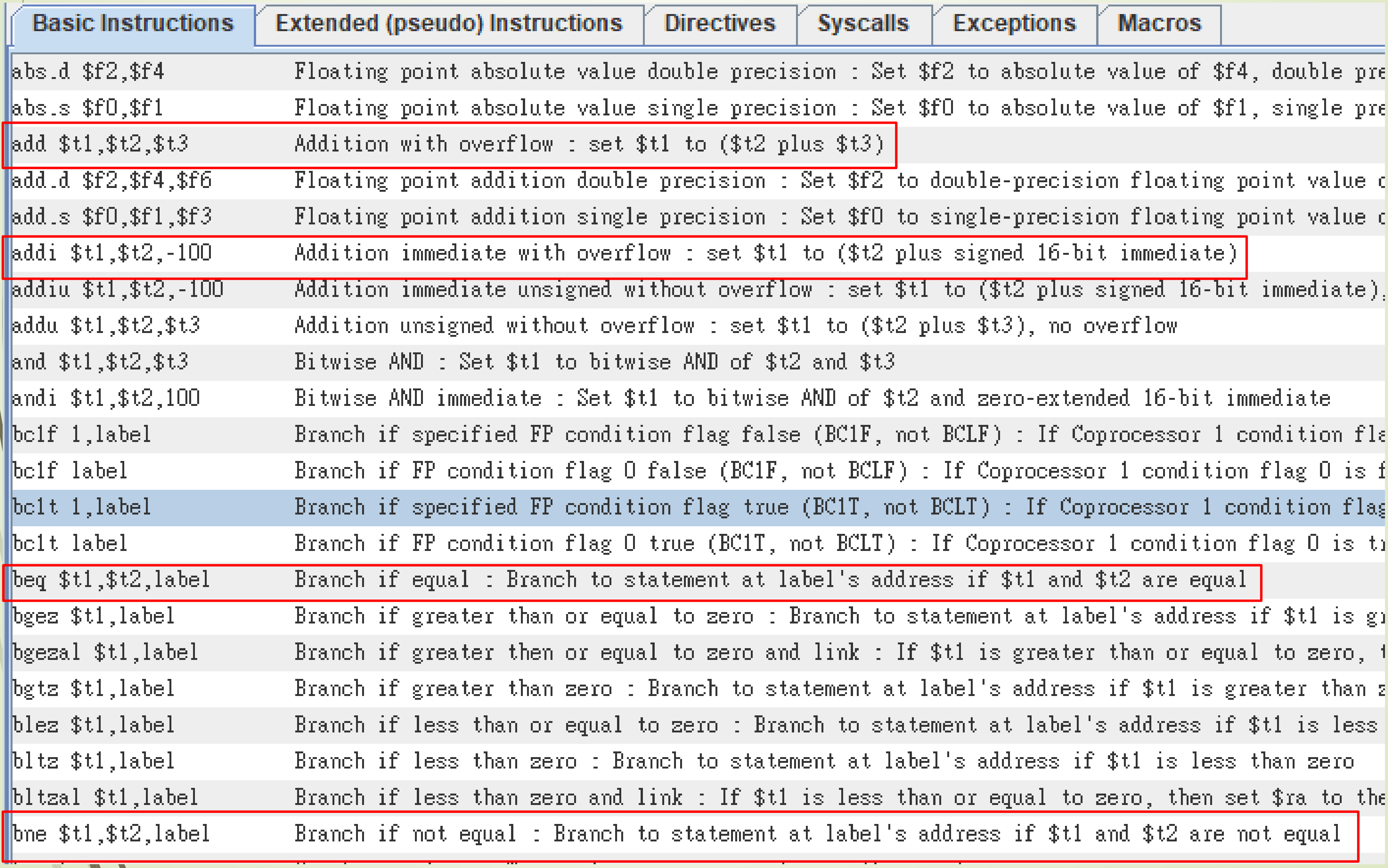
- 課後作業：50%

  - 功能正常：35%

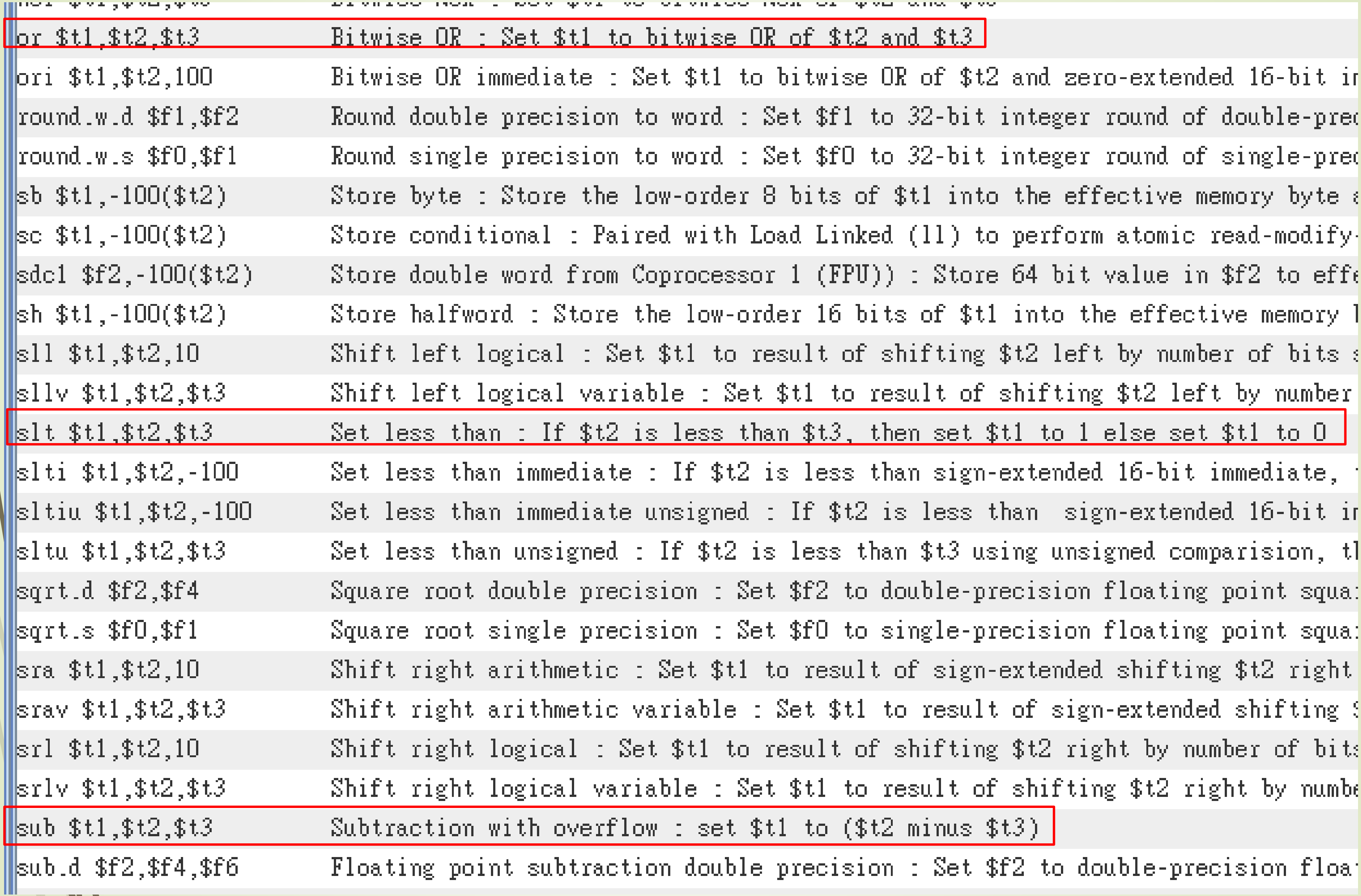
  - 降低執行指令數：1~5 名 15%

    6~20名 5%

附錄



abs.d $f2,$f4	Floating point absolute value double precision : Set $f2 to absolute value of $f4, double pre
abs.s $f0,$f1	Floating point absolute value single precision : Set $f0 to absolute value of $f1, single pre
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 o
add.s $f0,$f1,$f3	Floating point addition single precision : Set $f0 to single-precision floating point value o
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),
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 fla
bc1f label	Branch if FP condition flag 0 false (BC1F, not BCLF) : If Coprocessor 1 condition flag 0 is f
bc1t 1,label	Branch if specified FP condition flag true (BC1T, not BCLT) : If Coprocessor 1 condition flag
bc1t label	Branch if FP condition flag 0 true (BC1T, not BCLT) : If Coprocessor 1 condition flag 0 is tr
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 gr
bgezal $t1,label	Branch if greater then or equal to zero and link : If $t1 is greater than or equal to zero, t
bgtz $t1,label	Branch if greater than zero : Branch to statement at label's address if $t1 is greater than z
blez $t1,label	Branch if less than or equal to zero : Branch to statement at label's address if $t1 is less
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
bne $t1,$t2,label	Branch if not equal : Branch to statement at label's address if $t1 and $t2 are not equal



```
nor $t1,$t2,$t3          Bitwise NOR : Set $t1 to bitwise NOR of $t2 and $t3
or $t1,$t2,$t3           Bitwise OR : Set $t1 to bitwise OR of $t2 and $t3
ori $t1,$t2,100          Bitwise OR immediate : Set $t1 to bitwise OR of $t2 and zero-extended 16-bit i
round.w.d $f1,$f2        Round double precision to word : Set $f1 to 32-bit integer round of double-pre
round.w.s $f0,$f1        Round single precision to word : Set $f0 to 32-bit integer round of single-pre
sb $t1,-100($t2)         Store byte : Store the low-order 8 bits of $t1 into the effective memory byte
sc $t1,-100($t2)         Store conditional : Paired with Load Linked (ll) to perform atomic read-modify-
sdc1 $f2,-100($t2)       Store double word from Coprocessor 1 (FPU)) : Store 64 bit value in $f2 to eff
sh $t1,-100($t2)         Store halfword : Store the low-order 16 bits of $t1 into the effective memory
sll $t1,$t2,10           Shift left logical : Set $t1 to result of shifting $t2 left by number of bits
sllv $t1,$t2,$t3         Shift left logical variable : Set $t1 to result of shifting $t2 left by number
slt $t1,$t2,$t3          Set less than : If $t2 is less than $t3, then set $t1 to 1 else set $t1 to 0
slti $t1,$t2,-100        Set less than immediate : If $t2 is less than sign-extended 16-bit immediate,
sltiu $t1,$t2,-100       Set less than immediate unsigned : If $t2 is less than  sign-extended 16-bit i
sltu $t1,$t2,$t3         Set less than unsigned : If $t2 is less than $t3 using unsigned comparision, t
sqrt.d $f2,$f4           Square root double precision : Set $f2 to double-precision floating point squa
sqrt.s $f0,$f1           Square root single precision : Set $f0 to single-precision floating point squa
sra $t1,$t2,10           Shift right arithmetic : Set $t1 to result of sign-extended shifting $t2 right
srav $t1,$t2,$t3         Shift right arithmetic variable : Set $t1 to result of sign-extended shifting
srl $t1,$t2,10           Shift right logical : Set $t1 to result of shifting $t2 right by number of bit
srlv $t1,$t2,$t3         Shift right logical variable : Set $t1 to result of shifting $t2 right by numb
sub $t1,$t2,$t3          Subtraction with overflow : set $t1 to ($t2 minus $t3)
sub.d $f2,$f4,$f6        Floating point subtraction double precision : Set $f2 to double-precision floa
```

System Call

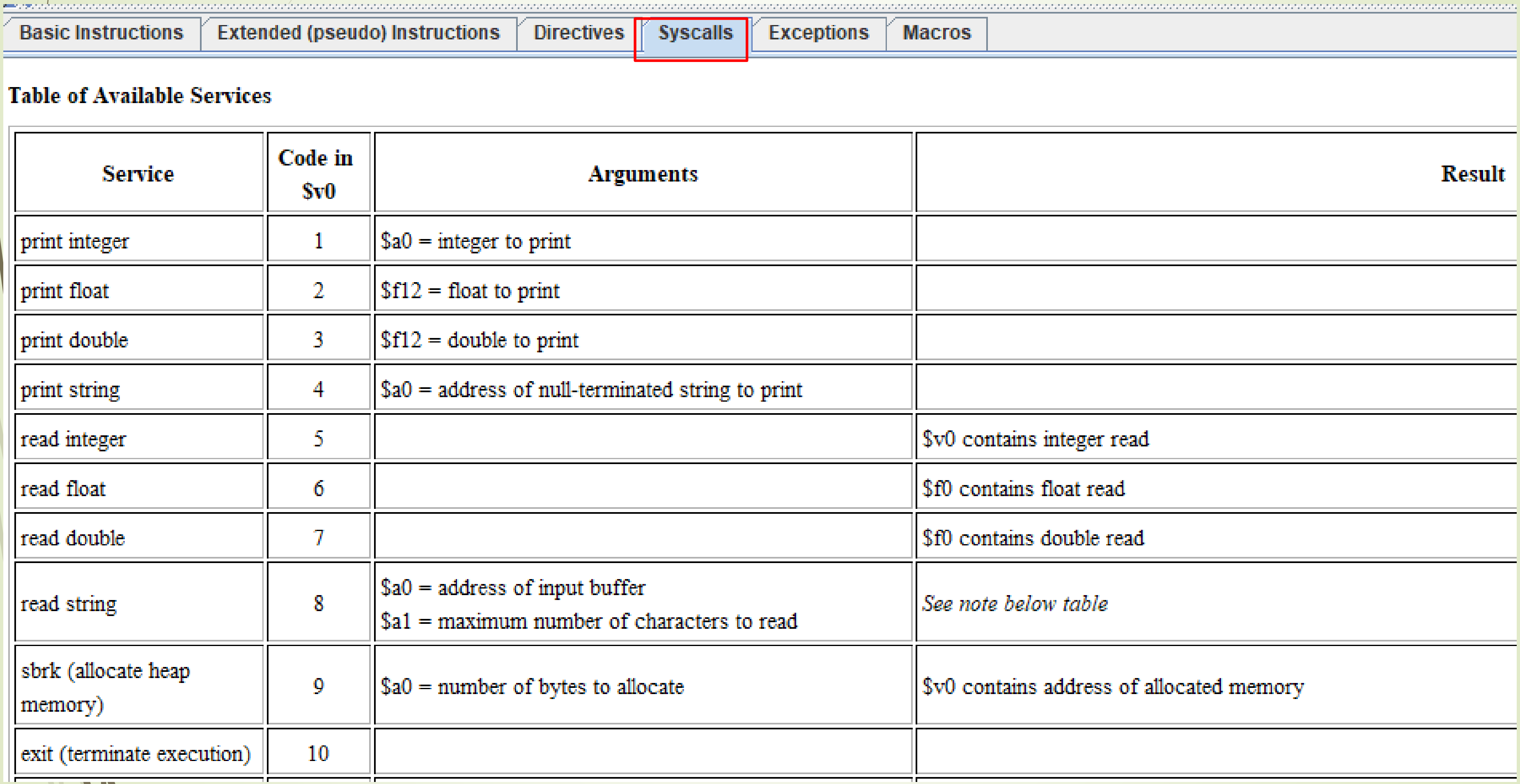


Table of Available Services

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	*See note below table*
sbrk (allocate heap memory)	9	$a0 = number of bytes to allocate	$v0 contains address of allocated memory
exit (terminate execution)	10		