**;ARM ALP to find the factorial of a number**

area reset,data,readonly

export \_\_Vectors

\_\_Vectors

dcd 0

dcd Reset\_Handler

area mycode,code,readonly

n equ 10

entry

export Reset\_Handler

Reset\_Handler

mov r1,#n; 10! = 3628800 (0x375f00)

mov r2,#1

rept mul r2,r1,r2

subs r1,r1,#1

cmp r1,#0

bne rept

stop b stop

end

**;ARM ALP to find the square-root of a number**

area reset,data,readonly

export \_\_Vectors

\_\_Vectors

dcd 0

dcd Reset\_Handler

area mycode,code,readonly

n equ 36

entry

export Reset\_Handler

Reset\_Handler

mov r0,#n ; number its sqrt to be found

mov r1,#1 ; first odd number

mov r2,#0

cont subs r0,r0,r1

blt stop

add r2,r2,#1

add r1,r1,#2 ; next odd number

b cont

stop b stop

end

;**ARM ALP to find the square of a number (1 to 10) using lookup table**

area reset,data,readonly

export \_\_Vectors

\_\_Vectors

dcd 0

dcd Reset\_Handler

area mycode,code,readonly

entry

export Reset\_Handler

Reset\_Handler

ldr r0,=ltable

ldr r1,=5 ;its square to be determined

mov r1,r1,lsl#2 ;generate the address corresponds to square of a no

add r0,r0,r1 ;address of lookup table

ldr r2,[r0] ; read the square into r2

stop b stop

ltable dcd 0x00000000

dcd 0x00000001

dcd 0x00000004

dcd 0x00000009

dcd 0x00000010

dcd 0x00000019

dcd 0x00000024

dcd 0x00000031

dcd 0x00000040

dcd 0x00000051

dcd 0x00000064

end

**;ARM ALP to find the sum of 3x+4y+9z, where x=2,y=3 and z=4**

area reset,data,readonly

export \_\_Vectors

\_\_Vectors

dcd 0

dcd Reset\_Handler

area mycode,code,readonly

x rn 1 ;register r1 is named as x

y rn 2 ;register r2 is named as y

z rn 3 ;register r3 is named as z

entry

export Reset\_Handler

Reset\_Handler

mov x,#2

mov y,#3

mov z,#4

add r1,r1,r1,lsl#1 ;r1=3x

mov r2,r2,lsl#2 ;r2=4y

add r3,r3,r3,lsl#3;r3=9z

add r1,r1,r2 ;r1=r1+r2 ie. 3x+4y

add r1,r1,r3 ;r1=r1+r3 ie. 3x+4y+9z

stop b stop

end

**;ARM ALP to calculate 3x2+5y2, where x=8 and y=5**

area reset,data,readonly

export \_\_Vectors

\_\_Vectors

dcd 0

dcd Reset\_Handler

area mycode,code,readonly

entry

export Reset\_Handler

Reset\_Handler

mov r2,#8

bl square ;call the square subroutine

add r1,r3,r3,lsl#1 ;3x2

mov r2,#5

bl square

add r0,r3,r3,lsl#2

add r4,r1,r0

stop b stop ;317=13d

square mul r3,r2,r2

bx lr; return lr back to pc

end

**;ARM ALP to generate first 20 natural numbers**

area reset,data,readonly

export \_\_Vectors

\_\_Vectors

dcd 0x10001000

dcd Reset\_Handler

area mycode,code,readonly

n equ 20

entry

export Reset\_Handler

Reset\_Handler

mov r0,#n ; n natural numbers

ldr r1,=natural

mov r2,#0

cont add r2,r2,#1

push {r2}

bl convert

strb r8,[r1],#1

pop {r2}

subs r0,r0,#1

bne cont

stop b stop

**;8-bit (0x00 t0 0x63) hexadecimal to decimal**

convert mov r5,#10

udiv r4,r2,r5

mul r6,r4,r5

sub r7,r2,r6 ;remainder (r2=r0-(r4\*r5)

add r8,r7,r4,lsl#4

bx lr

area mydata,data,readwrite

natural dcb 0

end

**;ARM ALP to generate first 10 odd numbers/even numbers**

area reset,data,readonly

export \_\_Vectors

\_\_Vectors

dcd 0x10001000;initialization of stack pointer

dcd Reset\_Handler ;initilization of PC

area mycode,code,readonly

entry

export Reset\_Handler

Reset\_Handler

mov r0,#10

ldr r1,=data1

mov r2,#1 ;mov r2,#0 for even numbers

cont strb r2,[r1],#1

add r2,r2,#2

subs r0,r0,#1

bne cont

stop b stop

area mydata,data,readwrite

data1 space 0

end

**;ARM ALP to generate 1st 10 Fibonacci series of numbers**

area reset,data,readonly

export \_\_Vectors

\_\_Vectors

dcd 0

dcd Reset\_Handler

area mycode,code,readonly

n equ 10

entry

export Reset\_Handler

Reset\_Handler

; 0 1 1 2 3 5 8 13 21 34 (0x00 0x01 0x01 0x02 0x03 0x05 0x08 0x0d 0x15 0x22)

mov r0,#n-1 ; numbers to be generated as counter

ldr r1,=fibo

strb r2,[r1],#1 ; 0x00

mov r3,#1

cont strb r3,[r1],#1 ; 0x01

mov r4,r3 ; exchange operation b/n r2 and r3

mov r3,r2

mov r2,r4

add r3,r2,r3 ; next fibonacii number

subs r0,r0,#1 ; decrement counter

cmp r0,#0

bne cont

stop b stop

area mydata,data,readwrite

fibo dcb 0

end

**;ARM ALP to find GCD of Two Numbers without based on conditional execution**

area reset,data,readonly

export \_\_Vectors

\_\_Vectors

dcd 0

dcd Reset\_Handler

area mycode,code,readonly

entry

export Reset\_Handler

Reset\_Handler

mov r1,#2

mov r2,#12

cont cmp r1,r2

beq over

blt lessthan

sub r1,r1,r2 ; if r1>r2, r1 = r1-r2

b cont

lessthan sub r2,r2,r1 ; if r1<r2, r2 = r2-r1

b cont

over ldr r3,=gcd

str r1,[r3]

stop b stop

area mydata,data,readwrite

gcd dcb 0

end

**;ARM ALP to find GCD of Two Numbers based on conditional execution**

area reset,data,readonly

export \_\_Vectors

\_\_Vectors

dcd 0

dcd Reset\_Handler

area mycode,code,readonly

entry

export Reset\_Handler

Reset\_Handler

mov r1,#2

mov r2,#12

cont cmp r1,r2

subgt r1,r1,r2 ; if r1>r2 , r1-r2

sublt r2,r2,r1 ; if r1<r2, r2-r1

bne cont

ldr r3,=gcd

str r1,[r3]

stop b stop

area mydata,data,readwrite

gcd dcb 0

end

**;ARM ALP to find LCM of Two Numbers without based on conditional execution**

area reset,data,readonly

export \_\_Vectors

\_\_Vectors

dcd 0

dcd Reset\_Handler

area mycode,code,readonly

entry

export Reset\_Handler

Reset\_Handler

mov r1,#2

mov r2,#12

mov r3,r1 ; save initial numbers

mov r4,r2

cont cmp r3,r4

beq over

blt lessthan

add r4,r4,r2 ; if r3>r4, r4 = r4+r2

b cont

lessthan add r3,r3,r1 ; if r3<r4, r3 = r3+r1

b cont

over ldr r5,=lcm

str r3,[r5]

stop b stop

area mydata,data,readwrite

lcm dcb 0

end

**;ARM ALP to find LCM of Two Numbers based on conditional execution**

area reset,data,readonly

export \_\_Vectors

\_\_Vectors

dcd 0

dcd Reset\_Handler

area mycode,code,readonly

entry

export Reset\_Handler

Reset\_Handler

mov r1,#2

mov r2,#12

mov r3,r1 ; save initial numbers

mov r4,r2

cont cmp r3,r4

beq over

addlt r3,r3,r1 ; if r3<r4, r3 = r3+r1

addgt r4,r4,r2 ; if r3>r4, r4 = r4+r2

bne cont

over ldr r5,=lcm

str r3,[r5]

stop b stop

area mydata,data,readwrite

lcm dcb 0

end