Kevin Martin
Syracuse University
CIS655 – Summer 2020, Tuesday @ 9:00pm EST
Homework 2

Question 1

To find the min and max from an array in MARS, I used a very simple double loop to sort the array. Then I took the first and last values from the array as the min and max values, respectively. I assumed a fix length of array that was known at the start of the program, which made the indexing much easier. Finally, I added a few lines related to printing the outputs in the I/O window in MARS. While the registers show the correct values, I enjoyed seeing the values printed in the terminal as well. Screenshots below. First, two of the code before it is assembled. Then the text/data segments after it is assembled but before it is run. Finally, the end result after it is successfully ran, including registers.

Edit | Execute

HW2_Q1_Kevin_Martin.asm

```
.data
#array: .word 1,3,5,7,9,11 #change array elements here
array: .word 0,4,900,5,3,2 #change array elements here
.text
main:
la $t0, array # address of array into t0
add $t0, $t0, 24 # length of array, 6 elements at 4 bytes each
loop1: # check if the entire array has been visited
add $t1, $0, $0 # set i = 0
la $a0, array # a0 has first element of array
loop2: # sorting
lw $t2, 0($a0) # t1 to current element, array[i]
lw $t3, 4($a0) # next element in array, move by 4 bytes
slt $t4, $t2, $t3 # check if t1 < t0
beq $t4, $0, next # if so, switch elements, if not go to next element
add $t1, $0, 1 # if so, check next element of array
sw $t2, 4($a0) # store the newly found higher number in next position
sw $t3, 0($a0) # newly found lower number in first position
next:
addi $a0, $a0, 4 # next element of array
bne $a0, $t0, loop2 # if all elemnts have not been examined, continue sort
bne $t1, $0, loop1 # check loop1 flag, continue

jal print #call print routine
li $v0, 10 #10 is syscall for exit
syscall

.data
buff:           .asciiz "buff\n"
max_text:       .asciiz "\nThe max value is:\n"
min_text:       .asciiz "\nThe min value is:\n"
.text
print:
li $v0, 4 #syscall 4 is print string
la $a0, buff # needed a buffer to print correctly
syscall
```

Line: 1 Column: 1 ☑ Show Line Numbers

Edit | Execute

HW2_Q1_Kevin_Martin.asm

```
li $v0, 10 #10 is syscall for exit
syscall

.data
buff:           .asciiz "buff\n"
max_text:       .asciiz "\nThe max value is:\n"
min_text:       .asciiz "\nThe min value is:\n"
.text
print:
li $v0, 4 #syscall 4 is print string
la $a0, buff # needed a buffer to print correctly
syscall

li $v0, 4 #syscall 4 for print string
la $a0, max_text # identify max value
syscall

li $v0, 1 # syscall for integer
la $a0, array # load sorted array into a0
lw $t1, 4($a0) # first element in array
add $a0, $t1, $zero # load element into a0 for print
syscall

li $v0, 4 # syscall 4 for print string
la $a0, min_text # identify min value
syscall

li $v0, 1 # syscall for integer
la $a0, array # load sorted array into a0
lw $t2, 24($a0) # last element in array
add $a0, $t2, $zero # load element into a0 for print
syscall

jr $ra #jump back to main part of function, ready to exit
```

Line: 1 Column: 1 ☑ Show Line Numbers

Edit | Execute

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x3c011001	lui $1,0x00001001	6: la $t0, array # address of array into t0
	0x00400004	0x34280000	ori $8,$1,0x00000000	
	0x00400008	0x21080018	addi $8,$8,0x00000018	7: add $t0, $t0, 24 # length of array, 6 elements at 4 bytes each
	0x0040000c	0x00004820	add $9,$0,$0	9: add $t1, $0, $0 # set i = 0
	0x00400010	0x3c011001	lui $1,0x00001001	10: la $a0, array # a0 has first element of array
	0x00400014	0x34240000	ori $4,$1,0x00000000	
	0x00400018	0x8c8a0000	lw $10,0x00000000($4)	12: lw $t2, 0($a0) # t1 to current element, array[i]
	0x0040001c	0x8c8b0004	lw $11,0x00000004($4)	13: lw $t3, 4($a0) # next element in array, move by 4 bytes
	0x00400020	0x014b602a	slt $12,$10,$11	14: slt $t4, $t2, $t3 # check if t1 < t0
	0x00400024	0x11800003	beq $12,$0,0x00000003	15: beq $t4, $0, next # if so, switch elements, if not go to next element
	0x00400028	0x20090001	addi $9,$0,0x00000001	16: add $t1, $0, 1 # if so, check next element of array
	0x0040002c	0xac8a0004	sw $10,0x00000004($4)	17: sw $t2, 4($a0) # store the newly found higher number in next position
	0x00400030	0xac8b0000	sw $11,0x00000000($4)	18: sw $t3, 0($a0) # newly found lower number in first position
	0x00400034	0x20840004	addi $4,$4,0x00000004	20: addi $a0, $a0, 4 # next element of array
	0x00400038	0x1488fff7	bne $4,$8,0xfffffff7	21: bne $a0, $t0, loop2 # if all elemnts have not been examined, continue sort
	0x0040003c	0x1520fff3	bne $9,$0,0xfffffff3	22: bne $t1, $0, loop1 # check loop1 flag, continue
	0x00400040	0x0c100013	jal 0x0040004c	24: jal print #call print routine

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000000	0x00000004	0x00000384	0x00000005	0x00000003	0x00000002	0x66667562	0x540a000a
0x10010020	0x6d206568	0x76207861	0x65756c61	0x3a736920	0x540a000a	0x6d206568	0x76206e69	0x65756c61
0x10010040	0x3a736920	0x0000000a	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010140	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010160	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010180	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values ☐ ASCII

Edit | Execute

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x3c011001	lui $1,0x00001001	6: la $t0, array # address of array into t0
	0x00400004	0x34280000	ori $8,$1,0x00000000	
	0x00400008	0x21080018	addi $8,$8,0x00000018	7: add $t0, $t0, 24 # length of array, 6 elements at 4 bytes each
	0x0040000c	0x00004820	add $9,$0,$0	9: add $t1, $0, $0 # set i = 0
	0x00400010	0x3c011001	lui $1,0x00001001	10: la $a0, array # a0 has first element of array
	0x00400014	0x34240000	ori $4,$1,0x00000000	
	0x00400018	0x8c8a0000	lw $10,0x00000000($4)	12: lw $t2, 0($a0) # t1 to current element, array[i]
	0x0040001c	0x8c8b0004	lw $11,0x00000004($4)	13: lw $t3, 4($a0) # next element in array, move by 4 bytes
	0x00400020	0x014b602a	slt $12,$10,$11	14: slt $t4, $t2, $t3 # check if t1 < t0
	0x00400024	0x11800003	beq $12,$0,0x00000003	15: beq $t4, $0, next # if so, switch elements, if not go to next element
	0x00400028	0x20090001	addi $9,$0,0x00000001	16: add $t1, $0, 1 # if so, check next element of array
	0x0040002c	0xac8a0004	sw $10,0x00000004($4)	17: sw $t2, 4($a0) # store the newly found higher number in next position
	0x00400030	0xac8b0000	sw $11,0x00000000($4)	18: sw $t3, 0($a0) # newly found lower number in first position
	0x00400034	0x20840004	addi $4,$4,0x00000004	20: addi $a0, $a0, 4 # next element of array
	0x00400038	0x1488fff7	bne $4,$8,0xfffffff7	21: bne $a0, $t0, loop2 # if all elemnts have not been examined, continue sort
	0x0040003c	0x1520fff3	bne $9,$0,0xfffffff3	22: bne $t1, $0, loop1 # check loop1 flag, continue
	0x00400040	0x0c100013	jal 0x0040004c	24: jal print #call print routine

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x66667562	0x00000384	0x00000005	0x00000004	0x00000003	0x00000002	0x00000000	0x540a000a
0x10010020	0x6d206568	0x76207861	0x65756c61	0x3a736920	0x540a000a	0x6d206568	0x76206e69	0x65756c61
0x10010040	0x3a736920	0x0000000a	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010140	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010160	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010180	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values ☐ ASCII

Mars Messages | Run I/O

```
The max value is:
900
The min value is:
0
-- program is finished running --
```

Clear

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0x00000000
$at	1	0x10010000
$v0	2	0x0000000a
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x10010018
$t1	9	0x00000384
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00400044
pc		0x0040004c
hi		0x00000000
lo		0x00000000