Programming in Assembler

Laboratory manual

Exercise 6

Interrupts handling

During the Exercise No.6 students are to analyze the programs using the CodeView Debugger. On the laboratory students will know some functions for interrupt handling and calling the default handlers. Programs are attached to the documentation in `lab6_1.asm lab6_2.asm` files.

Explanation of DOS functions used in the program:

1. **25h** – Set interrupt vector
   AL = number of the interrupt;
   DS:DX = far address of the procedure.

2. **35h** – Get interrupt vector
   AL = number of the interrupt;
   Returns ES:BX = far address of the procedure.

During the laboratory students are to:

1. Create the projects to the `lab6_1.asm` `lab6_2.asm` files with options for debugging and generating listing file.
2. Assemble the projects to the `*.exe` files and run the programs.
3. Analyze the programs with attention to methods of original interrupt handler calling.
4. Modify the program `lab6_1.asm` to display the ASCII code in HEX.
5. Modify the program `lab6_2.asm` to recognize end of the sentence with "!" and "?" characters (not only ".").

The report should consist of:

- Title page.
- Explanation of programs function.
- Far addresses of original interrupt handlers and new interrupt handlers.
- Modified program listing file.
- Description of used interrupts (09h, 60h).
- Comparison of calling methods of original interrupt handler.
- Explanation why in the `lab6_1.asm` `lab6_2.asm` using the function number 08h of DOS interrupt 21h for reading the character is forbidden.
- Conclusions.

Source code (file LAB6-1.ASM):

```
;*********************************************************************
;*                                                                   *
;*              LAB6-1.ASM - Assembler Laboratory ZMiTAC             *
;*                                                                   *
;*  Sample program that displays the ASCII code of pressed key       *
;*                                                                   *
;*********************************************************************
.8086
.MODEL SMALL
.STACK 100h
.CODE
;*********************************************************************
;*Variables
*
;*********************************************************************
old_proc    dd 0           ; address of original interrupt handler

;*********************************************************************
;*09h interrupt handler
*
;*********************************************************************
segment_kb  EQU 40h             ; beggining of keyboard data segment
wsk_kb      EQU 1Ch             ; offset of pointer to keyboard buffer
kb_buf_begin     EQU 80h             ; offset of address of begining of
the buffer
kb_buf_end  EQU 82h             ; offset of address of end of the buffer

disp_segment     EQU 0B800h          ; adres of display data segment
up_right    EQU 154             ; offset of third character from left
side of the screen

keys    PROC FAR

;--------------------------------------------------------------------
; Calling of original interrupt handler
;--------------------------------------------------------------------
  pushf                     ; push flags
  call old_proc             ; call the interrupt hadler

;--------------------------------------------------------------------
; Prepare registers
;--------------------------------------------------------------------
  push ax                   ; push registers on the stack
  push bx
  push ds
  push es
  mov ax,segment_kb         ; address of keyboard data segment to DS
  mov ds,ax
  mov ax,disp_segment       ; address of display data segment to DS
```

```
  mov es,ax

;-------------------------------------------------------------------
; Read the character and display ASCII code
;-------------------------------------------------------------------
  mov bx,ds:[wsk_kb]       ; actual pointer to BX
  mov ax,ds:[kb_buf_begin] ; buffer beggining to AX
  cmp bx,ax           ; is the beggining of the buffer ?
  jne mid_buf
  mov bx,ds:[kb_buf_end]       ; last character is at the end of the
buffer
mid_buf:
  mov ax,ds:[bx-2]         ; read last character

  xor ah,ah           ; code is in AL
  mov bl,10
  div bl                   ; divide code by 10
  add ah,'0'               ; remainder to ASCII code
  mov es:[up_right+4],ah           ; display ASCII code
       xor ah,ah
  div bl                   ; divide result by 10
  add ah,'0'               ; convert remainder to ASCII
  add al,'0'               ; convert result to ASCII
  mov es:[up_right+2],ah           ; display ASCII
  mov es:[up_right],al         ; display ASCII

;-------------------------------------------------------------------
; Pop registers and return from interrupt
;-------------------------------------------------------------------
  pop es
  pop ds
  pop bx
  pop ax
  iret
keys    ENDP

;*******************************************************************
;*Main program                                                     *
;*******************************************************************
;-------------------------------------------------------------------
; Get interrupt
;-------------------------------------------------------------------

start: mov ah,35h              ; function 35h - read handler address
  mov al,09h               ; of interrupt 09h
  int 21h
  mov word ptr old_proc,bx ; store 32-bit address
  mov word ptr old_proc+2,es    ; of original interrupt handler
  push cs
  pop ds                   ; handler code segment to DS
```

```
  mov dx,offset keys       ; offset of handler address to DX
  mov ah,25h               ; function 25h - set new handler
  mov al,09h               ; of interrupt 09h
  int 21h

;--------------------------------------------------------------------------
; Main loop
;--------------------------------------------------------------------------
looping:mov ah,08h              ; function 08h - read character
  int 21h                  ; ASCII code is returned in AL
  cmp al,1Bh               ; ESC
  je ending           ; if ESC end of the loop
  mov dl,al           ; not ESC - move char to DL
  mov ah,02h               ; function 02h - display character
  int 21h                  ; ASCII code of char in DL
  jmp looping

;--------------------------------------------------------------------------
; Restore original interrupt handler
;--------------------------------------------------------------------------
ending:mov dx,word ptr old_proc+2
  mov ds,dx
  mov dx,word ptr old_proc
  mov ah,25h               ; function 25h - set old handler
  mov al,09h               ; of interrupt 09h
  int 21h

  mov ah,4Ch               ; end of the program
  int 21h
END    start
```

Source code (file LAB6-2.ASM):

```
;*********************************************************************
;*                                                                   *
;*              LAB6-2.ASM - Assembler Laboratory ZMiTAC             *
;*                                                                   *
;*  Sample program that converts lowercase to uppercase             *
;*  when key pressed                                                 *
;*********************************************************************
.8086
.MODEL SMALL
.STACK 100h
.CODE
;*********************************************************************
;*Variables
*
;*********************************************************************
old_proc    dd 0           ; address of original interrupt handler
dot_flag    db 0           ; dot flag

;*********************************************************************
;*09h interrupt handler
*
;*********************************************************************
segment_kb  EQU 40h            ; beggining of keyboard data segment
wsk_kb      EQU 1Ch            ; offset of pointer to keyboard buffer
kb_buf_begin     EQU 80h            ; offset of address of begining of
the buffer
kb_buf_end  EQU 82h            ; offset of address of end of the buffer

keys    PROC FAR

;---------------------------------------------------------------------
; Calling of original interrupt handler
;---------------------------------------------------------------------
  int 60h
;---------------------------------------------------------------------
; Prepare registers
;---------------------------------------------------------------------
  push ax                  ; push registers on the stack
  push bx
  push dx
  push ds
  mov ax,segment_kb        ; address of keyboard data segment to DS
  mov ds,ax

;---------------------------------------------------------------------
; Read the character and check ASCII code
;---------------------------------------------------------------------
  mov bx,ds:[wsk_kb]       ; actual pointer to BX
  mov ax,ds:[kb_buf_begin] ; buffer beggining to AX
```

```
  cmp bx,ax           ; is the beggining of the buffer ?
  jne mid_buf
  mov bx,ds:[kb_buf_end]   ; last character is at the end of the buffer
mid_buf:
  mov ax,ds:[bx-2]         ; read last character

  cmp al,'.'               ; compare with dot
  je dot_found             ; if dot
  cmp al,'Z'               ; compare with 'Z'
  ja check_lowercase       ; if above check lowercase
  cmp al,'A'               ; compare with  'A'
  jb keys_end              ; end if less

  mov dot_flag,0           ; uppercase - clear flag
  jmp keys_end                  ; return

check_lowercase:
  cmp al,'z'               ; compare with 'z'
  ja keys_end              ; end if above
  cmp al,'a'               ; compare with 'a'
  jb keys_end              ; end if less
  cmp dot_flag,0           ; was dot pressed?
  je keys_end                   ; end if not

;-------------------------------------------------------------------------
; Change lowercase to uppercase
;-------------------------------------------------------------------------
  sub al,'a'-'A'           ; sub difference between cases
  mov ds:[bx-2],ax
  mov dot_flag,0           ; uppercase - clear flag
  jmp keys_end                  ; return

dot_found:
  mov dot_flag,1                ; set flag
  jmp keys_end             ; return

;-------------------------------------------------------------------------
; Pop registers and return from interrupt
;-------------------------------------------------------------------------
keys_end:
  pop ds
  pop dx
  pop bx
  pop ax
  iret
keys    ENDP

;*************************************************************************
;*Main program                                                           *
;*************************************************************************
```

```
;-------------------------------------------------------------------------
; Get interrupt
;-------------------------------------------------------------------------

start: mov ah,35h               ; function 35h - read handler address
  mov al,09h               ; of interrupt 09h
  int 21h
  mov word ptr old_proc,bx ; store 32-bit address
  mov word ptr old_proc+2,es    ; of original interrupt handler
  push cs
  pop ds                   ; handler code segment to DS
  mov dx,offset keys       ; offset of handler address to DX
  mov ah,25h               ; function 25h - set new handler
  mov al,09h               ; of interrupt 09h
  int 21h
  mov dx,word ptr old_proc+2
  mov ds,dx
  mov dx,word ptr old_proc
  mov ah,25h               ; function 25h - set new adress
  mov al,60h               ; of original interrupt handler
  int 21h                  ; 60h instead of 09h

;-------------------------------------------------------------------------
; Main loop
;-------------------------------------------------------------------------
looping:mov ah,08h              ; function 08h - read character
  int 21h                  ; ASCII code is returned in AL
  cmp al,1Bh               ; ESC
  je ending           ; if ESC end of the loop
  mov dl,al           ; not ESC - move char to DL
  mov ah,02h               ; function 02h - display character
  int 21h                  ; ASCII code of char in DL
  jmp looping

;-------------------------------------------------------------------------
; Odtworzenie adresu pierwotnej procedury obslugi przerwania
;-------------------------------------------------------------------------
ending: mov dx,word ptr old_proc+2
  mov ds,dx
  mov dx,word ptr old_proc
  mov ah,25h               ; function 25h - set old handler
  mov al,09h               ; of interrupt 09h
  int 21h

  mov ah,4Ch               ; end of the program
  int 21h
END    start
```

Source code (file LAB6-3.ASM):

```
;*************************************************************************
;*                                                                       *
;*               LAB6-3.ASM - Assembler Laboratory ZMiTAC                *
;*                                                                       *
;*         Sample COM program that displays actual system time           *
;*                                                                       *
;*************************************************************************
.8086
.MODEL TINY           ; memory model for COM programs

;*************************************************************************
;*Macro that diplays two BCD digits                                      *
;*Parameters:                                                            *
;*8-bit register with two BCD digits                                     *
;*address in the display memory where digits has to be displayed         *
;*************************************************************************
BCD    MACRO register,address

  mov al,register             ; two digits to AL
  and al,0Fh          ; lower digit
  add al,'0'          ; convert to ASCII
  mov es:[address][2],al   ; display digit

  mov ah,register          ; two digits to AH
  and ah,0F0h         ; upper digit
  mov cl,4       ; shift 4 bits right
  shr ah,cl
  add ah,'0'          ; convert to ASCII
  mov es:[address],ah ; display digit

  ENDM

.CODE
  org 100h       ; place for PSP
start: jmp install         ; jump over PSP

;*************************************************************************
;*Variables                                                              *
;*************************************************************************
impulses    db 0      ; interrupt counter
old_proc    dd 0      ; address of original interrupt handler

disp_segment     EQU 0B800h     ; adres of display data segment
up_right    EQU 144        ; offset of eight character from left side of
the screen

;*************************************************************************
;*1Ch interrupt handler                                                  *
;*************************************************************************
```

```
clock: push ax              ; push registers on the stack
        push bx
        push cx
        push dx
        push ds
        push es

        mov ax,cs           ; set DS register
        mov ds,ax

  inc impulses          ; increment counter
  cmp impulses,9        ; display every 0,5 second
        jne ending

        mov ax,disp_segment     ; load display segment address to ES
  mov es,ax
  mov impulses,0        ; clear counter

  mov ah,02h            ; read time
  int 1ah               ; function 02 of BIOS interrupt 1Ah
  mov bx,cx
  BCD bh,up_right             ; display hours
  BCD bl,up_right+6     ; display minuts
  BCD dh,up_right+12    ; display seconds

ending:pop es               ; restore registers
  pop ds
  pop dx
  pop cx
  pop bx
  pop ax
;-------------------------------------------------------------------------
; Jump to original interrupt handler
; and return from interrupt
;-------------------------------------------------------------------------
  jmp dword ptr cs:[old_proc]

;*************************************************************************
;*Main program                                                           *
;*************************************************************************
install:
;-------------------------------------------------------------------------
; Get interrupt
;-------------------------------------------------------------------------
  mov ah,35h                ; function 35h - read handler address
  mov al,1ch                ; of interrupt 1ch
  int 21h
  mov word ptr old_proc,bx ; store 32-bit address
  mov word ptr old_proc+2,es    ; of original interrupt handler
  mov dx,offset clock       ; offset of handler address to DX
```

```
    mov ah,25h                ; function 25h - set new handler
    mov al,1ch                ; of interrupt 1ch
    int 21h

;-------------------------------------------------------------------------
; Main loop
;-------------------------------------------------------------------------
    mov ah,08h          ; ending address of program in the memory
    int 21h

;-------------------------------------------------------------------------
; Restore interrupt
;-------------------------------------------------------------------------

    mov dx,word ptr old_proc+2
    mov ds,dx
    mov dx,word ptr old_proc
    mov ah,25h                ; function 25h - set old handler
    mov al,1ch                ; of interrupt 09h
    int 21h

    mov ah,4Ch
    int 21h
END     start
```

Source code (file LAB6-4.ASM):

```
;#########################################################################
;       Laboratorium Jezykow Asemblerowych - Cwiczenie 6
;       Zadanie 4
;       1. Dokonac asemblacji programu.
;       2. Wykorzystujac debuger podac pelny 32-bitowy adres pierwotnej
;          procedury obslugi przerwania 16h oraz procedury napisanej w
;          programie.
;       3. Przeanalizowac program ze szczegolnym uwzglednieniem:
;          - sposobu wywolania pierwotnej procedury obslugi przerwania
;          - sposobu pozostawienia czesci programu w pamieci jako TSR
;       4. Zaprogramowac w programie z poprzedniego zadania (cw6_3.asm)
;          zaiane sposobu pozostawiania
;          programu w pamieci jako rezydentnego na sposob wykorzystujacy
;          funkcje 31h przerwania 21h systemu MS-DOS.
;#########################################################################

;#########################################################################
;# Zamiana klawiszy Y i Z na klawiaturze                                 #
;#########################################################################
;.8086
;.MODEL SMALL

dane    SEGMENT
;#########################################################################
;# Zmienne                                                               #
;#########################################################################
stara_proc dd 0
funkcja    db 0
dane    ENDS

kod     SEGMENT
  ASSUME cs:kod,ds:dane
;#########################################################################
;# Procedura obslugi przerwania 16h                                      #
;#########################################################################
zamianaPROC FAR
  mov funkcja,ah      ; zapamietanie numeru wywolanej funkcji
;-------------------------------------------------------------------------
; Wywolanie pierwotnej procedury obslugi przerwania
;-------------------------------------------------------------------------
  pushf               ; odlozenie na stos rejestru znacznikow
  call stara_proc          ; wywolanie pierwotnej procedury obslugi
  push bp             ; zapamietanie rejestru bp
  mov bp,sp      ; aktualny adres wierzcholka stosu do bp
  push ax             ; zapamietanie rejestru ax
  pushf               ; przepisanie zawartosci rejestru znacznikow
  pop ax              ; do rejestru ax
  mov [bp+6],ax       ; modyfikacja odlozonych na stosie znacznikow
  pop ax              ; odtworzenie rejestru ax
```

```
  pop bp              ; odtworzenie rejestru bp

;-------------------------------------------------------------------------
; Sprawdzenie numeru wywolanej funkcji
;-------------------------------------------------------------------------
  cmp funkcja,00h          ; czy ktoras z funkcji czytajacych znak ?
  je zamiana_yz       ; jesli tak to zamiana liter
  cmp funkcja,10h
  je zamiana_yz
  cmp funkcja,01h
  je zamiana_yz
  cmp funkcja,11h
  je zamiana_yz
  jmp koniec          ; jesli inna funkcja to powrot

;-------------------------------------------------------------------------
; Sprawdzenie i zamiana kodu nacisnietego klawisza
;-------------------------------------------------------------------------
zamiana_yz:
  cmp al,'z'          ; jesli nacisnieto 'z'
  jnz nie_z_male
  mov al,'y'          ; to funkcja zwraca 'y'
  jmp koniec

nie_z_male:
  cmp al,'y'          ; jesli nacisnieto 'y'
  jnz nie_y_male
  mov al,'z'          ; to funkcja zwraca 'z'
  jmp koniec

nie_y_male:
  cmp al,'Z'          ; jesli nacisnieto 'Z'
  jnz nie_z_duze
  mov al,'Y'          ; to funkcja zwraca 'Y'
  jmp koniec

nie_z_duze:
  cmp al,'Y'          ; jesli nacisnieto 'Y'
  jnz nie_y_duze
  mov al,'Z'          ; to funkcja zwraca 'Z'
  jmp koniec

nie_y_duze:
koniec:
  iret
zamianaENDP

;#########################################################################
;# Program glowny                                                        #
;#########################################################################
```

```
;-------------------------------------------------------------------------
; Przejecie przerwania
;-------------------------------------------------------------------------
start: push ds                  ; zachowanie adresu segmentu naglowka
PSP
  mov ax,dane               ; zaladowanie rejestru segmentowego
  mov ds,ax
  mov ah,35h                ; funkcja 35h - odczytanie adresu procedury
  mov al,16h                ; przerwania 16h
  int 21h
  mov word ptr stara_proc,bx    ; zapamietanie 32-bitowego adresu
  mov word ptr stara_proc+2,es  ; pierwotnej procedury obslugi
  push cs
  pop ds                    ; do ds segment kodu w którym jest procedura
  mov dx,offset zamiana         ; do dx offset adresu procedury
  mov ah,25h                ; funkcja 25h - nowa procedura obslugi
  mov al,16h                ; przerwania 16h
  int 21h
  pop ds                    ; odtworzenie adresu segmentu naglowka PSP

;-------------------------------------------------------------------------
; Instalacja czesci rezydentnej
;-------------------------------------------------------------------------
  mov bx,cs             ; roznica adresu poczatku segmentu kodu
  mov ax,ds             ; i adresu poczatku segmentu naglowka PSP
  sub bx,ax             ; daje rozmiar wszystkich posrednich segmentow
  mov dx,offset start+15        ; adres do ktorego program zostanie w
pamieci
  mov cl,4              ; podzielony przez 16
  shr dx,cl
  add dx,bx             ; zsumowany z roznica adresow segmentow
  xor al,al             ; kod powrotu=0
  mov ah,31h                ; funkcja instalujaca czesc rezydentna
  int 21h
kod    ENDS
END    start

;#########################################################################
;#Koniec programu zamiany liter y i z                                    #
;#########################################################################
```