LAB 1

LAB 1 SOLVED EXERCISE:

Write an ARM assembly language program to copy 32 bit data from code memory to data memory.

```
        AREA Reset, DATA, READONLY
        EXPORT __Vectors
__Vectors
        DCD 0x10001000
        DCD Reset_Handler
        ALIGN
        AREA mycode, CODE, READONLY
        ENTRY
        EXPORT Reset_Handler

Reset_Handler
        LDR R0, =SRC
        LDR R1, =DST
        LDR R3, [R0]
        STR R3, [R1]
STOP
        B STOP
SRC DCD 8
        AREA mydata, DATA, READWRITE
DST DCD 0
        END
```

.....

LAB 1 Lab Exercises:

1. **Write an ARM assembly language program to transfer a 32 bit number from one location in the data memory to another location in the data memory.**

```
        AREA Reset, DATA, READONLY
        EXPORT __Vectors
__Vectors
```

```
        DCD 0x10001000
        DCD Reset_Handler
        ALIGN
        AREA mycode, CODE, READONLY
        ENTRY
        EXPORT Reset_Handler

Reset_Handler
        LDR R0, =SRC
        LDR R1, =DST
        LDR R3, [R0]
        STR R3, [R1]
STOP
        B STOP

        AREA mydata1, DATA, READONLY
SRC DCD 8
        AREA mydata2, DATA, READWRITE
DST DCD 0
        END
```

.....

LAB 1 Lab Exercises:

2. **A)_ Write an ARM assembly language program to transfer block of ten 32 bit numbers from one location to another when the source and destination blocks are non-overlapping (from code memory to data memory)**

```
        AREA Reset, DATA, READONLY
        EXPORT __Vectors
__Vectors
```

```
        DCD 0x10001000
        DCD Reset_Handler
        ALIGN
        AREA mycode, CODE, READONLY
        ENTRY
        EXPORT Reset_Handler

Reset_Handler
        LDR R0, =SRC
        LDR R1, =DST
        MOV R3, #10

UP
        LDR R4, [R0], #4
        STR R4, [R1], #4
        SUB R3, #1
        BNE UP
STOP
        B STOP


SRC DCD 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
        AREA mydata2, DATA, READWRITE
DST DCD 0
        END
```

2. B)_ Write an ARM assembly language program to transfer block of ten 32 bit numbers from one location to another when the source and destination blocks are overlapping.

```
        AREA RESET,DATA,READONLY
```

```
    EXPORT __Vectors

__Vectors
    DCD 0x10001000
    DCD Reset_Handler
    ALIGN
    AREA mycode,CODE,READONLY
    ENTRY
    EXPORT Reset_Handler

Reset_Handler
    LDR R0,=SRC
    MOV R1,#10
    MOV R3,#1
LOOP
    STR R3,[R0],#4
    ADD R3,#1
    SUBS R1,#1
    BNE LOOP

    LDR R0,=SRC
    ADD R0,R0,#(SIZE-1)*4
    SUB R1,R0,#(OL-1)*4
    ADD R1,R1,#(SIZE-1)*4
    MOV R3,#10
UP
    LDR R4,[R0],#-4
    STR R4,[R1],#-4
```

```
    SUBS R3,#1
    BNE UP
STOP
    B STOP

SIZE EQU 10
OL EQU 2
    AREA myData,DATA,READWRITE
SRC DCD 0
    END
```

.....

3. Reverse an array of ten 32 bit numbers in the memory.

```
AREA RESET,DATA,READONLY
EXPORT __Vectors

__Vectors
DCD 0x10001000
DCD Reset_Handler
ALIGN
AREA mycode,CODE,READONLY
ENTRY
EXPORT Reset_Handler

Reset_Handler
LDR R0,=SRC
MOV R1,#10
MOV R3,#1
LOOP
```

```
STR R3,[R0],#4
ADD R3,#1
SUBS R1,#1
BNE LOOP

LDR R0,=SRC
MOV R5,R0
ADD R0,R0,#(SIZE-1)*4
;SUB R1,R0,#(OL-1)*4
;ADD R1,R1,#(SIZE-1)*4
MOV R3,#5
UP
LDR R6,[R5]
LDR R7,[R0]
STR R7,[R5],#4
STR R6,[R0],#-4
SUBS R3,#1
BNE UP
STOP
B STOP

SIZE EQU 10
;OL EQU 2
AREA myData,DATA,READWRITE
SRC DCD 0
END
```

LAB 2

LAB NO 2 Solved Exercise:

Write a program to add two 32 bit numbers available in the code memory. Store the result in the data memory.

```
    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, =VALUE1
    LDR R1, [R0]
    LDR R0, =VALUE2
    LDR R3, [R0]
    ADDS R6, R1, R3
    LDR R2, =RESULT
    STR R6, [R2]

STOP
    B STOP

VALUE1 DCD 0X12345678
VALUE2 DCD 0XABCDEF12
```

```
    AREA mydata, DATA, READWRITE
RESULT DCD 0


    END
```

.....

LAB 2 Lab Exercises:

1. **Write a program to add ten 32 bit numbers available in code memory and store the result in data memory.**

```
        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, #0
                LDR R1, =NUM
                MOV R2, #10

LOOP
        LDR R3, [R1], #4
        ADD R0, R0, R3
        SUBS R2, R2, #1
        BNE LOOP

        LDR R1, =RES
        STR R0, [R1]
STOP
        B STOP

NUM DCD 0x01010101, 0x12121212, 0x23232323, 0x34343434, 0x45454545,
0x56565656, 0x67676767, 0x78787878, 0x89898989, 0x90909090
```

```
        AREA mydata, DATA, READWRITE
RES DCD 0
        END
```

2. **Write a program to add two 128 bit numbers available in code memory and store the result in data memory.**

```
        AREA Reset, DATA, READONLY
        EXPORT __Vectors
__Vectors
        DCD 0x40001000
        DCD Reset_Handler
        ALIGN
        AREA mycode, CODE, READONLY
        ENTRY
        EXPORT Reset_Handler

Reset_Handler
                LDR R1, =NUM1
                LDR R2, =NUM2
                LDR R3, =RESULT
                MOV R4, #4

LOOP
        LDR R5, [R1], #4
        LDR R6, [R2], #4
        ADC R7, R5, R6
        STR R7, [R3], #4
        SUBS R4, R4, #1
        BNE LOOP

STOP
        B STOP

NUM1 DCD 0x11223344, 0x55667788, 0x99AABBCC, 0xDDEEFF00
NUM2 DCD 0x12345678, 0x3456789A, 0xABCDEF12, 0x08624472

        AREA mydata, DATA, READWRITE
RESULT DCD 0
        END
```

3. **Write a program to subtract two 32 bit numbers available in the code memory and store the result in the data memory.**

```
        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, =VALUE1
        LDR R1, [R0]
        LDR R0, =VALUE2
        LDR R3, [R0]
        SUBS R6, R1, R3
        LDR R2, =RESULT
        STR R6, [R2]

STOP
        B STOP

VALUE1 DCD 0X12345678
VALUE2 DCD 0XABCDEF12

        AREA data, DATA, READWRITE
RESULT DCD 0
        END
```

4. **Write a program to subtract two 128 bit numbers available in the code memory and store the result in the data memory.**

```
        AREA Reset, DATA, READONLY
        EXPORT __Vectors
__Vectors
        DCD 0x40001000
        DCD Reset_Handler
        ALIGN
        AREA mycode, CODE, READONLY
        ENTRY
        EXPORT Reset_Handler
```

```
Reset_Handler
                LDR R1, =NUM1
                LDR R2, =NUM2
                LDR R3, =RESULT
                MOV R4, #4

LOOP
        LDR R5, [R1], #4
        LDR R6, [R2], #4
        SBC R7, R5, R6
        STR R7, [R3], #4
        SUBS R4, R4, #1
        BNE LOOP

STOP
        B STOP

NUM1 DCD 0x11223344, 0x55667788, 0x99AABBCC, 0xDDEEFF00
NUM2 DCD 0x12345678, 0x3456789A, 0xABCDEF12, 0x08624472

        AREA mydata, DATA, READWRITE
RESULT DCD 0
        END
```

EXTRA PROGRAMS

Write an assembly program to perform addition of 10 natural numbers.

```
AREA RESET, DATA, READONLY
EXPORT __Vectors
__Vectors
DCD 0x40001000 ; stack pointer value when stack is empty
DCD Reset_Handler ; reset vector
ALIGN
AREA mycode, CODE, READONLY
ENTRY
EXPORT Reset_Handler
Reset_Handler
  LDR r0, =result     ; Load the address where the result will be stored
MOV r1, #10
MOV r2, 0
```

```
SUM
ADD r2, r2, r1
SUBS r1, r1, #1
BNE SUM
MOV r3,r2
  STR r3, [r0]

  ; The result is now stored in the 'result' memory location

  ; Halt the program
STOP   B STOP


AREA result1, DATA, READWRITE
result  DCD 0

END
```

.....

LAB 3

LAB NO 3: Solved Exercise

Write an assembly program to multiply two 32 bit numbers.

```
AREA RESET, DATA, READONLY
        EXPORT __Vectors
__Vectors
        DCD 0x40001000          ; stack pointer value when stack is empty
        DCD Reset_Handler       ; reset vector
        ALIGN
        AREA mycode, CODE, READONLY
        ENTRY
        EXPORT Reset_Handler

Reset_Handler

        LDR R1, =VALUE1         ;pointer to the first value1
        LDR R5, [R1]
        LDR R2, =VALUE2         ;pointer to the second value
        LDR R6, [R2]
        UMULL R3, R4, R5, R6    ;Multiply the values from R1 and R2 and store
```

```
                                    ;least significant 32 bit number into R3 and
                                    most
                                    ;significant 32 bit number into R4.
        LDR R2, =RESULT
        STR R4, [R2]
        ADD R2, #4
        STR R3, [R2]                ; store result in memory
STOP
        B STOP

VALUE1 DCD 0X54000000               ; First 32 bit number
VALUE2 DCD 0X10000002               ; Second 32 bit number

        AREA data, DATA, READWRITE
RESULT DCD 0

END
```

LAB NO 3: Lab Exercises

1. **Write a program to multiply two 32 bit numbers using repetitive addition**

```
        AREA RESET, DATA, READONLY
        EXPORT __Vectors
    __Vectors
        DCD 0x10001000
        DCD Reset_Handler
        ALIGN
        AREA mycode, CODE, READONLY
        ENTRY
        EXPORT Reset_Handler

    Reset_Handler

        LDR R1, =VALUE1
        LDR R5, [R1]
        LDR R2, =VALUE2
        LDR R6, [R2]
        MOV R3, #0
        MOV R7, #0
        LDR R4, =RESULT
        LDR R8, =CARRY
```

```
LOOP    ADDS R3, R3, R5
        ADC R7, R7, #0
        SUBS R6, R6, #1
        BNE LOOP

        STR R3, [R4]
        STR R7, [R8]
STOP
    B STOP

VALUE1 DCD 0X78000000
VALUE2 DCD 0X00000004

    AREA data, DATA, READWRITE
RESULT DCD 0
CARRY DCD 0

END
```

Repeat the above program for BCD multiplication.

2. Find the sum of 'n' natural numbers using MLA instruction.

```
    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,#VALUEN
    MOV R1, #0
    MOV R3, #1
    LDR R4, =RESULT

LOOP  MLA R1, R1, R3, R0
        SUBS R0, R0, #1
```

```
        BNE LOOP

    STR R1, [R4]
STOP
    B STOP

VALUEN EQU 10

    AREA data, DATA, READWRITE
RESULT DCD 0

END
```

3. **Write an assembly language program to find GCD of two numbers**

Hint:

```
While(a!=b)
{
        If(a>b)
         a=a-b;
        else
        b=b-a;
} Return (a);
```

```
        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, =VALUE1
        LDR R1, [R0]
        LDR R2, =VALUE2
        LDR R3, [R2]

        LDR R4, =RESULT

LOOP
        CMP R1, R3
        BEQ DONE

        BGT GREATER
        SUBS R1, R3, R1
        B LOOP

GREATER
        SUBS R3, R1, R3
        B LOOP

DONE
        STR R1, [R4]

STOP
        B STOP

VALUE1 DCD 0x0000000C
VALUE2 DCD 0x00000006
```

```
        AREA data, DATA, READWRITE
RESULT DCD 0


        END
```

4. **Write an assembly language program to find LCM of two numbers.**

LAB 4

LAB NO 4: Lab Exercises

Write an assembly program to convert a 2-digit hexadecimal number into unpacked ASCII.

```
AREA RESET, DATA, READONLY
 EXPORT __Vectors
__Vectors
 DCD 0x40001000                ; stack pointer value when stack is empty
 DCD Reset_Handler             ; reset vector
 ALIGN
AREA mycode, CODE, READONLY
ENTRY
EXPORT Reset_Handler
Reset_Handler
LDR R0,=NUM
LDR R3,=RESULT
 LDRB R1,[R0]                  ; load hex number into register R1
AND R2,R1,#0x0F                ; mask upper 4 bits
CMP R2,#09                     ; compare the digit with 09
BLS DOWN                       ; if it is lower than 9 then jump to down label
ADD R2,#07                     ;else add 07 to that number
DOWN
```

```
ADD R2,#0x30            ; Add 30H to the number, Ascii value of first digit
STRB R2,[R3]
AND R4,R1,#0xF0         ; Mask the second digit
MOV R4,R4,LSR#04        ; Shift right by 4 bits
CMP R4,#09              ; check for >9 or not
BLS DOWN1
ADD R3,#07
DOWN1
ADD R4,#0x30            ; Ascii value of second digit
STRB R4,[R3,#01]
NUM DCD 0x000003A
AREA data, DATA, READWRITE
RESULT DCD 0
 END
```

ASCII to packed BCD conversion

```
AREA RESET, DATA, READONLY
 EXPORT __Vectors
__Vectors
 DCD 0x40001000              ; stack pointer value when stack is empty
 DCD Reset_Handler           ; reset vector
 ALIGN
AREA mycode, CODE, READONLY
ENTRY
EXPORT Reset_Handler
Reset_Handler
MOV R1,#0x37                 ;R1 = 0x37
MOV R2,#0x32                 ;R2 = 0x32
AND R1,R1,#0x0F              ;mask 3 to get unpacked BCD
```

```
AND R2,R2,#0x0F                 ;mask 3 to get unpacked BCD
MOV R3,R2,LSL #4                ;shift R2 4 bits to left to get R3 = 0x20
ORR R4,R3,R1                    ;OR them to get packed BCD, R4 = 0x27
STOP
        B STOP
END
```

Packed BCD to ASCII conversion

```
EXPORT __Vectors
__Vectors
 DCD 0x40001000                 ; stack pointer value when stack is empty
 DCD Reset_Handler              ; reset vector
 ALIGN
AREA mycode, CODE, READONLY
ENTRY
EXPORT Reset_Handler
Reset_Handler
MOV R0,#0x29
AND R1,R0,#0x0F                 ;mask upper four bits
ORR R1,R1,#0x30                 ;combine with 30 to get ASCII
MOV R2,R0,LSR #04               ;shift right 4 bits to get unpacked BCD
ORR R2,R2,#0x30                 ;combine with 30 to get ASCII
STOP
        B STOP
END
```