

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
```

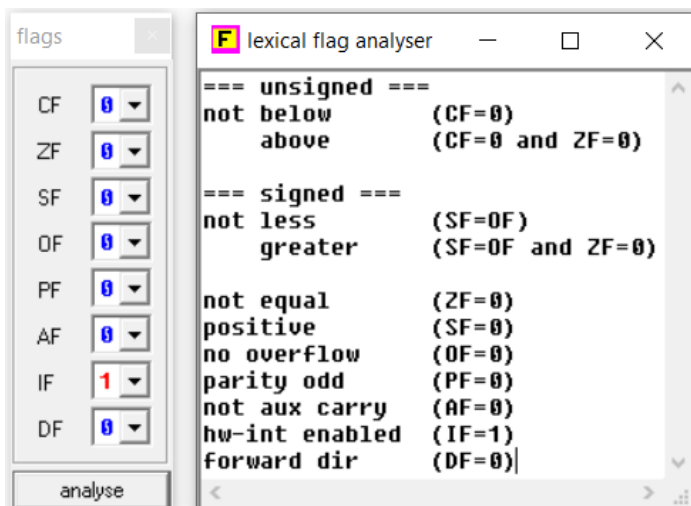
- **Koliko bajtova zauzima svaka napisana instrukcija?**

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:

| 0711:0005 | 0711:0005 | 0711:0016 | 0711:0016 |
|--------------------|------------------|--------------------|----------------|
| 07115: A1 161 í | MOV AX, [00000h] | 07126: 80 128 Ç | CMP AH, 00h |
| 07116: 00 000 NULL | ADD AX, [00002h] | 07127: FC 252 R | JZ 020h |
| 07117: 00 000 NULL | MOV [00004h], AX | 07128: 00 000 NULL | CMP AH, 01h |
| 07118: 03 003 ♥ | MOV AX, [00004h] | 07129: 74 116 t | JMP 023h |
| 07119: 06 006 ♠ | MOV BL, 02h | 0712A: 05 005 ♣ | MOV SI, 00001h |
| 0711A: 02 002 ☉ | DIU BL | 0712B: 80 128 Ç | JMP 023h |
| 0711B: 00 000 NULL | CMP AH, 00h | 0712C: FC 252 R | NOP |
| 0711C: A3 163 ú | JZ 020h | 0712D: 01 001 ☉ | NOP |
| 0711D: 04 004 ♦ | CMP AH, 01h | 0712E: EB 235 Ú | NOP |
| 0711E: 00 000 NULL | JMP 023h | 0712F: 03 003 ♥ | NOP |
| 0711F: A1 161 í | MOV SI, 00001h | 07130: BE 190 ž | NOP |
| 07120: 04 004 ♦ | JMP 023h | 07131: 01 001 ☉ | NOP |
| 07121: 00 000 NULL | NOP | 07132: 00 000 NULL | NOP |
| 07122: B3 179 | NOP | 07133: EB 235 U | NOP |
| 07123: 02 002 ☉ | NOP | 07134: FE 254 ■ | NOP |
| 07124: F6 246 ÷ | NOP | 07135: 90 144 É | NOP |
| 07125: F3 243 ˇ | NOP | 07136: 90 144 É | NOP |
| 07126: 80 128 Ç | NOP | 07137: 90 144 É | NOP |
| 07127: FC 252 R | NOP | 07138: 90 144 É | NOP |
| 07128: 00 000 NULL | NOP | 07139: 90 144 É | NOP |
| 07129: 74 116 t | ... | 0713A: 90 144 É | ... |

Promenljive (vrednosti izmenjene u odnosu na početni kod):

The screenshot shows the WinDbg interface with the 'variables' window open. The 'variables' window displays the current values of the BR1, BR2, and REZ registers. BR1 is 42, BR2 is 23, and REZ is 0000h. The 'size' is set to 'word' and 'elements' is 1. The 'show as' is set to 'signed'.

| Register | Value |
|----------|-------|
| BR1 | 42 |
| BR2 | 23 |
| REZ | 0000h |

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
```