

Lab 2 Report

COMP3350

Adam Biggs – 9/19/2024

The purpose of this lab was to translate functions from C to MIPS Assembly.

All code is original and based off of MIPS documentation

Task A:

Write a MIPS assembly program that prompts the user to enter a string less than 40 characters (no need to test with invalid inputs). The program should then print the entered string back to the console.

This task was very simple and can be broken down into variables, prompt, input, print. First I establish my variables for the message string and the user response (which I cap at 40 char). Then I print my prompt and listen for input. Following the input I print the input string and end. I had no issues writing this code and did it quickly.

#Lab 2a - Adam Biggs -----

```
.data
#variables
prompt: .asciiz "Enter a string: " #prompt message
userString: .space 40      #length of maximum response is 40

.text
.globl main
#main method -----
main:
    #print prompt
    li $v0, 4 #4 = print_string
    la $a0, prompt
    syscall

    #input detection
    li $v0, 8 #8 = read_string
    la $a0, userString
    li $a1, 40
    syscall

    #print input
    li $v0, 4 #4 = print_string
    la $a0, userString
    syscall

    #end
    li $v0, 10 #10 = exit
    Syscall
```

Before Run:

Name	Number	Value
\$zero	0	0
\$at	1	0
\$v0	2	0
\$v1	3	0
\$a0	4	0
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	0
\$t1	9	0
\$t2	10	0
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	0
\$s1	17	0
\$s2	18	0
\$s3	19	0
\$s4	20	0
\$s5	21	0
\$s6	22	0
\$s7	23	0
\$t8	24	0
\$t9	25	0
\$k0	26	0
\$k1	27	0
\$gp	28	268468224
\$sp	29	2147479548
\$fp	30	0
\$ra	31	0
pc		4194304
hi		0
lo		0

After Run:

Name	Number	Value
\$zero	0	0
\$at	1	268500992
\$v0	2	10
\$v1	3	0
\$a0	4	268501009
\$a1	5	40
\$a2	6	0
\$a3	7	0
\$t0	8	0
\$t1	9	0
\$t2	10	0
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	0
\$s1	17	0
\$s2	18	0
\$s3	19	0
\$s4	20	0
\$s5	21	0
\$s6	22	0
\$s7	23	0
\$t8	24	0
\$t9	25	0
\$k0	26	0
\$k1	27	0
\$gp	28	268468224
\$sp	29	2147479548
\$fp	30	0
\$ra	31	0
pc		4194364
hi		0
lo		0

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
268500992	e t n E	a r	i r t s	z g n	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
268501024	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
268501056	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
268501088	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
268501120	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
268501152	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
268501184	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
268501216	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
268501248	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
268501280	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
268501312	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0

0x10010000 (.data)

Hexadecimal AddressesHexadecimal ValuesASCII

Mars Messages

Run I/O

Reset: reset completed.

Clear

Reset: reset completed.

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
268500992	e t n E	a r	i r t s	z g n	s e t \0	d a t	\0 \n # a	\0 \0 \0 \0
268501024	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
268501056	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
268501088	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
268501120	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
268501152	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
268501184	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
268501216	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
268501248	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
268501280	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
268501312	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0

0x10010000 (.data)

Hexadecimal AddressesHexadecimal ValuesASCII

Mars Messages

Run I/O

Reset: reset completed.

Enter a string: test adam

test adam

Clear

-- program is finished running --

Task B:

Write a complete MIPS program that translates and calculates the equation in C shown below.

This task was also very simple. First I initialized my result Z to 0 (for use later). Then I set up each of my int vars. Then I perform the add and sub calculations. Finally I store the result in Z. This took me a few attempts, i kept mixing up the \$t temp registers.

#Lab 2b - Adam Biggs -----

```
.data
#initialize var
resultZ: .word 0 #Z

.text
.globl main

main:
#var setup
li $t0, 15 #A
li $t1, 10 #B
li $t2, 7 #C
li $t3, 2 #D
li $t4, 18 #E
li $t5, -3 #F

#basic operations
# Z = (A+B) + (C-D) + (E+F) - (A-C);
add $t6, $t0, $t1
sub $t7, $t2, $t3
add $t8, $t4, $t5
sub $t9, $t0, $t2

add $t6, $t6, $t7
add $t6, $t6, $t8
sub $t6, $t6, $t9

sw $t6, resultZ

li $v0, 10
syscall
```

Before:

The screenshot shows the Mars MIPS simulator interface. The 'Text Segment' window displays assembly instructions with their addresses, codes, basic instructions, and sources. The 'Data Segment' window shows memory addresses and values. The 'Registers' window on the right shows the state of registers, with \$t6 highlighted in red. The 'Mars Messages' window shows the execution progress.

Bkpt	Address	Code	Basic	Source
4194304	0x2408000f	addiu \$8,\$0,15	12: li \$t0, 15 #A	
4194308	0x2409000a	addiu \$9,\$0,10	13: li \$t1, 10 #B	
4194312	0x240a0007	addiu \$10,\$0,7	14: li \$t2, 7 #C	
4194316	0x240b0002	addiu \$11,\$0,2	15: li \$t3, 2 #D	
4194320	0x240c0012	addiu \$12,\$0,18	16: li \$t4, 18 #E	
4194324	0x240dffff	addiu \$13,\$0,-3	17: li \$t5, -3 #F	
4194328	0x01097020	add \$14,\$8,\$9	21: add \$t6, \$t0, \$t1	
4194332	0x014b7822	sub \$15,\$10,\$11	22: sub \$t7, \$t2, \$t3	
4194336	0x018dc020	add \$24,\$12,\$13	23: add \$t8, \$t4, \$t5	
4194340	0x010ac822	sub \$25,\$8,\$10	24: sub \$t9, \$t0, \$t2	
4194344	0x01cf7020	add \$14,\$14,\$15	26: add \$t6, \$t6, \$t7	
4194348	0x01d87020	add \$14,\$14,\$24	27: add \$t6, \$t6, \$t8	
4194352	0x01d97022	sub \$14,\$14,\$25	28: sub \$t6, \$t6, \$t9	

Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
268500992	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
268501024	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
268501056	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
268501088	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
268501120	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
268501152	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
268501184	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
268501216	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
268501248	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
268501280	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
268501312	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00

Registers	Coproc 1	Coproc 0
Name	Number	Value
\$zero	0	0
\$at	1	0
\$v0	2	0
\$v1	3	0
\$a0	4	0
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	0
\$t1	9	0
\$t2	10	0
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	0
\$s1	17	0
\$s2	18	0
\$s3	19	0
\$s4	20	0
\$s5	21	0
\$s6	22	0
\$s7	23	0
\$t8	24	0
\$t9	25	0
\$k0	26	0
\$k1	27	0
\$gp	28	268468224
\$sp	29	2147479548
\$fp	30	0
\$ra	31	4194304
\$pc		4194304
\$lo		0

Mars Messages: Go: execution completed successfully.

Assembly: assembling C:\Users\jeffb\Documents\GitHub\AssemblyMIPS\Lab-2\Lab2b

Assembly: operation completed successfully.

After:

The screenshot shows the Mars MIPS simulator interface after execution. The 'Text Segment' window displays assembly instructions. The 'Data Segment' window shows memory addresses and values. The 'Registers' window on the right shows the state of registers, with \$t6 highlighted in red. The 'Mars Messages' window shows the execution progress.

Bkpt	Address	Code	Basic	Source
4194304	0x2408000f	addiu \$8,\$0,15	12: li \$t0, 15 #A	
4194308	0x2409000a	addiu \$9,\$0,10	13: li \$t1, 10 #B	
4194312	0x240a0007	addiu \$10,\$0,7	14: li \$t2, 7 #C	
4194316	0x240b0002	addiu \$11,\$0,2	15: li \$t3, 2 #D	
4194320	0x240c0012	addiu \$12,\$0,18	16: li \$t4, 18 #E	
4194324	0x240dffff	addiu \$13,\$0,-3	17: li \$t5, -3 #F	
4194328	0x01097020	add \$14,\$8,\$9	21: add \$t6, \$t0, \$t1	
4194332	0x014b7822	sub \$15,\$10,\$11	22: sub \$t7, \$t2, \$t3	
4194336	0x018dc020	add \$24,\$12,\$13	23: add \$t8, \$t4, \$t5	
4194340	0x010ac822	sub \$25,\$8,\$10	24: sub \$t9, \$t0, \$t2	
4194344	0x01cf7020	add \$14,\$14,\$15	26: add \$t6, \$t6, \$t7	
4194348	0x01d87020	add \$14,\$14,\$24	27: add \$t6, \$t6, \$t8	
4194352	0x01d97022	sub \$14,\$14,\$25	28: sub \$t6, \$t6, \$t9	

Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
268500992	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
268501024	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
268501056	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
268501088	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
268501120	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
268501152	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
268501184	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
268501216	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
268501248	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
268501280	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
268501312	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00

Registers	Coproc 1	Coproc 0
Name	Number	Value
\$zero	0	0
\$at	1	268500992
\$v0	2	10
\$v1	3	0
\$a0	4	0
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	15
\$t1	9	10
\$t2	10	7
\$t3	11	2
\$t4	12	18
\$t5	13	0
\$t6	14	37
\$t7	15	5
\$s0	16	0
\$s1	17	0
\$s2	18	0
\$s3	19	0
\$s4	20	0
\$s5	21	0
\$s6	22	0
\$s7	23	0
\$t8	24	15
\$t9	25	0
\$k0	26	0
\$k1	27	0
\$gp	28	268468224
\$sp	29	2147479548
\$fp	30	0
\$ra	31	0
\$pc		4194304
\$lo		0

Mars Messages: test.asm

-- program is finished running --

-- program is finished running --

Task C:

Write a complete MIPS program that implements the same algorithm (in C) shown below.

This task was a lot more time consuming. First I once again init the result var resultZ. Then I set up my 3 main vars. Then i start comparing the vars (If $a > b$ || $c < 5$) and cycling through the cases. For this one I had to reference documentation on how to do case switches. Turns out to be way easier than I initially thought, it's just very tedious in assembly. The biggest thing I learned from these 3 labs was how important syscall works and how setting \$v0 effects the type of call.

#Lab 2C - Adam Biggs -----

```
.data
#initialize var
resultZ: .word 0

.text
.globl main

main:
    li $t0, 10 #a
    li $t1, 15 #b
    li $t2, 6 #c

    bgt $t0, $t1, checkC
    li $t3, 5
    blt $t2, $t3, case1

#parameter
checkC:
    addi $t2, $t2, 1
    li $t3, 7
    beq $t2, $t3, case2
    li $t4, 3
    b switch

#case to check against 1
case1:
    li $t4, 1
    b switch

#case to check against 2
case2:
    li $t4, 2

switch:
    beq $t4, 1, set1
    beq $t4, 2, set2
    li $t4, 0
    b store

set1:
    li $t4, -1
```

b store

set2:

li \$t4, -2

store: #store final result

sw \$t4, resultZ

li \$v0, 10

Syscall

Before:

The screenshot shows the Mars MIPS simulator interface. The Text Segment displays assembly code for setting up registers and performing a store. The Data Segment shows memory addresses and their values. The Registers window shows the current state of registers, with \$v0 at 10 and \$t4 at -2.

Registers	Coproc 1	Coproc 0
\$zero	0	0
\$at	1	0
\$v0	2	10
\$v1	3	0
\$a0	4	0
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	0
\$t1	9	0
\$t2	10	0
\$t3	11	0
\$t4	12	-2
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	0
\$s1	17	0
\$s2	18	0
\$s3	19	0
\$s4	20	0
\$s5	21	0
\$s6	22	0
\$s7	23	0
\$s8	24	0
\$s9	25	0
\$k0	26	0
\$k1	27	0
\$gp	28	268468224
\$fp	29	2147479548
\$ra	30	0
\$pc	31	4194304

After:

The screenshot shows the Mars MIPS simulator interface after execution. The Text Segment displays the same assembly code. The Data Segment shows the same memory addresses. The Registers window shows the state after execution, with \$v0 at 10 and \$t4 at -2.

Registers	Coproc 1	Coproc 0
\$zero	0	0
\$at	1	0
\$v0	2	10
\$v1	3	0
\$a0	4	0
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	0
\$t1	9	0
\$t2	10	0
\$t3	11	0
\$t4	12	-2
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	0
\$s1	17	0
\$s2	18	0
\$s3	19	0
\$s4	20	0
\$s5	21	0
\$s6	22	0
\$s7	23	0
\$s8	24	0
\$s9	25	0
\$k0	26	0
\$k1	27	0
\$gp	28	268468224
\$fp	29	2147479548
\$ra	30	0
\$pc	31	4194420