






Machine Oriented programming 02322

Project 1, group 28. 22-03-2022

Name	Contributions	Picture
Rasmus Prætorius s215777	<p>Code for:</p> <ul style="list-style-type: none">- AssignmentDeliverable1.asm- AssignmentDeliverable2.asm- AssignmentDeliverable3.asm- AssignmentDeliverable4.asm- AssignmentDeliverable5.asm <p>Diagrams:</p> <ul style="list-style-type: none">- Diagram 2A- Diagram 3A- Diagram 3B- Diagram 4A- Diagram 4B- Diagram 5B <p>- All report descriptions</p>	
Andrew Shereef s173750	<p>Contributed in code:</p> <ul style="list-style-type: none">-AssignmentDeliverable1-AssignmentDeliverable3 <p>Diagrams:</p> <ul style="list-style-type: none">- Diagram 3	
Alban Dalifi s205416	<p>Contributed in code:</p> <ul style="list-style-type: none">-AssignmentDeliverable1 <p>Diagrams:</p> <ul style="list-style-type: none">- Diagram 1A- Diagram 2A- Diagram 2B- Diagram 5A- Diagram 5B	

Deliverable Assignment 1

Solve the exercise 7.34 from page 259 of the lecture book.

It is often useful to find the midpoint between two values. For this problem, assume A and B are both even numbers, and A is less than B. For example, if $A = 2$ and $B = 8$, the midpoint is 5. The following program finds the midpoint of two even numbers A and B by continually incrementing the smaller number and decrementing the larger number. You can assume that A and B have been loaded with values before this program starts execution.

Your job: Insert the missing instructions.

```

                .ORIG  x3000
LD      R0,A
LD      R1,B
X          ----- (a)
                ----- (b)
ADD     R2,R2,R1
                ----- (c)
ADD     R1,R1,#-1
                ----- (d)
BRnzp   X
DONE     ST      R1,C
          TRAP   x25
A        .BLKW  1
B        .BLKW  1
C        .BLKW  1
          .END
```

To solve this exercise, a diagram deconstructing the operations needed to find the midpoint between two whole numbers was made. The idea represented in diagram 1A was that once could increment the lower of the two numbers, and decrement the higher number in a loop until the two numbers were equal. This diagram also documents the assembly program "AssignmentDeliverable1.asm".

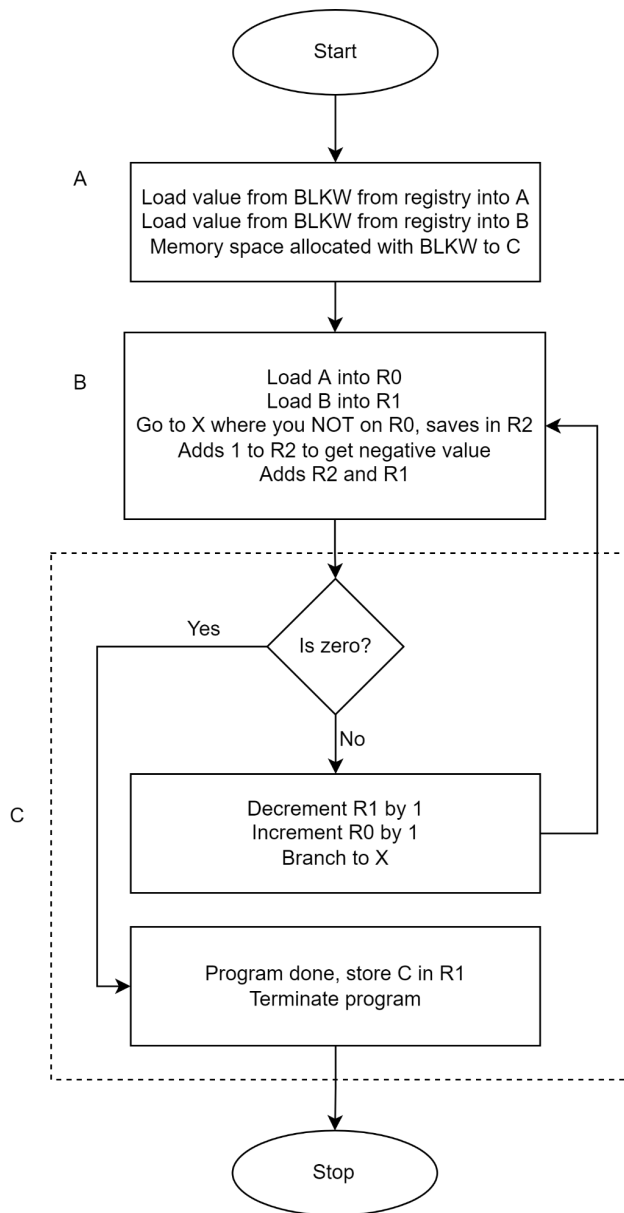


Diagram 1A.

The below screenshot Figure 1B is a snippet of the commented code implemented using the aforementioned diagram, inserting the missing instructions from the exercise.

```

1  .ORIG x3000
2  LD R0,A           ;Loads A into R0
3  LD R1,B           ;Loads B into R1
4  X NOT R2,R0       ;NOT on R0, saves in R2
5  ADD R2,R2,#1      ;Adds 1 to R2 to get negative value
6  ADD R2,R2,R1      ;Adds R2 and R1
7  BRZ DONE         ;Branch to DONE if result is 0
8  ADD R1,R1,#-1     ;Decrement R1 by 1
9  ADD R0,R0,#1      ;Increment R0 by 1
10 BRnzp X          ;Branch to X
11 DONE ST R1,C      ;Store C in R1
12 TRAP x25         ;Terminate program
13 A .BLKW 1
14 B .BLKW 1
15 C .BLKW 1
16 .END

```

Figure 1B.

Deliverable Assignment 2

Write a function called “readS” that reads a 2 digit decimal number from the console and returns its value in register R0. The function doesn’t have any input arguments. These are the steps performed by the function:

1. Write on the console the message: “Input a 2 digit decimal number:”
2. Read two characters from the console (with echo) and convert them to an integer. Echo means that when you type a character on the keyboard it is also shown on the console. As a simplification it is assumed that the user always types digits (he never types letters or other symbols), which means that you don’t have to validate the input. In other words you don’t have to check if the user has typed digits or some other symbols. As another simplification, numbers less than ten are typed with a zero in front. For example 02, 07 etc.
3. Returns the number read from the console in register R0

To solve this deliverable a simple diagram 2A was first made, deconstructing the tasks required to read two characters from the console with echo, convert them into digits, concatenate them together, and store the resulting value in register R1.

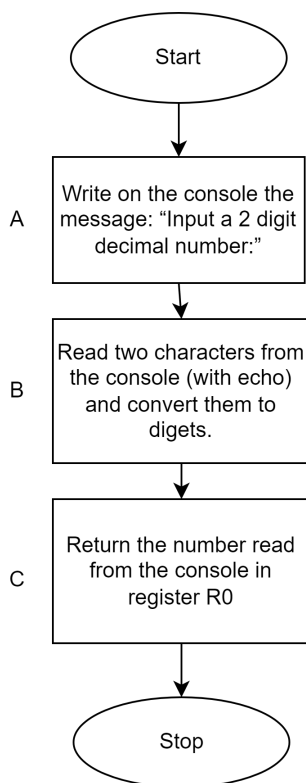


Diagram 2A.

On the back of the first diagram 2B, a second diagram further deconstructs this function down to its fundamental operations. This involved reading characters from the console with echo and converting them from ASCII characters to digit values by adding #-48. Branching was used to determine the first digit and to multiply it by 9 inside a loop and +1 outside the loop. Later the second digit could be added to the first, thus making up their concatenation. This diagram also documents the assembly program “AssignmentDeliverable2.asm”.

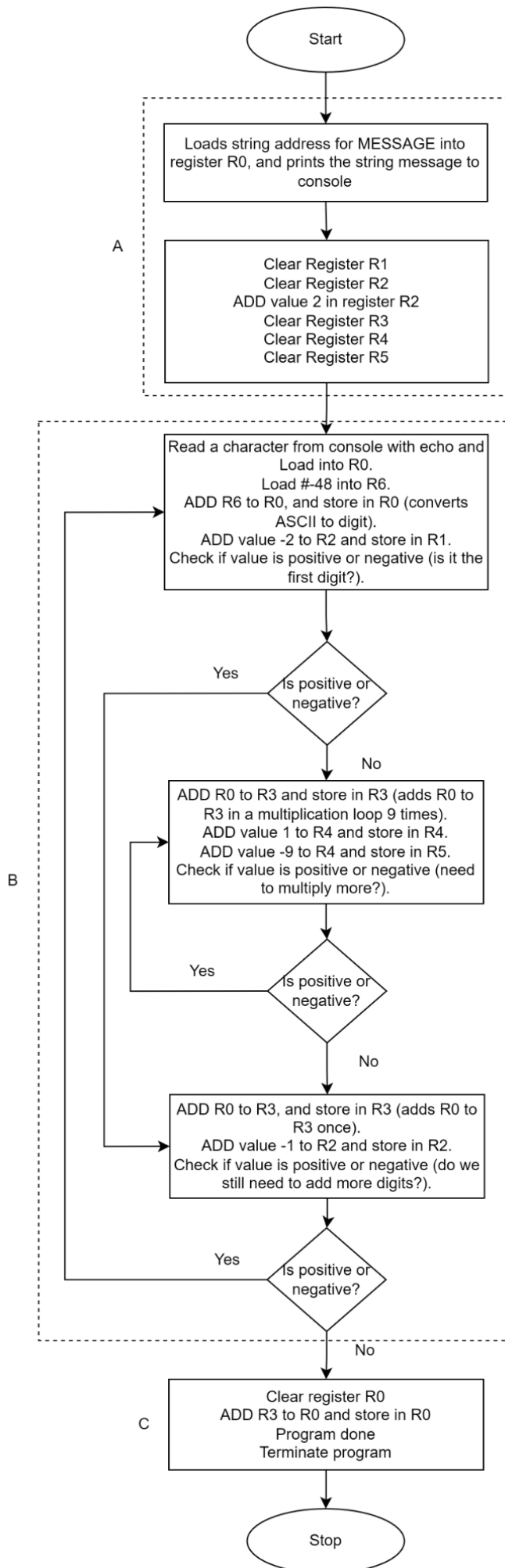


Diagram 2B.

Deliverable Assignment 3

Write a function called “isPrime” that takes as argument a number (with the decimal value between 0 and 99) stored in R0 and returns value 1 if the number is prime and value 0 if the number is not prime. The return value is also stored in register R0.

To solve this deliverable a simple diagram 3A was made, deconstructing the tasks needed to determine if a number between 0 and 99 stored in R0 is a prime number or not and store the result as either 1 or 0 in R0.

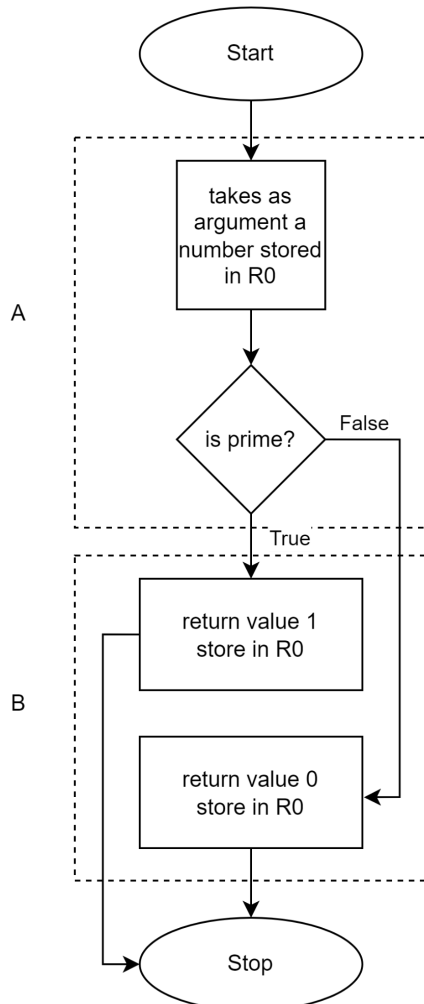
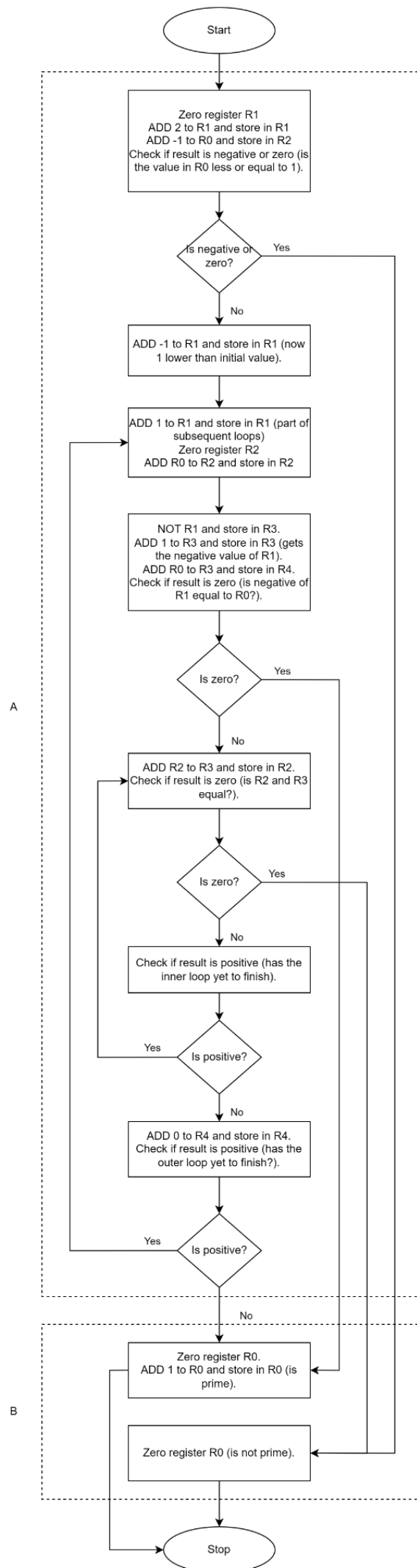


Diagram 3A.

To further detail the operations needed, another diagram 3B was made, deconstructing the tasks needed to test if the number in R0 is a prime number. This involved using an outer loop iterating over numbers from 2 to the value of the number in R0, and an inner loop testing if a particular value is a factor in the number. This was done by iteratively subtracting the test number to see if the result would become 0, in which case the number would be a prime number. If the R0 number is less than 1 or has no factors other than itself, the number would then be a prime number. This diagram also documents the assembly program “AssignmentDeliverable3.asm”.



Deliverable Assignment 4

Write a function called “resultS” that takes as argument a value stored in R0. If the value in R0 is not zero then the function should display on the console the message: “The number is prime” else it should display the message “The number is not prime”.

To solve this deliverable a simple diagram 4A was made, deconstructing the tasks needed to read the value stored in R0 and depending on whether it is a 1 or a 0, print a correct message to the console.

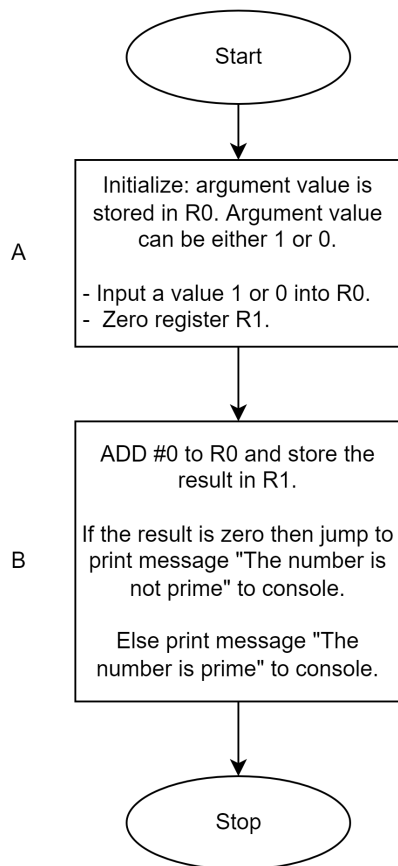


Diagram 4A.

To further detail the operations needed, another diagram 4B was made deconstructing the tasks further. Here, depending on the result of a branch used on R0, one of two possible string message addresses are loaded into R0 and printed to the console. This diagram also documents the assembly program “AssignmentDeliverable4.asm”.

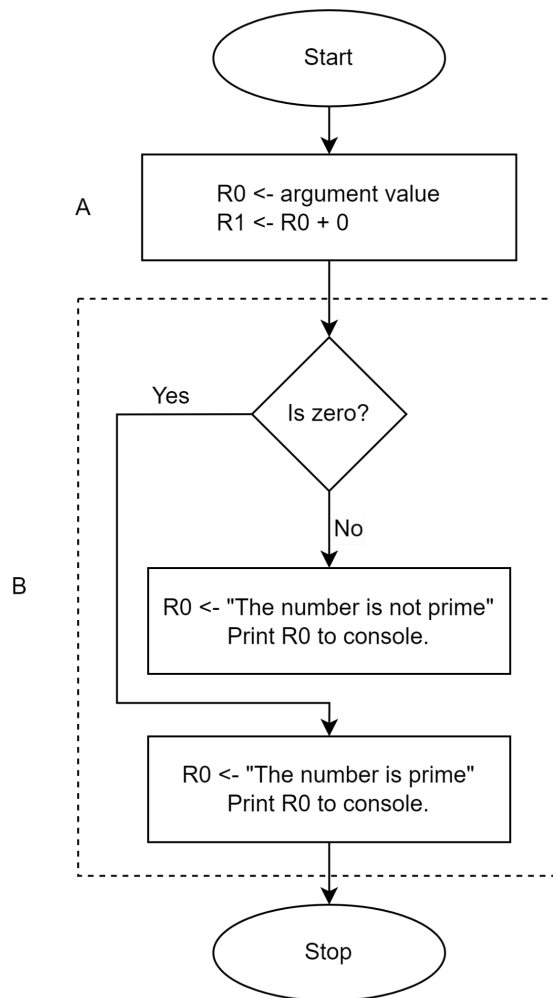


Diagram 4B.

Deliverable Assignment 5

Write an assembly program that first reads a 2 digit decimal number using the function “readS” then checks if the number is prime or not using the function “isPrime” and then shows on the display a message that says if the number was prime or not by using the function “resultS”. The program should run in an infinite loop (when the function resultS returns, the program jumps back to function readS). A typical screen dump of the console should look something like this:

```

Input a 2 digit decimal number:12
The number is not prime
Input a 2 digit decimal number:02
The number is prime
  
```

To solve this deliverable a simple diagram 5A was made, deconstructing the tasks needed to call subroutines readS, sPrime and resultS in an infinite loop. The program should be able to prompt the user infinitely for a two digit number, and display a message conveying whether the number is a prime number or not.

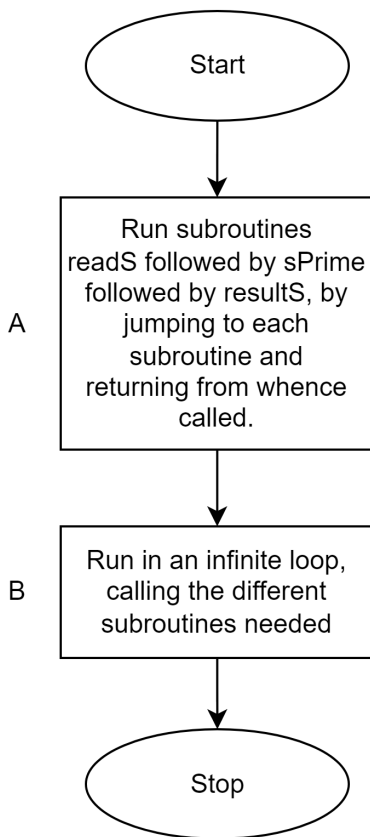


Diagram 5A.

To further detail the operations needed, another diagram 5B was made, deconstructing the subroutine calls and loop. Upon jump, each subroutine readS, sPrime and resultS, store the registers needed before their operations, and restore them to their original values again afterwards, before returning to whence they were called. A newline is printed for each iteration, so that the prompt messages from each subroutine remain visibly separate in the console. This diagram also documents the assembly program "AssignmentDeliverable5.asm".

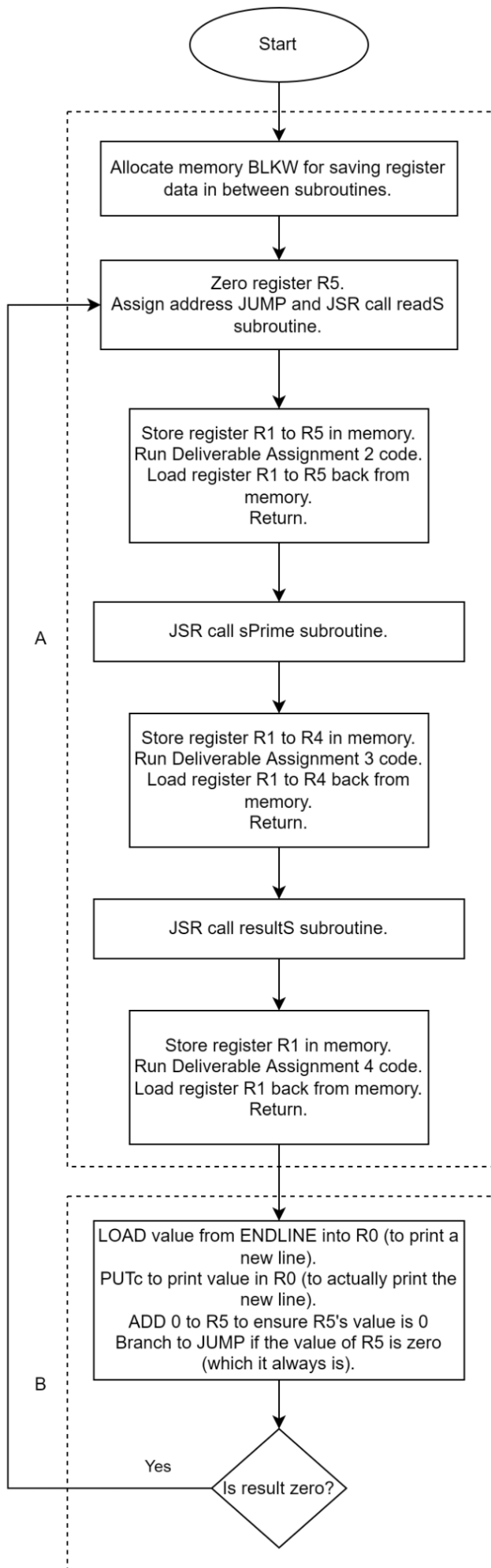


Diagram 5B.

Below is a screenshot Figure 5C of the console prompt and output messages running in an infinite loop. Numbers from 95 to 99 can here be seen tested for prime numbers.



```
Console (click to focus)

Input a character> 9

Input a character> 5
The number is not prime
Input a 2 digit decimal number:
Input a character> 9

Input a character> 6
The number is not prime
Input a 2 digit decimal number:
Input a character> 9

Input a character> 7
The number is prime
Input a 2 digit decimal number:
Input a character> 9

Input a character> 8
The number is not prime
Input a 2 digit decimal number:
Input a character> 9

Input a character> 9
The number is not prime
Input a 2 digit decimal number:
Input a character> █
```

Diagram 5C.