Week1

1. **@Add and Subtract 2 numbers in the register**

.text

mov r0, #13

mov r1, #12

add r2, r0, r1

sub r3, r0, r1

.end

1. **@Demonstrate logical operations**

.text

mov r0, #15

mov r1, #10

and r2, r0, r1

orr r3, r0, r1

eor r4, r0, r1

bic r5, r0, r1

mvn r6, r0

.end

1. **@Add 5 numbers where values are present in registers**

.text

mov r0, #1

mov r1, #2

mov r2, #3

mov r3, #4

mov r4, #5

add r5, r0, r1

add r5, r5, r2

add r5, r5, r3

add r5, r5, r4

.end

1. **@Check if a number is even or odd**

.text

mov r0, #5

ands r1, r0, #1 ; r1 = 1 if odd else r1 = 0

beq L1

mov r2, #0xff

b L2

L1: mov r2, #0x00 ; r2 = 0 if even else r2 = 255

L2: SWI 0x011

.end

1. **@Compare the value of R0 and R1, add if R0 = R1, else subtract**

.text

mov r0, #10

mov r1, #15

cmp r0, r1

beq L1

sub r3, r1, r0; store the difference in r3

b L2

L1: add r2, r1, r0 ; store the sum in r2

L2: SWI 0x011

.end

1. **@Check GCD of two numbers without LDR**

.text

mov r0, #10

mov r1, #15

mov r2, r0

mov r3, r1

gcd: cmp r2, r3

beq close

blt less

subs r2, r2, r3

B gcd

less: subs r3, r3, r2

b gcd

close: SWI 0x011

.end

1. **@Find the factorial of a number**

.text

mov r0, #5

mov r1, r0

fact: sub r0, r0, #1

mul r2, r1, r0

mov r1, r2

cmp r0, #1

bgt fact

close: SWI 0x011

.end

Week 2

1. **@Find the factorial of a number stored in R0. Store the value in R1**

.text

mov r0, #5

mov r2, r0

mov r1, r0

fact: sub r2, r2, #1

mul r3, r1, r2

mov r1, r3

cmp r2, #1

bne fact

close: SWI 0x011

.end

1. **@Based on the value of the number in R0 :**

**@ Store 1 in R1 if R0 is zero**

**@ Store 2 in R1 if R0 is positive**

**@ Store 3 in R1 if R0 is negative**

.text

mov r0, #-1

cmp r0, #0

blt less

bgt great

mov r1, #1

b close

less: mov r1, #3

b close

great: mov r1, #2

close: SWI 0x011

.end

1. **@Add two numbers loaded from memory and store the result in memory**

.data

a: .word 10

b: .word 9

c: .word 0

.text

ldr r0, =a

ldr r1, =b

ldr r2, =c

ldr r0, [r0]

ldr r1, [r1]

add r3, r0, r1

str r3, [r2]

.end

1. **@To copy n numbers from Location A to Location B**

.data

a: .word 10, 20, 30, 40

b: .word 0, 0, 0, 0

.text

ldr r6, =b

ldr r0, =a

ldr r1, =b

mov r3, #4

L1: ldr r4, [r0], #4

str r4, [r1], #4

subs r3, r3, #1

bne L1

.end

1. **@Find the GCD of given numbers loaded from memory**

.text

ldr r0, =a

ldr r1, =b

ldr r0, [r0]

ldr r1, [r1]

mov r2, r0

mov r3, r1

gcd: cmp r2, r3

beq close

blt less

subs r2, r2, r3

B gcd

less: subs r3, r3, r2

b gcd

close: SWI 0x011

.data

a: .word 10

b: .word 15

.end

1. **@Add an array of numbers from memory**

.data

a: .word 10, 20, 30, 40, 50

.text

ldr r0, =a

mov r1, #5

mov r3, #0

L1: ldr r2, [r0], #4

add r3, r3, r2

subs r1, r1, #1

cmp r1, #0

beq close

b L1

close: SWI 0x011

.end

Week 3

1. **@Add array of 10 numbers taking data from memory location**

**stored as byte data (use .byte to store the data instead of .word)**

.data

a: .byte 1, 2, 3, 4, 5, 6, 7, 8, 9, 0

.text

ldr r0, =a

mov r1, #10

mov r3, #0

L1: ldrb r4, [r0], #1

add r3, r3, r4

subs r1, r1, #1

cmp r1, #0

beq close

b L1

close: SWI 0x011

.end

1. **@Linear Search,Check whether the given number is present in array**

**@if found move it’s position to R7**

**@else move -1 to R6 (If the number not found)**

.data

a: .word 10, 20, 30, 40, 50

b: .word 1

.text

ldr r0, =a

ldr r1, =b

ldr r1, [r1]

mov r3, #5

L1:

cmp r3, #0

beq notFound

ldr r2, [r0], #4

cmp r2, r1

beq Found

subs r3, r3, #1

b L1

Found:

sub r7, r3, #5

b close

notFound:

mov r6, #-1

close: SWI 0x11

1. **@Generate Fibonacci Series**

.data

fib: .word 0, 1

n: .word 10

.text

ldr r0, =fib

ldr r1, =n

ldr r1, [r1]

genFib:

ldr r2, [r0], #4

ldr r3, [r0]

add r2, r2, r3

add r3, r0, #4

str r2, [r3]

subs r1, r1,#1

cmp r1, #0

bne genFib

SWI 0x11

.end

1. **Convolution:** 
   1. **@Perform convolution using MUL**

.data

a: .word 10, 20, 30, 40

b: .word 10, 20, 30, 40

.text

ldr r0, =a

ldr r1, =b

mov r2, #4

mov r5, #0

Convol:

ldr r3, [r0], #4

ldr r4, [r1], #4

mul r6, r3, r4

add r5, r5, r6

subs r2, r2, #1

cmp r2, #0

bne Convol

SWI 0x11

* 1. **@Perform convolution using MLA**

.data

a: .word 10, 20, 30, 40

b: .word 10, 20, 30, 40

.text

ldr r0, =a

ldr r1, =b

mov r2, #4

mov r5, #0

Convol:

ldr r3, [r0], #4

ldr r4, [r1], #4

mla r5, r3, r4, r5

subs r2, r2, #1

cmp r2, #0

beq close

b Convol

close: SWI 0x11

Week 4

1. **@Add 64bit numbers**

.data

a: .word 12312312, 42123123

b: .word 98798798, 23523122

c: .word 0

.text

ldr r0, =a

ldr r1, =b

ldr r3, =c

ldr r4, [r0], #4

ldr r5, [r1], #4

add r4, r4, r5

str r4, [r3], #4

ldr r4, [r0]

ldr r5, [r1]

add r4, r4, r5

str r4, [r3]

SWI 0x011

.end

1. **@Find smallest number in an array**

.data

a: .word 1, 2, 3, 4, 5, 6

.text

ldr r0, =a

mov r1, #5

ldr r2, [r0], #4

loop:

cmp r1, #0

beq close

subs r1, r1, #1

ldr r3, [r0], #4

cmp r3, r2

blt less

b loop

less:

mov r2, r3

b loop

close: swi 0x011

.end

1. **@Find factorial using subroutine**

.text

mov r0, #5

mov r1, r0

mov r2, #1

bl fact

b close

fact:

stmfd r13!, {r1,r2, lr}

cmp r0, #1

beq return

mov r1, r0

sub r0, r0, #1

bl fact

mul r2, r1, r0

mov r0, r2

ldmfd r13!, {r1, r2, lr}

mov pc, lr

return:

mov r0, #1

ldmfd r13!, {r1, r2, lr}

mov pc, lr

close: SWI 0x011

.end

1. **@c[k] = a[i] \* b[j]**

.data

a: .word 10, 20, 30, 40, 50

b: .word 10, 20, 30, 40, 50

c: .word 0

.text

ldr r0, =a

ldr r1, =b

ldr r2, =c

mov r6, #5

loop:

cmp r6, #0

beq close

sub r6, r6, #1

ldr r3, [r0], #4

ldr r4, [r1], #4

mul r5, r3, r4

str r5, [r2], #4

b loop

close: SWI 0x011

1. **@Write a function for MUL(add(a, b), c) => [(a+b)\*c]**

.text

mov r0, #5 ; =a

mov r1, #10; =b

mov r2, #15; =c

stmfd r13!, {r0-r2}

bl mul\_add

ldmfd r13!, {r0}

SWI 0x011

mul\_add:

ldmfd r13!, {r3, r4, r5}

add r3, r3, r4

mul r4, r3, r5

stmfd r13!, {r4}

mov pc, lr

.end

1. **@Find 1s compliment of a number**

.text

mov r0, #0b0001010101

mvn r1, r0

SWI 0x011

.end

1. **@ Scan a 32 bit number to find its negative**

.text

ldr r0,=0x1234abc

mvn r1, r0

add r1, r1, #1

swi 0x11

.end

1. **@ Add a series of 32 bit number using table address**

.text

ldr r0, =a

mov r1, #6

mov r2, #0

loop:

ldr r3, [r0], #4

add r2, r2, r3

sub r1, r1, #1

cmp r1, #0

bne loop

swi 0x11

.data

a: .word 10, 20, 30, 40, 50, 60

.end

1. **@ Find the no. of zeros, positive, negative numbers in a given array**

.text

ldr r0, =a

mov r1, #9

mov r3, #0

mov r4, #0

mov r5, #0

loop:

ldr r2, [r0], #4

cmp r2, #0

beq zero

bgt great

blt less

check:

sub r1, r1, #1

cmp r1, #0

swieq 0x11

b loop

zero:

add r4, r4, #1

b check

great:

add r5, r5, #1

b check

less:

add r3, r3, #1

b check

.data

a:.word 20,30,0,40,50,0,-2,-4,-9

.end

1. **@ Count the numbers of 0's and 1's in a given 32 bit number**

.text

ldr r0, =0b111011110001

mov r1, #32

loop:

movs r0, r0, lsr #1

addcs r2, r2, #1

addcc r3, r3, #1

sub r1, r1, #1

cmp r1, #0

bne loop

swi 0x11

.end

1. **@To read from a 2D array (b = a[i][j])**

.data

a: .word 1, 2, 3, 4, 5, 6, 7, 8, 9

c: .word 0

.text

ldr r0, =a

ldr r1, =c

mov r2, #3 ; rows

mov r3, #3 ; columns

mov r4, #0 ; i

mov r5, #0 ; j

for\_i:

for\_j:

stmfd r13!, {r4, r5}

bl get\_addr

ldmfd r13!, {r4, r5, r6}

add r7, r0, r6 ; address a[i][j]

add r8, r1, r6 ; address c[i][j]

ldr r6, [r7]

str r6, [r8]

add r5, r5, #1

cmp r5, r3

bne for\_j

mov r5, #0

add r4, r4, #1

cmp r4, r2

beq exit

b for\_i

get\_addr:

ldmfd r13!, {r4, r5}

mla r7, r3, r4, r5

mov r8, #4

mul r6, r7, r8

stmfd r13!, {r4, r5, r6}

bx lr

exit: SWI 0x011

.end

1. **@Find sum or all the numbers in 2D array (sum += a[i][j])**

.data

a: .word 1, 2, 3, 4, 5, 6, 7, 8, 9

.text

ldr r0, =a

mov r1, #0 ; =Sum

mov r2, #3 ; =rows

mov r3, #3 ; =columns

mov r4, #0 ; =i

mov r5, #0 ; =j

for\_i:

for\_j:

stmfd r13!, {r4, r5}

bl get\_addr

ldmfd r13!, {r4, r5, r6}

add r6, r0, r6 ; address a[i][j]

ldr r6, [r6] ; value a[i][j]

add r1, r1, r6 ; sum += a[i][j]

add r5, r5, #1

cmp r5, r3

bne for\_j

mov r5, #0

add r4, r4, #1

cmp r4, r2

beq exit

b for\_i

get\_addr:

ldmfd r13!, {r4, r5}

mla r7, r3, r4, r5

mov r8, #4

mul r6, r7, r8

stmfd r13!, {r4, r5, r6}

bx lr

exit: SWI 0x011

.end

1. **@Write program to perform c[i][j] = a[i][j] + b[i][j]**

.data

a: .word 1, 2, 3, 4, 5, 6, 7, 8, 9

b: .word 1, 2, 3, 4, 5, 6, 7, 8, 9

c: .word 0

.text

ldr r0, =a

ldr r1, =a

ldr r2, =c

mov r3, #3 ; rows

mov r4, #3 ; columns

mov r5, #0 ; i

mov r6, #0 ; j

for\_i:

for\_j:

stmfd r13!, {r5, r6}

bl get\_addr

ldmfd r13!, {r5, r6, r7}

add r8, r0, r7 ; address a[i][j]

add r9, r1, r7 ; address b[i][j]

ldr r8, [r8]

ldr r9, [r9]

add r8, r8, r9 ; r8 = a[i][j] + b[i][j]

add r9, r2, r7 ; address c[i][j]

str r8, [r9]

add r6, r6, #1 ; j++

cmp r6, r4

bne for\_j

mov r6, #0

add r5, r5, #1

cmp r5, r3

beq exit

b for\_i

get\_addr:

ldmfd r13!, {r5, r6}

mla r8, r4, r5, r6

mov r9, #4

mul r7, r8, r9

stmfd r13!, {r5, r6, r7}

bx lr

exit: SWI 0x011

.end

Week 5

1. **@Find length of a string**

.data

a: .asciz "Hello"

.text

ldr r0, =a

mov r1, #0 ;length of string

loop:

ldrb r2, [r0], #1

cmp r2, #0

beq close

add r1, r1, #1

b loop

close: swi 0x011

.end

1. **@Copy String from one address to another**

.data

a: .asciz "Hello"

b: .asciz ""

.text

ldr r0, =a

ldr r1, =b

loop:

ldrb r2, [r0], #1

strb r2, [r1], #1

cmp r2, #0

beq close

b loop

close: swi 0x011

.end

1. **@Search for a character in a string**

.data

a: .asciz "Hello World"

b: .asciz "z"

.text

ldr r0, =a

ldr r1, =b

ldrb r1, [r1]

mov r4, #-1 ; -1 if character not found

loop:

ldrb r3, [r0]

cmp r1, r3

beq found

add r0, r0, #1

cmp r3, #0

bne loop

beq close

found:

mov r4, r0 ; store the address of where it is found

close: SWI 0x011

1. **@Binary search**

.data

a: .word 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

b: .word 1

.text

ldr r0, =a

ldr r1, =b

ldr r1, [r1]

mov r2, #0

mov r4, #10

search:

add r3, r2, r4

mov r3, r3, lsr #1

mov r5, r3, lsl #2

add r5, r5, r0

ldr r5, [r5]

cmp r1, r5

beq found

bgt greater

blt lesser

greater:

add r2, r3, #1

b condition

lesser:

sub r4, r3, #1

b condition

condition:

cmp r2, r4

blt search

b not\_found

found:

SWI 0x011

not\_found:

mov r3, #-1

SWI 0x011