Department of Computer Science
Ashoka University

Computer Organization and Systems

Assignment 4

Name: Rushil Gupta and Dhruman Gupta

Problem 1: Sum of Numbers from 1 to N

Pseudocode

```
Prompt user for a positive integer N (<1000)
Read N
Set sum = 0, i = 1
While i <= N:
    sum = sum + i
    i = i + 1
Print "The Sum (S) is equal to: " and sum
```

MIPS Code

```
    .data
    prompt:      .asciiz "Enter a positive integer (<1000): "
    resultStr:   .asciiz "The Sum (S) is equal to: "

    .text
    main:
        # Print prompt message
        li   $v0, 4
        la   $a0, prompt
        syscall

        # Read integer input into register $t0 (N)
        li   $v0, 5 # We load into $v0 since it is a special register
        syscall
        move $t0, $v0      # $t0 holds N

        # Initialize sum = 0 and counter i = 1
        li   $t1, 0        # $t1 will hold the sum
        li   $t2, 1        # $t2 is our loop counter i

    sum_loop:
        bgt  $t2, $t0, sum_done # If i = $t2 > $t1 = N, then we are done with the loop
        add  $t1, $t1, $t2
        addi $t2, $t2, 1
        j    sum_loop

    sum_done:
        li   $v0, 4
        la   $a0, resultStr # Load the address of the result string
        syscall

        li   $v0, 1
        move $a0, $t1 # Load the sum into $a0
        syscall
```

Output

```
    Enter a positive integer (<1000): 5
    The Sum (S) is equal to: 15
    -- program is finished running (dropped off bottom) --
```

Problem 2: GCD by Repeated Subtraction

Pseudocode

```
Prompt user for two positive integers A and B
Read A and B
If A <= 0 or B <= 0, then print an error message and exit
While A != B:
    If A > B, then A = A - B
    Else, B = B - A
GCD is A # (or B since A = B)
Print "The GCD is: " and the result
```

MIPS Code

```
.data
promptA:    .asciiz "Enter the first positive integer: "
promptB:    .asciiz "Enter the second positive integer: "
error_msg:  .asciiz "Please enter a positive number"
gcdStr:     .asciiz "The GCD is: "

.text
main:
    # $v0 = 4 is for printing a string, here we ask the user for the first integer
    li   $v0, 4
    la   $a0, promptA
    syscall

    # $v0 = 5 is for reading an integer, here we read the first integer
    li   $v0, 5
    syscall
    move $t0, $v0      # A

    blez $t0, error    # If A <= 0, then we print an error message

    # Similar to the above, we ask the user for the second integer
    li   $v0, 4
    la   $a0, promptB
    syscall

    # Similar to the above, we read the second integer
    li   $v0, 5
    syscall
    move $t1, $v0      # B

    blez $t1, error    # If B <= 0, then we print an error message


gcd_loop:
    beq  $t0, $t1, gcd_done # If A = B, then we are done
    bgt  $t0, $t1, sub_A # If A > B, then we subtract B from A and go top of loop
    sub  $t1, $t1, $t0 # If B > A, then we subtract A from B
    j    gcd_loop

sub_A:
    sub  $t0, $t0, $t1
    j    gcd_loop


error:
    # Print the string "Please enter a positive number"
    li   $v0, 4
    la   $a0, error_msg
    syscall
    # Exit the program
    li  $v0, 10
    syscall

gcd_done:
    # Only here when A = B = gcd(A, B)

    # Print the string "The GCD is: "
    li   $v0, 4
    la   $a0, gcdStr
    syscall

    # Print the GCD
    li   $v0, 1
    move $a0, $t0
    syscall
```

Output

```
Enter the first positive integer: 89
Enter the second positive integer: 13
The GCD is: 1
-- program is finished running (dropped off bottom) --

Enter the first positive integer: 105
Enter the second positive integer: 35
The GCD is: 35
-- program is finished running (dropped off bottom) --

Enter the first positive integer: 2
Enter the second positive integer: -3
Please enter a positive number
-- program is finished running --
```

Problem 3: N-th Fibonacci Number

Pseudocode

```
Prompt user for a positive integer N (>2)
Read N
If N < 2, print an error message and exit
Initialize f0 = 0, f1 = 1, and counter i = 3
While i <= N:
    f2 = f0 + f1
    Set f0 = f1 and f1 = f2
    Increment i by 1
Print "The Fibonacci number is: " and f1
```

MIPS Code

```
.data
promptFib:  .asciiz "Enter a positive integer (>= 2) for Fibonacci: "
fibStr:     .asciiz "The Fibonacci number is: "
errorStr:  .asciiz "Please enter a number >= 2"

.text
main:
    li   $v0, 4
    la   $a0, promptFib
    syscall

    li   $v0, 5
    syscall
    move $t0, $v0      # N

    blt $t0, 2, fib_error # If N < 2, then we print an error message
    # Although the assignment says check for N < 0, we asked for N >= 2

    li   $t1, 0        # f0
    li   $t2, 1        # f1
    li   $t3, 3        # counter

fib_loop:
    bgt  $t3, $t0, fib_done # i > N => done
    add  $t4, $t1, $t2  # f2 = f0 + f1
    move $t1, $t2      # f0 = f1
    move $t2, $t4      # f1 = f2
    addi $t3, $t3, 1   # i = i + 1
    j    fib_loop

fib_error:
    li   $v0, 4
    la   $a0, errorStr
    syscall

    li  $v0, 10
    syscall

fib_done:
    li   $v0, 4
    la   $a0, fibStr
    syscall

    li   $v0, 1
    move $a0, $t2
    syscall
```

Output

```
Enter a positive integer (>= 2) for Fibonacci: -3
Please enter a number >= 2
-- program is finished running --

Enter a positive integer (>= 2) for Fibonacci: 10
The Fibonacci number is: 34
-- program is finished running (dropped off bottom) --
```

Problem 4: 32-bit 2's Complement Hexadecimal Representation

Pseudocode

```
Prompt user for an integer
Read integer N
Print "The 32-bit hexadecimal representation is: 0x"
Output N in hexadecimal format using syscall 34
```

MIPS Code

```
.data
prompt:  .asciiz "Enter an integer N: "
resultStr: .asciiz "The 32-bit 2's complement hexadecimal representation of N is: 0x"

.text
main:
    li   $v0, 4
    la   $a0, prompt
    syscall

    li   $v0, 5
    syscall
    move $t0, $v0      # N


    # Sanity check: N should be representable in 32 bits
    # But note, that since N is read from syscall 5,
    # it is already in the range of 32-bit signed integers
    # Otherwise, MIPS will give a out of bounds error

    # Conclusion: MIPS already has this worked out!
    li $v0, 34 # The syscall to output int as hex
    move $a0, $t0
    syscall

    li $v0, 10
    syscall
```

Output

```
    Enter an integer N: 10
    0x0000000a
    -- program is finished running --

    Enter an integer N: -15
    0xfffffff1
    -- program is finished running --

    Enter an integer N: 256
    0x00000100
    -- program is finished running --
```

Sources for Question 4

The syscall 34 was used to output the integer in hexadecimal format. The documentation for this syscall seems to be discontinued, but the GitHub repository of the website has the table of syscalls. The link to the GitHub repository is here. Specifically, the table of syscalls is here.