```
include \masm32\MasmBasic\MasmBasic.inc

atImmediate         = 36        ; 00100100
atGlobal            = 42        ; 00101010
atRegister          = 48        ; 00110000
atLocal             = 98        ; 01100010

bLoopCtFor          = 0
MbFor MACRO fline                       ;; For_ n=1 to eax-1 is OK
LOCAL ctStart, ctVar, ctEnd, iseq, isto, ArgType, tmp$, vtE$, otE$, vtS$, otS$, tmpReg, tregpush
.Repeat                                 ;; dummy, triggers error if matching .Until in Next not present
 ifndef MbForCounters
    .data?
        MbForCounters   dd 20 dup(?)
    .code
 endif
 iseq INSTR <fline>, <=>
 isto INSTR <fline>, <TO>
 if isto eq 0
                    isto INSTR <fline>, <To>
 endif
 ctVar              SUBSTR <fline>, 1, iseq-1
 ctStart            SUBSTR <fline>, iseq+1, isto-iseq-1
 ctEnd              SUBSTR <fline>, isto+2
 ArgType =          (opattr ctVar) AND 127
 if ArgType ne 48   ;; not a register
    % ifndef ctVar
    .data?
        ctVar   dd ?
    .code
    endif
 endif
 ctSm INSTR 2, ctEnd, <->
 ctSp INSTR 2, ctEnd, <+>
 if ctSp
    vtE$ SUBSTR ctEnd, 1, ctSp-1    ;; x plus
    otE$ SUBSTR ctEnd, ctSp+1               ;; if +1 To, there is a blank
;   echo PLUS
 elseif ctSm
    vtE$ SUBSTR ctEnd, 1, ctSm-1    ;; x minus
    otE$ SUBSTR ctEnd, ctSm+1               ;; if -1 To, there is a blank
;   echo MINUS
 else
    vtE$ equ ctEnd
 endif
 atVt = (opattr vtE$) AND 127
 tregpush = 0
 if atVt eq atRegister
    tmpReg equ vtE$         ;; we take the ctEnd register as is
    ifdef otE$
        tregpush = 1        ;; offset, we need to push a tmp register
    endif
 else
    tmpReg equ eax
    tregpush = 1            ;; not a register, we need one
 endif
 if tregpush
    push tmpReg
    ifdifi tmpReg, <vtE$>
        mov tmpReg, vtE$
    endif
 endif
 ifdef otE$
    ifidn otE$, <1>
        if ctSp
            inc tmpReg
        else
            dec tmpReg
        endif
    else; ---------------------------------------------------------------------------
        if ctSp
            add tmpReg, otE$
        else
            sub tmpReg, otE$
        endif
    endif
 endif
 mov MbForCounters[bLoopCtFor], tmpReg
 if tregpush
    pop tmpReg
 endif
 ctSm INSTR 2, ctStart, <->
 ctSp INSTR 2, ctStart, <+>
 if ctSp
    vtS$ SUBSTR ctStart, 1, ctSp-1  ;; x plus
    otS$ SUBSTR ctStart, ctSp+1     ;; if +1 To, there is a blank
 elseif ctSm
    vtS$ SUBSTR ctStart, 1, ctSm-1      ;; x minus
    otS$ SUBSTR ctStart, ctSm+1     ;; if -1 To, there is a blank
 else
    vtS$ equ ctStart
 endif
 atVt = (opattr vtS$) AND 127
 tregpush = 0
 if atVt eq atRegister
    tmpReg equ vtS$         ;; we take the ctStart register as is
    ifdef otS$
        tregpush = 1        ;; offset, we need to push a tmp register
    endif
 else
    tmpReg equ eax
    tregpush = 1            ;; not a register, we need one
 endif
 if tregpush
    push tmpReg
    ifdifi tmpReg, <vtS$>
        if atVt eq atImmediate
            if vtS$ eq 0
                xor tmpReg, tmpReg
            elseif vtS$ eq -1
                or tmpReg, -1
            elseif (vtS$ le 127) and (vtS$ ge -128)
                push vtS$
                pop tmpReg
            else
                mov tmpReg, vtS$
            endif
        else
            mov tmpReg, vtS$
        endif
    endif
 endif
 ifdef otS$
    ifidn otS$, <1>
        if ctSp
            inc tmpReg
        else
            dec tmpReg
        endif
    else
        if ctSp
            add tmpReg, otS$
        else
            sub tmpReg, otS$
        endif
    endif
 endif
 sub MbForCounters[bLoopCtFor], tmpReg
 ifidn tmpReg, <0>
    mov ctVar, tmpReg
 else
    mov ctVar, tmpReg ; ---------------------------------------------------------------------------
 endif
 if tregpush
    pop tmpReg
 endif
 bLoopCtNext=bLoopCtFor                 ;; first loop: 0
 ctVar4N CATSTR <ctVarL>, %bLoopCtFor   ;; ctVarL0
 % ctVar4N equ <ctVar>
 ctVar4E CATSTR <ctVarE>, %bLoopCtFor   ;; ctVarE0
 % ctVar4E equ <ctEnd>
 MbForLoop CATSTR <MbForLabel>, %bLoopCtFor
 MbForLoop:
 bLoopCtFor=bLoopCtFor+4                ;; increase for next nesting level
ENDM

MbNext MACRO ctVar1
LOCAL tmp$
 ctVar4N CATSTR <ctVarL>, %bLoopCtNext
 ctVar4E CATSTR <ctVarE>, %bLoopCtNext
 ifnb <ctVar1>
    % ifdif <ctVar4N>, <ctVar1>
        echo
        echo Next:
        tmp$ CATSTR <Expected   var = !<>, <ctVar4N>, <!>>
        % echo tmp$
```

```
            tmp$ CATSTR <Unexpected var = !<>, <ctVar1>, <!>>
            % echo tmp$
            echo
            .err
        endif
    endif
    inc ctVar4N
    MbForLoop CATSTR <MbForLabel>, %bLoopCtNext
    dec MbForCounters[bLoopCtNext]
    bLoopCtNext=bLoopCtNext-4
    jge MbForLoop
    .Until 1
ENDM

;               For_ esi=0 To My$(?)-1
;               xor esi, esi
;               push eax
;               dec eax
;               mov MbForCounters[4], eax
;               pop eax


.data?
MbForCounters   dd 20 dup(?)

.code
start:          mov ecx, esp
                GetFiles *.asm          ; get all assembler sources in the current folder
                Dim My$(5)
                xor ebx, ebx
                .Repeat
                        Let My$(ebx)=Files$(ebx)
                        inc ebx
                .Until ebx>5
                mov eax, My$(?)
                push eax
                mov ebx, eax            ; #files found
                ; Print Str$("%i files found:\nNo.\tBytes\tName", ebx)     ; \n is newline, \t is tab
                pop eax

                UseNops=1

;               int 3
                mov ecx, ecx
                manS:           ; ------------------------
                m2m esi, 2
                push eax
                dec eax
                mov MbForCounters[0], eax
                sub MbForCounters[0], esi
```