

Project 1

Program 1:

1. The user will input a string giving their name followed by the enter key and the program will output "Welcome, name"
2. Print out the greeting string, and then store the user input into the \$a0 register, then output that, followed by the end program string
- 3.

Registers	Purpose & Labels
\$v0	Call a print string
\$a0	Argument of syscall to print string
\$v0	Call a user input as int
\$a0	Argument of syscall to print int

4. Use the .data segments, Load registers, Get user input, Print Strings, Print ints

The screenshot displays the Mars IDE interface. The **Text Segment** window shows the following assembly code:

Bkpt	Address	Code	Basic	Source
	0x00400000	0x3c011001	lui \$1,0x000001001	7: la \$a0, greeting
	0x00400004	0x34240000	ori \$4,\$1,0x00000000	
	0x00400008	0x24020004	addiu \$2,\$0,0x00000000	8: li \$v0, 4 # Print out greeting
	0x0040000c	0x0000000c	syscall	9: syscall
	0x00400010	0x24020008	addiu \$2,\$0,0x00000000	11: li \$v0, 8 # take in input
	0x00400014	0x3c011001	lui \$1,0x000001001	13: la \$a0, newLine
	0x00400018	0x34240030	ori \$4,\$1,0x00000030	
	0x0040001c	0x24050014	addiu \$5,\$0,0x00000000	14: li \$a1, 20
	0x00400020	0x00044021	addu \$8,\$0,\$4	16: move \$t0, \$a0

The **Data Segment** window shows memory addresses and values for the .data segment. The **Registers** window on the right shows the state of the registers.

5.

Program 2:

1. The user will enter 4 positive integers separated by the enter key, and the program will output the function $f = (a+b) - (c+d) + (b+3)$
2. Store the user inputs into the registers \$t0, \$t1, \$t2, \$t3, and perform the operations on each register and store final value in \$t0, then output the string
- 3.

Registers	Column2
\$v0	Print string
\$a0	Set the output as the greeting message
\$v0	Print int
\$t1	Store the first int
\$t2	Store the second int
\$t3	Store the third int
\$t4	Store the fourth int
\$t0	Store the first operation
\$t1	Store the second operation
\$t2	Store the third operation
\$t0	Store the fourth and fifth operations and output

4. Load Registers, Perform subtraction on two registers, Perform addition on two registers, Move data from one register into another, Printing integers

The screenshot displays the Mars MIPS simulator interface. The 'Text Segment' window shows assembly code with addresses, hex codes, and comments. The 'Data Segment' window shows memory addresses and their corresponding values in various formats. The 'Registers' window on the right lists all MIPS registers and their current values.

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x00000000
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000
\$t1	9	0x00000000
\$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	0x7ffffc
\$fp	30	0x00000000
\$ra	31	0x00000000
\$pc		0x00400000

5.

Program 3:

1. Make a for loop that performs an operation on 3 numbers and outputs the result each time the loop runs
2. Store the value for i in \$s0, store the value for j which is 3 in \$s1, store the value for k in \$s2. In the loop, add i and j, and then subtract k to get f. Print out "f=" then print f and a newline character.
- 3.

Registers	Purpose & Labels
\$v0	Print string, and later int
\$a0	Argument of syscall to print string and later int
\$s0	Store the i value for the Loop
\$s1	Store 3 for the value of j
\$s2	Store 5 for the value of k
\$s3	Store i+j, then store that minus k

4. Load registers, Create a loop, Exit loop, Perform math on registers, Jump Loops, Print out strings and ints

The screenshot shows the Mars MIPS simulator interface. The 'Text Segment' window displays assembly code with comments. The 'Registers' window on the right shows the state of the MIPS registers.

Text Segment:

Bkpt	Address	Code	Basic	Source
	0x00400000	0x24020004	addiu \$2,\$0,0x00000...	8: li \$v0, 4
	0x00400004	0x3c011001	lui \$1,0x00001001	9: la \$a0, startMessage
	0x00400008	0x34240000	ori \$4,\$1,0x00000000	
	0x0040000c	0x0000000c	syscall	10: syscall
	0x00400010	0x20100000	addi \$16,\$0,0x00000...	12: addi \$s0, \$zero, 0
	0x00400014	0x24110003	addiu \$17,\$0,0x0000...	13: li \$s1, 3
	0x00400018	0x24120005	addiu \$18,\$0,0x0000...	14: li \$s2, 5
	0x0040001c	0x20010004	addi \$1,\$0,0x00000004	15: Loop: bgt \$s0, 4, exit
	0x00400020	0x0030082a	sllt \$1,\$1,\$16	

Registers:

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x00000000
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000
\$t1	9	0x00000000
\$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	0x7ffffcfc
\$fp	30	0x00000000
\$ra	31	0x00000000

- 5.

Program 4:

1. Make a loop that stores the index value + 2 in the array
2. Store the array index in \$s0, loop through the array, loop through the array counting down from 10 by 2 and add 2 to the index and store it back in the array at that index
- 3.

Registers	Purpose & Labels
\$v0	Print string, and later int
\$a0	Argument of syscall to print string and later int
\$s0	Store the array index at 0
\$s1	Store the value of the array at each index
\$t1	Increase the index value by 2
\$t2	Decrease the index by 2

4. Load Registers, Create array, Store array index, Move thru array, Store Values in array

5.

The screenshot displays the Mars Messages IDE interface. The top bar has 'Edit' and 'Execute' tabs. The main window is divided into two panes. The left pane, titled 'Text Segment', shows assembly code with columns for 'Bkpt', 'Address', 'Code', 'Basic', and 'Source'. The right pane, titled 'Data Segment', shows a memory dump with columns for 'Address' and various 'Value' offsets (+0, +4, +8, +c, +10, +14, +18, +1c). Below the panes are navigation arrows, a dropdown menu set to '0x10010000 (.data)', and checkboxes for 'Hexadecimal Addresses', 'Hexadecimal Values', and 'ASCII'. On the far right, a 'Registers' panel lists registers \$zero through \$ra, their numbers, and their current values.

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x00000000
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000
\$t1	9	0x00000000
\$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	0x7fffffc
\$fp	30	0x00000000
\$ra	31	0x00000000