Lab 1

Data transfer & Arithmetic Programs

A

1. WAP to transfer block of 10 32-bit nos. from one memory to another

a. Source & destn. blocks are non over-lapping

CODE

```
    Area RESET,DATA,READONLY
    EXPORT __Vectors
__Vectors
     DCD 0x40001000
     DCD Reset_Handler
     ALIGN
    AREA mycode,CODE, READONLY
ENTRY
     EXPORT Reset_Handler
Reset_Handler
       LDR R0,=src      ;load add. of src
       LDR R1,=dst      ;load add. of dst
       LDR R3,=10       ; initialize loop counter
UP     LDR R2,[R0],#4 ; ~~Str~~ load data pointed by R0 into R2 using post-index
       STR R2,[R1],#4 ; store data from R2 in R1
       SUBS R3,#1     ; decrement counter
       BNE UP
 Stop b stop
 src DCD 0x1,0x2,0x3,0x4,0x5,0x6,0x7,0x8,0x9,0x10;define array
     AREA mydata, DATA, READwrite
dst DCD 0x0
    end
```

Output

Before exe

```
Address : 0x10000000 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                       00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                       00 00 00 00 00 00 00 00
```

After exe

```
0x10000000: 01 00 00 00  02 00 00 00  03 00 00 00  04 00 00 00
            05 00 00 00  06 00 00 00  07 00 00 00  08 00 00 00
            09 00 00 00  10 00 00 00
                          ↑        ↑
                     0x10000027 ───┘
```

b. source and destination are over-lapping

CODE

```
    AREA RESET, DATA, READONLY
    EXPORT  __Vectors
__Vectors
    DCD 0x40001000
    DCD Reset_Handler
    ALIGN
    AREA mycode, CODE, READONLY
ENTRY
    EXPORT Reset_Handler
Reset_Handler
    LDR R0,=src
    LDR R1,= src+4*10-4    ; load address of last ~~byte~~ no. in array (1st byte)
    LDR R2,= src+(4*10*2-4)-OLP ; load address of destination of last no.
    LDR R3 ,=10       ; initialize counter
UP  LDR  R4,[R1],#-4  ; load last no. and decrement ~~va~~R1 pointer
```

```
        STR R4, [R2], #-4   ; Store no. in R2 and decrement R2 pointer
        SUBS R3,1
        BNE UP          ; branch to top
stop b stop
      AREA mydata, DATA, READwrite
src  DCD  0x0    ; define both arry in data memory as we need to write
dst  DCD  0x0      in array
OLP  EQU  30     ; initialize overlapping b/w 2 arrays
      end
```

Output

Before exe

```
0x10000000 : 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16
             17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
             33 34 35 36 37 38 39 40 00 00 00 00 00 00 00 00
             00 00
                                      ↑
                                  0x10000027
```

After exe:

```
0x10000000: 01 02 03 04 05 06 07 08 09 10 01 02 03 04 05 06
            07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22
            23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38
            39 40
```

2. Reverse an array of 10 32-bit no. in memory

CODE

```
      AREA RESET, DATA, READONLY
      EXPORT __Vectors
__Vectors
```

```
    DCD 0x40001000
    DCD reset_Handler
    ALIGN
    AREA mycode, CODE, READONLY

ENTRY          Reset_
    EXPORT Handler

Reset_Handler
    LDR R0,=src              ; load address of first no. in array
    LDR R1,=src+4*10-4       ; load address of last no. in array
    LDR R3,=5                ; initialise counter

UP  LDR R4,[R0]
    LDR R5,[R1]                       first and last
    STR R4,[R1]              ; swap both the no. into each other locations
    STR R5,[R0]
    ADD R0,4                 ; increment first counter
    SUB R1,4                 ; decrement last counter
    SUBS R3,1
    BNE UP

stop b stop
    AREA mydata, DATA,READWrite
src DCD 0x0      ; define both in data memory as we need to write
dst DCD 0x0        on array
    end
```

OUTPUT

Before exe

```
0x10000000 :  00 00 00 01 00 00 00 02 00 00 00 03 00 00 00 04 00 00 00
              05 00 00 00 06 00 00 00 07 00 00 00 08 00 00 00 09 00 00
              00 10
```

After exe:

```
0x10000000: 00 00 00 10  00 00 00 09  00 00 00 08  00 00 00 07  00
            00 00 06  00 00 00 05  06 00 00 04  00 00 00 03  00 00
            00 02  00 00 00 01  ~~00 00 00 00~~
```

B

1. WAP to add 10 32-bit nos stored in code segment and store result in data segment

CODE:

```
    AREA RESET, DATA, READONLY
    EXPORT __Vectors
__Vectors
    DCD 0x40001000
    DCD Reset_Handler
    ALIGN
    AREA mycode, CODE, READONLY
ENTRY
    EXPORT Reset_Handler
Reset_Handler
      LDR R0,=Arr        ; pointer to Array
      MOV R3,#0
      LDR R4,=RES        ; pointer to destination
      MOV R2,#10         ; initialize counter
UP
      LDR R1,[R0],#4     ; load first value into R1
      ADDS R3,R1         ; Add all 10 nos. to R3 in each iteration
      ADC R6,#0          ; move carry in R6
      SUBS R2,#1
      BNE UP             ; Branch to top
      STR R3,[R4],#4     ; store res sum in R4
      STR R6,[R4]        ; store final carry in R4+1 location.
```

```
Stop b Stop
Arr DCD 0x1,0x2 ,0x3,0x4 ,0x5 ,0x6,0x7,0x8,0x9, 0xA
   AREA mydata, DATA, READwrite
Res DCD 0x0
   end
```

Output

Before exe:

```
0x16000000: 00 00 00 00 00 00 00 00
```

After exe:

```
0x10000000: 37 0000 00 00 00 0000
            ↑
           Sum
```

2. WAP to add 2 128 bit nos. stored in code segment and store the result in data segment

CODE

```
    AREA RESET, DATA, READONLY
    EXPORT __vectors
__vectors
    DCD 0x40001000
    DCD Reset_Handler
    ALIGN
    AREA mycode, CODE, READONLY
ENTRY
    EXPORT Reset_Handler
Reset_Handler
       LDR R0,=First     ; Load Addr. of first No.
       LDR R1,= Second   ; Load Addr of Second No
       LDR R2,=RES       ; pointer to destination
```

```
        LDR R3,=4          ; initialize counter to run 4 times
UP
        LDR R4,[R0],#4     ; load value of first 32-bit of both no. in R4 and R5
        LDR R5,[R1],#4
        ADCS R6,R4,R5      ; Add and Store in R6 with carry
        STR R6,[R2],#4     ; Store the result in R2 and move pointer
        SUB R3,1               to 4th location to store next 32 bits
        TEQ R3,#0          ; Chk if R3=0 without changing C flag
                              and if equal exit the loop
        BNE UP
Stop b stop
First DCD 0x01, 0x02, 0x03, 0x04
Second DCD 0x01, 0x02, 0x03, 0x04
   Area mydata, DATA, READwrite
RES DCD 0x0
    end
```

Output

before exe

```
0X10000000: 00 00 00 00  00 00 00 00  00 00 00  00  00 00 00 00
```

After exe:

```
0X10000000: 02000000  04000000  06000000  08000000
```

3. WAP to subtract 2 128 bit nos.

CODE

```
    AREA RESET, DATA, READONLY
    EXPORT __Vectors
__Vectors
    DCD 0x40001000
    DCD Reset_Handler
    ALIGN
    AREA mycode, CODE, READONLY
ENTRY
    EXPORT Reset_Handler
Reset_Handler
    LDR R0,=FIRST       ; Load Addr of first No.
    LDR R1,=SECOND      ; Load Addr of second No.
    LDR R2,=RES         ; pointer to destn.
    LDR R3,=4           ; initialize counter
    LDR R8,=0X60000000
    MSR xPSR, R8        ; Update the Carry flag to 1 as C=1 then
                          Borrow = 0 for 1st subtraction
UP
    LDR R4,[R0],#4      ; Load 32-bits of both Nos. in R4 and R5
    LDR R5,[R1],#4
    SBCS R6,R4,R5       ; Subtract R5 from R4 and store in R6 with
                          borrow
    STR R6,[R2],#4      ; Store result in R2 and inc. R2 pointer to
                          Store next 32 bits
    SUB R3,#1
    TEQ R3,#0           ; Chk if R3=0 and if equal exit the loop.
     BNE UP
stop b Stop
FIRST DCD  0X00000061,0X00006000,0X00000600,0X00000600
SEOND DCD  0X00000002,0X00000003,0X00000004 ,0X00000005
      AREA mydata, DATA ,READwrite
RES DCD 0X0
    end
```

output

before exe:

```
0X10000000: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

After exe:

```
0x10000060: FF FF FF FC FF FF FF F8 FF FF FF FA FF FF FF
```

Lab 2

Arithmetic Programs

1. Find sum of 'n' natural nos. using MLA instruction

CODE

```
    Area Reset, Data, READONLY
    EXPORT __Vectors
__Vectors
    DCD 0x40001000
    DCD Reset_Handler
    ALIGN
    AREA mycode, CODE, READONLY
ENTRY
     EXPORT Reset_Handler
Reset_Handler
    LDR R5=RES   ; pointer to destination
    LDR R6,= N   ; Load ~~pointer to address~~ value of N
    LDR R0,=1    ; constant in Equation
    LDR R1,= 1
    LDR R2,= 0   ; will contain sum of (i-1) Nos.
UP
    MLA R3,R0,R6,R2 ; R3 = R0xR6 + R2 add i to sum of (i-1)th Nos.
    ~~ADD R0,#1~~ MOV R2, R3 ; Load R2 with sum of (i-1) Nos.
    ~~STR R3,[R5]~~
    SUBS R6,#1
    BNE UP          ; branch till Z≠0
    STR R2,[R5]     ; Store result in R5
Stop b Stop
N EQU 5    ; EQU means constant
   Area mydata, DATA, READwrite
RES DCD 0x0
```

```
end
```

OUTPUT

Before exe:

```
0X10000000 : 00 00 00 00
```

After exe

```
0x10000000 : 0F 00 00 00
```

2. WAP to find GCD of 2 nos.

CODE

```
    AREA RESET, DATA, READONLY
    EXPORT __Vectors
__Vectors
    DCD 0X40001000
    DCD Reset_Handler
    ALIGN
    AREA mycode, CODE, READONLY
ENTRY
    EXPORT Reset_Handler
Reset_Handler
        LDR R0,=NUM1    ; pointer to first no.
        LDR R1,=NUM2    ; pointer to second no.
        LDR R0,[R0]     ; load both nos. in R0 and R1
        LDR R1,[R1]
UP
        CMP R0,R1       ; cmp R0 and R1, loop will run till a!=b
        BEQ EXIT        ; if equal exit and ~~R0~~ first No. will contain GCD, store it in R2
        SUBHI R0,R1     ; Subtract R0 from R1 or R1 from R0, which ever is
        SUBLO R1,R0       greater
        B UP            ; branch to cmp
EXIT    LDR R2,=GCD
        STR R0,[R2]
```

```
stop b stop
NUM1 DCD 8
NUM2 DCD 4
  AREA mydata, DATA, Readwrite
GCD DCD 0
  end
```

OUTPUT:

```
0x10000000 : 04 00 00 00
```

3. WAP to find LCM of 2 nos.

CODE

```
  AREA RESET, DATA, READONLY
  EXPORT __Vectors
__Vectors
  DCD 0x40001000
  DCD Reset_Handler
  ALIGN
  Area mycode, CODE, READONLY
ENTRY
  EXPORT Reset_Handler
Reset_Handler
    MOV R0,#3    ; load value of first no. (a)
    MOV R5,#2    ; load value of second no. (b)
    MOV R3,#0
    MOV R8,#0    ; store 0
    MOV R6,#1    ; initialise i to 1
UP  MUL R4,R0,R6 ; remainder = a*i
    BL MOD       ; call MOD function to find remainder
```

```
        CMP R4,R8        ; compare remainder with 0
        BEQ EXIT         ;if equal exit loop
        ADD R6,#1        ;increment value of i
        B op             ;branch to top
EXIT    MUL R6,R6,R0     ; Multiply i*a to get LCM
stop b stop
MOD CMP R4,R5          ; perform division by repeated subtraction
        BCS LABEL1         to find remainder and store it in R4
        BX LR          ;return back to main
LABEL1 SUB R4,R5
        B MOD
        Area mydata, DATA, Readwrite
        end
```

Output

~~0x0000000~~R6: 0x00000006

Lab 3

Code Conversion

1. WAP to convert 2-digit hexadecimal no. in ASCII.

CODE

```
    LDR R0,=NUM          ; pointer to hexadecimal no.
    LDR R3,=RES          ; pointer to result
    LDR R1,[R0]          ; load the value
    AND R2,R1,0X0000000F ; mask upper 4 bits
    CMP R2,#09           ; compare digit with 9
    BCC DOWN             ; if its is lower then 9 then jump to down
    ADD R2,#07           else add 07 to that no.
DOWN
    ADD R2,#0x30         ; add 30h to the no., ascii value of first digit
    STRB R2,[R3],#1      ; store in R3 and inc R3 ptr. by 1
    AND R2,R1,0X000000F0 ; mask lower 4 bits
    LSR R2,#04           ; shift right by 4 bits
    CMP R2,#09           ;
    BCC DOWN1            ; repeat the same process
    ADD R2,#07
DOWN1
    ADD R2,# 0X30
    STRB R2,[R3]  ;
stop b stop
NUM DCD 0X0000003A
    Area mydata, DATA, READwrite
RES DCD 0
    end
```

Output

```
0x10000000 : 41 33 00 00
```

2. WAP to ~~find BCD of 2 nos~~ convert a 2-digit BCD no. to its equivalent hexadecimal

CODE

```
    LDR R0,=NUM        ; load the addr of number
    LDRB R6, [R0]      ; load the first byte (8 bits) of No. in R6
    LDR R3,=RES        ; pointer to destination
    AND R1,R6,#0x0F    ; mask the upper 4 bits
    AND R2,R6,#0xF0    ; mask the lower 4 bits
    LSR R2,#4          ; shift right by 4 bits
    MOV R4,#10
    MLA R5,R2,R4,R1 ; upper 4 bits x 4 + lower 4 bits, Store it in the first byte from R5
    STRB R5,[R3]       ; ~~load~~ store R5, the result in R3

stop b stop
NUM DCD 0x56
    Area mydata, DATA,READwrite
RES DCD 0
    end
```

Output

```
0x10000000 : 38 00 00 00
```

3. WAP to convert a 2 digit hex no. to its equivalent BCD no.

CODE

```
    LDR R0,=NUM       ; pointer to number
    LDR R6,=RES       ; pointer to destination
    LDR R0,[R0]       ; load value in R0
    MOV R5,#01
UP2 CMP R0,#0         ; Compare value with 0
    BEQ stop          ; if equal jump to stop
    BL DIV            ; divide value by 10 in DIV function
    MUL R3,R5         ; multiply remainder with powers of 10
    STRB R3,[R6],#1   ; Store value(res) in R6
    MOV R0,R2         ; mov quotient to R0
    MOV R2,#00
    B UP2             ; again compare the quotient with 0 and repeat
                        the process
stop b stop
DIV
    CMP R0,#10        ; divide by 10 and store remainder in R3 and
    BCC DOWN               quotient in R2.
    SUB R0,#10        ; divide by repeated subtraction
    ADD R2,#1
    B DIV
DOWN
    MOV R3,R0
    BX LR             ; jump back to main function
NUM DCD 0XAA
    AREA mydata,DATA,READwrite
RES DCD 0
    end
```

OUTPUT

```
0x10000000:00 07 01 3→170
```