Report of lab2

孟澍

3210101819

2022 年 7 月 18 日

1 Algorithm explanation

This is the flowchart of my program.

Basic idea: Store the number to be converted in R3, convert each 4-bit into hexadecimal then output.

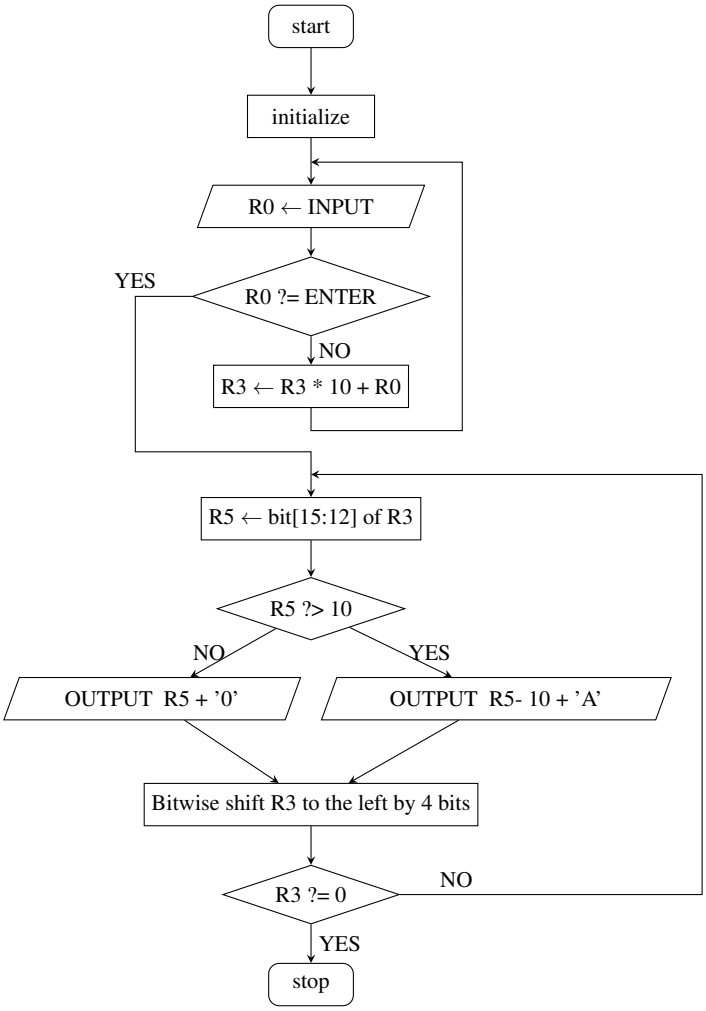

Fig 1: The flowchart of the program.

2 Source code

```
.orig x3000
; initialize
AND R0, R0, #0
AND R4, R4, #0
AND R5, R5, #0
AND R6, R6, #0

LD R1, ZERO
NOT R1, R1
ADD R1, R1, #1   ; R1 <- -'0'

LD R2, RETURN
NOT R2, R2
ADD R2, R2, #1   ; R2 <- -'\r'

AND R3, R3, #0   ; R3 store the number to be converted

INPUT GETC       ; R0 <- inputchar
OUT
ADD R6, R0, R2   ; test R0 ?= '\r'
BRz CONVERT
ADD R0, R0, R1
JSR MUL
BR INPUT

MUL AND R6, R6, #0
ADD R6, R6, #10
AND R5, R5, #0
ADD R5, R5, R3
AND R3, R3, #0
ADD R3, R3, R5       ; R3 <- R3 * 10
ADD R6, R6, #-1
BRp #-3
ADD R3, R3, R0       ; R3 <- R3 + R0
RET

CONVERT ADD R6, R6, #4
BR OUTPUT
CONT ADD R3, R3, R3
ADD R3, R3, R3
ADD R3, R3, R3
ADD R3, R3, R3
ADD R6, R6, #-1
BRp #-7
HALT

; get hexadecimal digit of the current highest 4-bit of R3
; the result is stored in R5
GETHEX LD R1, A
AND R2, R2, #0
ADD R2, R2, #1
AND R5, R5, #0
AND R4, R4, #0
AND R4, R1, R3
BRz #5
NOT R2, R2
NOT R5, R5
AND R5, R5, R2
NOT R5, R5
NOT R2, R2
ADD R2, R2, R2
ADD R1, R1, R1
BRz #1
```

```
BR #-12
RET

OUTPUT JSR GETHEX
ADD R5, R5, #-10
BRzp #4
LD R0, ZERO
ADD R0, R0, R5
ADD R0, R0, #10
BR #2
LD R0, CAPA
ADD R0, R0, R5
OUT
BR CONT

RETURN .fill x000A      ; ASCII code of ENTER
ZERO .fill x0030        ; ASCII code of '0'
CAPA .fill x0041   ; ASCII code of 'A'
LOWX .fill x0078    ; ASCII code of 'x'
A .fill x1000
.end
```

3 Questions TA asked you and your answer in check

Question: JSR 的具体流程是怎样的 Question: 在函数调用时如何保存之前寄存器的值

Answer: First, I initialize the registers. I set R0 into `0000 0000 0000 1111`, R1, R2, R3 into 0, and R4 into 12. I use R1 to store the number to be tested, use R2 to store tmp result, use R3 to count loop time, use R4 to store max loop time 12.

Then I load the word into R1.

Then, I store R0 AND R1 into R2. If now R2 - R0 = 0, it means there is a consecutive 4-bits of 1s, so I set R2 into 0 and stop the program. If R2 - R0 $\neq$ 0, it means there is no consecutive 4-bits of 1s so far, so I left shift R0 to test next 4 bits and add 1 to R3 to increase the loop time by 1.

Now I check whether I need to stop the loop. if R3 > R4, it means the loop needs to be terminated, so I terminiate the loop, set R2 into 0 and stop the program. If R3 $\leq$ R4, the loop continues, and the program goes back to the previous step.