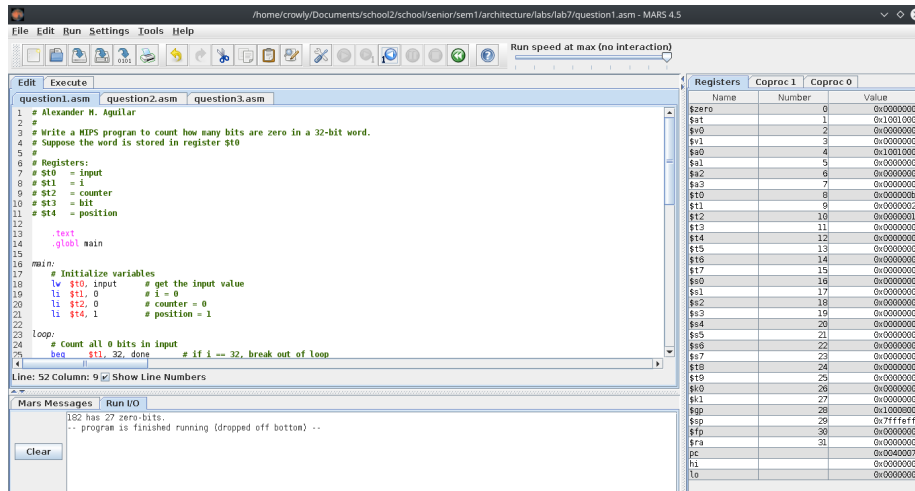


MIPS Lab Submission

Alexander M. Aguilar

April 22, 2020

1 First Question



The screenshot shows the Mars MIPS simulator interface. The main window displays the assembly code for 'question1.asm'. The code is as follows:

```
1 # Alexander M. Aguilar
2 #
3 # Write a MIPS program to count how many bits are zero in a 32-bit word.
4 # Suppose the word is stored in register $t0
5 #
6 # Registers:
7 # $t0 = input
8 # $t1 = i
9 # $t2 = counter
10 # $t3 = bit
11 # $t4 = position
12
13 .text
14 .globl main
15
16 main:
17     # Initialize variables
18     lw $t0, input    # get the input value
19     li $t1, 0        # i = 0
20     li $t2, 0        # counter = 0
21     li $t4, 1        # position = 1
22
23 Loop:
24     # Count all 0 bits in input
25     beq $t1, $2, done # if i == 32, break out of loop
26
27
28
29
30
31
32
```

The Register window on the right shows the state of the registers. The 'Zero' register is set to 1, indicating that the program has finished execution. The 'Number' register is 27, which is the number of zero bits in the input word. The 'Value' register is 0x00000000.

Name	Number	Coproc 1	Coproc 0	Value
\$zero	0			0x00000000
\$at	1			0x10010000
\$v0	2			0x00000004
\$v1	3			0x00000000
\$a0	4			0x10010008
\$a1	5			0x00000000
\$a2	6			0x00000000
\$a3	7			0x00000000
\$t0	8			0x00000006
\$t1	9			0x00000020
\$t2	10			0x0000001b
\$t3	11			0x00000000
\$t4	12			0x00000008
\$t5	13			0x00000000
\$t6	14			0x00000000
\$t7	15			0x00000000
\$t8	16			0x00000000
\$t9	17			0x00000000
\$s0	18			0x00000000
\$s1	19			0x00000000
\$s2	20			0x00000000
\$s3	21			0x00000000
\$s4	22			0x00000000
\$s5	23			0x00000000
\$s6	24			0x00000000
\$s7	25			0x00000000
\$s8	26			0x00000000
\$s9	27			0x00000000
\$k0	28			0x10000000
\$k1	29			0x7ffffcfc
\$k2	30			0x00000008
\$k3	31			0x00000000
pc				0x00400078
hi				0x00000000
lo				0x00000000

```
1 # Alexander M. Aguilar
2 #
3 # Write a MIPS program to count how many bits are zero in a 32-bit word.
4 # Suppose the word is stored in register $t0
5 #
6 # Registers:
7 # $t0 = input
8 # $t1 = i
9 # $t2 = counter
10 # $t3 = bit
11 # $t4 = position
12
13 .text
14 .globl main
```

```

15
16 main:
17     # Initialize variables
18     lw $t0, input # get the input value
19     li $t1, 0     # i = 0
20     li $t2, 0     # counter = 0
21     li $t4, 1     # position = 1
22
23 loop:
24     # Count all 0 bits in input
25     beq $t1, 32, done # if i == 32, break out of loop
26     nop                # delay
27
28     and $t3, $t0, $t4 # bit = input & position
29     bnez $t3, endif   # if (bit != 0), goto endif
30     nop                # delay
31     addiu $t2, $t2, 1 # counter++
32
33 endif:
34     addiu $t1, $t1, 1 # i++
35     sll $t4, $t4, 1  # position << 1
36
37     j loop           # loop again
38     nop              # delay
39
40 done:
41     # Finished, print results
42     li $v0, 1
43     move $a0, $t0
44     syscall
45
46     li $v0, 4
47     la $a0, hout
48     syscall
49
50     li $v0, 1
51     move $a0, $t2
52     syscall
53
54     li $v0, 4
55     la $a0, bout
56     syscall
57
58     .data
59 hout: .asciiz " has "
60 bout: .asciiz " zero-bits."
61 input: .word 182 # has 27 zero-bits
62 # 0000000000000000000000000000000010110110 in binary

```

2 Second Question

The screenshot shows the MARS MIPS simulator interface. The main window displays the MIPS assembly code for a program that calculates the product of two numbers (3 and 6) using a loop. The registers window on the right shows the state of the MIPS registers after execution. The registers \$t0, \$t1, and \$t3 contain the values 3, 6, and 18, respectively, which correspond to the variables 'a', 'b', and 'sum' in the program. The status bar at the bottom indicates that the program is finished running.

```
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 = i
11 # $t3 = sum
12 #
13 .text
14 .globl main
15
16 main:
17     # Initialize variables
18     lw $t0, a      # get input value a
19     lw $t1, b      # get input value b
20     li $t2, 0      # i = 0
21     li $t3, 0      # sum = 0
22
23 loop:
24     # Sum 'input a' a total of 'input b' times
25     sll $t2, $t2, $t2     # if i == input b, break out of loop
```

Name	Number	Value
\$zero	0	0x00000000
\$t1	1	0x10010000
\$t0	2	0x00000001
\$v1	3	0x00000000
\$a0	4	0x00000012
\$t1	5	0x00000000
\$t2	6	0x00000000
\$t3	7	0x00000000
\$t0	8	0x00000003
\$t1	9	0x00000006
\$t2	10	0x00000006
\$t3	11	0x00000012
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$t8	16	0x00000000
\$t9	17	0x00000000
\$t10	18	0x00000000
\$t11	19	0x00000000
\$t12	20	0x00000000
\$t13	21	0x00000000
\$t14	22	0x00000000
\$t15	23	0x00000000
\$t16	24	0x00000000
\$t17	25	0x00000000
\$t18	26	0x00000000
\$t19	27	0x00000000
\$t20	28	0x10000000
\$t21	29	0x7fffffff
\$t22	30	0x00000000
\$t23	31	0x00000000
PC		0x00400074
HI		0x00000000
LO		0x00000000

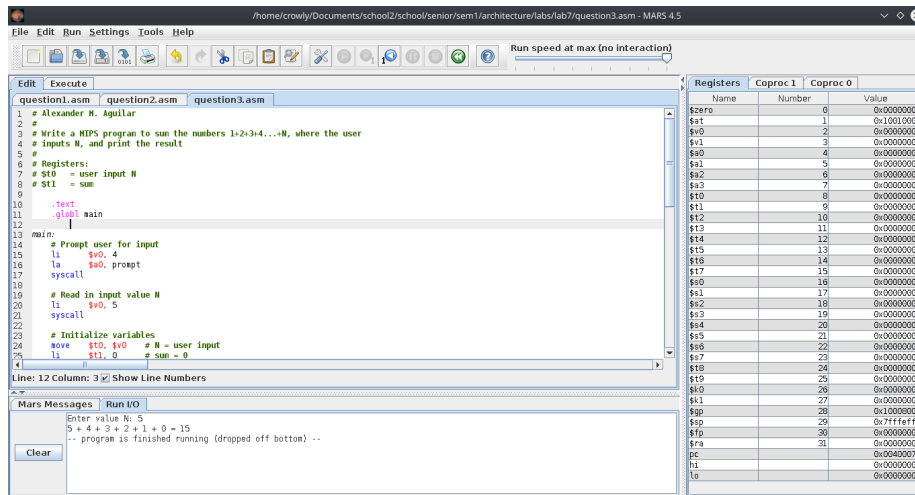
```
1 # Alexander M. Aguilar
2 #
3 # Write a MIPS program using a loop that multiplies two positive numbers
4 # by using repeated addition. For example, to multiply 3 * 6, the program
5 # would add 3 six times, or 3 + 3 + 3 + 3 + 3 + 3
6 #
7 # Registers:
8 # $t0 = input a
9 # $t1 = input b
10 # $t2 = i
11 # $t3 = sum
12 #
13 .text
14 .globl main
15
16 main:
17     # Initialize variables
18     lw $t0, a      # get input value a
19     lw $t1, b      # get input value b
20     li $t2, 0      # i = 0
21     li $t3, 0      # sum = 0
22
23 loop:
```

```

24     # Sum 'input a' a total of 'input b' times
25     beq    $t1, $t2, done    # if i == input b, break out of loop
26     nop                                # delay
27
28     addiu   $t2, $t2, 1      # i++
29     addu    $t3, $t3, $t0    # sum += input a
30
31     j      loop              # loop again
32     nop                                # delay
33
34 done:
35     # Finished, print results in the format of a * b = sum
36     li     $v0, 1
37     move   $a0, $t0
38     syscall
39
40     li     $v0, 4
41     la    $a0, mout
42     syscall
43
44     li     $v0, 1
45     move   $a0, $t1
46     syscall
47
48     li     $v0, 4
49     la    $a0, eout
50     syscall
51
52     li     $v0, 1
53     move   $a0, $t3
54     syscall
55
56     .data
57 mout: .asciiz " * "      # multiplication string
58 eout: .asciiz " = "     # equals string
59 a:    .word 3           # input a
60 bb:   .word 6           # input b

```

3 Third Question



```

1  # Alexander M. Aguilar
2  #
3  # Write a MIPS program to sum the numbers 1+2+3+4...+N, where the user
4  # inputs N, and print the result
5  #
6  # Registers:
7  # $t0 = user input N
8  # $t1 = sum
9
10 .text
11 .globl main
12
13 main:
14     # Prompt user for input
15     li    $v0, 4
16     la    $a0, prompt
17     syscall
18
19     # Read in input value N
20     li    $v0, 5
21     syscall
22
23     # Initialize variables
24     move  $t0, $v0    # N = user input
25     li    $t1, 0      # sum = 0
26
27 loop:
28     # Print current value of N
29     move  $a0, $t0

```

```

30     li    $v0, 1
31     syscall
32
33     # Add numbers 1+2+3+4...+N
34     beqz  $t0, done      # if N == 0, break out of loop
35     nop                    # delay
36
37     # Print plus sign
38     li    $v0, 4
39     la    $a0, pout
40     syscall
41
42     addu  $t1, $t1, $t0   # sum += N
43     subiu $t0, $t0, 1     # N--
44     j     loop           # loop again
45     nop                    # delay
46
47     done:
48     # Finished, print results
49     li    $v0, 4
50     la    $a0, eout
51     syscall
52
53     move  $a0, $t1
54     li    $v0, 1
55     syscall
56
57     .data
58     prompt: .asciiz  "Enter value N: "
59     pout:   .asciiz  " + "
60     eout:   .asciiz  " = "
61

```
