RĪGAS TEHNISKĀ UNIVERSITĀTE
Datorzinātnes un informācijas tehnoloģijas fakultāte
Lietišķo datorsistēmu institūts
Lietišķo datorzinātņu katedra

prof. Uldis Sukovskis

Datoru organizācija un asambleri

P I E M Ē R I

Saturs

Vienkāršas COM programmas piemērs

```
; Illustrates full segment directives for COM program

TEXT        SEGMENT                         ; Code segment
            ASSUME  cs:TEXT, ds:TEXT
            ORG     100h

start:      jmp   go

msg         DB    "Sveiks!", 7, 13, 10, "$"

go:         mov   ah, 9h            ; Request operating system Function 9
            mov   dx, OFFSET msg    ; Load DX with offset of string
                                    ;   (segment already in DS)
            int   21h               ; Display String to Standard Output

            int   20h         ; Exit

TEXT        ENDS
            END   start       ; End with reference to first statement
```

```
; Illustrates simplified segment directives for COM program

            .MODEL      tiny

            .DATA
msg         DB    "Sveiks!", 7, 13, 10, "$"

            .CODE
            .STARTUP

            mov   ah, 9h            ; Request operating system Function 9
            mov   dx, OFFSET msg    ; Load DX with offset of string
                                    ;   (segment already in DS)
            int   21h               ; Display String to Standard Output

            .EXIT 0
            END
```

Vienkāršas EXE programmas piemērs

```
; Illustrates full segment directives for EXE program

ASSUME  cs:CSEG, ds:DSEG, ss:SSEG

CSEG        SEGMENT; Code segment
begin:      mov    ax, DSEG         ; Set data segment
            mov    ds, ax

            mov    ah, 9h           ; operating system function 9
            mov    dx, OFFSET msg   ; Load DX with offset of string
                                    ; (segment is in DS)
            int    21h              ; Display string to standard output

            mov    ah, 4ch          ; operating system function 4ch
            mov    al, 0            ; Return code
            int     21h             ; Return to operating system
CSEG        ENDS

DSEG        SEGMENT; Data segment
msg         db     "Sveiks!", 7, 13, 10, "$"
DSEG        ENDS

SSEG        SEGMENT STACK           ; Stack segment
            dw     64 dup(0)
SSEG        ENDS

            END    begin            ; End with reference to first statement
```

Vienkāršas COM programmas piemērs ar skaitlisku rezultātu izvadi

```
code      segment
          assume            cs:code, ds:code
          org   100h

start:    jmp   go

string    db    '01234567891*ABC', 0
buf       db    '000000$'

go:       mov   si,0
          mov   ah,'*'
check:    cmp   string[si],0
          je    notfound
          cmp   ah,string[si]
          je    found
          inc   si
          jmp   check

found:
          inc   ax
          mov   ax,si
          mov   si,5
          mov   bl,10
d:        div   bl          ; ax/bl =  ah - atlikums, al -dalijums
          add   ah,30h      ; make ASCII digit
          mov   buf[si],ah  ;
          cmp   al,0        ; dalijums = 0?
          je    put
          mov   ah,0
          dec   si
          jmp   d
put:
          mov   ah,9
          mov   dx, offset buf
          int   21h
          jmp   done
notfound:
          mov   dl,'?'
          mov   ah,6
          int   21h
done:
          int   20h

code      ends
          end   start
```

Operētājsistēmas funkcijas ievadei un izvadei

```
text    segment
        assume  cs:text,ds:text
        org     100h

start:  jmp     go

inbuf   equ     this byte
maxlen  db      20
actlen  db      0
intext  db      20 dup(0)

chr     db      ?
msg1    db      'Ievadi
simbolu:$'
msg2    db      'Ievadi virkni:$'
newline db      13,10,'$'
outbuf  db      '00000$'


go:
        mov     ah, 9
        mov     dx,offset msg1
        int     21h

        mov     ah,1
        int     21h
        mov     chr,al

        mov     ah,9
        mov     dx,offset newline
        int     21h

        mov     ah, 9
        mov     dx,offset msg2
        int     21h

        mov     ah, 0ah
        mov     dx,offset inbuf
        int     21h

        mov     ah,9
        mov     dx,offset newline
        int     21h

        cmp     actlen,0
        je      notext

        xor     ax,ax
        xor     cx,cx
        mov     cl,actlen
        xor     si,si
        mov     dl,chr




c:      cmp     intext[si],dl
        jne     next
        inc     ax
next:   inc     si
        loop    c

output:
        mov     si,4
        mov     bl,10
d:      div     bl
        add     ah,30h
        mov     outbuf[si],ah
        cmp     al,0
        je      put
        mov     ah,0
        dec     si
        jmp     d

notext:
put:
        mov     ah,9
        mov     dx, offset outbuf
        int     21h

        int     20h

text    ends
        end     start
```

Parametru saņemšana no komandrindas *COM* programmā

```
EXPARM      SEGMENT
            ASSUME CS:EXPARM, DS:EXPARM
            ORG    100H
start:      jmp    go
;           ...
go:         xor    cx,cx
            mov    cl,ds:[80h]            ; length of command line
            cmp    cx,0
            jna    noparms
            mov    si,81h                 ; offset of parameters in PSP
chklwr:     cmp    byte ptr [si],'a'      ; convert
            jb     nolwr                  ; command
            cmp    byte ptr [si],'z'      ; line
            ja     nolwr                  ; characters
            sub    byte ptr [si],32       ; to
            mov    [si],al                ; uppercase
nolwr:      inc    si                     ;
            loop   chklwr                 ;
;           ...process parm list ...
;           ...
noparms:
;           ...
EXPARM      ENDS
            END    start
```

Programma ar apakšprogrammu, kura saņem parametrus reģistros

```
CSEG        SEGMENT
            ASSUME cs:CSEG
            ORG    100h
start:      jmp    go
wrd         dw     0ffh
buf         db     '00000$'

; Procedure counts ones in the first CX bits of register AX.
; Result is in BX.

ones        proc   near
            push   ax
            push   cx
            xor    bx,bx
  tst:      test   ax,0001h
            jz     next
            inc    bx
  next:     shr    ax,1                    ; shift right
            loop   tst
            pop    cx
            pop    ax
            ret
ones        endp
go:
            mov    ax,wrd
            mov    cx,16
            call   ones
; ...
; convertion of binary value of BX to decimal ASCII string and output.
; ...
            int    20h
CSEG        ENDS
            END    start
```

Programma ar apakšprogrammu, kura saņem parametrus stekā

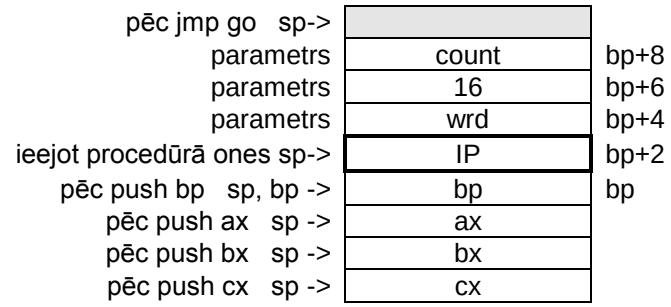
```
CSEG          SEGMENT
              ASSUME  cs:CSEG
              ORG     100h
start:        jmp     go
wrd           dw      005fh
count         dw      ?
buf           db      '00000$'

ones          proc    near
              push    bp
              mov     bp,sp
              push    ax
              push    bx
              push    cx
              mov     bx,0
              mov     cx,[bp+6]
              mov     ax,[bp+4]
  tst:        test    ax,0001h
              jz      next
              inc     bx
  next:       shr     ax,1
              loop    tst
              mov     [bp+8],bx
              pop     cx
              pop     bx
              pop     ax
              pop     bp
              ret     4
ones          endp

go:           push    count                     ;bp+8
              push    16                        ;bp+6
              push    wrd                       ;bp+4
              call    ones
              pop     count
; convertion of binary value to decimal ASCII string and output.
              mov     ax,count

;             ...

put:          mov     ah,9
              mov     dx, offset  buf
              int     21h
              int     20h
CSEG          ENDS
              END     start
```



Procedūras kompilēšana atsevišķā failā

CALLMAIN.ASM

```
CSEG    SEGMENT
        EXTRN  ones:far
        ASSUME  cs:CSEG
        ORG     100h
start:  jmp     go
wrd     dw      005fh
count   dw      0000h
buf     db      '00000$'

go:
        push    count   ;bp+8
        push    16      ;bp+6
        push    wrd     ;bp+4
        call    ones
        pop     count
        mov     ax,count
        mov     si,4
        mov     bl,10
    d:  div     bl              ; ax/bl =  ah - atlikums
        add     ah,30h          ; make ASCII digit
        mov     buf[si], ah
        cmp     al,0            ; dalîjums = 0?
        je      put
        mov     ah,0
        dec     si
        jmp     d
put:
        mov     ah,9
        mov     dx, offset  buf
        int     21h
        int     20h
CSEG    ENDS
        END     start
```

ONES.ASM

```
CSEG SEGMENT

        PUBLIC ones
        ASSUME CS:CSEG
ones    proc    far
        push    bp
        mov     bp,sp
        push    cx
        mov     word ptr [bp+10],0
        mov     cx,[bp+8]
  tst:  test    word ptr [bp+6],0001h
        jz      next
        inc     word ptr [bp+10]
  next: shr     word ptr [bp+6],1
        loop    tst
        pop     cx
        pop     bp
        ret     4
ones    endp
CSEG    ENDS
        END
```

```
tasm ones
tasm callmain
tlink /t callmain+ones,callmain
```

Rezidenta klaviatūras pārtraukuma apstrādes programma

```
kbd         segment
            assume                          cs:kbd
            org   100h
start:      jmp   go
flag        db    '123456'
oldint9     dd    0
status      db    08h                       ; Alt
scan        db    1                         ; Esc
int9h       proc  far                       ; Interrupt handler
            push  ds
            push  es
            push  ax
            push  bx
            push  cx
            mov   bx,cs
            mov   ds,bx
            xor   bx,bx
            mov   es,bx
            test  byte ptr es:[0417h],20h   ; Numlock status ?
            jz    getscan                   ; OFF - go on
            jmp   retold                    ; ON  - return
getscan:    in    al,60h
            mov   ah,status
            and   ah,es:[0417h]
            cmp   ah,status                 ; status ?
            jne   retold
            cmp   al,scan                   ; scan code  ?
            jne   retold

            mov   ax,0b800h
            mov   es,ax
            mov   byte ptr es:[0],65        ; character 'A'
            mov   byte ptr es:[1],16*12+15 ; attribute
            jmp   rethw
retold:
            pop   cx
            pop   bx
            pop   ax
            pop   es
            pop   ds
            jmp   [oldint9]

rethw:      in    al,61h                    ; hardware housekeeping
            mov   ah,al                     ;
            or    al,80h                    ;
            out   61h,al                    ;
            xchg  ah,al                     ;
            out   61h,al                    ;
            mov   al,20h                    ;
            out   20h,al                    ;
            pop   cx
            pop   bx
            pop   ax
            pop   es
            pop   ds
            iret
int9h       endp
```

```
highbyte    equ   this byte
ownflag     db    'LRKBDU'
msgok       db    'Keyboard   Driver  installed',13,10,'$'
msgerr      db    'Keyboard driver is already active!',7,13,10,'$'
env         dw    0
go:         xor   cx,cx
            mov   cl,ds:[80h]             ; length of command line
            cmp   cx,0
            jna   noparms
            mov   si,81h                  ; offset of parms in PSP
chklwr:     cmp   byte ptr [si],'a'       ; convert
            jb    nolwr                   ; command
            cmp   byte ptr [si],'z'       ; line
            ja    nolwr                   ; characters
            sub   byte ptr [si],32        ; to
            mov   [si],al                 ; uppercase
nolwr:      inc   si                      ;
            loop  chklwr                  ;
;           ...process parm list ...
noparms:
;-----------------------------------------
            mov   ax,3509h                ; get vector
            int   21h                     ; es = segment from vector
            mov   di,offset flag
            mov   si,offset ownflag
            mov   cx,6
            repe  cmpsb                   ; es:di  ==  ds:si  ?
            jne   install                 ; flags do not match - install
            mov   dx,offset msgerr        ; flags match - message
            mov   ah,9
            int   21h
            int   20h
;-----------------------------------------
install:    mov   si,offset ownflag       ; set flag
            mov   di,offset flag
            mov   ax,ds
            mov   es,ax
            mov   cx,6
            rep   movsb                   ; ds:si -> es:di
;-----------------------------------------
            mov   ax,3509h                ; get vector
            int   21h
            mov   word ptr oldint9,bx
            mov   word ptr oldint9+2,es
            mov   dx,offset int9h         ; set vector
            mov   ax,2509h
            int   21h
;-----------------------------------------
            mov   dx,offset msgok
            mov   ah,9
            int   21h
;-----------------------------------------
            mov   es,ds:[2ch]             ; Environment seg from PSP
            mov   ah,49h
            int   21h                     ; release env seg
;-----------------------------------------
            mov   dx,offset highbyte + 10h
            int   27h
kbd         ends
            end   start
```

Darbs ar videoterminālu grafiskajā 16 krāsu režīmā

setpx.c

```
void setpx(unsigned short x, unsigned short y, unsigned short c)
{      _asm{  mov    ax, y
              mov    dx, 80
              mul    dx                  ;ax = y * 80
              mov    bx, x
              mov    cl, 3
              shr    bx, cl              ;bx = x / 8
              add    bx, ax              ;offset = ax + bx
              mov    ax, 0a000h          ;segment of the video page 0
              mov    es, ax

              mov    cx, 7               ;mask
              and    cx, x               ;get 3 bits from x
              mov    ah, 80h
              shr    ah, cl              ;make the mask of bits

              mov    dx, 3CEh            ;addr. reg.
              mov    al, 5               ;reg. 5 - mode reg
              out    dx, al
              inc    dx                  ;data reg. 3CFh
              mov    al, 2               ;mode = 2
              out    dx, al

              mov    dx, 3CEh            ;addr. reg.
              mov    al, 8               ;reg. 8
              out    dx, al
              inc    dx                  ;data reg. 3CFh
              mov    al, ah              ;mask of bits
              out    dx, al

              mov    dx, 3C4h            ;sequencer addr. reg.
              mov    al, 2               ;reg. 2 - map mask
              out    dx, al
              inc    dx                  ;data reg. 3C5h
              mov    al, 0Fh             ;mask of planes = all
              out    dx, al

              mov    al, es:[bx]         ;set latch registers
              mov    ax, c               ;color
              mov    es:[bx], al         ;set pixel
       }
}
```

graph.c

```
#include <graphics.h>
#include <conio.h>
void main()
{      void setpx(unsigned short x, unsigned short y, unsigned short c);
       int x, y;
       int driver = VGA, mode = VGAHI;
       initgraph(&driver, &mode,"");
       for (x = 0  ; x < 640; ++x)
       for (y = 100; y < 200; ++y) putpixel(x, y, x+y);
       for (x = 0  ; x < 640; ++x)
       for (y = 300; y < 400; ++y) setpx(x, y, x*y);

       getch();
       restorecrtmode();
}
```

Taimera programmēšana. Skaņas ģenerēšana

```
TEXT    SEGMENT
        ASSUME  cs:TEXT, ds:TEXT
        ORG     100h
start:  jmp     go

msg1    DB      "Start", 13, 10, "$"
msg2    DB      "Stop", 13, 10, "$"
go:
        mov     al, 10110110b   ;10-ch,11-2 bytes,011-regime,0-bin
        out     43h, al         ; command
        mov     ax, 1193        ; count = 1193180 / 1000Hz
        out     42h, al
        mov     al, ah
        out     42h, al

        in      al, 61h         ; read port
        push    ax              ; and save
        or      al,03h          ; enable gate and speaker
        out     61h, al

        mov     ah, 9
        lea     dx, msg1
        int     21h
                                ;delay loop
        mov     cx, 1000
l2:     push    cx
        mov     cx, 30000
l1:     loop    l1
        pop     cx
        loop    l2

        pop     ax              ;restore port value
        out     61h, al

        mov     ah, 9
        lea     dx, msg2
        int     21h

        int     20h

TEXT    ENDS
        END     start
```

Diska *boot sector* nolasīšana

```
;........

buffer      db          512 dup (0)
boot        equ         buffer
res         db          11 dup (0)
sectSize    dw          0
clustSize   db          0
resSects    dw          0
fatCount    db          0
rootSize    dw          0
totalSects  dw          0
media       db          0
fatSize     dw          0
trackSects  dw          0
heads       dw          0
hinSects    dw          0

; .......

; read disk information
            mov ah,36h  ; operating system function
            mov dl,3    ; 0-current, 1-A, 2-B, ...
            int 21h
; ax = sect per cluster
; bx = available clusters
; cx = bytes per sector
; dx = clusters per drive

; read boot sector
            mov dl, 0   ; 0-A, 1-B, ...
            mov dh, 0   ; head
            mov ch, 0   ; cyl
            mov cl, 1   ; sector
            mov al, 1   ; count
            mov ah, 2   ; read
            mov bx, offset boot     ;es:bx buffer
            int 13h

; read boot sector using operating system
            mov al, 0   ; 0-A, 1-B, ...
            mov cx, 1   ; count
            mov dx, 0   ; sector number 0,1,....
            mov bx, offset boot     ;ds:bx buffer
            int 25h

;........
```