**;ARM ALP to find the sum of two 64-bit numbers in registers**

area reset,data,readonly

export \_\_Vectors

\_\_Vectors

dcd 0

dcd Reset\_Handler

area mycode,code,readonly

entry

export Reset\_Handler

Reset\_Handler

; 1st no. = 0x77777777 99999999

; 2nd no. = 0x66666666 80000000

ldr r0,=0x77777777 ;higher word of 1st no.

ldr r1,=0x99999999 ;lower word of 1st no.

ldr r2,=0x66666666 ;higher word of 2nd no.

ldr r3,=0x80000000 ;lower word of 2nd no.

adds r4,r1,r3;lower word result

adc r5,r0,r2 ;higher word result

ldr r6,=sum

rev r7,r5

str r7,[r6],#4 ; higher word result

rev r8,r4

str r8,[r6] ; lower word result

stop b stop

area mydata,data,readwrite

sum space 0

end

**;ARM ALP to find the sum of n-bit numbers in memory**

area reset,data,readonly

export \_\_Vectors

\_\_Vectors

dcd 0

dcd Reset\_Handler

area mycode,code,readonly

n equ 4

entry

export Reset\_Handler

Reset\_Handler ;proc

; 1st no. = 0x77777777 99999999 11111111 12222222

; 2nd no. = 0x66666666 80000000 44444444 f0000000

mov r7,#n ; no. of words

ldr r0,=num1+(4\*(n-1))

ldr r1,=num2+(4\*(n-1))

ldr r5,=sum+(4\*(n-1))

repeat ldr r2,[r0],#-4

ldr r3,[r1],#-4

msr psr,r8 ; retrive the status of cpsr back

adcs r4,r2,r3

mrs r8,psr ; save the status of cpsr

rev r6,r4

str r6,[r5],#-4

subs r7,r7,#1

cmp r7,#0

bne repeat

stop b stop

num1 dcd 0x77777777,0x99999999,0x11111111,0x12222222

num2 dcd 0x66666666,0x80000000,0x44444444,0xf0000000

area mydata,data,readwrite

sum space 0

end

**;ARM ALP to find the sum of array of n elements**

area reset,data,readonly

export \_\_Vectors

\_\_Vectors

dcd 0

dcd Reset\_Handler

area mycode,code,readonly

n equ 5

entry

export Reset\_Handler

Reset\_Handler

strt

mov r0,#n ;no. of additions

mov r1,#0 ; to hold carry if generated

ldr r2,=elements

cont ldr r3,[r2],#4

adds r4,r4,r3

bcc nxt

add r1,r1,#1

nxt subs r0,r0,#1

cmp r0,#0

bne cont

ldr r5,=sum

rev r6,r1

str r6,[r5],#4

rev r7,r4

str r7,[r5]

nop

nop

elements dcd 0x11111111,0x22222222,0x33333333,0x44444444,0x80000000

area mydata,data,readwrite

sum dcd 0

end

**;ARM ALP to find the sum of two array of n elements**

area reset,data,readonly

export \_\_Vectors

\_\_Vectors

dcd 0

dcd Reset\_Handler

area mycode,code,readonly

n equ 4

entry

export Reset\_Handler

Reset\_Handler

mov r0,#n ;no. of words

ldr r1,=array1+12 ;1st array

ldr r2,=array2+12 ;2nd array

ldr r3,=array3+16;result

cont ldr r4,[r1],#-4

ldr r5,[r2],#-4

adcs r6,r4,r5

rev r7,r6

str r7,[r3],#-4

subs r0,r0,#1

cmp r0,#0

bne cont

bcc stop ;if no carry stop

mov r8,#1

rev r9,r8

str r9,[r3]

stop b stop

array1 dcd 0x11111111,0x22222222,0x33333333,0x44444444

array2 dcd 0xf5555555,0xf6666666,0xf7777777,0xf8888888

area mydata,data,readwrite

array3 dcd 0

end

**;ARM ALP to find the sum of 1st 10 natural numbers**

area reset,data,readonly

export \_\_Vectors

\_\_Vectors

dcd 0

dcd Reset\_Handler

area hello,code,readonly

n equ 10

entry

export Reset\_Handler

Reset\_Handler

mov r1,#n ; counter

mov r2,#0 ; to hold sum

loop

add r2,r2,r1

subs r1,r1,#1

bne loop

stop b stop

end

**;ARM ALP to find the difference of two numbers**

area reset,data,readonly

export \_\_Vectors

\_\_Vectors

dcd 0

dcd Reset\_Handler

area hello,code,readonly

entry

export Reset\_Handler

Reset\_Handler

**;using sub instruction**

ldr r0,= 0x55555555

ldr r1,= 0x22222222

subs r2,r0,r1

**;using mvn instruction based on 2's complement**

ldr r3,= 0x66666666

ldr r4,= 0x22222222

mvn r5,r4 ;1's complement

adds r6,r3,r5

adds r6,r6,#1 ; 2's complement

stop b stop

end

**;ARM ALP to demonstrate the multiplication operation**

area reset,data,readonly

export \_\_Vectors

\_\_Vectors

dcd 0

dcd Reset\_Handler

area mycode,code,readonly

entry

export Reset\_Handler

Reset\_Handler

;16-bit multiplication

mov r0,#0x1234

mov r1,#0x5678

mul r2,r1,r0 ;[r2]=[r1\*r0]

;16-bit multiply and accumulate

mov r12,#1

mla r11,r1,r0,r12 ; [r11]=([r1\*r0]+r12)

;32-bit multiplication

ldr r3,=0x12345678

ldr r4,=0x90abcdef

umull r6,r5,r4,r3 ;[r6,r5]=r4\*r3

;32-bit signed multiplication

ldr r7,=-0x1234

ldr r8,=-0x5678

smull r10,r9,r8,r7 ;[r10,r9]=r8\*r7

;32-bit unsigned multiply and accumulate

ldr r3,=0x12345678

ldr r4,=0x90abcdef

umlal r6,r5,r4,r3 ;[r6,r5]=[r6,r5]+r4\*r3

;32-bit signed multiply and accumulate

ldr r7,=-0x1234

ldr r8,=-0x5678

smull r10,r9,r8,r7 ;[r10,r9]=[r10,r9]+r8\*r7

stop b stop

end

**;ARM ALP to find the product of 32-bit x 16-bit using mul instruction**

area reset,data,readonly

export \_\_Vectors

\_\_Vectors

dcd 0

dcd Reset\_Handler

area mycode,code,readonly

entry

export Reset\_Handler

Reset\_Handler

**; 0x12345678 X 0x1234 = 0x014b60b60060**

mov r0,#0x1234 ; Higher multiplicand

mov r1,#0x5678 ; lower multiplicand

mov r2,#0x1234 ; multiplier

mul r3,r1,r2 ; 1st phase multiplication

; (r3) <--0x06260060

mov r4,#0xffff

movt r5,#0xffff

and r6,r3,r4 ; (r6)<--0x00000060 1st 16-bit product

and r7,r3,r5 ; (r7)<--0x06260000

mul r8,r0,r2 ; 2nd phase multiplication

; (r8) <--0x014b5a90

and r9,r8,r4 ; (r9)<--0x00005a90

and r10,r8,r5 ; (r10)<--0x014b0000

add r11,r9,r7,lsr#16 ;(r11) <--0x000060b6 ;2nd 16-bit product

add r11,r6,r11,lsl#16 ; (r11) <--0x60b60060 ;2nd and 1st 16-bit product

adc r12,r10,lsr#16 ; (r12) <--0x0000014b ;3rd 16-bit product

; Final Product 🡪 (r12 r11) = 0x0000014b 60b60060

stop b stop

end

|  |  |  |
| --- | --- | --- |
|  | 0626 | 0060 |
| 014b | 5a90 |  |
| 014b | 60b6 | 0060 |

**logic : 0x12345678 x 0x1234**

0x5678 X 0x1234

0x1234 X 0x1234

Final Product

**;ARM ALP to realize division by successive subtraction**

area reset,data,readonly

export \_\_Vectors

\_\_Vectors

dcd 0

dcd Reset\_Handler

area hello,code,readonly

entry

export Reset\_Handler

Reset\_Handler

**; using successive subtraction**

mov r0,#0 ; to hold quotient

mov r1,#0 ; to hold remainder

mov r2,#500 ; dividend

mov r3,#16 ; divisor

repeat subs r2,r2,r3 ; (r2) <--(r2) - (r3)

addge r0,r0,#1 ; (r0) <--(r0) + 1

bge repeat

add r1,r2,r3

**; using udiv instruction**

mov r5,#500

mov r6,#16

udiv r5,r6

stop b stop

end

**;ARM ALP to demonstrate all arithmetic operation using subroutine**

area reset,data,readonly

export \_\_Vectors

\_\_Vectors

dcd 0

dcd Reset\_Handler

area hello,code,readonly

entry

export Reset\_Handler

Reset\_Handler

mov r0,#05

mov r1,#02

ldr r7,=result

bl addition

strb r2,[r7],#1

bl subtraction

strb r3,[r7],#1

bl subtraction1

strb r4,[r7],#1

bl multiplication

strb r5,[r7],#1

bl division

strb r6,[r7],#1

strb r0,[r7]

stop b stop

addition add r2,r1,r0 ;addition

bx lr

subtraction sub r3,r0,r1 ;substraction

bx lr

subtraction1 subs r4,r1,r0 ;2's complement

bx lr

multiplication mul r5,r1,r0 ;multiplication

bx lr

division cmp r0,r1 ;division

bcc stop1

sub r0,r1

add r6,#1 ;r6=2,r0=1

b division

stop1 nop

bx lr

area mydata,data,readwrite

result space 0

end