Programming in Assembler

Laboratory manual

Exercise 2

Programming and Debugging tools

During the Exercise No.2 students are to debug simple program using the CodeView Debugger. The program is attached to the documentation in `lab2.asm` file.
CodeView allows not only debugging but also analyzing the programs to improve their speed and memory usage.

During the laboratory students are to:

1. Create the project with options for debugging and generating listing file.
   - The project description can be stored in `*.mak` file.
   - The project should have the file `lab2.asm` in it.
   - In the listing file should be: symbol table, machine codes, execution times.

2. Assemble the source to the `*.exe` file and run the program.

3. Analyze the output listing file `*.lst`
   - Put special attention how macros: `.STARTUP` and `.EXIT` are expanded.
   - Identify fields (prefix, opcode, arguments) in some more complex instructions.
   - Analyze execution time of instructions.

4. Run the CodeView debugger and analyze program execution line by line observing registers and flags.

5. Modify the program to call other procedures: `Seek_2`, `Seek_3` and `Seek_4` and analyze them using CodeView debugger.

6. Make a comparison of those four procedures. Compare execution time and memory usage of addressing modes in programs.

The report should consist of:

- Title page.
- Project file with explanation of lines and sections.
- Listing file with description of some instruction fields and execution time (especially the conditional jumps).
- Comparison of four memory addressing modes - execution time and memory usage.
- Conclusions.

Source code:

```
;******************************************************************************
;*                                                                            *
;*              LAB2.ASM - Assembler Laboratory ZMiTAC                        *
;*                                                                            *
;*  Sample program for seeking the character in the String variable          *
;*                                                                            *
;******************************************************************************

    TITLE   JA Lab1
    .MODEL  small, pascal
    .DOSSEG

    .STACK                          ; stack segment
    .DATA                           ; data segment
       String  DB  'AGIJKSZ', 0FFH  ; text string definition

    .CODE                           ; code segment
    .STARTUP                        ; beginning of the program
       CALL  Seek_1                 ; calling of first procedure (1)
 ;     CALL  Seek_2                 ; calling of second procedure (2)
 ;     CALL  Seek_3                 ; calling of third procedure (3)
 ;     CALL  Seek_4                 ; calling of fourth procedure (4)
    .EXIT 0                         ; end of the program

;*******************************************************************
;*    Procedure Seek_1 seeking for 'J' in the String variable      *
;*                                                                 *
;*      Index addressing mode                                      *
;*      Input parameters:                                          *
;*      Reg:  SI - offset of 'String' variable                     *
;*            AH - character to find 'J'                           *
;*      Output parameters:                                         *
;*            None                                                 *
;*                                                                 *
;*******************************************************************
  Seek_1  PROC  NEAR                ; Seek_1 procedure declaration
        MOV   SI, OFFSET String     ; load offset of 'String' variable to SI
        MOV   AH, 'J'               ; load 'J' code to AH
    Check_End_1:
        CMP   BYTE PTR [SI], 0FFH   ; end of the string ? (special char FF)
        JE    Not_Find_1            ; ending character found
        CMP   AH, [SI]              ; compare char with 'String' element
        JE    Got_Equal_1           ; character found!
        ADD   SI, 1                 ; increment the offset
        JMP   Check_End_1           ; seeking loop
    Got_Equal_1:
        MOV   DL, [SI]              ; load found character to DL
        JMP   Done_1
    Not_Find_1:
        MOV   DL, '?'               ; load '?' to DL
    Done_1:
        MOV   AH, 6               ; display character on the screen
        INT   21H
        RET                            ; return from the procedure
  Seek_1  ENDP                         ; end of Seek_1



;*******************************************************************
;*    Procedure Seek_2 seeking for 'J' in the String variable       *
;*                                                                  *
;*      Index addressing mode with displacement                     *
;*      Input parameters:                                           *
;*      Reg:  SI - offset of 'String' variable                      *
;*            AH - character to find 'J'                            *
;*      Output parameters:                                          *
;*            None                                                  *
;*                                                                  *
;*******************************************************************
  Seek_2  PROC  NEAR                   ; Seek_2 procedure declaration
        MOV   SI, 0                    ; load index of 'String' to SI
        MOV   AH, 'J'                  ; load 'J' code to AH
    Check_End_2:
        CMP   String[SI], 0FFH         ; end of the string ? (special char FF)
        JE    Not_Find_2               ; ending character found
        CMP   AH, String[SI]           ; compare char with 'String' element
        JE    Got_Equal_2              ; character found!
        ADD   SI, 1                    ; increment the index
        JMP   Check_End_2              ; seeking loop
    Got_Equal_2:
        MOV   DL, String[SI]           ; load found character to DL
        JMP   Done_2
    Not_Find_2:
        MOV   DL, '?'                  ; load '?' to DL
    Done_2:
        MOV   AH, 6                    ; display character on the screen
        INT   21H
        RET                            ; return from the procedure
  Seek_2  ENDP                         ; end of Seek_2


;*******************************************************************
;*    Procedure Seek_3 seeking for 'J' in the String variable       *
;*                                                                  *
;*      Base + Displacement addressing mode                         *
;*      Input parameters:                                           *
;*      Reg:  BX - offset of 'String' variable                      *
;*      Output parameters:                                          *
;*            None                                                  *
;*                                                                  *
;*******************************************************************
  Seek_3  PROC  NEAR
        MOV   BX, OFFSET String         ; load offset of 'String' to BX
        MOV   DL, 'J'                  ; load 'J' code to DL
        CMP   BYTE PTR [BX+0], 'J'     ; compare char with 'String' element
        JE    Got_It                   ; character found!
        CMP   BYTE PTR [BX+1], 'J'     ; compare char with 'String' element
        JE    Got_It                   ; character found!
        CMP   BYTE PTR [BX+2], 'J'  ; compare char with 'String' element
        JE    Got_It                ; character found!
        CMP   BYTE PTR [BX+3], 'J'  ; compare char with 'String' element
        JE    Got_It                ; character found!
        CMP   BYTE PTR [BX+4], 'J'  ; compare char with 'String' element
        JE    Got_It                ; character found!
        CMP   BYTE PTR [BX+5], 'J'  ; compare char with 'String' element
        JE    Got_It                ; character found!
        CMP   BYTE PTR [BX+6], 'J'  ; compare char with 'String' element
        JE    Got_It                ; character found!
    Not_Find_3:
        MOV   DL, '?'               ; load '?' to DL
    Got_It:
        MOV   AH, 6                 ; display character on the screen
        INT   21H
        RET                         ; return from the procedure
  Seek_3  ENDP                      ; end of Seek_3


;*********************************************************************
;*    Procedure Seek_4 seeking for 'J' in the String variable        *
;*                                                                   *
;*      Base + Index addressing mode                                 *
;*      Input parameters:                                            *
;*      Reg:  BX - offset of 'String' variable                       *
;*            SI - index                                             *
;*            AH - character to find 'J'                             *
;*      Output parameters:                                           *
;*            None                                                   *
;*                                                                   *
;*********************************************************************
  Seek_4  PROC  NEAR
        MOV   BX, OFFSET String       ; load offset of 'String' to BX
        MOV   SI, 0                   ; load index of 'String' to SI
        MOV   AH, 'J'                 ; load 'J' code to AH
    Check_End_4:
        CMP   BYTE PTR [BX+SI], 0FFH  ; end of the string ? (special char FF)
        JE    Not_Find_4              ; ending character found
        CMP   AH, BYTE PTR [BX+SI]    ; compare char with 'String' element
        JE    Got_Equal_4             ; character found!
        ADD   SI, 1                   ; increment the index
        JMP   Check_End_4             ; seeking loop
    Got_Equal_4:
        MOV   DL, [BX+SI]             ; load found character to DL
        JMP   Done_4
    Not_Find_4:
        MOV   DL, '?'                 ; load '?' to DL
    Done_4:
        MOV   AH, 6                   ; display character on the screen
        INT   21H
        RET                           ; return from the procedure
  Seek_4  ENDP                        ; end of Seek_4


END                                   ; koniec programu
```