Procedures with Parameter

Invoke:
This directive works like call but it can pass parameters to the procedure. It pushes the parameters on stack and call the procedure.

ADDR Operator
The ADDR operator can be used to pass a pointer argument when calling a procedure using INVOKE. The following INVOKE statement, for example, passes the address of **myArray** to the **FillArray** procedure:

```
INVOKE FillArray, Param1, Param2, ADDR myArray
```

PROTO Directive
The PROTO directive creates a prototype for an existing procedure. A *prototype* declares a procedure's name and parameter list. It allows you to call a procedure before defining it and to verify that the number and types of arguments match the procedure definition. (The C and C++ languages use function prototypes to validate function calls at compile time.)

Uses Reg

Example – 1 : Add two Number

```
TITLE MASM Template                                    (main.asm)

INCLUDE Irvine32.inc
.data
var1 Dword 5
var2 Dword 10
buffer dword 0

.code
addTwo proto, val1:DWORD, Val2:Dword

main PROC

       invoke addTwo, var1,var2
       call writeint
       exit
main ENDP

addTwo proc uses ebx, val1:DWORD, Val2:Dword
              mov eax, val1
              add eax, val2
              mov ebx, eax
              ret
addTwo ENDP
END main
```

Example – 2 : Sum of Array

```
TITLE MASM Template                                          (main.asm)

INCLUDE Irvine32.inc
.data
arr dword 5,5,5,5,5
arrLen dword ?
.code
sumArray proto, arr: ptr dword, arSize:dword

main PROC
       mov eax, lengthof arr
       mov arrLen, eax
       invoke sumArray, Addr arr, arrlen
       call writeint
       exit
main ENDP

sumArray proc, ar: ptr dword, arLen:dword

              mov esi,ar ; address of the array
              mov ecx,arLen ; size of the array
              mov eax,0 ; set the sum to zero
              cmp ecx,0 ; length = zero?
              je L2
       L1: add eax,[esi] ; add each integer to sum
              add esi,4 ; point to next integer
              loop L1 ; repeat for array size
       L2: ret
sumArray ENDP
END main
```