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                   #FUNCTION POWER CALLED
        move  $t5, $v0

        li $v0, 10
        syscall

.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 Regs | nt Regs [10] | Data | Text

Int Regs [10]

```
PC       = 4194368
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 805371664

HI       = 0
LO       = 216

R0  [r0] = 0
R1  [at] = 0
R2  [v0] = 10
R3  [v1] = 0
R4  [a0] = 6
R5  [a1] = 3
R6  [a2] = 2147481984
R7  [a3] = 0
R8  [t0] = 3
R9  [t1] = 0
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 216
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
```

Text

```
                                        User Text Segment [00400000]..[00440000]
[00400000] 8fa40000  lw $4, 0($29)           ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004  addiu $5, $29, 4        ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004  addiu $6, $5, 4         ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080  sll $2, $4, 2           ; 186: sll $v0 $a0 2
[00400010] 00c23021  addu $6, $6, $2         ; 187: addu $a2 $a2 $v0
[00400014] 0c100009  jal 0x00400024 [main]   ; 188: jal main
[00400018] 00000000  nop                     ; 189: nop
[0040001c] 3402000a  ori $2, $0, 10          ; 191: li $v0 10
[00400020] 0000000c  syscall                 ; 192: syscall # syscall 10 (exit)
[00400024] 3c011001  lui $1, 4097            ; 12: lw $a0 , x
[00400028] 8c240000  lw $4, 0($1)
[0040002c] 3c011001  lui $1, 4097            ; 13: lw $a1 , y
[00400030] 8c250004  lw $5, 4($1)
[00400034] 0c100011  jal 0x00400044 [power]  ; 14: jal power
[00400038] 00026821  addu $13, $0, $2        ; 15: move $t5, $v0
[0040003c] 3402000a  ori $2, $0, 10          ; 17: li $v0, 10
[00400040] 0000000c  syscall                 ; 18: syscall
[00400044] 34020001  ori $2, $0, 1           ; 26: li $v0,1
[00400048] 34080000  ori $8, $0, 0           ; 27: li $t0,0
[0040004c] 70441002  mul $2, $2, $4          ; 30: mul $v0, $v0, $a0
[00400050] 21080001  addi $8, $8, 1          ; 31: add $t0, $t0, 1
[00400054] 0105082a  slt $1, $8, $5          ; 32: blt $t0, $a1, powLoop
[00400058] 1420fffd  bne $1, $0, -12 [powLoop-0x00400058]
[0040005c] 03e00008  jr $31                  ; 33: jr $ra

                                        Kernel Text Segment [80000000]..[80010000]
[80000180] 0001d821  addu $27, $0, $1        ; 90: move $k1 $at # Save $at
[80000184] 3c019000  lui $1, -28672          ; 92: sw $v0 s1 # Not re-entrant and we can't trust $sp
[80000188] ac220200  sw $2, 512($1)
[8000018c] 3c019000  lui $1, -28672          ; 93: sw $a0 s2 # But we need to use these registers
[80000190] ac240204  sw $4, 516($1)
[80000194] 401a6800  mfc0 $26, $13           ; 95: mfc0 $k0 $13 # Cause register
[80000198] 001a2082  srl $4, $26, 2          ; 96: srl $a0 $k0 2 # Extract ExcCode Field
[8000019c] 3084001f  andi $4, $4, 31         ; 97: andi $a0 $a0 0x1f
[800001a0] 34020004  ori $2, $0, 4           ; 101: li $v0 4 # syscall 4 (print_str)
[800001a4] 3c049000  lui $4, -28672 [__m1_]  ; 102: la $a0 __m1_
```

FP Regs | nt Regs [10] | Data | Text

Int Regs [10]

```
PC       = 4194368
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 805371664

HI       = 0
LO       = 216

R0  [r0] = 0
R1  [at] = 0
R2  [v0] = 10
R3  [v1] = 0
R4  [a0] = 6
R5  [a1] = 3
R6  [a2] = 2147481984
R7  [a3] = 0
R8  [t0] = 3
R9  [t1] = 0
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 216
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
```

Data

```
User data segment [10000000]..[10040000]
[10000000]..[1000ffff]  00000000
[10010000]    00000006  00000003  00000000  00000000    . . . . . .
[10010010]..[1003ffff]  00000000


User Stack [7ffff974]..[80000000]
[7ffff974]    00000001  7ffffa3a  00000000              . . . . : . . . . . .
[7ffff980]    7fffffe2  7fffffe0  7fffffd1  7fffffbc    . . . . . .
[7ffff990]    7fffffa9  7fffffa1  7fffff8d  7fffff7b    . . . . . . . . . . { . . .
[7ffff9a0]    7fffff65  7fffff42  7fffff26  7ffffec4    e . . . B . . . & . . . .
[7ffff9b0]    7ffffe8f  7ffffe7e  7ffffe6b  7ffffe39    . . . . ~ . . . k . . . 9 . . .
[7ffff9c0]    7ffffe28  7ffffe16  7ffffe09  7ffffd74    ( . . . . . t . . .
[7ffff9d0]    7ffffd55  7ffffd4a  7ffffd3f  7ffffd25    U . . . J . . . ? . . . % . . .
[7ffff9e0]    7ffffd11  7ffffcf7  7ffffcce  7ffffca5    . . . . . .
[7ffff9f0]    7ffffc95  7ffffc82  7ffffc71  7ffffc66    . . . . . q . . . f . . .
[7ffffa00]    7ffffc54  7ffffc41  7ffffc2b  7ffffbff    T . . . A . . . + . . . .
[7ffffa10]    7ffffb96  7ffffb6a  7ffffb53  7ffffb03    . . . . j . . . S . . . .
[7ffffa20]    7ffffac2  7ffffa9f  7ffffa87  7ffffa66    . . . . . f . . .
[7ffffa30]    00000000  00000000  682f0000  2f656d6f    . . . . . / h o m e /
[7ffffa40]    656a6172  6172646e  7365442f  706f746b    r a j e n d r a / D e s k t o p
[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 _
[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
[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
[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
[7ffffae0]    6168732f  612f6572  696c7070  69746163    / s h a r e / a p p l i c a t i
[7ffffaf0]    2f736e6f  70737471  642e6d69  746b7365    o n s / q t s p i m . d e s k t
[7ffffb00]    5300706f  49535345  4d5f4e4f  47414e41    o p . S E S S I O N _ M A N A G
[7ffffb10]    6c3d5245  6c61636f  6e69532f  403a6867    E R = l o c a l / S i n g h : @
[7ffffb20]    706d742f  43492e2f  6e752d45  312f7869    / t m p / . I C E - u n i x / 1
[7ffffb30]    2c313531  78696e75  6e69532f  2f3a6867    1 5 1 , u n i x / S i n g h : /
[7ffffb40]    2f706d74  4543492e  696e752d  31312f78    t m p / . I C E - u n i x / 1 1
```

(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 → Settings → MIPS tab→ 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 :

```
.data                                   #DATA
n:      .word 5
sumOfSquares:   .word 0

.text                                   #TEXT
.globl main

Main:                                   #MAIN FUNCTION

lw      $t0 , n
li      $t1 ,1
li $t2,0

sumLoop:                                #SUM LOOP
        mul    $t3 , $t1, $t1
        add    $t2 , $t2, $t3

        add $t1, $t1, 1
        ble $t1, $t0 , sumLoop          #CONDITION CHECK
        sw $t2, sumOfSquares

li $v0, 10
syscall
.end main                               #MAIN END
```

FP Regs | nt Regs [10] | Data | Text

Int Regs [10]

```
PC       = 4194388
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 805371664

HI       = 0
LO       = 25

R0  [r0] = 0
R1  [at] = 268500992
R2  [v0] = 10
R3  [v1] = 0
R4  [a0] = 1
R5  [a1] = 2147481976
R6  [a2] = 2147481984
R7  [a3] = 0
R8  [t0] = 5
R9  [t1] = 6
R10 [t2] = 55
R11 [t3] = 25
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
```

Text

```
                                          User Text Segment [00400000]..[00440000]
[00400000] 8fa40000  lw $4, 0($29)          ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004  addiu $5, $29, 4       ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004  addiu $6, $5, 4        ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080  sll $2, $4, 2          ; 186: sll $v0 $a0 2
[00400010] 00c23021  addu $6, $6, $2        ; 187: addu $a2 $a2 $v0
[00400014] 0c100009  jal 0x00400024 [main]  ; 188: jal main
[00400018] 00000000  nop                    ; 189: nop
[0040001c] 3402000a  ori $2, $0, 10         ; 191: li $v0 10
[00400020] 0000000c  syscall                ; 192: syscall # syscall 10 (exit)
[00400024] 3c011001  lui $1, 4097           ; 10: lw $t0 , n
[00400028] 8c280000  lw $8, 0($1)
[0040002c] 34090001  ori $9, $0, 1          ; 11: li $t1 ,1
[00400030] 340a0000  ori $10, $0, 0         ; 12: li $t2,0
[00400034] 71295802  mul $11, $9, $9        ; 15: mul $t3 , $t1, $t1
[00400038] 014b5020  add $10, $10, $11      ; 16: add $t2 , $t2, $t3
[0040003c] 21290001  addi $9, $9, 1         ; 18: add $t1, $t1, 1
[00400040] 0109082a  slt $1, $8, $9         ; 19: ble $t1, $t0 , sumLoop
[00400044] 1020fffc  beq $1, $0, -16 [sumLoop-0x00400044]
[00400048] 3c011001  lui $1, 4097           ; 20: sw $t2, sumOfSquares
[0040004c] ac2a0004  sw $10, 4($1)
[00400050] 3402000a  ori $2, $0, 10         ; 22: li $v0, 10
[00400054] 0000000c  syscall                ; 23: syscall

                                          Kernel Text Segment [80000000]..[80010000]
[80000180] 0001d821  addu $27, $0, $1       ; 90: move $k1 $at # Save $at
[80000184] 3c019000  lui $1, -28672         ; 92: sw $v0 s1 # Not re-entrant and we can't trust $sp
[80000188] ac220200  sw $2, 512($1)
[8000018c] 3c019000  lui $1, -28672         ; 93: sw $a0 s2 # But we need to use these registers
[80000190] ac240204  sw $4, 516($1)
[80000194] 401a6800  mfc0 $26, $13          ; 95: mfc0 $k0 $13 # Cause register
[80000198] 001a2082  srl $4, $26, 2         ; 96: srl $a0 $k0 2 # Extract ExcCode Field
[8000019c] 3084001f  andi $4, $4, 31        ; 97: andi $a0 $a0 0x1f
[800001a0] 34020004  ori $2, $0, 4          ; 101: li $v0 4 # syscall 4 (print_str)
[800001a4] 3c049000  lui $4, -28672 [__m1_] ; 102: la $a0 __m1_
[800001a8] 0000000c  syscall                ; 103: syscall
[800001ac] 34020001  ori $2, $0, 1          ; 105: li $v0 1 # syscall 1 (print_int)
```

FP Regs | nt Regs [10] | Data | Text

Int Regs [10]

```
PC       = 4194388
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 805371664

HI       = 0
LO       = 25

R0  [r0] = 0
R1  [at] = 268500992
R2  [v0] = 10
R3  [v1] = 0
R4  [a0] = 1
R5  [a1] = 2147481976
R6  [a2] = 2147481984
R7  [a3] = 0
R8  [t0] = 5
R9  [t1] = 6
R10 [t2] = 55
R11 [t3] = 25
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
```

Data

```
User data segment [10000000]..[10040000]
[10000000]..[1000ffff]  00000000
[10010000]    00000005  00000037  00000000  00000000    . . . . 7 . . . . . . . . . . . .
[10010010]..[1003ffff]  00000000


User Stack [7ffff974]..[80000000]
[7ffff974]    00000001  7ffffa3a  00000000              . . . . : . . . . . . . .
[7ffff980]    7fffffe2  7fffffe0  7fffffd1  7fffffbc    . . . . . . . . . . . . . . . .
[7ffff990]    7ffffff9  7ffffff1  7fffff8d  7fffff7b    . . . . . . . . . . . . . . . {
[7ffff9a0]    7fffff65  7fffff42  7fffff26  7ffffec4    e . . . B . . . & . . . . . . .
[7ffff9b0]    7ffffe8f  7ffffe7e  7ffffe6b  7ffffe39    . . . . ~ . . . k . . . 9 . . .
[7ffff9c0]    7ffffe28  7ffffe16  7ffffe09  7ffffd74    ( . . . . . . . . . . . . t . . .
[7ffff9d0]    7ffffd55  7ffffd4a  7ffffd3f  7ffffd25    U . . . J . . . ? . . . % . . .
[7ffff9e0]    7ffffd11  7ffffcf7  7ffffcce  7ffffca5    . . . . . . . . . . . . . . . .
[7ffff9f0]    7ffffc95  7ffffc82  7ffffc71  7ffffc66    . . . . . . . . q . . . f . . .
[7ffffa00]    7ffffc54  7ffffc41  7ffffc2b  7ffffbff    T . . . A . . . + . . . . . . .
[7ffffa10]    7ffffb96  7ffffb6a  7ffffb53  7ffffb03    . . . . j . . . S . . . . . . .
[7ffffa20]    7ffffac2  7ffffa9f  7ffffa87  7ffffa66    . . . . . . . . . . . . . f . . .
[7ffffa30]    00000000  00000000  682f0000  2f656d6f    . . . . . . . . . . . / h o m e /
[7ffffa40]    656a6172  6172646e  7365442f  706f746b    r a j e n d r a / D e s k t o p
[7ffffa50]    326f632f  2f303136  3842414c  2f57454e    / c o 2 6 1 0 / L A B 8 N E W /
[7ffffa60]    73612e32  4a47006d  45445f53  5f475542    2 . a s m . G J S _ D E B U G _
[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
[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
[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
[7ffffae0]    6168732f  612f6572  696c7070  69746163    / s h a r e / a p p l i c a t i
[7ffffaf0]    2f736e6f  70737471  642e6d69  746b7365    o n s / q t s p i m . d e s k t
[7ffffb00]    5300706f  49535345  4d5f4e4f  47414e41    o p . S E S S I O N _ M A N A G
[7ffffb10]    6c3d5245  6c61636f  6e69532f  403a6867    E R = l o c a l / S i n g h : @
[7ffffb20]    706d742f  43492e2f  6e752d45  312f7869    / t m p / . I C E - u n i x / 1
[7ffffb30]    2c313531  78696e75  6e69532f  2f3a6867    1 5 1 , u n i x / S i n g h : /
[7ffffb40]    2f706d74  4543492e  696e752d  31312f78    t m p / . I C E - u n i x / 1 1
```

CODE WITH NOP :

```
.data                                   #DATA

n:      .word 5
sumOfSquares:   .word 0

.text
.globl main

Main:                                   #MAIN

lw      $t0 , n
nop
li      $t1 ,1
li $t2,0

sumLoop:                                #SUM LOOP
        mul    $t3 , $t1, $t1
        add    $t2 , $t2, $t3

        add $t1, $t1, 1
        ble $t1, $t0 , sumLoop
        Nop                                     #NOP
        sw $t2, sumOfSquares
```

```
li $v0, 10                  #EXITING

syscall
.end main
```

FP Regs | nt Regs [10] | Data | Text

Int Regs [10]

```
PC      = 4194396
EPC     = 0
Cause   = 0
BadVAddr = 0
Status  = 805371664

HI      = 0
LO      = 25

R0  [r0] = 0
R1  [at] = 268500992
R2  [v0] = 10
R3  [v1] = 0
R4  [a0] = 1
R5  [a1] = 2147481976
R6  [a2] = 2147481984
R7  [a3] = 0
R8  [t0] = 5
R9  [t1] = 6
R10 [t2] = 55
R11 [t3] = 25
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
```

Text

```
                                    User Text Segment [00400000]..[00440000]
[00400000] 8fa40000  lw $4, 0($29)           ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004  addiu $5, $29, 4        ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004  addiu $6, $5, 4         ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080  sll $2, $4, 2           ; 186: sll $v0 $a0 2
[00400010] 00c23021  addu $6, $6, $2         ; 187: addu $a2 $a2 $v0
[00400014] 0c100009  jal 0x00400024 [main]   ; 188: jal main
[00400018] 00000000  nop                     ; 189: nop
[0040001c] 3402000a  ori $2, $0, 10          ; 191: li $v0 10
[00400020] 0000000c  syscall                 ; 192: syscall # syscall 10 (exit)
[00400024] 3c011001  lui $1, 4097            ; 10: lw $t0 , n
[00400028] 8c280000  lw $8, 0($1)
[0040002c] 00000000  nop                     ; 11: nop
[00400030] 34090001  ori $9, $0, 1           ; 12: li $t1 ,1
[00400034] 340a0000  ori $10, $0, 0          ; 13: li $t2,0
[00400038] 71295802  mul $11, $9, $9         ; 16: mul $t3 , $t1, $t1
[0040003c] 014b5020  add $10, $10, $11       ; 17: add $t2 , $t2, $t3
[00400040] 21290001  addi $9, $9, 1          ; 19: add $t1, $t1, 1
[00400044] 0109082a  slt $1, $8, $9          ; 20: ble $t1, $t0 , sumLoop
[00400048] 1020fffc  beq $1, $0, -16 [sumLoop-0x00400048]
[0040004c] 00000000  nop                     ; 21: nop
[00400050] 3c011001  lui $1, 4097            ; 22: sw $t2, sumOfSquares
[00400054] ac2a0004  sw $10, 4($1)
[00400058] 3402000a  ori $2, $0, 10          ; 24: li $v0, 10
[0040005c] 0000000c  syscall                 ; 25: syscall

                                    Kernel Text Segment [80000000]..[80010000]
[80000180] 0001d821  addu $27, $0, $1        ; 90: move $k1 $at # Save $at
[80000184] 3c019000  lui $1, -28672          ; 92: sw $v0 s1 # Not re-entrant and we can't trust $sp
[80000188] ac220200  sw $2, 512($1)
[8000018c] 3c019000  lui $1, -28672          ; 93: sw $a0 s2 # But we need to use these registers
[80000190] ac240204  sw $4, 516($1)
[80000194] 401a6800  mfc0 $26, $13           ; 95: mfc0 $k0 $13 # Cause register
[80000198] 001a2082  srl $4, $26, 2          ; 96: srl $a0 $k0 2 # Extract ExcCode Field
[8000019c] 3084001f  andi $4, $4, 31         ; 97: andi $a0 $a0 0x1f
[800001a0] 34020004  ori $2, $0, 4           ; 101: li $v0 4 # syscall 4 (print_str)
[800001a4] 3c049000  lui $4, -28672 [__m1_]  ; 102: la $a0 __m1_
```

FP Regs | nt Regs [10] | Data | Text

Int Regs [10]

```
PC       = 4194396
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 805371664

HI       = 0
LO       = 25

R0  [r0] = 0
R1  [at] = 268500992
R2  [v0] = 10
R3  [v1] = 0
R4  [a0] = 1
R5  [a1] = 2147481976
R6  [a2] = 2147481984
R7  [a3] = 0
R8  [t0] = 5
R9  [t1] = 6
R10 [t2] = 55
R11 [t3] = 25
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
```

Text

```
                                    User Text Segment [00400000]..[00440000]
[00400000] 8fa40000  lw $4, 0($29)           ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004  addiu $5, $29, 4        ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004  addiu $6, $5, 4         ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080  sll $2, $4, 2           ; 186: sll $v0 $a0 2
[00400010] 00c23021  addu $6, $6, $2         ; 187: addu $a2 $a2 $v0
[00400014] 0c100009  jal 0x00400024 [main]   ; 188: jal main
[00400018] 00000000  nop                     ; 189: nop
[0040001c] 3402000a  ori $2, $0, 10          ; 191: li $v0 10
[00400020] 0000000c  syscall                 ; 192: syscall # syscall 10 (exit)
[00400024] 3c011001  lui $1, 4097            ; 10: lw $t0 , n
[00400028] 8c280000  lw $8, 0($1)
[0040002c] 00000000  nop                     ; 11: nop
[00400030] 34090001  ori $9, $0, 1           ; 12: li $t1 ,1
[00400034] 340a0000  ori $10, $0, 0          ; 13: li $t2,0
[00400038] 71295802  mul $11, $9, $9         ; 16: mul $t3 , $t1, $t1
[0040003c] 014b5020  add $10, $10, $11       ; 17: add $t2 , $t2, $t3
[00400040] 21290001  addi $9, $9, 1          ; 19: add $t1, $t1, 1
[00400044] 0109082a  slt $1, $8, $9          ; 20: ble $t1, $t0 , sumLoop
[00400048] 1020fffc  beq $1, $0, -16 [sumLoop-0x00400048]
[0040004c] 00000000  nop                     ; 21: nop
[00400050] 3c011001  lui $1, 4097            ; 22: sw $t2, sumOfSquares
[00400054] ac2a0004  sw $10, 4($1)
[00400058] 3402000a  ori $2, $0, 10          ; 24: li $v0, 10
[0040005c] 0000000c  syscall                 ; 25: syscall

                                    Kernel Text Segment [80000000]..[80010000]
[80000180] 0001d821  addu $27, $0, $1        ; 90: move $k1 $at # Save $at
[80000184] 3c019000  lui $1, -28672          ; 92: sw $v0 s1 # Not re-entrant and we can't trust $sp
[80000188] ac220200  sw $2, 512($1)
[8000018c] 3c019000  lui $1, -28672          ; 93: sw $a0 s2 # But we need to use these registers
[80000190] ac240204  sw $4, 516($1)
[80000194] 401a6800  mfc0 $26, $13           ; 95: mfc0 $k0 $13 # Cause register
[80000198] 001a2082  srl $4, $26, 2          ; 96: srl $a0 $k0 2 # Extract ExcCode Field
[8000019c] 3084001f  andi $4, $4, 31         ; 97: andi $a0 $a0 0x1f
[800001a0] 34020004  ori $2, $0, 4           ; 101: li $v0 4 # syscall 4 (print_str)
[800001a4] 3c049000  lui $4, -28672 [__m1_]  ; 102: la $a0 __m1_
```

QUESTION (3) Illustration of delayed load:

(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
add $t3, $t1, $0
add $t4, $t1, $0
```

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.

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 WITHOUT NOP:

```
.data                                   #DATA
num1 : .word 2
num2 : .word 4

.text                                   #TEXT
.globl main
.ent main
main:

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


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

li $v0 , 10                             #EXITING
syscall

.end main
```

FP Regs | nt Regs [10] | Data | Text

Int Regs [10]

```
PC       = 4194396
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 805371664

HI       = 0
LO       = 25

R0  [r0] = 0
R1  [at] = 268500992
R2  [v0] = 10
R3  [v1] = 0
R4  [a0] = 1
R5  [a1] = 2147481976
R6  [a2] = 2147481984
R7  [a3] = 0
R8  [t0] = 5
R9  [t1] = 6
R10 [t2] = 55
R11 [t3] = 25
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
```

Text

```
                                    User Text Segment [00400000]..[00440000]
[00400000] 8fa40000  lw $4, 0($29)           ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004  addiu $5, $29, 4        ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004  addiu $6, $5, 4         ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080  sll $2, $4, 2           ; 186: sll $v0 $a0 2
[00400010] 00c23021  addu $6, $6, $2         ; 187: addu $a2 $a2 $v0
[00400014] 0c100009  jal 0x00400024 [main]   ; 188: jal main
[00400018] 00000000  nop                     ; 189: nop
[0040001c] 3402000a  ori $2, $0, 10          ; 191: li $v0 10
[00400020] 0000000c  syscall                 ; 192: syscall # syscall 10 (exit)
[00400024] 3c011001  lui $1, 4097            ; 10: lw $t0 , n
[00400028] 8c280000  lw $8, 0($1)
[0040002c] 00000000  nop                     ; 11: nop
[00400030] 34090001  ori $9, $0, 1           ; 12: li $t1 ,1
[00400034] 340a0000  ori $10, $0, 0          ; 13: li $t2,0
[00400038] 71295802  mul $11, $9, $9         ; 16: mul $t3 , $t1, $t1
[0040003c] 014b5020  add $10, $10, $11       ; 17: add $t2 , $t2, $t3
[00400040] 21290001  addi $9, $9, 1          ; 19: add $t1, $t1, 1
[00400044] 0109082a  slt $1, $8, $9          ; 20: ble $t1, $t0 , sumLoop
[00400048] 1020fffc  beq $1, $0, -16 [sumLoop-0x00400048]
[0040004c] 00000000  nop                     ; 21: nop
[00400050] 3c011001  lui $1, 4097            ; 22: sw $t2, sumOfSquares
[00400054] ac2a0004  sw $10, 4($1)
[00400058] 3402000a  ori $2, $0, 10          ; 24: li $v0, 10
[0040005c] 0000000c  syscall                 ; 25: syscall

                                    Kernel Text Segment [80000000]..[80010000]
[80000180] 0001d821  addu $27, $0, $1        ; 90: move $k1 $at # Save $at
[80000184] 3c019000  lui $1, -28672          ; 92: sw $v0 s1 # Not re-entrant and we can't trust $sp
[80000188] ac220200  sw $2, 512($1)
[8000018c] 3c019000  lui $1, -28672          ; 93: sw $a0 s2 # But we need to use these registers
[80000190] ac240204  sw $4, 516($1)
[80000194] 401a6800  mfc0 $26, $13           ; 95: mfc0 $k0 $13 # Cause register
[80000198] 001a2082  srl $4, $26, 2          ; 96: srl $a0 $k0 2 # Extract ExcCode Field
[8000019c] 3084001f  andi $4, $4, 31         ; 97: andi $a0 $a0 0x1f
[800001a0] 34020004  ori $2, $0, 4           ; 101: li $v0 4 # syscall 4 (print_str)
[800001a4] 3c049000  lui $4, -28672 [__m1_]  ; 102: la $a0 __m1_
```

CODE WITH NOP :

```
.data                                   #DATA
num1 : .word 2
num2 : .word 4

.text                                   #TEXT
.globl main
.ent main
Main:                                   #MAIN

lw $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 Regs | nt Regs [10] | Data | Text

Int Regs [10]

```
PC       = 4194388
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 805371664

HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 268500992
R2  [v0] = 10
R3  [v1] = 0
R4  [a0] = 1
R5  [a1] = 2147481976
R6  [a2] = 2147481984
R7  [a3] = 0
R8  [t0] = 0
R9  [t1] = 4
R10 [t2] = 4
R11 [t3] = 4
R12 [t4] = 4
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
```

Text

```
                                    User Text Segment [00400000]..[00440000]
[00400000] 8fa40000  lw $4, 0($29)          ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004  addiu $5, $29, 4       ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004  addiu $6, $5, 4        ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080  sll $2, $4, 2          ; 186: sll $v0 $a0 2
[00400010] 00c23021  addu $6, $6, $2        ; 187: addu $a2 $a2 $v0
[00400014] 0c100009  jal 0x00400024 [main]  ; 188: jal main
[00400018] 00000000  nop                    ; 189: nop
[0040001c] 3402000a  ori $2, $0, 10         ; 191: li $v0 10
[00400020] 0000000c  syscall                ; 192: syscall # syscall 10 (exit)
[00400024] 3c011001  lui $1, 4097           ; 10: lw $t1, num1
[00400028] 8c290000  lw $9, 0($1)
[0040002c] 00000000  nop                    ; 11: nop
[00400030] 3c011001  lui $1, 4097           ; 12: lw $t2, num2
[00400034] 8c2a0004  lw $10, 4($1)
[00400038] 00000000  nop                    ; 13: nop
[0040003c] 3c011001  lui $1, 4097           ; 14: lw $t1, num2
[00400040] 8c290004  lw $9, 4($1)
[00400044] 00000000  nop                    ; 15: nop
[00400048] 01205820  add $11, $9, $0        ; 16: add $t3, $t1, $0
[0040004c] 01206020  add $12, $9, $0        ; 17: add $t4, $t1, $0
[00400050] 3402000a  ori $2, $0, 10         ; 18: li $v0 , 10
[00400054] 0000000c  syscall                ; 19: syscall

                                    Kernel Text Segment [80000000]..[80010000]
[80000180] 0001d821  addu $27, $0, $1       ; 90: move $k1 $at # Save $at
[80000184] 3c019000  lui $1, -28672         ; 92: sw $v0 s1 # Not re-entrant and we can't trust $sp
[80000188] ac220200  sw $2, 512($1)
[8000018c] 3c019000  lui $1, -28672         ; 93: sw $a0 s2 # But we need to use these registers
[80000190] ac240204  sw $4, 516($1)
[80000194] 401a6800  mfc0 $26, $13          ; 95: mfc0 $k0 $13 # Cause register
[80000198] 001a2082  srl $4, $26, 2         ; 96: srl $a0 $k0 2 # Extract ExcCode Field
[8000019c] 3084001f  andi $4, $4, 31        ; 97: andi $a0 $a0 0x1f
[800001a0] 34020004  ori $2, $0, 4          ; 101: li $v0 4 # syscall 4 (print_str)
[800001a4] 3c049000  lui $4, -28672 [__m1_] ; 102: la $a0 __m1_
[800001a8] 0000000c  syscall                ; 103: syscall
[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 :

```
.data                                   #DATA
Mult_arr:                               #MATIX
        .word 6 , 3
        .word 4 , 0

size:  .word 2
ans:   .word 0
DATASIZE = 4                            #DATASIZE
#   addr = baseAddr+(rowIndex*colSize+colIndex) * dataSize

.text                                   #TEXT
.globl main
.ent main
Main:                                   #MAIN
        la $a0 , mult_arr
        lw $a1 , size
        jal det                         #DET CALLED
        move $a0 , $v0
        li $v0 , 1
        syscall
        sw $a0 , ans
        li $v0, 10
        syscall
.end main                               #MAIN END
```

```
                                        # 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
        lw $t0 , ($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
        lw $t1 , ($t1)
        # # # # # # # # # # #
        mul $t0 , $t0 , $t1
        sub $v0 , $v0 , $t0
        jr $ra
.end det                                #END DET
```

FP Regs | nt Regs [10] | Data | Text

Int Regs [10]

```
PC       = 4194380
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 805371664

HI       = 0
LO       = 12

R0  [r0] = 0
R1  [at] = 268500992
R2  [v0] = 10
R3  [v1] = 0
R4  [a0] = -12
R5  [a1] = 2
R6  [a2] = 2147481984
R7  [a3] = 0
R8  [t0] = 12
R9  [t1] = 4
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
```

Text

```
                                        User Text Segment [00400000]..[00440000]
[00400000] 8fa40000  lw $4, 0($29)           ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004  addiu $5, $29, 4        ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004  addiu $6, $5, 4         ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080  sll $2, $4, 2           ; 186: sll $v0 $a0 2
[00400010] 00c23021  addu $6, $6, $2         ; 187: addu $a2 $a2 $v0
[00400014] 0c100009  jal 0x00400024 [main]   ; 188: jal main
[00400018] 00000000  nop                     ; 189: nop
[0040001c] 3402000a  ori $2, $0, 10          ; 191: li $v0 10
[00400020] 0000000c  syscall                 ; 192: syscall # syscall 10 (exit)
[00400024] 3c041001  lui $4, 4097 [mult_arr] ; 17: la $a0 , mult_arr
[00400028] 3c011001  lui $1, 4097            ; 18: lw $a1 , size
[0040002c] 8c250010  lw $5, 16($1)
[00400030] 0c100014  jal 0x00400050 [det]    ; 19: jal det
[00400034] 00022021  addu $4, $0, $2         ; 20: move $a0 , $v0
[00400038] 34020001  ori $2, $0, 1           ; 21: li $v0 , 1
[0040003c] 0000000c  syscall                 ; 22: syscall
[00400040] 3c011001  lui $1, 4097            ; 23: sw $a0 , ans
[00400044] ac240014  sw $4, 20($1)
[00400048] 3402000a  ori $2, $0, 10          ; 24: li $v0, 10
[0040004c] 0000000c  syscall                 ; 25: syscall
[00400050] 8c820000  lw $2, 0($4)            ; 41: lw $v0 , ($a0)
[00400054] 20a80001  addi $8, $5, 1          ; 43: addi $t0 , $a1 , 1
[00400058] 34010004  ori $1, $0, 4           ; 44: mul $t0 , $t0 , DATASIZE
[0040005c] 71014002  mul $8, $8, $1
[00400060] 01044020  add $8, $8, $4          ; 45: add $t0 , $t0 , $a0
[00400064] 8d080000  lw $8, 0($8)            ; 46: lw $t0 , ($t0)
[00400068] 71021002  mul $2, $8, $2          ; 47: mul $v0 , $t0, $v0
[0040006c] 34080004  ori $8, $0, 4           ; 49: li $t0 , DATASIZE
[00400070] 01044020  add $8, $8, $4          ; 50: add $t0 , $t0 , $a0
[00400074] 8d080000  lw $8, 0($8)            ; 51: lw $t0 , ($t0)
[00400078] 00054821  addu $9, $0, $5         ; 53: move $t1 , $a1
[0040007c] 34010004  ori $1, $0, 4           ; 54: mul $t1 , $t1 , DATASIZE
[00400080] 71214802  mul $9, $9, $1
[00400084] 01244820  add $9, $9, $4          ; 55: add $t1 , $t1 , $a0
[00400088] 8d290000  lw $9, 0($9)            ; 56: lw $t1 , ($t1)
[0040008c] 71094002  mul $8, $8, $9          ; 58: mul $t0 , $t0 , $t1
```

Console

-12

CODE WITH NOP :

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

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                                     #NOP AFTER LOAD
        mul $v0 , $t0, $v0
        # (0,1)
        li $t0 , DATASIZE
        add $t0 , $t0 , $a0
        lw $t0 , ($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
```

FP Regs | nt Regs [10] | Data | Text

Int Regs [10]

```
PC       = 4194392
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 805371664

HI       = 0
LO       = 12

R0  [r0] = 0
R1  [at] = 268500992
R2  [v0] = 10
R3  [v1] = 0
R4  [a0] = -12
R5  [a1] = 2
R6  [a2] = 2147481976
R7  [a3] = 0
R8  [t0] = 12
R9  [t1] = 4
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
```

Text

```
                                    User Text Segment [00400000]..[00440000]
[00400000] 8fa40000  lw $4, 0($29)           ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004  addiu $5, $29, 4        ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004  addiu $6, $5, 4         ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080  sll $2, $4, 2           ; 186: sll $v0 $a0 2
[00400010] 00c23021  addu $6, $6, $2         ; 187: addu $a2 $a2 $v0
[00400014] 0c100009  jal 0x00400024 [main]   ; 188: jal main
[00400018] 00000000  nop                     ; 189: nop
[0040001c] 3402000a  ori $2, $0, 10          ; 191: li $v0 10
[00400020] 0000000c  syscall                 ; 192: syscall # syscall 10 (exit)
[00400024] 3c041001  lui $4, 4097 [mult_arr] ; 17: la $a0 , mult_arr
[00400028] 00000000  nop                     ; 18: nop
[0040002c] 3c011001  lui $1, 4097            ; 19: lw $a1 , size
[00400030] 8c250010  lw $5, 16($1)
[00400034] 00000000  nop                     ; 20: nop
[00400038] 0c100017  jal 0x0040005c [det]    ; 21: jal det
[0040003c] 00000000  nop                     ; 22: nop
[00400040] 00022021  addu $4, $0, $2         ; 23: move $a0 , $v0
[00400044] 34020001  ori $2, $0, 1           ; 24: li $v0 , 1
[00400048] 0000000c  syscall                 ; 25: syscall
[0040004c] 3c011001  lui $1, 4097            ; 26: sw $a0 , ans
[00400050] ac240014  sw $4, 20($1)
[00400054] 3402000a  ori $2, $0, 10          ; 27: li $v0, 10
[00400058] 0000000c  syscall                 ; 28: syscall
[0040005c] 00000000  nop                     ; 44: nop
[00400060] 8c820000  lw $2, 0($4)            ; 45: lw $v0 , ($a0)
[00400064] 00000000  nop                     ; 46: nop
[00400068] 20a80001  addi $8, $5, 1          ; 48: addi $t0 , $a1 , 1
[0040006c] 34010004  ori $1, $0, 4           ; 49: mul $t0 , $t0 , DATASIZE
[00400070] 71014002  mul $8, $8, $1
[00400074] 01044020  add $8, $8, $4          ; 50: add $t0 , $t0 , $a0
[00400078] 8d080000  lw $8, 0($8)            ; 51: lw $t0 , ($t0)
[0040007c] 00000000  nop                     ; 52: nop
[00400080] 71021002  mul $2, $8, $2          ; 53: mul $v0 , $t0, $v0
[00400084] 34080004  ori $8, $0, 4           ; 55: li $t0 , DATASIZE
[00400088] 01044020  add $8, $8, $4          ; 56: add $t0 , $t0 , $a0
[0040008c] 8d080000  lw $8, 0($8)            ; 57: lw $t0 , ($t0)
```

Console

-12