console, argument, math

== apollo_asm ==

```
*---------------------------------------------------------------------------
*--- program:   print_fib2 program
*--- author:    pisklak
*--- description: This is a small program that calculates and prints all
fibbonacci
*---            values from 1 to Nelement, where Nelement must be given to
prg as CLI parameter.
*---            Valid parameter is a number in range 1-1000.
*---            So usage is print_fib2 100 for example. You may try give it
invalid parameters
*---            which should be rejected and error msg printed.




    include exec/types.i
    include exec/libraries.i
    include    exec/exec_lib.i
    include exec/exec.i
    include    dos/dos_lib.i
    include    dos/dos.i

    MACHINE    MC68020

Max_Number  EQU    1000
*---------------------- opening DOS library -----------------------------
openlibs
    move.l  $4,_ExecBase
    lea        DosName,A1
    move.l    #0,d0
    CALLEXEC OpenLibrary
    move.l    d0,_DOSBase
*---------------------- getting stdinput and output ----------------------
    CALLDOS    Output                    ; get _stdout
    move.l    d0,_stdout
    CALLDOS    Input                    ; get _stdinput
    move.l     d0,_stdinput

*---------------------- getting argument ---------------------------------
get_one_arg
    move.l     #arg_string,d1            ; move buffer ptr in d1
    move.l     #10,d2                    ; move buffer size in d2
    clr.l     d3                        ; no input specified = read from
Input()
    CALLDOS ReadItem                ; read one argument, now we have it in
our buffer
```

```
    cmp.l     #ITEM_NOTHING,d0        ; check do user give argument
    beq     prt_error               ; branch to eror printing message if
user do not do that
*----------------------- convert string routine ------------------------
convertstr
    move.l    #arg_string,d1           ; load pointer to arg_string into d1
    move.l    #Nelement,d2            ; load pointer to our Nelement
variable into d2
    CALLDOS StrToLong                ; try convert string to long value
    cmp.l    #-1,d0                   ; checkout do user give any valid
argument (number)
    beq    prt_error1                   ; if not print error msg
    move.l    Nelement,d0               ; copy converted value into d0
    cmp.l    #1,d0                    ; is value is >0
    blo    prt_error2                   ; if not prt error mgs
    cmp.l    #Max_Number,d0           ; now checkout do value<max range
    bhi    prt_error2                   ; if not prt error msg

*----------------------- Now we allocate memory for our fibbonacci buffer --
-------
*----------------------- size of allocated memory depends of given argument
-------
allocmem
    move.l     Nelement,d0            ; number of our array entries
    mulu.l     #4,d0                ; now we calculate actual lenght in
bytes
    move.l    #MEMF_PUBLIC|MEMF_CLEAR,d1    ; we require public cleared
memory
    CALLEXEC AllocMem            ; there we go - we 'eat' some mem from
system
    cmp.l     #0,d0                ; check do we sucessfull eat it
    beq        prt_error3            ; if not prt error msg
    move.l     d0,fib_array        ; store our allocated mem pointer

*------------------- OK so now we have space to fill. Let's do that ! ------
-------

    move.l     fib_array,a3
    move.l    #1,(a3)+             ; write 1st valuee
    move.l     #1,(a3)+            ; and 2nd one
    move.l     #3,d0                ; for counter
    move.l    #1,d1                ; fib1
    move.l    #1,d2                ; fib2
    clr.l    d3                    ; tmp
for     cmp.l    Nelement,d0        ; i<=Nelement
        bhi    setup_print            ; if we calc last value then go to
setup print
        move.l    d1,d3
        add.l    d2,d3            ; temp= fib1+fib2
        move.l    d2,d1            ; fib1=fib2
        move.l    d3,d2            ; fib2=temp
```

```
        add.l    #1,d0
        move.l    d2,(a3)+        ; write value to our array
        bra     for

*------------------- now when we have values we should print them right ? --
---------
setup_print
    move.l     fib_array,d3        ; from where we will print our values
    move.l     Nelement,d2            ; mow many elements in d2
    mulu.l    #4,d2                ; how big is that in bytes
    add.l     d2,d3                ; now we calculate where our array ends
    move.l    d3,array_end        ; store that addr in variable
    move.l    fib_array,d3        ; reset addr
.loop
        cmp.l     array_end,d3    ; checkout do we reach end
        bhs        prt_LF
        move.l    #prt_val_str,d1    ; our formatted string
        move.l    d3,d2             ; pointer to value
        CALLDOS VPrintf            ; print value
        add.l    #4,d3            ; move to next array value
        bra        .loop
prt_LF
    move.l     #LF_str,d1            ; this just prints some ---- after all
values are printed
    CALLDOS    PutStr
*------------------- we done our job but do not forget free our memory
buffers ---------
freemem
    move.l    fib_array,a1        ; pointer to our fib array
    move.l     Nelement,d0            ; again calc array size in bytes
    mulu.l     #4,d0
    CALLEXEC FreeMem            ; free our mem

*------------------- close opened libs ----------------------------------
--------------
closelibs
    move.l    _DOSBase,A1
    CALLEXEC CloseLibrary
*------------------- and finnaly end of the prg -----------------------------
--------------
prgend
        RTS

*------------------- there are prt error routines -------------------------
--------------
prt_error
    move.l    #prt_eror_str,d1
    CALLDOS PutStr
    bra     closelibs

prt_error1
```

```
    move.l    #prt_eror_str1,d1
    CALLDOS    PutStr
    bra     closelibs

prt_error2
    move.l     #prt_eror_str2,d1
    CALLDOS    PutStr
    bra        closelibs

prt_error3
    move.l     #prt_eror_str3,d1
    CALLDOS    PutStr
    bra     closelibs
*------------------ variable and constans definitions ----------------------
--------
fib1                dc.l    1
fib2                dc.l    1
temp                dc.l    0
fib_array           dc.l    0
Nelement            dc.l     0
_ExecBase           dc.l    0
_DOSBase            dc.l    0
DosName             dc.b    "dos.library",0
_stdout             dc.l    0
_stdinput           dc.l    0
arg_string          ds.b     10
prt_eror_str        dc.b    "You do not give any angument !",LF,0
prt_eror_str1       dc.b    "You do not give valid number as argument
!",LF,0
prt_eror_str2       dc.b    "Your argument is not in range 1 - 1000!",LF,0
prt_eror_str3       dc.b     "Can't allocate memory !",LF,0
prt_val_str         dc.b     "%lu ",0
LF_str              dc.b    LF,"----------------------------------------
----------",LF,0
array_end           dc.l    0
```