EE 447-Introduction to Microprocessors

16.11.2016

Experiment-2

Preliminary Work

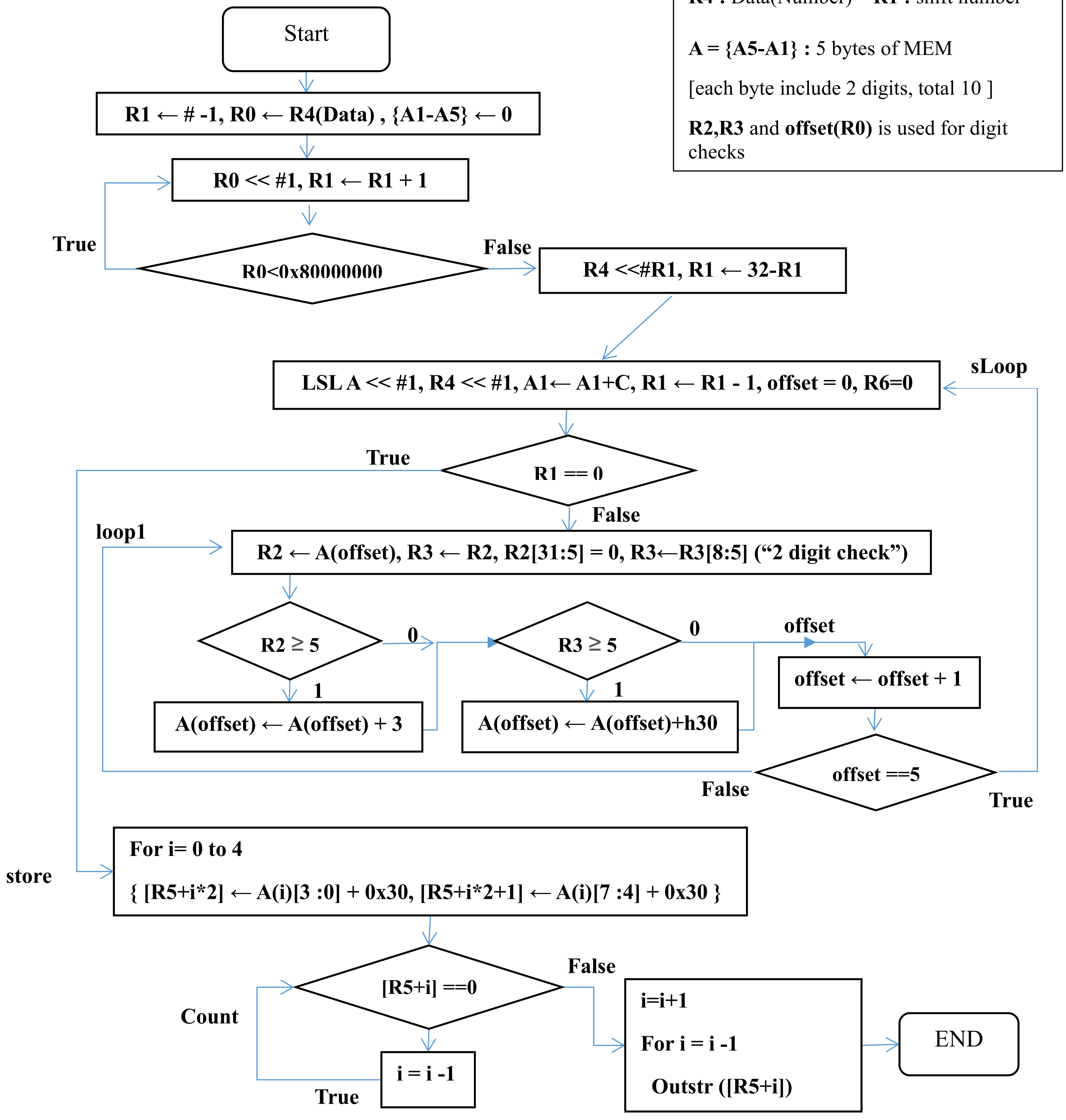
1) **CONVRT** Subroutine, **<u>Flow Chart</u>**

R4 : Data(Number) **R1 :** shift number

A = {A5-A1} : 5 bytes of MEM

[each byte include 2 digits, total 10]

R2,R3 and **offset(R0)** is used for digit checks


Start
R1 ← # -1, R0 ← R4(Data) , {A1-A5} ← 0
R0 << #1, R1 ← R1 + 1
True
R0<0x80000000
False
R4 <<#R1, R1 ← 32-R1
LSL A << #1, R4 << #1, A1← A1+C, R1 ← R1 - 1, offset = 0, R6=0
sLoop
True
R1 == 0
False
loop1
R2 ← A(offset), R3 ← R2, R2[31:5] = 0, R3←R3[8:5] ("2 digit check")
R2 ≥ 5
0
1
A(offset) ← A(offset) + 3
R3 ≥ 5
0
1
A(offset) ← A(offset)+h30
offset
offset ← offset + 1
offset ==5
False
True
store
For i= 0 to 4
{ [R5+i*2] ← A(i)[3 :0] + 0x30, [R5+i*2+1] ← A(i)[7 :4] + 0x30 }
[R5+i] ==0
Count
True
i = i -1
False
i=i+1
For i = i -1
Outstr ([R5+i])
END

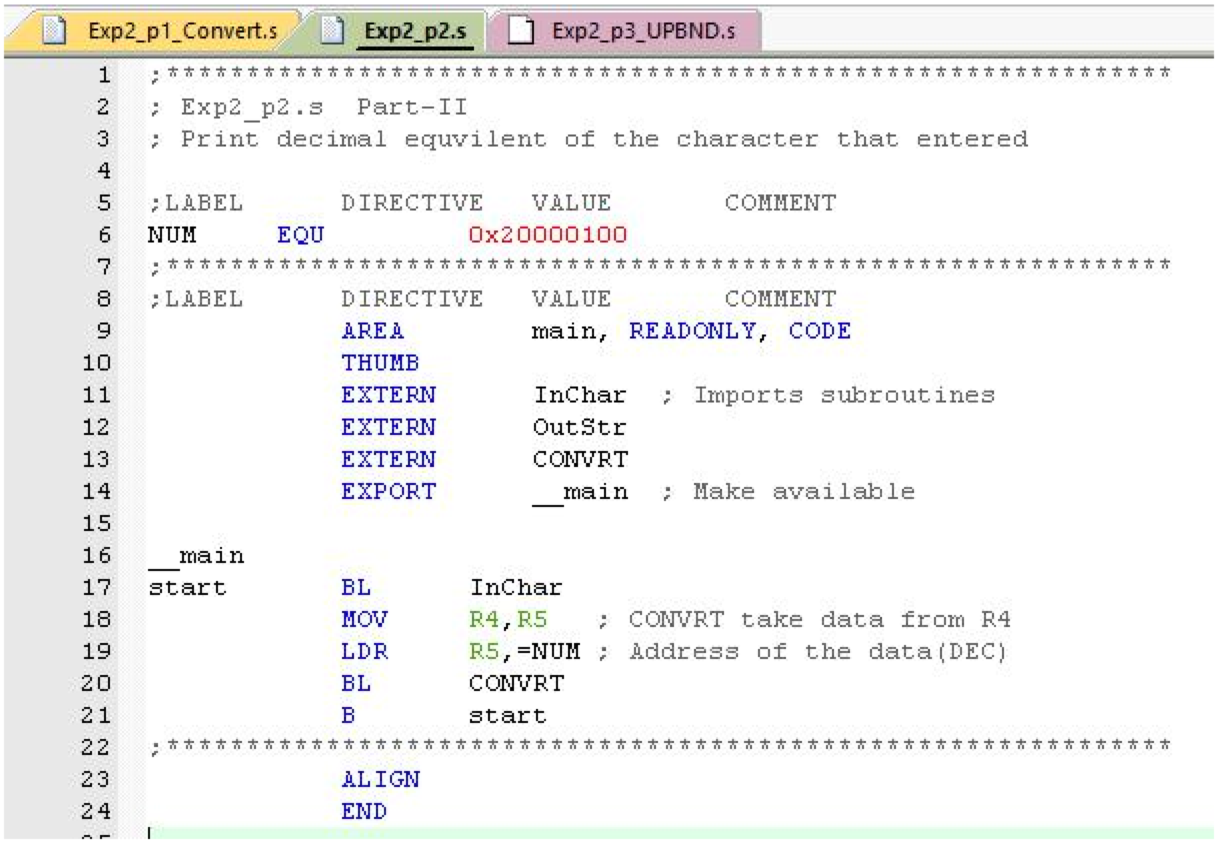
```
;*****************************************************************
; Experiment 2 ; Preliminary Work 1;
; Subroutine
; Converts an m-digit decimal number represented by n bits
;*****************************************************************

;LABEL      DIRECTIVE   VALUE       COMMENT
            AREA        routines, READONLY, CODE
            THUMB
            EXTERN      OutStr
            EXPORT      CONVRT
CONVRT
        PUSH{R0,R1,R2,R3,R4,R6,R7,LR}
        PUSH{R5}
        MOV     R0,#0
        MOV     R6,R5
        ;clear the work area
clear   STR     R0,[R6],#4  ;clear 2 words | A is cleared
        SUB     R1,R6,R5
        CMP     R1,#0x2C    ;28
        BNE     clear
        SUB     R6,#8
        MOV     R1,#-1
        MOV     R0,R4        ;  R0 temp
say     LSLS    R0,R0,#1
        ADD     R1,#1
        BCC     say     ; if C=0 stay in the loop
        LSL     R4,R1   ; R4'ü sola hizala
        RSB     R1,R1,#32

        ; 1 shift left
sLoop   LDRD    R2,R3,[R6]
        LSLS    R3,R3,#1    ; {S} used to clear PSR
        LSLS    R2,R2,#1
        ADCS    R3,#0       ;if carry exists add to R3 and Clear PRS
        LSLS    R4,R4,#1
        ADCS    R2,#0       ;if carry exists add to R2 and clear PRS
        STRD    R2,R3,[R6]

        MOV     R0,#0       ; R0 = offset
        MOV     R2,#0
        SUB     R1,R1,#1    ; R1 <= R1-1
        CMP     R1,#0
        BEQ     store

loop1   LDRB    R2,[R6,R0]
        MOV     R3,R2
        BFC     R2,#4,#28   ; get R2
        LSR     R3,#4       ; get R3

        CMP     R2,#5
        BLO     chk7
        LDR     r7,[R6,R0]
        ADD     r7,#0x00000003  ; ADDS?
        STR     r7,[R6,R0]
        CMP     R2,#13
        ADC     R3,#0
```

```
chk7    CMP     R3,#5
        BLO     offset
        LDR     r7,[R6,R0]
        ADD     r7,#0x00000030
        STR     r7,[R6,R0]

offset  ADD     R0,#1
        CMP     R0,#5
        BNE     loop1
        B       sLoop

store   MOV     R0,#0
        MOV     R2,#0
        MOV     R1,#4
Count   LDRB    R2,[R6,R1]
        MOV     R3,R2
        LSR     R3,#4       ; check Ax(7:4)
        CMP     R3,#0
        BNE     wLoop
        BFC     R2,#4,#28   ; check Ax(3:0)
        CMP     R2,#0
        ADDNE   R0,#1
        BNE     wLoop
        SUB     R1,R1,#1
        CMP     R1,#0xffffffff
        BEQ     exit
        B       Count

wLoop   LDRB    R2,[R6,R1]
        SUB     R1,R1,#1
        MOV     R3,R2
        LSR     R3,#4       ; get A1(7:4)
        CMP     R0,#1
        BEQ     pass
        ADD     R3,#0x30    ; get ASCII value
        STRB    R3,[R5],#1
pass    BFC     R2,#4,#28   ; get A1(3:0)
        ADD     R2,#0x30    ; get ASCII value
        STRB    R2,[R5],#1
        SUBEQ   R0,#1
        CMP     R1,#0xFFFFFFFF
        BNE     wLoop

        ; Output
        MOV     R0,#0x0D
        STRB    R0,[R5],#1
        MOV     R0,#0x04
        STRB    R0,[R5],#1
        POP{R5} ;load the address

        BL      OutStr
exit    POP{R0,R1,R2,R3,R4,R6,R7,LR}
        BX      LR

            ALIGN
            END
```

EE 447-Introduction to Microprocessors

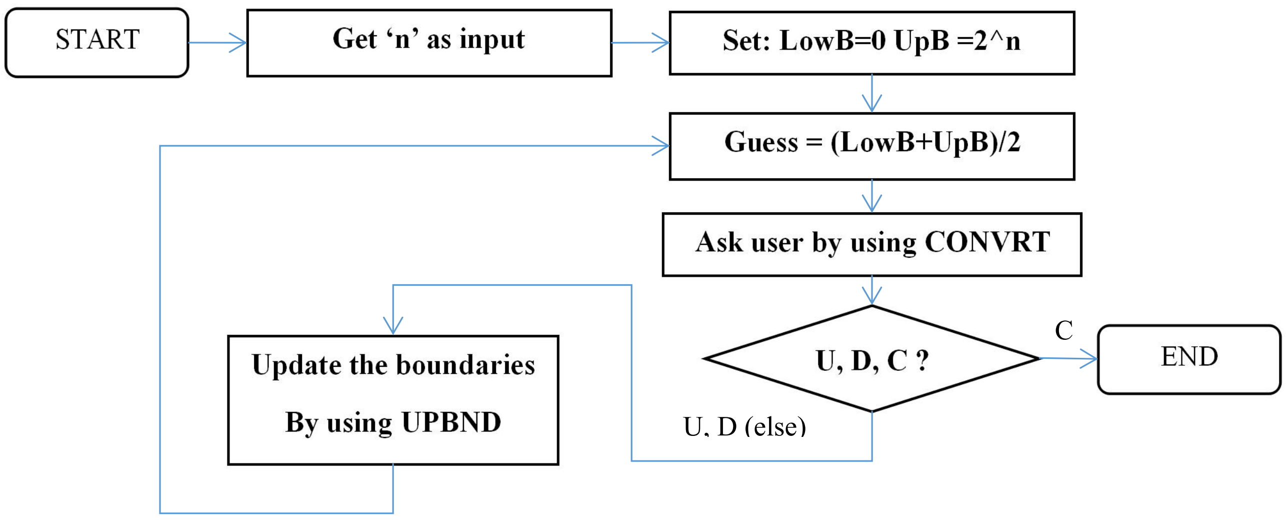
16.11.2016

2) InChar subroutine save the input data to R5. CONVRT subroutine use R5 register for addressing and R4 register for data. Therefore, it is needed to save R5 to R4 then load NUM to R5. Other registers which used in the CONVRT subroutine should be PUSH at the beginning of the CONVRT and should be returned back with POP command at the end of the subroutine.



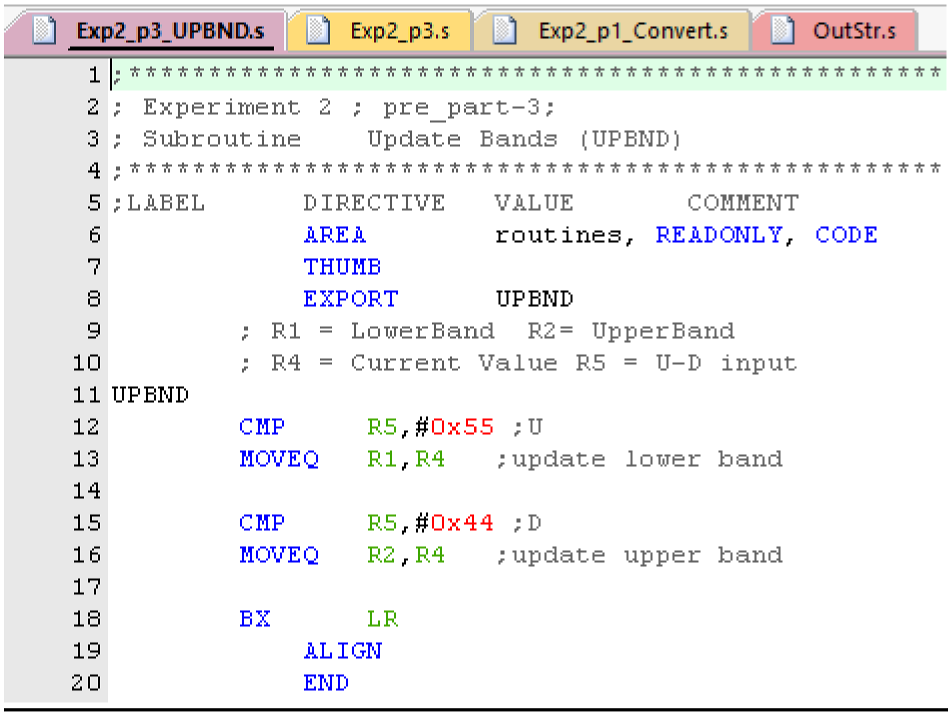
```
;*****************************************************************
; Exp2_p2.s  Part-II
; Print decimal equvilent of the character that entered

;LABEL      DIRECTIVE   VALUE       COMMENT
NUM      EQU         0x20000100
;*****************************************************************
;LABEL      DIRECTIVE   VALUE       COMMENT
            AREA        main, READONLY, CODE
            THUMB
            EXTERN      InChar  ; Imports subroutines
            EXTERN      OutStr
            EXTERN      CONVRT
            EXPORT      __main  ; Make available

__main
start       BL      InChar
            MOV     R4,R5   ; CONVRT take data from R4
            LDR     R5,=NUM ; Address of the data(DEC)
            BL      CONVRT
            B       start
;*****************************************************************
            ALIGN
            END
```

3) Number guess program **Flow Chart**



```
;****************************************************************
; Exp2_p2.s  Part-III
; Number guess game

ADDR        EQU         0x20000400
;****************************************************************
        AREA        sdata, DATA, READONLY
        THUMB
MSG     DCB         "Set the n value.."
        DCB         0x0D
        DCB         0x04
;****************************************************************
;LABEL  DIRECTIVE   VALUE       COMMENT
        AREA        main, READONLY, CODE
        THUMB
        EXTERN      InChar  ; Imports subroutines
        EXTERN      OutStr
        EXTERN      CONVRT
        EXTERN      UPBND
        EXPORT      __main

__main
start   MOV     R0,#10
        MOV     R3,#1   ;for right shift
        LDR     R5,=MSG
        BL      OutStr
        MOV     R5,#0
        BL      InChar
        SUB     R1,R5,#0x30 ; convert hex to DEC
        MUL     R1,R1,R0    ; 1st digit*10
        BL      InChar
        SUB     R0,R5,#0x30 ; hex to DEC
        ADD     R0,R1   ; R0 = n
        MOV     R1,#1   ; R1 = Lower Band Limit
        MOV     R2,#1
        LSL     R2,R0   ; R2 = Upper Band Limit

loop    ADD     R4,R1,R2
        LSR     R4,R3   ; R4 current guess
        LDR     R5,=ADDR
        BL      CONVRT      ; Out R4 value
        MOV     R5,#0
        BL      InChar      ; Get information
        CMP     R5,#0x43    ; C  check
        BEQ     done
        BL      UPBND   ; Update the band limits
        B       loop
done    B       start
            ALIGN
            END
```

16.11.2016



```
;****************************************************
; Experiment 2 ; pre_part-3;
; Subroutine    Update Bands (UPBND)
;****************************************************
;LABEL       DIRECTIVE   VALUE       COMMENT
             AREA        routines, READONLY, CODE
             THUMB
             EXPORT      UPBND
        ; R1 = LowerBand  R2= UpperBand
        ; R4 = Current Value R5 = U-D input
UPBND
        CMP     R5,#0x55 ;U
        MOVEQ   R1,R4   ;update lower band

        CMP     R5,#0x44 ;D
        MOVEQ   R2,R4   ;update upper band

        BX      LR
            ALIGN
            END
```