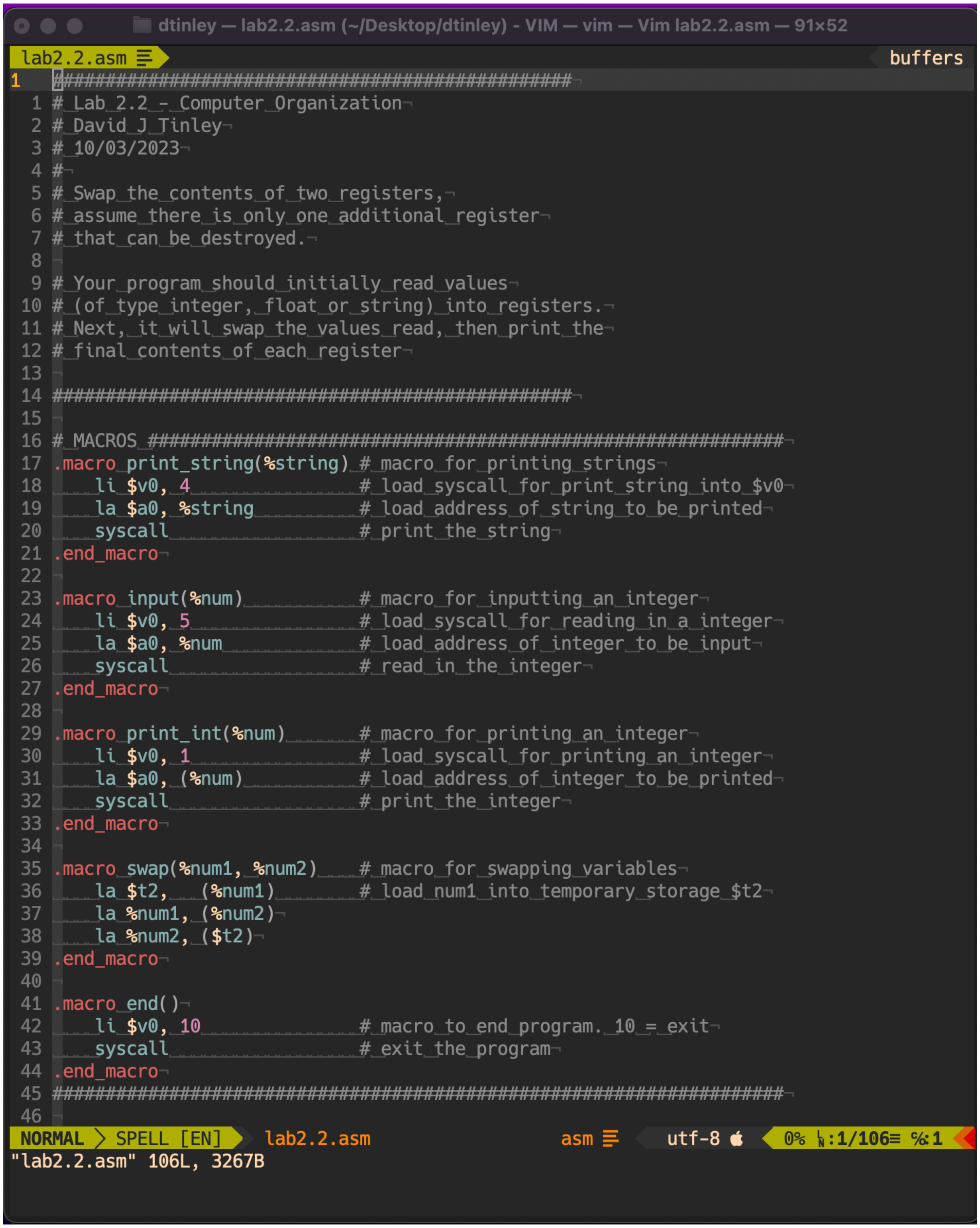
David J Tinley
10/03/2023
Lab 2.2 Report
Computer Organization

Again, I was able to use all my previously written macros for part 2 of this lab. The only new macro that I wrote was for the swapping of the two values.

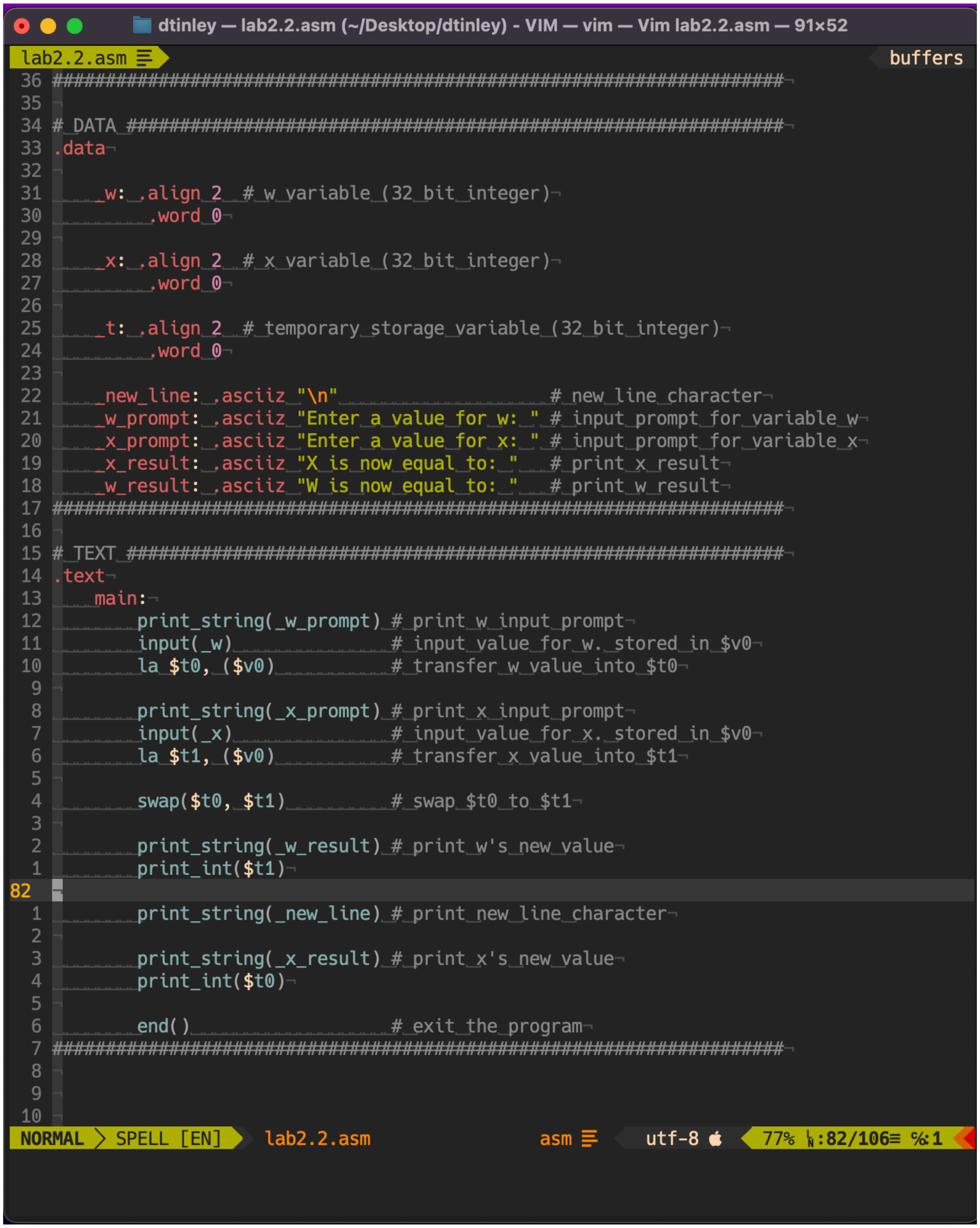


dtinley — lab2.2.asm (~/Desktop/dtinley) - VIM — vim — Vim lab2.2.asm — 91×52

```
#################################################
# Lab 2.2 - Computer Organization
# David J Tinley
# 10/03/2023
#
# Swap the contents of two registers,
# assume there is only one additional register
# that can be destroyed.

# Your program should initially read values
# (of type integer, float or string) into registers.
# Next, it will swap the values read, then print the
# final contents of each register

#################################################

# MACROS ############################################################
.macro print_string(%string) # macro for printing strings
    li $v0, 4                # load syscall for print string into $v0
    la $a0, %string          # load address of string to be printed
    syscall                  # print the string
.end_macro

.macro input(%num)           # macro for inputting an integer
    li $v0, 5                # load syscall for reading in a integer
    la $a0, %num             # load address of integer to be input
    syscall                  # read in the integer
.end_macro

.macro print_int(%num)       # macro for printing an integer
    li $v0, 1                # load syscall for printing an integer
    la $a0, (%num)           # load address of integer to be printed
    syscall                  # print the integer
.end_macro

.macro swap(%num1, %num2)    # macro for swapping variables
    la $t2,   (%num1)        # load num1 into temporary storage $t2
    la %num1, (%num2)
    la %num2, ($t2)
.end_macro

.macro end()
    li $v0, 10               # macro to end program. 10 = exit
    syscall                  # exit the program
.end_macro
#####################################################################

```

```
NORMAL > SPELL [EN]  lab2.2.asm        asm   utf-8   0%  :1/106≡ %:1
"lab2.2.asm" 106L, 3267B
```

The screenshot below continues with all the data that is declared for the program. Three integer variables are made for holding the two values input and the third is for the swapping function. The rest of the data are strings for printing prompts and printing results. Next in the text section I start with displaying the prompts one at a time and reading the input for the two variables. After input, the variables are loaded into temporary registers and then the swap macro is performed on them. The swap macro uses the basic algorithm of introducing a third variable to temporarily store the values in as you swap them. And finally, the new values of w and x are printed and the program calls the end macro that I made.



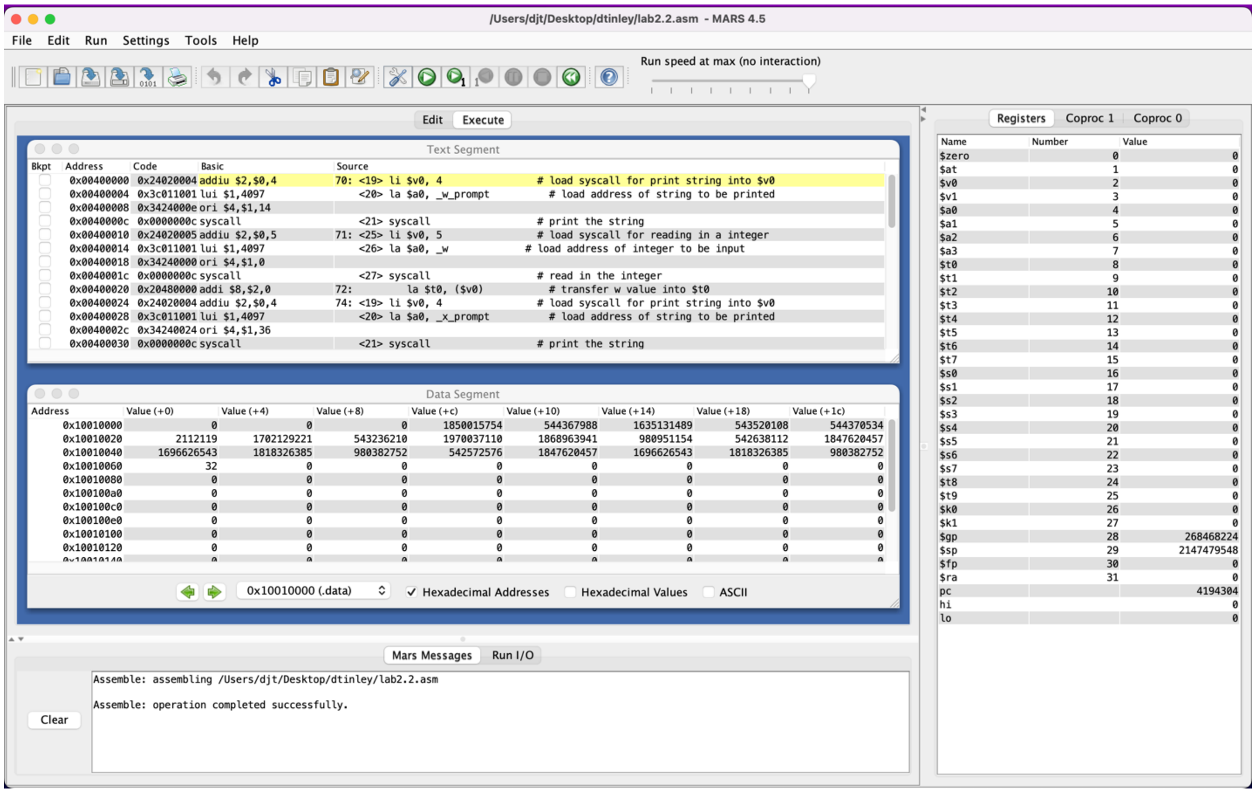
dtinley — lab2.2.asm (~/Desktop/dtinley) - VIM — vim — Vim lab2.2.asm — 91×52

lab2.2.asm ☰ buffers

```
#####################################################################

# DATA ##############################################################
.data

    _w:  .align 2  # w variable (32 bit integer)
         .word 0

    _x:  .align 2  # x variable (32 bit integer)
         .word 0

    _t:  .align 2  # temporary storage variable (32 bit integer)
         .word 0

    _new_line:  .asciiz "\n"                     # new line character
    _w_prompt:  .asciiz "Enter a value for w: "  # input prompt for variable w
    _x_prompt:  .asciiz "Enter a value for x: "  # input prompt for variable x
    _x_result:  .asciiz "X is now equal to: "    # print x result
    _w_result:  .asciiz "W is now equal to: "    # print w result
#####################################################################

# TEXT ##############################################################
.text
    main:
        print_string(_w_prompt) # print w input prompt
        input(_w)               # input value for w. stored in $v0
        la $t0, ($v0)           # transfer w value into $t0

        print_string(_x_prompt) # print x input prompt
        input(_x)               # input value for x. stored in $v0
        la $t1, ($v0)           # transfer x value into $t1

        swap($t0, $t1)          # swap $t0 to $t1

        print_string(_w_result) # print w's new value
        print_int($t1)

        print_string(_new_line) # print new line character

        print_string(_x_result) # print x's new value
        print_int($t0)

        end()                   # exit the program
#####################################################################
```

NORMAL > SPELL [EN] lab2.2.asm asm ☰ utf-8 77% ㏑:82/106≡ %:1

In conclusion I found part two of the lab assignment to be the easier one. I did not run into any issues for this one after learning more about the memory alignment from part 1.


/Users/djt/Desktop/dtinley/lab2.2.asm - MARS 4.5
File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Text Segment
Data Segment
Registers Coproc 1 Coproc 0
Mars Messages Run I/O
Clear

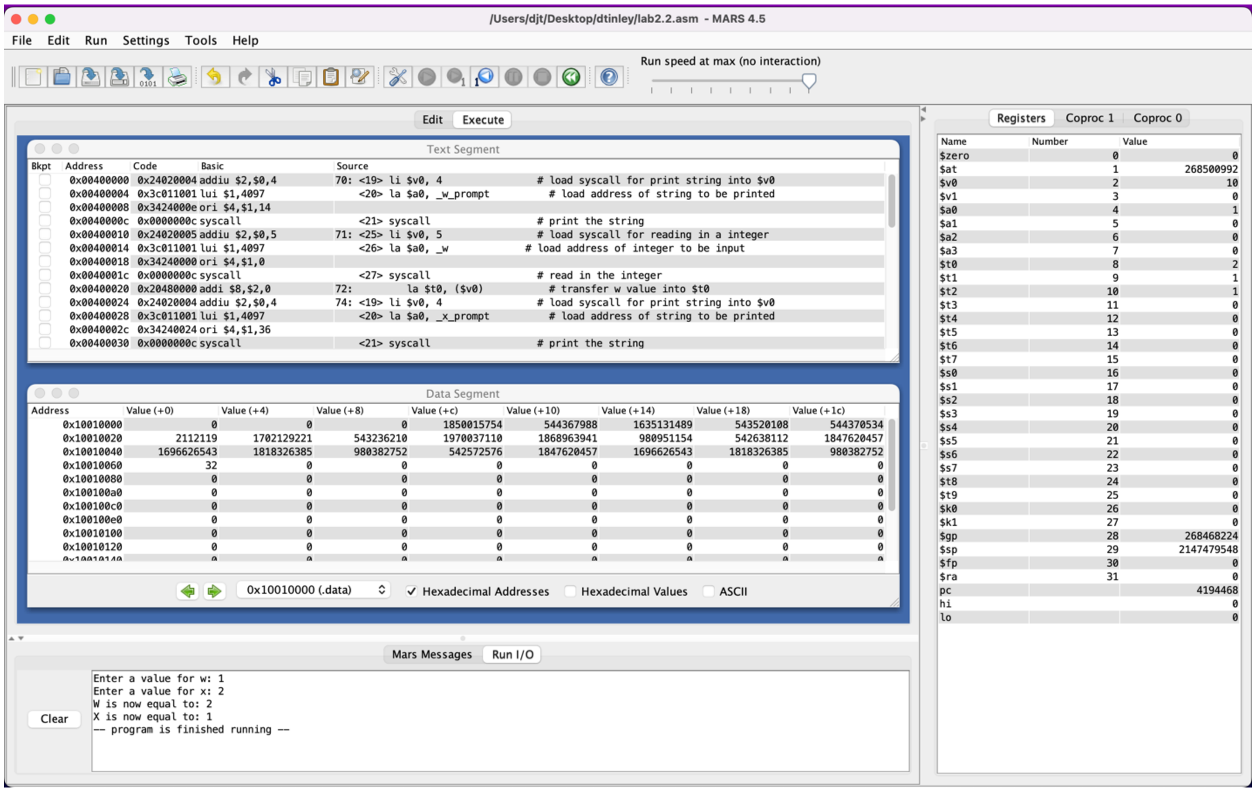
```
Bkpt Address    Code       Basic             Source
     0x00400000 0x24020004 addiu $2,$0,4     70: <19> li $v0, 4             # load syscall for print string into $v0
     0x00400004 0x3c011001 lui $1,4097           <20> la $a0, _w_prompt       # load address of string to be printed
     0x00400008 0x3424000e ori $4,$1,14
     0x0040000c 0x0000000c syscall               <21> syscall               # print the string
     0x00400010 0x24020005 addiu $2,$0,5     71: <25> li $v0, 5             # load syscall for reading in a integer
     0x00400014 0x3c011001 lui $1,4097           <26> la $a0, _w          # load address of integer to be input
     0x00400018 0x34240000 ori $4,$1,0
     0x0040001c 0x0000000c syscall               <27> syscall               # read in the integer
     0x00400020 0x20480000 addi $8,$2,0      72:         la $t0, ($v0)        # transfer w value into $t0
     0x00400024 0x24020004 addiu $2,$0,4     74: <19> li $v0, 4             # load syscall for print string into $v0
     0x00400028 0x3c011001 lui $1,4097           <20> la $a0, _x_prompt       # load address of string to be printed
     0x0040002c 0x34240024 ori $4,$1,36
     0x00400030 0x0000000c syscall               <21> syscall               # print the string
```

```
Address     Value (+0)  Value (+4)  Value (+8)  Value (+c)  Value (+10) Value (+14) Value (+18) Value (+1c)
0x10010000           0           0           0  1850015754   544367988  1635131489   543520108   544370534
0x10010020     2112119  1702129221   543236210  1970037110  1868963941   980951154   542638112  1847620457
0x10010040  1696626543  1818326385   980382752   542572576  1847620457  1696626543  1818326385   980382752
0x10010060          32           0           0           0           0           0           0           0
0x10010080           0           0           0           0           0           0           0           0
0x100100a0           0           0           0           0           0           0           0           0
0x100100c0           0           0           0           0           0           0           0           0
0x100100e0           0           0           0           0           0           0           0           0
0x10010100           0           0           0           0           0           0           0           0
0x10010120           0           0           0           0           0           0           0           0
```

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

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

```
Assemble: assembling /Users/djt/Desktop/dtinley/lab2.2.asm

Assemble: operation completed successfully.
```


/Users/djt/Desktop/dtinley/lab2.2.asm - MARS 4.5
File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Text Segment
Data Segment
Registers Coproc 1 Coproc 0
Mars Messages Run I/O
Clear

```
Bkpt Address    Code       Basic             Source
     0x00400000 0x24020004 addiu $2,$0,4     70: <19> li $v0, 4             # load syscall for print string into $v0
     0x00400004 0x3c011001 lui $1,4097           <20> la $a0, _w_prompt       # load address of string to be printed
     0x00400008 0x3424000e ori $4,$1,14
     0x0040000c 0x0000000c syscall               <21> syscall               # print the string
     0x00400010 0x24020005 addiu $2,$0,5     71: <25> li $v0, 5             # load syscall for reading in a integer
     0x00400014 0x3c011001 lui $1,4097           <26> la $a0, _w          # load address of integer to be input
     0x00400018 0x34240000 ori $4,$1,0
     0x0040001c 0x0000000c syscall               <27> syscall               # read in the integer
     0x00400020 0x20480000 addi $8,$2,0      72:         la $t0, ($v0)        # transfer w value into $t0
     0x00400024 0x24020004 addiu $2,$0,4     74: <19> li $v0, 4             # load syscall for print string into $v0
     0x00400028 0x3c011001 lui $1,4097           <20> la $a0, _x_prompt       # load address of string to be printed
     0x0040002c 0x34240024 ori $4,$1,36
     0x00400030 0x0000000c syscall               <21> syscall               # print the string
```

```
Address     Value (+0)  Value (+4)  Value (+8)  Value (+c)  Value (+10) Value (+14) Value (+18) Value (+1c)
0x10010000           0           0           0  1850015754   544367988  1635131489   543520108   544370534
0x10010020     2112119  1702129221   543236210  1970037110  1868963941   980951154   542638112  1847620457
0x10010040  1696626543  1818326385   980382752   542572576  1847620457  1696626543  1818326385   980382752
0x10010060          32           0           0           0           0           0           0           0
0x10010080           0           0           0           0           0           0           0           0
0x100100a0           0           0           0           0           0           0           0           0
0x100100c0           0           0           0           0           0           0           0           0
0x100100e0           0           0           0           0           0           0           0           0
0x10010100           0           0           0           0           0           0           0           0
0x10010120           0           0           0           0           0           0           0           0
```

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

Name	Number	Value
$zero	0	0
$at	1	268500992
$v0	2	10
$v1	3	0
$a0	4	1
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	2
$t1	9	1
$t2	10	1
$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		4194468
hi		0
lo		0

```
Enter a value for w: 1
Enter a value for x: 2
W is now equal to: 2
X is now equal to: 1
-- program is finished running --
```