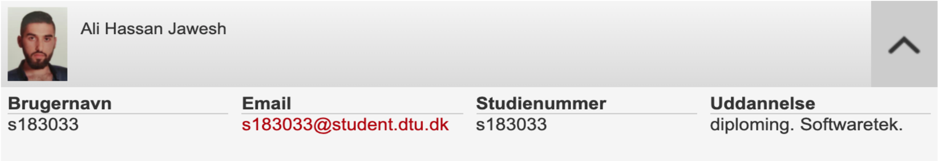
02322
Machine Oriented programming

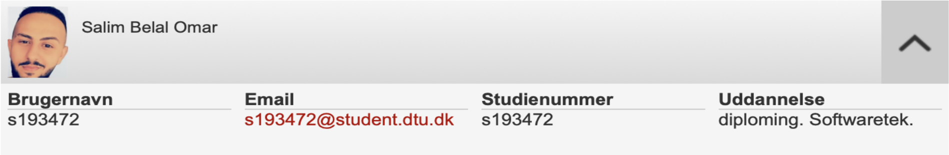
Project 1

Gruppe 16



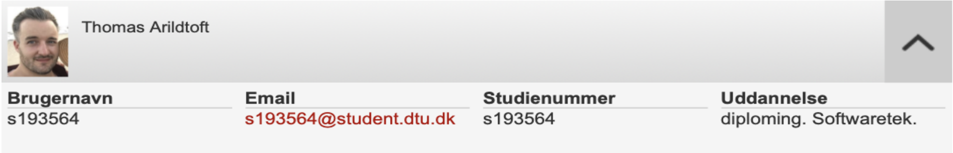
Ali Hassan Jawesh

Brugernavn	Email	Studienummer	Uddannelse
s183033	s183033@student.dtu.dk	s183033	diploming. Softwaretek.



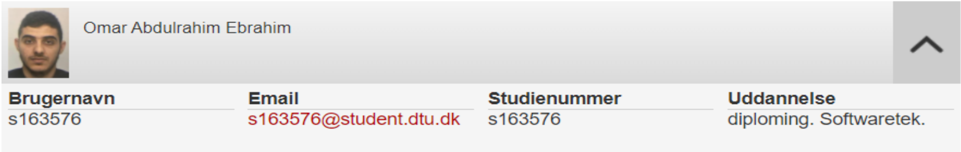
Salim Belal Omar

Brugernavn	Email	Studienummer	Uddannelse
s193472	s193472@student.dtu.dk	s193472	diploming. Softwaretek.



Thomas Arildtoft

Brugernavn	Email	Studienummer	Uddannelse
s193564	s193564@student.dtu.dk	s193564	diploming. Softwaretek.

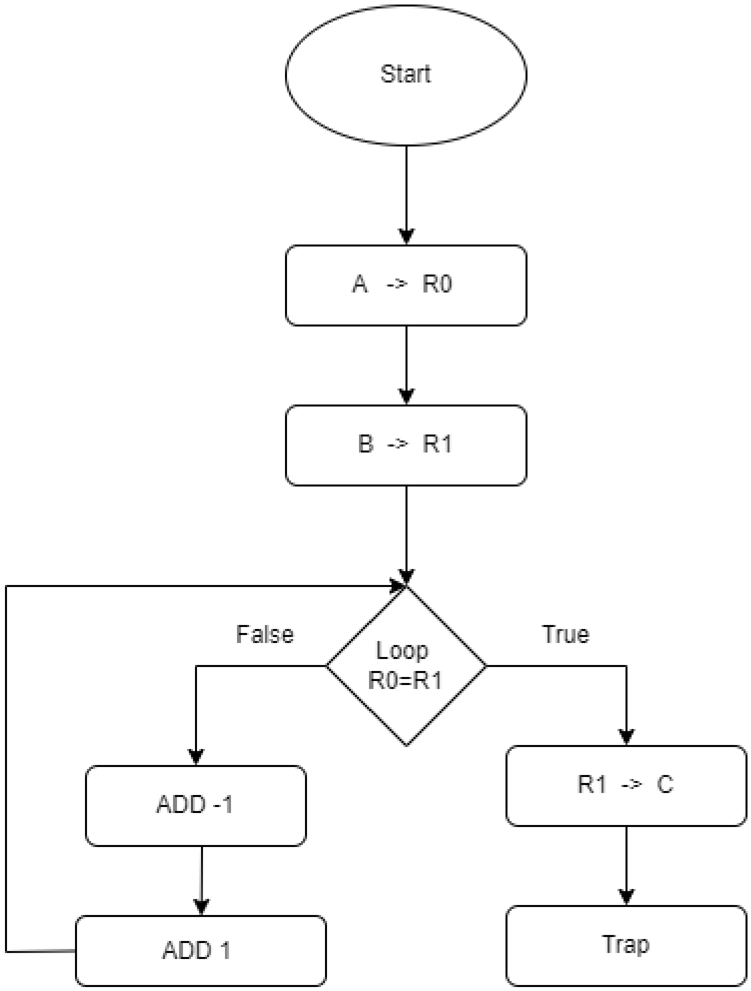


Omar Abdulrahim Ebrahim

Brugernavn	Email	Studienummer	Uddannelse
s163576	s163576@student.dtu.dk	s163576	diploming. Softwaretek.

Assignment 1

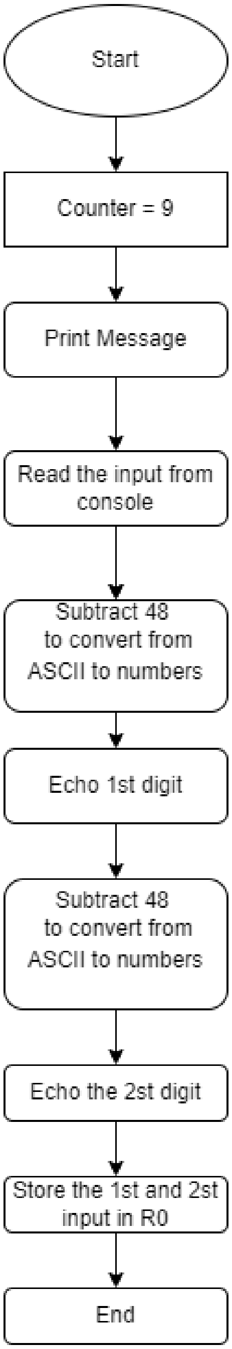
In this assignment we need to save the value of A in register 0, and the value of B in register 1. The values of A and B should get closer to each other until it reach the same value so we get the midpoint.



```
.ORIG x3000                    ;Start from  the address x3000

        LD R0,A                 ;Loads the value A in to R0
        LD R1,B                 ;Loads the value B in to R1
      X  NOT R2,R0             ; Finds the inverse value of R0 and stores it in R2       (a)
        ADD R2,R2,#1           ; add 1 to R2  and save it in R2                          (b)
        ADD R2,R2,R1           ; add R2 to R1 and stores in R2 R1)
        BRz DONE               ; If the previous instruction gave 0: pass on to DONE (c)
        ADD R1,R1,#-1          ; subtract 1 from B
        ADD R0,R0,#1           ;   Add  1 to A                                           (d)
        BRnzp X                ; If the previous instruction gave something negative, 0 or positive:
give to X
     DONE ST R1,C              ;Save the value in R1 in C
        TRAP x25               ;Stop the program
       A .BLKW 1               ;Reserve 1 location in memory and call it
       B .BLKW 1               ;Reserve 1 location in memory and call it
       C .BLKW 1               ;Reserve 1 location in memory and call it
        .END
```

Assignment 2

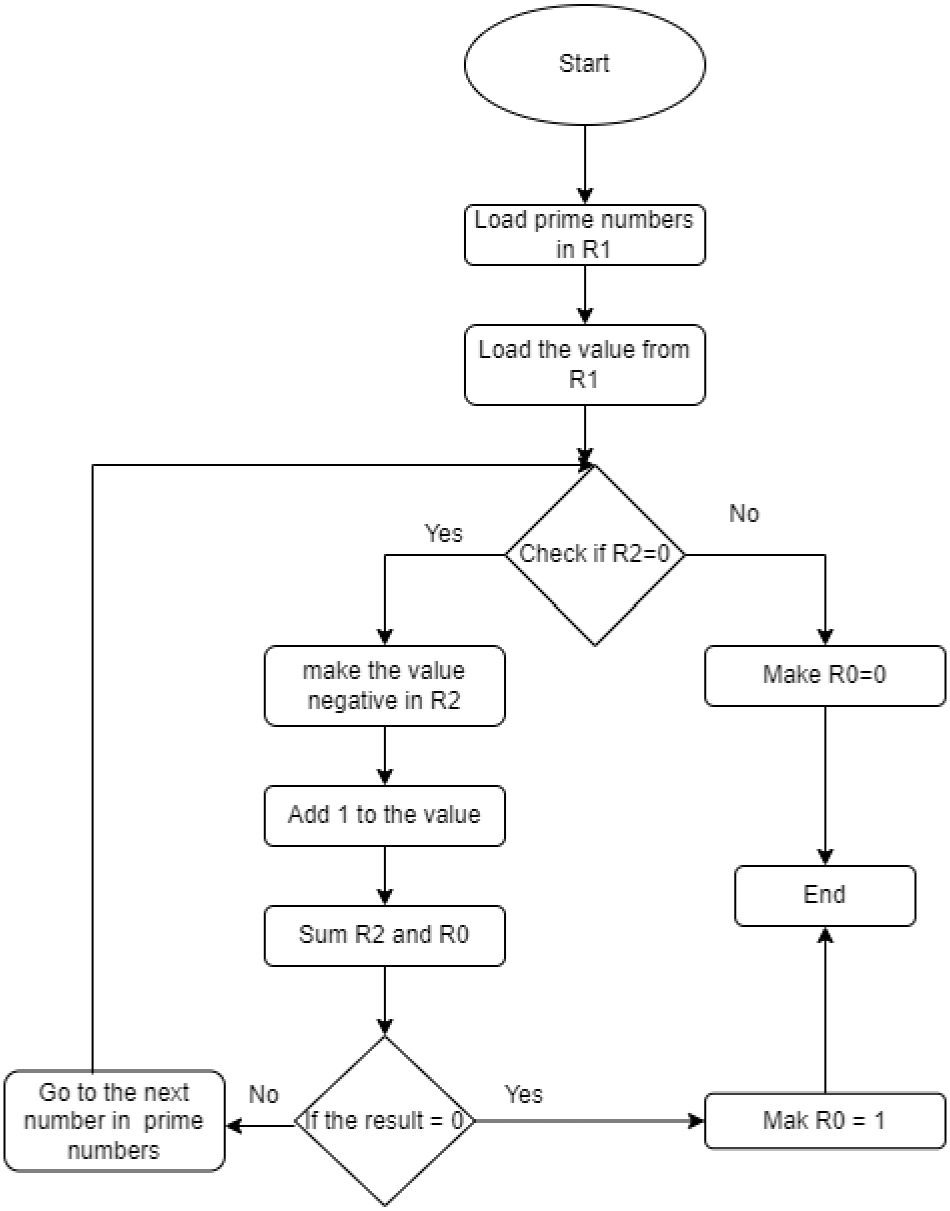
The next flowchart diagram shows how the function works. It's able to convert the input from the user into integer in register 0.


Start
Counter = 9
Print Message
Read the input from console
Subtract 48 to convert from ASCII to numbers
Echo 1st digit
Subtract 48 to convert from ASCII to numbers
Echo the 2st digit
Store the 1st and 2st input in R0
End

```
        .ORIG x3000

readS       LD R5, MULTI       ; give number 9 to R5
        LEA R0,Message     ; Retrieves the address of the message to be written to R0
        PUTS               ; Prints the message string

        GETC               ; Receives a character in R0
        OUT                ; Prints draws again
        ;
        LD  R4,fASCII      ; Sets R4 to -48
        ADD R0, R0, R4     ; Subtracts 48 from input, to convert from ASCII to numbers
        AND R4, R4, #0     ; Sets R4 to 0
        ;
        AND R1,R1,#0       ; Set R1 til 0

        ADD R1,R0,R1       ; Set R0 til R1

        ADD R2,R1,x0       ; set R1 to R2
loopm   ADD R1,R1,R2       ; set R1 og R2 i R1
        ADD R5,R5,x-1      ; minus 1 fra R5
        BRp loopm

        GETC               ; Receives a character in R0
        OUT                ; Prints draws again (echo)

        LD  R4,fASCII      ; Sets R4 to -48
        ADD R0, R0, R4     ; Subtracts 48 from input, to convert from ASCII to numbers
        AND R4, R4, #0     ; Sets R4 to 0

        AND R2,R2,#0       ; Set R2 til 0
        ADD R2,R2,R0       ; Set R2 til R1
        ADD R0,R1,R2       ; put the input til R0

Message     .STRINGZ "write a 2 digit decimal number: "
fASCII      .FILL #-48
MULTI       .FILL #9
        .END
```

Assignment 3

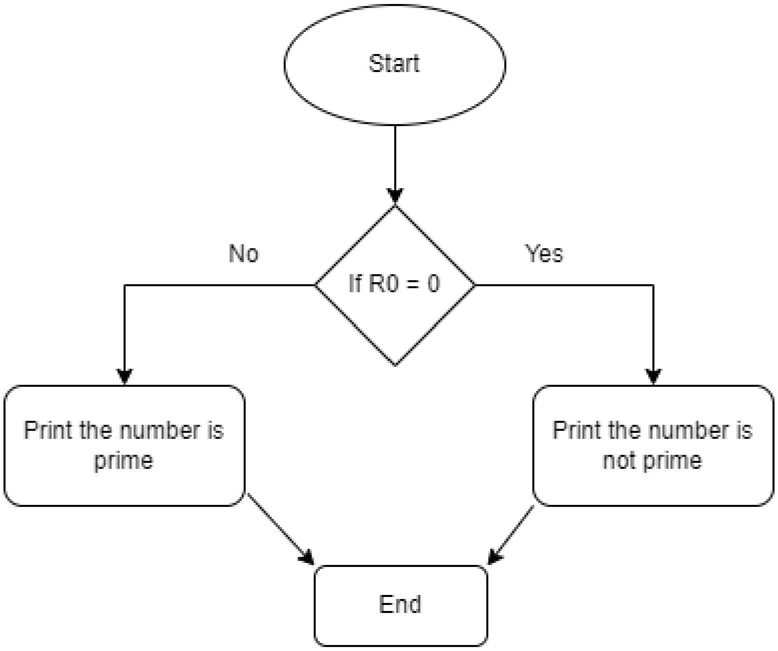
In this assignment we have the prime numbers loaded from data in register 1 and we compare them one by one with the user input to check if the input is one of the prime numbers.



```
isPrime      AND R1,R1,#0            ; clear R1
             AND R2,R2,#0            ; clear R2
            LEA R1,PrimeNumber       ; get the address for first primeNumber in R1
loops        LDR R2,R1,#0            ; get the value from R1
             BRz NOtprime            ; check if the number in R2 is equal to 0, if it is not jump to NOtprime

              NOT R2,R2              ; make the value to negative value
             ADD R2,R2,#1            ; add 1 to R2
             ADD R3,R2,R0            ; put R2 and R0 in R3
             BRz myPrime             ; if =0 so skip over to myPrime, if not continue the code

            ADD R1,R1,#1             ; go to the next number in our primeNumber
            BR loops;                ; go back to the loop

            AND R0,R0,x0             ; make R0 =0

myPrime     AND R0,R0,#0
            ADD R0,R0,#1

NOtprime    AND R0,R0,#0
            ADD R0,R0,#0

PrimeNumber .FILL #2                 ; place prime values at code lines
            .FILL #3
            .FILL #5
            .FILL #7
            .FILL #11
            .FILL #13
            .FILL #17
            .FILL #23
            .FILL #29
            .FILL #31
            .FILL #37
            .FILL #41
            .FILL #43
            .FILL #47
            .FILL #53
            .FILL #59
            .FILL #61
            .FILL #67
            .FILL #71
            .FILL #73
            .FILL #79
            .FILL #83
            .FILL #89
            .FILL #97

            .END
```

Assignment 4

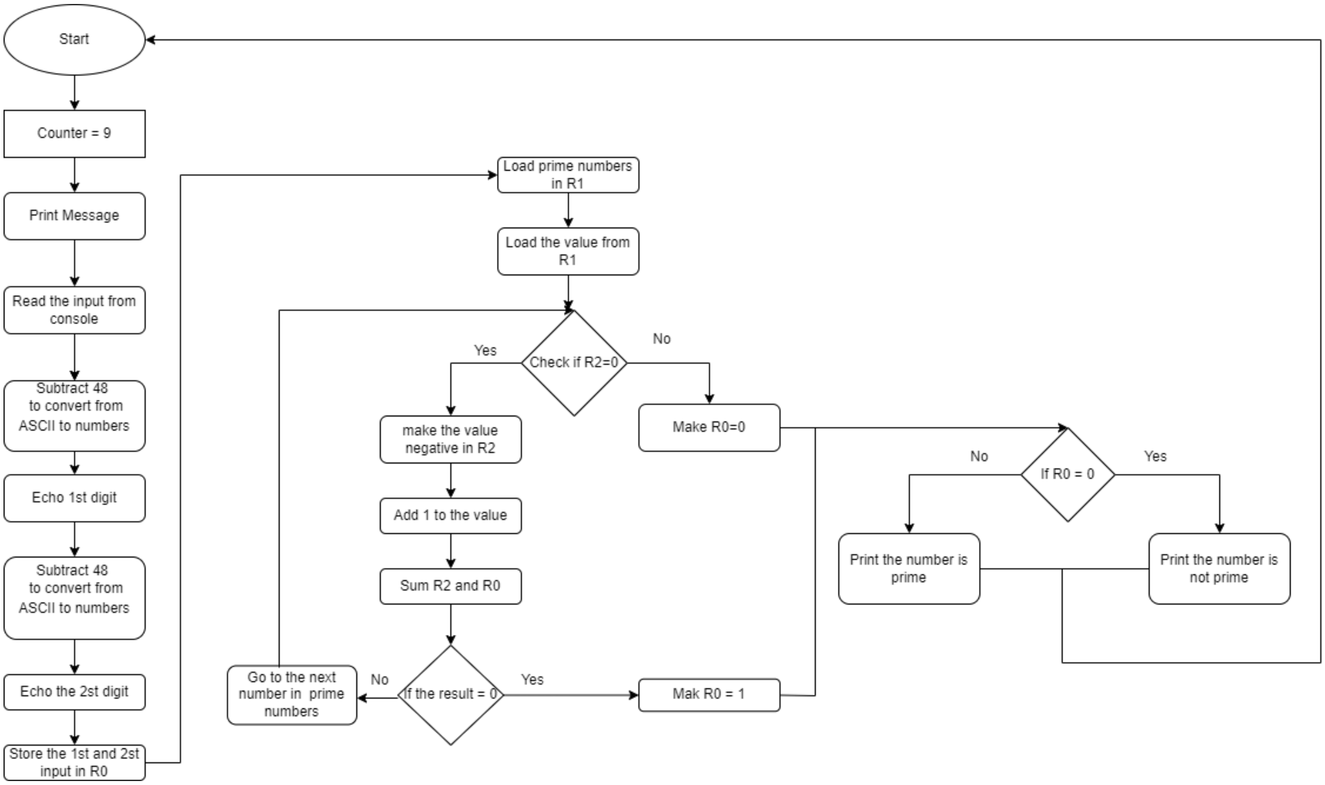
The diagram shows that when the register 0 = 0 (that means the input number is not prime), the output shows the message ”The number is not prime”. when the register 0 is not 0 (that means the input number is prime), the output shows the message ”The number is prime”



```
resultS
                    ADD R1,R0,#0               ; copy R0
                    BRp  printPrime            ;
             LEA R0,MSGNotPrime               ; Retrieves the address of the message to be written to R0
                     PUTS                        ; Prints the message string

   printPrime       LEA R0,MSGPrime           ; Retrieves the address of the message to be written to R0
                     PUTS                        ; Prints the message string

 MSGPrime     .STRINGZ " \nthe number is prime.\n "
MSGNotPrime    .STRINGZ "\nthe number is not prime.\n "

 .END
```

Assignment 5

The diagram here shows how the functions are connected and gives a picture all over the process and how it works.


Start
Counter = 9
Print Message
Read the input from console
Subtract 48 to convert from ASCII to numbers
Echo 1st digit
Subtract 48 to convert from ASCII to numbers
Echo the 2st digit
Store the 1st and 2st input in R0
Load prime numbers in R1
Load the value from R1
Yes
Check if R2=0
No
make the value negative in R2
Make R0=0
Add 1 to the value
Sum R2 and R0
Go to the next number in prime numbers
No
If the result = 0
Yes
Mak R0 = 1
No
If R0 = 0
Yes
Print the number is prime
Print the number is not prime

```
            .ORIG x3000                     ; starting at address x3000

Loop          JSR readS
              ; call function readS
              JSR isPrime
              ; call function isPrime

              JSR resultS
              ; call function resultS
              BR Loop
              HALT


readS       ST R7,RETURNreadS       ; Saves where readS should return in RETURN
            LD R5, MULTI              ; give number 9 to R5
            LEA R0,Message            ; Retrieves the address of the message to be written to R0
            PUTS                      ; Prints the message string

            GETC                      ; Receives a character in R0
            OUT                       ; Prints draws again

            LD  R4,fASCII             ; Sets R4 to -48
            ADD R0, R0, R4            ; Subtracts 48 from input, to convert from ASCII to numbers
            AND R4, R4, #0            ; Sets R4 to 0

            AND R1,R1,#0              ; Set R1 til 0

            ADD R1,R0,R1              ; Set R0 til R1

            ADD R2,R1,x0              ; set R1 to R2
loopm       ADD R1,R1,R2            ; set R1 og R2 i R1
            ADD R5,R5,x-1             ; minus 1 fra R5
            BRp loopm

            GETC                      ; Receives a character in R0
            OUT                       ; Prints draws again (echo)


            LD  R4,fASCII             ; Sets R4 to -48
            ADD R0, R0, R4            ; Subtracts 48 from input, to convert from ASCII to numbers
            AND R4, R4, #0            ; Sets R4 to 0

            AND R2,R2,#0              ; Set R2 til 0
            ADD R2,R2,R0              ; Set R2 til R1
            ADD R0,R1,R2              ; put the input til R0
            LD R7,RETURNreadS         ; Set R7 back to  readS should return to

            RET                       ; JMP R7; Returns



isPrime     ST R7,RETURNisPrime     ; Saves where isPrime should return in RETURN
            AND R1,R1,#0              ; clear R1
            AND R2,R2,#0                    ; clear R2
            LEA R1,PrimeNumber              ; get the address for first primeNumber in R1
loops        LDR R2,R1,#0                 ; get the value from R1
            BRz NOtprime     ; check if the number in R2 is equal to 0, if it is not jump to NOtprime

            NOT R2,R2                       ; make the value to negative value
            ADD R2,R2,#1                    ; add 1 to R2
            ADD R3,R2,R0                    ; put R2 and R0 in R3
            BRz myPrime                     ; if =0 so skip over to myPrime, if not continue the code

            ADD R1,R1,#1                   ; go to the next number in our primeNumber
            BR loops;                      ; go back to the loop

            AND R0,R0,x0                   ; make R0 =0

myPrime       AND R0,R0,#0
            ADD R0,R0,#1
            LD R7,RETURNisPrime            ; set R7 back to  isPrime should return to I
            RET                            ; JMP R7; Returns

NOtprime      AND R0,R0,#0
            ADD R0,R0,#0
            LD R7,RETURNisPrime            ; set R7 back to isPrime  should return to
            RET                            ; JMP R7; Returns




resultS        ST R7,RETURNresultS         ; Saves where resultS should return in RETURN
RETURNresultS
           ADD R1,R0,#0                    ; copy R0
           BRp  printPrime

           LEA R0,MSGNotPrime            ; Retrieves the address of the message to be written to
R0
           PUTS                          ; Prints the message string
           LD R7,RETURNresultS           ; set R7 back to isPrime  should return to
           RET ;


printPrime      LEA R0,MSGPrime             ; Retrieves the address of the message to be written
to R0
           PUTS                            ; Prints the message string
```

```
            LD R7,RETURNresultS                ; Sets R7 back to the resultS should return to
            RET                                ; JMP R7; Returns



;------------------------------------------------------------------- Variables ----------------------------------

RETURNreadS       .BLKW #1                    ; Saves the address to which the readS   must return
when it is finished.
RETURNisPrime      .BLKW #1                    ; Saves the address to which the isPrime must return
when it is finished.

RETURNresultS      .BLKW #1                   ; Saves the address to which the resultS must return
when it is finished.



PrimeNumber       .FILL #2                       ; place prime values at code lines
            .FILL #3
            .FILL #5
            .FILL #7
            .FILL #11
            .FILL #13
            .FILL #17
            .FILL #23
            .FILL #29
            .FILL #31
            .FILL #37
            .FILL #41
            .FILL #43
            .FILL #47
            .FILL #53
            .FILL #59
            .FILL #61
            .FILL #67
            .FILL #71
            .FILL #73
            .FILL #79
            .FILL #83
            .FILL #89
            .FILL #97
```

```
Message            .STRINGZ "Write a 2 digit decimal number: "
fASCII             .FILL #-48
MULTI              .FILL #9
MSGPrime           .STRINGZ "\nthe number is prime.     \n "
MSGNotPrime        .STRINGZ "\nthe number is not prime. \n "


                   .END

;---------------------------------------------  THE END ------------------------------------------------
```