;ARM ALP to read memory content directly but direct write is invalid

```
 area reset,data,readonly
              export __Vectors
__Vectors

     dcd 0
              dcd Reset_Handler


  area mycode,code,readonly

  entry
  export Reset_Handler
Reset_Handler
                         ldr r0,nums
                         ldr r1,nums+4
                         ldr r2,nums+8
                         ldr r3,nums+12
                         ldr r4,nums+16

            ;            str r0,data1    ; invalid
            ;            str r1,data1+4   ; invalid
            ;            str r2,data1+8   ; invalid
            ;            str r3,data1+12   ; invalid
            ;            str r4,data1+16   ; invalid

stop b stop

nums       dcd 0x12345678
              dcd 0x01234567
              dcd 0x23456789
             dcd 0x90abcdef
            dcd 0x34567890
            area mydata,data,readwrite
data1 dcd 0

     end
```

;ARM ALP to demonstrate nested procedure

```
 area reset,data,readonly
		export __Vectors
__Vectors
     dcd 0x10001000 ; initialization of stack pointer
		dcd Reset_Handler ;initilization of PC


  area hello,code,readonly
  entry
  export Reset_Handler
Reset_Handler
start

  ldr r0,=nest
  ldr r1,=nest1
  mov r2,#1
  mov r3,#2
  mov r4,#3

  bl proc1
  add r5,r2,r3
  add r6,r5,r4
  str r6,[r0,#4]

stop b stop

proc1     stmea sp!,{lr,r2-r6}
		mov r2,#4
		mov r3,#5
		mov r4,#6
		bl proc2
		str r6,[r0]
		ldmea sp!,{r2-r6,lr}
		bx lr

proc2 add r5,r2,r3
     add r6,r5,r4
		bx lr

 area mydata,data,readwrite
nest dcd 0
   area mydata,data,readwrite
nest1 dcd 0
 end
```

```
;ARM ALP to find largest/smallest of two numbers

 area reset,data,readonly
              export __Vectors
__Vectors

              dcd 0
              dcd Reset_Handler

  area mycode,code,readonly
  entry
  export Reset_Handler
Reset_Handler
             ldr r0,=data1
             ldr r1,=largest

             ldr r2,[r0],#4
             ldr r3,[r0]
            cmp r2,r3
             bhi max    ; blt min for smallest number
             mov r2,r3
max       str r2,[r1]

stop      b stop

data1 dcd 0xabcedf01,0x12345678

    area mydata,data,readwrite
largest dcd 0

     end
```

;ARM ALP to find the largest/smallest number in an array of 5 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


             mov r0,#n-1 ;no. of comparisons where n is the no. of elements
             ldr r1,=nums

           ldr r2,[r1],#4 ;1st no.
           loop ldr r3,[r1],#4 ;2nd no.
           cmp r2,r3
            bhi max ;if the 1st no. is greater than (bls min for smaller no.)
            mov r2,r3 ;keep the greater no. in r2
max         subs r0,r0,#1 ; decrement the counter
            cmp r0,#0
            bne loop

            ldr r4,=result
            rev r5,r2 ;reverse the data
            str r5,[r4]

stop b stop


nums       dcd 0x12345678
            dcd 0x01234567
            dcd 0x23456789
           dcd 0x90abcdef
           dcd 0x34567890

 area mydata,data,readwrite
result dcd 0

   end
```

;ARM ALP to arrange the elements in an array in ascending/descending order

```
 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
   ; trnsfering to data memory
   mov r0,#n
   ldr r1,=nums
   ldr r2,=sorted
loop ldr r3,[r1],#4
     str r3,[r2],#4
              subs r0,r0,#1
              cmp r0,#0
              bne loop
            ; sorting
    mov r4,#n-1
pass ldr r5,=sorted
    mov r6,r4
comp ldr r7,[r5],#4
     ldr r8,[r5]
              cmp r7,r8
              bcc nch  ; ascending order (bcs for descending order)
              str r7,[r5],#-4
              str r8,[r5],#4
nch           subs r6,r6,#1
              cmp r6,#0
              bne comp
              subs r4,r4,#1
              cmp r4,#0
              bne pass
stop b stop

nums      dcd 0x12345678
              dcd 0x01234567
              dcd 0x23456789
              dcd 0x90abcdef
              dcd 0x34567890

 area mydata,data,readwrite
sorted dcd 0

   end
```

;ARM ALP to search for a given word and its position in an array of 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

   mov r0,#n ;no. of elements
   ldr r1,=nums
   mov r2,#0 ;position of the element

loop       ldr r3,[r1],#4
           ldr r4,value
           cmp r3,r4
           bne notfound
           mov r2,r1
           sub r2,r2,#4
           mov r5,#'A'
           b stop
notfound subs r0,r0,#1
           cmp r0,#0
           bne loop
           mov r5,#'a'

stop b stop

value     dcd 0x12345678

nums      dcd 0x12345678
            dcd 0x01234567
            dcd 0x23456789
           dcd 0x90abcdef
           dcd 0x34567890

    end
```

;ARM ALP to find the length of the string

```
 area reset,data,readonly
             export __Vectors
__Vectors

     dcd 0
             dcd Reset_Handler


  area mycode,code,readonly

  entry
  export Reset_Handler
Reset_Handler



            mov r0,#0
            ldr r1,=str

loop    ldrb r2,[r1],#1
          cmp r2,#0
          beq stop
         add r0,r0,#1
         b loop

stop   b stop

str dcb "Dept. of E&C",0

   end
```

;ARM ALP to monitor cpsr conditional flag status

```
 area reset,data,readonly
		export __Vectors
__Vectors

		dcd 0x10001000
		dcd Reset_Handler

  area mycode,code,readonly
  entry
  export Reset_Handler
Reset_Handler

			ldr r0,=0x12345678
			movs r0,#0  ; Z=1

			 ldr r3,=-0x12345678
			movs r4,r3 ; N=1

			ldr r5,=0xffffffff
			ldr r6,=0x10000001
			adds r7,r5,r6 ; C=1

			ldr r8,=-0x8fffffff
			ldr r9,=-0x8fffffff
			adds r10,r8,r9 ; V=1

stop b stop

     end
```