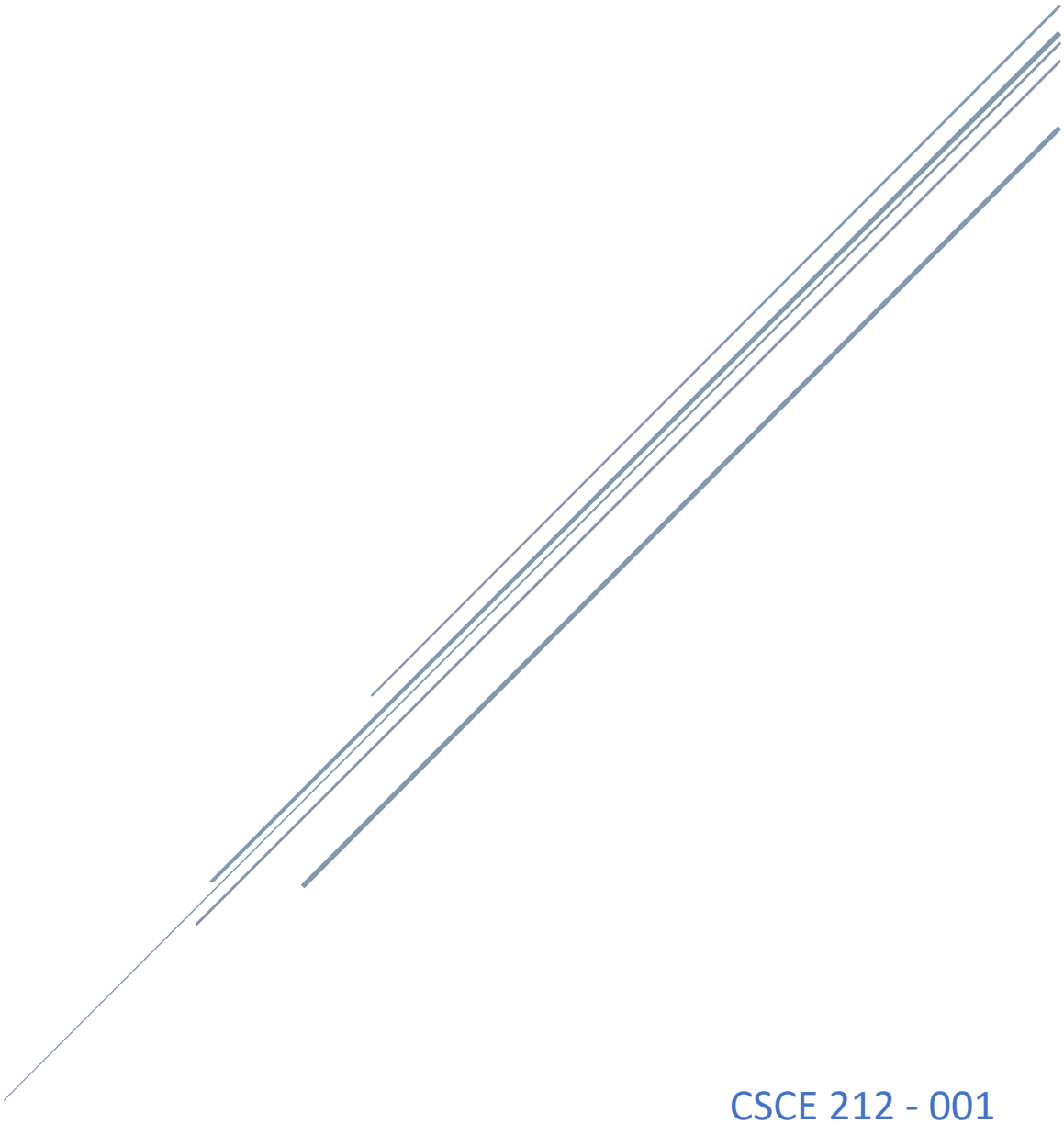
PROJECT 1 REPORT

Vu Nguyen



CSCE 212 - 001
Spring 2023

Table of Contents

1. PROGRAM INPUT/OUTPUT 2

1.1. PROGRAM 1: 2
1.2 PROGRAM 2: 2
1.3 PROGRAM 3: 2
1.4 PROGRAM 4: 2

2. PROGRAM DESIGN 2

2.1. PROGRAM 1: 2
2.2. PROGRAM 2: 2
2.3. PROGRAM 3: 3
2.4. PROGRAM 4: 3

3. SYMBOL TABLE 3

3.1. PROGRAM 1: 3
3.2. PROGRAM 2: 3
3.3. PROGRAM 3: 3
3.4. PROGRAM 4: 3

4. LEARNING COVERAGE 4

5. TEST RESULTS 4

5.1. PROGRAM 1: 4
5.2. PROGRAM 2: 5
5.3. PROGRAM 3: 5
5.4. PROGRAM 4: 6

Date: January 17, 2023
To: Dr. Rasha Karakchi
From: Viet Hoang Vu Nguyen
Subject: Project 1 Report
Class: CSCE 212

1. Program Input/Output

1.1. Program 1:

Output 1: “Hello, may I have your name, please?”
Input: “Vu Nguyen”
Output: “Welcome, Vu Nguyen”

1.2 Program 2:

Output 1: “Enter a number for a: ”
Input 1: 2
Output 2: “Enter a number for b: ”
Input 2: 3
Output 3: “Enter a number for c: ”
Input 3: 5
Output 4: “Enter a number for d: ”
Input 4: 2
Output 5: “Result (F) = 4”

1.3 Program 3:

Output 1: “Program starts”
Output 2: f = -2
Output 3: f = -1
Output 4: f = 0
Output 5: f = 1
Output 6: f = 2
Output 7: “Program ends”

1.4 Program 4:

Output 1: “Loop starts”
Output 2: “Loop ends”

2. Program Design

2.1. Program 1:

This program has two string constants, *prompt1* and *prompt2* which assigned with preset strings “Hello, may I have your name, please?\n” and “Welcome, ” accordingly. It also has *name* string variable to get user input string. The program will display *prompt1* to ask user for their names and return to new line with ‘\n’. The user then input their names which is store into *name*. Finally, the program will display *prompt2* and the input name to the terminal and exit.

2.2. Program 2:

This program has five string constants *promptInputA, promptInputB, promptInputC, promptInputD* to prompt for user number input as a, b, c, d accordingly, and *display* with the content of “Result (F) = ” to display the final result of arithmetic process at the end. The program would let user input 4 numbers a, b, c, d into the terminal, and store them in t0, t1, t2, t3 accordingly. The program would perform the math of F = (a+b) – (c+d) + (b+3). With the arithmetic process, the program would add t0 and t1, store in s0, then t2 and t3 store in s1,

then add immediate number 3 to t1, store in t1, then subtract s0 by s1, store in s0, and add s0 and t1, store in s0. Finally, the program prints *display* followed by the value of s0 and exit.

2.3. Program 3:

This program has four string constants, *promptStart* "Program starts\n", *promptF* "f = ", *newLine* "\n", and *promptEnd* "Program ends\n". The program assigns $s0 with 0, $s1 with 3, and $s2 with 5 in the beginning as i, j, k accordingly. Initially, it displays *promtStart.* It performs the math of f = i + j – k and print f to the terminal on each time it loops following the structure of *promptF* <f's value> *newLine*. The loop stops when the value of i is equal to 5. Before exiting the program, it would show *promptEnd* to notify its exit.

2.4. Program 4:

This program has two string constants, *promptStart* "Loop starts\n", *promptEnd* "Loop ends\n", and an array *myArray* with its buffer size is 40 bytes although there are only 5 elements used total. The program assigns 10 to $s0 and 40 to $t0 as loop index i, and array index t0 accordingly and displays *promptStart*. It then enters a loop until the loop index is equal to 0. Each time it loops through, it does the math of adding i and 2 and stores in $s1, then assigns $s1 to an address of *myArray*. Loop index decreases by 2 each time, and array index decreases by 8 each time. The program prompts ending loop message *promptEnd* before exiting. The reason *myArray* has buffer size of 40 bytes is because of array index t0 has to match with the loop index which start at 10. Although it does not utilize all of the elements, its maximum index is 10. Array index t0 decreases by 8 in order to match with the decrement of 2 of the loop index, so it skips one address in array each time.

3. Symbol Table

3.1. Program 1:

Registers	Purpose & Labels
$v0	System call service of results
$a0	Argument of syscall to store and print string
$a1	Argument of string buffer length

3.2. Program 2:

Registers	Purpose & Labels
$v0	System call service of results
$a0	Argument of syscall to print string
$t0 - $t3	Integer variable for the 4 numbers input
$s0	Integer variable for result output
$s1	Temporary integer variable to store the arithmetic function's result

3.3. Program 3:

Registers	Purpose & Labels
$v0	System call service of results
$a0	Argument of syscall to print string
$s0	Integer variable of loop index - i
$s1	Integer variable resembles j
$s2	Integer variable resembles k
$t0	Integer variable of result - f

3.4. Program 4:

Registers	Purpose & Labels

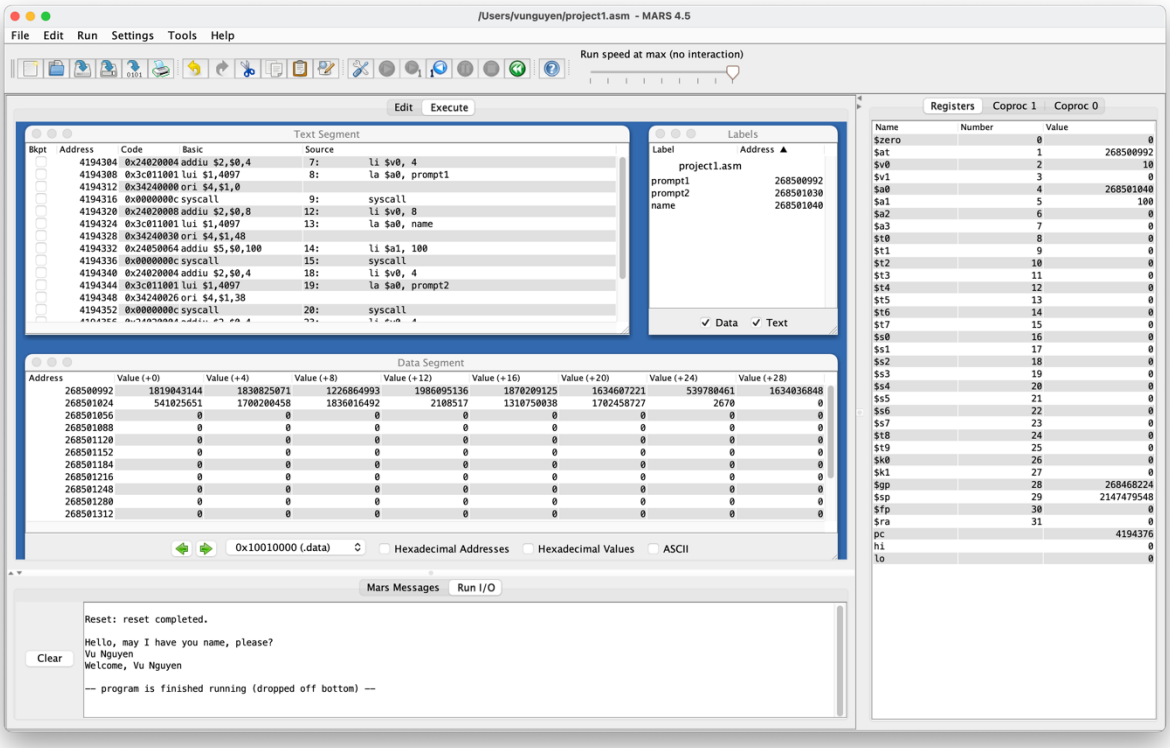
$v0	System call service of results
$a0	Argument of syscall to print string
$s0	Integer variable of loop index - i
$s1	Temporary integer variable to store the arithmetic function's result
$t0	Integer variable of array index

4. Learning Coverage

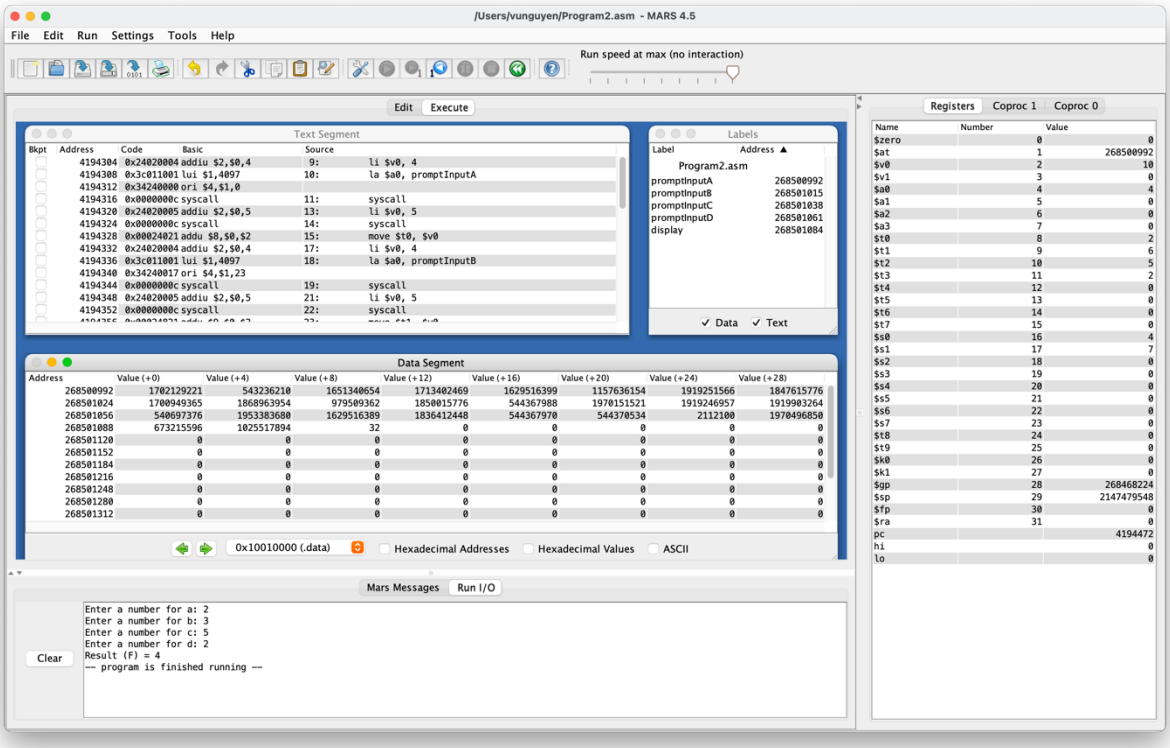
1. Loading a string from data and display it to the terminal
2. Getting user input from the terminal and store it into a variable for later uses
3. Declaring string and array and assign value or buffer size to them
4. Getting user input from the terminal for integer and store/move it in a variable
5. Displaying integer to the terminal
6. Adding, subtracting and overloading variables
7. Looping and conditioning the loop
8. Storing values to array's addresses
9. Changing array's addresses
10. Exiting the program

5. Test Results

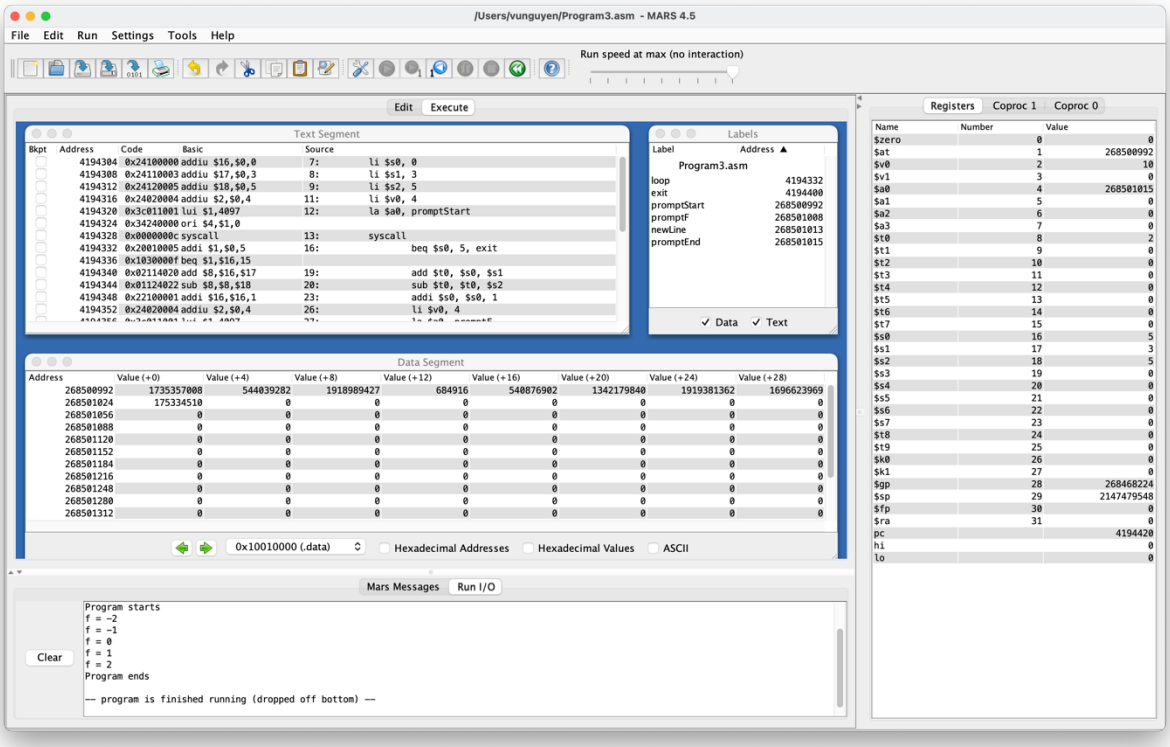
5.1. Program 1:



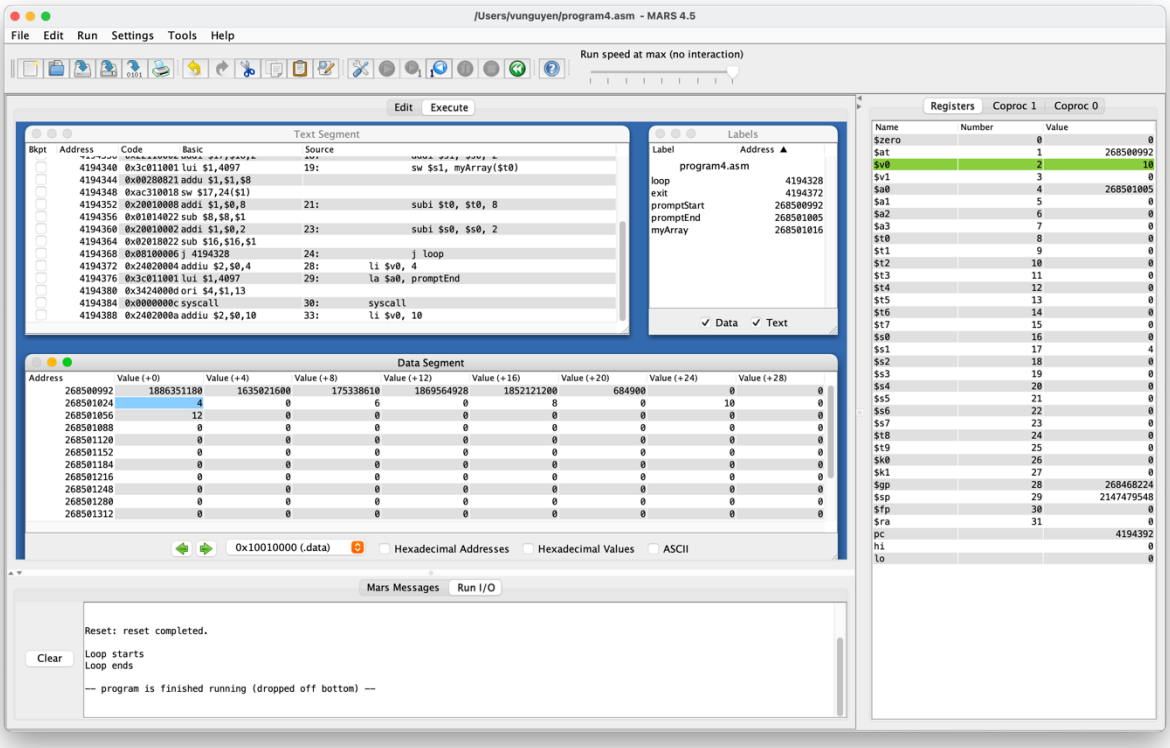
5.2. Program 2:



5.3. Program 3:



5.4. Program 4:


/Users/vunguyen/program4.asm - MARS 4.5
File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Text Segment
Bkpt Address Code Basic Source
4194340 0x3c011001 lui $1,4097 19: sw $s1, myArray($t0)
4194344 0x00280821 addu $1,$1,$8
4194348 0xac310018 sw $17,24($1)
4194352 0x20010008 addi $1,$0,8 21: subi $t0, $t0, 8
4194356 0x01014022 sub $8,$8,$1
4194360 0x20010002 addi $1,$0,2 23: subi $s0, $s0, 2
4194364 0x02018022 sub $16,$16,$1
4194368 0x08100006 j 4194328 24: j loop
4194372 0x24020004 addiu $2,$0,4 28: li $v0, 4
4194376 0x3c011001 lui $1,4097 29: la $a0, promptEnd
4194380 0x3424000d ori $4,$1,13
4194384 0x0000000c syscall 30: syscall
4194388 0x2402000a addiu $2,$0,10 33: li $v0, 10
Labels
Label Address
program4.asm
loop 4194328
exit 4194372
promptStart 268500992
promptEnd 268501005
myArray 268501016
Data Text
Data Segment
Address Value (+0) Value (+4) Value (+8) Value (+12) Value (+16) Value (+20) Value (+24) Value (+28)
268500992 1886351180 1635021600 175338610 1869564928 1852121200 684900 0 0
268501024 4 0 6 0 8 0 10 0
268501056 12 0 0 0 0 0 0 0
268501088 0 0 0 0 0 0 0 0
268501120 0 0 0 0 0 0 0 0
268501152 0 0 0 0 0 0 0 0
268501184 0 0 0 0 0 0 0 0
268501216 0 0 0 0 0 0 0 0
268501248 0 0 0 0 0 0 0 0
268501280 0 0 0 0 0 0 0 0
268501312 0 0 0 0 0 0 0 0
0x10010000 (.data)
Hexadecimal Addresses
Hexadecimal Values
ASCII
Mars Messages Run I/O
Clear
Reset: reset completed.
Loop starts
Loop ends
-- program is finished running (dropped off bottom) --
Registers Coproc 1 Coproc 0
Name Number Value
$zero 0 0
$at 1 268500992
$v0 2 10
$v1 3 0
$a0 4 268501005
$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 4
$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 4194392
hi 0
lo 0