

## Prvi domaći rad

**Tekst zadatka:** Napisati assembler program koji proverava da li je rezultat zbira dva broja paran ili ne. Parnost sačuvati kao informaciju u registru SI (ako je broj paran postaviti ga na 1, u suprotnom na 0).

**Rešenje zadatka:**

```
data_seg SEGMENT
```

```
    br1 dw 5
```

```
    br2 dw 2
```

```
    rez dw ?
```

```
data_seg ENDS
```

```
code_seg SEGMENT
```

```
    ASSUME cs:code_seg, ds:data_seg
```

```
start: MOV dx, offset data_seg
```

```
    MOV ds, dx
```

```
    MOV ax, br1
```

```
    ADD ax, br2
```

```
    MOV rez, ax
```

```
    MOV ax, rez
```

```
    MOV bl, 2
```

```
    DIV bl
```

```
    CMP ah, 0
```

```
    je paran
```

```
    CMP ah, 1
```

```
    jmp kraj
```

```
paran:
```

```
    mov si, 1
```

```
kraj: jmp kraj
```

```
end start
```

```
code_seg ENDS
```

```
END
```

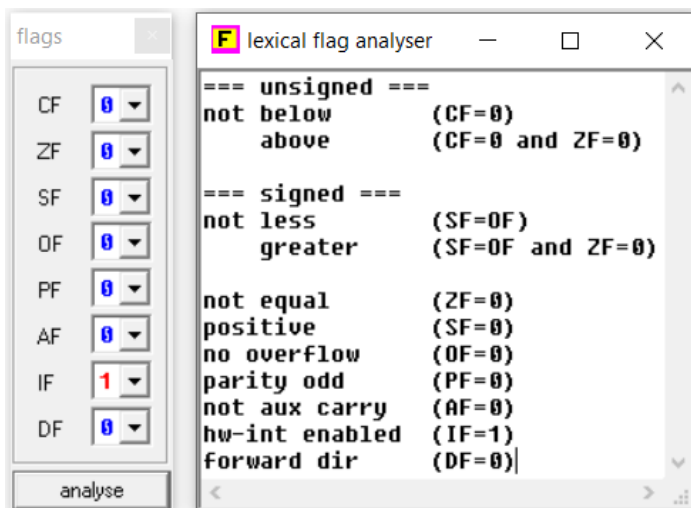
- **Koliko bajtova zauzima svaka napisana instrukcija?**

Promenljive veličine dw zauzimaju po 2B (db bi zauzelo jedan bajt, dd četiri itd), *MOV ax, br1*, *MOV rez, ax*, *MOV ax, rez*, *CMP ah, 0*, *CMP ah, 1*, *mov si, 1* po tri, *mov si, 1* četiri, i ostale instrukcije po dva.

- **Kakav je status flegova nakon što se rezultat prebaci u AX registar? Zašto?**

Status flegova nakon prebacivanja rezultata u AX registar prikazan je fotografijom u okviru ovog dokumenta. Statusni flegovi nakon izvršenja neke aritmetičke instrukcije dobijaju vrednost (0 ili 1) koja oslikava osobine rezultata te operacije. Njihova vrednost može se čitati ali ne i menjati. S druge strane, kontrolni flegovi kontrolišu način izvršavanja pojedinih instrukcija i menjaju svoju vrednost pre izvršavanja istih. Stanje interrupt enable flega je postavljeno na vrednost 1. Ovo je kontrolni fleg koji omogućava spoljašnje prekide (da je vrednost bila jednaka nuli ovakvi prekidi bili bi zanemareni). *Ostala stanja su nepromenjena (0) u ovom slučaju, ali npr. moglo je doći do promene flega parnosti (PF, ako bi krajnji rezultat imao paran broj bitova koji su jedan).*

**Stanje flegova:**



**Mesto u memoriji:**

The screenshot displays four memory windows in a debugger, each showing assembly code. The windows are titled 0711:0005, 0711:0005, 0711:0016, and 0711:0016. The code includes instructions like MOV AX, [00000h], ADD AX, [00002h], MOV [00004h], AX, MOV AX, [00004h], MOV BL, 02h, DIV BL, CMP AH, 00h, JZ 020h, CMP AH, 01h, JMP 023h, MOV SI, 00001h, and various NOP instructions. The code is color-coded by instruction type.

**Promenljive (vrednosti izmenjene u odnosu na početni kod):**

The screenshot displays a memory window and a variables window in a debugger. The memory window is titled 0710:0000 and shows assembly code. The variables window is titled 0710:0004 and shows the values of variables BR1 (42), BR2 (23), and REZ (0000h). The variables window also includes settings for size (word) and elements (1).

**Mašinski kod:**

```
[ 1]      :      data_seg SEGMENT
[ 2] 0000: 05 00      br1 dw 5
[ 3] 0002: 02 00      br2 dw 2
[ 4] 0004: 00 00      rez dw ?
[ 5]      :      data_seg ENDS
[ 6]      :
[ 7]      :      code_seg SEGMENT
[ 8]      :      ASSUME cs:code_seg, ds:data_seg
[ 9]      :
[10] 0010: BA 00 00      start: MOV dx, offset data_seg
[11] 0013: 8E DA      MOV ds, dx
[12]      :
[13] 0015: A1 00 00      MOV ax, br1
[14] 0018: 03 06 02 00    ADD ax, br2
[15] 001C: A3 04 00      MOV rez, ax
[16]      :
[17] 001F: A1 04 00      MOV ax, rez
[18] 0022: B3 02      MOV bl, 2
[19] 0024: F6 F3      DIV bl
[20]      :
[21] 0026: 80 FC 00      CMP ah, 0
[22] 0029: 74 05      je paran
[23] 002B: 80 FC 01      CMP ah, 1
[24] 002E: EB 03      jmp kraj
[25]      :
[26] 0030:      paran:
[27] 0030: BE 01 00      mov si, 1
[28]      :
[29] 0033: EB FE      kraj: jmp kraj
[30]      :      end start
[31]      :
[32]      :      code_seg ENDS
[33]      :      END
```