Chris Carr

Manchester metropolitian university

1CWK50 – part 4

# Flowchart

Generate random number

(1-6)

Output random number

Increment counter

Add random number to sum of numbers

Has counter reached 8?

Output sum of numbers

**NO**

**Yes**

Registers used: $v0, $a0, $a1

Instructions used: li, la, addi, syscall

Registers used: $t0

Instructions used: addi

Registers used: $v0

Instructions used: li, syscall

Registers used: $t1, $a0

Instructions used: add

Registers used: $t0

Instructions used: bne

Registers used: $a0, $t1, $v0

Instructions used: move, li, syscall

Registers used: $v0

Instructions used: li, syscall

# Discussion

In Task D, basic loop and function (called “branching” in MiPS) procedures were implemented. There are multiple ways to implement loops within MiPS. The methodology which was implemented into this particular program was using an instruction call “bne”, which stands for “Branch if Not Equal”. The program uses an iterative counter and the “bne” statement to determine if the dice has been rolled 8 times; after the dice has been rolled 8 times, the sum is outputted. Another methodology of fulfilling the objectives of this program would be to use the “jump” instruction in conjunction with the “$ra” register. Both achieve the same result, however the “branching” functionality is a lot easier to implement.

# Code

**.data**

**seed:** **.asciiz** "Machine code is awesome, so is Chris!"

**sum\_text:** **.asciiz** "Sum of all numbers = "

**newline:** **.asciiz** "\n"

**.text**

**li** $t0, 0 #counter

**li** $t1, 0 #value of all added sums

#random generator setup

## Set Seed ##

**li** $v0, 40

**li** $a0, 1 # id of random # generator

**la** $a1, **seed** # set seed for generator

**syscall**

**random\_int\_loop:**

**addi** $t0, $t0, 1 #increment counter

## Generate random number ##

**li** $v0, 42

**li** $a0, 1 #set id of random # generator

**li** $a1, 5 #set upper bound to 5

**syscall**

## Add 1 to generated number - dice never roll 0 ##

**addi** $a0, $a0, 1

## Print the number we just generated ##

**li** $v0, 1

**syscall**

## Add the number we just generated to our TOTAL value ##

**add** $t1, $t1, $a0

## Print newline ##

**la** $a0, **newline**

**li** $v0, 4

**syscall**

## Check if rolled 8 times, if not, roll again ##

**bne** $t0, 8, **random\_int\_loop**

## Print Sum of All numbers ##

#print sum text

**la** $a0, **sum\_text**

**li** $v0, 4

**syscall**

#print sum

**move** $a0, $t1

**li** $v0, 1

**syscall**

## End ##

**li** $v0, 10

**syscall**

Github commits: <https://github.com/jynxmagic/MIPS-Assembly-Language/commits/master/Task_D/Task_D.asm>