#### Thomas Szklarzewski

## ECE 366 Project 1 Report

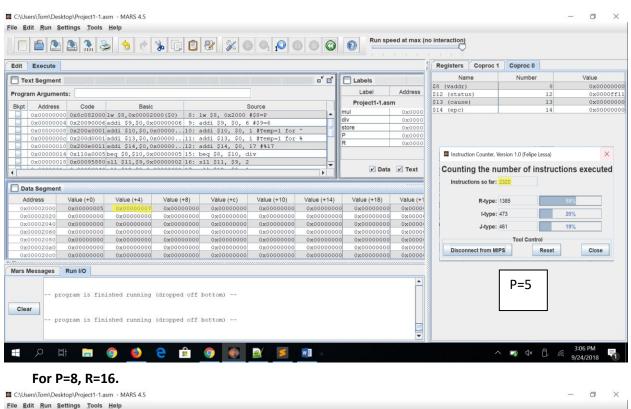
### i) PROGRAM 1

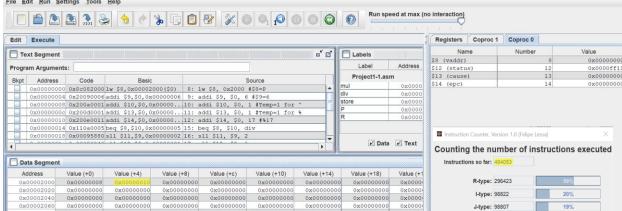
- 1) For Program 1, my project achieved Level 1
- 2) For P=5, R=7.

0x000020c0

Clear

Mars Messages Run I/O





0x00000000 0x00000

- program is finished running (dropped off bottom) --

program is finished running (dropped off bottom) --

(3)

Tool Control

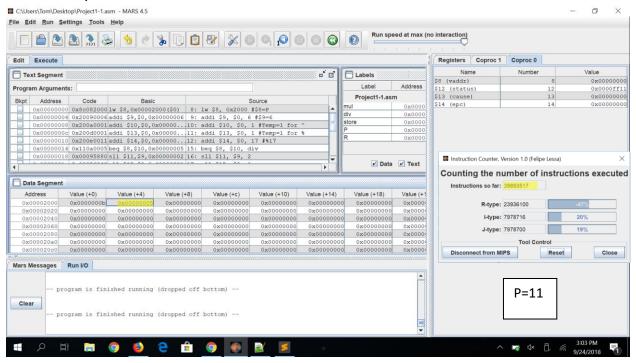
P=8

Disconnect from MIPS

Reset

Close

#### For P=11, R=5



- 3) First, my program loads in the value of (P) into a register. It multiplies 6 (P) number of times. It uses sll to shift the correct amount of times for the binary of 6. It then moves on to modulo, where 17 is subtracted from the correct value of 6^(P) right before a negative number is reached. That number is the remainder of 6^(P) % 17 and is stored into (R).
- 4) Registers \$0, \$8 \$15 are used. The program uses the following instructions: lw, sw, add, addi, sub, sll, slt, beq, and j.

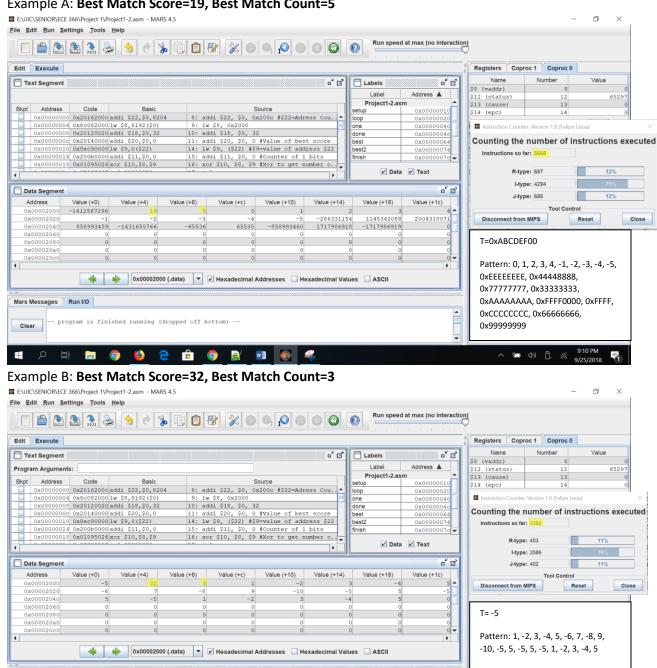
#### **PROGRAM 2**

Mars Messages Run I/O

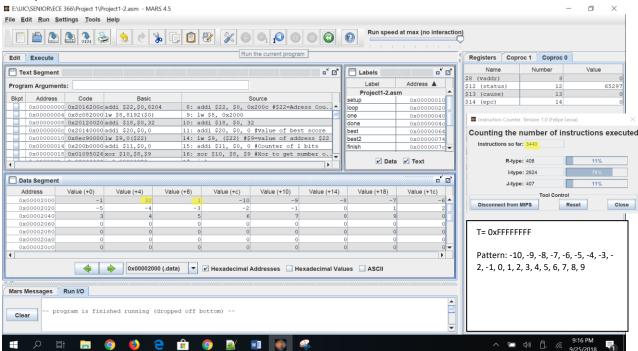
Clear

program is finished running (dropped off bottom) --

- 5) For Program 2, my project achieved Level 2
- 6) Example A: Best Match Score=19, Best Match Count=5



^ \$\frac{1}{2} \$\frac{1}{2}\$ \$\frac{1}{2}\$



Example C: Best Match Score=32, Best Match Count=1

- 7) My program finds the count of best matchings by using a counter. The initial best matching score is set to zero. Any value that has a greater best matching score is then placed in a temp register. The counter will increase by one any time a matching score is found. If a value has a higher matching score than the previous highest, that value will be placed in the temp register and the count will be reset. Once the program has finished running, the best matching score and best matching counts are stored in the correct adresses.
- 8) Registers \$0, \$8 \$13, \$18-\$23 are used. The program uses the following instructions: add, addi, lw, sw, beq, srl, xor, andi, sub, slt, and j.
- 9) I have spent about 20 hours on this project. Most of the time was spent trying to achieve level 2 for each program after reaching level 1.
- 10) I would improve my program by trying to achieve level 2 for both programs and optimize the speed and DIC for each program.

## ii) Program 1 Assembly Code:

```
.data
P: .word 11
R: .word -1
#(6^P)%17=R
.text
lw $8, 0x2000 #$8=P
addi $9, $0, 6 #$9=6
addi $10, $0, 1 #Temp=1 for ^
addi $13, $0, 1 #Temp=1 for %
addi $14, $0, 17 #%17
mul:
beq $8, $10, div #Finished when (number of loops)=P
sll $11, $9, 2 #6=(1*2^2)+(1*2^1)+(0*2^0)
sll $12, $9, 1
add $9, $11, $12
addi $10, $10, 1 #Counter for number of multiplications
j mul
div:
sub $15, $9, $14 #Subtract 17 from value of $9
slt $13, $0, $15 #Check for negative
beq $13, $0, store
add $9, $0, $15 #New number to subtract from
```

j div

store:

sw \$9, 0x2004 #Store remainder

# iii) Program 2 Assembly Code:

.data

T: .word 0xFFFFFFF # 0x2000

best\_matching\_scre: .word -1 # best score = ? within [0, 32] 0x2004

best\_matching\_count: .word -1 # how many patterns achieve the best score? #2008

Pattern\_Array: .word -10, -9, -8, -7, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 #start 0x200c

.text

addi \$22, \$0, 0x200c #\$22=Adress Counter

lw \$8, 0x2000

addi \$18, \$0, 32

addi \$20, \$0, 0 #Value of best score

setup:

lw \$9, (\$22) #\$9=value of address \$22

addi \$11, \$0, 0 #Counter of 1 bits

xor \$10, \$8, \$9 #Xor to get number of matching 1 bits

j loop

loop:

xor \$13, \$10, 0 #Xor to know when to exit loop

beq \$13, 0, done

andi \$12, \$10, 1 #Matching ones

```
beq $12, 1, one
srl $10, $10, 1 #Shift to calculate next bit
j loop
one:
addi $11, $11, 1 #Add matching bit to counter
srl $10, $10, 1 #Shift to calculate next bit
j loop
done:
sub $19, $18, $11 #32-number of matching bits
slt $21, $20, $19 #Find best score
beq $21, 1, best
beq $19, $20, best2
j finish
best:
sw $19, 0x2004 #Store best score
add $20, $0, $19 #Store value of best score
addi $23, $0, 0 #Reset counter for best scores
j best2
best2:
addi $23, $23, 1 #Number of best scores
j finish
finish:
addi $22, $22, 4 #Shift address of array by 4
beq $22, 0x205c, exit #20th number in array
```

j setup

exit:

sw \$23, 0x2008 #Store number of best scores