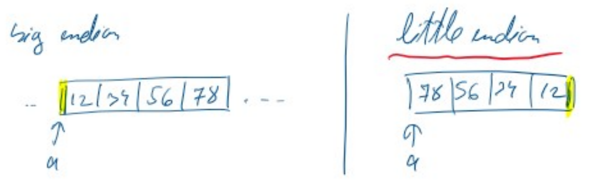
Seminar III 212

Tuesday, October 27, 2020 11:51 AM

a dd 12345678h

big endian | little endian



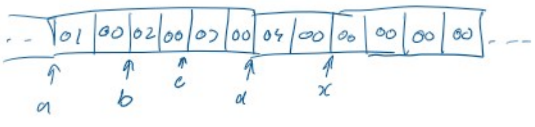
adc d,s ; d ← d+s+CF

sbb d,s ; d ← d−s−CF

$x = a\cdot b + c\cdot d$

a,b,c,d cuvinte fără semn

```
.data        00|01|
  a  dw 1
  b  dw 2
  c  dw 3
  d  dw 4
  x  dd 0
```

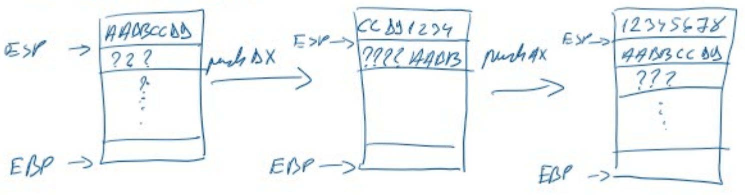


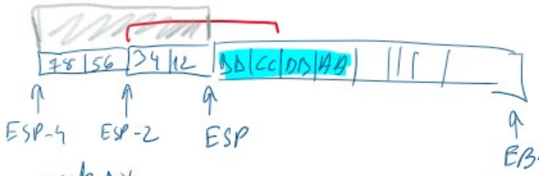
```
.code
start:
    mov ax, [a]
    mul word [b]; DX:AX = a·b
    push DX
    push AX
    pop ebx
    mov ax, [c]
    mul word [d]
    push dx
    push ax
    pop eax
    add eax, ebx ; eax=a·b+c·d
    mov [x], eax
```

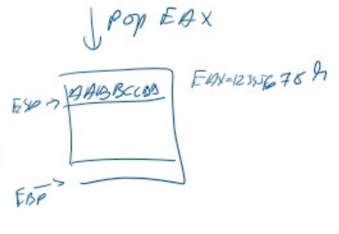
```
mov [x], ax
mov [x+2], dx
mov ax, [c]
mul word [d]
add [x], ax      ← clc
adc [x+2], dx
```

```
push DX          ~~push AX~~
push AX          ~~push DX~~
pop EAX          ~~pop EAX~~
```

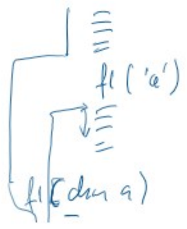
DX=1234h ; AX=5678h





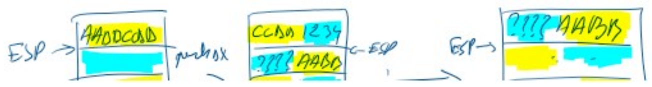


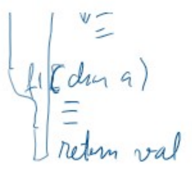
push DX

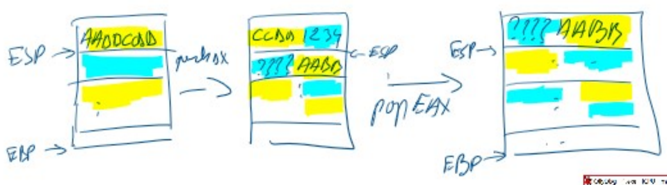


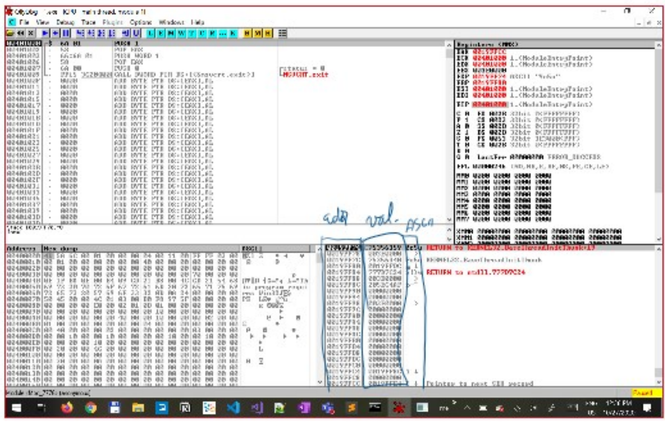
push DX ;DX=1234h

pop EAX









```
File  View  Debug  Trace  Plugins  Options  Windows  Help

PUSH 1
POP EAX
PUSH WORD 1
POP EAX
PUSH 0
CALL DWORD PTR DS:[<&msvcrt.exit>]        status = 0
                                          MSVCRT.exit
ADD BYTE PTR DS:[EAX],AL
ADD BYTE PTR DS:[EAX],AL
ADD BYTE PTR DS:[EAX],AL

Registers (MMX)

RETURN to KERNEL32.BaseThreadInitThunk+19
RETURN to ntdll.77D07C24
Pointer to next SEH record
```

- instr. condiționale de salt

"if"

```
if ( ) {
} else {
  ...
}
```

```
cmp  D , S      ;  D-S și modifică reg. EFLAGS
                   CF, ZF, SF, OF
```

$D \gtreqless S$

instr. de salt care interpretează nr. fără semn

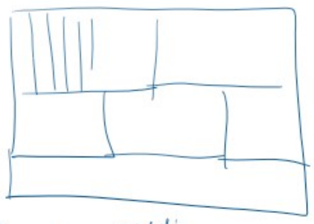
$D = S \begin{cases} CF = 0 \\ ZF = 1 \end{cases}$ $\quad D < S \begin{cases} CF = 1 \\ ZF = 0 \end{cases}$

$D > S \begin{cases} CF = 0 \\ ZF = 0 \end{cases}$

```
jb etichetă  ; salt la eticheta dacă  D < S
jbe  —"—     ; —"—    —"—            D ≤ S
ja   —"—       —"—    —"—            D > S
je   —"—       —"—    —"—            D = S
jnbe —"—       —"—    —"—            D > S
```

nr cu semn   SF, OF

```
jl etichetă
jle  —"—
jg   —"—
je   —"—
jnle —"—
```



segment_selector : offset

16 biți, 32 biți

```
mov  ax, [a]
```

$$\text{offset} = [\text{baza}] + [\text{index} * \text{scala}] + [\text{ct}]$$

$$\begin{bmatrix} EAX \\ \vdots \\ EDI \end{bmatrix} \left[\begin{bmatrix} EAX \\ \vdots \\ \cancel{ESP} \\ EDI \end{bmatrix} * \begin{pmatrix} 1 \\ 2 \\ 4 \\ 8 \end{pmatrix} \right] \begin{bmatrix} \text{nimic} \\ \text{byte} \\ \text{word} \\ \text{dword} \end{bmatrix}$$

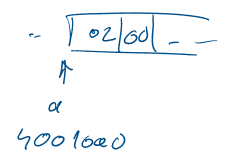
```
     D    S
mov  ax,[eax]       ;  mov ax, word [eax]
mov  ax, [a+eax+2] ⇐
mov  eax, [ a+2 + ebx·2]
                              a dw 2
mov  ax,[a]
```

[illegible]

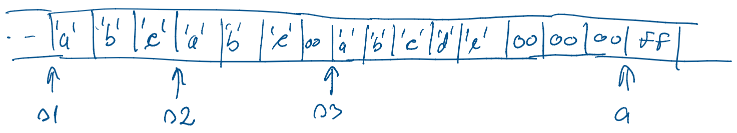
```
mov eax, [a+7+ebx·2]

mov ax, [a]
```

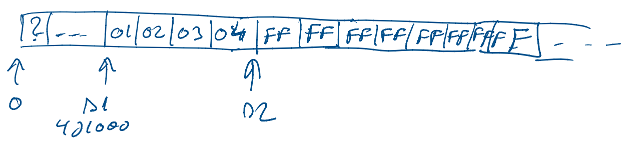
a dw 2

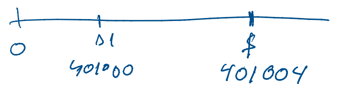


```
.data
s1 db 'abc'
s2 dw 'abc'
s3 dd 'abcde'
a  db -1
```



```
.data
s1 db 1,2,3,4
l_s1 equ $-s1
s2 times l_s1 dw -1
```





```
bits 32

global start

extern exit
import exit msvcrt.dll

segment data use32 class=data
   s1  db  1,2,3,4
   l_s1    equ $-s1
   s2  times l_s1 dw -1

segment code use32 class=code
start:
   ; var 1: folosesc reg ca si index in sir (sir[i])
   ;mov esi, 0; esi - index in sir

   ;repeta_instr:
      ; mov al, [s1+esi]
       ;mul al
       ;mov [s2+esi*2], ax

       ; i++
       ;inc esi

       ;for
       ;cmp esi, l_s1
       ;jb repeta_instr

   ; var 2: [sir]
   mov esi, s1; esi - adresa s1
   ; val initiala esi = 401000
   mov edi, s2; edi - adresa s2
   mov ecx, 0
   repeta_instr:
       mov al, [esi]
       mul al
       mov [edi], ax


       inc esi
       add edi, 2

       ;for (i, i<, i++)
       inc ecx
       cmp ecx, l_s1
       jb repeta_instr

   push    dword 0
   call    [exit]
```