Kontrola toka programa - procedure i prekidi

Procedure

Procedure

- Procedure su segmenti koda koji se mogu pozivati proizvoljan broj puta iz programa kako bi se izvršili pojedini zadaci.
- Procedure čine program znatno strukturalnijim i jednostavnijim za razumevanje.
- Pored toga izbegava se ponavljanje pojedinih identičnih segmenata koda u programu.
- Samim tim i otklanjanje eventualnih grešaka je lakše.

Procedure

Sintaksa procedura je sledeća:

```
nazivprocedure PROC

    ; ovde se nalaze
    ; instrukcije procedure

RET
nazivprocedure ENDP
```

Procedure

PROC i ENDP su direktive i ne prevode se u mašinski kod već služe kompajleru da zapamti adresu procedure.

Procedure

Procedura se poziva CALL instrukcijom iza koje se navodi naziv procedure.
CALL nazivprocedure

Procedure

Primer: program sa procedurom koja menja vrednosti u registima AX i BX

```
MOV AX, 100
MOV BX, 200
CALL zameni
RET
zameni PROC
MOV CX, AX
MOV AX, BX
MOV BX, CX
RET
zameni ENDP
```

Procedure

Prisetimo se 4 instrukcije koje omogućavaju da se sadržaj registara ili registra stanja stavi na stek ili skine sa steka. To su:

- PUSHA koja sadržaje registara AX, CX, DX, BX, SP, BP, SI, DI stavlja na stek,
- POPA koja je inverzna instrukciji PUSHA i koja sa steka puni registre DI, SI, BP, SP, BX, DX, CX, AX,
- PUSHF koja statusni registar stavlja na stek i
- POPF koja sa steka postavlja statusne bitove tj. puni statusni registar.

Procedure

- Da bi smo sačuvali stanje svih registara ili registra stanja možemo kao prvu instrukciju u proceduri pozvati PUSHA ili PUSHF i time sacuvati vrednosti registara na stek, a prilikom izlaska iz procedure kao poslednju instrukciju navodimo POPA ili POPF.
- Time ćemo stanje registara vratiti u prvobitno stanje, pre poziva procedure.

Prekidi

- Prekid je softverska tehnika koja, kao što joj samo ime kaže, omogućava prekid normalnog izvršenja programa i u zavisnosti od vrste prekida prelazi na izvršavanje odgovarajuće procedure.
- Po završetku izvršavanja te procedure program dalje normalno nastavlja sa radom.

Prekidi

Prekidi mogu biti **hardverski** i **softverski**.

Prekidi

- Hardverske prekide može da uzrokuje neki periferni uredjaj, zahtevajući da mu se posveti odgovarajuće procesorsko vreme i neki drugi resursi.
- Primer, pritisak na dugme tastature izaziva generisanje prekida koji će uzrokovati obradu pritisnutog tastera.
- U ovom primeru može se desiti da se npr. pozove grupa instrukcija koja će omogućiti ispisivanje slova na ekranu i dodavanje u fajl.

Prekidi

- Softverske prekide generiše sam programer pozivom odgovarajuće instrukcije asemblera.
- Za svaki prekid izvršava se unapred definisani kod.
- Zato je potrebno poznavati listu prekida kao i efekte pozivanja svakog od njih.

Prekidi

U asembleru za mikroprocesor 8086 softverski prekidi se pozivaju instrukcijom INT iza koje sledi osmobitni broj koji označava vrstu prekida.

INT broj

Prekidi

INT 21h

AH = 01h

očitava taster sa standardnog ulaza i smešta ASCII kod u registar AL

Prekidi

INT 21h

AH = 02h

upisuje karakter iz registra DL na standardni izlaz; nakon ispisa je AL=DL

Prekidi

INT 21h

AH=2Ah

vraća sistemski datum

CX = godina, DH = mesec,
DL = dan, AL = dan u sedmici

Prekidi

INT 21h

AH=2Ch

CH = sati, CL = minute
DH = sekunde

Prekidi

INT 21h

AH= 47h

vraća trenutni folder u kome se nalazimo
i smešta u memoriju DS:SI

Prekidi

INT 21h

AH=4Ch

vraća kontrolu operativnom sistemu
tj. završava program

Prekidi

Primer: segment programa koji očitava pritisak tastera tastature

```
mov ah, 01h
int 21h
```

Prekidi

Primer: segment programa koji na displej ispisuje slovo a

```
mov ah, 02h
mov dl, 'a'
int 21h
```

Prekidi

Primer: segment programa koji u odgovarajuće registre smešta trenutni datum

```
mov ah, 2Ah
int 21h
```

Prekidi

Primer: segment programa koji u odgovarajuće registre smešta trenutno vreme

```
mov ah, 2Ch
int 21h
```

Prekidi

Primer: segment programa koji u memorijsku lokaciju DS:SI smešta naziv foldera u kom se trenutno nalazimo

```
mov ah, 47h
int 21h
```

Prekidi

Primer: segment programa koji vraća kontrolu operativnom sistemu i završava izvršavanje programa

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

ZADATAK 1: Napisati program koji računa sumu kvadrata od 1 do n, gde je n broj iz opsega [0,255] koji se nalazi u registru BX. Za računanje kvadrata broja koristiti proceduru kvadrat.
REŠENJE: U registru CX ćemo formirati sumu kvadrata brojeva, a registar AX ćemo koristiti za računanje kvadrata broja. Kako je n broj iz opsega [0,255] možemo koristiti 8-bitno množenje pomoću MUL instrukcije. Pri tome rezultat ne može imati više od 16 bitova tj. biće čuvan u AX registru (pogledati 8-bitno množenje MUL instrukcije).

Prekidi

```
MOV BX, 10
MOV CX, 0

saberi:
call kvadrat
ADD CX, AX
DEC BX
JNZ saberi

RET

kvadrat PROC
MOV AX, BX
MUL AL
RET
kvadrat ENDP
```

ZADATAK 2: Napisati program koji računa zbir 5! + 6!. Realizovati proceduru faktorijel.

```
MOV CX, 0

MOV BX, 5
CALL faktorijel
ADD CX, AX

MOV BX, 6
CALL faktorijel
ADD CX, AX

RET

faktorijel PROC
        MOV AX, 1
        pomnozi:
        MUL BX
        DEC BX
        JNZ pomnozi
RET
faktorijel ENDP
```

ZADATAK 3: Napisati program koji vrši množenje 16-bitnih brojeva sa memorijske lokacije 2000:0001/0002 i 2000:0003/0004 i 32-bitni rezultat množenja iz registara AX i DX smešta u 4 lokacije 2000:0005/0008.
Množenje realizovati u proceduri.
Izvršiti još jedno množenje 16-bitnih brojeva na isti način sa naredne 4 lokacije i rezultat upisati u memoriju.
Prilikom pozivanja procedure čuvati stanje svih registara i statusnog registra

REŠENJE:

```
MOV CX,2000h
MOV DS,CX
MOV AX,1          ; postavljamo inicijalne vrednosti
MOV BX,2          ; pojedinih registara kako bi smo
MOV CX,3          ; mogli uociti da poziv procedure
MOV DX,4          ; cuva vrednosti u njima
MOV SI, 1         ; postavimo pomeraj za prvo mnozenje
CALL pomnozi      ; prvo mnozenje
MOV SI, 9         ; postavimo pomeraj za drugo
                  ;mnozenje
CALL pomnozi      ; drugo mnozenje
RET
```

```
pomnozi PROC
PUSHA           ; vrednosti svih registara cuvamo na steku
PUSHF           ; vrednost statusnog registra cuvamo na steku
MOV AX, [SI]        ; prvi faktor za mnozenje u AX
MOV BX, [SI+2]    ; drugi faktor za mnozenje u BX
MUL BX              ; pomnozi AX sa BX
MOV [SI+4], AX     ; upisi rezultat iz AX u memoriju
MOV [SI+4], DX     ; upisi rezultat iz DX u memoriju
POPF               ; vrati prvobitno stanje statusnog registra
POPA               ; vrati prvobitno stanje svih registara
RET
pomnozi ENDP
```

- Posebno treba obratiti pažnju na redosled instrukcija POPF i POPA.
- Da su ove dve instrukcije pozvane drugim redosledom registri ne bi bili ispravno vraćeni u prvobitno stanje.
- Takodje, treba primetiti da smo proceduru „pomnozi“ pozivali 2 puta.

- Iako procedure u asemblerskom jeziku nemaju mogućnost prosledjivanja parametara, u našem programu smo za ovu namenu koristili registar SI.
- Analizom procedure „pomnozi“ vidi se da su korišceni registri AX, BX i DX kao interni registri za aritmetičke operacije, dok se registar SI postavlja izvan procedure pa ima ulogu parametra za proceduru.

ZADATAK 4: Napisati program koji u rastućem redosledu sortira 10 brojeva upisanih na memorijske lokacije 2000:0001 – 2000:000A. Koristiti metodu "bubble sort", a zamenu brojeva realizovati procedurom.

REŠENJE:

MOV CX, 2000h
MOV DS, CX
MOV CX, 1
spoljasnjaa:
CMP CX, 10
JE kraj

```
MOV BX, 1
unutrasnja:
CMP BX, 10
JE krajunutrasnje
MOV AL, [BX]
MOV DL, [BX+1]
CMP AL, DL
JNA dalje
CALL zameni
dalje:
```

```
INC BX
JMP unutrasnja
krajunutrasnje:
```

```
INC CX
JMP spoljasnja
kraj:
RET
zameni PROC
MOV [BX], DL
MOV [BX+1], AL
RET
zameni ENDP
```