8051 Assembly Assignment

Author: Ahmed Borchani

NEPTUN: EFKTV2

2021/2022

Task description:

Convert a hexadecimal number given by a variable-length ASCII character string in the internal memory (0..FFFF) to binary format.

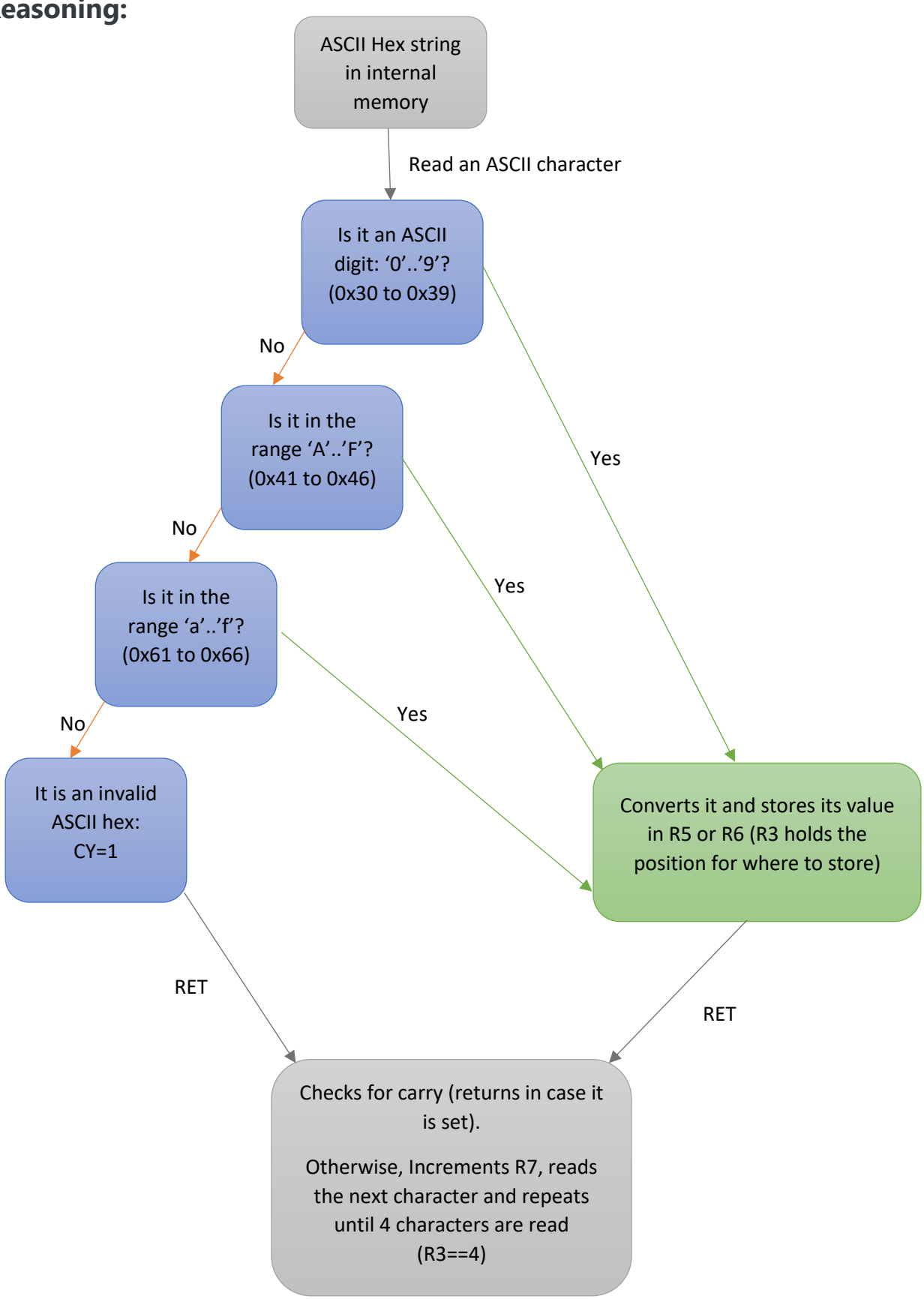
If an invalid ASCII character is detected in the input, indicate by CY=1.

Input: Start address of the string (pointer)

Output: Converted binary number in 2 registers, CY flag

-I declare that the following solution is my own work-

Reasoning:


ASCII Hex string in internal memory
Read an ASCII character
Is it an ASCII digit: '0'..'9'? (0x30 to 0x39)
No
Yes
Is it in the range 'A'..'F'? (0x41 to 0x46)
No
Yes
Is it in the range 'a'..'f'? (0x61 to 0x66)
No
Yes
It is an invalid ASCII hex: CY=1
Converts it and stores its value in R5 or R6 (R3 holds the position for where to store)
RET
RET
Checks for carry (returns in case it is set).
Otherwise, Increments R7, reads the next character and repeats until 4 characters are read (R3==4)

Functionality:

I used the following subroutines:

- ASCIIHEX_2_BIN:
  - This subroutine initializes R3 with 0x00 value. This register is needed to store the position of the valid, converted hex character into the output registers R5 and R6.

  - Then it calls STRING_CALLER subroutine which calls VALID_CONV subroutine, increments R7 (which holds the address for the next ASCII character) and gets into a loop until R3 is equal to 4. Which means that 4 characters have been read.

- VALID_CONV:
  - Checks if the character is in the range '0'..'9'. If so, it converts it, writes the result to R4, and jumps to WRITING to put the character in R5 or R6 (depending on R3 which holds the position.)
    ('0' corresponds to 0x30, and '9'=0x39. If this character is in range, when converting we simply subtract 0x30 from the hex value to get the digit in hex.)

  - Otherwise it jumps to INVALID_DIGIT which checks again and does the same with the range 'A'..'F'.
    (If this character is in this range, we have 'A'=0x41 so we subtract 0x41 then we add 0x0A to get the hex letter. We can simply subtract 0x37, but I used this approach for better clarity of the code.)
    If the character is not in the range 'A'..'F', it jumps to INVALID_UP_LETTER.

- INVALID_UP_LETTER checks if the character is in the range 'a'..'f' (If so, we have 'a'=0x61 so we subtract 0x61 then we add 0x0A to get the hex letter.)
  If the character is not in this range, it jumps to INVALID_LOW_LETTER. Which returns since the ASCII hex character is invalid. It returns with CY==1.

- WRITING: When jumping here, according to the value of R3 it jumps to one of the WRITING_CHARx code parts (x=1,2,3 or 4). Each WRITING_CHARx code part writes the converted hex byte in a specific position in R5 and R6, to get the result. Then it RETurns to STRING_CALLER subroutine.

➢ READ_CHAR:

- Since indirect addressing is not possible with register R7 we need to copy its value to register R0. Then using indirect addressing on the latter, we store the value from the memory into the register R2.