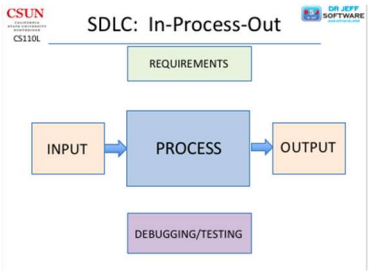
Student: Joshua Cohen **Instructor: Dr. Jeff Drobman**

Lab #: 4 Software Interrupts

LAB FORM



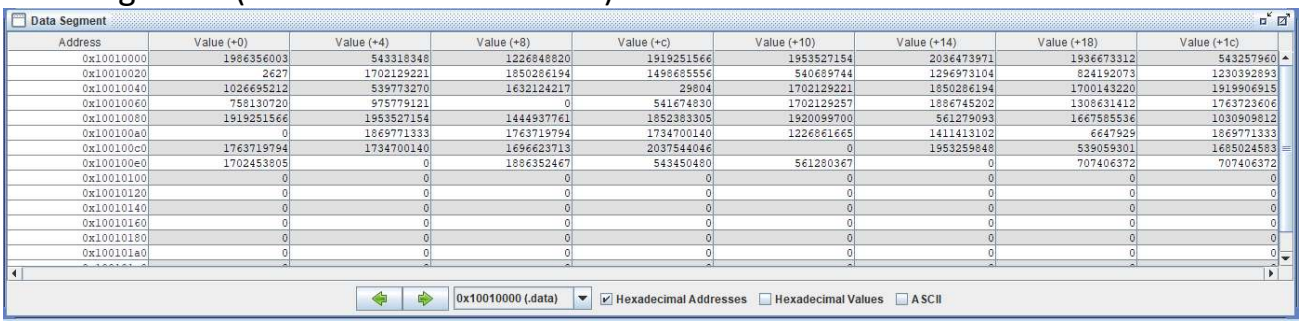
Requirements

1. Decode Pending Interrupts
2. Allocate memory for Handlers
3. Use Jump Table
    a) Order by Priority
    b) Test and Jump
    c) Handlers as subroutines: jal -> jr $ra

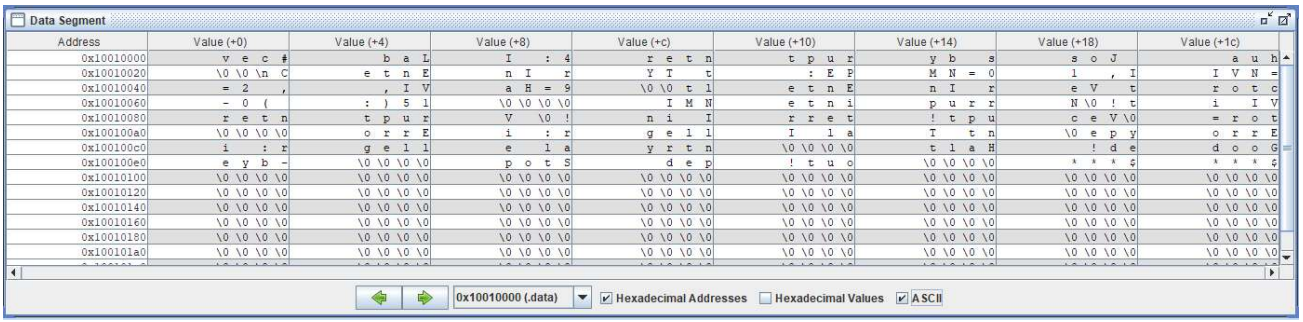
Inputs

paste here screenshots of all inputs (not code): .data segment & GUI/console

Data Segment: (Hexadecimal Addresses)



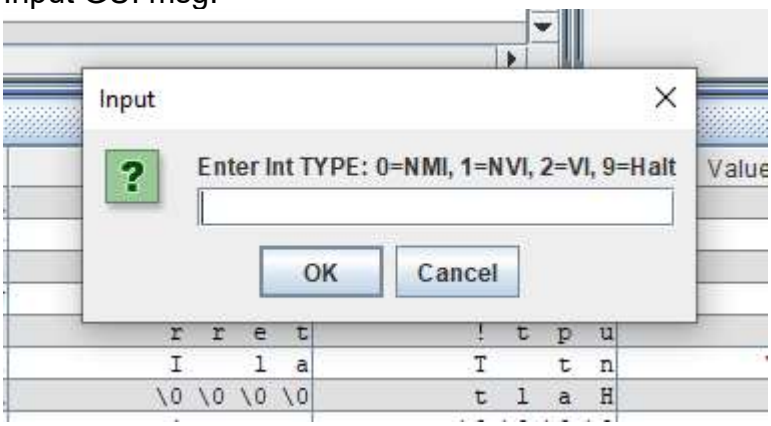
(ASCII):



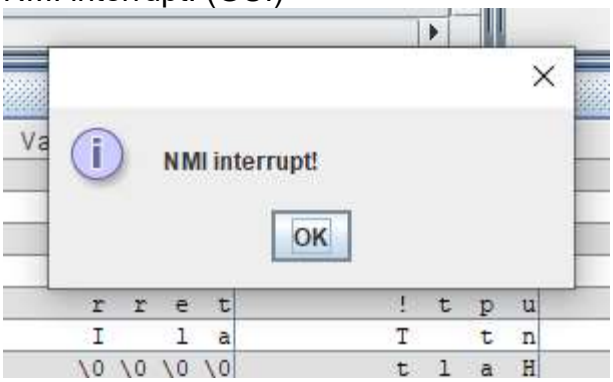
Outputs (note: you must execute the code to get outputs)

paste here screenshots of all outputs (not code): console/GUI, .data, .heap, registers

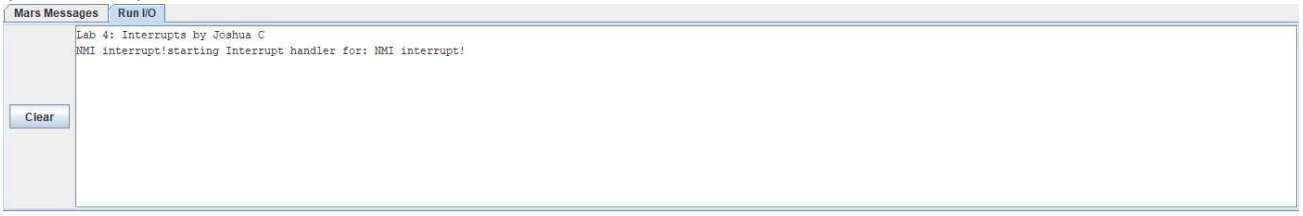
Input GUI msg:



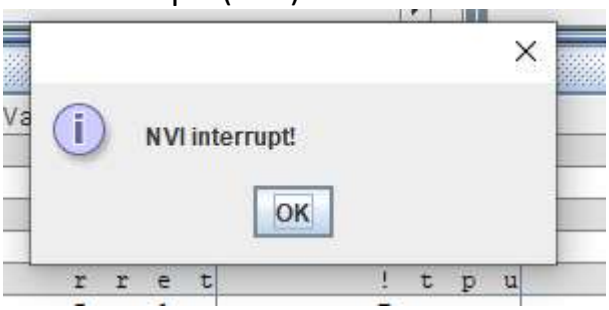
NMI interrupt: (GUI)



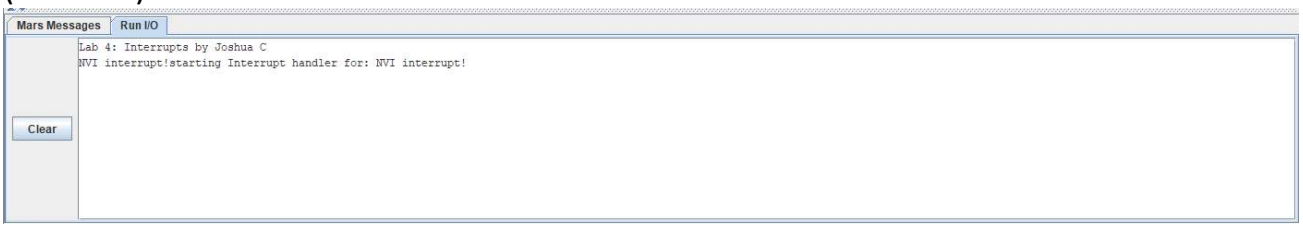
(Console):



NVI interrupt: (GUI)

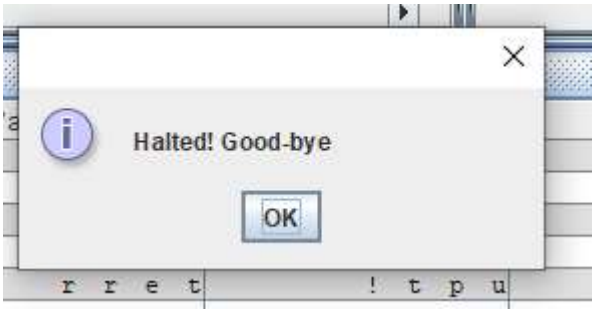


(Console):

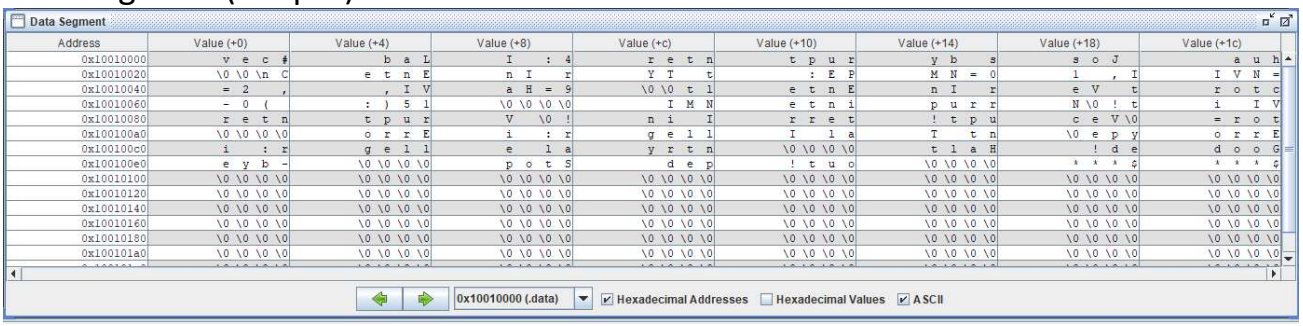


VI Vector Interrupt:
Console Vectors Outputs (V0-V3):

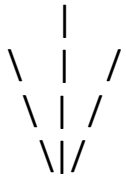
Halt:
Console & GUI Halt msg:



Data Segment (Output) ASCII:

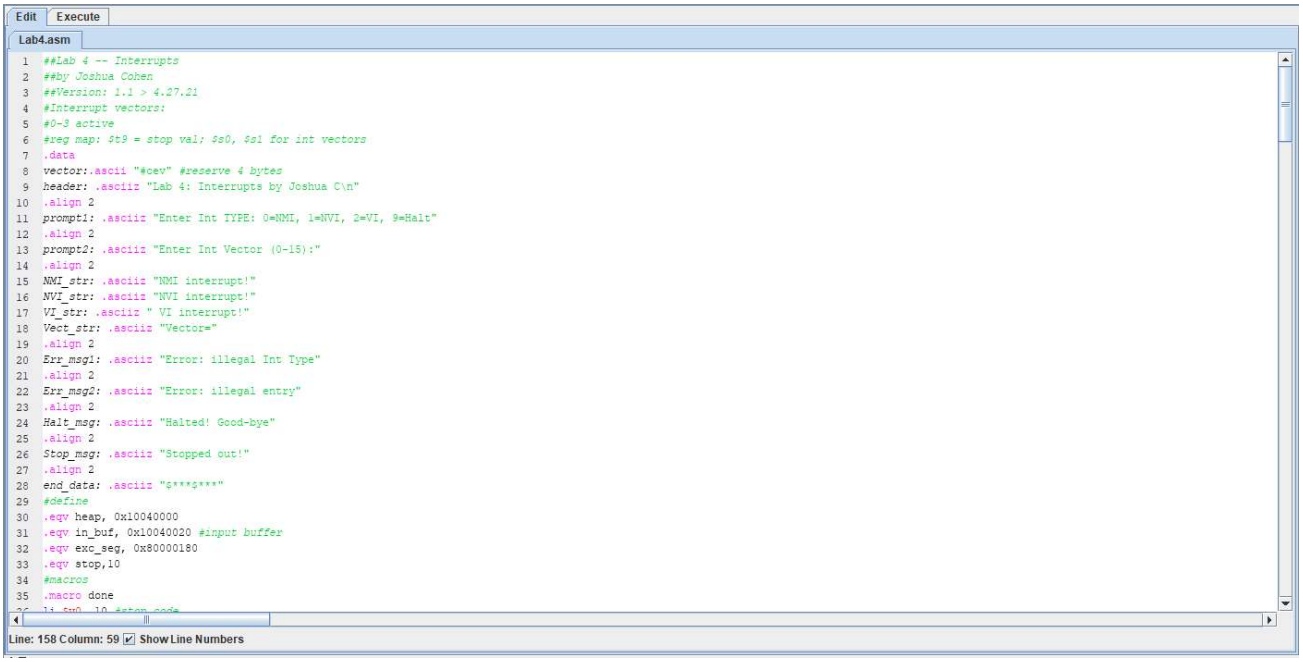


Skip to next page for source code



Process (algorithms): *source code*

(paste original syntax colored view here; include line numbers – make legible)

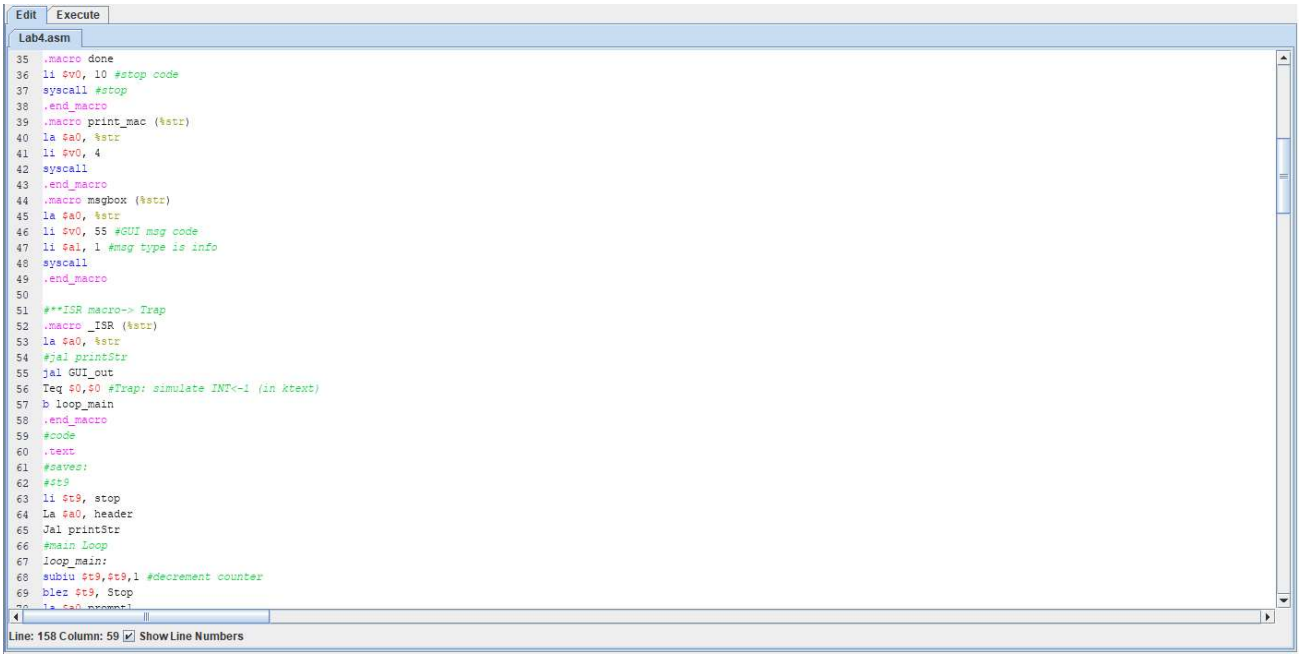


Edit Execute

Lab4.asm

```
1  ##Lab 4 -- Interrupts
2  ##by Joshua Cohen
3  ##Version: 1.1 > 4.27.21
4  #Interrupt vectors:
5  #0-3 active
6  #reg map: $t9 = stop val; $s0, $s1 for int vectors
7  .data
8  vector:.ascii "#cev" #reserve 4 bytes
9  header: .asciiz "Lab 4: Interrupts by Joshua C\n"
10 .align 2
11 prompt1: .asciiz "Enter Int TYPE: 0=NMI, 1=NVI, 2=VI, 9=Halt"
12 .align 2
13 prompt2: .asciiz "Enter Int Vector (0-15):"
14 .align 2
15 NMI_str: .asciiz "NMI interrupt!"
16 NVI_str: .asciiz "NVI interrupt!"
17 VI_str: .asciiz " VI interrupt!"
18 Vect_str: .asciiz "Vector="
19 .align 2
20 Err_msg1: .asciiz "Error: illegal Int Type"
21 .align 2
22 Err_msg2: .asciiz "Error: illegal entry"
23 .align 2
24 Halt_msg: .asciiz "Halted! Good-bye"
25 .align 2
26 Stop_msg: .asciiz "Stopped out!"
27 .align 2
28 end_data: .asciiz "s***s***"
29 #define
30 .eqv heap, 0x10040000
31 .eqv in_buf, 0x10040020 #input buffer
32 .eqv exc_seg, 0x80000180
33 .eqv stop,10
34 #macros
35 .macro done
```

Line: 158 Column: 59 Show Line Numbers

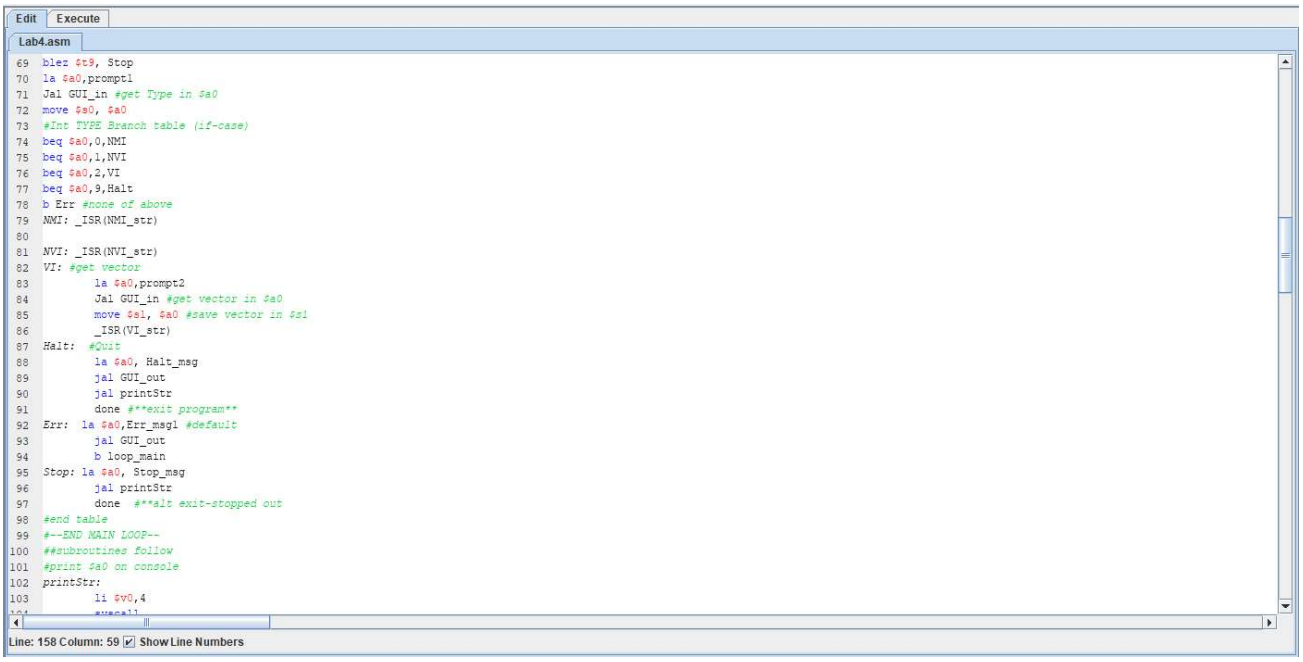


Edit Execute

Lab4.asm

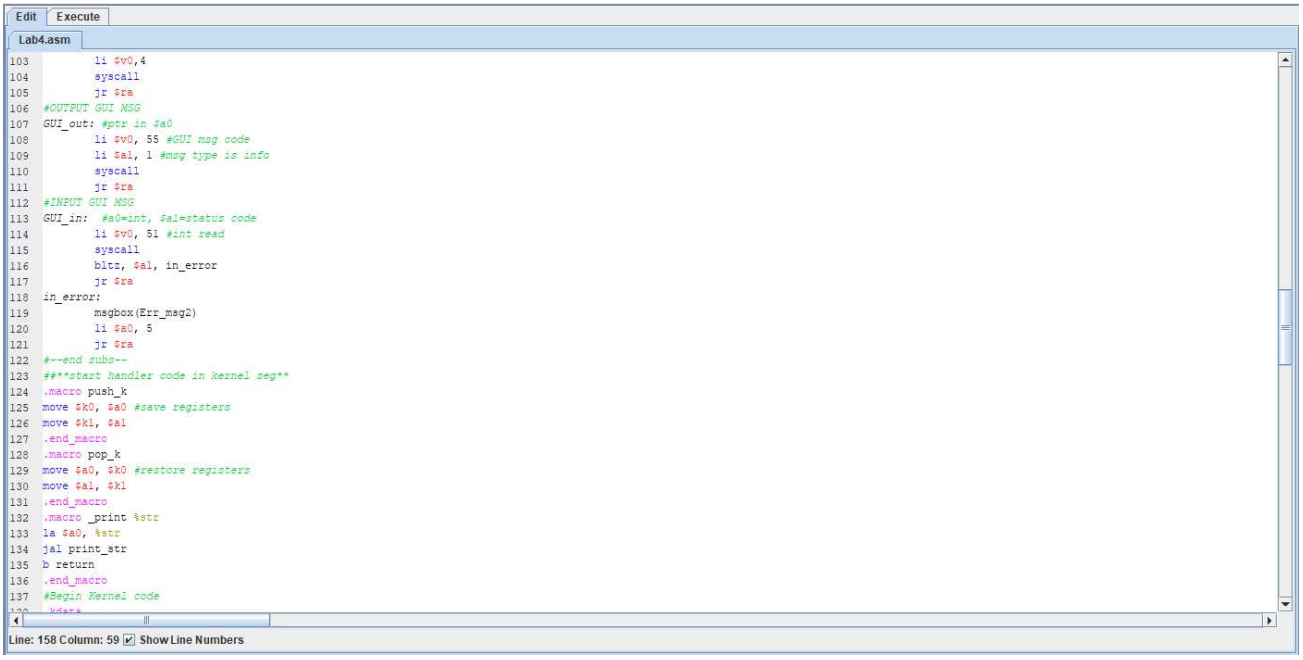
```
35 .macro done
36 li $v0, 10 #stop code
37 syscall #stop
38 .end_macro
39 .macro print_mac (%str)
40 la $a0, %str
41 li $v0, 4
42 syscall
43 .end_macro
44 .macro msgbox (%str)
45 la $a0, %str
46 li $v0, 55 #GUI msg code
47 li $a1, 1 #msg type is info
48 syscall
49 .end_macro
50
51 #**ISR macro-> Trap
52 .macro _ISR (%str)
53 la $a0, %str
54 #jal printStr
55 jal GUI_out
56 Teq $0,$0 #Trap: simulate INT<-1 (in ktext)
57 b loop_main
58 .end_macro
59 #code
60 .text
61 #saves:
62 #$t9
63 li $t9, stop
64 La $a0, header
65 Jal printStr
66 #main Loop
67 loop_main:
68 subiu $t9,$t9,1 #decrement counter
69 blez $t9, Stop
```

Line: 158 Column: 59 Show Line Numbers


Edit Execute
Lab4.asm

```
69  blez $t9, Stop
70  la $a0,prompt1
71  Jal GUI_in #get Type in $a0
72  move $s0, $a0
73  #Int TYPE Branch table (if-case)
74  beq $a0,0,NMI
75  beq $a0,1,NVI
76  beq $a0,2,VI
77  beq $a0,9,Halt
78  b Err #none of above
79  NMI: _ISR(NMI_str)
80
81  NVI: _ISR(NVI_str)
82  VI: #get vector
83          la $a0,prompt2
84          Jal GUI_in #get vector in $a0
85          move $s1, $a0 #save vector in $s1
86          _ISR(VI_str)
87  Halt:  #Quit
88          la $a0, Halt_msg
89          jal GUI_out
90          jal printStr
91          done #**exit program**
92  Err:  la $a0,Err_msg1 #default
93          jal GUI_out
94          b loop_main
95  Stop: la $a0, Stop_msg
96          jal printStr
97          done  #**alt exit-stopped out
98  #end table
99  #--END MAIN LOOP--
100 ##subroutines follow
101 #print $a0 on console
102 printStr:
103         li $v0,4
```

Line: 158 Column: 59 Show Line Numbers


Edit Execute
Lab4.asm

```
103         li $v0,4
104         syscall
105         jr $ra
106 #OUTPUT GUI MSG
107 GUI_out: #ptr in $a0
108         li $v0, 55 #GUI msg code
109         li $a1, 1 #msg type is info
110         syscall
111         jr $ra
112 #INPUT GUI MSG
113 GUI_in:  #a0=int, $a1=status code
114         li $v0, 51 #int read
115         syscall
116         bltz, $a1, in_error
117         jr $ra
118 in_error:
119         msgbox(Err_msg2)
120         li $a0, 5
121         jr $ra
122 #--end subs--
123 ##**start handler code in kernel seg**
124 .macro push_k
125 move $k0, $a0 #save registers
126 move $k1, $a1
127 .end_macro
128 .macro pop_k
129 move $a0, $k0 #restore registers
130 move $a1, $k1
131 .end_macro
132 .macro _print %str
133 la $a0, %str
134 jal print_str
135 b return
136 .end_macro
137 #Begin Kernel code
```

Line: 158 Column: 59 Show Line Numbers

Edit Execute

Lab4.asm

```
#Begin Kernel code
.kdata
kmsg: .asciiz "starting Interrupt handler for: "
def_msg: .asciiz "error: unimplemented vector\n"
.align 2
newLn: .asciiz "\n"
.align 2
end_Kdata: .asciiz "ENDkDATA$$$$"
ioDevice1: .asciiz " Keyboard"
ioDevice2: .asciiz " Mouse"
ioDevice3: .asciiz " Phone"
ioDevice4: .asciiz " Speaker"
.ktext exc_seg
#save state
push_k
li $v0, 4
syscall
print_mac kmsg #prt msg via macro
mfc0 $t0, $14 #EPC
addi $t0,$t0, 4 #incrementing RA in EPC
mtc0 $t0,$14 #EPC+4 (for ERET)
#saves: $s0=int code, $s1=vector  $s5=pointer to iodevices
#--INT Branch Table
Beq $s0, 0, NMI_Handler
Beq $s0, 1, NVI_Handler
#else VI
#--Branch Table--
Beq $s1, 0, v0
Beq $s1, 1, v1
Beq $s1, 2, v2
Beq $s1, 3, v3
#default
b def
#end Br table
#start ISR's
NMI_Handler:
```

Line: 158 Column: 59 Show Line Numbers

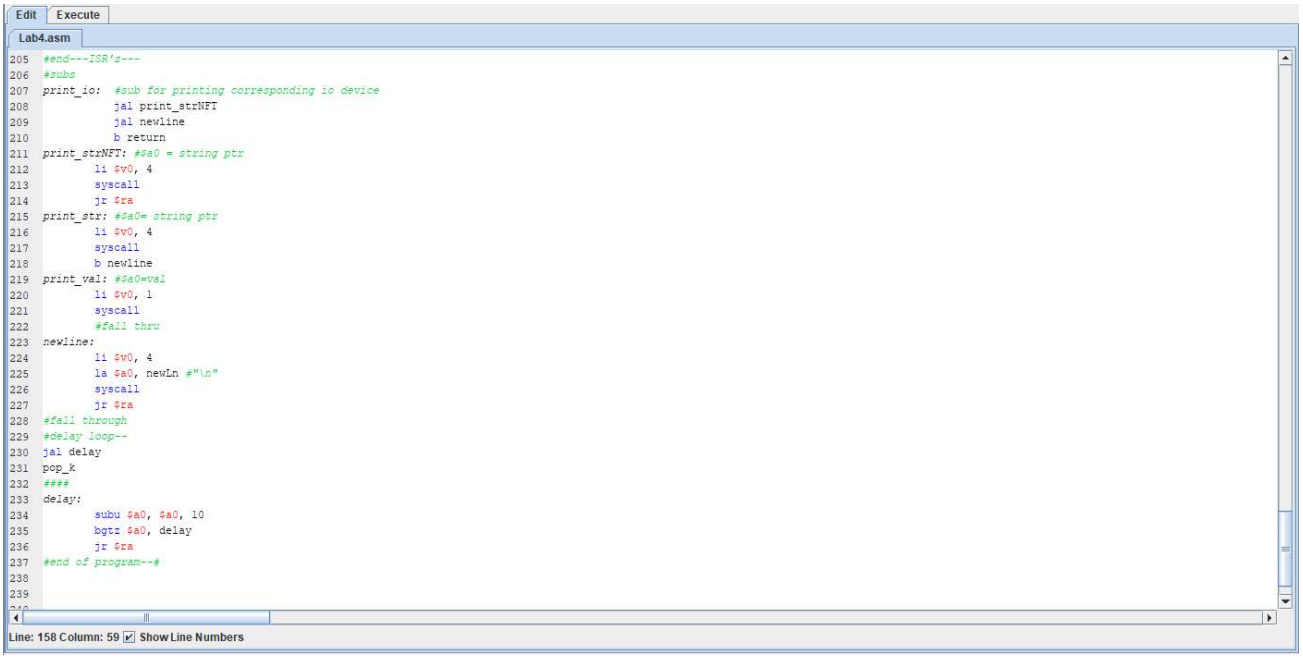
Edit Execute

Lab4.asm

```
#start ISR's
NMI_Handler:
_print NMI_str
NVI_Handler:
_print NVI_str
#--Vector Table--
v0:

        la $s5, ioDevice1
        b finish#ioDevice1
v1:
        la $s5, ioDevice2
        b finish#ioDevice2
v2:
        la $s5, ioDevice3
        b finish#ioDevice3
v3:
        la $s5, ioDevice4
        b finish#ioDevice4
def: #un-impl
        print_mac def_msg
        eret
finish:
        print_mac Vect_str
        move $a0, $s1 #set vector
        jal print_val
        la $a0,($s5) #set IO device
        jal print_io
b return
#end Tabledef: #un-impl
print_mac def_msg
return:
        pop_k
        eret
#end---ISR's---
```

Line: 158 Column: 59 Show Line Numbers



(END of Source Code)