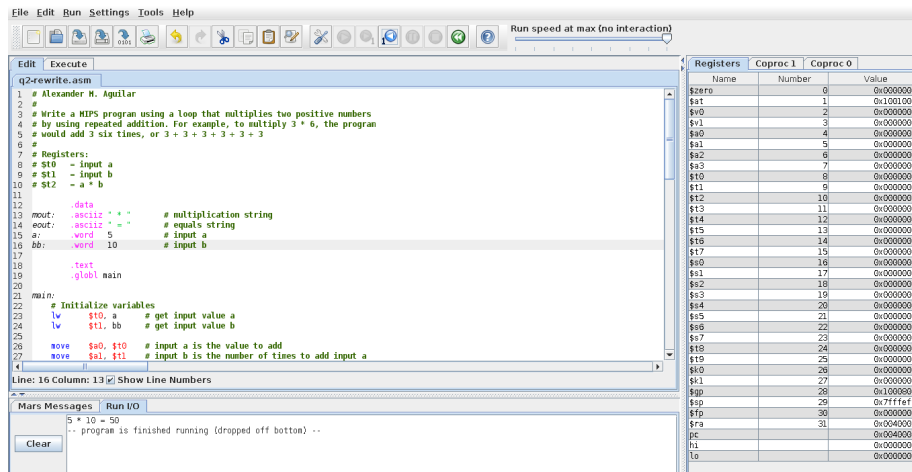


MIPS Lab Submission II

Alexander M. Aguilar

April 29, 2020

1 Question 2 Rewrite



The screenshot shows a MIPS simulator window with the following assembly code in the editor:

```
1 # Alexander M. Aguilar
2 #
3 # Write a MIPS program using a loop that multiplies two positive numbers
4 # by using repeated addition. For example, to multiply 3 * 6, the program
5 # would add 3 six times, or 3 + 3 + 3 + 3 + 3 + 3
6 #
7 # Registers:
8 # $t0 = input a
9 # $t1 = input b
10 # $t2 = a * b
11
12
13 .data
14 mout: .asciiz " * " # multiplication string
15 eout: .asciiz " = " # equals string
16 a: .word 5 # input a
17 b: .word 10 # input b
18
19 .text
20 .globl main
21
22 main:
23 # Initialize variables
24 lw $t0, a # get input value a
25 lw $t1, b # get input value b
26
27 move $s0, $t0 # input a is the value to add
28 move $s1, $t1 # input b is the number of times to add input a
29
30
```

The registers window on the right shows the following values:

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x10010000
\$v0	2	0x00000001
\$v1	3	0x00000000
\$a0	4	0x00000032
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000005
\$t1	9	0x0000000A
\$t2	10	0x00000032
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x00000005
\$s1	17	0x0000000A
\$s2	18	0x00000000
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$s8	24	0x00000000
\$s9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10000000
\$fp	29	0x7FFFFFFF
\$f0	30	0x00000000
\$ra	31	0x00400020
pc		0x00400008
hi		0x00000000
lo		0x00000000

```
1 # Alexander M. Aguilar
2 #
3 # Write a MIPS program using a loop that multiplies two positive numbers
4 # by using repeated addition. For example, to multiply 3 * 6, the program
5 # would add 3 six times, or 3 + 3 + 3 + 3 + 3 + 3
6 #
7 # Registers:
8 # $t0 = input a
9 # $t1 = input b
10 # $t2 = a * b
11
12
13 .data
14 mout: .asciiz " * " # multiplication string
15 eout: .asciiz " = " # equals string
16 a: .word 5 # input a
```

```

16  bb:      .word   10          # input b
17
18          .text
19          .globl main
20
21  main:
22      # Initialize variables
23      lw     $t0, a           # get input value a
24      lw     $t1, bb         # get input value b
25
26      move   $a0, $t0        # input a is the value to add
27      move   $a1, $t1        # input b is the number of times to add input a
28      li    $v0, 0           # set the resulting sum to 0
29
30      jal   multiply         # a * b
31      nop
32
33      move   $t2, $v0        # store multiplication result
34
35      j     done
36      nop
37
38  # Multiplication subroutine
39  #
40  # $a0          - number to add
41  # $a1          - number of times to add
42  # return $v0  - result of multiplication
43  multiply:
44      sub    $sp, $sp, 4     # push the return address
45      sw     $ra, ($sp)
46
47      beqz   $a1, mcomplete # if $a1 == 0, return
48
49      addu   $v0, $v0, $a0   # sum += input a
50      subi   $a1, $a1, 1     # a1--
51
52      # recursive call
53      jal   multiply
54      nop
55
56  mcomplete:
57      lw     $ra, ($sp)      # pop return address
58      add    $sp, $sp, 4
59
60      jr    $ra              # return to caller
61      nop
62
63  done:

```

```
64     # Finished, print results in the format of a * b = sum
65     li      $v0, 1
66     move   $a0, $t0
67     syscall
68
69     li      $v0, 4
70     la     $a0, mout
71     syscall
72
73     li      $v0, 1
74     move   $a0, $t1
75     syscall
76
77     li      $v0, 4
78     la     $a0, eout
79     syscall
80
81     li      $v0, 1
82     move   $a0, $t2
83     syscall
```
