**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] 001f8821  **addu $17, $0, $31**        *; 30: move $s1, $ra # store return address*   
[00400028] 34020004  **ori $2, $0, 4**            *; 32: li $v0, 4 # print string*   
[0040002c] 3c041001  **lui $4, 4097 [prompt]**    *; 33: la $a0, prompt*   
[00400030] 0000000c  **syscall**                  *; 34: syscall*   
[00400034] 34020005  **ori $2, $0, 5**            *; 36: li $v0, 5 # read integer from the user*   
[00400038] 0000000c  **syscall**                  *; 37: syscall*   
[0040003c] 0002c821  **addu $25, $0, $2**         *; 39: move $t9, $v0 # store n to t9*   
[00400040] 00192880  **sll $5, $25, 2**           *; 41: sll $a1, $t9, 2 # space = size \* 4*   
[00400044] 34080000  **ori $8, $0, 0**            *; 43: li $t0, 0 # counter*   
[00400048] 34100000  **ori $16, $0, 0**           *; 44: li $s0, 0 # number of integers initialize*   
[0040004c] 0119082a  **slt $1, $8, $25**          *; 47: bge $t0, $t9, sortStack # if t0 >= n, jump to sortStack*   
[00400050] 1020000a  **beq $1, $0, 40 [sortStack-0x00400050]**   
[00400054] 34020004  **ori $2, $0, 4**            *; 49: li $v0, 4 # print string*   
[00400058] 3c011001  **lui $1, 4097 [inputMsg]**  *; 50: la $a0, inputMsg*   
[0040005c] 34240035  **ori $4, $1, 53 [inputMsg]**   
[00400060] 0000000c  **syscall**                  *; 51: syscall*   
[00400064] 34020005  **ori $2, $0, 5**            *; 53: li $v0, 5 # read integer from the user*   
[00400068] 0000000c  **syscall**                  *; 54: syscall*   
[0040006c] 0c10003c  **jal 0x004000f0 [addIntegerToStack]**  
[00400070] 21080001  **addi $8, $8, 1**           *; 58: addi $t0, $t0, 1 # increase counter*   
[00400074] 08100013  **j 0x0040004c [readIntegerLoop]***; 60: j readIntegerLoop*   
[00400078] 1010002e  **beq $0, $16, 184 [end-0x00400078]**  
[0040007c] 001d4021  **addu $8, $0, $29**         *; 64: move $t0,$sp # outerloop counter*   
[00400080] 03b06020  **add $12, $29, $16**        *; 65: add $t4,$sp,$s0 # outerloop termination*   
[00400084] 218cfffc  **addi $12, $12, -4**        *; 66: addi $t4,$t4,-4*   
[00400088] 001d4821  **addu $9, $0, $29**         *; 69: move $t1,$sp # innerloop counter*   
[0040008c] 011d7022  **sub $14, $8, $29**         *; 70: sub $t6,$t0,$sp # inner loop end condition*   
[00400090] 018e7022  **sub $14, $12, $14**        *; 71: sub $t6,$t4,$t6*   
[00400094] 8d2a0000  **lw $10, 0($9)**            *; 74: lw $t2, 0($t1) # get first int*   
[00400098] 8d2b0004  **lw $11, 4($9)**            *; 75: lw $t3, 4($t1) # get second int*   
[0040009c] 014b682a  **slt $13, $10, $11**        *; 76: slt $t5, $t2, $t3 # if n[0]   
[004000a0] 15a00006****bne $13, $0, 24 [swapFalse-0x004000a0]*** *[004000a4] 000a6821****addu $13, $0, $10****; 78: move $t5, $t2 # move $t2 into $t5   
[004000a8] 000b5021****addu $10, $0, $11****; 79: move $t2, $t3 # move $t3 into $t2   
[004000ac] 000d5821****addu $11, $0, $13****; 80: move $t3, $t5 # move $t5 into $t3   
[004000b0] ad2a0000****sw $10, 0($9)****; 81: sw $t2, 0($t1) # restore back to stack swapped   
[004000b4] ad2b0004****sw $11, 4($9)****; 82: sw $t3, 4($t1)   
[004000b8] 21290004****addi $9, $9, 4****; 85: addi $t1,$t1,4 # increment to next int   
[004000bc] 152efff6****bne $9, $14, -40 [sortStackLoopCont-0x004000bc]*** *[004000c0] 21080004****addi $8, $8, 4****; 88: addi $t0,$t0,4 # increment outer loop counter   
[004000c4] 150cfff1****bne $8, $12, -60 [sortStackLoop-0x004000c4]*** *[004000c8] 34020004****ori $2, $0, 4****; 91: li $v0, 4   
[004000cc] 3c011001****lui $1, 4097 [newLine]****; 92: la $a0, newLine   
[004000d0] 342400bb****ori $4, $1, 187 [newLine]***  *[004000d4] 0000000c****syscall****; 93: syscall   
[004000d8] 34020004****ori $2, $0, 4****; 95: li $v0, 4   
[004000dc] 3c011001****lui $1, 4097 [stackMsg]****; 96: la $a0, stackMsg   
[004000e0] 34240046****ori $4, $1, 70 [stackMsg]***  *[004000e4] 0000000c****syscall****; 97: syscall   
[004000e8] 0c100040****jal 0x00400100 [printArray]****; 99: jal printArray   
[004000ec] 08100052****j 0x00400148 [binarySearch]****; 101: j binarySearch   
[004000f0] 23bdfffc****addi $29, $29, -4****; 104: addi $sp, $sp, -4 # move the pointer to create space   
[004000f4] afa20000****sw $2, 0($29)****; 105: sw $v0, 0($sp) # store the input to stack   
[004000f8] 22100004****addi $16, $16, 4****; 106: addi $s0, $s0, 4 # store stack counter in s0   
[004000fc] 03e00008****jr $31****; 107: jr $ra   
[00400100] 001d6821****addu $13, $0, $29****; 110: move $t5, $sp # print counter   
[00400104] 020d5020****add $10, $16, $13****; 111: add $t2,$s0,$t5 # end condition   
[00400108] 34020001****ori $2, $0, 1****; 114: li $v0, 1 # opcode print int   
[0040010c] 8da40000****lw $4, 0($13)****; 115: lw $a0,0($t5) # arg for print int   
[00400110] 0000000c****syscall****; 116: syscall   
[00400114] 34020004****ori $2, $0, 4****; 118: li $v0,4 # print string   
[00400118] 3c011001****lui $1, 4097 [newSpace]****; 119: la $a0,newSpace # space   
[0040011c] 342400bd****ori $4, $1, 189 [newSpace]***  *[00400120] 0000000c****syscall****; 120: syscall   
[00400124] 21ad0004****addi $13, $13, 4****; 122: addi $t5,$t5,4 # next int   
[00400128] 15aafff8****bne $13, $10, -32 [printArrayLoop-0x00400128]*** *[0040012c] 03e00008****jr $31****; 125: jr $ra # return to caller   
[00400130] 34020004****ori $2, $0, 4****; 129: li $v0, 4   
[00400134] 3c011001****lui $1, 4097 [endMsg]****; 130: la $a0, endMsg   
[00400138] 342400bf****ori $4, $1, 191 [endMsg]***  *[0040013c] 0000000c****syscall****; 131: syscall   
[00400140] 3402000a****ori $2, $0, 10****; 133: li $v0, 10 # Exit Program   
[00400144] 0000000c****syscall****; 134: syscall   
[00400148] 34020004****ori $2, $0, 4****; 141: li $v0, 4   
[0040014c] 3c011001****lui $1, 4097 [msg\_inputSearch]****; 142: la $a0, msg\_inputSearch # load msg of for the input   
[00400150] 3424005d****ori $4, $1, 93 [msg\_inputSearch]***  *[00400154] 0000000c****syscall****; 143: syscall   
[00400158] 34020005****ori $2, $0, 5****; 145: li $v0, 5 # read input   
[0040015c] 0000000c****syscall****; 146: syscall   
[00400160] 00024021****addu $8, $0, $2****; 148: move $t0, $v0 # assign input to $t0   
[00400164] 001d4821****addu $9, $0, $29****; 150: move $t1, $sp # assign the end address   
[00400168] 0019b821****addu $23, $0, $25****; 152: move $s7, $t9 # $s7 always contains max index   
[0040016c] 34160001****ori $22, $0, 1****; 153: li $s6, 1 # $s6 always contains min index   
[00400170] 34030002****ori $3, $0, 2****; 155: li $v1, 2   
[00400174] 0323001a****div $25, $3****; 156: div $t9, $v1 # divide the size by 2   
[00400178] 0000c012****mflo $24****; 158: mflo $t8 # $t8 always contains mid index   
[0040017c] 01254820****add $9, $9, $5****; 162: add $t1, $t1, $a1   
[00400180] 2129fffc****addi $9, $9, -4****; 163: addi $t1, $t1, -4   
[00400184] 8d2a0000****lw $10, 0($9)****; 164: lw $t2, 0($t1) # load the largest value to $t2   
[00400188] 0148082a****slt $1, $10, $8****; 166: bgt $t0, $t2, notFound # if the input is greater than largest number in stack,   
[0040018c] 1420002c****bne $1, $0, 176 [notFound-0x0040018c]***  *[00400190] 110a0026****beq $8, $10, 152 [found-0x00400190]*** *[00400194] 001d4821****addu $9, $0, $29****; 172: move $t1, $sp # restore the address   
[00400198] 8d2a0000****lw $10, 0($9)****; 174: lw $t2, 0($t1) # load the smallest value to $t2   
[0040019c] 110a0023****beq $8, $10, 140 [found-0x0040019c]*** *[004001a0] 010a082a****slt $1, $8, $10****; 178: blt $t0, $t2, notFound # if the input is less than smallest number in stack,   
[004001a4] 14200026****bne $1, $0, 152 [notFound-0x004001a4]***  *[004001a8] 02d7082a****slt $1, $22, $23****; 182: ble $s7, $s6, notFound # if max is less than equal to min, branch not found   
[004001ac] 10200024****beq $1, $0, 144 [notFound-0x004001ac]***  *[004001b0] 00187821****addu $15, $0, $24****; 184: move $t7, $t8 # take $t7 as counter   
[004001b4] 0c100072****jal 0x004001c8 [getTheMidInt]****; 186: jal getTheMidInt # the mid integer stored in $s1   
[004001b8] 1111001c****beq $8, $17, 112 [found-0x004001b8]*** *[004001bc] 0111502a****slt $10, $8, $17****; 191: slt $t2 $t0, $s1 # if input is less than the mid int, $t2=1   
[004001c0] 11400009****beq $10, $0, 36 [searchHigh-0x004001c0]*** *[004001c4] 0810007f****j 0x004001fc [searchLow]****; 193: j searchLow # else search low   
[004001c8] 8d310000****lw $17, 0($9)****; 198: lw $s1, 0($t1) # the midpoint number stores in $s1   
[004001cc] 11f60005****beq $15, $22, 20 [endGet-0x004001cc]*** *[004001d0] 21290004****addi $9, $9, 4****; 202: addi $t1, $t1, 4   
[004001d4] 11f60003****beq $15, $22, 12 [endGet-0x004001d4]*** *[004001d8] 21efffff****addi $15, $15, -1****; 205: addi $t7, $t7, -1   
[004001dc] 08100072****j 0x004001c8 [getTheMidInt]****; 207: j getTheMidInt   
[004001e0] 03e00008****jr $31****; 210: jr $ra   
[004001e4] 0018b021****addu $22, $0, $24****; 214: move $s6, $t8 # min becomes mid   
[004001e8] 02d7c020****add $24, $22, $23****; 215: add $t8, $s6, $s7 # add mid and max   
[004001ec] 34030002****ori $3, $0, 2****; 216: li $v1, 2   
[004001f0] 0303001a****div $24, $3****; 217: div $t8, $v1 # divide the sum of index by 2   
[004001f4] 0000c012****mflo $24****; 218: mflo $t8 # the higher mid index stores in $t8   
[004001f8] 0810006a****j 0x004001a8 [search]****; 220: j search   
[004001fc] 0018b821****addu $23, $0, $24****; 223: move $s7, $t8 # max becomes mid   
[00400200] 02d7c020****add $24, $22, $23****; 224: add $t8, $s6, $s7 # add mid and max   
[00400204] 34030002****ori $3, $0, 2****; 225: li $v1, 2   
[00400208] 0303001a****div $24, $3****; 226: div $t8, $v1 # divide the sum of index by 2   
[0040020c] 0000c012****mflo $24****; 227: mflo $t8 # the lower mid index stores in $t8   
[00400210] 34030004****ori $3, $0, 4****; 229: li $v1, 4   
[00400214] 02c30018****mult $22, $3****; 230: mult $s6, $v1   
[00400218] 00007012****mflo $14****; 231: mflo $t6   
[0040021c] 001d4821****addu $9, $0, $29****; 233: move $t1, $sp # assign the end address   
[00400220] 012e4820****add $9, $9, $14****; 234: add $t1, $t1, $t6 # shift to min address   
[00400224] 0810006a****j 0x004001a8 [search]****; 236: j search   
[00400228] 34020004****ori $2, $0, 4****; 240: li $v0, 4   
[0040022c] 3c011001****lui $1, 4097 [msg\_found]****; 241: la $a0, msg\_found   
[00400230] 34240082****ori $4, $1, 130 [msg\_found]***  *[00400234] 0000000c****syscall****; 242: syscall   
[00400238] 0810004c****j 0x00400130 [end]****; 244: j end   
[0040023c] 34020004****ori $2, $0, 4****; 247: li $v0, 4   
[00400240] 3c011001****lui $1, 4097 [msg\_notFound]****; 248: la $a0, msg\_notFound   
[00400244] 34240095****ori $4, $1, 149 [msg\_notFound]***  *[00400248] 0000000c****syscall****; 249: syscall   
[0040024c] 0810004c****j 0x00400130 [end]****; 251: j end*

***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   
[800001bc] 34020004****ori $2, $0, 4****; 110: li $v0 4 # syscall 4 (print\_str)   
[800001c0] 3344003c****andi $4, $26, 60****; 111: andi $a0 $k0 0x3c   
[800001c4] 3c019000****lui $1, -28672****; 112: lw $a0 \_\_excp($a0)   
[800001c8] 00240821****addu $1, $1, $4*** *[800001cc] 8c240180****lw $4, 384($1)*** *[800001d0] 00000000****nop****; 113: nop   
[800001d4] 0000000c****syscall****; 114: syscall   
[800001d8] 34010018****ori $1, $0, 24****; 116: bne $k0 0x18 ok\_pc # Bad PC exception requires special checks   
[800001dc] 143a0008****bne $1, $26, 32 [ok\_pc-0x800001dc]***  *[800001e0] 00000000****nop****; 117: nop   
[800001e4] 40047000****mfc0 $4, $14****; 119: mfc0 $a0 $14 # EPC   
[800001e8] 30840003****andi $4, $4, 3****; 120: andi $a0 $a0 0x3 # Is EPC word-aligned?   
[800001ec] 10040004****beq $0, $4, 16 [ok\_pc-0x800001ec]*** *[800001f0] 00000000****nop****; 122: nop   
[800001f4] 3402000a****ori $2, $0, 10****; 124: li $v0 10 # Exit on really bad PC   
[800001f8] 0000000c****syscall****; 125: syscall   
[800001fc] 34020004****ori $2, $0, 4****; 128: li $v0 4 # syscall 4 (print\_str)   
[80000200] 3c019000****lui $1, -28672 [\_\_m2\_]****; 129: la $a0 \_\_m2\_   
[80000204] 3424000d****ori $4, $1, 13 [\_\_m2\_]*** *[80000208] 0000000c****syscall****; 130: syscall   
[8000020c] 001a2082****srl $4, $26, 2****; 132: srl $a0 $k0 2 # Extract ExcCode Field   
[80000210] 3084001f****andi $4, $4, 31****; 133: andi $a0 $a0 0x1f   
[80000214] 14040002****bne $0, $4, 8 [ret-0x80000214]****; 134: bne $a0 0 ret # 0 means exception was an interrupt   
[80000218] 00000000****nop****; 135: nop   
[8000021c] 401a7000****mfc0 $26, $14****; 145: mfc0 $k0 $14 # Bump EPC register   
[80000220] 275a0004****addiu $26, $26, 4****; 146: addiu $k0 $k0 4 # Skip faulting instruction   
[80000224] 409a7000****mtc0 $26, $14****; 148: mtc0 $k0 $14   
[80000228] 3c019000****lui $1, -28672****; 153: lw $v0 s1 # Restore other registers   
[8000022c] 8c220200****lw $2, 512($1)*** *[80000230] 3c019000****lui $1, -28672****; 154: lw $a0 s2   
[80000234] 8c240204****lw $4, 516($1)*** *[80000238] 001b0821****addu $1, $0, $27****; 157: move $at $k1 # Restore $at   
[8000023c] 40806800****mtc0 $0, $13****; 160: mtc0 $0 $13 # Clear Cause register   
[80000240] 401a6000****mfc0 $26, $12****; 162: mfc0 $k0 $12 # Set Status register   
[80000244] 375a0001****ori $26, $26, 1****; 163: ori $k0 0x1 # Interrupts enabled   
[80000248] 409a6000****mtc0 $26, $12****; 164: mtc0 $k0 $12   
[8000024c] 42000018****eret****; 167: eret*