Algorithm explanation

- First continually reading the ASCII characters until a `\n` is found.
    - While reading, transform each ASCII to a decimal number.
    - Calculate: $N_{tot,new} = N_{tot,old} * 10 + N_{new}$.
- Convert the decimal number into hexadecimal, and each digit for a byte.
    - Get each digit by: $N\%16$(which means `N & #000f`), and $N = N/16$ for next digit.
- Output each digit after transforming them into ASCII.

Essential parts of your code with sufficient comments

```
; Read the Inpout Data and transform it from ASCII to decimal      (ignore most
problems)
; R2 act as the tmp register of input decimal result here.
; R3 act as the tmp register of total input decimal result here.
RDIN        TRAP    x20             ; Read in one char on screen.
            TRAP    x21             ; Print the input.
            STR     R0, R6, #0      ; Store the char read in. It also pass
arguments to function AtoD.
            JSR     incSP           ; Increase SP after store.
            ADD     R0, R0, #-10    ; Reach end? (ASCII = 10)
            ; Branch here           ;
            BRz     endRDIN         ; If zero, means equal, than exit.
            ; ASCII 2 decimal       ;
            JSR     AtoD            ; Calculate decimal here.
            JSR     decSP           ; Find return result.
            LDR     R2, R6, #0      ; Load return result.
            ; Add to R2             ;
            STR     R3, R6, #0      ; Pass arguments to function MulTen.
            JSR     incSP           ; Increase SP after store.
            JSR     MulTen          ;
            JSR     decSP           ; Find return result.
            LDR     R3, R6, #0      ; Load return result.
            ADD     R3, R3, R2      ; Add input number to total.
                                    ;
            JSR     RDIN            ; Continue readin.
            ; Exit of Loop:
endRDIN     ST      R3, tmpN_D      ; Store the decimal of input.

; Transform the decimal stored in tmpN_D into hexadecimal and store it to SC_Os.
            LD      R2, tmpN_D      ; Load the decimal of input.
DtoH        AND     R3, R2, x000f   ; R3 <- R2 mod 16.
            STR     R3, R5, #0      ; Push R3 into output Stack
            JSR     incoSP          ; Increase oSP after store.
            STR     R2, R6, #0      ; Pass arguments to fucntion RShift
            JSR     incSP           ; Increase SP after store.
            JSR     RShift16        ;
            JSR     decSP           ; Find return result.
            LDR     R2, R6, #0      ; Load the return result.
            BRz     endDtoH         ; If zero, than exit.
            JSR     DtoH            ; Continue to transform.
endDtoH     LEA     R5, tmpN_H      ; Reset oSP for hexadecimal.

; Output the things in output Stack one by one.
            JSR     ENDL            ; Print '\n' before output.
            LDR     R2, R5, #3      ; ___*
            JSR     OUTPUT          ;
            LDR     R2, R5, #2      ; __*_
            JSR     OUTPUT          ;
            LDR     R2, R5, #1      ; _*__
            JSR     OUTPUT          ;
            LDR     R2, R5, #0      ; *___
            JSR     OUTPUT          ;

; Process Codes Tail
EXIT        TRAP    x25
; ... with something else.
```

Questions TA asked you and your answer in Check

- Q: How to solve the problems without using Right Shift?
- A: Use masks `xf000` `x0f00` `x00f0` `x000f` to get the true value of each digit of hexadecimal. And for each result, iterate from `0000` to `1111` and check whether it fits the gotten value and then we get each digit's value.