MARS 4.5 Quick Start

CM Lee

About MARS

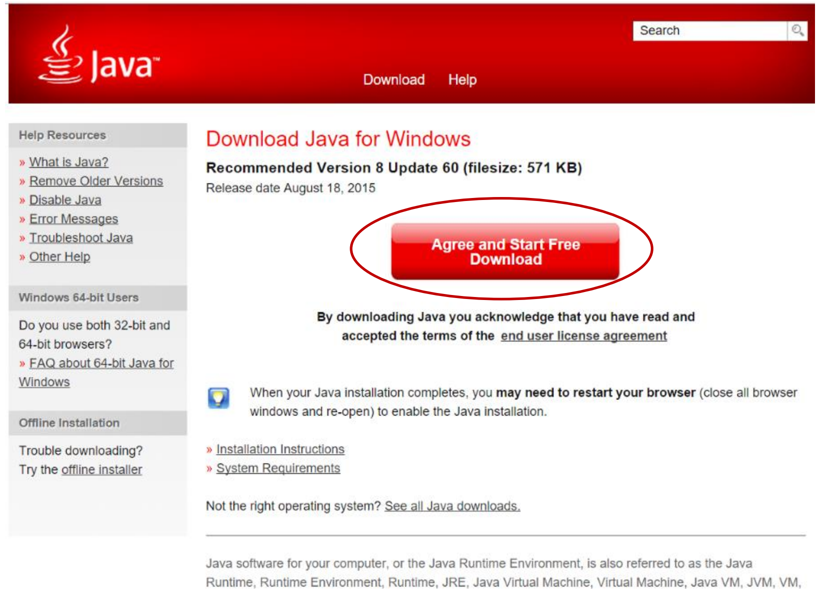
- **M**IPS **A**ssembler and **R**untime **S**imulator
    - Maintained by Missouri State University
    - Based on Java
- You can develop your MIPS assembly and do simulation in MARS
- Featuring useful tools that help you analyze and trace your assembly
- For more information, you can visit: http://courses.missouristate.edu/kenvollmar/mars/

Pre-requisite: Java Runtime Environment (JRE) for Windows

- Follow this page to make sure you have Java installed:

  https://www.java.com/zh_TW/download/help/version_manual.xml

- If not installed yet, go to https://java.com/zh_TW/download/ Download and install java



Help Resources

- What is Java?
- Remove Older Versions
- Disable Java
- Error Messages
- Troubleshoot Java
- Other Help

Windows 64-bit Users

Do you use both 32-bit and 64-bit browsers?

- FAQ about 64-bit Java for Windows

Offline Installation

Trouble downloading? Try the offline installer

Download Java for Windows

Recommended Version 8 Update 60 (filesize: 571 KB)

Release date August 18, 2015

Agree and Start Free Download

By downloading Java you acknowledge that you have read and accepted the terms of the end user license agreement

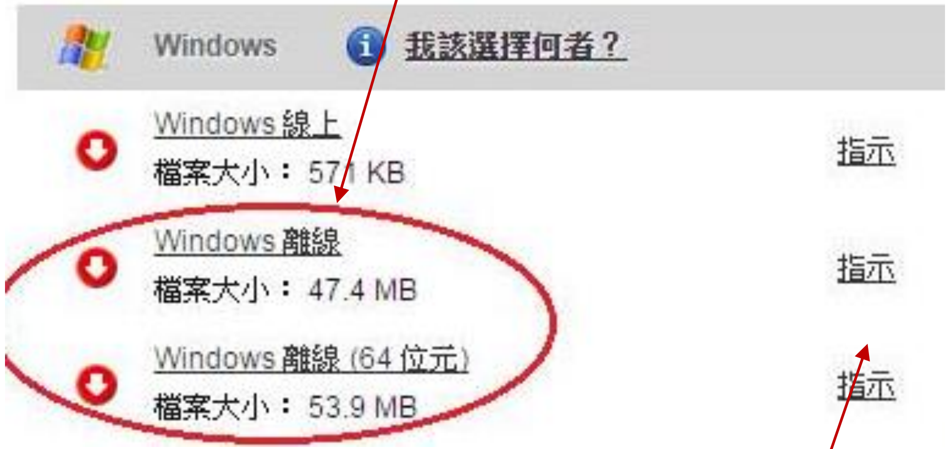
When your Java installation completes, you **may need to restart your browser** (close all browser windows and re-open) to enable the Java installation.

- Installation Instructions
- System Requirements

Not the right operating system? See all Java downloads.

Java software for your computer, or the Java Runtime Environment, is also referred to as the Java Runtime, Runtime Environment, Runtime, JRE, Java Virtual Machine, Virtual Machine, Java VM, JVM, VM,

Carefully choose the link that matches your OS.



Windows 我該選擇何者？

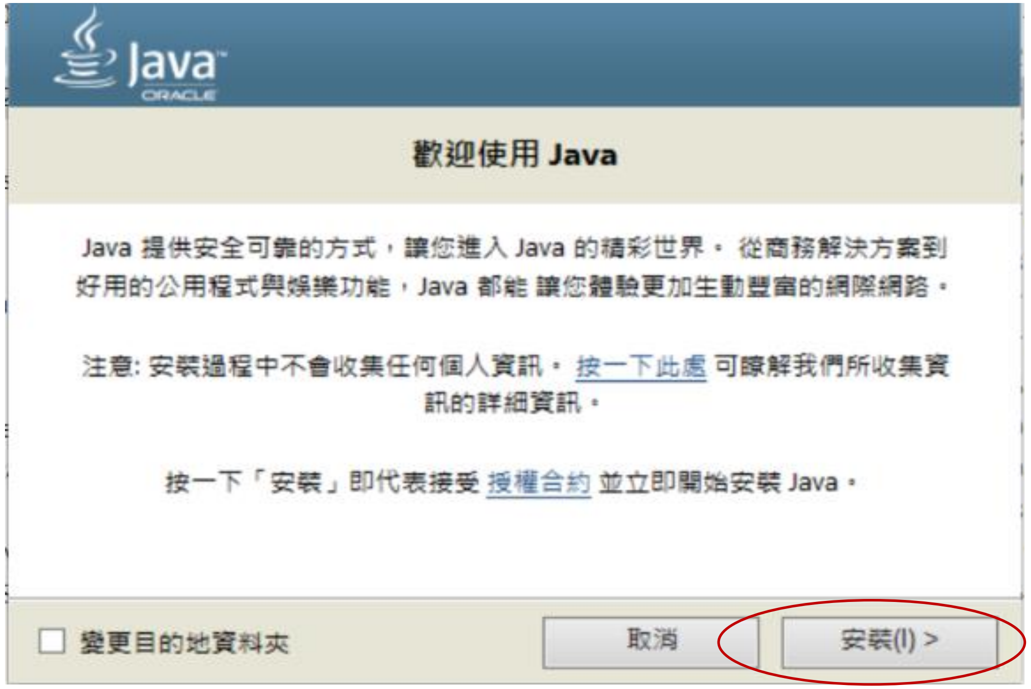
Windows 線上
檔案大小： 571 KB
指示

Windows 離線
檔案大小： 47.4 MB
指示

Windows 離線 (64 位元)
檔案大小： 53.9 MB
指示

Access more information if you fail during installation

Windows installation



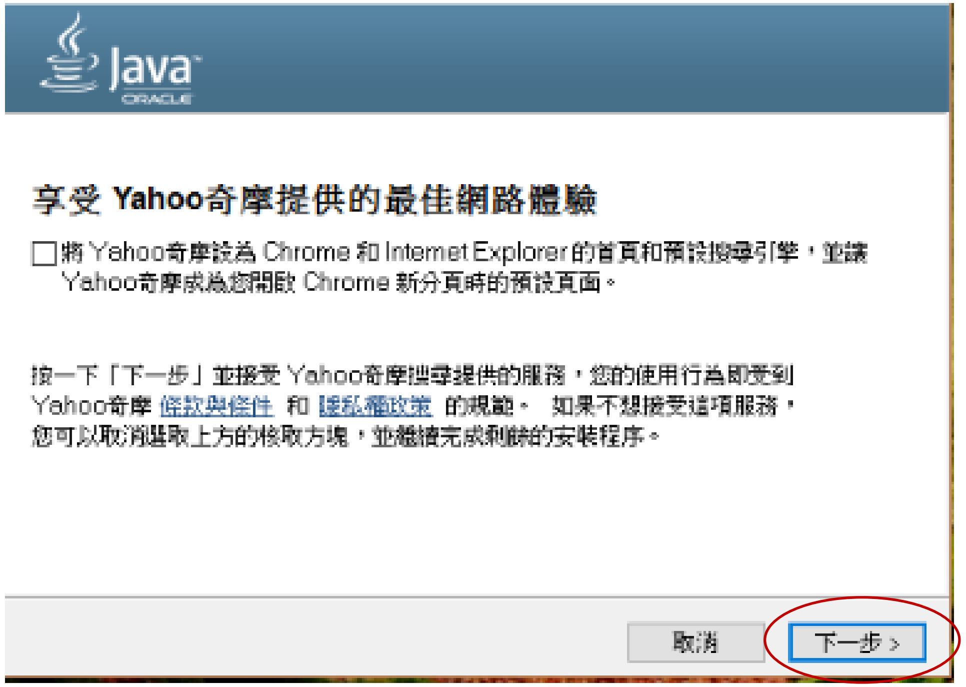
歡迎使用 Java

Java 提供安全可靠的方式，讓您進入 Java 的精彩世界。 從商務解決方案到好用的公用程式與娛樂功能，Java 都能 讓您體驗更加生動豐富的網際網路。

注意: 安裝過程中不會收集任何個人資訊。 按一下此處 可瞭解我們所收集資訊的詳細資訊。

按一下「安裝」即代表接受 授權合約 並立即開始安裝 Java。

□ 變更目的地資料夾

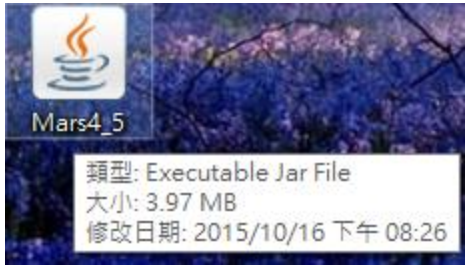


享受 Yahoo奇摩提供的最佳網路體驗

□將 Yahoo奇摩設為 Chrome 和 Internet Explorer 的首頁和預設搜尋引擎，並讓 Yahoo奇摩成為您開啟 Chrome 新分頁時的預設頁面。

按一下「下一步」並接受 Yahoo奇摩搜尋提供的服務，您的使用行為即受到 Yahoo奇摩 條款與條件 和 隱私權政策 的規範。 如果不想接受這項服務，您可以取消選取上方的核取方塊，並繼續完成剩餘的安裝程序。

- After installation, open Mars by double clicks Mars4_5

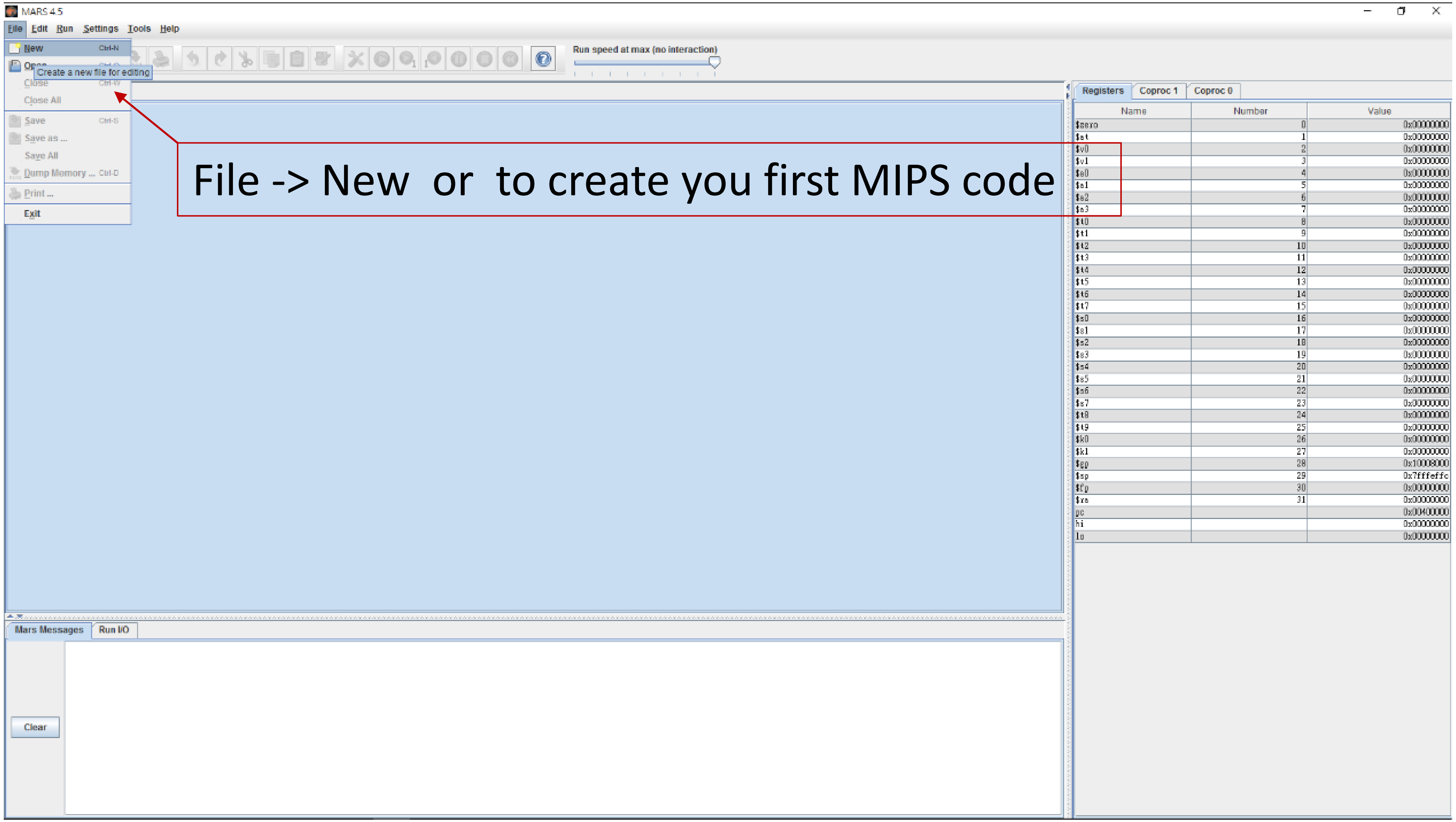


類型: Executable Jar File
大小: 3.97 MB
修改日期: 2015/10/16 下午 08:26

JRE for linux

- Use the command:
  $ java --version
- If not installed yet:
  - Debian based distribution (Ubuntu, Mint, etc…) users can open package manager:
    $ sudo apt-get install default-jre
  - Or follow this page:
    https://java.com/zh_TW/download/help/linux_x64rpm_install.xml
- After installation, open MARS by the command:
  $ java -jar Mars4_5.jar

First glance on MARS



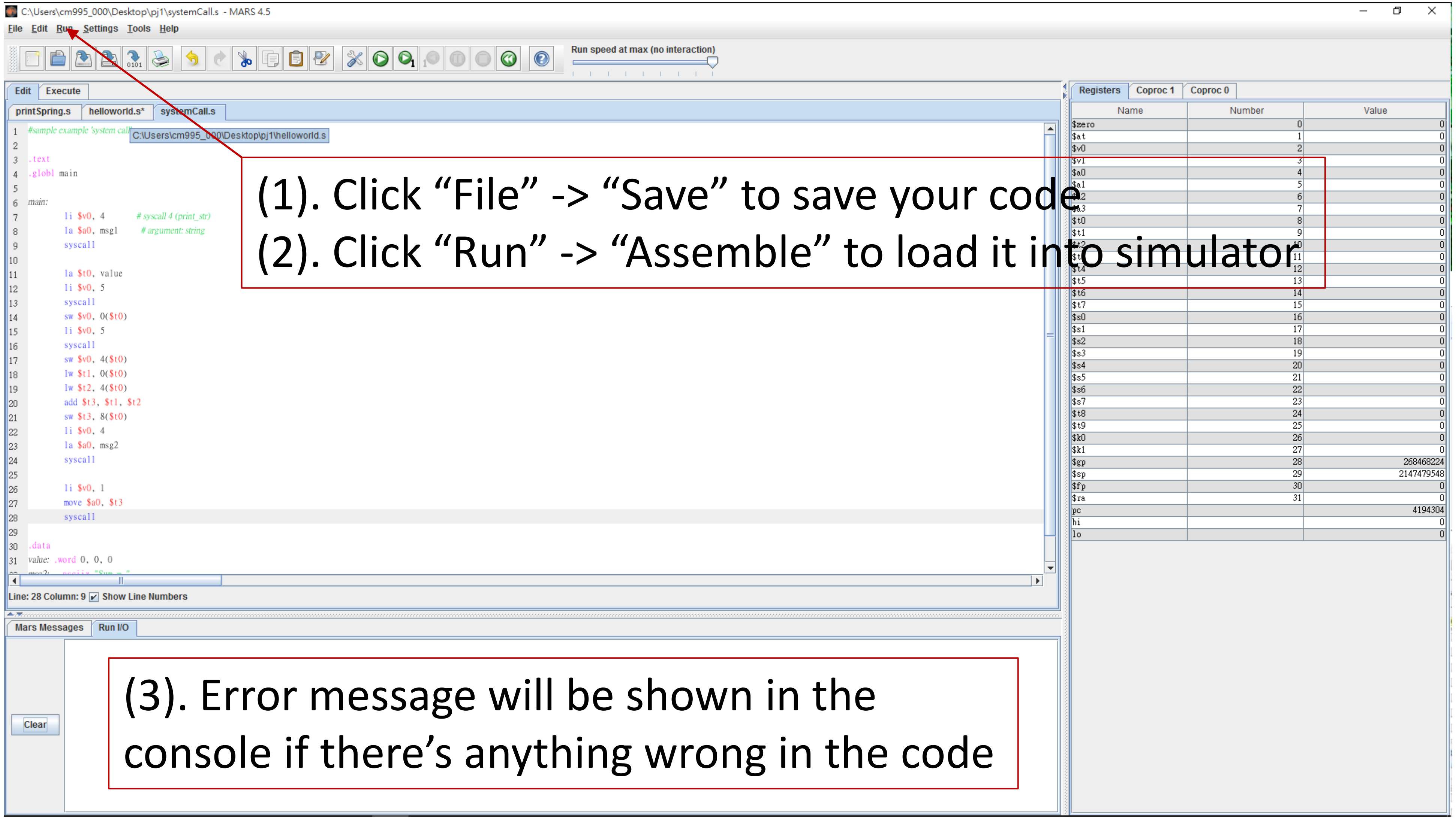
Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	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		0x00400000
hi		0x00000000
lo		0x00000000

File -> New or to create you first MIPS code

Sample code: add.asm

```
.text
.globl main
main:
        li $v0, 4             # prepare syscall 4 (print string)
        la $a0, msg1    # argument: msg1
        syscall
        la $t0, value    # get address of 'value seg' and we will store
integers into value[0~2] later
        li $v0, 5            # prepare syscall 5 (get int)
        syscall
        sw $v0, 0($t0) # store first int from syscall to value[0]
        li $v0, 5            # prepare syscall 5 (get int)
        syscall
        sw $v0, 4($t0)      # store second int from syscall to value[1]
        lw $t1, 0($t0)            # load first int back
        lw $t2, 4($t0)            # laod second int back
add $t3, $t1, $t2      # put sum to $t3
        sw $t3, 8($t0)           # store sum to value[2]
        li $v0, 4                    # prepare syscall 4 (print string)
        la $a0, msg2             # argument: msg2
        syscall
        li $v0, 1                    # prepare syscall 1 (print int)
        move $a0, $t3           # argument: sum
        syscall
 exit:
         li      $v0, 10             # terminate program run and
         syscall                    # Exit
.data
value: .word 0, 0, 0
msg2:  .asciiz "Sum = "
msg1:   .asciiz "Please give 2 integers separated by enter:\n "
```

Development Environment



C:\Users\cm995_000\Desktop\pj1\systemCall.s - MARS 4.5
File
Edit
Run
Settings
Tools
Help
Run speed at max (no interaction)
Edit
Execute
printSpring.s
helloworld.s*
systemCall.s
C:\Users\cm995_000\Desktop\pj1\helloworld.s
Registers
Coproc 1
Coproc 0
Mars Messages
Run I/O
Clear

```
#sample example 'system call'

.text
.globl main

main:
        li $v0, 4          # syscall 4 (print_str)
        la $a0, msg1       # argument: string
        syscall

        la $t0, value
        li $v0, 5
        syscall
        sw $v0, 0($t0)
        li $v0, 5
        syscall
        sw $v0, 4($t0)
        lw $t1, 0($t0)
        lw $t2, 4($t0)
        add $t3, $t1, $t2
        sw $t3, 8($t0)
        li $v0, 4
        la $a0, msg2
        syscall

        li $v0, 1
        move $a0, $t3
        syscall

.data
value: .word 0, 0, 0
msg2:  .asciiz "Sum = "
```

Line: 28 Column: 9 ☑ Show Line Numbers

Name	Number	Value
$zero	0	0
$at	1	0
$v0	2	0
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	0
$t1	9	0
$t2	10	0
$t3	11	0
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	0
$s1	17	0
$s2	18	0
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	0
pc		4194304
hi		0
lo		0

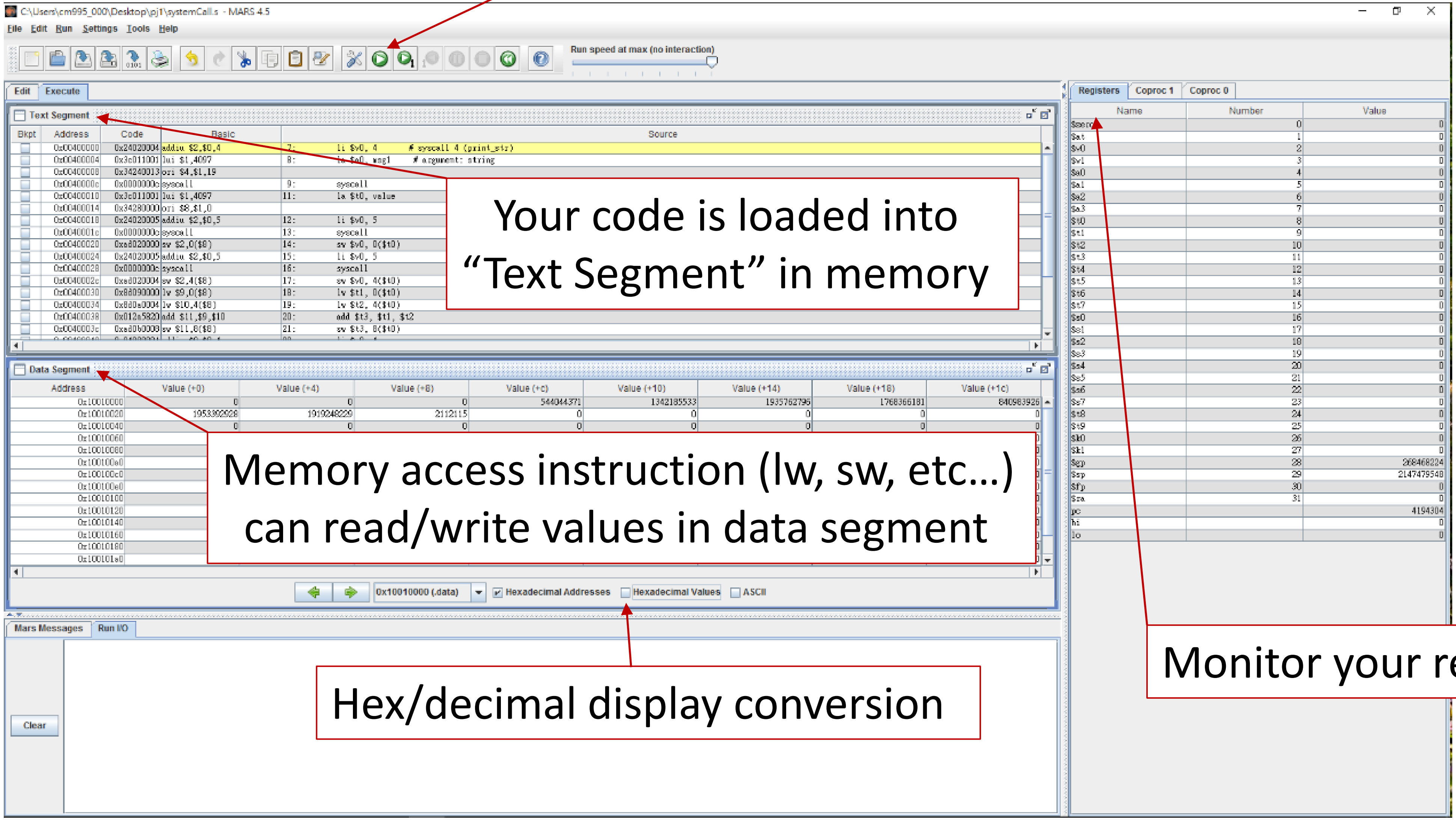
(1). Click “File” -> “Save” to save your code

(2). Click “Run” -> “Assemble” to load it into simulator

(3). Error message will be shown in the console if there’s anything wrong in the code

After assemble

Click run button to start simulation



Bkpt	Address	Code	Basic		Source
□	0x00400000	0x24020004	addiu $2,$0,4	7:	li $v0, 4 # syscall 4 (print_str)
□	0x00400004	0x3c011001	lui $1,4097	8:	la $a0, msg1 # argument: string
□	0x00400008	0x34240013	ori $4,$1,19		
□	0x0040000c	0x0000000c	syscall	9:	syscall
□	0x00400010	0x3c011001	lui $1,4097	11:	la $t0, value
□	0x00400014	0x34280000	ori $8,$1,0		
□	0x00400018	0x24020005	addiu $2,$0,5	12:	li $v0, 5
□	0x0040001c	0x0000000c	syscall	13:	syscall
□	0x00400020	0xad020000	sw $2,0($8)	14:	sw $v0, 0($t0)
□	0x00400024	0x24020005	addiu $2,$0,5	15:	li $v0, 5
□	0x00400028	0x0000000c	syscall	16:	syscall
□	0x0040002c	0xad020004	sw $2,4($8)	17:	sw $v0, 4($t0)
□	0x00400030	0x8d090000	lw $9,0($8)	18:	lw $t1, 0($t0)
□	0x00400034	0x8d0a0004	lw $10,4($8)	19:	lw $t2, 4($t0)
□	0x00400038	0x012a5820	add $11,$9,$10	20:	add $t3, $t1, $t2
□	0x0040003c	0xad0b0008	sw $11,8($8)	21:	sw $t3, 8($t0)

Your code is loaded into “Text Segment” in memory

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0	0	0	544044371	1342185533	1935762796	1768366181	840983926
0x10010020	1953392928	1919248229	2112115	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060								0
0x10010080								0
0x100100a0								0
0x100100c0								0
0x100100e0								0
0x10010100								0
0x10010120								0
0x10010140								0
0x10010160								0
0x10010180								0
0x100101a0								0

Memory access instruction (lw, sw, etc…) can read/write values in data segment

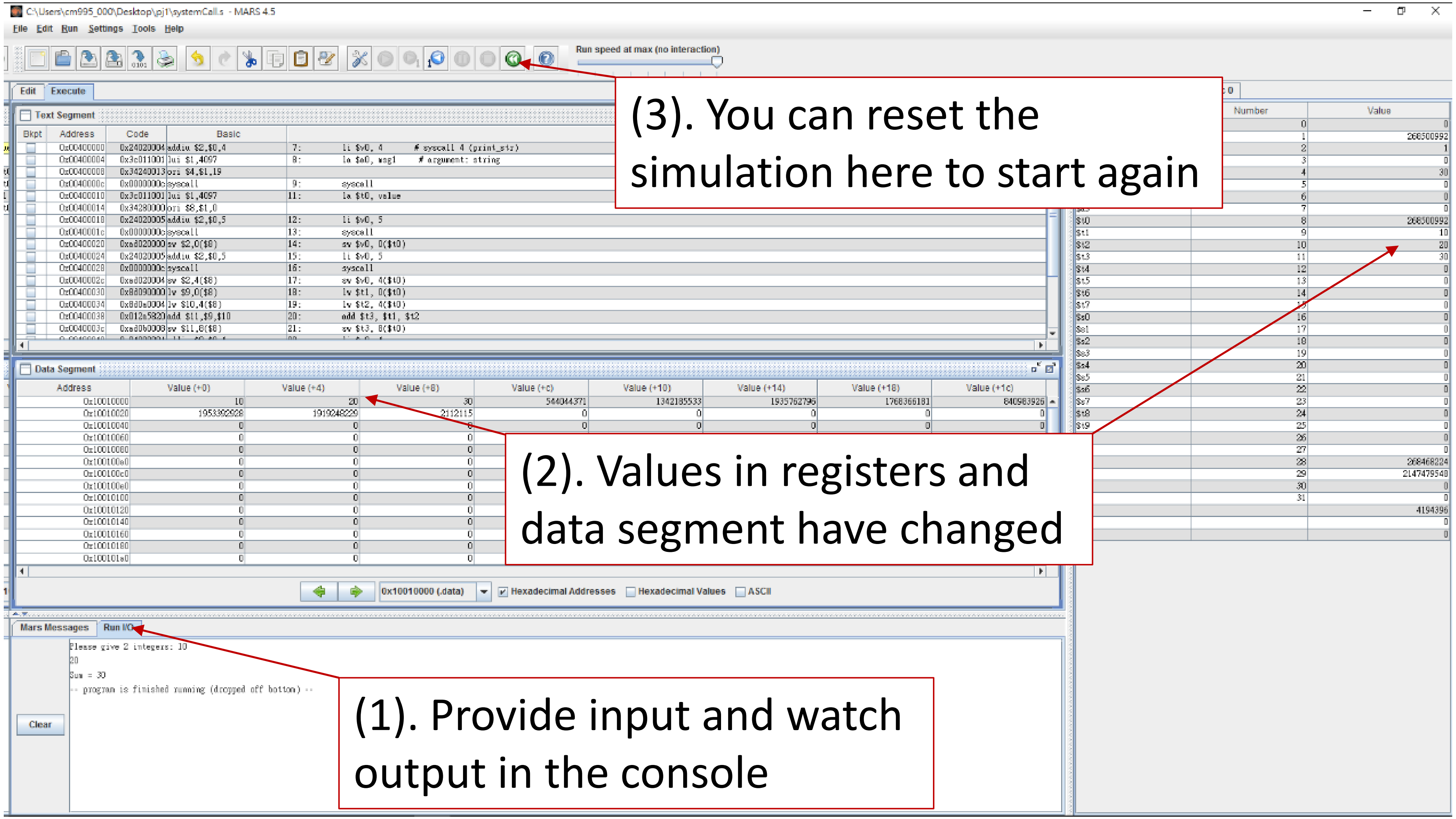
0x10010000 (.data)   ☑ Hexadecimal Addresses   □ Hexadecimal Values   □ ASCII

Name	Number	Value
$zero	0	0
$at	1	0
$v0	2	0
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	0
$t1	9	0
$t2	10	0
$t3	11	0
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	0
$s1	17	0
$s2	18	0
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	0
pc		4194304
hi		0
lo		0

Hex/decimal display conversion

Monitor your register file here

After simulation started



C:\Users\cm995_000\Desktop\pj1\systemCall.s - MARS 4.5

Run speed at max (no interaction)

Text Segment

Bkpt	Address	Code	Basic	
□	0x00400000	0x24020004	addiu $2,$0,4	7: li $v0, 4 # syscall 4 (print_str)
□	0x00400004	0x3c011001	lui $1,4097	8: la $a0, msg1 # argument: string
□	0x00400008	0x34240013	ori $4,$1,19	
□	0x0040000c	0x0000000c	syscall	9: syscall
□	0x00400010	0x3c011001	lui $1,4097	11: la $t0, value
□	0x00400014	0x34280000	ori $8,$1,0	
□	0x00400018	0x24020005	addiu $2,$0,5	12: li $v0, 5
□	0x0040001c	0x0000000c	syscall	13: syscall
□	0x00400020	0xad020000	sw $2,0($8)	14: sw $v0, 0($t0)
□	0x00400024	0x24020005	addiu $2,$0,5	15: li $v0, 5
□	0x00400028	0x0000000c	syscall	16: syscall
□	0x0040002c	0xad020004	sw $2,4($8)	17: sw $v0, 4($t0)
□	0x00400030	0x8d090000	lw $9,0($8)	18: lw $t1, 0($t0)
□	0x00400034	0x8d0a0004	lw $10,4($8)	19: lw $t2, 4($t0)
□	0x00400038	0x012a5820	add $11,$9,$10	20: add $t3, $t1, $t2
□	0x0040003c	0xad0b0008	sw $11,8($8)	21: sw $t3, 8($t0)

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	10	20	30	544044371	1342185533	1935762796	1768366181	840983926
0x10010020	1953392928	1919248229	2112115	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0					
0x10010080	0	0	0					
0x100100a0	0	0	0					
0x100100c0	0	0	0					
0x100100e0	0	0	0					
0x10010100	0	0	0					
0x10010120	0	0	0					
0x10010140	0	0	0					
0x10010160	0	0	0					
0x10010180	0	0	0					
0x100101a0	0	0	0					

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

	Number	Value
	0	0
	1	268500992
	2	1
	3	0
	4	30
	5	0
	6	0
	7	0
$t0	8	268500992
$t1	9	10
$t2	10	20
$t3	11	30
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	0
$s1	17	0
$s2	18	0
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
	26	0
	27	0
	28	268468224
	29	2147479548
	30	0
	31	0
		4194396
		0
		0

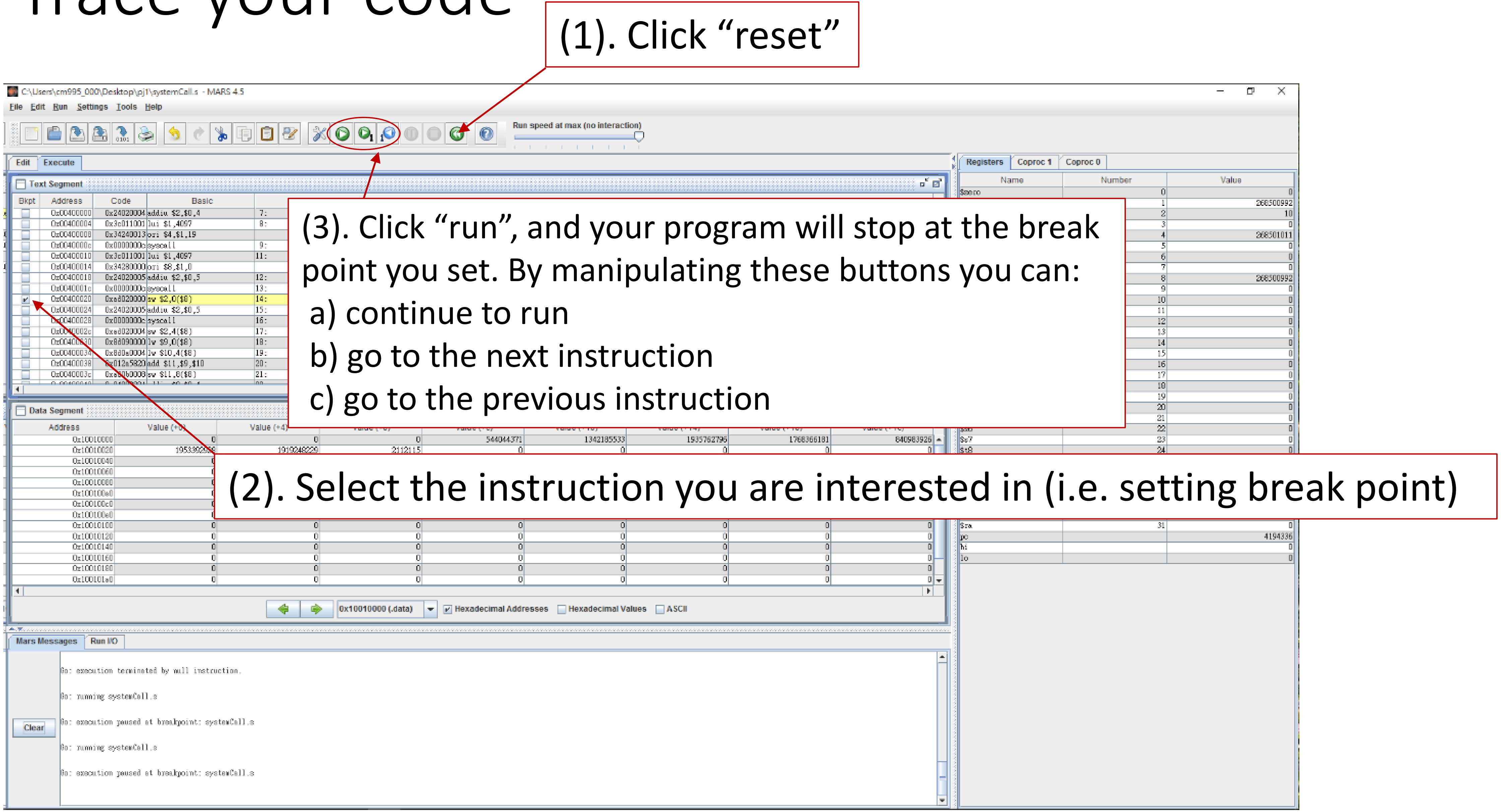
```
Please give 2 integers: 10
20
Sum = 30
-- program is finished running (dropped off bottom) --
```

(3). You can reset the simulation here to start again

(2). Values in registers and data segment have changed

(1). Provide input and watch output in the console

Trace your code



(1). Click “reset”

(2). Select the instruction you are interested in (i.e. setting break point)

(3). Click “run”, and your program will stop at the break point you set. By manipulating these buttons you can:

a) continue to run

b) go to the next instruction

c) go to the previous instruction

C:\Users\cm995_000\Desktop\pj1\systemCall.s - MARS 4.5
File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Text Segment
Data Segment
Registers Coproc 1 Coproc 0
Mars Messages Run I/O
Clear

Bkpt	Address	Code	Basic	
□	0x00400000	0x24020004	addiu $2,$0,4	7:
□	0x00400004	0x3c011001	lui $1,4097	8:
□	0x00400008	0x34240013	ori $4,$1,19	
□	0x0040000c	0x0000000c	syscall	9:
□	0x00400010	0x3c011001	lui $1,4097	11:
□	0x00400014	0x34280000	ori $8,$1,0	
□	0x00400018	0x24020005	addiu $2,$0,5	12:
□	0x0040001c	0x0000000c	syscall	13:
☑	0x00400020	0xad020000	sw $2,0($8)	14:
□	0x00400024	0x24020005	addiu $2,$0,5	15:
□	0x00400028	0x0000000c	syscall	16:
□	0x0040002c	0xad020004	sw $2,4($8)	17:
□	0x00400030	0x8d090000	lw $9,0($8)	18:
□	0x00400034	0x8d0a0004	lw $10,4($8)	19:
□	0x00400038	0x012a5820	add $11,$9,$10	20:
□	0x0040003c	0xad0b0008	sw $11,8($8)	21:

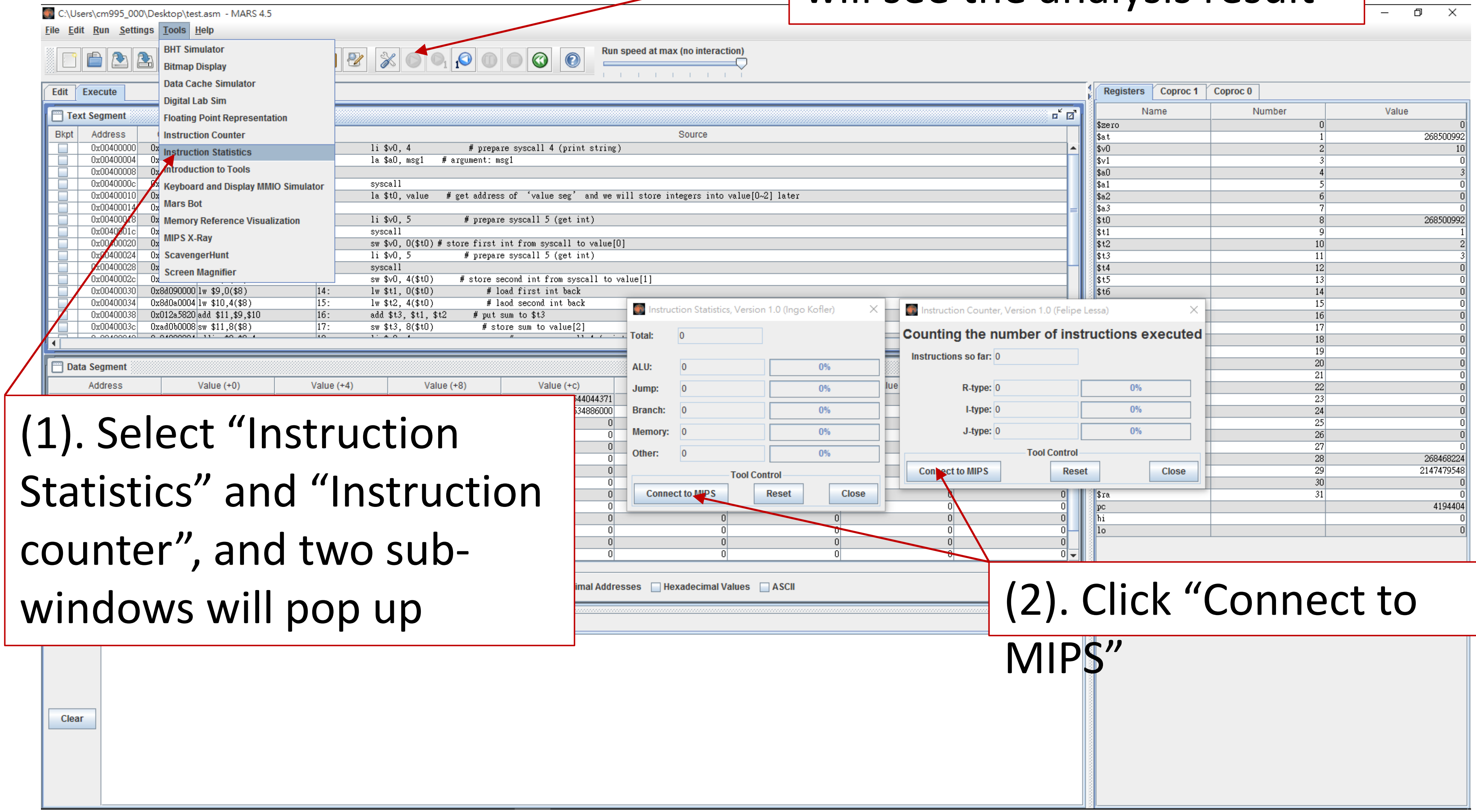
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0	0	0	544044371	1342185533	1935762796	1768366181	840983926
0x10010020	1953392[illegible]	1919248229	2112115	0	0	0	0	0
0x10010040	0							
0x10010060	0							
0x10010080	0							
0x100100a0	0							
0x100100c0	0							
0x100100e0	0							
0x10010100	0	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0	0
0x10010140	0	0	0	0	0	0	0	0
0x10010160	0	0	0	0	0	0	0	0
0x10010180	0	0	0	0	0	0	0	0
0x100101a0	0	0	0	0	0	0	0	0

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

Name	Number	Value
$zero	0	0
	1	268500992
	2	10
	3	0
	4	268501011
	5	0
	6	0
	7	0
	8	268500992
	9	0
	10	0
	11	0
	12	0
	13	0
	14	0
	15	0
	16	0
	17	0
	18	0
	19	0
	20	0
	21	0
	22	0
$s7	23	0
$t8	24	0
$ra	31	0
pc		4194336
hi		0
lo		0

```
Go: execution terminated by null instruction.

Go: running systemCall.s

Go: execution paused at breakpoint: systemCall.s

Go: running systemCall.s

Go: execution paused at breakpoint: systemCall.s
```

Analyze your code



```
Bkpt  Address     Code        Basic           Source
      0x00400000  0x                              li $v0, 4             # prepare syscall 4 (print string)
      0x00400004  0x                              la $a0, msg1    # argument: msg1
      0x00400008  0x
      0x0040000c  0x                              syscall
      0x00400010  0x                              la $t0, value    # get address of ‘value seg’ and we will store integers into value[0~2] later
      0x00400014  0x
      0x00400018  0x                              li $v0, 5            # prepare syscall 5 (get int)
      0x0040001c  0x                              syscall
      0x00400020  0x                              sw $v0, 0($t0) # store first int from syscall to value[0]
      0x00400024  0x                              li $v0, 5            # prepare syscall 5 (get int)
      0x00400028  0x                              syscall
      0x0040002c  0x                              sw $v0, 4($t0)       # store second int from syscall to value[1]
      0x00400030  0x8d090000  lw $9,0($8)     14:   lw $t1, 0($t0)             # load first int back
      0x00400034  0x8d0a0004  lw $10,4($8)    15:   lw $t2, 4($t0)             # laod second int back
      0x00400038  0x012a5820  add $11,$9,$10  16:   add $t3, $t1, $t2      # put sum to $t3
      0x0040003c  0xad0b0008  sw $11,8($8)    17:   sw $t3, 8($t0)             # store sum to value[2]
```

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)

Instruction Statistics, Version 1.0 (Ingo Kofler)

Total:	0	
ALU:	0	0%
Jump:	0	0%
Branch:	0	0%
Memory:	0	0%
Other:	0	0%

Tool Control

Connect to MIPS | Reset | Close

Instruction Counter, Version 1.0 (Felipe Lessa)

Counting the number of instructions executed

Instructions so far: 0

R-type:	0	0%
I-type:	0	0%
J-type:	0	0%

Tool Control

Connect to MIPS | Reset | Close

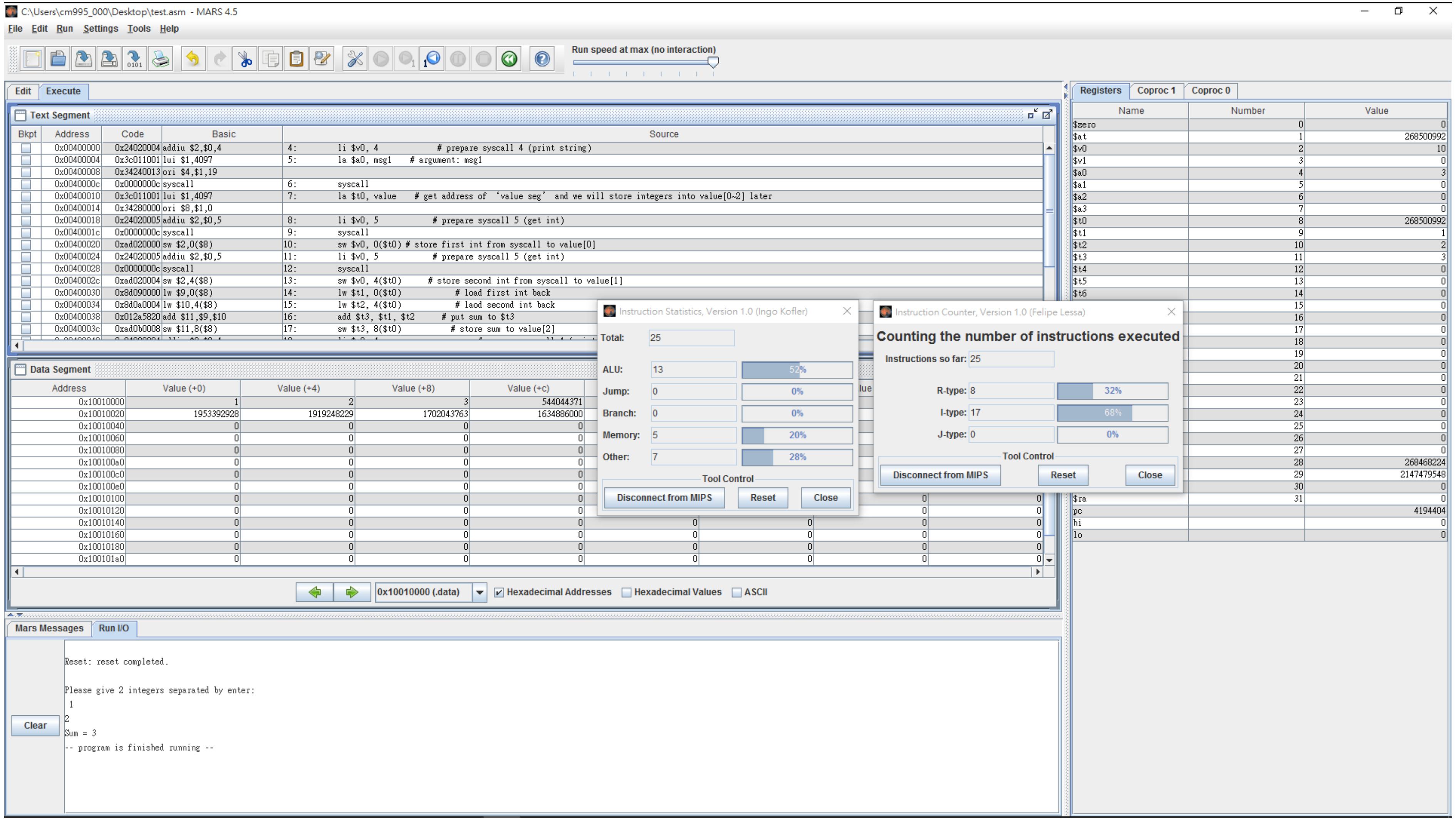
Name	Number	Value
$zero	0	0
$at	1	268500992
$v0	2	10
$v1	3	0
$a0	4	3
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	268500992
$t1	9	1
$t2	10	2
$t3	11	3
$t4	12	0
$t5	13	0
$t6	14	0
	15	0
	16	0
	17	0
	18	0
	19	0
	20	0
	21	0
	22	0
	23	0
	24	0
	25	0
	26	0
	27	0
	28	268468224
	29	2147479548
	30	0
$ra	31	0
pc		4194404
hi		0
lo		0

(3). Click “RUN” and you will see the analysis result

(1). Select “Instruction Statistics” and “Instruction counter”, and two sub-windows will pop up

(2). Click “Connect to MIPS”

Analysis results (if terminated normally)



Bkpt	Address	Code	Basic	Source
☐	0x00400000	0x24020004	addiu $2,$0,4	4: li $v0, 4 # prepare syscall 4 (print string)
☐	0x00400004	0x3c011001	lui $1,4097	5: la $a0, msg1 # argument: msg1
☐	0x00400008	0x34240013	ori $4,$1,19	
☐	0x0040000c	0x0000000c	syscall	6: syscall
☐	0x00400010	0x3c011001	lui $1,4097	7: la $t0, value # get address of ‘value seg’ and we will store integers into value[0~2] later
☐	0x00400014	0x34280000	ori $8,$1,0	
☐	0x00400018	0x24020005	addiu $2,$0,5	8: li $v0, 5 # prepare syscall 5 (get int)
☐	0x0040001c	0x0000000c	syscall	9: syscall
☐	0x00400020	0xad020000	sw $2,0($8)	10: sw $v0, 0($t0) # store first int from syscall to value[0]
☐	0x00400024	0x24020005	addiu $2,$0,5	11: li $v0, 5 # prepare syscall 5 (get int)
☐	0x00400028	0x0000000c	syscall	12: syscall
☐	0x0040002c	0xad020004	sw $2,4($8)	13: sw $v0, 4($t0) # store second int from syscall to value[1]
☐	0x00400030	0x8d090000	lw $9,0($8)	14: lw $t1, 0($t0) # load first int back
☐	0x00400034	0x8d0a0004	lw $10,4($8)	15: lw $t2, 4($t0) # laod second int back
☐	0x00400038	0x012a5820	add $11,$9,$10	16: add $t3, $t1, $t2 # put sum to $t3
☐	0x0040003c	0xad0b0008	sw $11,8($8)	17: sw $t3, 8($t0) # store sum to value[2]

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)
0x10010000	1	2	3	544044371
0x10010020	1953392928	1919248229	1702043763	1634886000
0x10010040	0	0	0	0
0x10010060	0	0	0	0
0x10010080	0	0	0	0
0x100100a0	0	0	0	0
0x100100c0	0	0	0	0
0x100100e0	0	0	0	0
0x10010100	0	0	0	0
0x10010120	0	0	0	0
0x10010140	0	0	0	0
0x10010160	0	0	0	0
0x10010180	0	0	0	0
0x100101a0	0	0	0	0

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

Name	Number	Value
$zero	0	0
$at	1	268500992
$v0	2	10
$v1	3	0
$a0	4	3
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	268500992
$t1	9	1
$t2	10	2
$t3	11	3
$t4	12	0
$t5	13	0
$t6	14	0
	15	0
	16	0
	17	0
	18	0
	19	0
	20	0
	21	0
	22	0
	23	0
	24	0
	25	0
	26	0
	27	0
	28	268468224
	29	2147479548
	30	0
$ra	31	0
pc		4194404
hi		0
lo		0

Instruction Statistics, Version 1.0 (Ingo Kofler)

Total: 25

ALU: 13 52%

Jump: 0 0%

Branch: 0 0%

Memory: 5 20%

Other: 7 28%

Tool Control: Disconnect from MIPS | Reset | Close

Instruction Counter, Version 1.0 (Felipe Lessa)

Counting the number of instructions executed

Instructions so far: 25

R-type: 8 32%

I-type: 17 68%

J-type: 0 0%

Tool Control: Disconnect from MIPS | Reset | Close

```
Reset: reset completed.

Please give 2 integers separated by enter:
 1
2
Sum = 3
-- program is finished running --
```

Hints

- Program needs to call “exit” at the end of execution, otherwise unexpected error might occurs.
- You must write comments to explain your code very carefully.
- You might try other features in MARS to enrich your report and get bonus.
- Any one copying codes from the internet or classmates will get 0.