

## ASM1 Report

### **first part:**

variable used: delta \$s3

1st item->

address stored in register: 0x10010044

2nd item->

1st instruction for \$s3 register (lui): lui \$1, 0x00001001

2nd instruction (ori): ori \$19, \$1, 0x00000044

3rd item->

1st instruction 32-bit hex encoding: 0x3c011001

2nd instruction 32-bit hex encoding: 0x34330044

### **second part:**

string: textSort: first four: "Sort

hex codes:

S: 53

o: 6F

r: 72

t: 74

address of string in memory: 0x10010000

word in memory containing ascii codes: Value (+0) : 0x74726F53

Well, for one I learned to use breakpoints which is so helpful in accessing what happens throughout the code and debugging really. Also differentiating between addresses and hex codes.

questions that arose:

1. Why are the lines of code not in order in the execute section?

## 2. what are the “Value (+x)” in the Data Segment?

I had to use breakpoints a lot, I kept undoing and running one step at a time which, like the professor mentioned before, is very helpful.

I also had to keep an eye on the right bar which shows you the registers and how data shifts inside them.

I think they are stored in words of multiples of 4. Like I saw in the Data Segment.



Text Segment				
Bkpt	Address	Code	Basic	Source
<input type="checkbox"/>	0x004000d8	0x000c5820	add \$11,\$0,\$12	86: add \$t3, \$zero, \$t4
<input type="checkbox"/>	0x004000dc	0x000f6020	add \$12,\$0,\$15	87: add \$t4, \$zero, \$t7 # val4
<input type="checkbox"/>	0x004000e0	0x08100024	j 0x00400090	89: j LOOP
<input type="checkbox"/>	0x004000e4	0x0810003a	j 0x004000e8	92: j EXIT
<input type="checkbox"/>	0x004000e8	0x20020004	addi \$2,\$0,0x00000004	98: addi \$v0, \$zero, 4
<input checked="" type="checkbox"/>	0x004000ec	0x3c011001	lui \$1,0x00001001	99: la \$a0, textSort
<input type="checkbox"/>	0x004000f0	0x34240000	ori \$4,\$1,0x00000000	
<input type="checkbox"/>	0x004000f4	0x0000000c	syscall	100: syscall
<input type="checkbox"/>	0x004000f8	0x20020001	addi \$2,\$0,0x00000001	102: addi \$v0, \$zero, 1

Data Segment							
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Va
0x10010000	0x74726f53	0x73655220	0x73746c75	0x2000203a	0x4c000a00	0x0a535345	
0x10010020	0x0a45524f	0x72754300	0x6c615620	0x3a736575	0x000a0020	0x00000020	
0x10010040	0x00000000	0x0000000a	0x00000001	0x00000001	0x00000001	0x00000001	
0x10010060	0x56206573	0x61697261	0x20656c62	0x706d7544	0x0000203a	0x00000000	
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	

0x10010000 (.data)    Hexadecimal Addresses    Hexadecimal Values