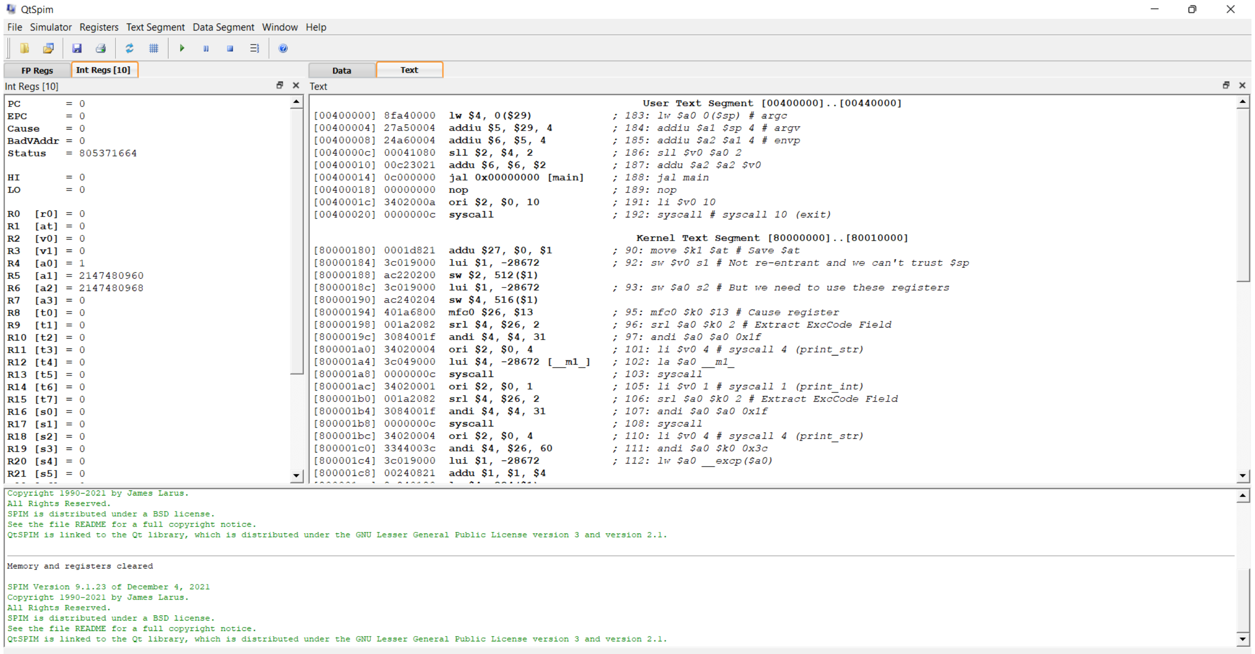
Lab6
CS211
Dynamic Memory Allocation

Name-Pranav Tambe
Roll No-2106339



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

HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 0
R2  [v0] = 0
R3  [v1] = 0
R4  [a0] = 1
R5  [a1] = 2147480960
R6  [a2] = 2147480968
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
```

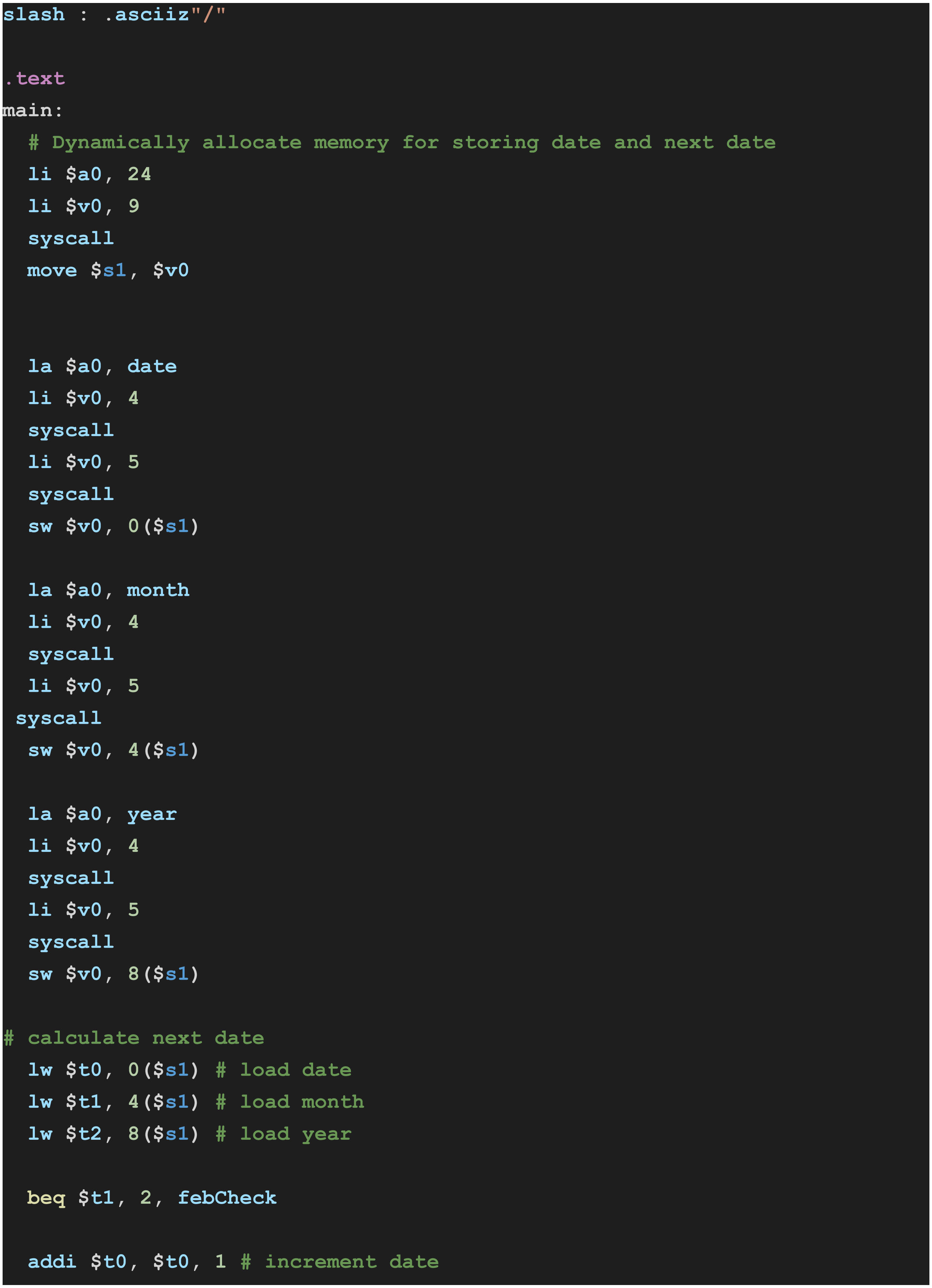
```
                                    User Text Segment [00400000]..[00440000]
[00400000] 8fa40000  lw $4, 0($29)            ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004  addiu $5, $29, 4         ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004  addiu $6, $5, 4          ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080  sll $2, $4, 2            ; 186: sll $v0 $a0 2
[00400010] 00c23021  addu $6, $6, $2          ; 187: addu $a2 $a2 $v0
[00400014] 0c000000  jal 0x00000000 [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 10 (exit)

                                   Kernel Text Segment [80000000]..[80010000]
[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
[80000188] ac220200  sw $2, 512($1)
[8000018c] 3c019000  lui $1, -28672           ; 93: sw $a0 s2 # But we need to use these registers
[80000190] ac240204  sw $4, 516($1)
[80000194] 401a6800  mfc0 $26, $13            ; 95: mfc0 $k0 $13 # Cause register
[80000198] 001a2082  srl $4, $26, 2           ; 96: srl $a0 $k0 2 # Extract ExcCode Field
[8000019c] 3084001f  andi $4, $4, 31          ; 97: andi $a0 $a0 0x1f
[800001a0] 34020004  ori $2, $0, 4            ; 101: li $v0 4 # syscall 4 (print_str)
[800001a4] 3c049000  lui $4, -28672 [__m1_]   ; 102: la $a0 __m1_
[800001a8] 0000000c  syscall                  ; 103: syscall
[800001ac] 34020001  ori $2, $0, 1            ; 105: li $v0 1 # syscall 1 (print_int)
[800001b0] 001a2082  srl $4, $26, 2           ; 106: srl $a0 $k0 2 # Extract ExcCode Field
[800001b4] 3084001f  andi $4, $4, 31          ; 107: andi $a0 $a0 0x1f
[800001b8] 0000000c  syscall                  ; 108: syscall
[800001bc] 34020004  ori $2, $0, 4            ; 110: li $v0 4 # syscall 4 (print_str)
[800001c0] 3344003c  andi $4, $26, 60         ; 111: andi $a0 $k0 0x3c
[800001c4] 3c019000  lui $1, -28672           ; 112: lw $a0 __excp($a0)
[800001c8] 00240821  addu $1, $1, $4
```

```
Copyright 1990-2021 by James Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.

Memory and registers cleared

SPIM Version 9.1.23 of December 4, 2021
Copyright 1990-2021 by James Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.
```

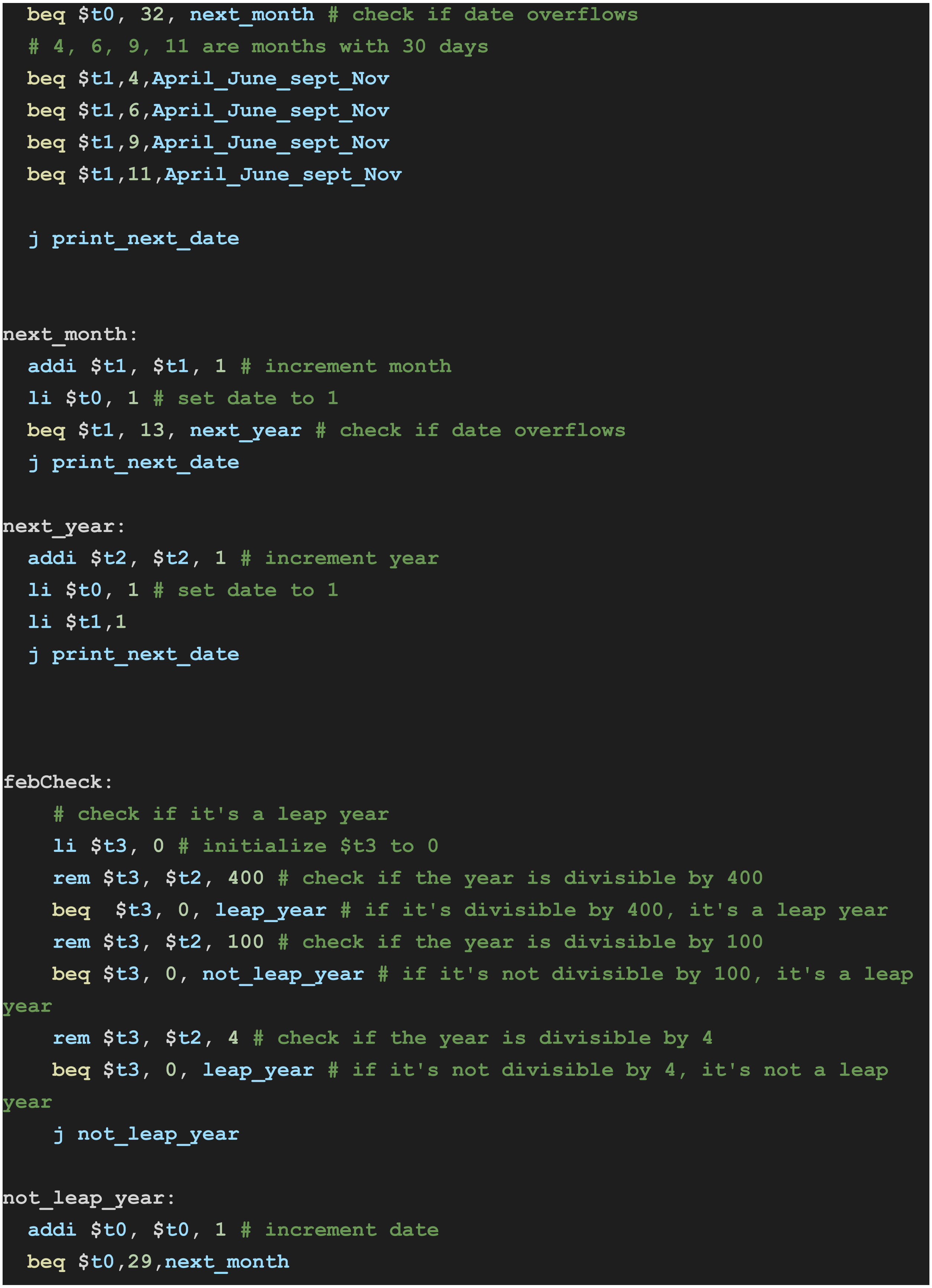
Interface before loading any file

1) Dynamically allocate memory to store a structure ‘Date’, which has three variables of integer type, namely date, month and year. The values for the same have to be entered by the user.Place the next date in next memory location and also display it on the screen.

```
# Author-Pranav Tambe
# Roll No-2106339

.data
date:  .asciiz "Enter the date : "
month:.asciiz "Enter the month : "
year  :.asciiz "Enter the year : "
next_date:  .asciiz "\nThe next date is: "
```



```
slash : .asciiz"/"

.text
main:
  # Dynamically allocate memory for storing date and next date
  li $a0, 24
  li $v0, 9
  syscall
  move $s1, $v0


  la $a0, date
  li $v0, 4
  syscall
  li $v0, 5
  syscall
  sw $v0, 0($s1)

  la $a0, month
  li $v0, 4
  syscall
  li $v0, 5
 syscall
  sw $v0, 4($s1)

  la $a0, year
  li $v0, 4
  syscall
  li $v0, 5
  syscall
  sw $v0, 8($s1)

# calculate next date
  lw $t0, 0($s1) # load date
  lw $t1, 4($s1) # load month
  lw $t2, 8($s1) # load year

  beq $t1, 2, febCheck

  addi $t0, $t0, 1 # increment date
```



```
  beq $t0, 32, next_month # check if date overflows
  # 4, 6, 9, 11 are months with 30 days
  beq $t1,4,April_June_sept_Nov
  beq $t1,6,April_June_sept_Nov
  beq $t1,9,April_June_sept_Nov
  beq $t1,11,April_June_sept_Nov

  j print_next_date


next_month:
  addi $t1, $t1, 1 # increment month
  li $t0, 1 # set date to 1
  beq $t1, 13, next_year # check if date overflows
  j print_next_date

next_year:
  addi $t2, $t2, 1 # increment year
  li $t0, 1 # set date to 1
  li $t1,1
  j print_next_date



febCheck:
    # check if it's a leap year
    li $t3, 0 # initialize $t3 to 0
    rem $t3, $t2, 400 # check if the year is divisible by 400
    beq  $t3, 0, leap_year # if it's divisible by 400, it's a leap year
    rem $t3, $t2, 100 # check if the year is divisible by 100
    beq $t3, 0, not_leap_year # if it's not divisible by 100, it's a leap
year
    rem $t3, $t2, 4 # check if the year is divisible by 4
    beq $t3, 0, leap_year # if it's not divisible by 4, it's not a leap
year
    j not_leap_year

not_leap_year:
  addi $t0, $t0, 1 # increment date
  beq $t0,29,next_month
  j print_next_date


leap_year:
  addi $t0, $t0, 1 # increment date
  beq $t0,30,next_month
  j print_next_date

April_June_sept_Nov:
  beq $t0,31,next_month
  j print_next_date

print_next_date:
  # prompt user for input
  la $a0, next_date
  li $v0, 4 # print string syscall
  syscall

  li $v0,1 # system call for printing the integer
  move $a0,$t0 # move that integer from $t1 to $a0
  sw $t0, 12($s1)
  syscall # call operating system to perform the operation

  la $a0, slash
  li $v0, 4 # print string syscall
  syscall

  li $v0,1 # system call for printing the integer
  move $a0,$t1 # move that integer from $t1 to $a0
  sw $t1, 16($s1)
  syscall # call operating system to perform the operation

  la $a0, slash
  li $v0, 4 # print string syscall
  syscall

  li $v0,1 # system call for printing the integer
  move $a0,$t2 # move that integer from $t1 to $a0
  sw $t2, 20($s1)
  syscall # call operating system to perform the operation
```

```
li $v0, 10 # System call code for exit
syscall # Exit program
```

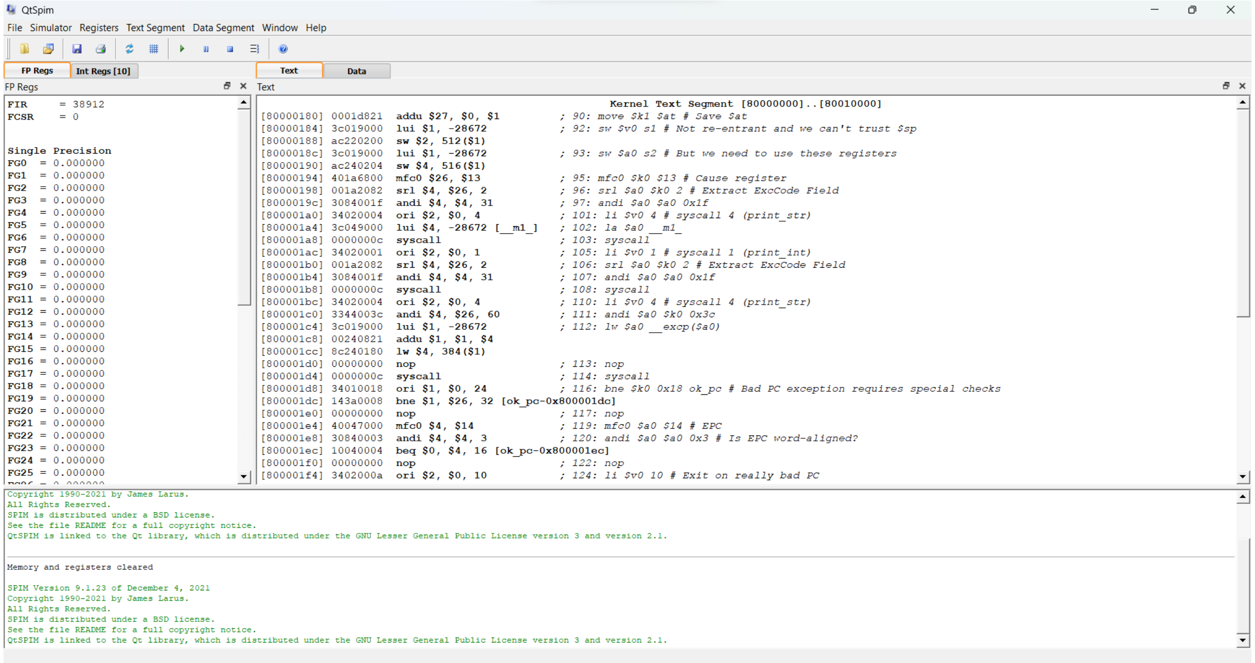
Brief overview of the code section

The program dynamically allocates memory for storing the user's input and next date. Then, it prompts the user to input the date, month, and year, and the values are stored in the allocated memory using the "sw" instruction.

The program then calculates the next date using a series of conditional statements. If the month is February, it checks whether the year is a leap year or not to determine the number of days in the month. If the month has 30 days, the program increments the date if it is not the last day of the month. If the month has 31 days, the program increments the date unless it is the last day of the month.

Finally, the program prints the next date by retrieving the values stored in memory using the "lw" instruction and prints them using the "syscall" instruction.

After loading the file

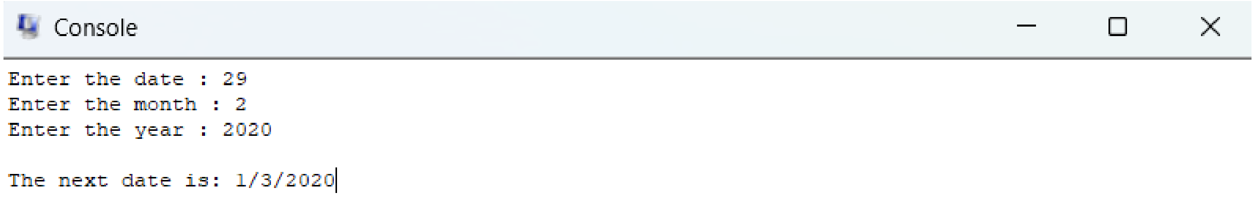


```
FIR     = 38912
FCSR    = 0


Single Precision
FG0  = 0.000000
FG1  = 0.000000
FG2  = 0.000000
FG3  = 0.000000
FG4  = 0.000000
FG5  = 0.000000
FG6  = 0.000000
FG7  = 0.000000
FG8  = 0.000000
FG9  = 0.000000
FG10 = 0.000000
FG11 = 0.000000
FG12 = 0.000000
FG13 = 0.000000
FG14 = 0.000000
FG15 = 0.000000
FG16 = 0.000000
FG17 = 0.000000
FG18 = 0.000000
FG19 = 0.000000
FG20 = 0.000000
FG21 = 0.000000
FG22 = 0.000000
FG23 = 0.000000
FG24 = 0.000000
FG25 = 0.000000
```

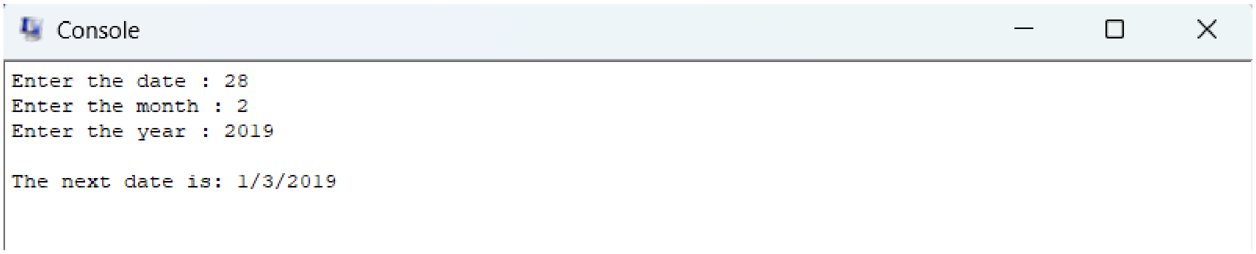
```
                                                 Kernel Text Segment [80000000]..[80010000]
[80000180] 0001d821  addu $27, $0, $1            ; 90: move $k1 $at # Save $at
[80000184] 3c019000  lui $1, -28672              ; 92: sw $v0 s1 # Not re-entrant and we can't trust $sp
[80000188] ac220200  sw $2, 512($1)
[8000018c] 3c019000  lui $1, -28672              ; 93: sw $a0 s2 # But we need to use these registers
[80000190] ac240204  sw $4, 516($1)
[80000194] 401a6800  mfc0 $26, $13               ; 95: mfc0 $k0 $13 # Cause register
[80000198] 001a2082  srl $4, $26, 2              ; 96: srl $a0 $k0 2 # Extract ExcCode Field
[8000019c] 3084001f  andi $4, $4, 31             ; 97: andi $a0 $a0 0x1f
[800001a0] 34020004  ori $2, $0, 4               ; 101: li $v0 4 # syscall 4 (print_str)
[800001a4] 3c049000  lui $4, -28672 [__m1_]      ; 102: la $a0 __m1_
[800001a8] 0000000c  syscall                     ; 103: syscall
[800001ac] 34020001  ori $2, $0, 1               ; 105: li $v0 1 # syscall 1 (print_int)
[800001b0] 001a2082  srl $4, $26, 2              ; 106: srl $a0 $k0 2 # Extract ExcCode Field
[800001b4] 3084001f  andi $4, $4, 31             ; 107: andi $a0 $a0 0x1f
[800001b8] 0000000c  syscall                     ; 108: syscall
[800001bc] 34020004  ori $2, $0, 4               ; 110: li $v0 4 # syscall 4 (print_str)
[800001c0] 3344003c  andi $4, $26, 60            ; 111: andi $a0 $k0 0x3c
[800001c4] 3c019000  lui $1, -28672              ; 112: lw $a0 __excp($a0)
[800001c8] 00240821  addu $1, $1, $4
[800001cc] 8c240180  lw $4, 384($1)
[800001d0] 00000000  nop                         ; 113: nop
[800001d4] 0000000c  syscall                     ; 114: syscall
[800001d8] 34010018  ori $1, $0, 24              ; 116: bne $k0 0x18 ok_pc # Bad PC exception requires special checks
[800001dc] 143a0008  bne $1, $26, 32 [ok_pc-0x800001dc]
[800001e0] 00000000  nop                         ; 117: nop
[800001e4] 40047000  mfc0 $4, $14                ; 119: mfc0 $a0 $14 # EPC
[800001e8] 30840003  andi $4, $4, 3              ; 120: andi $a0 $a0 0x3 # Is EPC word-aligned?
[800001ec] 10040004  beq $0, $4, 16 [ok_pc-0x800001ec]
[800001f0] 00000000  nop                         ; 122: nop
[800001f4] 3402000a  ori $2, $0, 10              ; 124: li $v0 10 # Exit on really bad PC
```

```
Copyright 1990-2021 by James Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.

Memory and registers cleared

SPIM Version 9.1.23 of December 4, 2021
Copyright 1990-2021 by James Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.
```

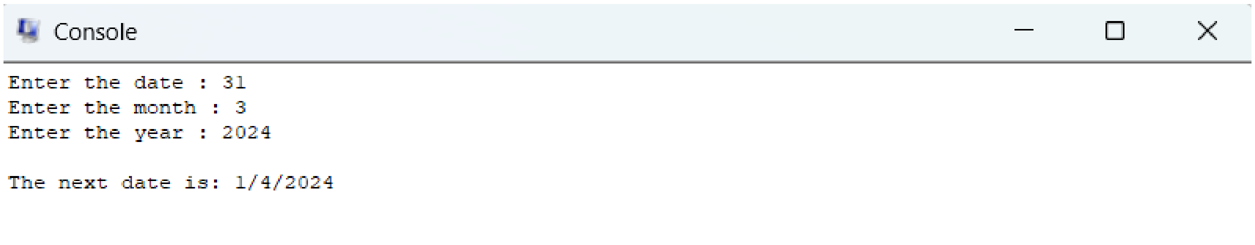
Console



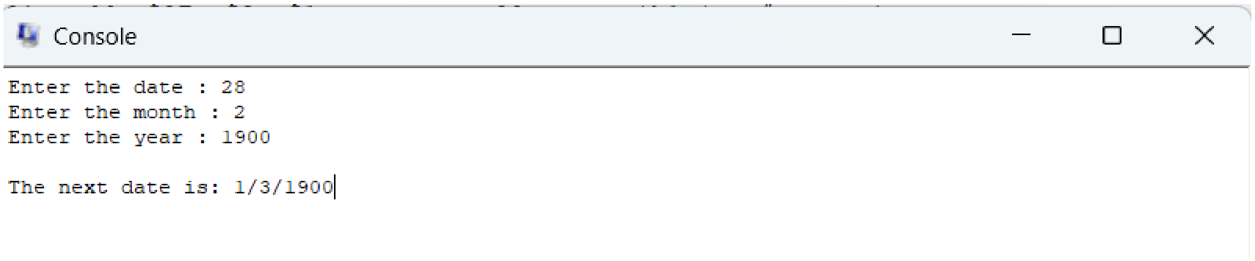
```
Enter the date : 29
Enter the month : 2
Enter the year : 2020

The next date is: 1/3/2020
```



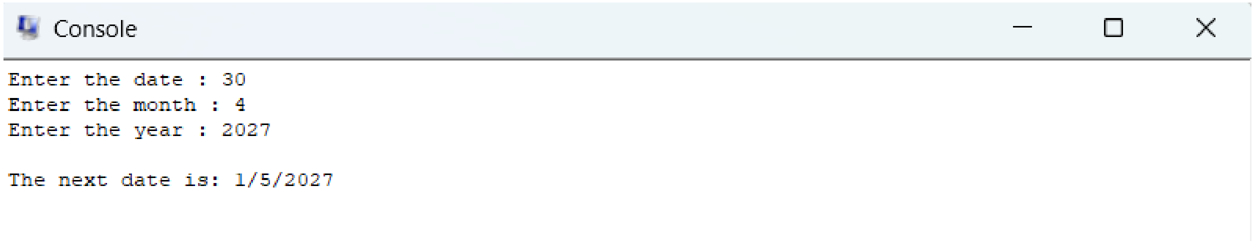
```
Enter the date : 28
Enter the month : 2
Enter the year : 2019

The next date is: 1/3/2019
```



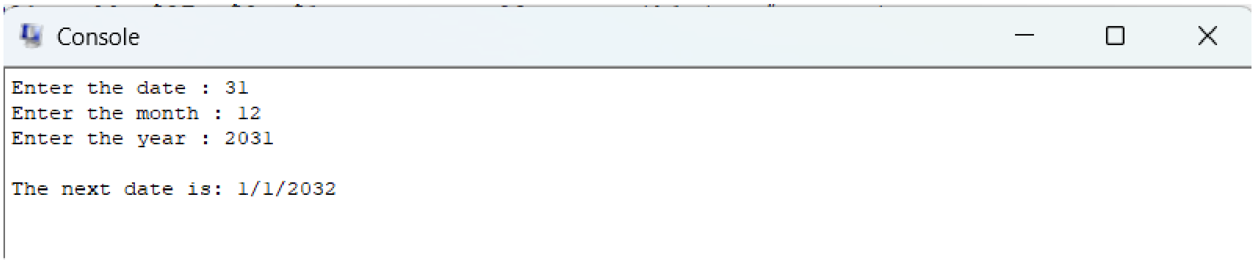
```
Enter the date : 31
Enter the month : 3
Enter the year : 2024

The next date is: 1/4/2024
```



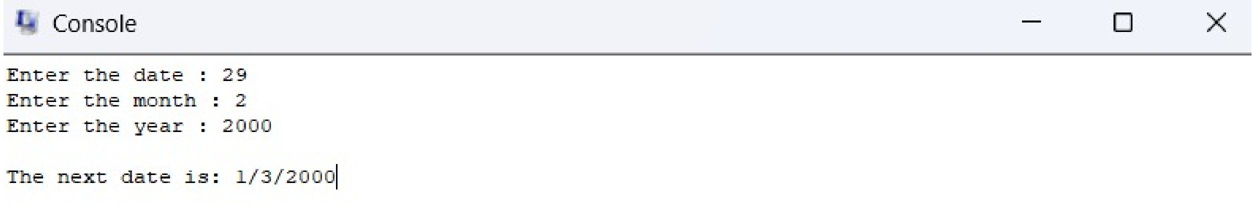
```
Enter the date : 28
Enter the month : 2
Enter the year : 1900

The next date is: 1/3/1900
```



```
Enter the date : 30
Enter the month : 4
Enter the year : 2027

The next date is: 1/5/2027
```

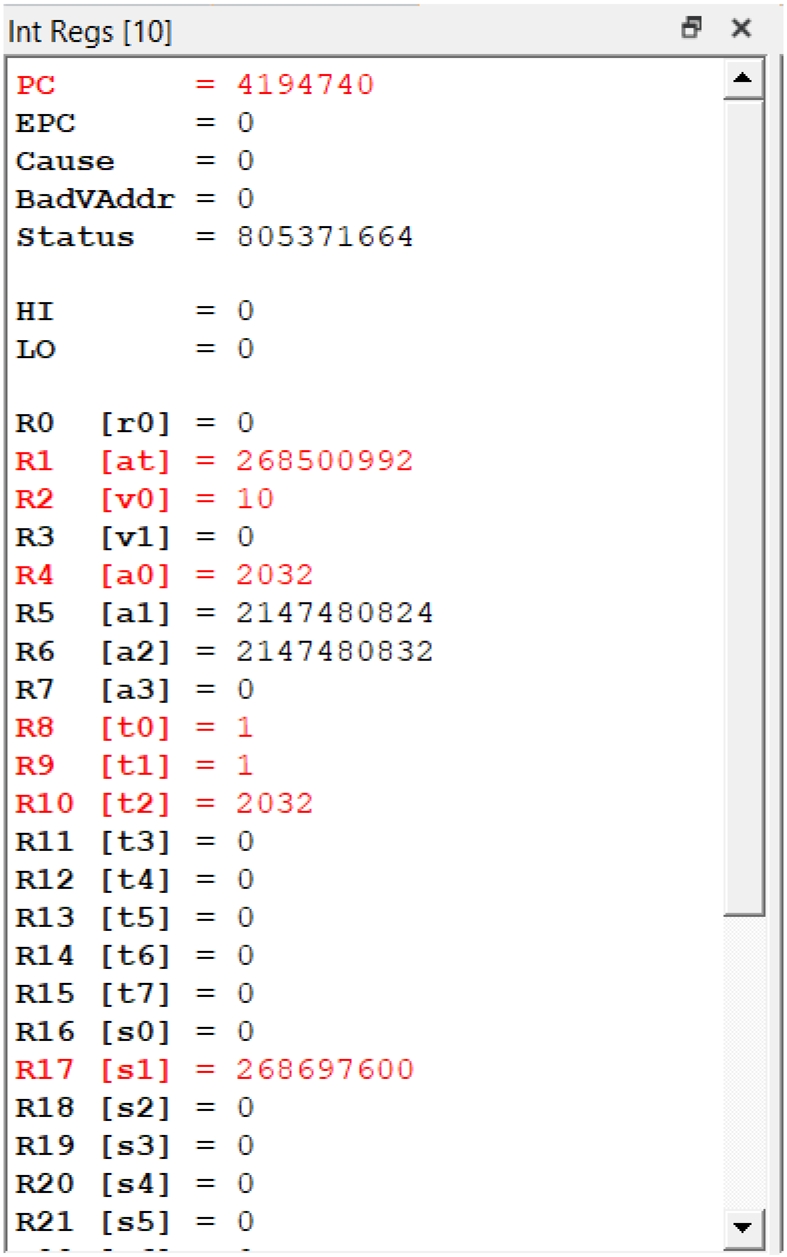


```
Enter the date : 31
Enter the month : 12
Enter the year : 2031

The next date is: 1/1/2032
```



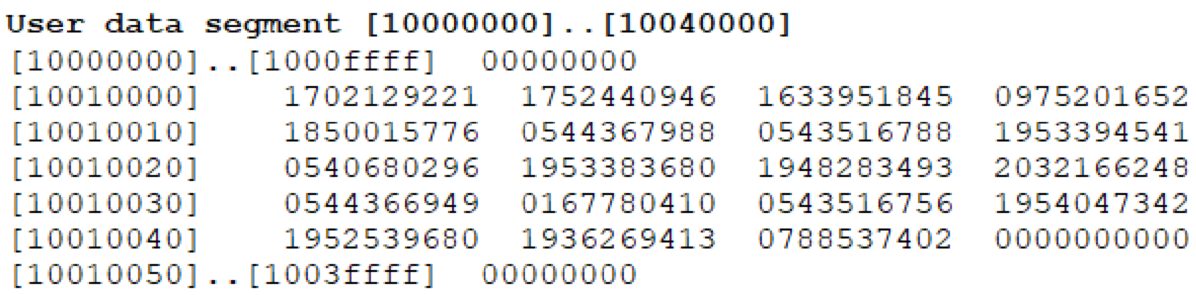
```
Enter the date : 29
Enter the month : 2
Enter the year : 2000

The next date is: 1/3/2000
```

Registers after execution of the code



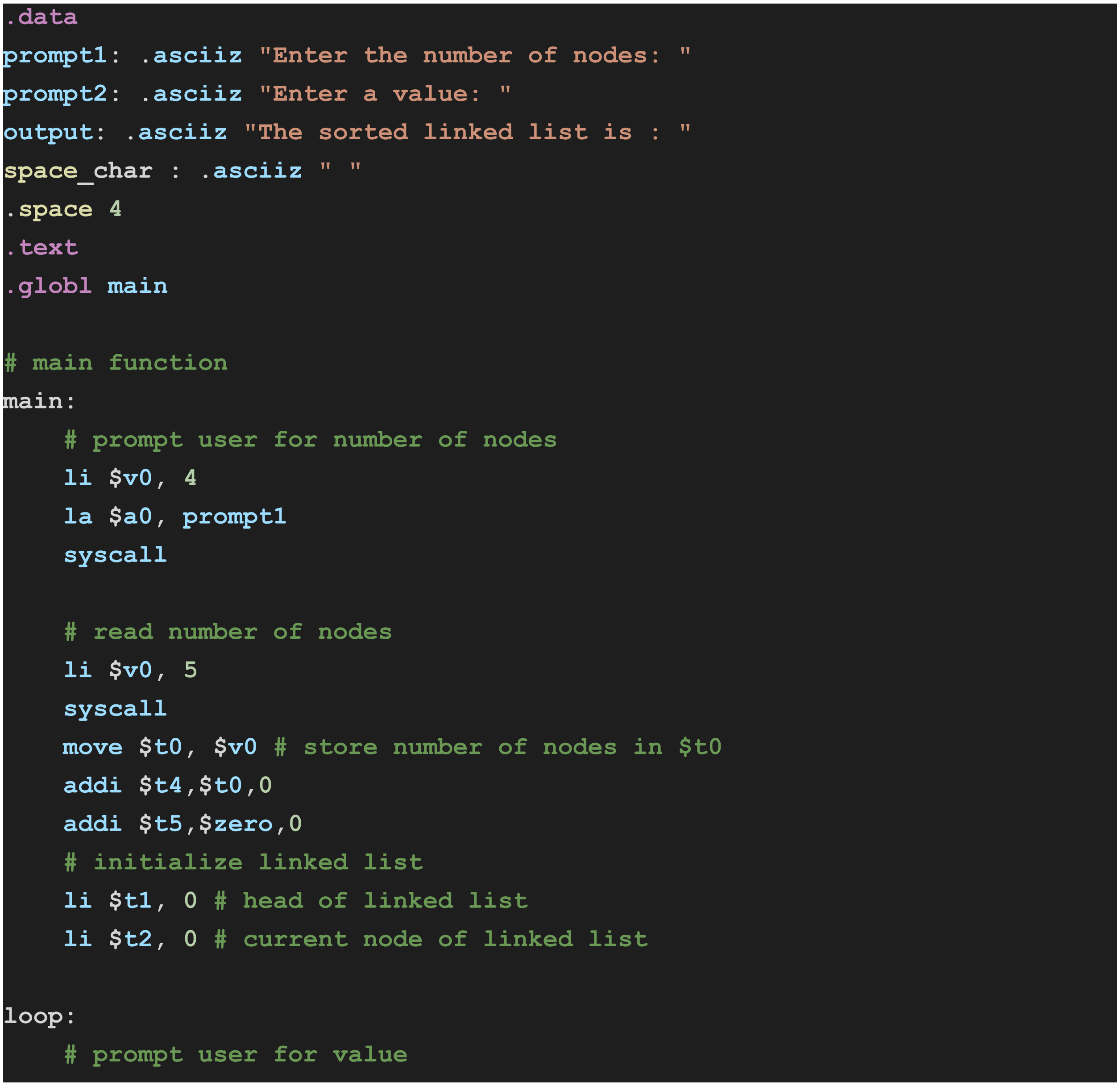
```
PC          = 4194740
EPC         = 0
Cause       = 0
BadVAddr    = 0
Status      = 805371664

HI          = 0
LO          = 0

R0   [r0]   = 0
R1   [at]   = 268500992
R2   [v0]   = 10
R3   [v1]   = 0
R4   [a0]   = 2032
R5   [a1]   = 2147480824
R6   [a2]   = 2147480832
R7   [a3]   = 0
R8   [t0]   = 1
R9   [t1]   = 1
R10  [t2]   = 2032
R11  [t3]   = 0
R12  [t4]   = 0
R13  [t5]   = 0
R14  [t6]   = 0
R15  [t7]   = 0
R16  [s0]   = 0
R17  [s1]   = 268697600
R18  [s2]   = 0
R19  [s3]   = 0
R20  [s4]   = 0
R21  [s5]   = 0
```

Dynamic memory allocation



```
User data segment [10000000]..[10040000]
[10000000]..[1000ffff]  00000000
[10010000]    1702129221  1752440946  1633951845  0975201652
[10010010]    1850015776  0544367988  0543516788  1953394541
[10010020]    0540680296  1953383680  1948283493  2032166248
[10010030]    0544366949  0167780410  0543516756  1954047342
[10010040]    1952539680  1936269413  0788537402  0000000000
[10010050]..[1003ffff]  00000000
```

Q.2) Create a linked list of integers entered by user. Sort the list and print the same. Take user input initially for the total number of nodes.

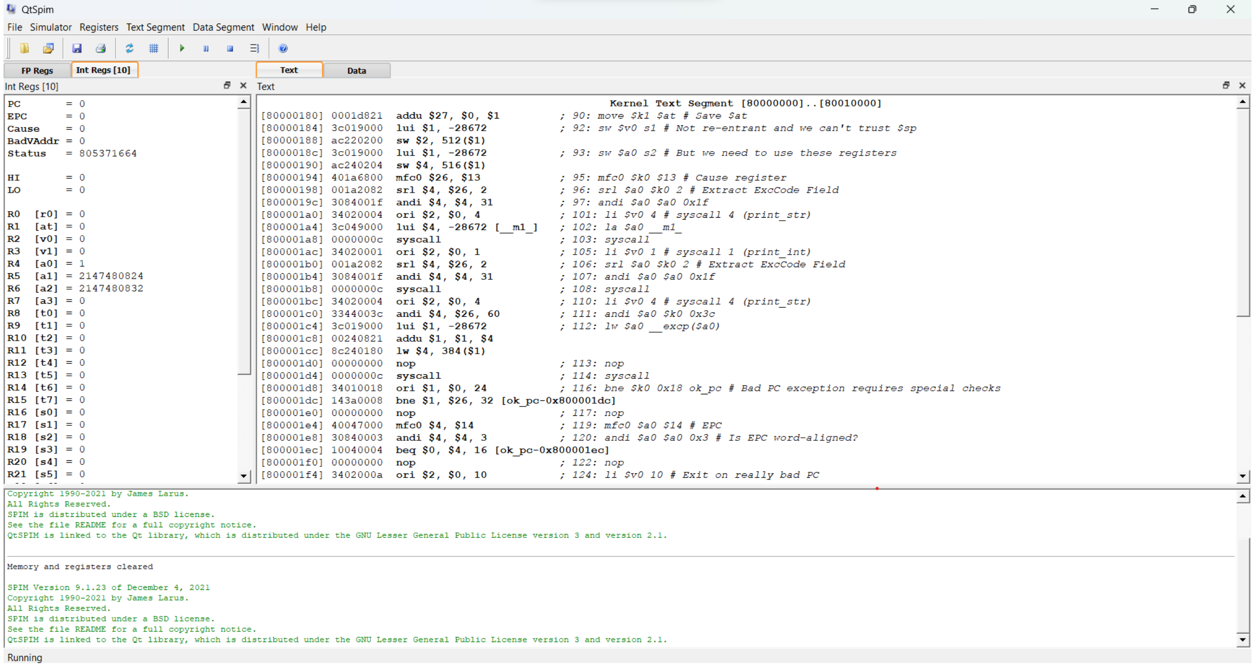


```
.data
prompt1: .asciiz "Enter the number of nodes: "
prompt2: .asciiz "Enter a value: "
output: .asciiz "The sorted linked list is : "
space_char : .asciiz " "
.space 4
.text
.globl main

# main function
main:
    # prompt user for number of nodes
    li $v0, 4
    la $a0, prompt1
    syscall

    # read number of nodes
    li $v0, 5
    syscall
    move $t0, $v0 # store number of nodes in $t0
    addi $t4,$t0,0
    addi $t5,$zero,0
    # initialize linked list
    li $t1, 0 # head of linked list
    li $t2, 0 # current node of linked list

loop:
    # prompt user for value
    li $v0, 4
    la $a0, prompt2
    syscall

    # read value
    li $v0, 5
    syscall
    move $t3, $v0 # store value in $t3

    # create new node
    li $v0, 9 # allocate memory for a new node
    li $a0, 8 # size of node is 8 bytes (4 for value, 4 for pointer)
    syscall
    sw $t3, ($v0) # store value in node
    sw $t1, 4($v0) # store address of next node in node
    move $t1, $v0
    addi $t0,$t0,-1
    # add node to linked list
    beq $t0, $zero, end_loop # if head of linked list is null, set head to new node
    j loop
end_loop:
    # print linked list
    li $v0, 4
    la $a0, output
    syscall
    la $t2, 0($t1) # start at head of linked list
    addi $t6,$zero,1
bubble_sort_outer_loop:
    beq $t6,$t4,print_loop # exit the function if $t6 == $t4
    la $t2, 0($t1) # load the address of the head of the linked list into $t2
    addi $t6,$t6,1 # increment the outer loop counter ($t6)

bubble_sort_inner_loop:
    lw $t7,0($t2) # load the value at the current node of the linked list into $t7
    lw $t8,4($t2) # load the address of the next node in the linked list into $t8
    beq $t8,$0,bubble_sort_outer_loop # if the next node is NULL, exit the inner loop
    lw $t9,($t8) # load the value at the next node into $t9
    slt $s0,$t7,$t9 # if $t7<$t8, set $s0 to 1, otherwise set it to 0
    beq $s0,$0,swap_function # if $t7>=$t8, skip the swap_function and continue with the inner loop
    addi $s1, $s1,-1 # decrement the inner loop counter ($s1)
    lw $t2, 4($t2) # move to the next node in the linked list
    j bubble_sort_inner_loop # jump back to the beginning of the inner loop
swap_function:
    sw $t9,($t2) # store the value of the next node into the current node
    sw $t7,($t8) # store the value of the current node into the next node
    addi $s1, $s1,1 # increment the inner loop counter
    lw $t2, 4($t2) # move to the next node in the linked list
    j bubble_sort_inner_loop # jump back to the beginning of the inner loop
print_loop:
    beq $t4,$t5,end_print_loop # exit the function if $t4 == $t5
    lw $a0, ($t1) # load the value at the current node of the linked list into $a0
    li $v0, 1 # load the print integer system call code into $v0
    syscall # print the integer value in $a0
    li $v0, 4 # load the print string system call code into $v0
    la $a0,space_char # load the address of the space character into $a0
    syscall # print a space character
    addi $t5,$t5,1 # increment the print loop counter ($t5)
    lw $t1, 4($t1) # move to the next node in the linked list
    j print_loop # jump back to the beginning of the print loop

end_print_loop:
    li $v0, 10 # load the exit system call code into $v0
    syscall # exit the program
```

Brief overview of the code section

program prompts the user to enter the number of nodes for a linked list, reads the values, and sorts the linked list using the bubble sort algorithm. The program then prints the sorted linked list to the console. The main function first prompts the user for the number of nodes and reads the value. It then creates a new node and adds it to the linked list. The program then enters a loop to prompt the user for more values until the linked list is complete. After creating the linked list, the program enters the bubble sort algorithm to sort the linked list. It uses two nested loops to iterate over the linked list and compare each node to its next node. If a node is greater than its next node, the nodes are swapped. After sorting the linked list, the program enters a loop to print the values of the sorted linked list to the console. Finally, the program exits.

After loading the file

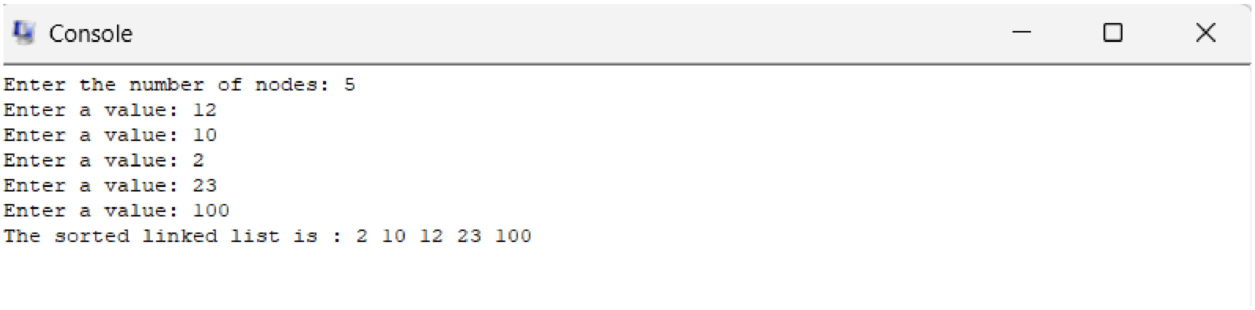


```
PC       = 0
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 805371664

HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 0
R2  [v0] = 0
R3  [v1] = 0
R4  [a0] = 1
R5  [a1] = 2147480824
R6  [a2] = 2147480832
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
```

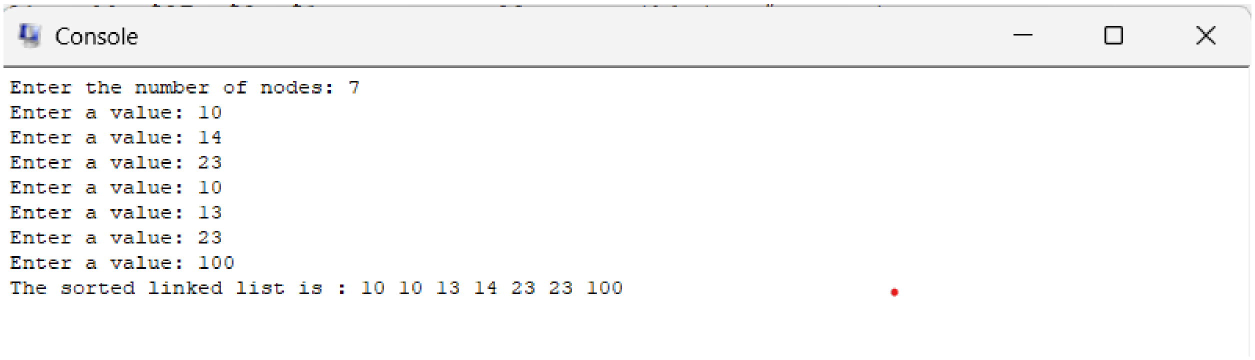
```
                                   Kernel Text Segment [80000000]..[80010000]
[80000180] 0001d821  addu $27, $0, $1          ; 90: move $k1 $at # Save $at
[80000184] 3c019000  lui $1, -28672            ; 92: sw $v0 s1 # Not re-entrant and we can't trust $sp
[80000188] ac220200  sw $2, 512($1)
[8000018c] 3c019000  lui $1, -28672            ; 93: sw $a0 s2 # But we need to use these registers
[80000190] ac240204  sw $4, 516($1)
[80000194] 401a6800  mfc0 $26, $13             ; 95: mfc0 $k0 $13 # Cause register
[80000198] 001a2082  srl $4, $26, 2            ; 96: srl $a0 $k0 2 # Extract ExcCode Field
[8000019c] 3084001f  andi $4, $4, 31           ; 97: andi $a0 $a0 0x1f
[800001a0] 34020004  ori $2, $0, 4             ; 101: li $v0 4 # syscall 4 (print_str)
[800001a4] 3c049000  lui $4, -28672 [__m1_]    ; 102: la $a0 __m1_
[800001a8] 0000000c  syscall                   ; 103: syscall
[800001ac] 34020001  ori $2, $0, 1             ; 105: li $v0 1 # syscall 1 (print_int)
[800001b0] 001a2082  srl $4, $26, 2            ; 106: srl $a0 $k0 2 # Extract ExcCode Field
[800001b4] 3084001f  andi $4, $4, 31           ; 107: andi $a0 $a0 0x1f
[800001b8] 0000000c  syscall                   ; 108: syscall
[800001bc] 34020004  ori $2, $0, 4             ; 110: li $v0 4 # syscall 4 (print_str)
[800001c0] 3344003c  andi $4, $26, 60          ; 111: andi $a0 $k0 0x3c
[800001c4] 3c019000  lui $1, -28672            ; 112: lw $a0 __excp($a0)
[800001c8] 00240821  addu $1, $1, $4
[800001cc] 8c240180  lw $4, 384($1)
[800001d0] 00000000  nop                       ; 113: nop
[800001d4] 0000000c  syscall                   ; 114: syscall
[800001d8] 34010018  ori $1, $0, 24            ; 116: bne $k0 0x18 ok_pc # Bad PC exception requires special checks
[800001dc] 143a0008  bne $1, $26, 32 [ok_pc-0x800001dc]
[800001e0] 00000000  nop                       ; 117: nop
[800001e4] 40047000  mfc0 $4, $14              ; 119: mfc0 $a0 $14 # EPC
[800001e8] 30840003  andi $4, $4, 3            ; 120: andi $a0 $a0 0x3 # Is EPC word-aligned?
[800001ec] 10040004  beq $0, $4, 16 [ok_pc-0x800001ec]
[800001f0] 00000000  nop                       ; 122: nop
[800001f4] 3402000a  ori $2, $0, 10            ; 124: li $v0 10 # Exit on really bad PC
```

```
Copyright 1990-2021 by James Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.

Memory and registers cleared

SPIM Version 9.1.23 of December 4, 2021
Copyright 1990-2021 by James Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.
```

Console images

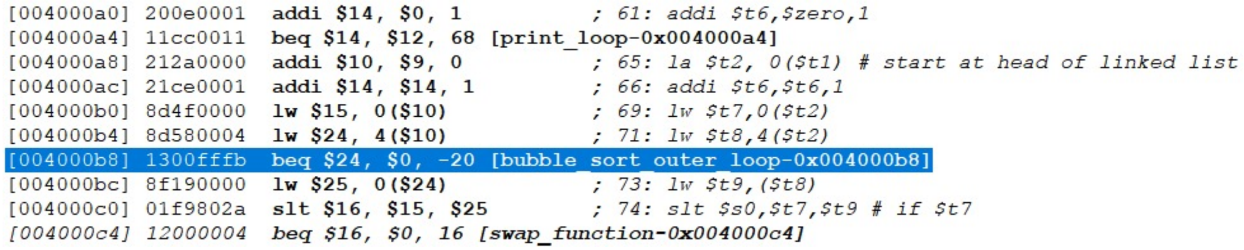


```
Enter the number of nodes: 5
Enter a value: 12
Enter a value: 10
Enter a value: 2
Enter a value: 23
Enter a value: 100
The sorted linked list is : 2 10 12 23 100
```

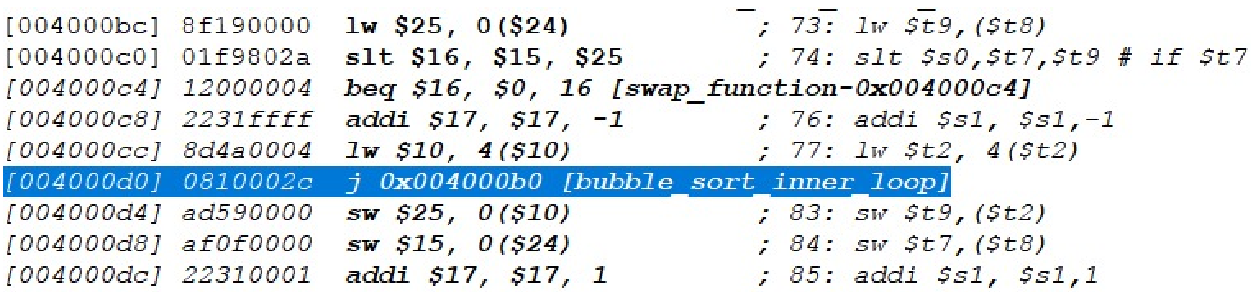


```
Enter the number of nodes: 7
Enter a value: 10
Enter a value: 14
Enter a value: 23
Enter a value: 10
Enter a value: 13
Enter a value: 23
Enter a value: 100
The sorted linked list is : 10 10 13 14 23 23 100
```

Bubble_sort_loops:

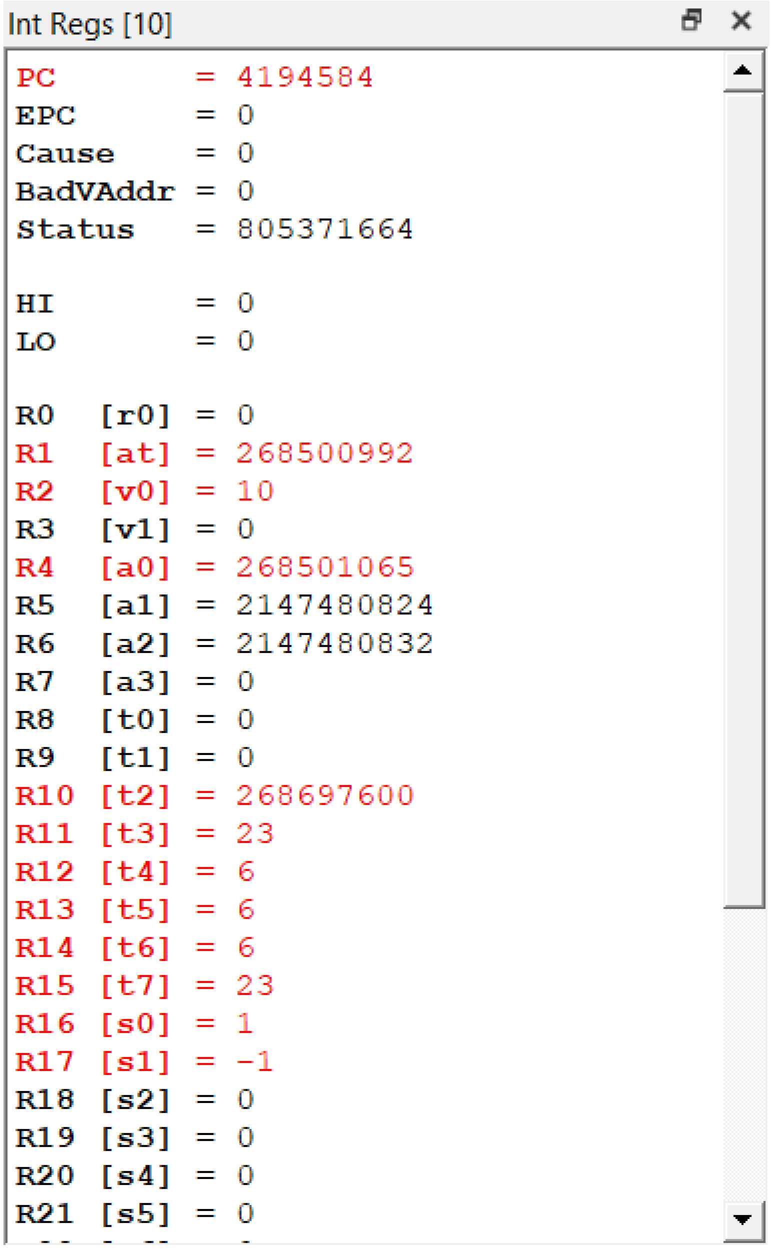


```
[004000a0] 200e0001  addi $14, $0, 1          ; 61: addi $t6,$zero,1
[004000a4] 11cc0011  beq $14, $12, 68 [print_loop-0x004000a4]
[004000a8] 212a0000  addi $10, $9, 0          ; 65: la $t2, 0($t1) # start at head of linked list
[004000ac] 21ce0001  addi $14, $14, 1         ; 66: addi $t6,$t6,1
[004000b0] 8d4f0000  lw $15, 0($10)           ; 69: lw $t7,0($t2)
[004000b4] 8d580004  lw $24, 4($10)           ; 71: lw $t8,4($t2)
[004000b8] 1300fffb  beq $24, $0, -20 [bubble_sort_outer_loop-0x004000b8]
[004000bc] 8f190000  lw $25, 0($24)           ; 73: lw $t9,($t8)
[004000c0] 01f9802a  slt $16, $15, $25        ; 74: slt $s0,$t7,$t9 # if $t7
[004000c4] 12000004  beq $16, $0, 16 [swap_function-0x004000c4]
```



```
[004000bc] 8f190000  lw $25, 0($24)            ; 73: lw $t9,($t8)
[004000c0] 01f9802a  slt $16, $15, $25         ; 74: slt $s0,$t7,$t9 # if $t7
[004000c4] 12000004  beq $16, $0, 16 [swap_function-0x004000c4]
[004000c8] 2231ffff  addi $17, $17, -1         ; 76: addi $s1, $s1,-1
[004000cc] 8d4a0004  lw $10, 4($10)            ; 77: lw $t2, 4($t2)
[004000d0] 0810002c  j 0x004000b0 [bubble_sort_inner_loop]
[004000d4] ad590000  sw $25, 0($10)            ; 83: sw $t9,($t2)
[004000d8] af0f0000  sw $15, 0($24)            ; 84: sw $t7,($t8)
[004000dc] 22310001  addi $17, $17, 1          ; 85: addi $s1, $s1,1
```

Registers after execution of the code



```
PC        = 4194584
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] = 268501065
R5   [a1] = 2147480824
R6   [a2] = 2147480832
R7   [a3] = 0
R8   [t0] = 0
R9   [t1] = 0
R10  [t2] = 268697600
R11  [t3] = 23
R12  [t4] = 6
R13  [t5] = 6
R14  [t6] = 6
R15  [t7] = 23
R16  [s0] = 1
R17  [s1] = -1
R18  [s2] = 0
R19  [s3] = 0
R20  [s4] = 0
R21  [s5] = 0
```