Portfolio 2 Report EEL3801C - C011

Yousef Awad

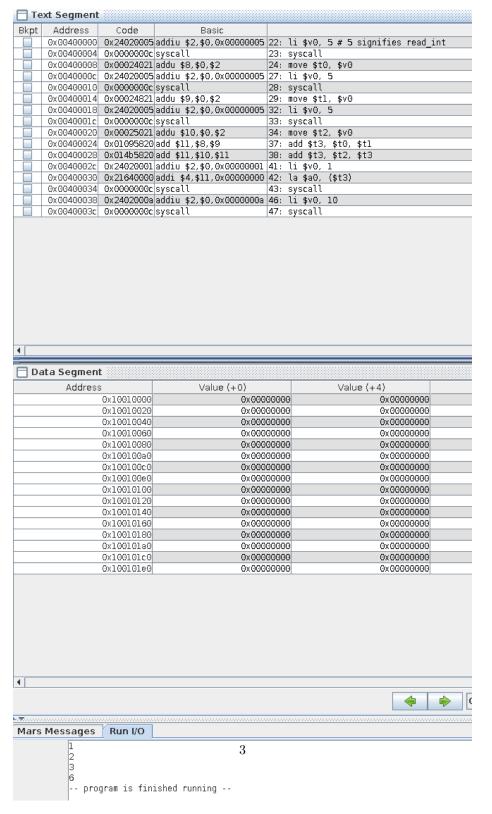
June 2025

Contents

1a																			2
	code .																		
	output		 																3
1b																			4
	code .																		4
	output		 																6
2a																			7
	code .																		7
	output																		
2b																			9
	code .																		9
	output																		10

1a

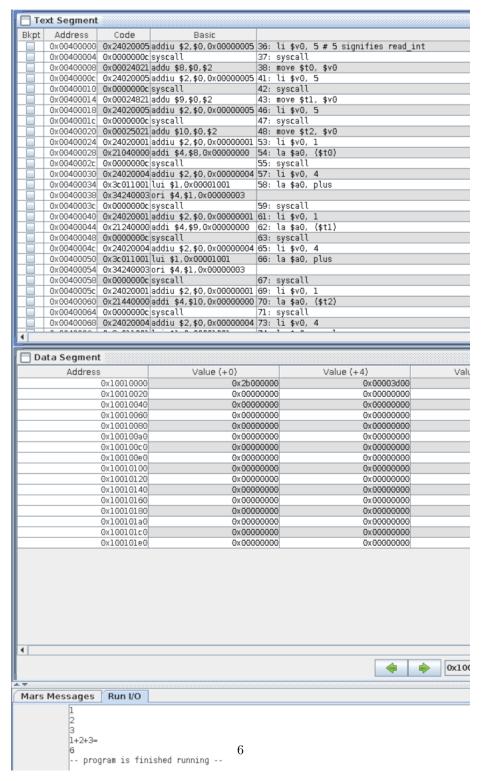
```
add.asm counter.asm counter-hexa.asm
                                    add-showinput.asm
 1 # Yousef Awad -- 06/02/2025
 2
3 # add.asm
 4 #
 5
   # A program that computes
 6
   # and prints the
 7
   # summation of three
   # numbers specficied at
 8
 9
   # runtime by the user.
10
   # Registers Used:
11
   # $t0 - used to hold the first number
12
   # $t1 - used to hold the second number
13
   # $t2 - used to hold the third number
14
   # $t3 - used to hold the sum
15
   # $v0 - syscall parameter and return value
16
    # $a0 - syscall parameter (print value)
17
18
19
   main:
20
21
    # Getting first number ($t0) from user
    li $v0, 5 # 5 signifies read int
22
23
    syscall
24
    move $t0, $v0
25
26
    # Getting second number ($t1) from user.
27
    li $v0, 5
28
    syscall
29
    move $t1, $v0
30
31
    # Getting third number ($t2) from user.
32
    li $v0, 5
33
    syscall
34
    move $t2, $v0
35
36
    # Computing triple sum
37
    add $t3, $t0, $t1
38
    add $t3, $t2, $t3
39
40
    # Printing out $t3 (sum)
41
    li $v0, 1
42
    la $a0, ($t3)
                          2
43
    syscall
44
45
    # Ending program
46
   li $v0, 10
47
    syscall
```



1b

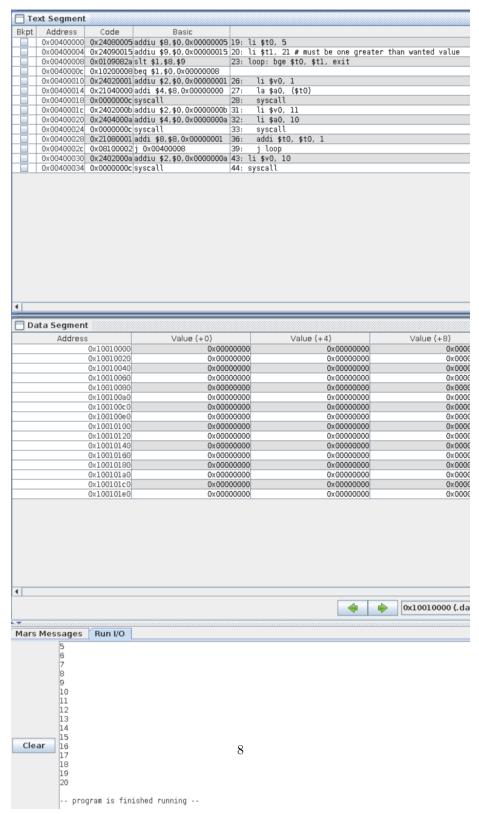
```
add.asm counter.asm counter-hexa.asm add-showinput.asm
 1 # Yousef Awad -- 06/02/2025
 2 #
3 # add-showinput.asm
 5 # A program that computes
 6
   # and prints the
 7 # summation of three
 8
   # numbers specficied at
   # runtime by the user.
 9
10 #
    # Program also shows
11
12 # the user's inputted
13 # three numbers unlike
14
   # add.asm
15 #
16 # Registers Used:
   # $t0 - used to hold the first number
17
18 # $t1 - used to hold the second number
   # $t2 - used to hold the third number
19
   # $t3 - used to hold the sum
20
21 # $v0 - syscall parameter and return value
22 # $a0 - syscall parameter (print value)
24 .data
25 # holds user inputted string
26 number1: .space 1
27 number2: .space 1
28 number3: .space 1
29 plus: .asciiz "+"
30 equal: .asciiz "="
31
      text
32 |
33 mai
    main:
35 # Getting first number ($t0) from user
36 li $v0, 5 # 5 signifies read_int
37 syscall
38 move $t0, $v0
39
40 # Getting second number ($t1) from user.
41
    li $v0, 5
42 syscall
43 move $t1, $v0
44
45 # Getting third number ($t2) from user.
46 li $v0, 5
47 syscall
48 move $t2, $v0
49
50 # Printing out user input
51 #
   # Printing first number
   li $v0, 1
53
   la $a0, ($t0)
syscall
54
55
56 # Printing
57
    li $v0, 4
58 la $a0, plus
59
   syscall
   # Printing second number
61
   li $v0, 1
   la $a0, ($t1)
syscall
62
63
   # Printing '+'
   li $v0, 4
65
   la $a0, plus
66
67 syscall
```

```
add.asm counter.asm counter-hexa.asm add-showinput.asm
24 .data
25 # holds user inputted string
26 number1: .space 1
27 number2: .space 1
28 number3: .space 1
29 plus: .asciiz "+"
30 equal: .asciiz "="
31 .text
32
33 main:
35 # Getting first number ($t0) from user
36 li $v0, 5 # 5 signifies read_int
37 syscall
38 move $t0, $v0
39
40 # Getting second number ($t1) from user.
41 li $v0, 5
42 syscall
43 move $t1, $v0
44
45 # Getting third number ($t2) from user.
46 li $v0, 5
47 syscall
48 move $t2, $v0
49
50 # Printing out user input
51 #
52 # Printing first number
53 li <mark>$v0</mark>, l
54
    la $a0, ($t0)
    syscall
56 # Printing '+'
57 li $v0, 4
58 la $a0, plus
59 syscall
60 # Printing second number
61 li $v0, 1
62 la $a0, ($t1)
63 syscall
64 # Printing '+'
65 li $v0, 4
66 la $a0, plus
67 syscall
68 # Printing third number
    li $v0, 1
69
70 la $a0, ($t2)
    syscall
# Printing
72
73 li $v0, 4
74 la $a0, equal
75 syscall
76 # Printing newline
     li $v0, 11
78 li $a0, 10
79
    syscall
80
81 # Computing triple sum
82 add $t3, $t0, $t1
83 add $t3, $t2, $t3
84
85
    # Printing out $t3 (sum)
86
    li $v0, 1
     la $a0, ($t3)
87
88
    syscall
89
                                             5
    # Ending program
90
91
    li $v0, 10
92 syscall
```



2a

```
add.asm counter.asm counter-hexa.asm add-showinput.asm
 1 # Yousef Awad -- 06/02/2025
   #
 2
 3 # counter.asm
 4 #
 5
   # A program that counts
  # from 5 to 20 while
 6
 7
   # displaying such to
   # the user in binary.
 8
 9
   #
10
   # Registers Used:
   # $t0 - Current Number
11
12
   # $t1 - Max Number
13 # $v0 - Syscall/Return Parameter
   # $a0 - Syscall Parameter (print)
14
15
16
   main:
17
18
   # initializing current and max
19
    li $t0, 5
20
   li $t1, 21 # must be one greater than wanted value
21
22
   # loop proper
23
   loop: bge $t0, $t1, exit
24
25
      # printing current number ($t0)
      li $v0, 1
26
27
      la $a0, ($t0)
28
      syscall
29
30
      # printing newline
31
      li $v0, 11
32
      li $a0, 10
33
      syscall
34
35
      # incrementing current by 1
36
      addi $t0, $t0, 1
37
38
      # jump to check if $t0 > $t1 - 1
39
      loop
40
41
   # exit branch (if $t0 > $t1 - 1)
42 exit:
43 li $v0, 10
44 syscall
```



2b

```
add.asm counter.asm counter-hexa.asm add-showinput.asm
1 # Yousef Awad -- 06/02/2025
3 # counter-hexa.asm
4 #
5 # A program that counts
6 # from 5 to 20 while
7 # displaying such to
8 # the user in hexadecimal.
9 #
10 # Registers Used:
11 # $t0 - Current Number
12 # $t1 - Max Number
13 # $v0 - Syscall/Return Parameter
   # $a0 - Syscall Parameter (print)
14
15
16
   main:
17
18
   # initializing current and max
19
   li $t0, 5
20
   li $t1, 21 # must be one greater than wanted value
21
22
   # loop proper
23
   loop: bge $t0, $t1, exit
24
25
      # printing current number ($t0)
26
      li $v0, 34 # code for printing in hexa
27
      la $a0, ($t0)
28
      syscall
29
30
      # printing newline
      li $v0, 11
li $a0, 10
31
32
33
      syscall
34
35
      # incrementing current by 1
36
      addi $t0, $t0, 1
37
38
      # jump to check if t0 > t1 - 1
39
      loop
40
   # exit branch (if $t0 > $t1 - 1)
41
42
   exit:
   li $v0, 10
43
44 syscall
```

