## POST-LAB 8

NAME - RAJENDRA SINGH ROLL NO. 111601017

#### **QUESTION (1) Introduction to procedures:**

Given two numbers 'x' and 'y', create a program to compute x y (i.e., x to the power y).

The inputs need not be taken during run time. The operation x y should be implemented as a Procedure. The values 'x' and 'y' should be passed as arguments to the function.

# CODE WITHOUT NOP:

```
.data #DATA
```

x: .word 6 y: .word 3

answer: .word 0

.text .globl main .ent main Main:

#MAIN

```
lw $a0 , x
lw $a1 , y
jal power
move $t5, $v0
```

li \$v0, 10 syscall #FUNCTION POWER CALLED

.end main

```
.globl power
.ent power
power:
     li $v0,1
```

li \$t0,0 powLoop: **#POWER LOOP** mul \$v0, \$v0, \$a0 add \$t0, \$t0, 1 blt \$t0, \$a1, powLoop jr \$ra

.end power

FP Re	gs	nt Regs [10]		Data	Text		
Int Regs	[10]		0×	Text			
PC	=	4194368					User Text Segment [00400000][00440000]
EPC	=	0		[00400000]	8fa40000	lw \$4, 0(\$29)	; 183: 1w \$a0 0(\$sp) # argc
Cause	=	0				addiu \$5, \$29, 4	; 184: addiu \$a1 \$sp 4 # argv
BadVAd	dr =	0		[00400008]			; 185: addiu \$a2 \$a1 4 # envp
Status	=	805371664		The state of the s		sl1 \$2, \$4, 2	; 186: sll \$v0 \$a0 2
				[00400010]	00c23021	addu \$6, \$6, \$2	; 187: addu \$a2 \$a2 \$v0
HI	=	0				jal 0x00400024 [main]	; 188: jal main
LO	=	216		[00400018]	00000000	nop	; 189: nop
( in a 1 ft)						ori \$2, \$0, 10	; 191: 11 \$v0 10
R0 [r	0] =	0		[00400020]			; 192: syscall # syscall 10 (exit)
	t] =			[00400024]	3c011001	lui \$1, 4097	; 12: 1w \$a0 , x
R2 [v	0] =	10		[00400028]	8c240000	lw \$4, 0(\$1)	
R3 [v	1] =	0		[0040002c]	3c011001	lui \$1, 4097	; 13: 1w \$a1 , y
R4 [a	0] =	6		[00400030]	8c250004	lw \$5, 4(\$1)	
R5 [a	1] =	3		[00400034]	0c100011	jal 0x00400044 [power]	; 14: jal power
R6 [a	2] =	2147481984		[00400038]	00026821	addu \$13, \$0, \$2	; 15: move \$t5, \$v0
R7 [a	3] =	0		[0040003c]	3402000a	ori \$2, \$0, 10	; 17: 1i \$v0, 10
R8 [t	0] =	3		[00400040]	0000000c	syscall	; 18: syscall
R9 [t	1] =	0		[00400044]	34020001	ori \$2, \$0, 1	; 26: 1i \$v0,1
R10 [t	2] =	0		[00400048]	34080000	ori \$8, \$0, 0	; 27: 1i \$t0,0
R11 [t	3] =	0		[0040004c]	70441002	mul \$2, \$2, \$4	; 30: mul \$v0, \$v0, \$a0
R12 [t	4] =	0		[00400050]	21080001	addi \$8, \$8, 1	; 31: add \$t0, \$t0, 1
R13 [t	5] =	216		[00400054]	0105082a	slt \$1, \$8, \$5	; 32: blt \$t0, \$a1, powLoop
R14 [t	6] =	0		[00400058]	1420fffd	bne \$1, \$0, -12 [powLoop	0-0x00400058]
R15 [t	7] =	0		[0040005c]	03e00008	jr \$31	; 33: jr \$ra
R16 [s	0] =	0		CSS 111 (A)			
R17 [s	1] =	0					Kernel Text Segment [80000000][80010000]
R18 [s	2] =	0				addu \$27, \$0, \$1	; 90: move \$k1 \$at # Save \$at
R19 [s	3] =	0		[80000184]	3c019000	lui \$1, -28672	; 92: sw \$v0 sl # Not re-entrant and we can't trust \$sp
R20 [s	4] =	0		[80000188]	ac220200	sw \$2, 512(\$1)	
R21 [s	5] =	0		[8000018c]	3c019000	lui \$1, -28672	; 93: sw \$a0 s2 # But we need to use these registers
R22 [s	6] =	0	6.60	[80000190]	ac240204	sw \$4, 516(\$1)	
R23 [s	7] =	0	1000	[80000194]	401a6800	mfc0 \$26, \$13	; 95: mfc0 \$k0 \$13 # Cause register
R24 [t	8] =	0		[80000198]	001a2082	srl \$4, \$26, 2	; 96: srl \$a0 \$k0 2 # Extract ExcCode Field
R25 [t	9] =	0		[8000019c]	3084001f	andi \$4, \$4, 31	; 97: andi \$a0 \$a0 0x1f
R26 [k	0] =	0				ori \$2, \$0, 4	; 101: 1i \$v0 4 # syscall 4 (print_str)
R27 [k	1] =	0	v	[800001a4]	3c049000	lui \$4, -28672 [m1_]	; 102: 1a \$a0m1_

-		
FP Regs	nt Regs [10]	Data Text
Int Regs [10]	<b>② ※</b>	Data
EPC =	4194368 0	User data segment [10000000][10040000] [10000000][1000ffff] 00000000 [10010000] 00000006 00000003 00000000 00000000
BadVAddr =		[10010010][1003ffff] 00000000
	0 216	User Stack [7ffff974][80000000] [7ffff974] 00000001 7ffffa3a 00000000
R0 [r0] = R1 [at] =	0	[7fffff990] 7ffffffa9 7ffffffa1 7ffffff8d 7ffffff7b
R2 [v0] = R3 [v1] = R4 [a0] =	0	[7ffff9b0] 7ffffe8f 7ffffe7e 7ffffe6b 7ffffe39~k9 [7ffff9c0] 7ffffe28 7ffffe16 7ffffe09 7ffffd74 (t [7ffff9d0] 7ffffd55 7ffffd4a 7ffffd3f 7ffffd25 UJ?%
R5 [a1] =		[7ffff9e0] 7ffffd11 7ffffc67 7ffffcce 7ffffca5
R7 [a3] = R8 [t0] =	3	[7ffffa00] 7ffffc54 7ffffc41 7ffffc2b 7ffffbff T A + [7ffffa10] 7ffffb96 7ffffb6a 7ffffb53 7ffffb03 j S
R9 [t1] = R10 [t2] = R11 [t3] =	0	[7ffffa20] 7ffffa22 7ffffa9f 7ffffa87 7ffffa66f [7ffffa30] 00000000 00000000 682f0000 2f656d6f/home/ [7ffffa40] 656a6172 6172646e 7365442f 706f746b rajendra/Desktop
R12 [t4] = R13 [t5] =	0	[7ffffa50] 326f632f 2f303136 3842414c 2f57454e / c o 2 6 1 0 / L A B 8 N E W / [7ffffa60] 73612e31 4a47006d 45445f53 5f475542 1 . a s m . G J S _ D E B U G _
R14 [t6] = R15 [t7] =	0	[7ffffa70] 49504f54 4a3d5343 52452053 3b524f52 T O P I C S = J S E R R O R; [7ffffa80] 4c20534a 4700474f 445f534a 47554245 J S L O G . G J S _ D E B U G
R16 [s0] = R17 [s1] = R18 [s2] =	0	[7ffffa90] 54554f5f 3d545550 65647473 47007272 _ O U T P U T = s t d e r r . G [7ffffaa0] 4c5f4f49 434e5541 5f444548 4b534544 I O _ L A U N C H E D _ D E S K [7ffffab0] 5f504f54 454c4946 4449505f 3038323d T O P F I L E P I D = 2 8 0
R19 [s3] = R20 [s4] =	0	[7ffffac0] 49470031 414c5f4f 48434e55 445f4445 1 . G I O _ L A U N C H E D _ D [7ffffad0] 544b5345 465f504f 3d454c49 7273752f E S K T O P _ F I L E = / u s r
R21 [s5] = R22 [s6] = R23 [s7] =	0	[7ffffae0] 6168732f 612f6572 696c7070 69746163 / share/applicati [7ffffaf0] 2f736e6f 70737471 642e6d69 746b7365 ons/qtspim.deskt [7ffffb00] 5300706f 49535345 4d5f4e4f 47414e41 op.SESSION_MANAG
R24 [t8] = R25 [t9] =	0	[7ffffb10] 6c3d5245 6c61636f 6e69532f 403a6867 ER = 1 o c a 1 / S i n g h : @ [7ffffb20] 706d742f 43492e2f 6e752d45 312f7869 / t m p / . I C E - u n i x / 1
R26 [k0] = R27 [k1] =		[7ffffb30] 2c313531 78696e75 6e69532f 2f3a6867 151,unix/Singh:/ [7ffffb40] 2f706d74 4543492e 696e752d 31312f78 tmp/.ICE-unix/11

CODVITOR 1990-2017 By Dames Larus.

- (2) Illustration of delayed branch:
- Note: In delayed branches, the instruction just next to branch instructions get executed irrespective of the branch decision.
- (a) Write a program to compute the sum of squares of numbers from 1 to 'n'. The number 'n' can be either given during run time or can be directly given in the code. Use only 'main' procedure. Verify the output.
- Reorder the code to efficiently utilise the branch delay slot. You may reorder the code or use 'nop' instruction. Use the efficient option. Run your code again and verify the output.

  (b) Run the same program by enabling 'delayed branch' and verify the output.
- To enable delayed branch go to, Simulator → Settings → M

### Simulator $\rightarrow$ Settings $\rightarrow$ MIPS tab $\rightarrow$ Enable Delayed Branches. Explain your observations.

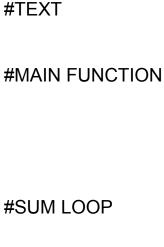
A delay slot is an instruction slot that gets executed without the effects of a preceding instruction. The most common form is a single arbitrary instruction located immediately after a branch instruction on a RISC or DSP architecture; this instruction will execute even if the preceding branch is taken. Thus, by design, the instructions appear to execute in an illogical or incorrect order. It is typical for assemblers to automatically reorder instructions by default, hiding the awkwardness from assembly developers and compilers.

## CODE **WITHOUT** NOP:

#### .word 5 n: sumOfSquares: .word 0 .text Main: lw li

li \$v0, 10 syscall .end main

.data



**#DATA** 

- **#CONDITION CHECK #MAIN END**
- li \$t2,0 sumLoop: mul \$t3, \$t1, \$t1 add \$t2, \$t2, \$t3 add \$t1, \$t1, 1

ble \$t1, \$t0, sumLoop sw \$t2, sumOfSquares

- \$t0, n \$t1,1
- .globl main

```
nt Regs [10]
FP Regs
                                      Data
                                                Text
Int Regs [10]
                              PC
         = 4194388
                                                                                   User Text Segment [00400000]..[00440000]
                                   [00400000] 8fa40000 lw $4, 0($29)
                                                                                 ; 183: 1w $a0 0($sp) # argc
EPC
         = 0
         = 0
                                   [00400004] 27a50004 addiu $5, $29, 4
                                                                                ; 184: addiu $al $sp 4 # argv
Cause
BadVAddr = 0
                                   [00400008] 24a60004 addiu $6, $5, 4
                                                                                 ; 185: addiu $a2 $a1 4 # envp
                                   [0040000c] 00041080 sll $2, $4, 2
Status
         = 805371664
                                                                                 ; 186: s11 $v0 $a0 2
                                   [00400010] 00c23021 addu $6, $6, $2
                                                                                 ; 187: addu $a2 $a2 $v0
HI
         = 0
                                   [00400014] 0c100009 jal 0x00400024 [main]
                                                                                 ; 188: jal main
        = 25
LO
                                   [00400018] 00000000 nop
                                                                                 ; 189: nop
                                   [0040001c] 3402000a ori $2, $0, 10
                                                                                 ; 191: 11 $v0 10
                                   [00400020] 0000000c syscall
                                                                                 ; 192: syscall # syscall 10 (exit)
   [r0] = 0
   [at] = 268500992
                                   [00400024] 3c011001 lui $1, 4097
                                                                                 ; 10: 1w $t0 , n
R1
   [v0] = 10
                                   [00400028] 8c280000 lw $8, 0($1)
R2
                                   [0040002c] 34090001 ori $9, $0, 1
                                                                                ; 11: 11 $t1 ,1
R3
   [v1] = 0
                                   [00400030] 340a0000 ori $10, $0, 0
                                                                                ; 12: 11 $t2,0
   [a0] = 1
R4
                                   [00400034] 71295802 mul $11, $9, $9
                                                                                ; 15: mul $t3 , $t1, $t1
R5 [a1] = 2147481976
R6 [a2] = 2147481984
                                   [00400038] 014b5020 add $10, $10, $11
                                                                                ; 16: add $t2 , $t2, $t3
R7 [a3] = 0
                                   [0040003c] 21290001 addi $9, $9, 1
                                                                                ; 18: add $t1, $t1, 1
R8 [t0] = 5
                                   [00400040] 0109082a slt $1, $8, $9
                                                                                ; 19: ble $t1, $t0 , sumLoop
R9 [t1] = 6
                                   [00400044] 1020fffc beq $1, $0, -16 [sumLoop-0x00400044]
R10 [t2] = 55
                                   [00400048] 3c011001 lui $1, 4097
                                                                                 ; 20: sw $t2, sumOfSquares
R11 [t3] = 25
                                   [0040004c] ac2a0004 sw $10, 4($1)
                                   [00400050] 3402000a ori $2, $0, 10
R12 [t4] = 0
                                                                                 ; 22: 11 $v0, 10
                                   [004000541 0000000c syscall
                                                                                ; 23: syscall
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
                                                                                  Kernel Text Segment [80000000]..[80010000]
R16 [s0] = 0
                                   [80000180] 0001d821 addu $27, $0, $1
                                                                                 ; 90: move $k1 $at # Save $at
                                   [80000184] 3c019000 lui $1, -28672
                                                                                 ; 92: sw $v0 sl # Not re-entrant and we can't trust $sp
R17 [s1] = 0
R18 [s2] = 0
                                   [80000188] ac220200 sw $2, 512($1)
                                   [8000018c] 3c019000 lui $1, -28672
R19 [s3] = 0
                                                                                 ; 93: sw $a0 s2 # But we need to use these registers
                                   [80000190] ac240204 sw $4, 516($1)
R20 [s4] = 0
R21 [s5] = 0
                                   [80000194] 401a6800 mfc0 $26, $13
                                                                                ; 95: mfc0 $k0 $13 # Cause register
R22 [s6] = 0
                                   [80000198] 001a2082 srl $4, $26, 2
                                                                                ; 96: srl $a0 $k0 2 # Extract ExcCode Field
R23 [s7] = 0
                                   [8000019c] 3084001f andi $4, $4, 31
                                                                                ; 97: andi $a0 $a0 0x1f
                                   [800001a0] 34020004 ori $2, $0, 4
                                                                                 ; 101: 1i $v0 4 # syscall 4 (print_str)
R24 [t8] = 0
R25 [t9] = 0
                                   [800001a4] 3c049000 lui $4, -28672 [_ml_]
                                                                               ; 102: la $a0 __m1_
                                   [800001a8] 0000000c syscall
                                                                                 ; 103: syscall
R26 [k0] = 0
R27 [k1] = 0
                                   [800001ac] 34020001 ori $2, $0, 1
                                                                                 ; 105: 1i $v0 1 # syscall 1 (print int)
```

FPF	Regs	nt Regs [10]		Data	Text				
Int Re	gs [10	1	Ø X	Data					
1500	e :		•	[10000000] [10010000]	segment [1000 [1000ffff] 00000005 [1003ffff]	0000][10 00000000 00000037 00000000	040000]	00000000	7
HI LO		= 805371664 = 0 = 25		User Stack [7ffff974] [7ffff980]		[80000000] 7ffffa3a 7fffffe0	00000000 7fffffdl	7fffffbc	
R1 R2	[v0]	= 268500992 = 10		[7ffff990] [7ffff9a0] [7ffff9b0]	7fffffa9 7fffff65 7ffffe8f	7ffffffal 7ffffff42 7ffffe7e	7fffff8d 7fffff26 7ffffe6b	7ffffff7b 7ffffec4 7ffffe39	eB&
R4 R5 R6	[a2]	= 1 = 2147481976 = 2147481984		[7ffff9c0] [7ffff9d0] [7ffff9e0] [7ffff9f0]	7ffffc95	7ffffe16 7ffffd4a 7ffffcf7 7ffffc82	7ffffe09 7ffffd3f 7ffffcce 7ffffc71	7ffffd74 7ffffd25 7ffffca5 7ffffc66	(
R8 R9	[a3] = [t0] = [t1] =	= 5 = 6		[7ffffa00] [7ffffa10] [7ffffa20] [7ffffa30]	7ffffc54 7ffffb96 7ffffac2 00000000	7ffffc41 7ffffb6a 7ffffa9f 00000000	7ffffc2b 7ffffb53 7ffffa87 682f0000	7ffffbff 7ffffb03 7ffffa66 2f656d6f	T A +
R11 R12 R13	[t3] = [t4] = [t5] =	= 25 = 0 = 0		[7ffffa40] [7ffffa50] [7ffffa60]	656a6172 326f632f 73612e32	6172646e 2f303136 4a47006d	7365442f 3842414c 45445f53	706f746b 2f57454e 5f475542	rajendra/Desktop /co2610/LAB8NEW/ 2.asm.GJS_DEBUG_
R15 R16	[t6] : [t7] : [s0] : [s1] :	= 0 = 0		[7ffffa70] [7ffffa80] [7ffffa90] [7ffffaa0]	49504f54 4c20534a 54554f5f 4c5f4f49	4a3d5343 4700474f 3d545550 434e5541	52452053 445f534a 65647473 5f444548	3b524f52 47554245 47007272 4b534544	TOPICS = JS ERROR; JS LOG.GJS_DEBUG _OUTPUT = stderr.G IO_LAUNCHED_DESK
R19 R20	[s2] = [s3] = [s4] = [s5] =	= 0 = 0		[7ffffab0] [7ffffac0] [7ffffad0] [7ffffae0]	5f504f54 49470031 544b5345 6168732f	454c4946 414c5f4f 465f504f 612f6572	4449505f 48434e55 3d454c49 696c7070	3038323d 445f4445 7273752f 69746163	T O P _ F I L E _ P I D = 2 8 0 1 . G I O _ L A U N C H E D _ D E S K T O P _ F I L E = / u s r / s h a r e / a p p l i c a t i
R22 R23 R24	[s6] : [s7] : [t8] :	= 0 = 0 = 0		[7ffffaf0] [7ffffb00] [7ffffb10]	2f736e6f 5300706f 6c3d5245	70737471 49535345 6c61636f	642e6d69 4d5f4e4f 6e69532f	746b7365 47414e41 403a6867	ons/qtspim.deskt op.SESSION_MANAG ER=1oca1/Singh: @
R26 R27	[k0] = [k1] =	= 0	-	[7ffffb20] [7ffffb30] [7ffffb40]	706d742f 2c313531 2f706d74	43492e2f 78696e75 4543492e	6e752d45 6e69532f 696e752d	312f7869 2f3a6867 31312f78	/tmp/.ICE-unix/1 151,unix/Singh:/ tmp/.ICE-unix/11

#### **CODE WITH** NOP:

.data

.text

Main:

li \$t2,0

sumLoop:

lw nop li

n:

.globl main

.word 5

\$t0, n

\$t1,1

Nop

mul \$t3, \$t1, \$t1 add \$t2, \$t2, \$t3

ble \$t1, \$t0, sumLoop

sw \$t2, sumOfSquares

add \$t1, \$t1, 1

sumOfSquares: .word 0

#DATA

#MAIN

**#SUM LOOP** 

#NOP

li \$v0, 10 #EXITING

syscall .end main

FPR	egs	nt Regs [10]		Data	Text		
Int Re	gs [10]		ØX	Text			
PC		4194396	_				User Text Segment [00400000][00440000]
EPC	9	= 0		[00400000]	8fa40000	lw \$4, 0(\$29)	; 183: 1w \$a0 0(\$sp) # argc
Cause	=	= 0		[00400004]	27a50004	addiu \$5, \$29, 4	; 184: addiu \$al \$sp 4 # argv
BadVA	ddr =	= 0		[00400008]	24a60004	addiu \$6, \$5, 4	; 185: addiu \$a2 \$a1 4 # envp
Statu	ıs =	= 805371664		[0040000c]	00041080	sll \$2, \$4, 2	; 186: s11 \$v0 \$a0 2
				[00400010]	00c23021	addu \$6, \$6, \$2	; 187: addu \$a2 \$a2 \$v0
HI	9	= 0		[00400014]	0c100009	jal 0x00400024 [main]	; 188: jal main
LO	=	= 25		[00400018]	00000000	nop	; 189: nop
100				[0040001c]	3402000a	ori \$2, \$0, 10	; 191: 11 \$v0 10
R0 [	r0] =	= 0		[00400020]	0000000c	syscall	; 192: syscall # syscall 10 (exit)
R1 [	at] =	= 268500992		[00400024]	3c011001	lui \$1, 4097	; 10: 1w \$t0 , n
R2 [	v0] =	= 10		[00400028]	8c280000	lw \$8, 0(\$1)	
R3 [	v1] =	= 0		[0040002c]	00000000	nop	; 11: nop
R4 [	a0] =	= 1		[00400030]	34090001	ori \$9, \$0, 1	; 12: 1i \$t1 ,1
R5 [	a1] =	= 2147481976		[00400034]	340a0000	ori \$10, \$0, 0	; 13: 11 \$t2,0
R6 [	a2] =	= 2147481984		[00400038]	71295802	mul \$11, \$9, \$9	; 16: mul \$t3 , \$t1, \$t1
R7 [	a3] =	= 0		[0040003c]	014b5020	add \$10, \$10, \$11	; 17: add \$t2 , \$t2, \$t3
R8 [	t0] =	= 5		[00400040]	21290001	addi \$9, \$9, 1	; 19: add \$t1, \$t1, 1
R9 [	t1] =	= 6		[00400044]	0109082a	slt \$1, \$8, \$9	; 20: ble \$t1, \$t0 , sumLoop
R10 [	t2] =	= 55		[00400048]	1020fffc	beq \$1, \$0, -16 [sumLoo	p-0x00400048]
R11 [	t3] =	= 25		[0040004c]	00000000	nop	; 21: nop
R12 [	t4] =	= 0		[00400050]	3c011001	lui \$1, 4097	; 22: sw \$t2, sumOfSquares
R13 [	t5] =	= 0		[00400054]	ac2a0004	sw \$10, 4(\$1)	
R14 [	t6] =	= 0		[00400058]	3402000a	ori \$2, \$0, 10	; 24: 11 \$v0, 10
R15 [	t7] =	= 0		[0040005c]		syscall	; 25: syscall
R16 [	s0] =	= 0		100	•	90	
R17 [	s1] =	= 0					Kernel Text Segment [80000000][80010000]
R18 [	s2] =	= 0		[80000180]	0001d821	addu \$27, \$0, \$1	; 90: move \$k1 \$at # Save \$at
R19 [	s3] =	= 0		[80000184]	3c019000	lui \$1, -28672	; 92: sw \$v0 s1 # Not re-entrant and we can't trust \$sp
R20 [	s4] =	= 0		[80000188]	ac220200	sw \$2, 512(\$1)	
R21 [	s5] =	= 0		[8000018c]	3c019000	lui \$1, -28672	; 93: sw \$a0 s2 # But we need to use these registers
R22 [	s6] =	= 0		[80000190]	ac240204	sw \$4, 516(\$1)	
R23 [	s7] =	= 0		[80000194]	401a6800	mfc0 \$26, \$13	; 95: mfc0 \$k0 \$13 # Cause register
R24 [	t8] =	= 0		[80000198]	001a2082	srl \$4, \$26, 2	; 96: srl \$a0 \$k0 2 # Extract ExcCode Field
R25 [	t9] =	= 0		[8000019c]	3084001f	andi \$4, \$4, 31	; 97: andi \$a0 \$a0 0x1f
R26 [	k0] =	= 0		[800001a0]	34020004	ori \$2, \$0, 4	; 101: 1i \$v0 4 # syscall 4 (print_str)
R27 [	k1] =	= 0	-	[800001a4]	3c049000	lui \$4, -28672 [m1_]	; 102: 1a \$a0m1_
соруг	rgnc	1990-Zolf by James	Larus.				

FPI	Regs	nt Regs [10]		Data	Text		
	gs [10		P X		200000000000000000000000000000000000000		
				TOAL			W M C [00400000] [00440000]
PC EPC		= 4194396 = 0		100400000	05-10000	lw \$4, 0(\$29)	User Text Segment [00400000][00440000] ; 183: 1w \$a0 0(\$sp) # argc
Caus		= 0				addiu \$5, \$29, 4	; 184: addiu \$a1 \$sp 4 # argv
100	Addr					addiu \$6, \$5, 4	; 185: addiu \$a2 \$a1 4 # envp
Stati	us	= 805371664				s11 \$2, \$4, 2	; 186: s11 \$v0 \$a0 2
						addu \$6, \$6, \$2	; 187: addu \$a2 \$a2 \$v0
HI		= 0				jal 0x00400024 [main]	; 188: jal main
LO		= 25		[00400018]			; 189: nop
esses m						ori \$2, \$0, 10	; 191: 11 \$v0 10
	[r0]			[00400020]			; 192: syscall # syscall 10 (exit)
		= 268500992				lui \$1, 4097	; 10: 1w \$t0 , n
	[v0]					lw \$8, 0(\$1)	590
R3	[v1]	= 0		[0040002c]			; 11: nop
	[a0] :					ori \$9, \$0, 1	; 12: 1i \$t1 ,1
R5	[a1] :	= 2147481976				ori \$10, \$0, 0	; 13: 11 \$t2,0
R6	[a2] :	= 2147481984		[00400038]	71295802	mul \$11, \$9, \$9	; 16: mul \$t3 , \$t1, \$t1
R7	[a3] :	= 0	No.	[0040003c]	014b5020	add \$10, \$10, \$11	; 17: add \$t2 , \$t2, \$t3
R8	[t0] :	= 5	1	[00400040]	21290001	addi \$9, \$9, 1	; 19: add \$t1, \$t1, 1
R9	[t1] :	= 6				slt \$1, \$8, \$9	; 20: ble \$t1, \$t0 , sumLoop
R10	[t2]	= 55		[00400048]	1020fffc	beq \$1, \$0, -16 [sumLoo	p-0x00400048]
R11	[t3] :	= 25		[0040004c]	00000000	nop	; 21: nop
R12	[t4] :	= 0		[00400050]	3c011001	lui \$1, 4097	; 22: sw \$t2, sumOfSquares
R13	[t5]	= 0		[00400054]	ac2a0004	sw \$10, 4(\$1)	
R14	[t6] :	= 0		[00400058]	3402000a	ori \$2, \$0, 10	; 24: 1i \$v0, 10
R15	[t7] :	= 0		[0040005c]	0000000c	syscall	; 25: syscall
R16	[s0] :	= 0					
R17	[s1] :	= 0					Kernel Text Segment [80000000][80010000]
	[s2] :			[800001801	0001d821	addu \$27, \$0, \$1	; 90: move \$k1 \$at # Save \$at
	[s3] :					lui \$1, -28672	; 92: sw \$v0 s1 # Not re-entrant and we can't trust \$sp
	[s4] :					sw \$2, 512(\$1)	
	[s5] :					lui \$1, -28672	; 93: sw \$a0 s2 # But we need to use these registers
	[s6]					sw \$4, 516(\$1)	
	[s7]					mfc0 \$26, \$13	; 95: mfc0 \$k0 \$13 # Cause register
	[t8]					srl \$4, \$26, 2	; 96: srl \$a0 \$k0 2 # Extract ExcCode Field
	[t9]					andi \$4, \$4, 31	; 97: andi \$a0 \$a0 0x1f
	[k0]					ori \$2, \$0, 4	; 101: 1i \$v0 4 # syscall 4 (print_str)
	[k1]					lui \$4, -28672 [ml_]	
			~	[00000144]	35043000		, 102, 10 , 100m1_
copy	Tgnc	real value of the same	S Larus.				

(a) A code snippet is given below. Run the code by including the necessary directives, syscalls and by providing values for num1 and num2. The values provided for num1 and

num2 should not equal.

lw \$t1, num1
lw \$t2, num2
lw \$t1, num2

Verify the output in \$t3 and \$t4.

(b) Run the same program by enabling 'delayed load' and verify the output.

To enable delayed load go to, Simulator → Settings → MIPS tab→ Enable Delayed Load.

**QUESTION (3) Illustration of delayed load:** 

add \$t3, \$t1, \$0 add \$t4, \$t1, \$0

Compare your observations in (a) and (b). How many delay slots exist for the 'delayed load' instruction?

Modify the code to get the same logical output as in (a).

CODE	.data
WITHOUT	num1 : .wor num2 : .wor
NOP:	.text .globl main .ent main

word 2 word 4

**#TEXT** 

**#EXITING** 

#DATA

main: lw \$t1, num1 lw \$t1, num2 lw \$t2, num2

add \$t4, \$t1, \$0

syscall

add \$t3, \$t1, \$0

li \$v0, 10

.end main

FP Regs	nt Regs [10]		Data	Text		
nt Regs [10]		Ø X	Text			
PC =	4194396					User Text Segment [00400000][00440000]
EPC =	: 0		[00400000]	8fa40000	lw \$4, 0(\$29)	; 183: lw \$a0 0(\$sp) # argc
Cause =	= 0		[00400004]	27a50004	addiu \$5, \$29, 4	; 184: addiu \$a1 \$sp 4 # argv
BadVAddr =	: 0		[00400008]	24a60004	addiu \$6, \$5, 4	; 185: addiu \$a2 \$a1 4 # envp
Status =	805371664		[0040000c]	00041080	sl1 \$2, \$4, 2	; 186: sll \$v0 \$a0 2
			[00400010]	00c23021	addu \$6, \$6, \$2	; 187: addu \$a2 \$a2 \$v0
HI =	= 0		[00400014]	0c100009	jal 0x00400024 [main]	; 188: jal main
LO =	25		[00400018]	00000000	nop	; 189: nop
			[0040001c]	3402000a	ori \$2, \$0, 10	; 191: 1i \$v0 10
R0 [r0] =	= 0		[00400020]	0000000c	syscall	; 192: syscall # syscall 10 (exit)
R1 [at] =	268500992		[00400024]	3c011001	lui \$1, 4097	; 10: lw \$t0 , n
R2 [v0] =	: 10		[00400028]	8c280000	lw \$8, 0(\$1)	
R3 [v1] =	0		[0040002c]	00000000	nop	; 11: nop
R4 [a0] =	: 1		[00400030]	34090001	ori \$9, \$0, 1	; 12: 11 \$t1 ,1
R5 [a1] =	2147481976		[00400034]	340a0000	ori \$10, \$0, 0	; 13: 11 \$t2,0
R6 [a2] =	2147481984		[00400038]	71295802	mul \$11, \$9, \$9	; 16: mul \$t3 , \$t1, \$t1
R7 [a3] =	= 0		[0040003c]	014b5020	add \$10, \$10, \$11	; 17: add \$t2 , \$t2, \$t3
R8 [t0] =	= 5		[00400040]	21290001	addi \$9, \$9, 1	; 19: add \$t1, \$t1, 1
R9 [t1] =	6		[00400044]	0109082a	slt \$1, \$8, \$9	; 20: ble \$t1, \$t0 , sumLoop
R10 [t2] =	55		[00400048]	1020fffc	beq \$1, \$0, -16 [sumLoc	p-0x00400048]
R11 [t3] =	25		[0040004c]	00000000	nop	; 21: nop
R12 [t4] =	: 0		[00400050]	3c011001	lui \$1, 4097	; 22: sw \$t2, sumOfSquares
R13 [t5] =	: 0		[00400054]	ac2a0004	sw \$10, 4(\$1)	& Date   11   11   12   13   14   15   14   15   15   15   15   15
R14 [t6] =	: 0		[00400058]	3402000a	ori \$2, \$0, 10	; 24: 1i \$v0, 10
R15 [t7] =	: 0		[0040005c]	0000000c	syscall	; 25: syscall
R16 [s0] =	= 0		100		90	
R17 [s1] =	= 0					Kernel Text Segment [80000000][80010000]
R18 [s2] =	: 0		[80000180]	0001d821	addu \$27, \$0, \$1	; 90: move \$k1 \$at # Save \$at
R19 [s3] =	: 0				lui \$1, -28672	; 92: sw \$v0 sl # Not re-entrant and we can't trust \$sp
R20 [s4] =					sw \$2, 512(\$1)	
R21 [s5] =	= 0		[8000018c]	3c019000	lui \$1, -28672	; 93: sw \$a0 s2 # But we need to use these registers
R22 [s6] =	= 0				sw \$4, 516(\$1)	
R23 [s7] =			[80000194]	401a6800	mfc0 \$26, \$13	; 95: mfc0 \$k0 \$13 # Cause register
R24 [t8] =			[80000198]	001a2082	srl \$4, \$26, 2	; 96: srl \$a0 \$k0 2 # Extract ExcCode Field
R25 [t9] =	= 0		[8000019c]	3084001f	andi \$4, \$4, 31	; 97: andi \$a0 \$a0 0x1f
R26 [k0] =	= 0		[800001a0]	34020004	ori \$2, \$0, 4	; 101: 1i \$v0 4 # syscall 4 (print_str)
R27 [k1] =	· 0		[800001a41	3c049000	lui \$4, -28672 [_ml_]	

## CODE WITH NOP:

.data #DATA num1:.word 2 num2:.word 4 **#TEXT** .text .globl main .ent main Main: #MAIN Iw \$t1, num1 Nop **#NOOP AFTER LOAD** lw \$t2, num2 Nop **#NOOP AFTER LOAD** lw \$t1, num2 Nop **#NOOP AFTER LOAD** add \$t3, \$t1, \$0 add \$t4, \$t1, \$0 li \$v0, 10 syscall .end main #EXIT

FP R	Regs	nt Regs [10]		Data	Text		
Int Re	gs [10]		OX.	Text			
PC	=	4194388	_				User Text Segment [00400000][00440000]
EPC	=	0		[004000001	8fa40000	lw \$4, 0(\$29)	; 183: 1w \$a0 0(\$sp) # argc
Cause	=	0				addiu \$5, \$29, 4	; 184: addiu \$a1 \$sp 4 # argv
	Addr =	0				addiu \$6, \$5, 4	; 185: addiu \$a2 \$a1 4 # envp
Statu	ıs =	805371664				sll \$2, \$4, 2	; 186: sll \$v0 \$a0 2
						addu \$6, \$6, \$2	; 187: addu \$a2 \$a2 \$v0
HI	=	0				jal 0x00400024 [main]	; 188: jal main
LO	=	0		[00400018]		_	; 189: nop
				[0040001c1	3402000a	ori \$2, \$0, 10	; 191: 11 \$v0 10
RO I	[r0] =	0		[00400020]			; 192: syscall # syscall 10 (exit)
		268500992				lui \$1, 4097	; 10: 1w \$t1, num1
	[v0] =					lw \$9, 0(\$1)	A MILES TO A TO
R3 [	[v1] =	0		[0040002c]			; 11: пор
R4 [	[a0] =	1		[00400030]	3c011001	lui \$1, 4097	; 12: 1w \$t2, num2
R5 [	[a1] =	2147481976		[00400034]	8c2a0004	lw \$10, 4(\$1)	
R6 [	[a2] =	2147481984		[00400038]			; 13: nop
R7 [	[a3] =	0	0.00	[0040003c]	3c011001	lui \$1, 4097	; 14: 1w \$t1, num2
R8 [	[t0] =	0		[00400040]	8c290004	lw \$9, 4(\$1)	
R9 [	[t1] =	4		[00400044]	00000000	nop	; 15: nop
R10 [	[t2] =	4		[00400048]	01205820	add \$11, \$9, \$0	; 16: add \$t3, \$t1, \$0
R11 [	[t3] =	4		[0040004c]	01206020	add \$12, \$9, \$0	; 17: add \$t4, \$t1, \$0
R12 [	[t4] =	4		[00400050]	3402000a	ori \$2, \$0, 10	; 18: 1i \$v0 , 10
R13 [	[t5] =	0		[00400054]	0000000c	syscall	; 19: syscall
R14 [	[t6] =	0		M. Co.			
R15 [	[t7] =	0					Kernel Text Segment [80000000][80010000]
R16 [	[s0] =	0		[80000180]	0001d821	addu \$27, \$0, \$1	; 90: move \$k1 \$at # Save \$at
R17 [	[s1] =	0		[80000184]	3c019000	lui \$1, -28672	; 92: sw \$v0 s1 # Not re-entrant and we can't trust \$sp
R18 [	[s2] =	0		[80000188]	ac220200	sw \$2, 512(\$1)	
R19 [	[ <b>s3</b> ] =	0		[8000018c]	3c019000	lui \$1, -28672	; 93: sw \$a0 s2 # But we need to use these registers
R20 [	[s4] =	0		[80000190]	ac240204	sw \$4, 516(\$1)	
R21 [	[s5] =	0		[80000194]	401a6800	mfc0 \$26, \$13	; 95: mfc0 \$k0 \$13 # Cause register
R22 [	[s6] =	0		[80000198]	001a2082	srl \$4, \$26, 2	; 96: srl \$a0 \$k0 2 # Extract ExcCode Field
R23 [	[ <b>s</b> 7] =	0	1000	[8000019c]	3084001f	andi \$4, \$4, 31	; 97: andi \$a0 \$a0 0x1f
R24 [	[t8] =	0		[800001a0]	34020004	ori \$2, \$0, 4	; 101: 1i \$v0 4 # syscall 4 (print_str)
R25 [	[t9] =	0					; 102: 1a \$a0m1_
R26 [	[k0] =	0		[800001a8]			; 103: syscall
R27 [	[k1] =	0	-	[800001ac]	34020001	ori \$2, \$0, 1	; 105: li \$v0 1 # syscall 1 (print_int)

#### **QUESTION (4) Introduction to 2-D arrays:**

Write a MIPS assembly program to calculate and display the determinant of a 2x2 matrix using the concept of 2-D arrays. Use procedure to calculate determinant.

Assume that the array is accessed using the row major concept where any the address of the element in ith row and jth column is given by:

```
Addr[i][j] = Base address of the array + [( (row index * column size) + column index) * size of data]

Where,
row index = i
column index = j
column size = Total number of columns
size of data = The size of data that you are using (Eg: For integers the size =4)

All the programs should include a header section with the program title(what the program does), comments wherever required and display statements.
```

#### CODE **WITHOUT** NOP:

.word 6, 3 .word4,0 size: .word 2 ans: .word 0

#DATASIZE # addr = baseAddr+(rowIndex\*colSize+colIndex) \* dataSize

.text .globl main .ent main Main: la \$a0, mult arr lw \$a1, size jal det move \$a0, \$v0 li \$v0, 1 syscall sw \$a0, ans li \$v0, 10

syscall

.end main

DATASIZE = 4

.data

Mult\_arr:

**#DET CALLED** #MAIN END

#DATA

#MATIX

**#TEXT** 

#MAIN

```
# argument to det procedure:
                      # base address : $a0
                      # colSize : $a1
                      # return the result
                      # determinant : $v0
.globl det
.ent det
                      #DET FUNTION
det:
     \#(0,0)
     lw $v0 , ($a0)
     # (1,1)
     addi $t0, $a1, 1
     mul $t0, $t0, DATASIZE
     add $t0, $t0, $a0
     Iw $t0, ($t0)
     mul $v0, $t0, $v0
     \#(0,1)
     li $t0, DATASIZE
     add $t0, $t0, $a0
     lw $t0, ($t0)
     # (1,0)
     move $t1, $a1
```

```
mul $t1, $t1, DATASIZE
    add $t1, $t1, $a0
    ###########
    mul $t0, $t0, $t1
    sub $v0 , $v0 , $t0
```

**#END DET** 

lw \$t1, (\$t1)

jr \$ra

.end det

	F101				
FP Regs	nt Regs [10]	Data	Text		
Int Regs [10]		Text			
PC =	4194380				User Text Segment [00400000][00440000]
EPC =	0	[00400000]	8fa40000	lw \$4, 0(\$29)	; 183: 1w \$a0 0(\$sp) # argc
Cause =	0	[00400004]	27a50004	addiu \$5, \$29, 4	; 184: addiu \$a1 \$sp 4 # argv
BadVAddr =	0				; 185: addiu \$a2 \$a1 4 # envp
Status =	805371664	[0040000c]	00041080	sll \$2, \$4, 2	; 186: sll \$v0 \$a0 2
S0-12863	200				; 187: addu \$a2 \$a2 \$v0
A STATE OF THE STA	0	[00400014]	0c100009	jal 0x00400024 [main]	; 188: jal main
LO =	12	[00400018]			; 189: nop
Service Service Control	100	[0040001c]	3402000a	ori \$2, \$0, 10	; 191: 11 \$v0 10
R0 [r0] =	0	[00400020]			; 192: syscall # syscall 10 (exit)
R1 [at] =	268500992	[00400024]	3c041001	lui \$4, 4097 [mult_arr]	; 17: la \$a0 , mult_arr
R2 [v0] =	10	[00400028]	3c011001	lui \$1, 4097	; 18: lw \$al , size
R3 [v1] =	0	[0040002c]	8c250010	lw \$5, 16(\$1)	
R4 [a0] =	-12			jal 0x00400050 [det]	; 19: jal det
R5 [a1] =	2	[00400034]	00022021		; 20: move \$a0 , \$v0
R6 [a2] =	2147481984	[00400038]	34020001	ori \$2, \$0, 1	; 21: 1i \$v0 , 1
R7 [a3] =	0	[0040003c]	0000000c	syscall	; 22: syscall
R8 [t0] =	12	[00400040]	3c011001	lui \$1, 4097	; 23: sw \$a0 , ans
R9 [t1] =	4			sw \$4, 20(\$1)	
R10 [t2] =	0	[00400048]	3402000a	ori \$2, \$0, 10	; 24: 1i \$v0, 10
R11 [t3] =	0	[0040004c]			; 25: syscall
R12 [t4] =	0			lw \$2, 0(\$4)	; 41: 1w \$v0 , (\$a0)
R13 [t5] =	0	[00400054]	20a80001	addi \$8, \$5, 1	; 43: addi \$t0 , \$a1 , 1
R14 [t6] =	0			ori \$1, \$0, 4	; 44: mul \$t0 , \$t0 , DATASIZE
R15 [t7] =	0	[0040005c]	71014002	mul \$8, \$8, \$1	
R16 [s0] =	0	[00400060]	01044020	add \$8, \$8, \$4	; 45: add \$t0 , \$t0 , \$a0
R17 [s1] =	0	[00400064]	000080b8	lw \$8, 0(\$8)	; 46: lw \$t0 , (\$t0)
R18 [s2] =	0				; 47: mul \$v0 , \$t0, \$v0
R19 [s3] =	0	[0040006c]	34080004		; 49: li \$t0 , DATASIZE
R20 [s4] =	0	[00400070]	01044020	add \$8, \$8, \$4	; 50: add \$t0 , \$t0 , \$a0
R21 [s5] =	0	[00400074]	000080b8	lw \$8, 0(\$8)	; 51: 1w \$t0 , (\$t0)
R22 [s6] =	0	[00400078]	00054821	addu \$9, \$0, \$5	; 53: move \$t1 , \$a1
R23 [s7] =	0			ori \$1, \$0, 4	; 54: mul \$t1 , \$t1 , DATASIZE
R24 [t8] =	0			mul \$9, \$9, \$1	
R25 [t9] =	0				; 55: add \$t1 , \$t1 , \$a0
R26 [k0] =	0				; 56: 1w \$t1 , (\$t1)
R27 [k1] =	0	[0040008c]	71094002	mul \$8, \$8, \$9	; 58: mul \$t0 , \$t0 , \$t1
	No. 27			A	10 10 10 10 10 10 10 10 10 10 10 10 10 1

	Console –		×
-12			

## **CODE WITH** NOP:

```
mult_arr:
      .word 6, 3
      .word 4, 0
```

.data

```
size: .word 2
ans: .word 0
DATASIZE = 4
# addr =
baseAddr+(rowIndex*colSize+colIn
dex) * dataSize
.text
.globl main
.ent main
main:
     la $a0 , mult_arr
     nop
     lw $a1, size
     nop
     jal det
     nop
```

```
move $a0, $v0
     li $v0, 1
     syscall
     sw $a0, ans
     li $v0, 10
     syscall
.end main
# argument to det procedure:
# base address : $a0
# colSize : $a1
# return the result
# determinant : $v0
.globl det
.ent det
det:
     # (0,0)
     nop
     lw $v0 , ($a0)
     Nop
```

```
# (1,1)
     addi $t0, $a1, 1
     mul $t0, $t0, DATASIZE
     add $t0, $t0, $a0
     lw $t0, ($t0)
                                         #NOP AFTER LOAD
     Nop
     mul $v0, $t0, $v0
     \#(0,1)
     li $t0, DATASIZE
     add $t0, $t0, $a0
     Iw $t0, ($t0)
     Nop
                                         #NOP AFTER LOAD
     # (1,0)
     move $t1, $a1
     mul $t1, $t1, DATASIZE
     add $t1, $t1, $a0
     lw $t1, ($t1)
     Nop
                                         #NOP AFTER LOAD
     ############
     mul $t0, $t0, $t1
     sub $v0, $v0, $t0
     jr $ra
     Nop
                                         #NOP AFTER JUMP
.end det
```

	<b>2</b> (3) 2 ++++				0		
FP Regs	nt Regs [10]		Data	Text			
Int Regs [10	0]	ØX	Text				
PC	= 4194392						User Text Segment [00400000][00440000]
EPC	= 0		[00400000]	8fa40000	lw \$4, 0(\$29)	;	183: 1w \$a0 0(\$sp) # argc
Cause	= 0					;	184: addiu \$a1 \$sp 4 # argv
BadVAddr	= 0		[00400008]	24a60004	addiu \$6, \$5, 4	;	185: addiu \$a2 \$a1 4 # envp
Status	= 805371664		[0040000c]	00041080	sll \$2, \$4, 2	;	186: s11 \$v0 \$a0 2
						;	187: addu \$a2 \$a2 \$v0
HI	= 0		[00400014]	0c100009	jal 0x00400024 [main]	;	188: jal main
LO	= 12		[00400018]	00000000	nop	;	189: nop
			[0040001c]	3402000a	ori \$2, \$0, 10	;	191: 11 \$v0 10
R0 [r0]	= 0		[00400020]	0000000c	syscall	;	192: syscall # syscall 10 (exit)
R1 [at]	= 268500992		[00400024]	3c041001	lui \$4, 4097 [mult_arr]	;	17: la \$a0 , mult_arr
R2 [v0]	= 10		[00400028]	00000000	nop	;	18: nop
R3 [v1]	= 0		[0040002c]	3c011001	lui \$1, 4097	;	19: 1w \$a1 , size
R4 [a0]	= -12		[00400030]	8c250010	lw \$5, 16(\$1)		
R5 [a1]	= 2		[00400034]	00000000	nop	;	20: nop
R6 [a2]	= 2147481976		[00400038]	0c100017	jal 0x0040005c [det]	;	21: jal det
R7 [a3]	= 0	1	[0040003c]	00000000			22: nop
R8 [t0]	= 12		[00400040]	00022021	addu \$4, \$0, \$2	;	23: move \$a0 , \$v0
R9 [t1]	= 4		[00400044]	34020001	ori \$2, \$0, 1	;	24: 1i \$v0 , 1
R10 [t2]	= 0		[00400048]			;	25: syscall
R11 [t3]	= 0		[0040004c]	3c011001	lui \$1, 4097	;	26: sw \$a0 , ans
R12 [t4]	= 0				sw \$4, 20(\$1)		
R13 [t5]	= 0		[00400054]	3402000a	ori \$2, \$0, 10	;	27: 1i \$v0, 10
R14 [t6]	= 0						28: syscall
R15 [t7]	= 0		[0040005c]	00000000	nop	;	44: nop
R16 [s0]	= 0		[00400060]	8c820000	lw \$2, 0(\$4)	;	45: 1w \$v0 , (\$a0)
R17 [s1]	= 0		[00400064]	00000000	nop	;	46: nop
R18 [s2]	= 0		[00400068]	20a80001	addi \$8, \$5, 1	;	48: addi \$t0 , \$a1 , 1
R19 [s3]	= 0		[0040006c]	34010004	ori \$1, \$0, 4	;	49: mul \$t0 , \$t0 , DATASIZE
R20 [s4]	= 0		[00400070]	71014002	mul \$8, \$8, \$1		
R21 [s5]	= 0		[00400074]	01044020	add \$8, \$8, \$4	;	50: add \$t0 , \$t0 , \$a0
R22 [s6]	= 0		[00400078]	8d080000	lw \$8, 0(\$8)	;	51: 1w \$t0 , (\$t0)
R23 [s7]	= 0		[0040007c]	00000000			52: nop
R24 [t8]	= 0		[00400080]	71021002	mul \$2, \$8, \$2	;	53: mul \$v0 , \$t0, \$v0
R25 [t9]	= 0		[00400084]	34080004	ori \$8, \$0, 4	;	55: 1i \$t0 , DATASIZE
R26 [k0]	= 0					;	56: add \$t0 , \$t0 , \$a0
R27 [k1]	= 0	-	[0040008c]	000080b8	lw \$8, 0(\$8)	;	57: 1w \$t0 , (\$t0)

	Console -	=	×
-12			