**Project 1 Report**

**Part 1**

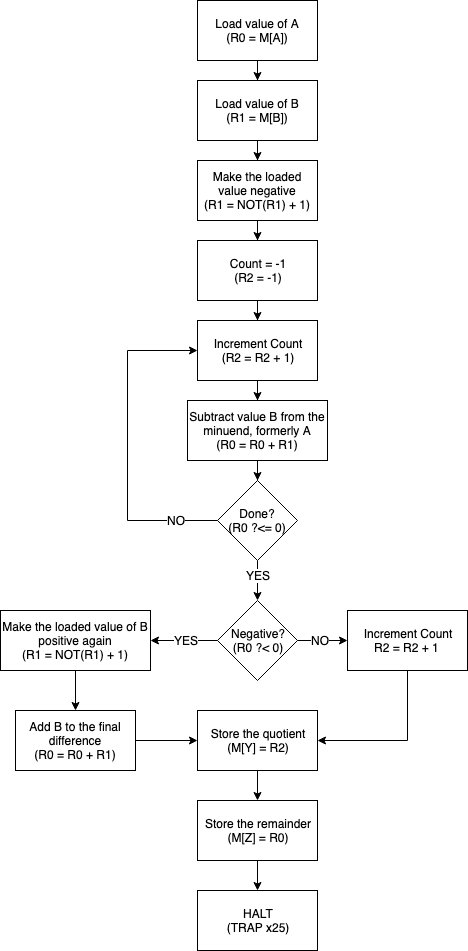
**Machine Problem 2: Integer Division and Modulo**

Summary:

The program is fairly straightforward. First, it loading the input values: A, the dividend, and B, the divisor. In order to find out how many times B can go into A, the program must subtract B from A continuously until the result either reaches zero or is negative. While this is happening, a counter, which will eventually become the quotient, counts the amount of times B can subtract from A. As you can see in line 9 in the program, the counter starts at -1 in order to compensate for the fact that the loop is a “do-while” loop add performs an iteration before it reaches and evaluates the conditional statement. If the final result is zero, the program will compensate by adding 1 to the quotient (again, due to the fact that it is a do-while loop), and the remainder would be zero. However, if the final result is negative, then the program would calculate the remainder by adding the value B to the final result and store the values in Y and Z. For example, if the program is performing (16 / 13), then:

If I were to write this program again, I might do the iterative statement before the conditional statement exactly like a regular while-loop as a matter of preference.

Flowchart:



Source Code:

.ORIG x3000

LDI R0, A

LDI R1, B

NOT R1, R1

ADD R1, R1, #1 ; Make R1 negative to subtract from R0

AND R2, R2, #0 ; Clear R2, it will be the quotient

ADD R2, R2, #-1 ; Starts R2 as -1

ITER ADD R2, R2, #1 ; First iteration would set R2 to 0

ADD R0, R0, R1 ; Subtract value B from value A

BRp ITER

BRn NEG

BRz ZERO

NEG NOT R1, R1

ADD R1, R1, #1

ADD R0, R0, R1

BRnzp FIN

ZERO ADD R2, R2, #1 ; If zero, add 1 to compensate for starting as -1

FIN STI R2, Y

STI R0, Z

HALT

A .FILL x3100

B .FILL x3101

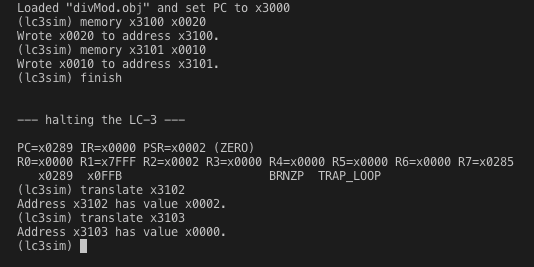
Y .FILL x3102 ; Location to store quotient

Z .FILL x3103 ; Location to store remainder

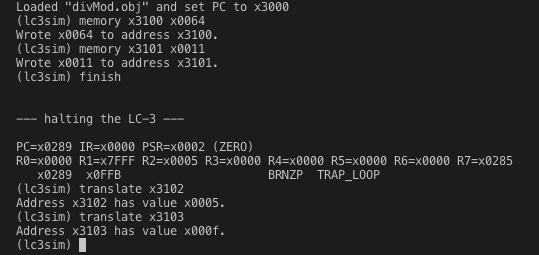
.END

Results:

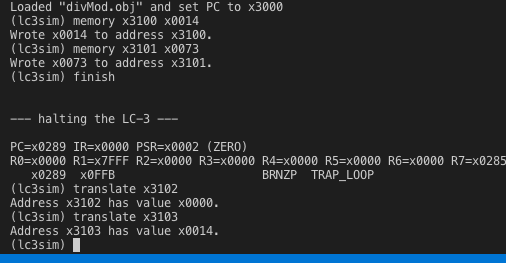
32 / 16 = 2 R 0



100 / 17 = 5 R 15



20 / 115 = 0 R 20

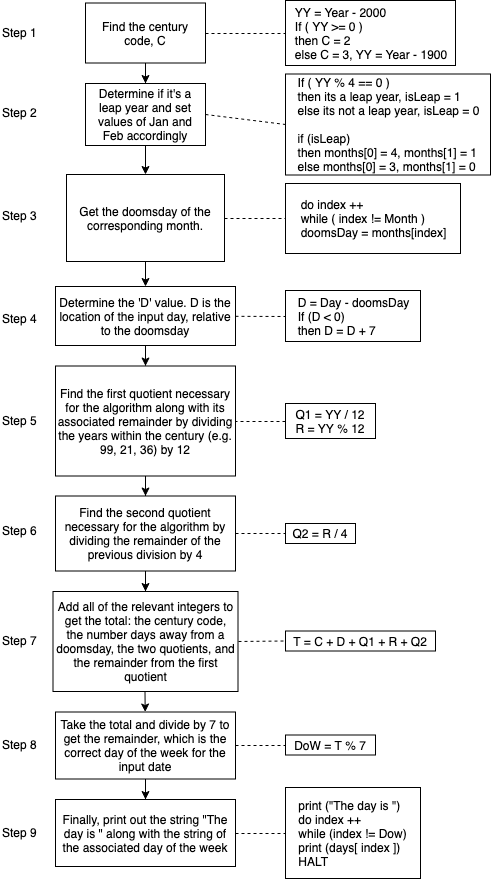


**Part 2**

**Assembly Problem 2: Day of the Week**

Summary:

This program uses the doomsday algorithm, created by John Conway, in order to find the day of the week for any given date between the first day of 1900 and the final day of 2099. The algorithm works by breaking down into several parts which calculates five important values. All of which would be added together and modulated by 7 to produce the final value: the day of the week. The graph below details the steps of the algorithm along with its equivalent pseudocode:



The algorithm relies on the fact that the Gregorian calendar repeats every four hundred years. John Conway discovered that certain days (“doomsdays” as he calls them) consistently fall into a specific day of the week. Once you find the doomsday by considering the century, year, and the month, you can find where the input day falls relative to the doomsday, and calculate the day of the week from there. The original intention of this algorithm was for people to calculate the day of the week in their head which requires a lot a memorization and practice. I decided to implement the algorithm in a similar way and as a result, it required a lot of predefined values. The algorithm also requires the use of the division operation, which is code I previously implemented in the machine problem of this project.

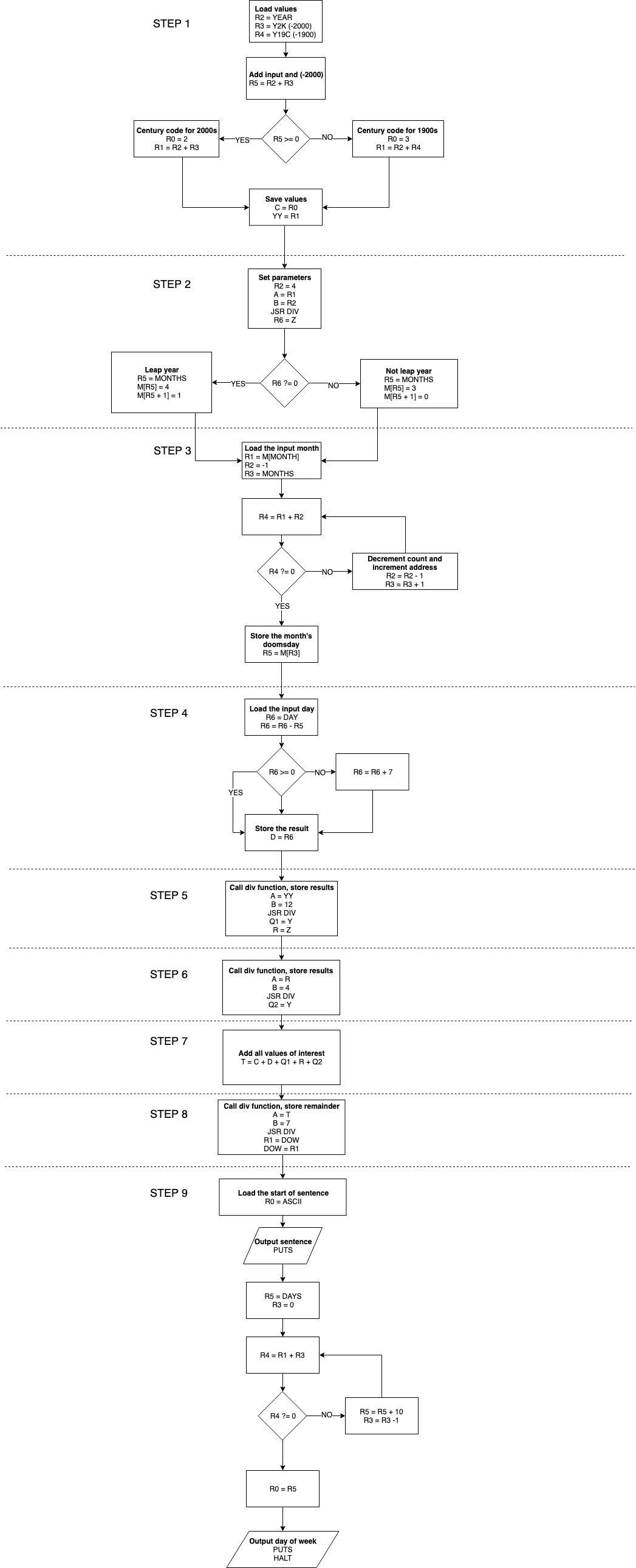
This program utilizes four “If statements,” two “do-while loops,” and two arrays. My first attempt at a solution did not have any loops but instead used statements reminiscent of “case-switch.” However, I realized that I could iterate through the address value since the elements in the array are in succession in memory. As for the array of strings (“Monday,” “Tuesday,” etc.) I had to add additional spaces to meet a consistent value to iterate with. In this case, I made sure each string had 9 characters total to meet the length of “Wednesday” and iterate the address value with 10 as shown in line 127.

References I used to implement the algorithm:

<https://en.wikipedia.org/wiki/Doomsday_rule>

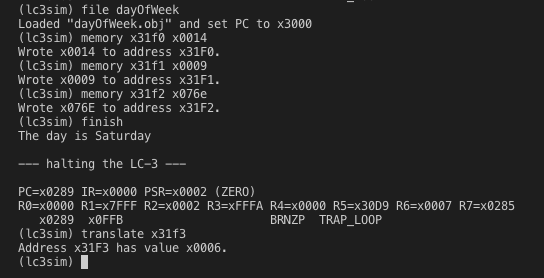
<https://www.youtube.com/watch?v=714LTMNJy5M>

Flow Chart:

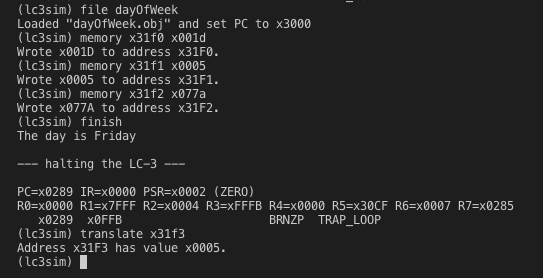


Results:

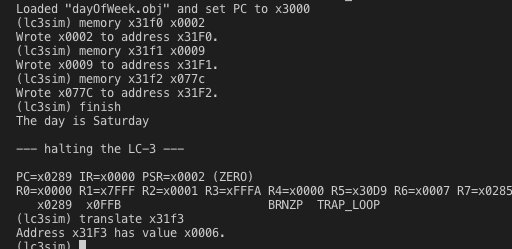
20-September-1902



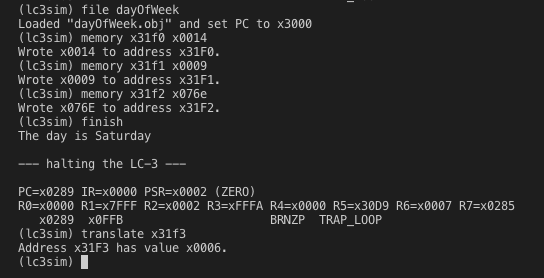
29-May-1914



2-September-1916



20-December-1922



28-February-1977

