**Tyler Holland**

9/29/09

Lab 2

No Cheating Signature:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Demo Signature:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Part 1:**

**Initials**: TH

**Hex**: x54 48

**Binary**: 0101 0100 0100 1000

**Instruction**: AND R2, R1, R0

**Part 2:**

**Total:** -32458 with signed binary or 33078 in real numbers.

At the end of the program, the number in registry 1 where the sum is being stored ends up being -32458, but that is because the last bit was signed so everything got flipped during one of the later parts of the loop.

The actual number, if added in decimal, would equal 33078. I figured this out by looking at R3 because it holds the number of the next number to be added to the sum which is stored in R1. I then wrote down all 10 of the numbers loaded into R3 and added them normally.

**Part 3 Code:**

;Name: Tyler Holland

;Date: 10/7/09

;Lab 2

;R1: Holds the answer to be put in x3100

;R2: Holds the "counter", the number that x3100 is being ANDed with next

;R3: Holds the number that will be added to R1 if the AND is negative or positive

;R4: Holds the result of the current AND procedure and is used to loop the R2 initialization

;R5: Holds the number in x3100

;R6: Holds the counter for the major loop

0011 0000 0000 0000 ; start the program at location x3000

0101 001 001 1 00000 ; clear R1

0101 010 010 1 00000 ; clear R2

0101 011 011 1 00000 ; clear R3

0101 100 100 1 00000 ; clear R4

0101 101 101 1 00000 ; clear R5

0101 110 110 1 00000 ; clear R6

0001 110 110 1 01111 ; Load R6 with #15 for a loop counter

0001 011 011 1 00001 ; Load R3 with #1 to add to the answer

0010 101 011110111 ; Load number in x3100 into R5

0001 100 100 1 01111 ; Load R4 with #15, to loop the R2 multiplier

0001 010 010 1 00001 ; Load R2 with #1, to multiply up to the right #

0001 100 100 1 11111 ; Subtract #1 from the counter

0000 100 000000010 ; If the counter is still positive keep going

0001 010 010 000 010 ; ADD R2 to R2 to make it larger

0000 111 111111100 ; Jump back to subtracting 1 from the counter

;Enter major loop

0101 100 101 000 010 ; AND R5 with R2 and put the result in R4

0000 010 000000001 ; If the next number should be a 1, move through, otherwise jump ahead

0001 001 001 000 011 ; ADD R1 to R3 and store in R1

0001 011 011 000 011 ; Double R3 in preparation for next cycle

0101 010 010 1 00000 ; clear R2 again for another loop

0101 100 100 1 00000 ; clear R4

0001 100 110 1 11111 ; Load R4 with R6 - 1, to loop the R2 multiplier

0001 110 110 1 11111 ; Decrement the major loop counter

0000 100 000000001 ; Jump to end of program if the counter is complete

0000 011 111110001 ; Jump back to R2 loop and do it all over again until it goes through all the digits

0011 001 011100111 ; Store R1 into x3101

1111 0000 00100101 ; halt, DONE!