2020 Computer Architecture Project

배한준

qwerty2901@korea.ac.kr

공학관 236호

Project

▣ Main Purpose of this Project

- Understand MIPS assembly language and Recursive Function algorithm
- **QtSpim**
  - It reads and executes assembly language programs
  - It contains a simple debugger
  - Using this program, it's possible to run a MIPS assembly code
  - Download Link
    - https://sourceforge.net/projects/spimsimulator/files/
    - You can download the latest version for your operating system.

▣ You can use developer guide documents in reference folder

- MIPStestSMv11.pdf
- QtSpim-User-Manual-Final-Draft.pdf

▣ Due Date : 2020. 6. 5.

Start SPIM

QtSpim

File Simulator Registers Text Segment Data Segment Window Help

FP Regs | Int Regs [16] | Text | Data

Int Regs [16]

```
PC       = 0
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 3000ff10

HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 0
R2  [v0] = 0
R3  [v1] = 0
R4  [a0] = 1
R5  [a1] = 7ffff578
R6  [a2] = 7ffff580
R7  [a3] = 0
R8  [t0] = 0
R9  [t1] = 0
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
```

Register visualization

Text

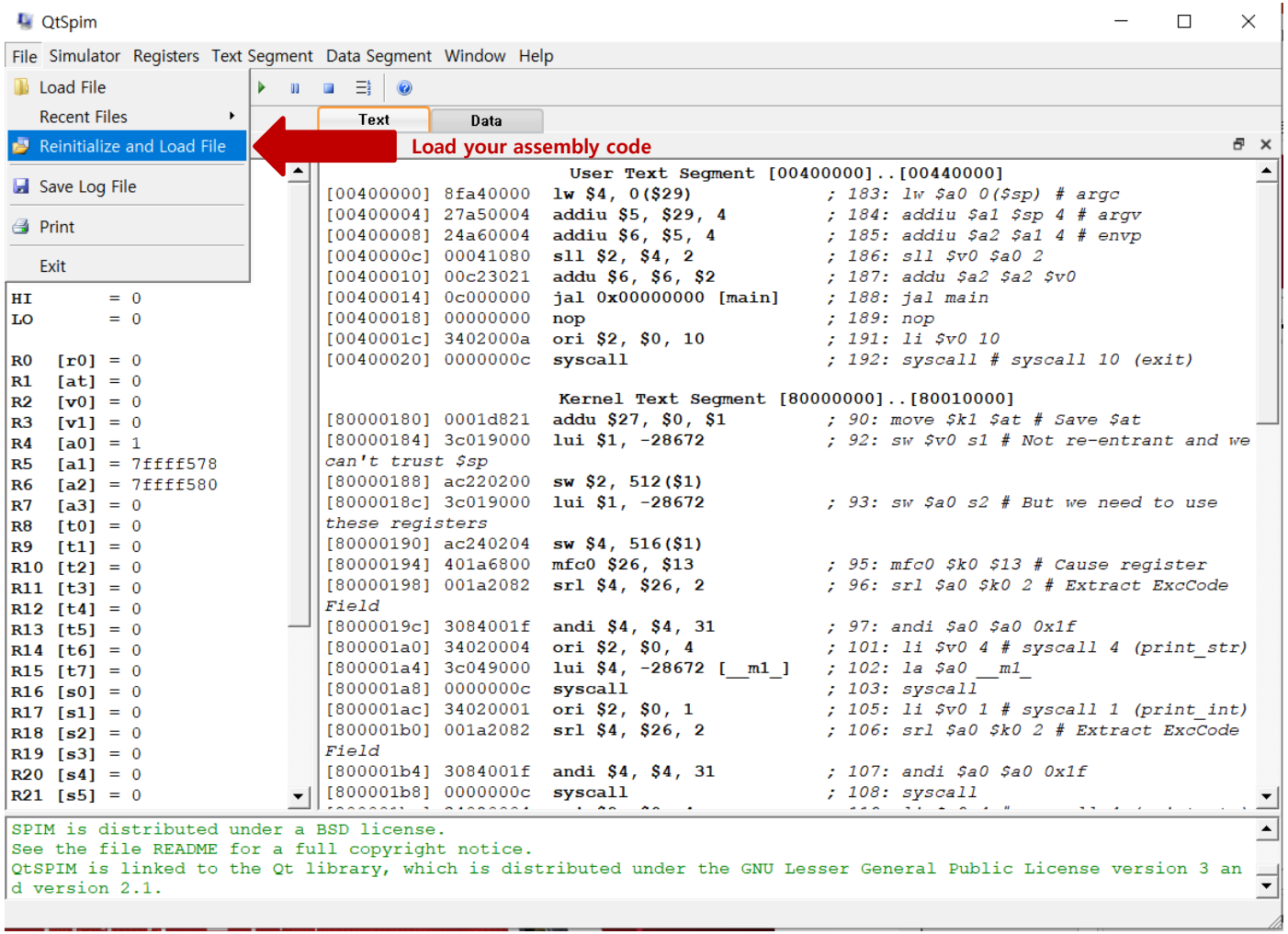
```
                    User Text Segment [00400000]..[00440000]
[00400000] 8fa40000  lw $4, 0($29)              ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004  addiu $5, $29, 4           ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004  addiu $6, $5, 4            ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080  sll $2, $4, 2              ; 186: sll $v0 $a0 2
[00400010] 00c23021  addu $6, $6, $2            ; 187: addu $a2 $a2 $v0
[00400014] 0c000000  jal 0x00000000 [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)

                    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)
[800001b0] 001a2082  srl $4, $26, 2             ; 106: srl $a0 $k0 2 # Extract ExcCode
Field
[800001b4] 3084001f  andi $4, $4, 31            ; 107: andi $a0 $a0 0x1f
[800001b8] 0000000c  syscall                    ; 108: syscall
```

Program Code

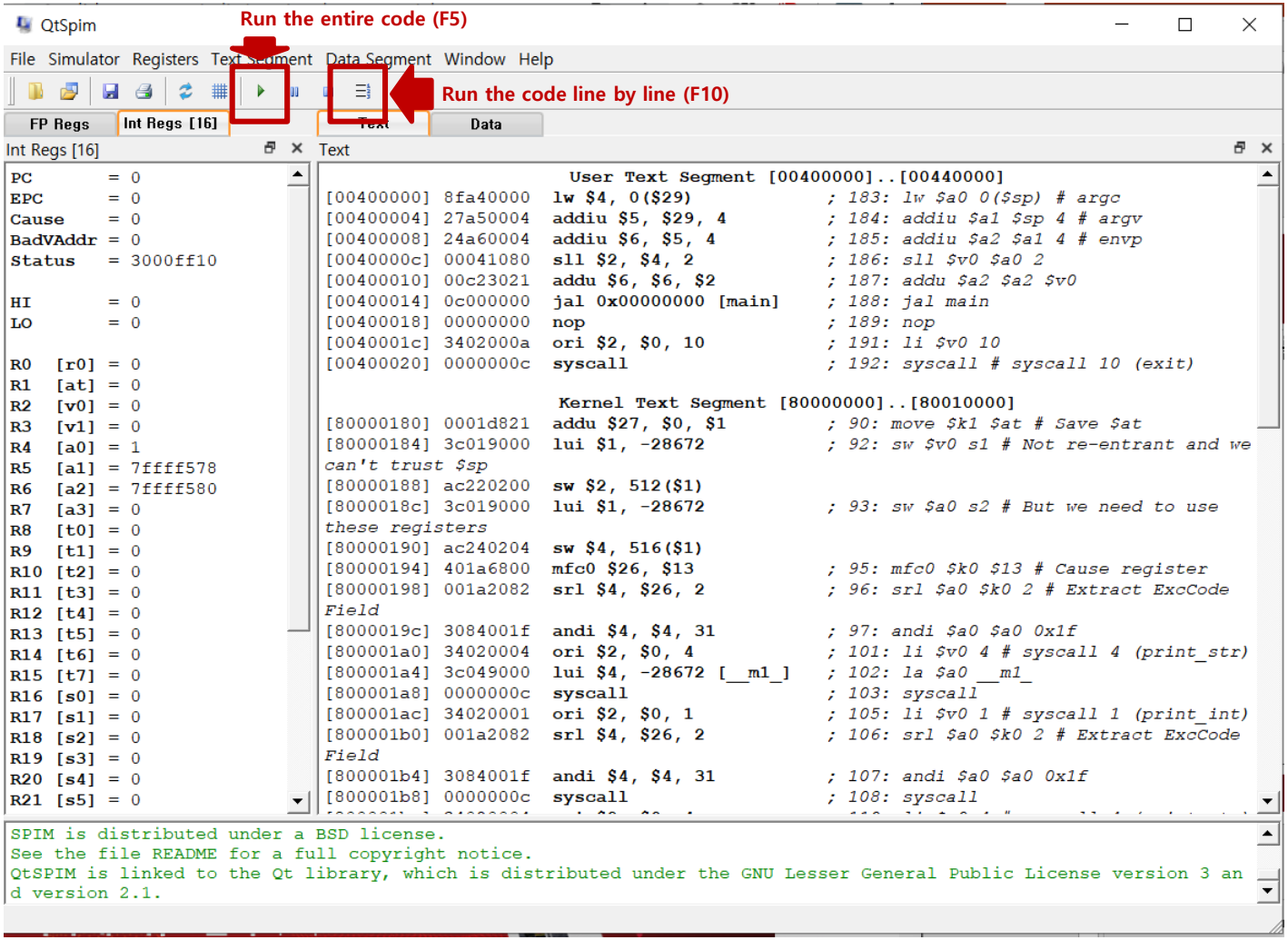
```
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 an
d version 2.1.
```

Messages

Load Program



How to run program



Execute program

QtSpim

File Simulator Registers Text Segment Data Segment Window Help

FP Regs | Int Regs [16] | Text | Data

Int Regs [16]

```
PC       = 400004
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 3000ff10

HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 0
R2  [v0] = 0
R3  [v1] = 0
R4  [a0] = 1
R5  [a1] = 7ffff578
R6  [a2] = 7ffff580
R7  [a3] = 0
R8  [t0] = 0
R9  [t1] = 0
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
```

Register change resulting from previous instruction

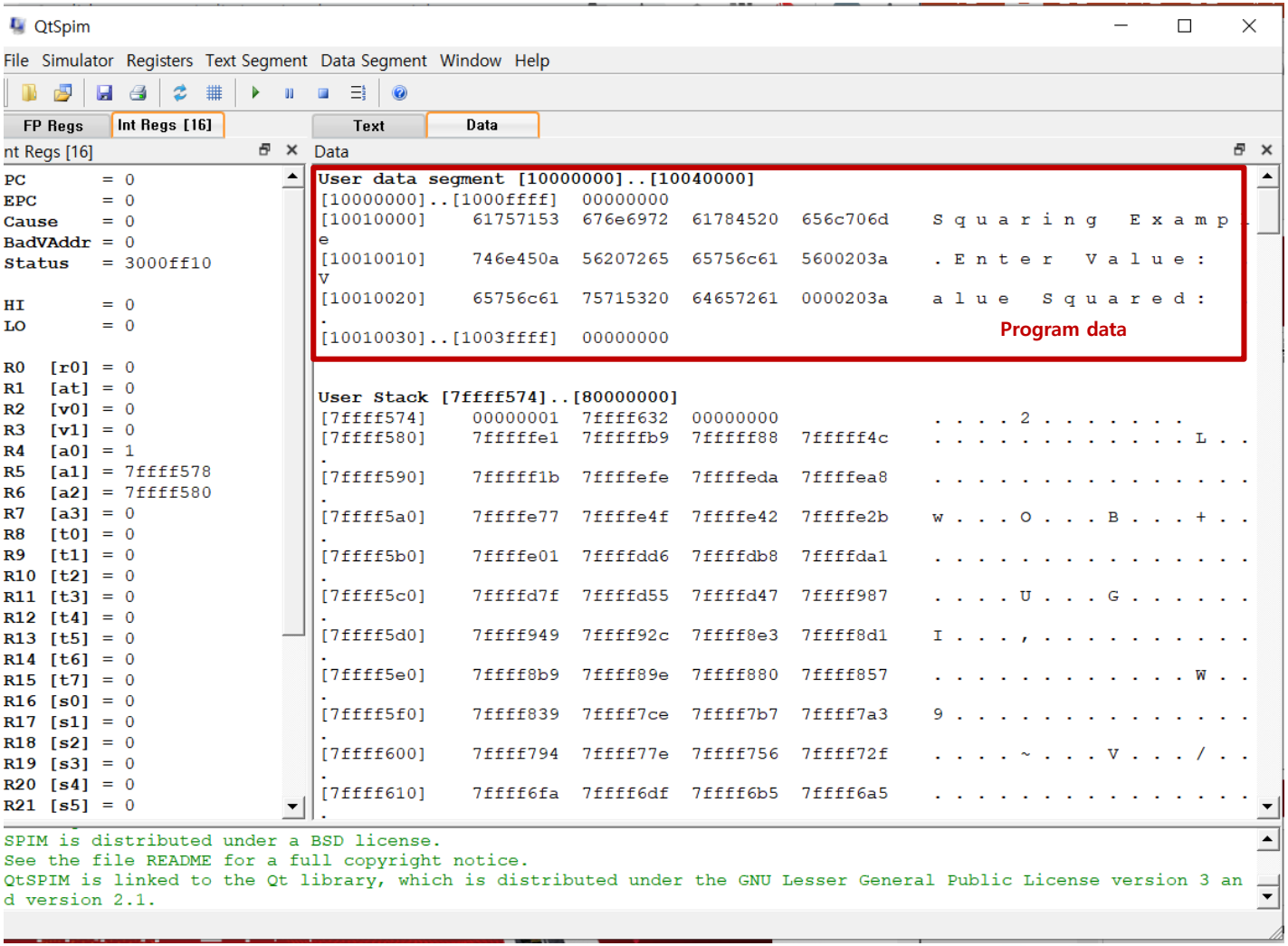
Text

```
                    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] 34020004  ori $2, $0, 4              ; 19: li $v0, 4 # call code for print
string
[00400028] 3c041001  lui $4, 4097 [hdr]         ; 20: la $a0, hdr # addr of NULL
terminated str
[0040002c] 0000000c  syscall                    ; 21: syscall # system call
[00400030] 34020005  ori $2, $0, 5              ; 22: li $v0, 5 # call code for read
integer
[00400034] 0000000c  syscall                    ; 23: syscall # system call (result in
$v0)
[00400038] 70424002  mul $8, $2, $2             ; 24: mul $t0, $v0, $v0 # square answer
[0040003c] 3c011001  lui $1, 4097               ; 25: sw $t0, value # save to variable
[00400040] ac280030  sw $8, 48($1)
[00400044] 34020004  ori $2, $0, 4              ; 26: li $v0, 4 # call code for print
string
[00400048] 3c011001  lui $1, 4097 [ansMsg]      ; 27: la $a0, ansMsg # addr of NULL
terminated str
[0040004c] 3424001f  ori $4, $1, 31 [ansMsg]
[00400050] 0000000c  syscall                    ; 28: syscall # system call
[00400054] 34020001  ori $2, $0, 1              ; 29: li $v0, 1 # call code for print
integer
[00400058] 3c011001  lui $1, 4097               ; 30: lw $a0, value # value for integer
```

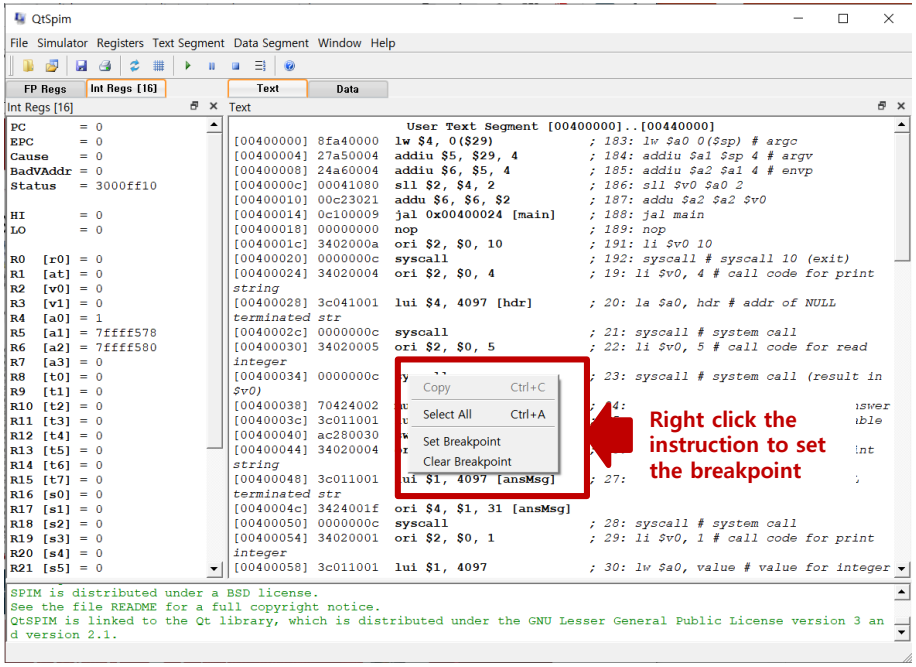
Current instruction

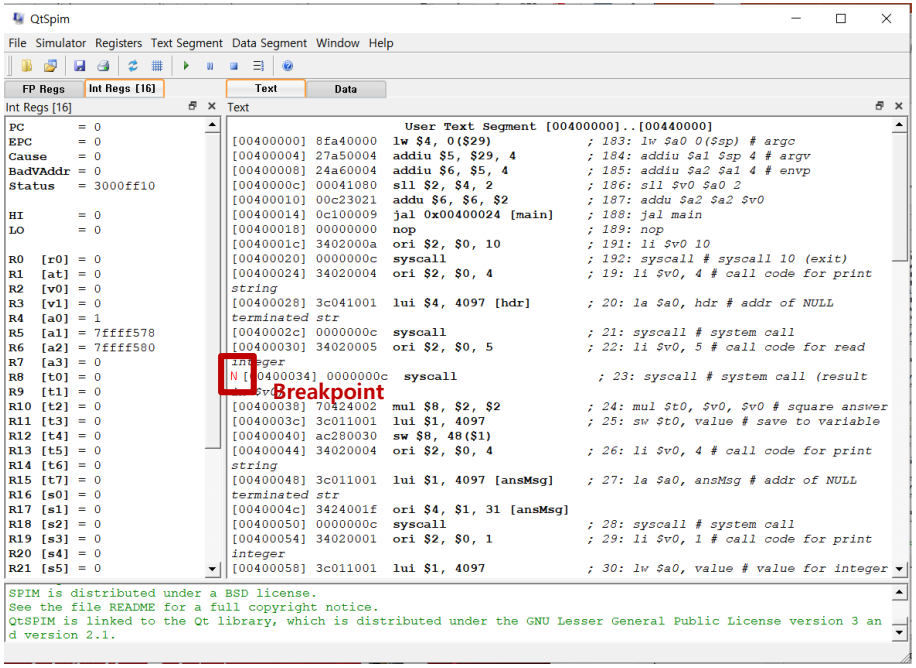
```
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 an
d version 2.1.
```

Program data



Set a breakpoint





How to write MIPS assembly language programs

Section

- The ".globl name" and ".ent name" directives are used to define the name of the initial or main procedure.

Label

- Labels are code locations, typically used as a function/procedure name or as the target of a jump.
  - Must start with a letter
  - May be followed by letters, numbers, or an "_" (underscore).
  - Must be terminated with a ":" (colon).
  - May only be defined once.

```
# Example program to display an array.
# Demonstrates use of QtSpim system service calls.
# ---------------------------------------------------
# Data Declarations

 .data
hdr:     .ascii  "Squaring Example₩n"
         .asciiz     "Enter Value: "
ansMsg:  .asciiz     "Value Squared: "
value:   .word   0

# ---------------------------------------------------
# text/code section

 .text
 .globl  main
 .ent    main
main:
li  $v0, 4      # call code for print string
la  $a0, hdr    # addr of NULL terminated str
syscall         # system call
li  $v0, 5      # call code for read integer
syscall         # system call (result in $v0)
mul     $t0, $v0, $v0 # square answer
sw  $t0, value  # save to variable
li  $v0, 4      # call code for print string
la  $a0, ansMsg     # addr of NULL terminated str
syscall         # system call
li $v0, 1       # call code for print integer
lw $a0, value       # value for integer to print
syscall         # system call
# -----
# Done, terminate program.
    li $v0, 10      # terminate
   syscall          # system call
    .end main
```

Example - Print and scan

```
# Example program to display an array.
# Demonstrates use of QtSpim system service calls.
# ---------------------------------------------------
# Data Declarations

 .data
hdr:     .ascii  "Squaring Example₩n"
    .asciiz     "Enter Value: "
ansMsg: .asciiz     "Value Squared: "
value:  .word   0

# ---------------------------------------------------
# text/code section

 .text
 .globl  main
 .ent    main
main:
 li  $v0, 4      # call code for print string
 la  $a0, hdr    # addr of NULL terminated str
 syscall         # system call
 li  $v0, 5      # call code for read integer
 syscall         # system call (result in $v0)
 mul     $t0, $v0, $v0 # square answer
 sw  $t0, value  # save to variable
 li  $v0, 4      # call code for print string
 la  $a0, ansMsg     # addr of NULL terminated str
 syscall         # system call
 li $v0, 1       # call code for print integer
 lw $a0, value       # value for integer to print
 syscall         # system call
 # -----
# Done, terminate program.
    li $v0, 10      # terminate
   syscall         # system call
    .end main
```

```
User data segment [10000000]..[10040000]
[10000000]..[1000ffff]  00000000
[10010000]    61757153  676e6972  61784520  656c706d    S q u a r i n g   E x a m p l
e
[10010010]    746e450a  56207265  65756c61  5600203a    . E n t e r   V a l u e :   .
V
[10010020]    65756c61  75715320  64657261  0000203a    a l u e   S q u a r e d :   .
.
[10010030]..[1003ffff]  00000000
```

Print 'hdr' label

Scan data from console

Square the data and save to 'value'

Print 'ansMsg' label

Print 'value'

Terminate program

Console

```
Squaring Example
Enter Value:
```

Console

```
Squaring Example
Enter Value: 3
Value Squared: 9
```

프로젝트 문제

- **1. 3X3 역행렬**
  - 숫자는 아무거나 상관없음(미리 저장 or 실행시킨 후 입력)
- **2. Heap sort**
  - 입력한 숫자가 Heap 의 마지막 노드로 추가가 되고, 즉시 Heap Sort로 내림차순 정렬 후 출력시켜주는 프로그램

입력 예시 1 : 4	출력 예시 1 : 4
입력 예시 2 : 10	출력 예시 2 : 10 4
입력 예시 3 : 7	출력 예시 3 : 10 7 4
입력 예시 4 : 15	출력 예시 4 : 15 10 7 4
입력 예시 5 : 12	출력 예시 5 : 15 12 10 7 4

  - 알고리즘 설명은 아래 사이트 참고
    https://ratsgo.github.io/data%20structure&algorithm/2017/09/27/heapsort/

What to submit

- **A code file(.s) : 40%**
  - Include detailed comments inside your code

- **A WORD/HWP document that describes your algorithm : 60%**
  - File name is same as code file, ex) 학번_이름.xxx
  - You can use any word processor.
  - You must explain your code and algorithm in detail.
  - Your document will be used to determine partial credit if the code fails to run.
  - Save your files in [student number] folder, and zip.
  - Follow this example. ex) 학번_이름.zip

- **Submit your zip file through BlackBoard**

Q & A