Programming in Assembler

Laboratory manual

Exercise 5

Procedures, macros

During the Exercise No.5 students are to analyze the program using the CodeView Debugger. Next step is to modify the macros in the program changing them into procedures. On the last step the program should be modified to conditional version.
Program is attached to the documentation in lab5.asm file.

Explanation of DOS functions used in the program:

1. **62h** – Get PSP address
   Gets the segment address of the PSP for the current process.
   Returns in BX: segment address of PSP for current process.

2. **29h** – Parse Filename.
   CP/M based function that fills the FCB block with file name.
   ES:DI holds the address of filled FCB block;
   DS:SI holds address of the file name string;
   AL=01 for ignoring separate characters.

3. **4Ah** – Resize memory block
   Adjusts the size of a previously allocated block of memory.
   BX = new size of memory block in paragraphs;
   ES = segment address of previously allocated memory block.
   Returns in carry flag; clear if successful; set otherwise
   Returns in AX:
   - 07 if memory control blocks damaged;
   - 08 if insufficient memory to allocate as requested;
   - 09 if incorrect memory segment specified;

   Returns in BX: maximum number of paragraphs available (if an increase was requested).

4. **4Bh** – Load and execute program (EXEC) - Loads a program file into memory and optional executed the program. This function can also be used to load a program overlay.
   AL =
   - 00 to load and execute program;
   - 03 to load overlay;

   DS:DX = address of ASCIIZ pathname for an executable program file;
   ES:BX = address of parameter block.
   Returns in carry flag: clear if successful; set otherwise;
   Returns in AX:
   - 01 if invalid function (AL not 00 or 03);
   - 02 if file not found;
   - 03 if path not found;
   - 05 if access denied;
   - 08 if insufficient memory;
   - 0Ah if bad environment;
   - 0Bh if bad format (for AL=00 only).

   Note: With MS-DOS 2.x all registers except CS and IP can be destroyed; with 3.x registers are preserved.

During the laboratory students are to:

1. Create the project to the lab5.asm file with options for debugging and generating listing file.
2. Assemble the project to the *.exe file and run the program in debugger using step-by-step mode.
3. Analyze the program with attention to macros' expantion.
4. Modify the program replacing the macros with procedures.
5. Modify the program to conditional version that:
   - Assemble to the version with macros when USEMACRO =1
   - Assemble to the version with procedures when USEMACRO =0
6. Analyze and compare the execution time of two program versions.
7. Analyze and compare the memory space of two program versions.
8. Comment the program.

The report should consist of:

- Title page.
- Explanation of program function.
- Modified program listing file.
- Expantions of two chosen macros.
- Comparison of memory usage and execution time of two versions.
- Conclusions.

Source code:

```
;*********************************************************************
;*                                                                   *
;*                LAB5.ASM - Assembler Laboratory ZMiTAC             *
;*                                                                   *
;*                         Calculate exe time                        *
;*                                                                   *
;*********************************************************************
; Program's work:
;  1. Check the presence of FPU - exit if not
;  2. Take PSP (Program Segment Prefix)
;  3. Decrease the memory taken by the program (initially takes whole
memory)
;  4. Taking executable name from parameters
;  5. Create EPB (Exec Parameter Block) for executable file
;  6. Take time 1
;  7. Run executable
;  8. Take time 2 if execution correct
;  9. Count the time of executable's work
; 10. Format the time
; 11. Display the time
;---------------------------------------------------------------------

  .DOSSEG                     ; DOS order of segments
  .MODEL  small               ; small memory model

;---------------------------------------------------------------------
;                          DATA TYPES DEFINITIONS
;---------------------------------------------------------------------
FPBYTE TYPEDEF FAR PTR BYTE     ; far pointer to the byte
PSEG   TYPEDEF WORD             ; pointer to the segment

;---------------------------------------------------------------------
; Structure of parameters' block (EPB) for EXEC function (4B00h) in MS
DOS
;---------------------------------------------------------------------
PARMBLK STRUCT
  env  PSEG ?          ; environment segment
  taddrFPBYTE    ?          ; parameters address
  fcb1 FPBYTE    ?                ; address of 1-st FCB
  fcb2 FPBYTE    ?             ; address of 2-nd FCB
PARMBLK ENDS

PPARMBLK  TYPEDEF PTR PARMBLK   ; pointer to the parameters' block

;---------------------------------------------------------------------
;                          MACROS
;---------------------------------------------------------------------

;---------------------------------------------------------------------
```

```
; PRINT - macro that displays string on the screen
; input:
;   text – string address
;-----------------------------------------------------------------------

PRINT  MACRO text
  push  AX                        ; push registers
  push  DX
  lea   DX,text                   ; DS:DX - address of the string ended
with '$'
  mov   AH,9                      ; display string function
  int   21h                       ; call dos function
  pop   DX
  pop   AX                        ; pop registers
ENDM

;-----------------------------------------------------------------------
; GETTICKS - macro that reads the ticks of system clock
; var – address of the place for result
;-----------------------------------------------------------------------

GETTICKS  MACRO  var
  push  AX                        ; push registers
  push  CX
  push  DX
  xor   AX, AX                    ; clear AX - read the clock function
  int   1ah                       ; Interrupt Time I/O
                                  ; Result: CX (hi), DX (lo) - ticks from
restart
  mov   word ptr var, DX          ; Store the result (lo)
  mov   word ptr var+2, CX        ; (hi)
  pop   DX
  pop   CX
  pop   AX                        ; pop registers
ENDM

;-----------------------------------------------------------------------
; macro that converts the number to ASCII string
;
; number – number to convert
; string – address of place for result, it should contain “0” in place
of
;         digits, for example: 00:00:00, 0000, 0.0.0
; pos    - count of digits
; divtab – table of digits in radix
;-----------------------------------------------------------------------

NUM2STRMACRO     number, string, pos, divtab
LOCAL  next, zero                ; local labels
  xor  EDX, EDX
  mov  ESI, number
  lea  BX, divtab
  lea  DI, string
  mov  CX, pos
next:  mov  EAX, ESI
  div  dword ptr [BX]
  cmp  byte ptr [DI], '0'  ; skips the separate characters
  je   zero
  inc  DI
zero:  or   [DI], AL       ; write the character
  inc  DI
  xor  EDX, EDX
  mul  dword ptr [BX]
  sub  ESI, EAX
  add  BX, 4                  ; next divider from the table
  loop next
ENDM

PGMSIZEEQU  500h              ; max. program size in paragraphs
;---------------------------------------------------------------------
;                          SEGMENTS
;---------------------------------------------------------------------
  .STACK                      ; stack segment
  .DATA                       ; data segment

_psp   PSEG 0                 ; pointer to the PSP segment
_env   PSEG 0                 ; pointer to the environment segment

Fspec  BYTE 250 DUP (0)            ; name of the program to execute
Tail   BYTE 300 DUP (0)            ; parameters for executable

Fcblk1 BYTE 0                 ; 1-st FCB
       BYTE 11 DUP (0)
       BYTE 25 DUP (0)
Fcblk2 BYTE 0                 ; 2-nd FCB
       BYTE 11 DUP (0)
       BYTE 25 DUP (0)

pb     PARMBLK     <>         ; parameters block

na_txt BYTE " Use: PERF program [parameters]",13,10,"$"
err_txtBYTE " DOS execution error", 13, 10, "$"
FPU_txtBYTE " This program needs FPU", 13, 10, "$"
perf_txt    BYTE " Time: "
hours  BYTE "00:00:00.00 ("
numba  BYTE "0000000 ticks )","$"

ticks1 DWORD     0                 ; starting time
ticks2 DWORD     0                 ; ending time
TPS    REAL8     0.182                  ; ticks/sec*100
```

```
;-----------------------------------------------------------------------
; converting tables
;-----------------------------------------------------------------------
divtab1LABEL     DWORD                  ; decimal positions
       IRP  val, <1000000,100000,10000,1000,100,10,1>
       DWORD     val
       ENDM


divtab2LABEL     DWORD                  ; time + hundred parts of second
       IRP  val, <3600000,360000,60000,6000,1000,100,10,1>
       DWORD     val
       ENDM

;-----------------------------------------------------------------------
; pogram code
;-----------------------------------------------------------------------

  .CODE                       ; code segment
  .STARTUP               ; startup code for DOS
  .486                   ; type of procesor

  int  11h               ; FPU presence check
  test AL, 2
  jnz  is_FPU
  PRINT     FPU_txt
  jmp  quit

is_FPU:
  mov  AH, 62h                ; get PSP address
  int  21h                     ; result in BX
  mov  ES, BX

  mov  _psp, ES
  mov  ax, ES:[2ch]           ; PSP segment
  mov  _env, AX

;-----------------------------------------------------------------------
; decreasing the program size
;-----------------------------------------------------------------------
  mov  AX, _psp               ; segment
  mov  ES, AX                  ; ES - segment of memory block
  mov  BX, PGMSIZE            ; new size
  mov  AH, 4ah                 ; Shrink or Expand Memory Block function
  int  21h

  mov  DX, DS
  mov  ES, DX
  mov  DS, _psp
  xor  CX, CX
  mov  CL, byte ptr [DS:80h]    ; length of parameters
  cmp  CL, 1
  jbe  no_args             ; no paremeter
  dec  CL              ; SPACE at the beginning

  mov  SI, 82h             ; first parameter is the name of the program
  lea  DI, FSpec               ; DI <- offset FSpec (FSpec - name of the
executable file)

lp1:
  dec  CL                      ; count length of parameters for
executable
  lodsb                           ; get character from DS:SI to AL and
inc SI
  cmp  AL, ' '             ; more parameters ...
  je   copy_args
  cmp  AL, 0Dh
  je   run                     ; program to execute without parameters
  stosb                           ; store character from AL to ES:DI and
inc DI
  jmp  lp1

copy_args:                 ; copy parameters for the program
        dec     SI                      ; copy with the leading space
        inc     CL
  mov  ES:Tail, CL             ; length of parameters
        inc     CL                      ; copy with the ending CR
  lea  DI, Tail+1
  rep  movsb                   ; copy DS:SI to ES:DI

run:
  mov  DS, DX                  ; DS - data segment
;-------------------------------------------------------------------------
; fill the parameters block
;-------------------------------------------------------------------------
  mov  AX, _env        ; environment segment
  mov  pb.env, AX              ; pb.env <- _env
  mov  AX, @data               ; @data points the data segment
  lea  BX, Tail                ; parameters string
  mov  word ptr pb.taddr[0], BX        ; program parameters address
  mov  word ptr pb.taddr[2], AX

  mov  AX, @data
  mov  BX, offset Fcblk1        ; 1-st FCB
  mov  word ptr pb.fcb1[0], BX         ; offset
  mov  word ptr pb.fcb1[2], AX         ; segment
  mov  BX, offset Fcblk2        ; 2-nd FCB
  mov  word ptr pb.fcb2[0], BX         ; offset
  mov  word ptr pb.fcb2[2], AX         ; segment
  lea  BX, pb              ; parameters block address in DS:BX
  push DS
  les  DI, (PARMBLK ptr [BX]).fcb1   ; 1-st FCB address to ES:DI
  lds  SI, (PARMBLK ptr [BX]).taddr  ; parameters address to DS:SI
  inc  SI
  mov  AX, 2901h      ; write file name to FCB
  int  21h                    ; Parse Filename function

       pop  ES
  les  DI, (PARMBLK ptr ES:[BX]).fcb2  ; 2-nd FCB to ES:DI
  mov  AX, 2901h      ; file name to FCB
  int  21h

  lea  BX, pb              ; parameters block address in ES:BX
  lea  DX, Fspec      ; program name address DS:DX

;-----------------------------------------------------------------------
; program execution
;-----------------------------------------------------------------------
  mov  AX, 4B00h      ; DOS:EXEC AL = 0 load and execute
  GETTICKS ticks1              ; starting time
  int     21h             ; program execution
        jc      err                     ; cecking if program executed
properly
  GETTICKS ticks2              ; execution time
  jmp  oki
err:
  PRINT     err_txt                 ; error message
  jmp  quit           ; error at the end

;-----------------------------------------------------------------------
; program execution time calculation
;-----------------------------------------------------------------------
oki:
  mov  EAX, ticks2             ; calculate execution time
  sub  EAX, ticks1
  mov  ticks1, EAX             ; result
  finit                           ; init the FPU
  fild ticks1                  ; load integer into FPU
  fdiv TPS                     ; convert ticks into seconds
  fistp     ticks2

;-----------------------------------------------------------------------
; formatting the result
;-----------------------------------------------------------------------
  NUM2STR ticks1, numba, 7, divtab1
  NUM2STR   ticks2, hours, 8, divtab2
  PRINT     perf_txt                ; result message
  jmp quit
```

```
;-----------------------------------------------------------------------
; if no parameters
;-----------------------------------------------------------------------
no_args:
  mov  DS, DX
  PRINT     na_txt                  ; no parameters message

quit:
  .EXIT 0                  ; return to MS DOS

END
;-----------------------------------------------------------------------
; end of the program
;-----------------------------------------------------------------------
```