Computer Architecture Project

Group ID - 7

Problem # Assigned : 3

Problem Description:

Write a C program that inputs a temperature in Fahrenheit and converts it to degrees Celsius. The translation is given by the following formula: C = (5/9) * (F - 32).

Group Members:

Sidakpal Singh Sachdeva
(13114064)

Vikash Kumar
(13114073)

C Code

13114064_13114073.c

```
#include <stdio.h>

int main() {
    float fahren;
    printf("Please enter the temperature in
fahrenheit\n");
    scanf("%f", &fahren);
    float cel=(fahren-32);
    cel*=5.0;
    cel/=9.0;
    printf("Converted temp in celsius is :
%f",cel );
    return 0;
}
```

Generated Assembly File (.s)

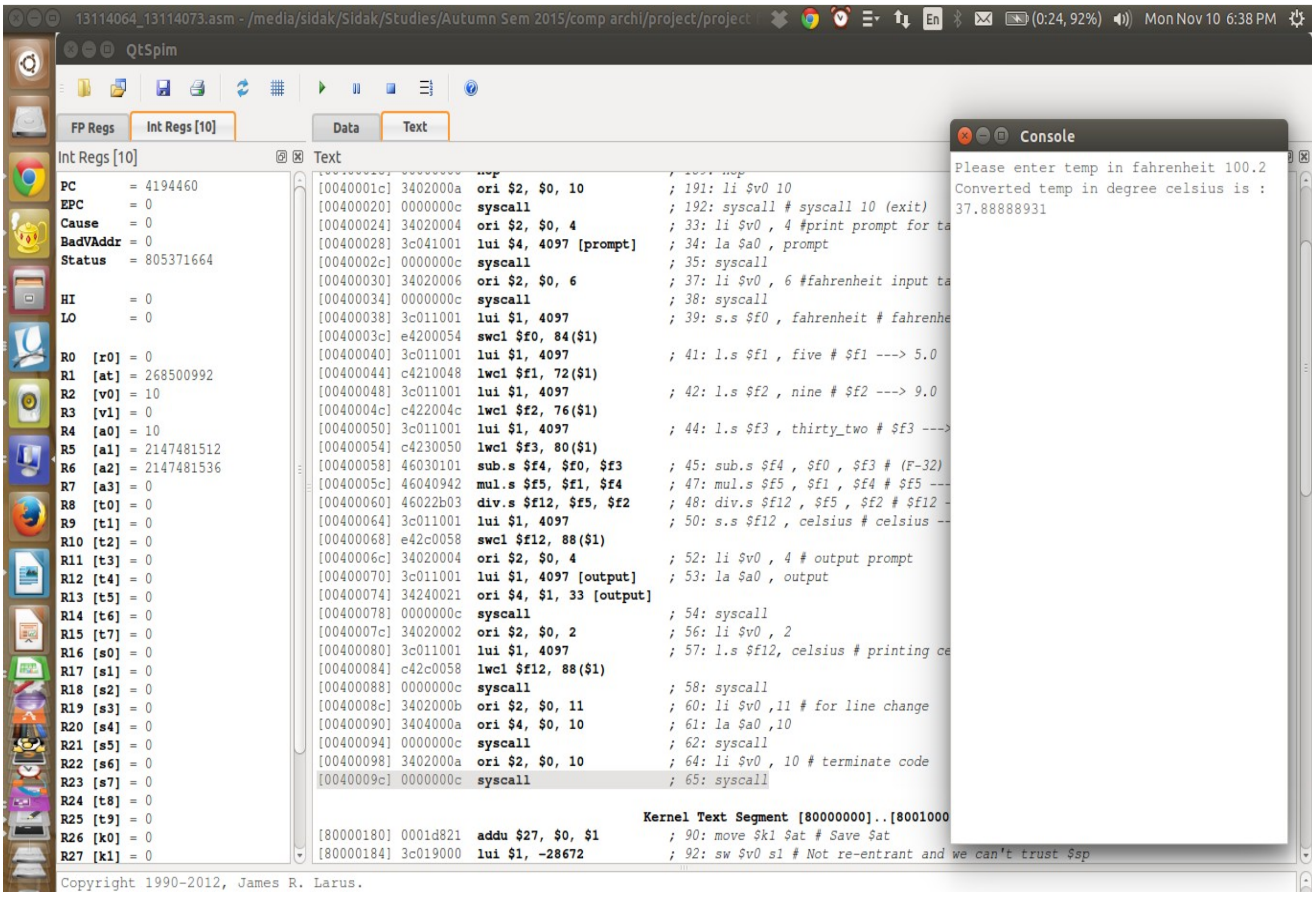
13114064_13114073.s

```
	.file	"13114064_13114073.c"
	.section .rodata
	.align 8
.LC0:
	.string	"Please enter the temperature in fahrenheit"
.LC1:
	.string	"%f"
	.align 8
.LC5:
	.string	"Converted temp in celsius is : %f"
	.text
	.globl	main
	.type	main, @function
main:
.LFB0:
	.cfi_startproc
	pushq	%rbp
	.cfi_def_cfa_offset 16
	.cfi_offset 6, -16
	movq	%rsp, %rbp
	.cfi_def_cfa_register 6
	subq	$16, %rsp
	movl	$.LC0, %edi
	call	puts
	leaq	-8(%rbp), %rax
	movq	%rax, %rsi
	movl	$.LC1, %edi
	movl	$0, %eax
	call	__isoc99_scanf
	movss	-8(%rbp), %xmm0
	movss	.LC2(%rip), %xmm1
	subss	%xmm1, %xmm0
	movss	%xmm0, -4(%rbp)
	movss	-4(%rbp), %xmm1
	movss	.LC3(%rip), %xmm0
	mulss	%xmm1, %xmm0
	movss	%xmm0, -4(%rbp)
	movss	-4(%rbp), %xmm0
	movss	.LC4(%rip), %xmm1
	divss	%xmm1, %xmm0
	movss	%xmm0, -4(%rbp)
	movss	-4(%rbp), %xmm0
	cvtps2pd %xmm0, %xmm0
	movl	$.LC5, %edi
	movl	$1, %eax
	call	printf
	movl	$0, %eax
	leave
	.cfi_def_cfa 7, 8
	ret
	.cfi_endproc
.LFE0:
	.size	main, .-main
	.section .rodata
	.align 4
.LC2:
	.long	1107296256
	.align 4
.LC3:
	.long	1084227584
	.align 4
.LC4:
	.long	1091567616
	.ident	"GCC: (Ubuntu 4.8.2-19ubuntu1) 4.8.2"
	.section .note.GNU-stack,"",@progbits
```

Assembly Program for assigned problem

```
##########################################################################
## GroupID- 7 (13114073_13114064) - Vikash Kumar & Sidakpal Singh Sachdeva
## Date: 08/11/2014
## 13114064_13114073.asm -
##    Mips program to change input fahrenheit value to celsius ##
      value , as C = 5*(F-32)/9

##Registers used:
##  $f0 - used to hold fahrenheit input (can be float value) $f0---> F
##  $f1 - used to hold value 5.0
##  $f2 - used to hold value 9.0
##  $f3 - used to hold value 32.0
##  $f4 - used to hold value (F-32.0)
##  $f5 - used to hold value 5.0*(F-32.0)
##  $f12 - used to hold value 5.0*(F-32.0)/9.0 , which is converted
           celsius value
##########################################################################
.data
     prompt:    .asciiz         "Please enter temp in fahrenheit "
                                     # input prompt for fahrenheit input
     output:    .asciiz         "Converted temp in degree celsius is : "
                                     # output prompt for celcius

     five:         .float       5.0        # 5.0 value as float

     nine:         .float       9.0        # 9.0 value as float

     thirty_two:   .float       32.0


     fahrenheit:   .float       0.0
                                # fahrenheit as float for input storage
     celsius:      .float       0.0
                                # celsius as float for output storage

.text
                                # main starts here
.globl main
.ent main

main:
     li   $v0 , 4               #print prompt for taking fahrenheit input
     la   $a0 , prompt
     syscall

     li  $v0 , 6                #fahrenheit input taken
     syscall
     s.s $f0 , fahrenheit       # fahrenheit ---> $f0

     l.s  $f1 , five             # $f1 ---> 5.0
     l.s  $f2 , nine            # $f2 ---> 9.0
     l.s   $f3 , thirty_two      # $f3 ---> 32.0
     sub.s $f4 , $f0 , $f3       # $f4 ---> (F-32.0)

     mul.s $f5 , $f1 , $f4       # $f5 ---> 5.0*(F-32.0)
     div.s $f12 , $f5 , $f2      # $f12 ---> 5.0/9.0*(f-32.0)

     s.s   $f12 , celsius        # celsius ---> $f12

     li    $v0 , 4               # output prompt
     la    $a0 , output
     syscall

     li         $v0 , 2
     l.s   $f12, celsius         # printing celsius value
     syscall

     li         $v0 ,11          # for line change
     la         $a0 ,10
     syscall

     li         $v0 , 10         # terminate code
     syscall
.end main
```

Screenshot for above program



```
Int Regs [10]
PC       = 4194460
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 805371664

HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 268500992
R2  [v0] = 10
R3  [v1] = 0
R4  [a0] = 10
R5  [a1] = 2147481512
R6  [a2] = 2147481536
R7  [a3] = 0
R8  [t0] = 0
R9  [t1] = 0
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
```

```
Text
[0040001c] 3402000a  ori $2, $0, 10            ; 191: li $v0 10
[00400020] 0000000c  syscall                   ; 192: syscall # syscall 10 (exit)
[00400024] 34020004  ori $2, $0, 4             ; 33: li $v0 , 4 #print prompt for ta
[00400028] 3c041001  lui $4, 4097 [prompt]     ; 34: la $a0 , prompt
[0040002c] 0000000c  syscall                   ; 35: syscall
[00400030] 34020006  ori $2, $0, 6             ; 37: li $v0 , 6 #fahrenheit input ta
[00400034] 0000000c  syscall                   ; 38: syscall
[00400038] 3c011001  lui $1, 4097              ; 39: s.s $f0 , fahrenheit # fahrenhe
[0040003c] e4200054  swc1 $f0, 84($1)
[00400040] 3c011001  lui $1, 4097              ; 41: l.s $f1 , five # $f1 ---> 5.0
[00400044] c4210048  lwc1 $f1, 72($1)
[00400048] 3c011001  lui $1, 4097              ; 42: l.s $f2 , nine # $f2 ---> 9.0
[0040004c] c422004c  lwc1 $f2, 76($1)
[00400050] 3c011001  lui $1, 4097              ; 44: l.s $f3 , thirty_two # $f3 --->
[00400054] c4230050  lwc1 $f3, 80($1)
[00400058] 46030101  sub.s $f4, $f0, $f3       ; 45: sub.s $f4 , $f0 , $f3 # (F-32)
[0040005c] 46040942  mul.s $f5, $f1, $f4       ; 47: mul.s $f5 , $f1 , $f4 # $f5 ---
[00400060] 46022b03  div.s $f12, $f5, $f2      ; 48: div.s $f12 , $f5 , $f2 # $f12 -
[00400064] 3c011001  lui $1, 4097              ; 50: s.s $f12 , celsius # celsius --
[00400068] e42c0058  swc1 $f12, 88($1)
[0040006c] 34020004  ori $2, $0, 4             ; 52: li $v0 , 4 # output prompt
[00400070] 3c011001  lui $1, 4097 [output]     ; 53: la $a0 , output
[00400074] 34240021  ori $4, $1, 33 [output]
[00400078] 0000000c  syscall                   ; 54: syscall
[0040007c] 34020002  ori $2, $0, 2             ; 56: li $v0 , 2
[00400080] 3c011001  lui $1, 4097              ; 57: l.s $f12, celsius # printing ce
[00400084] c42c0058  lwc1 $f12, 88($1)
[00400088] 0000000c  syscall                   ; 58: syscall
[0040008c] 3402000b  ori $2, $0, 11            ; 60: li $v0 ,11 # for line change
[00400090] 3404000a  ori $4, $0, 10            ; 61: la $a0 ,10
[00400094] 0000000c  syscall                   ; 62: syscall
[00400098] 3402000a  ori $2, $0, 10            ; 64: li $v0 , 10 # terminate code
[0040009c] 0000000c  syscall                   ; 65: syscall

                                          Kernel Text Segment [80000000]..[8001000
[80000180] 0001d821  addu $27, $0, $1          ; 90: move $k1 $at # Save $at
[80000184] 3c019000  lui $1, -28672            ; 92: sw $v0 s1 # Not re-entrant and we can't trust $sp
```

```
Console
Please enter temp in fahrenheit 100.2
Converted temp in degree celsius is :
37.88888931
```

```
Copyright 1990-2012, James R. Larus.
```

Assembly Program for assigned problem

```
##########################################################################
## GroupID-7 (13114064_13114073) - Sidakpal Singh Sachdeva & Vikash Kumar
## Date: 09/11/2014
##13114064_13114073_ascii.asm -
##   Mips Assembly Program to verify that multidigit ASCII addition of
##   the strings gives the same result as sum calculated by converting
##   the ASCII strings into integer.
##
## Registers used in main :  no temporary or saved registers used
##
## Registers used in method1 :
##          $t5 -> to hold the final answer
##          $t1 -> to hold the first integer returned from method
##          $t2 -> to hold the second integer returned from method
##
## Registers used in method2 :
##         $t0 -> to store the len of string
##         $t1 -> to hold current char from str1
##         $t2 -> to hold current char from str2
##         $t3 -> to store base address of str3
##         $t4 -> to hold the value of endCarry
##         $t5 -> to hold value to check for carry
##         $t6 -> to hold intermediate value of $t1
##         $t7 -> to hold value for checking if carry took place
##
## Registers used in asciiToInt :
##          $t4 -> To hold the base address of the input string
##          $t0 -> To hold the final result
##         $t1 -> To hold the length of the string
##          $t2 -> To be used as counter
##          $t3 -> To store a particular byte from the input string
##
##########################################################################

# Data Declarations

.data
     str1: .space 9        # str1 stores the value of 1st enrollment no.
     str2: .space 9        # str2 stores the value of 2nd enrollment no.
     str3: .space 10       # str3 stores the value of final answer (sum).
     temp: .space 9
     len:  .word 8         # len stores the value of length of string
     int1: .word 0
     int2: .word 0
     prompt:    .asciiz "Please enter the two enrollment numbers \n"
     msg1: .asciiz "\n Answer calculated by converting ascii to int \n"
     msg2: .asciiz "\n Answer calculated by multi-digit ascii addition \n"
     flag: .word 0

#
```

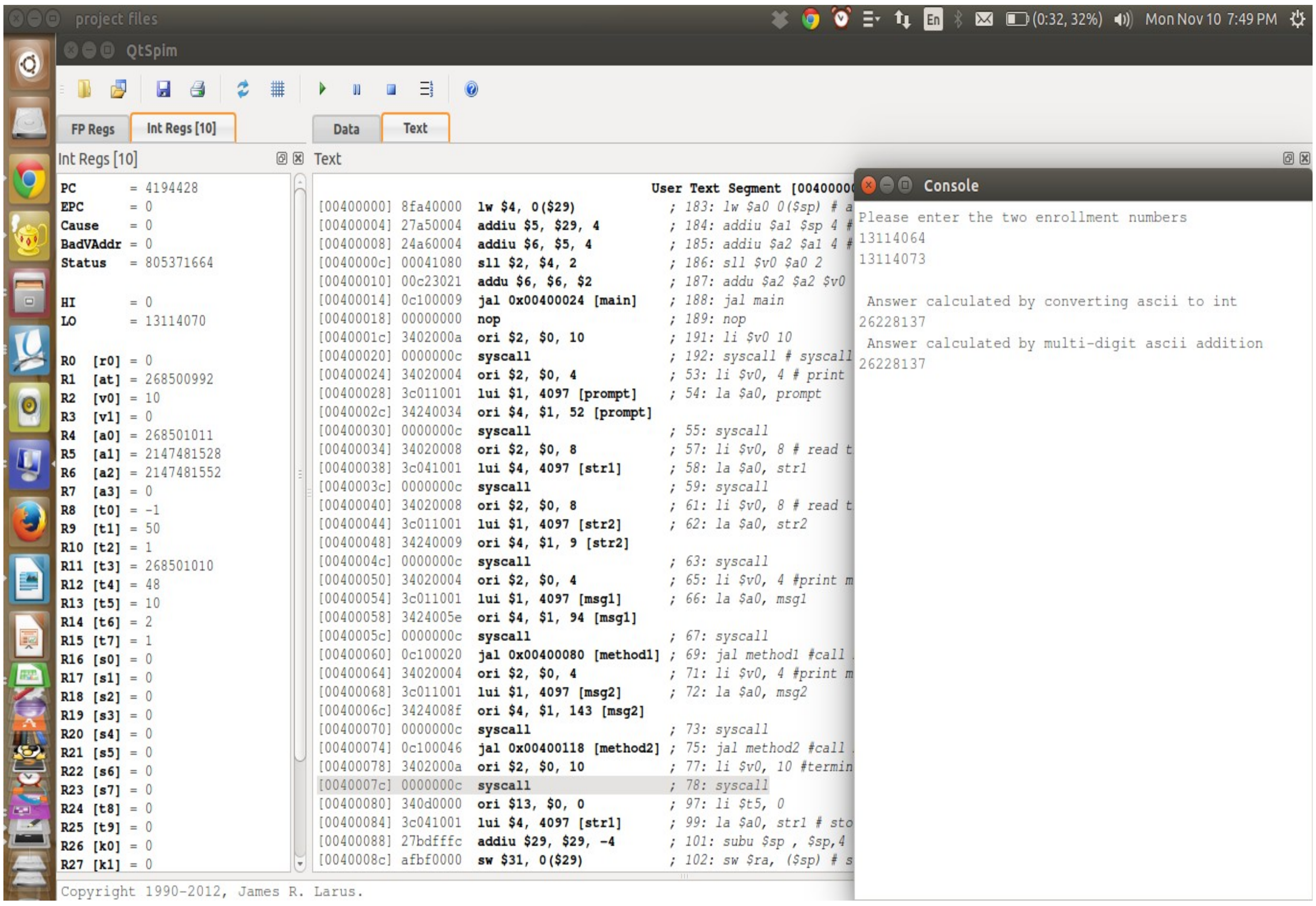
```
--------------------------------------------------------------------------
-----
# Code Section

.text

# -----The main entry program--------#

.globl main
.ent main
main:

      li $v0, 4              # print the prompt string
      la $a0, prompt
      syscall

      li $v0, 8              # read the str1
      la $a0, str1
      syscall

      li $v0, 8              # read the str2
      la $a0, str2
      syscall

      li $v0, 4              #print msg1
      la $a0, msg1
      syscall

      jal method1                 #call method1

      li $v0, 4              #print msg2
      la $a0, msg2
      syscall

      jal method2                 #call method2

      li $v0, 10             #terminate program
      syscall

.end main

# -----This method prints the sum by converting each string to
integer--------#
      # Arguments
      # NULL

      # Returns
      # NULL

      # Registers used
      # $t5 -> to hold the final answer
      # $t1 -> to hold the first integer returned from method
      # $t2 -> to hold the second integer returned from method
```

```
#---------------------------------------------------------------------------
-----#
.globl method1
.ent method1
method1:
      li $t5, 0

      la $a0, str1    # store the address of str1 in $a0 to pass as an
                        argument
      subu $sp , $sp,4      # allocate space in stack
      sw $ra, ($sp)         # store the value of $ra in stack

      jal asciiToInt  # call helper method to convert the str1 to integer
                        word

      lw $ra , ($sp)              # restore the original value of $ra from
                                    stack
      addu $sp, $sp, 4            # free space in stack

      move $t1, $v0               # store the return value of the method in
                                   $t1

      add $t5,$t5, $t1

      la $a0, str2    # store the address of str2 in $a0 to pass as an
                        argument

      subu $sp , $sp,4             # allocate space in stack
      sw $ra, ($sp)               # store the value of $ra in stack

jal asciiToInt  # call helper method to convert the str2 to integer word

      lw $ra , ($sp)              # restore the original value of $ra from
                                    stack
      addu $sp, $sp, 4           # free space in stack

      move $t2, $v0               # store the return value of the method in
                                    $t2

      add $t5,$t5, $t2

      li $v0, 1                   # output the final sum
      move $a0, $t5
      syscall

      j $ra                       # jump back to calling routine

.end method1

# -----Helper method to convert ASCII to integer--------#

      # Arguments
      # $a0 -> Argument passed on by the caller method
      #    which stores the base address of the string


      # Returns
      # $v0 -> Integer value of the string

      # Registers used
      # $t4 -> To hold the base address of the input string
      # $t0 -> To hold the final result
      # $t1 -> To hold the length of the string
      # $t2 -> To be used as counter
      # $t3 -> To store a particular byte from the input string
# -------------------------------------------------------#
.globl asciiToInt
.ent asciiToInt

asciiToInt:
                                      # load registers with initial
values
      move $t4, $a0

      li $t0, 0
      lw $t1, len

      li $t2, 0

      loop:

            lb $t3, 0($t4)        # $t3 = Mem[ $t4 + 0]
            sub $t3, $t3, 48# $t3 = $t3 - 48 , convert from ascii
                                               representation to integer
            mul $t0, $t0, 10# $t0 = $t0 * 10
            add $t0, $t0, $t3     # $t0 = $t0 + $t3

            add $t2, $t2, 1       # increment the value in register
                                    (counter)
            add $t4, $t4, 1       # increment the offset for accessing the
                                    next byte

            blt $t2, $t1, loop  # if $t2 < $t1 , branch to loop

      move $v0, $t0               # store the final integer result in $v0
      j $ra                       # jump back to caller method
.end asciiToInt

# -----This method prints the sum by digit-by-digit addition--------#

      # Arguments
      # NULL

      # Returns
      # NULL
```

```
     # Registers used
     # $t0 -> to store the len of string
     # $t1 -> to hold current char from str1
     # $t2 -> to hold current char from str2
     # $t3 -> to store base address of str3
     # $t4 -> to hold the value of endCarry
     # $t5 -> to hold value to check for carry
     # $t6 -> to hold intermediate value of $t1
     # $t7 -> to hold value for checking if carry took place
#-----------------------------------------------------------------------#
.globl method2
.ent method2
method2:


     lw $t0, len                # store the len of string in $t0
     sub $t0, $t0, 1       # index i for str1 and str2

     la $t3, str3               #store base address of str3

     li $t4, 0                  #$t4 is endCarry

     li $t5, 10

     addiu $t3, $t3, 9          # get to the last byte in str3
     sb $zero, 0($t3)      # store null character there
     subu $t3, $t3, 1

     addStr:
          lb $t1, str1($t0)     # $t1= Mem[str1 + $t0]
          lb $t2, str2($t0)     # $t2= Mem[str2 + $t0]
          sub $t1, $t1,48       # $t1= $t1-48
          sub $t2, $t2,48       # $t2= $t2-48

          add $t1, $t1, $t2     # $t1= $t1 + $t2
          add $t1, $t1, $t4     # $t1 = $t1 + $t4
          move $t6, $t1
          bge $t6, $t5, handleCarry # if $t6>=$t5 , goto handleCarry
          blt $t6, $t5, noCarry   # if $t6<$t5 , goto noCarry

     cont:
          sub $t0, $t0, 1       # decrease counter
          subu $t3, $t3, 1# decrease index
          bltz $t0, lastCase    # if $t0<0 , goto lastCase
          j addStr


     handleCarry:
          addi $t1, $t1, 38
          sb $t1, 0($t3)        # Mem[$t3 +0] = $t1
          li $t4, 1

          j cont                     # continue execution in addStr
```

```
    noCarry:
         addi $t1, $t1, 48
         sb $t1, 0($t3)       # Mem[$t3 +0] = $t1
         li $t4, 0

         j cont                    # continue execution in addStr

    lastCase:
         move $t7, $t4
         addi $t4, $t4, 48
         sb $t4, 0($t3)       # Mem[$t3 +0] = $t4

         blez $t7, printWithoutCarry #$t7 <=0, goto printWithoutCarry

         move $t7, $zero
         j printWithCarry



    printWithoutCarry:
         la $t7, str3
         addiu $t7, $t7, 1
         li $v0, 4       # print ans string
         move $a0, $t7
         syscall

         lw $t7, flag
         addi $t7, $t7, 1

    printWithCarry:
         bgtz $t7, return# $t7 >0, goto return
         li $v0, 4       # print ans string
         la $a0, str3
         syscall

    return:
         j $ra

.end method2

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

Screenshot for above program



```
PC       = 4194428
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 805371664

HI       = 0
LO       = 13114070

R0  [r0] = 0
R1  [at] = 268500992
R2  [v0] = 10
R3  [v1] = 0
R4  [a0] = 268501011
R5  [a1] = 2147481528
R6  [a2] = 2147481552
R7  [a3] = 0
R8  [t0] = -1
R9  [t1] = 50
R10 [t2] = 1
R11 [t3] = 268501010
R12 [t4] = 48
R13 [t5] = 10
R14 [t6] = 2
R15 [t7] = 1
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
```

```
                                                  User Text Segment [00400000
[00400000] 8fa40000  lw $4, 0($29)              ; 183: lw $a0 0($sp) # a
[00400004] 27a50004  addiu $5, $29, 4           ; 184: addiu $a1 $sp 4 #
[00400008] 24a60004  addiu $6, $5, 4            ; 185: addiu $a2 $a1 4 #
[0040000c] 00041080  sll $2, $4, 2              ; 186: sll $v0 $a0 2
[00400010] 00c23021  addu $6, $6, $2            ; 187: addu $a2 $a2 $v0
[00400014] 0c100009  jal 0x00400024 [main]      ; 188: jal main
[00400018] 00000000  nop                        ; 189: nop
[0040001c] 3402000a  ori $2, $0, 10             ; 191: li $v0 10
[00400020] 0000000c  syscall                    ; 192: syscall # syscall
[00400024] 34020004  ori $2, $0, 4              ; 53: li $v0, 4 # print
[00400028] 3c011001  lui $1, 4097 [prompt]      ; 54: la $a0, prompt
[0040002c] 34240034  ori $4, $1, 52 [prompt]
[00400030] 0000000c  syscall                    ; 55: syscall
[00400034] 34020008  ori $2, $0, 8              ; 57: li $v0, 8 # read t
[00400038] 3c041001  lui $4, 4097 [str1]        ; 58: la $a0, str1
[0040003c] 0000000c  syscall                    ; 59: syscall
[00400040] 34020008  ori $2, $0, 8              ; 61: li $v0, 8 # read t
[00400044] 3c011001  lui $1, 4097 [str2]        ; 62: la $a0, str2
[00400048] 34240009  ori $4, $1, 9 [str2]
[0040004c] 0000000c  syscall                    ; 63: syscall
[00400050] 34020004  ori $2, $0, 4              ; 65: li $v0, 4 #print m
[00400054] 3c011001  lui $1, 4097 [msg1]        ; 66: la $a0, msg1
[00400058] 3424005e  ori $4, $1, 94 [msg1]
[0040005c] 0000000c  syscall                    ; 67: syscall
[00400060] 0c100020  jal 0x00400080 [method1]   ; 69: jal method1 #call
[00400064] 34020004  ori $2, $0, 4              ; 71: li $v0, 4 #print m
[00400068] 3c011001  lui $1, 4097 [msg2]        ; 72: la $a0, msg2
[0040006c] 3424008f  ori $4, $1, 143 [msg2]
[00400070] 0000000c  syscall                    ; 73: syscall
[00400074] 0c100046  jal 0x00400118 [method2]   ; 75: jal method2 #call
[00400078] 3402000a  ori $2, $0, 10             ; 77: li $v0, 10 #termin
[0040007c] 0000000c  syscall                    ; 78: syscall
[00400080] 340d0000  ori $13, $0, 0             ; 97: li $t5, 0
[00400084] 3c041001  lui $4, 4097 [str1]        ; 99: la $a0, str1 # sto
[00400088] 27bdfffc  addiu $29, $29, -4         ; 101: subu $sp , $sp,4
[0040008c] afbf0000  sw $31, 0($29)             ; 102: sw $ra, ($sp) # s
```

```
Please enter the two enrollment numbers
13114064
13114073

 Answer calculated by converting ascii to int
26228137
 Answer calculated by multi-digit ascii addition
26228137
```

Copyright 1990-2012, James R. Larus.

Thanks