

```

; Vector Table Mapped to Address 0 at Reset
; Linker requires __Vectors to be exported

```

```

        AREA    RESET, DATA, READONLY
        EXPORT  __Vectors

```

```

__Vectors
        DCD    0x20000100    ; stack pointer value when stack
is empty
        DCD    Reset_Handler ; reset vector
intvarx  DCD    0x4          ; value of base
posintn  DCD    0x5          ; value of exponent

```

```

        ALIGN
; data area that can be modified (readwrite)
        AREA    Data, DATA, READWRITE
result   DCD    0x0          ; result value
        ALIGN
; The program
; Linker requires Reset_Handler

```

```

        AREA    MYCODE, CODE, READONLY

```

```

        ENTRY
        ALIGN
        EXPORT  Reset_Handler

```

```

power PROC

```

```

;; r0 contains value of x to be used for x^n ; integer value
;; r1 contains value of n to be usef for x ^ n ; non negative integer
value
;; r0 returns the result value of x^n

```

```

        STMFD   SP!, {R0-R3, LR}    ; push all registers
used in subroutine to prevent corruption
        ADD     r2,sp,#0x14          ; Get SP
value before reg contents were stacked above
        LDM     r2,{r0,r1,r2}       ; POP function
parameters -- r0 = X, r1= n, r2 = return value
; if (n == 0) return 1;
        CMP     r1,#0x0              ;n==0
        MOVEQ   r2,#0x1              ; return 1
        BEQ     exitproc
; here if n != 0;
; check if n is odd
        TST     r1,#0x1              ; check
LSB
        BEQ     evenpower            ; LSB is clear
so N is even
; here if N is odd
        SUB     r1,r1,#0x1           ; N = N -1

```

```

        STMFD      SP!,{r0,r1,r2}          ; prepare stack for
call to function -- r2 = X, r1 = N, r2 - return value
        BL        power                    ; recursive call
to function
        LDMFD      SP!,{r0,r1,r2}         ; Pop returned values
after function call
        MUL        r2,r2,r0                ; return x *
Power(x,n-1)
        ;; return from here
        B         exitproc
evenpower
        ;; here if N is even
        LSR        r1,r1,#0x1              ; N = N/2 by
right shift
        STMFD      SP!,{r0,r1,r2}         ; prepare stack for
call to function -- r2 = X, r1 = N, r2 - return value
        BL        power                    ; call
Power(x,N/2)
        LDMFD      SP!,{r0,r1,r2}         ; Pop returned values
after function call
        ;; result is obtained in r2
        MUL        r2,r2,r2                ; return value
in r2 as y * y
        ;; return from here
exitproc
        ADD        r3,sp,#0x14             ; Get SP
value before reg contents were stacked at function entry
        STM        r3!,{r0,r1,r2}         ; Push function
parameters back to stack -- r0 = X, r1= n, r2 = return value
        LDMFD      SP!, {R0-R3, PC}       ; pop all registers
and branch back to main program
        ENDP

```

#### Reset\_Handler

```

;;;;;;;;;;User Code Starts from the next line;;;;;;;;;;
;; code to copy data from ROM to RAM on startup i.e. from readonly to
readwrite area in memory

; any other startup code here
        B         mainprog
        ALIGN 2
mainprog ;;add your code here
        LDR        r0,intvarx              ; load value of
variable X
        LDR        r1,posintn              ; load value of
variable N
        STMFD      SP!,{r0,r1,r2}         ; create stack frame
for function parameters.r0=X, r1=N, r2= return value
        BL        power                    ; Call power
function
        LDMFD      SP!,{r0,r1,r2}         ; retrieve function parameters , r0=X, r1=N,r2=return value

```

```

LDR      r1,=result      ; load address of result
STR      r2,[r1]         ; store result
B        .                ; end of program
END

```

Stack Frame structure used to pass arguments to the function:

Value	Register
Return Value	r2
N	r1
X	r0

Stack Frame structure used with in function to prevent corruption of registers:

Value	Register
Any value	r0
Any value	r1
Any value	r2
Any value	r3
Return address for the branch	LR