Module Seven Assignment Summer 2017

Lottery Jackpot Probabilities and Odds

There are many lottery schemes available across the world. Write a MIPS Assembly language program that will calculate the odds of winning the Jackpot prize.

For example:

- The **Euro Millions** lottery requires matching 5 numbers from a pool of 50 numbers and matching 2 numbers from a second pool of 11 numbers.
- The **Mega Millions** lottery requires matching 5 from 75 and 1 from 15.
- The **Thunderball** lottery requires matching 5 from 39 and matching 1 from 14.
- The **PowerBall** lottery requires matching 5 from 69 and 1 from 26.

The subroutine example on the following page, calculates the Factorial of an input integer. Starting with the code in the example, write a correct program in MIPS - QtSpim assembly language that:

1) Calculates the odds of winning lottery jackpot grand prizes.

2) The calculated value is to be displayed on the QtSpim console screen with an appropriate commentary text. Such as "The odds are 1 in nnnn."

3) The program is to accept as input, four values:
   - An integer representing the large pool of possible numbers.
   - A second integer representing the count of numbers to be selected from the large pool.
   - An integer representing the size of the second smaller pool of numbers.
   - A fourth integer representing the count of numbers to be selected from the second pool.

4) Test your program by calculating the odds of choosing a set of 8 numbers from a pool of 15 numbers, and 1 from 2, small pool. The value is "1 in 12870". Test other combinations.

Note: Because the largest MIPS single precision integer value will not hold the value of more than 12!, you will need to use some algebra to simplify the calculations. With the simplifications, all of the math can be done using the integer multiply and divide instructions.

Your program should display a message when it stops.

Use the System Service calls on page A-44 of the textbook for the input and output.

The work products of this assignment are:

1) A copy of the source program text file. (.txt; or .asm; or .s)
2) Screen captures showing test output results.

[150 points]

```
#########  Factorial Subroutine  Fall 2016
#
#   Given n, in register $a0;
#   calculate n!, store and return the result in register $v0

factrl:    sw   $ra, 4($sp)    # save the return address
           sw   $a0, 0($sp)    # save the current value of n
           addi $sp, $sp, -8   # move stack pointer
           slti $t0, $a0, 2    # save 1 iteration, n=0 or n=1; n!=1
           beq  $t0, $zero, L1 # not less than 2, calculate n(n-1)!
           addi $v0, $zero, 1  # n=1; n!=1
           jr   $ra            # now multiply

L1:        addi $a0, $a0, -1   # n = n-1

           jal factrl          # now (n-1)!

           addi $sp, $sp, 8    # reset the stack pointer
           lw   $a0, 0($sp)    # fetch saved (n-1)
           lw   $ra, 4($sp)    # fetch return address
           mul  $v0, $a0, $v0  # multiply (n)*(n-1)
           jr   $ra            # return value n!

# P Snyder 14 August 2016
#########   End of the subroutine
```