COMP 273 – INTRODUCTION TO COMPUTER SYSTEMS - **Classical CPU Project**

Team Members: Jesse Islam (260639034), Danny Le Kim (260636584), Hoang Anh Khoi Pham (260583816)

CPU NAME: MIPS ROMney 2017

OVERVIEW OF CIRCUIT (**PLEASE USE LOGISIM – EVOLUTION**):

The program written in this circuit will multiply two numbers (keypad input and RAM value) without using the MULT instruction. First, the user must select a number on the keypad. Then, by clicking the ROM button, the exact sequence of instructions to achieve MULT will be downloaded directly into the RAM. Afterwards, enabling ticking will run the program. The sequence is described below:

ADDRESS IN RAM	INSTRUCTION	EXPLANATION	INSTRUCTION (BINARY)
Byte 0: 0000	INPUT D0	From keypad, select number to multiply (0-9)	0011 0000
Byte 1: 0001	BEQ D0, D1, 1100	If keypad input is initially 0, print out the result 0 immediately.	0001 1011
Byte 2: 0010	SUB D0, D1	Calculate the # of times ADD must still loop	0100 1000
Byte 3: 0011	STORE D0, 0000	Save the # of iterations left in addr 0000	0110 0000
Byte 4: 0100	LOAD D0, 1111	Load into D0 the result value	1010 1111
Byte 5: 0101	LOAD D1, 1110	Load into D1 the RAM value to be multiplied	1011 1110
Byte 6: 0110	ADD D0, D1	Add D0 value to D1 value and save it in D0	1000 1000
Byte 7: 0111	STORE D0, 1111	Update the result value in RAM with D0 value	0110 1111
Byte 8: 1000	LOAD D1, STOP	Load into D1 the STOP binary value	1011 1101
Byte 9: 1001	LOAD D0, 0000	Load into D0 the # of iterations left	1010 0000
Byte 10: 1010	BNQ D0, D1, 0010	Perform “mult” by reiterating the instructions between byte 3 and 10 until the # of iterations has the same value as STOP.	1101 0010
Byte 11: 1011	LOAD D0, 1111	Load into D0 the now complete result value	1010 1111
Byte 12: 1100	PRINT D0	Print out result value onto 3-digit display	1100 0000
Byte 13: 1101	STOP	Stop the running the program.	0000 0001
Byte 14: 1110	$RAM VALUE	(where the RAM mult value is stored)	XXXX XXXX
Byte 15: 1111	$FINAL VALUE	(where the result value is stored)	0000 0000

OTHER SPECIFICATIONS:

Each 8-bits instruction has a variable **OP CODE** ranging between 2-4 bits that is decoded in the CU. The remaining bits are the arguments, if applicable. To illustrate, refer to the table below, in which **R** and **r** are either D0 or D1, **Addr** refers to a 4-bit address in RAM and X is unspecified.

INSTRUCTION	INSTRUCTION (BINARY)
STOP	**0000** XXXX
INPUT	**0011 R**XXX
BEQ	**00Rr Addr**
SUB	**010R r**XXX
STORE	**011R Addr**
ADD	**100R r**XXX
LOAD	**101R Addr**
PRINT	**1100 R**XXX
BNQ	**11Rr Addr**