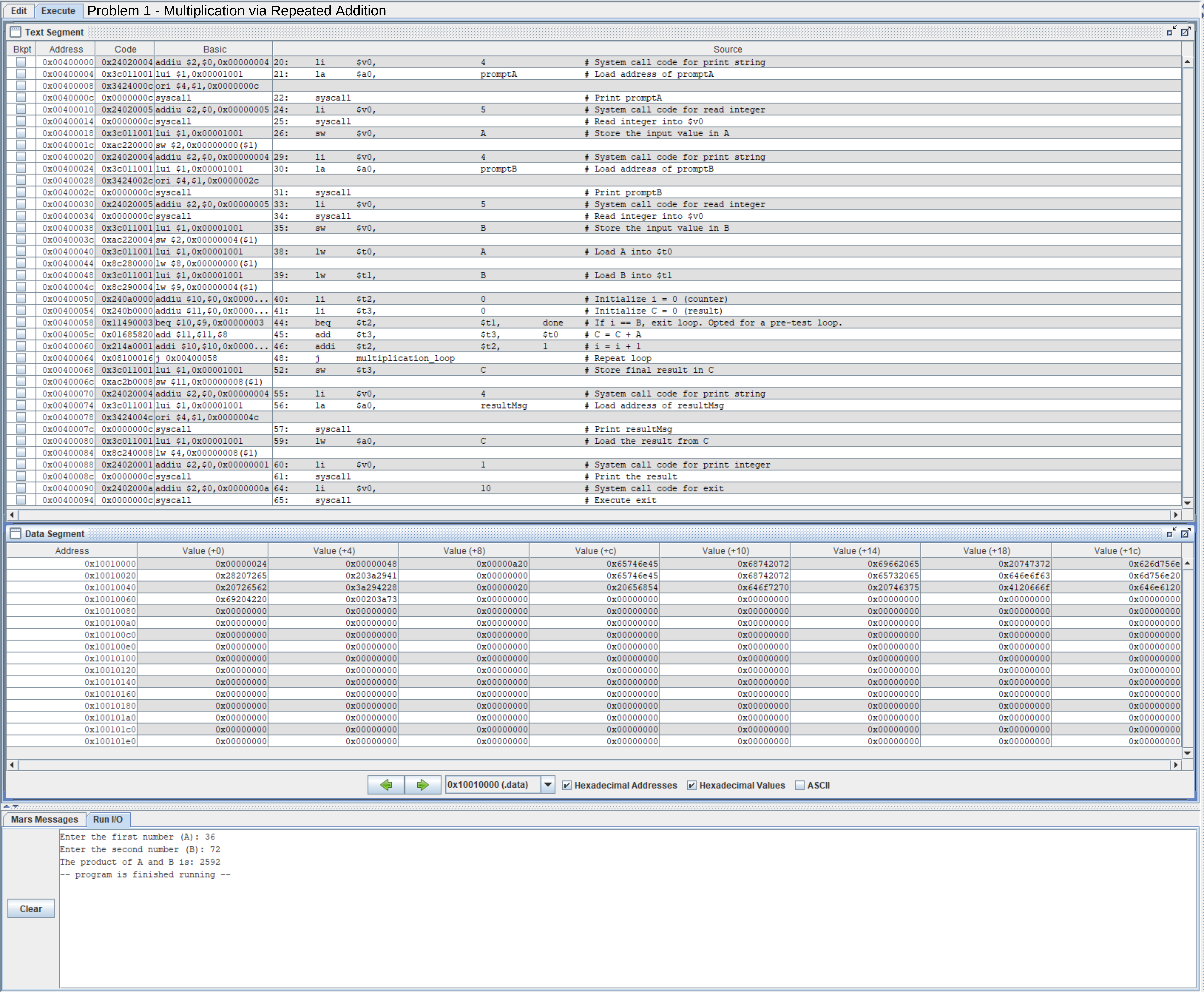
Edit Execute

Problem 1 - Multiplication via Repeated Addition

Text Segment

Bkpt	Address	Code	Basic	Source
☐	0x00400000	0x24020004	addiu $2,$0,0x00000004	`li $v0, 4 # System call code for print string`
☐	0x00400004	0x3c011001	lui $1,0x00001001	`la $a0, promptA # Load address of promptA`
☐	0x00400008	0x3424000c	ori $4,$1,0x0000000c	
☐	0x0040000c	0x0000000c	syscall	`syscall # Print promptA`
☐	0x00400010	0x24020005	addiu $2,$0,0x00000005	`li $v0, 5 # System call code for read integer`
☐	0x00400014	0x0000000c	syscall	`syscall # Read integer into $v0`
☐	0x00400018	0x3c011001	lui $1,0x00001001	`sw $v0, A # Store the input value in A`
☐	0x0040001c	0xac220000	sw $2,0x00000000($1)	
☐	0x00400020	0x24020004	addiu $2,$0,0x00000004	`li $v0, 4 # System call code for print string`
☐	0x00400024	0x3c011001	lui $1,0x00001001	`la $a0, promptB # Load address of promptB`
☐	0x00400028	0x3424002c	ori $4,$1,0x0000002c	
☐	0x0040002c	0x0000000c	syscall	`syscall # Print promptB`
☐	0x00400030	0x24020005	addiu $2,$0,0x00000005	`li $v0, 5 # System call code for read integer`
☐	0x00400034	0x0000000c	syscall	`syscall # Read integer into $v0`
☐	0x00400038	0x3c011001	lui $1,0x00001001	`sw $v0, B # Store the input value in B`
☐	0x0040003c	0xac220004	sw $2,0x00000004($1)	
☐	0x00400040	0x3c011001	lui $1,0x00001001	`lw $t0, A # Load A into $t0`
☐	0x00400044	0x8c280000	lw $8,0x00000000($1)	
☐	0x00400048	0x3c011001	lui $1,0x00001001	`lw $t1, B # Load B into $t1`
☐	0x0040004c	0x8c290004	lw $9,0x00000004($1)	
☐	0x00400050	0x240a0000	addiu $10,$0,0x0000...	`li $t2, 0 # Initialize i = 0 (counter)`
☐	0x00400054	0x240b0000	addiu $11,$0,0x0000...	`li $t3, 0 # Initialize C = 0 (result)`
☐	0x00400058	0x11490003	beq $10,$9,0x00000003	`beq $t2, $t1, done # If i == B, exit loop. Opted for a pre-test loop.`
☐	0x0040005c	0x01685820	add $11,$11,$8	`add $t3, $t3, $t0 # C = C + A`
☐	0x00400060	0x214a0001	addi $10,$10,0x0000...	`addi $t2, $t2, 1 # i = i + 1`
☐	0x00400064	0x08100016	j 0x00400058	`j multiplication_loop # Repeat loop`
☐	0x00400068	0x3c011001	lui $1,0x00001001	`sw $t3, C # Store final result in C`
☐	0x0040006c	0xac2b0008	sw $11,0x00000008($1)	
☐	0x00400070	0x24020004	addiu $2,$0,0x00000004	`li $v0, 4 # System call code for print string`
☐	0x00400074	0x3c011001	lui $1,0x00001001	`la $a0, resultMsg # Load address of resultMsg`
☐	0x00400078	0x3424004c	ori $4,$1,0x0000004c	
☐	0x0040007c	0x0000000c	syscall	`syscall # Print resultMsg`
☐	0x00400080	0x3c011001	lui $1,0x00001001	`lw $a0, C # Load the result from C`
☐	0x00400084	0x8c240008	lw $4,0x00000008($1)	
☐	0x00400088	0x24020001	addiu $2,$0,0x00000001	`li $v0, 1 # System call code for print integer`
☐	0x0040008c	0x0000000c	syscall	`syscall # Print the result`
☐	0x00400090	0x2402000a	addiu $2,$0,0x0000000a	`li $v0, 10 # System call code for exit`
☐	0x00400094	0x0000000c	syscall	`syscall # Execute exit`

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000024	0x00000048	0x00000a20	0x65746e45	0x68742072	0x69662065	0x20747372	0x626d756e
0x10010020	0x28207265	0x203a2941	0x00000000	0x65746e45	0x68742072	0x65732065	0x646e6f63	0x6d756e20
0x10010040	0x20726562	0x3a294228	0x00000020	0x20656854	0x646f7270	0x20746375	0x4120666f	0x646e6120
0x10010060	0x69204220	0x00203a73	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
0x100101c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

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

Mars Messages | Run I/O

```
Enter the first number (A): 36
Enter the second number (B): 72
The product of A and B is: 2592
-- program is finished running --
```

Clear

Edit | Execute

Problem 2 - Bubble Sort on Array of Size 1-20

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x24020004	addiu $2,$0,0x00000004	23: li $v0, 4 # Print string system call
	0x00400004	0x3c011001	lui $1,0x00001001	24: la $a0, prompt1
	0x00400008	0x34240054	ori $4,$1,0x00000054	
	0x0040000c	0x0000000c	syscall	25: syscall
	0x00400010	0x24020005	addiu $2,$0,0x00000005	27: li $v0, 5 # Read integer system call
	0x00400014	0x0000000c	syscall	28: syscall
	0x00400018	0x3c011001	lui $1,0x00001001	29: sw $v0, size # Store size
	0x0040001c	0xac220050	sw $2,0x00000050($1)	
	0x00400020	0x3c011001	lui $1,0x00001001	32: la $t0, array # Array base address
	0x00400024	0x34280000	ori $8,$1,0x00000000	
	0x00400028	0x3c011001	lui $1,0x00001001	33: lw $t1, size # Counter for loop
	0x0040002c	0x8c290050	lw $9,0x00000050($1)	
	0x00400030	0x240a0000	addiu $10,$0,0x0000...	34: li $t2, 0 # Index counter
	0x00400034	0x1149000b	beq $10,$9,0x0000000b	37: beq $t2, $t1, sort_array # If we've read all elements, start sorting. ...
	0x00400038	0x24020004	addiu $2,$0,0x00000004	40: li $v0, 4 # Print string system call
	0x0040003c	0x3c011001	lui $1,0x00001001	41: la $a0, prompt2
	0x00400040	0x3424007c	ori $4,$1,0x0000007c	
	0x00400044	0x0000000c	syscall	42: syscall
	0x00400048	0x24020005	addiu $2,$0,0x00000005	45: li $v0, 5 # Read integer system call
	0x0040004c	0x0000000c	syscall	46: syscall
	0x00400050	0x000a5880	sll $11,$10,0x00000002	49: sll $t3, $t2, 2 # Multiply index by 4 for word alignment
	0x00400054	0x010b5820	add $11,$8,$11	50: add $t3, $t0, $t3 # Calculate address
	0x00400058	0xad620000	sw $2,0x00000000($11)	51: sw $v0, ($t3) # Store value
	0x0040005c	0x214a0001	addi $10,$10,0x0000...	53: addi $t2, $t2, 1 # Increment counter
	0x00400060	0x0810000d	j 0x00400034	54: j input_loop
	0x00400064	0x3c011001	lui $1,0x00001001	58: la $a0, array # First argument - array address
	0x00400068	0x34240000	ori $4,$1,0x00000000	
	0x0040006c	0x3c011001	lui $1,0x00001001	59: lw $a1, size # Second argument - array size
	0x00400070	0x8c250050	lw $5,0x00000050($1)	
	0x00400074	0x0c100037	jal 0x004000dc	60: jal bubble_sort
	0x00400078	0x24020004	addiu $2,$0,0x00000004	63: li $v0, 4 # Print string system call
	0x0040007c	0x3c011001	lui $1,0x00001001	64: la $a0, output # Load address of output string
	0x00400080	0x3424008c	ori $4,$1,0x0000008c	
	0x00400084	0x0000000c	syscall	65: syscall
	0x00400088	0x3c011001	lui $1,0x00001001	68: la $t0, array # Array base address
	0x0040008c	0x34280000	ori $8,$1,0x00000000	
	0x00400090	0x3c011001	lui $1,0x00001001	69: lw $t1, size # Size
	0x00400094	0x8c290050	lw $9,0x00000050($1)	
	0x00400098	0x240a0000	addiu $10,$0,0x0000...	70: li $t2, 0 # Counter
	0x0040009c	0x1149000d	beq $10,$9,0x0000000d	73: beq $t2, $t1, exit # If we've printed all elements, exit. Pre-te...
	0x004000a0	0x000a5880	sll $11,$10,0x00000002	76: sll $t3, $t2, 2
	0x004000a4	0x010b5820	add $11,$8,$11	77: add $t3, $t0, $t3
	0x004000a8	0x8d640000	lw $4,0x00000000($11)	78: lw $a0, ($t3)
	0x004000ac	0x24020001	addiu $2,$0,0x00000001	79: li $v0, 1
	0x004000b0	0x0000000c	syscall	80: syscall
	0x004000b4	0x212cffff	addi $12,$9,0xffffffff	83: addi $t4, $t1, -1 # size - 1
	0x004000b8	0x114c0004	beq $10,$12,0x00000004	84: beq $t2, $t4, print_loop_end # Skip comma if last element
	0x004000bc	0x24020004	addiu $2,$0,0x00000004	85: li $v0, 4
	0x004000c0	0x3c011001	lui $1,0x00001001	86: la $a0, comma
	0x004000c4	0x342400bc	ori $4,$1,0x000000bc	
	0x004000c8	0x0000000c	syscall	87: syscall
	0x004000cc	0x214a0001	addi $10,$10,0x0000...	90: addi $t2, $t2, 1
	0x004000d0	0x08100027	j 0x0040009c	91: j print_loop
	0x004000d4	0x2402000a	addiu $2,$0,0x0000000a	94: li $v0, 10
	0x004000d8	0x0000000c	syscall	95: syscall
	0x004000dc	0x23bdfffc	addi $29,$29,0xffff...	103: addi $sp, $sp, -4
	0x004000e0	0xafbf0000	sw $31,0x00000000($29)	104: sw $ra, ($sp)
	0x004000e4	0x00044021	addu $8,$0,$4	106: move $t0, $a0 # Array address
	0x004000e8	0x20a9ffff	addi $9,$5,0xffffffff	107: addi $t1, $a1, -1 # Outer loop bound (size - 1)
	0x004000ec	0x240a0000	addiu $10,$0,0x0000...	108: li $t2, 0 # i = 0
	0x004000f0	0x1149000f	beq $10,$9,0x0000000f	111: beq $t2, $t1, bubble_sort_end
	0x004000f4	0x240b0000	addiu $11,$0,0x0000...	113: li $t3, 0 # j = 0
	0x004000f8	0x012a6022	sub $12,$9,$10	114: sub $t4, $t1, $t2 # size - 1 - i
	0x004000fc	0x116c000a	beq $11,$12,0x0000000a	117: beq $t3, $t4, outer_loop_end
	0x00400100	0x000b6880	sll $13,$11,0x00000002	120: sll $t5, $t3, 2
	0x00400104	0x010d6820	add $13,$8,$13	121: add $t5, $t0, $t5 # address of arr[j]
	0x00400108	0x8dae0000	lw $14,0x00000000($13)	122: lw $t6, ($t5) # value of arr[j]
	0x0040010c	0x8daf0004	lw $15,0x00000004($13)	123: lw $t7, 4($t5) # value of arr[j+1]
	0x00400110	0x01eec02a	slt $24,$15,$14	126: slt $t8, $t7, $t6 # $t8 = 1 if $t7 < $t6
	0x00400114	0x13000002	beq $24,$0,0x00000002	127: beq $t8, $zero, no_swap # branch if $t7 >= $t6
	0x00400118	0xadaf0000	sw $15,0x00000000($13)	130: sw $t7, ($t5)
	0x0040011c	0xadae0004	sw $14,0x00000004($13)	131: sw $t6, 4($t5)
	0x00400120	0x216b0001	addi $11,$11,0x0000...	134: addi $t3, $t3, 1 # j++
	0x00400124	0x0810003f	j 0x004000fc	135: j inner_loop
	0x00400128	0x214a0001	addi $10,$10,0x0000...	138: addi $t2, $t2, 1 # i++
	0x0040012c	0x0810003c	j 0x004000f0	139: j outer_loop
	0x00400130	0x8fbf0000	lw $31,0x00000000($29)	143: lw $ra, ($sp)
	0x00400134	0x23bd0004	addi $29,$29,0x0000...	144: addi $sp, $sp, 4
	0x00400138	0x03e00008	jr $31	145: jr $ra

Problem 1 - Multiplication via Repeated Addition

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0xffffff12	0xffffff37	0xfffffff3	0xfffffff3	0x00000000	0x00000002	0x00000005	0x00000008
0x10010020	0x00000012	0x00000041	0x0000009f	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x0000000b	0x65746e45	0x68742072	0x756e2065
0x10010060	0x7265626d	0x20666f20	0x6d656c65	0x73746e65	0x2d312820	0x3a293032	0x00000020	0x65746e45
0x10010080	0x6c652072	0x6e656d65	0x00203a74	0x20656854	0x6d656c65	0x73746e65	0x726f7320	0x20646574
0x100100a0	0x61206e69	0x6e656373	0x676e6964	0x64726f20	0x61207265	0x203a6572	0x00000000	0x0000202c
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
0x100101c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0x10010000 (.data) — [✓ Hexadecimal Addresses] [✓ Hexadecimal Values] ASCII

Mars Messages | Run I/O

```
Enter the number of elements (1-20): 11
Enter element: 65
Enter element: -13
Enter element: 18
Enter element: -238
Enter element: 159
Enter element: -201
Enter element: -13
Enter element: 5
Enter element: 2
Enter element: 8
Enter element: 0
The elements sorted in ascending order are: -238, -201, -13, -13, 0, 2, 5, 8, 18, 65, 159
-- program is finished running --
```

Clear