UNIVERZITET U BEOGRADU

ELEKTROTEHNIČKI FAKULTET

Katedra za elektroniku

Predmet: Računarska elektronika

Projekat: 18 Moj Broj

Projekat radili:

Ime	Prezime	broj indeksa
Uroš	Cvjetinović	2016/0093
Nikola	Jugović	2016/0408

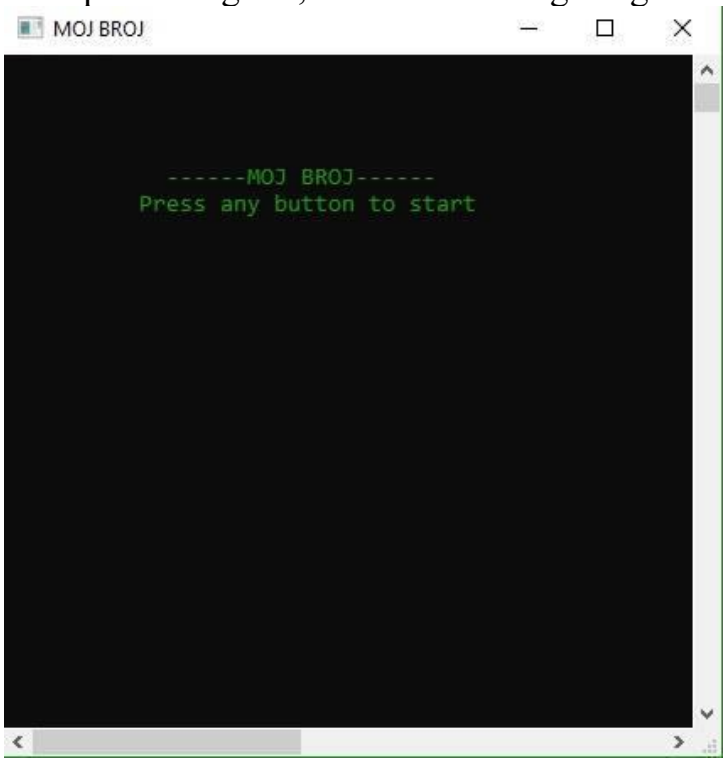
Teskt zadatka:

Potrebno je realizovati igru “Moj broj” iz kviza “Slagalica”. Kada se započne igra, generiše se nasumičan broj u opsegu 1-999, kao i četiri broja u opsegu 1-9, jedan broj iz skupa {10, 15, 20} i jedan broj iz skupa {25, 50, 75, 100}. Potrebno je dobiti prvi generisani broj koristeći se isključivo ostalim generisanim brojevima (svaki broj se sme koristi samo jednom), osnovnim računskim operacijama (sabiranje, oduzimanje, množenje i deljenje) i zagradama. Potrebno je sve generisane brojeve (osim prvog) ispisati u zasebnim poljima i omogućiti kretanje kursora po njima i njihov izbor pritiskom na odgovarajuće tastere. Nakon što se broj izabere, onemogućiti novi izbor istog broja. Obezbediti zasebna polja i za svaku od operacija i zagrade. Pritiskom na taster ,,space”, bira se broj/operacija na kojoj sekursor nalazi. Jasno označiti polje na kojem se nalazi kursor. Potrebno je i ispisvati matematičku formulu koju igrač koristi za dobijanje traženog broja. Nakon što igrač završi sa ispisivanjem formule, pritiskom na taster ,,enter” rezultat se izračunava i igraču se šalje poruka ukoliko je uspeo da pogodi tačan broj ili koliko mu fali do tačnog broja.

Realizacija aplikacije:

Program je napisan koristeći Irvine32.inc biblioteku i Irvinove preporuke za modele koji se koriste i veličinu steka za rad u VS2017.

Pri pokretanju aplikacije otvara se „Menu“ prozor, koji pritiskom na bilo koje dugme osim ESC pokreće igricu, dok taster ESC gasi igricu.



Do prozora u kom se igra ,,Moj broj“ se dolazi iz prozora ,,Menu“.

Gornji broj je traženi broj, dok brojevi ispod su brojevi koji su na raspolaganju. U toku igre ako se iskoristi neki broj on više nije dostupan i menja se boja teksta kojom je ispisan iz zelene (na raspolaganju) u crvenu (iskorišćen broj). Belo polje ispod nasumično generisanih brojeva je traka u koju korisnik upisuje izraz. Ispod polja za izraz nalaze se operacije, zagrade, koje u zavisnosti od toga da li trenutno mogu da se iskoriste u izrazu menjaju boju: crna boja (na raspolaganju), siva boja (zabranjen unos). Ovim ograničavanjem korisnika se izbegava neispravan unos formule. Korisnik upisuje formulu korišćenjem tastera ,,space“ i strelica. Strelicama se kreće po poljima koja može uneti u formulu pritiskom na taster ,,space“. Dodata su dva polja, koja su van opsega zadatka, polje ,,DEL“, koje briše prethodni uneti simbol/broj u formulu, i polje ,,CE“ koja pokreće igru ispočetka sa istim generisanim brojevima. Ovim je olakšan unos korisniku.

Pritiskom na taster ,,enter“ se objavljuje rezultat unetog izraza, i apsolutna vrednost razlike rezultata i traženog broja.

Realizacija koda:

Kod je realizovan modularno. Iz više nezavisnih procedura je realizovan interfejs korisnika i igre i izračunavanje krajnjeg rezultata. Pomoću globalnih promenljivi Selected, Used, Changed, CursorPos,Numbers, Formula, FormulaSize, NeedToClose

Inicijalno generisanje nasumičnih brojeva je realizovano procedurama u **_InitLogic.asm** (*MyRandom*, *InitLogic*, *RestartVar*).

- MyRandom(dword) – prosleđuje joj se 32bitni broj koji određuje maksimalnu vrednost nasumičnog broja. Povratna vrednost je broj koji može da uzme vrednosti između 1 i prosleđenog broja
- InitLogic(PTR WORD) – prosleđuje se pokazivač na početak niza u koji se smeštaju nasumično generisani brojevi koji odgovaraju zadatim opsezima

Pomoćne procedure koje su omogućile efikasnije iscrtavanje i pregledniji kod **_Verifier.asm**(*DecToBin, CheckIfUsed, CheckIfSelected, CheckIfChanged*)

- DecToBin(BYTE) – prosleđuje se decimalni broj koji se konvertuje u broj 2^(N-1), gde je N prosleđeni broj. Pomoćna funkcija koja olakšava proveru zabranjenosti nekog polja
- CheckIfUsed(BYTE) – proverava da li je polje koje se razmatra (prosleđeni broj) već iskorišćen
- CheckIfSelected(BYTE) – proverava da li je selektovano polje koje je prosleđeno
- CheckIfChanged(WORD) – proverava da li treba polje koje je prosleđeno da se iscrtava ponovo

Procedure pomoću kojih se iscrtavaju polja pri igranju **_Output.asm**(*CrateBlock, CrateDecLabeledBlock, CrateCharLabeledBlock, CrateStrLabeledBlock*)

- CrateBlock(BYTE, BYTE, BYTE, BYTE, BYTE) – procedura koja iscrtava obojeni pravougaonik sa parametrima redom prosleđenim proceduri: gornja leva ivica (x,y), dužina stranica (x,y) i boja (1-crvena, 2-zelena, 3 – siva, 4 – crna, 5 – plava, 6 – bela). Pravougaonik se iscrtava tako što se prazni znakovi ' ' sa određenom obojenom pozadinom ispisuju od početne ivice udesno do određene dužine, nakon čega se vrati do leve stranice bez ispisivanja znakova, spusti se za jedan simbol niže i iscrtava ponovo nadesno, i tako sve dok ne stigne do određene dužine vertikalne ivice.
- CrateDecLabeledBlock (BYTE, BYTE, BYTE, BYTE, BYTE, BYTE, WORD, BYTE, BYTE) – Uže povezana procedure za funkcionalnost igre.

Kreira se pravougaonik sa brojem unutar njega pozivom CrateBlock koji poziciju i veličinu dobija direktno od prosleđenih argumenata, dok boju bloka i broja koji će se upisati u njega određuju poslednja dva argumenta. Prva četiri argumenta su pozicija i veličina potrebna za iscrtavanje bloka, druga dva određuju poziciju broja. Sedmi argument je broj koji se upisuje u pravougaonik. Poslednja dva argumenta predstavljaju redom, da li je polje selektovano (prosleđena 1) i da li je polje iskorišćeno. Zavisno od prosleđene vrednosti, ako je polje selektovano polje je obojeno plavom bojom, ako nije onda je sivom. Ako je polje iskorišćeno (prosleđena 1) broj koji se upisuje se upisuje crvenom bojom, dok ako nije onda se upisuje zalenom.

- CrateCharLabeledBlock (BYTE, BYTE, BYTE, BYTE, BYTE, BYTE, BYTE, BYTE, BYTE) – Procedura je slična proceduri CrateDecBlockLabeled u postupku iscrtavanja i prosleđivanja argumenata. Razlikuje je što se upisuje karakter (namenjeno za operacije „+,-,*,/,(,)“) a ne broj unutar pravougaonika i što se boja da je iskorišćeno polje siva, dok ako je raspoloživo onda je crno
- CrateStrLabeledBlock (BYTE, BYTE, BYTE, BYTE, BYTE, BYTE, PTR BYTE, BYTE, BYTE) – Procedura je slična proceduri CrateCharLabeledBlock u postupku iscrtavanja i prosleđivanja argumenata. Razlikuje se u tome što ne ispisuje karakter već prosleđen string. (Namenjeno za „DEL“ i „CE“)
- Refresh (PTR WORD) – Prosleđuje joj se pokazivač na početak niza u kojem su smešteni generisani brojevi. Glavna procedura za iscrtavanje layouta igre, poziva se kada treba da se osveži interfejs, što se dešava svakim pritiskom tastera. Procedura proverava da li polje treba da se iscrta ponovo (da li je promenjeno), što se dešava ako je bilo selektovano pa se kursor pomerio sa njega ili u slučaju da se tek selektovao, ili ako se izabrao pritiskom na taster „space“ ili ako je potrebno zabraniti operaciju nakon što se druga upravo izabrala. Ako polje treba da se promeni, proveri se na koji način treba i pozove se njoj odgovarajuća procedura za iscrtavanje. U ovoj proceduri se i ispisuje izraz koji korisnik sastavlja, tako što globalnom CursorPos se prati položaj u izrazu, i zavisno od unetog simbola adekvatno upisuje ili briše unutar tog polja, tako što se dopisuje u izraz i poveća CursorPos ili preboji belom bojom i smanji CursorPos. Nakon brisanja se adekvatno nameštaju zabrane i iskorišćenosti svih polja.

Procedure koje su najuže povezane sa igrom Moj broj

_GameMechanic.asm(*Menu, StartGame, PlayGame, Results*)

- Menu – Procedura setuje veličinu i naslov prozora i predstavlja početnu stranu za korisnika. Pritiskom na dugme koje nije ESC odvodi korsnika na stranu pri kojoj može započeti igru.
- StartGame – Pozivaju se *RestartVar*, *InitLogic, Refresh* kojim se podešavaju promenljive za početak igre.
- PlayGame – U ovoj proceduri se prati unos korisnika, i u zavisnosti od prethodnih i trenutnog stanja adekvatno namešta da li su polja zabranjena, iskorišćena i promenjena, nakon čega poziva proceduru *Refresh* koja iscrtava polja koja su promenjena ponovo. Pamti trenutno stanje izraza u globalnu promenljivu Formula (niz) u kojoj su smešteni redni brojevi polja koja su aktivirana onim redom kojim su se aktivirali. U slučaju da korisnik pokušava zabranjeno polje da izabere, program će ignorisati taj pokušaj i neće se iscrtavati iznova. Kraj unosa je kad se pritisne taster „enter“
- Results – Izračunava izraz koji je korisnik sastavio, koristi prostiji algoritam za izračunavanje za bolje performanse zbog kojeg program zahteva korišćenje zagrada. Izraz se nalazi u globalnoj promenljivoj Formula, i u njoj su smešteni redni brojevi polja, čime se lakše određuje prioritetnost pri računanju, (broj 1 je namenjen za vrednost koja je izračunata i na steku se nalazi). Zbog svega mogućih sedam brojeva za kombinovanje koristi se sledeći algoritam za izračunavanje izraza: pronađe se prva najprijoritetnija zagrada na koju se naiđe (krećući se ->); unutar te zagrade traži prvo množenje/deljenje na koje se naiđe (krećući se ->); izračuna se vrednost i stavi na stek (ako operand bio na steku, skini sa steka i računaj sa njim); ponovi se pretraga i računanje množenja/deljenja; ako nema više množenja/deljenja ponovi se postupak za sabiranje/oduzimanje isto kao i za množenje/deljenje samo što se sad kreće (<-); izračuna se vrednost i stavi na stek (ako operand bio na steku, skini sa steka i računaj sa njim); ponovi se pretraga i računanje sabiranje/oduzimanje svaki put kad se izračuna podizraz prepišu se 1 na mesta operanada I operacije; prepiši 1 preko zagrada čiji se izraz izračunao; ponovi postupak za sledeće zagrade; ako ne postoje više zagrada, prođe kroz računanje još jednom kao da su zagrade na početku I na kraju izraza. Izračuna vrednost I ispiše na ekran korisniku, I razliku između traženog I izračunatog broja.Ukoliko je izračunat broj negativan,ispisuje se 0
- RestartVar – Resetuje globalne promenljive za početak igre ponovo

KOD:

```
comment &
/************** MOJ BROJ ****************************************
Aplikacija je napisana koristeci Irvine32.inc biblioteku i Irvine-ove
preporuke za modele koji se koriste i velicinu steka za rad
u VS2017.
Po pokretanju igre korsnik se nalazi u Menu-u odakle moze zapoceti
igru. U bilo kom trenutku tokom igranja moguce je napustiti igricu
pritiskom na ESC taster, cime se korisnik vraca u Menu
odakle se moze ponovnim pritiskom na ESC tester izaci iz aplikacije.
Kod je realizovan modularno, i moze se podeliti u manje nezavisne
procedure, i par procedura koje su uze povezane za samo logiku igranja.
Glavni program je realizovan tako sto je podeljen na faze igranja
pomocu procedura sa nazivom koji to i implicira.
Uros Cvjetinovic 2016/0093
Nikola Jugovic    2016/0408
*****************************************************************/ &
.386
.model flat, stdcall
.stack 4096
ExitProcess proto, dwExitCode:dword

INCLUDE Irvine32.inc

;//----------Konstante
.const
;//Broj random brojeva koji su potrebni za igru MOJ BROJ
ArrayLength = 7

;//Definisanje velicine prozora: leva,desna,gornja,donja ivica
xmin = 0
xmax = 50
ymin = 0
ymax = 24
;//Oznake za levo, desno, gore, dole, ESC, ASCII
LEFT_KEY  = 025h
UP_KEY    = 026h
RIGHT_KEY = 027h
DOWN_KEY  = 028h

ENTER_KEY = 00Dh
SPACE_KEY = 020h
ESC_KEY   = 01Bh

.data
        Selected BYTE 5                                         ;//Pokazuje da li je polje selektovano
        Used     WORD  1110111110000000b            ;//Pokazuje da li je polje sme da se koristi
```

```
	Changed WORD 1111111111111111b		;//Pokazuje da li je polje promenjeno pri refresovanju displeja
	CursorPos WORD 0508h				;//Polozaj kursora unutar trake za ispis izraza
	FormulaSize WORD 0				;//velicina izraza koju je korsnik uneo
	NeedToClose BYTE 0				;//broj zagrada potrebnih da se zatvore

	Array1 WORD 10, 15, 20
	Array2 WORD 25, 50, 75, 100
	;// Tekst potreban za polja koja obelezavaju brisanje iz izraza
	txtDelete BYTE "DEL", 0
	txtCE BYTE "CE", 0
	;// Tekst potreban za interfejs
	txtResults1 BYTE " VAS RAZULTAT JE :	", 0
	txtResults2 BYTE " RAZLIKUJE SE ZA :	", 0
	txtStart1	BYTE " ------MOJ BROJ------ ", 0
	txtStart2	BYTE " Press any button to start ", 0

	windowRect SMALL_RECT <xmin, ymin, xmax, ymax>		;//Velicina prozora

;//NAZIV PROGRAMA
	winTitle byte "MOJ BROJ", 0
;//Informacije o polozaju kursora
	cursorInfo CONSOLE_CURSOR_INFO <>

.data?
	openPar		DWORD ?				;//Pamtimo gde se nalazi otvorena zagrada, pri racunanju
	closedPar	DWORD ?				;//Pamtimo gde se nalazi zatvorena zagrada, pri racunanju
	Numbers WORD  7 DUP(? )			;//Niz u koji upisujemo generisane brojeve
	Formula BYTE 26 DUP (?)			;//Niz u koji upisujemo redne brojeve polja koje se aktiviraju
	stdOutHandle handle ?		;//Handle za ispis podataka
	stdInHandle handle ?		;//Handle za upis podataka

.code

;//***********Procedure koje pripadaju _InitLogic.asm***********
;//------MyRandom vraca eax = nasumican broj u opsegu( 1 - myRange)
MyRandom PROC,
	myRange: DWORD
		call Randomize
		mov  eax, myRange
		call RandomRange
		inc  eax
		ret
MyRandom ENDP

;//--------Inicjalizuje nasumicne brojeve potrebne za igru
InitLogic PROC,
```

```
Arr : PTR WORD
	push edx
	push edi
	push esi
	push ecx

	mov edi, Arr
	mov ecx, 4

	;//Trazeni broj
	INVOKE MyRandom, 999
		mov WORD PTR[edi], ax
		inc edi
		inc edi

	;// 4 x (1-9)
	L1 :
		mov eax, 1
		call Delay
		INVOKE MyRandom, 9
			mov WORD PTR[edi], ax
			inc edi
			inc edi
	loop L1

	;// Broj iz opsega {10,15,20}
	INVOKE MyRandom, 3
		mov ecx, eax
		dec ecx
		mov esi, OFFSET Array1
		imul ecx, 2
		add esi, ecx
		mov dx, WORD PTR[esi]
		mov WORD PTR[edi], dx
		inc edi
		inc edi

	;// Broj iz opsega {25,50,75,100}
	INVOKE MyRandom, 4
		mov ecx, eax
		dec ecx
		mov esi, OFFSET Array2
		imul ecx, 2
		add esi, ecx
		mov dx, WORD PTR[esi]
		mov WORD PTR[edi], dx
		inc edi
		inc edi

		pop ecx
		pop esi
```

```
		pop edi
		pop edx
		ret
InitLogic ENDP

;//*************Procedure koje pripadaju _Verifier.asm*************
;//------------ Pretvara decimalni broj N u 2^(N-1)
DecToBin PROC,
	Num : BYTE ;// N
		;//Procedura izracunava 2^(N-1)

	push ecx

	mov eax, 1		;//smesti jedinicu i siftuj je ulevo
	movsx ecx, Num
	dec ecx
	jz _end_l
	_loop:
		shl eax, 1
	loop _loop
_end_l:
	pop ecx
	ret
DecToBin ENDP

;//-------Proverava da li je broj vec u upotrebi&
CheckIfUsed PROC,
		Masked: BYTE ;//Polje koje proveravamo da li je iskorisceno

		push ebx
		INVOKE DecToBin, Masked
		mov bx, ax
		mov ax, Used
		and ax, bx
		jnz isused
		mov eax, 0
		jmp end_l
	isused:
		mov eax, 1
end_l:
		pop ebx
		ret
CheckIfUsed ENDP

;//------Proverava da li je selektovano to polje
CheckIfSelected PROC,
		Masked : BYTE;//Polje koje proveravamo da li je selektovano

		mov al, Selected
		sub al, Masked
		jnz isselected
```

```
		mov eax, 1
		jmp end_l
	isselected :
		mov eax, 0
	end_l :
		ret
CheckIfSelected ENDP

;//------Proveravamo da li je treba da se crta polje iznova (da li je promenjeno)
CheckIfChanged PROC,
		Masked: WORD;//Polje koje proveravamo da li je promenjeno

		push ebx
		mov bx, Masked
		INVOKE DecToBin, bh
		mov bh, al
		INVOKE DecToBin, bl
		mov bl, al
		mov ax, Changed
		and ax, bx
		jnz ischanged
		mov eax, 0
		jmp end_l
	ischanged:
		mov eax, 1
end_l:
		pop ebx
		ret
CheckIfChanged ENDP

;//***********Procedure koje pripadaju _Output.asm ******************
;//-------Kreira obojen pravougaonik
CrateBlock PROC,
			xpos:  BYTE,	;// x vrednost gornjeg levog ugla
			ypos:  BYTE,	;// y vrednost gornjeg levog ugla
			xsize: BYTE,	;// broj karaktera koje se boje udesno od xpos
			ysize: BYTE,	;// broj karaktera koje se boje nanize od ypos
			color: BYTE		;// boja kojom se boji pravougaonik

		push edx
		push ebx
		push ecx

		mov al, 1
		cmp al, color
			je crvena
		mov al, 2
		cmp al, color
			je zelena
		mov al, 3
		cmp al, color
```

```
                je siva
        mov al, 4
        cmp al, color
                je crna
        mov al, 5
        cmp al, color
                je plava
        mov al, 6
        cmp al, color
                je bela


                ;//Bira se boja koja je prosledjena
    crvena:
        mov ax, red + (lightGray * 16)
        jmp crtaj
    zelena:
        mov ax, green + (lightGray * 16)
        jmp crtaj
    siva:
        mov ax, lightGray + (lightGray * 16)
        jmp crtaj
    crna:
        mov ax, black + (lightGray * 16)
        jmp crtaj
    plava:
        mov ax, blue + (lightGray * 16)
        jmp crtaj
    bela:
        mov ax, white + (lightGray * 16)
        jmp crtaj


    crtaj:
        call SetTextColor           ;//Podesavanje boje teksta
        mov  dl, xpos               ;//x pozicija kursora
        mov  dh, ypos               ;//y pozicija kursora
        mov     bl, ysize               ;//y pozicija kvadrata koja govori o tome
da je gotovo crtanje
        add bl, dh
        mov bh, 061h
        movsx ecx, xsize            ;//broj prolazaka kroz petlju
        mov al, 0DBh
        jmp xinc                        ;//skok na labelu za crtanje
pravougaonika

    xinc:;//prvo ispisujemo udesno
        call gotoxy
        call writechar
        inc dl
    loop xinc
```

```
		movsx ecx, xsize

	xdec: ;//vratimo se na skroz levi
		dec dl
	loop xdec

	yinc:;//spustimo se jedno nanize
		movsx ecx, xsize
			inc dh
			cmp bl, dh
		jne xinc;//ako nije stiglo do najnizeg ponovi proceduru

			mov ax, white
			call SetTextColor

			pop ecx
			pop ebx
			pop edx
			ret
CrateBlock ENDP

;//--------Kreira blok sa broj unutra
CrateDecLabeledBlock PROC,
			xposb:  BYTE,		;// x vrednost gornjeg levog ugla
			yposb:  BYTE,		;// y vrednost gornjeg levog ugla
			xsize: BYTE,		;// broj karaktera koje se boje udesno od xpos
			ysize: BYTE,		;// broj karaktera koje se boje nanize od ypos
			xpost : BYTE,		;// x polozaj teksta
			ypost : BYTE,		;// y polozaj teksta
			numb : WORD,		;//Broj koji se upisuje
			isSelected : BYTE,	;//Da li je selektovano polje
			inUse : BYTE		;//Da li se koristi vec simbol iz polja

			push edx

			;//Crta polje koje ako je selektovano je pravougaonik plave boje a ako
			;//nije onda je sive boje. Ako je iskoriscen broj onda je crvene boje
			;//obojen a ako nije onda zelenom. Uvedena je i crna boja kojom se boji
			;//trazeni broj (crno se boji ako je isUse = 2)

			mov al, isSelected
			dec al
			jz sel

			INVOKE CrateBlock, xposb, yposb, xsize, ysize, 3
				mov al, inUse
				dec al
				jnz _greenorblack1

				add eax, red + (LightGray * 16)
				jmp next
```

```
                _greenorblack1:
                        dec al
                        jz black1
                        mov eax, green + (LightGray * 16)
                        jmp next
        black1:
                        mov eax, black + (LightGray * 16)
                        jmp next
sel:
                INVOKE CrateBlock, xposb, yposb, xsize, ysize, 5
                        mov al, inUse
                        dec al
                        jnz _greenorblack2
                        add eax, red + (blue * 16)
                        jmp next
        _greenorblack2:
                        dec al
                        jz black2
                        mov eax, green + (blue * 16)
                        jmp next
        black2:
                        mov eax, black + (blue * 16)
        next:
                call SetTextColor

                        mov  dl, xpost
                        mov  dh, ypost
                        call gotoxy
                        movsx eax, numb
                        call WriteDec

                        mov ax, white
                        call SetTextColor
                        mov  dl, 15
                        mov  dh, 15
                        call gotoxy

                        pop edx
        ret
CrateDecLabeledBlock ENDP

;//--------Kreira blok sa charom unutra &
CrateCharLabeledBlock PROC,
                xposb:  BYTE,           ;// x vrednost gornjeg levog ugla
                yposb:  BYTE,           ;// y vrednost gornjeg levog ugla
                xsize: BYTE,            ;// broj karaktera koje se boje udesno od xpos
                ysize: BYTE,            ;// broj karaktera koje se boje nanize od ypos
                xpost: BYTE,            ;// x polozaj teksta
                ypost: BYTE,            ;// y polozaj teksta
                text: BYTE,                     ;//Broj koji se upisuje
                isSelected: BYTE,       ;//Da li je selektovano polje
```

```
                inUse: BYTE                 ;//Da li se koristi vec simbol iz polja


                ;//Crta polje koje ako je selektovano je pravougaonik plave boje a ako
                ;//nije onda je sive boje. Ako je zabranjena operacija onda je sive boje
                ;//obojen a ako nije onda crnom.

                push edx

                mov al, isSelected
                dec al
                jz sel
        INVOKE CrateBlock, xposb, yposb, xsize, ysize, 3
                mov al, inUse
                dec al
                jnz _green1
                add eax, gray + (LightGray * 16)
                jmp next
_green1:
        mov eax, black + (LightGray * 16)
                jmp next
sel:
        INVOKE CrateBlock, xposb, yposb, xsize, ysize, 5
                mov al, inUse
                dec al
                jnz _green2
                add eax, gray + (blue * 16)
                jmp next
_green2:
        mov eax, black + (blue * 16)
                jmp next
next:
        call SetTextColor

                mov  dl, xpost
                mov  dh, ypost
                call gotoxy
                movsx eax, text
                call WriteChar

                mov ax, white
                call SetTextColor
                mov  dl, 0
                mov  dh, 0
                call gotoxy

                pop edx
                ret
CrateCharLabeledBlock ENDP

;//--------Kreira blok sa stringom unutra &
```

```
CrateStrLabeledBlock PROC,
                xposb:  BYTE, ;// x vrednost gornjeg levog ugla
                yposb:  BYTE, ;// y vrednost gornjeg levog ugla
                xsize: BYTE, ;// broj karaktera koje se boje udesno od xpos
                ysize: BYTE, ;// broj karaktera koje se boje nanize od ypos
                xpost: BYTE, ;// x polozaj teksta
                ypost: BYTE, ;// y polozaj teksta
                text: PTR BYTE, ;//Broj koji se upisuje
                isSelected: BYTE, ;//Da li je selektovano polje
                inUse: BYTE;//Da li se koristi vec simbol iz polja


                ;//Crta polje koje ako je selektovano je pravougaonik plave boje a ako
                ;//nije onda je sive boje. Ako je zabranjena operacija onda je sive boje
                ;//obojen a ako nije onda crnom.

                push edx

                mov al, isSelected
                dec al
                jz sel


        INVOKE CrateBlock, xposb, yposb, xsize, ysize, 3
                mov al, inUse
                dec al
                jnz _green1
                add eax, gray + (LightGray * 16)
                jmp next
_green1:
                mov eax, black + (LightGray * 16)
                jmp next
                sel :
        INVOKE CrateBlock, xposb, yposb, xsize, ysize, 5
                mov al, inUse
                dec al
                jnz _green2
                add eax, gray + (blue * 16)
                jmp next
_green2:
        mov eax, black + (blue * 16)
                jmp next
next:
        call SetTextColor

                mov  dl, xpost
                mov  dh, ypost
                call gotoxy
                mov edx, text
                call WriteString
```

```
                pop edx
                ret
CrateStrLabeledBlock ENDP

;//********Procedure koje pripadaju _GameMechanic.asm***********
;//---------Osvezavamo displej u toku igre
Refresh PROC,
        Arr: PTR WORD

    push edx
    push ebx
    push ecx
    push esi
    push edi

_drawAgain:
    mov esi, Arr
    movsx edi, Changed
                    ;//   Trazeni broj
    ;// 1.) 1-999
    shr edi, 1
        jnc _draw2
        mov dx, WORD PTR[esi]
        INVOKE CrateDecLabeledBlock, 18, 1, 9, 1, 21, 1, dx, 0, 2
        mov ax, Changed
        and ax, 1111111111111110b      ;//ne treba se vise menjati
        mov Changed,ax
                    ;// Upotrebljivi brojevi
    ;// 2.) 1-4
        _draw2:
            inc esi
            inc esi
        shr edi, 1
        jnc _draw3
        INVOKE CheckIfSelected,  2
            mov bh, al
        INVOKE CheckIfUsed,  2
            mov bl, al
        mov dx, WORD PTR[esi]
        INVOKE CrateDecLabeledBlock, 5, 3, 3, 1, 6, 3, dx, bh, bl
    ;// 3.) 1-4
        _draw3:
        inc esi
        inc esi
        shr edi, 1
        jnc _draw4
        INVOKE CheckIfSelected,  3
            mov bh, al
        INVOKE CheckIfUsed,  3
            mov bl, al
        mov dx, WORD PTR[esi]
```

```
        INVOKE CrateDecLabeledBlock, 9, 3, 3, 1, 10, 3, dx, bh, bl
;// 4.) 1-4
        _draw4:
                inc esi
                inc esi
                shr edi, 1
                jnc _draw5
        INVOKE CheckIfSelected,  4
                mov bh, al
        INVOKE CheckIfUsed, 4
                mov bl, al
        mov dx, WORD PTR[esi]
        INVOKE CrateDecLabeledBlock, 13, 3, 3, 1, 14, 3, dx, bh, bl
;// 5.) 1-4
        _draw5:
                inc esi
                inc esi
                shr edi, 1
        jnc _draw6
        INVOKE CheckIfSelected, 5
                mov bh, al
        INVOKE CheckIfUsed, 5
                mov bl, al
        mov dx, WORD PTR[esi]
        INVOKE CrateDecLabeledBlock, 17, 3, 3, 1, 18, 3, dx, bh, bl
;// 6.) {10,15,20}
        _draw6:
                inc esi
                inc esi
        shr edi, 1
        jnc _draw7
        INVOKE CheckIfSelected,  6
                mov bh, al
        INVOKE CheckIfUsed, 6
                mov bl, al
                mov dx, WORD PTR[esi]
        INVOKE CrateDecLabeledBlock, 25, 3, 6, 1, 27, 3, dx, bh, bl
;// 7.) {25,50,75,100}
        _draw7:
                inc esi
                inc esi
        shr edi, 1
        jnc _draw8
        INVOKE CheckIfSelected,  7
                mov bh, al
        INVOKE CheckIfUsed, 7
                mov bl, al
                mov dx, WORD PTR[esi]
        INVOKE CrateDecLabeledBlock, 32, 3, 9, 1, 35, 3, dx, bh, bl
        jmp _draw8
        ;// ----Operacije
```

```
;// 9.) +
        _draw9:
        shr edi, 1
        jnc _draw10
        INVOKE CheckIfSelected, 9
                mov bh, al
                mov dl, 02Bh
        INVOKE CheckIfUsed, 9
                mov bl, al
        INVOKE CrateCharLabeledBlock, 5, 7, 3, 1, 6, 7, dl, bh, bl
;// 10.) -
        _draw10:
        shr edi, 1
        jnc _draw11
        INVOKE CheckIfSelected, 10
                mov bh, al
                mov dl, 02Dh
        INVOKE CheckIfUsed, 10
                mov bl, al
        INVOKE CrateCharLabeledBlock, 9, 7, 3, 1, 10, 7, dl, bh, bl
;// 11.) *
        _draw11:
        shr edi, 1
        jnc _draw12
        INVOKE CheckIfSelected, 11
                mov bh, al
                mov dl, 2Ah
        INVOKE CheckIfUsed, 11
                mov bl, al
        INVOKE CrateCharLabeledBlock, 13, 7, 3, 1, 14, 7, dl, bh, bl
;// 12.) /
        _draw12:
        shr edi, 1
        jnc _draw13
        INVOKE CheckIfSelected, 12
                mov bh, al
                mov dl, 2Fh
        INVOKE CheckIfUsed, 12
                mov bl, al
        INVOKE CrateCharLabeledBlock, 17, 7, 3, 1, 18, 7, dl, bh, bl
;// 13) (
        _draw13:
        shr edi, 1
        jnc _draw14
        INVOKE CheckIfSelected, 13
                mov bh, al
                mov dl, 28h
        INVOKE CheckIfUsed, 13
                mov bl, al
        INVOKE CrateCharLabeledBlock, 21, 7, 3, 1, 22, 7, dl, bh, bl
;// 14.) )
```

```
        _draw14:
        shr edi, 1
        jnc _draw15
        INVOKE CheckIfSelected, 14
                mov bh, al
                mov dl, 29h
        INVOKE CheckIfUsed, 14
                mov bl, al
        INVOKE CrateCharLabeledBlock, 25, 7, 3, 1, 26, 7, dl, bh, bl
;// 15.) DEL
        _draw15:
        shr edi, 1
        jnc _draw16
        INVOKE CheckIfSelected,  15
                mov bh, al
                mov edx, OFFSET txtDelete
        INVOKE CheckIfUsed, 15
                mov bl, al
        INVOKE CrateStrLabeledBlock, 31, 7, 5, 1, 32, 7, edx, bh, bl
;// 16.) CE
        _draw16:
        shr edi, 1
        jnc _end_l
        INVOKE CheckIfSelected, 16
                mov bh, al
                mov edx, OFFSET txtCE
        INVOKE CheckIfUsed, 16
                mov bl, al
        INVOKE CrateStrLabeledBlock, 37, 7, 4, 1, 38, 7, edx, bh, bl
        jmp _end_l
;// 8.) FORMULA
        _draw8:
        shr edi, 1
        jnc _draw9
                movsx esi, FormulaSize
                cmp esi, 0
                jne _ChangeFormula
                        ;inicijalna formula
                        INVOKE CrateBlock, 5, 5, 36, 1, 6
                        mov ax, black + (16 * white)
                        call SetTextColor
                        jmp _draw9
        _ChangeFormula:
                mov ax, black + (16 * white)
                call SetTextColor
                dec esi
                add esi, OFFSET Formula
                mov dx, CursorPos       ;//pozicija kursora u formuli
                call gotoxy

                movsx eax, BYTE PTR[esi]       ;//proveravamo sta upisujemo
```

```
                cmp eax, 8
                jb _dec                                 ;//upisujemo broj
                cmp eax, 14
                ja _dir                                 ;//brisemo
        ;//Ispisujemo iz formule znak
                mov ebx, eax
                mov eax, 02Bh  ;//'+'
                sub ebx, 9
                jz _operation
                mov eax, 02Dh  ;//'-'
                dec ebx
                jz _operation
                mov eax, 02Ah  ;//'*'
                dec ebx
                jz _operation
                mov eax, 02Fh  ;//'/'
                dec ebx
                jz _operation
                mov eax, 028h  ;//'('
                dec ebx
                jz _operation
                mov eax, 029h  ;//')'
                dec ebx
                jz _operation
                mov eax, 03Fh  ;//'?'
_operation:
                call WriteChar
                inc dx
                jmp _wrote
        ;//---Upisujemo u formulu broj
_dec:
                dec eax
                imul eax, 2
                mov esi, OFFSET Numbers
                add esi, eax
                movsx eax, BYTE PTR[esi]
                call WriteDec
                cmp eax, 100
                je _dec3
                cmp eax, 9
                ja _dec2
        _dec1:  ;//jednocifreni
                inc dx
                jmp _wrote
        _dec2: ;//dvocifreni
                inc dx
                inc dx
                jmp _wrote
        _dec3: ;//trocifreni
                inc dx
                inc dx
```

```
                inc dx
                jmp _wrote
        ;//----BRISEMO
_dir:
                mov ax, FormulaSize
                cmp ax, 1
                je _draw9
                dec FormulaSize
                mov bx, FormulaSize
                cmp bx, 1
                je _CE
                mov Changed, 0FF7Fh
                mov al, BYTE PTR[esi];//proveravamo koje polje aktiviramo
                cmp al, 16
                je _CE
        ;//_DEL
                        dec FormulaSize
                        dec esi
                        mov al, BYTE PTR[esi];//proveravamo koje polje brisemo
                        cmp al,  8
                        jb _decdel
                        cmp al, 14
                        je _closedel
                        cmp al, 13
                        je _opendel
                ;//--Brisemo operaciju
                        mov ax, 010FFh
                        and ax, Used
                        mov Used, ax
                        mov ax, Changed
                        and ax, 03F00h
                        mov Changed, ax
                        jmp _dec1del
                ;//--Brisemo broj
        _decdel:
                        INVOKE DecToBin, al         ;//dozvoljavamo broj koji smo obrisali
                        mov ah, 0
                        mov bx, ax
                        not ax
                        and ax, Used
                        mov ah, 02Fh                ;//dozvoljavmo '('
                        mov Used, ax
                        mov bh, 03Fh
                        and bx, Changed
                        mov Changed, bx
                        movsx eax, BYTE PTR[esi]    ;//proveravamo koje polje aktiviramo
                        dec eax
                        imul eax, 2
                        mov esi, OFFSET Numbers
                        add esi, eax
                        movsx eax, BYTE PTR[esi]    ;//proveravamo sta brisemo
```

```
        cmp eax, 10
        jb _dec1del
        cmp eax, 100
        jb _dec2del
        jmp _dec3del
_dec1del: ;//brisemo jednocifreni
        dec dx
        dec CursorPos
        INVOKE CrateBlock, dl, 5, 2, 1, 6
        jmp _drawAgain
_dec2del: ;//brisemo dvocifreni
        dec dx
        dec dx
        dec CursorPos
        dec CursorPos
        INVOKE CrateBlock, dl, 5, 3, 1, 6
        jmp _drawAgain
_dec3del: ;//brisemo trocifreni
        dec dx
        dec dx
        dec dx
        dec CursorPos
        dec CursorPos
        dec CursorPos
        INVOKE CrateBlock, dl, 5, 4, 1, 6
        jmp _drawAgain
_opendel: ;//brisemo '('
        dec NeedToClose
        mov ax, 0010111110000000b          ;//zabrane se operacije
        or ax, Used
        and ax, 0010111111111111b          ;//dozvoli se '('
        mov Used, ax
        mov Changed, 0FF00h
        dec dx
        dec CursorPos
        INVOKE CrateBlock, dl, 5, 3, 1, 6
        jmp _drawAgain
_closedel: ;//brisemo ')'
        inc NeedToClose
        mov ax, 0000000011111111b          ;//dozvole se operacije
        or  ax, 0001000000000000b          ;//zabrani se '('
        and ax, Used
        mov Used, ax
        mov Changed, 0FF80h
        dec CursorPos
        dec dx
        dec esi
        movsx eax, BYTE PTR[esi]       ;//proveravamo koje polje brisemo
        dec al
        imul eax, 2
        mov esi, OFFSET Numbers
```

```
                    add esi, eax
                    movsx eax, BYTE PTR[esi]      ;//proveravamo sta brisemo
                    cmp eax, 10
                    jb _dec1del
                    cmp eax, 100
                    jb _dec2del
                    jmp _dec3del
        _CE:    ;//Brisemo sve iz formule i stavljamo da je dostupno
                INVOKE CrateBlock, 5, 5, 36, 1, 6
                mov ax, black + (16 * white)
                call SetTextColor
                mov FormulaSize, 0
                mov NeedToClose, 0
                mov Selected, 2
                mov Used, 1110111110000000b
                mov CursorPos, 0508h
                mov Changed, 0FFFFh
                jmp _drawAgain

        _wrote:
                mov CursorPos, dx
                jmp _draw9
    _end_l:
                mov Changed, 0
                pop edi
                pop esi
                pop ecx
                pop ebx
                pop edx
                ret
Refresh ENDP

;//**********Procedure iz _GameMechanic.asm**********************
;//----------Resetuje sve parametre igre
RestartVar PROC

    mov Selected, 5                                 ;//Da li je selektovan
    mov Used, 1110111110000000b             ;//Da li je koriscen
    mov Changed,1111111111111111b           ;//Da li je promenjen
    mov CursorPos, 0508h
    mov FormulaSize, 0
    mov NeedToClose, 0

ret
RestartVar ENDP

;//----------Pocetna strana Igre
Menu PROC
        invoke GetStdHandle, STD_OUTPUT_HANDLE
         ;// Postavlja handle za ispis podataka
        mov  stdOutHandle, eax
```

```
		invoke GetConsoleCursorInfo, stdOutHandle, addr cursorInfo	;// Cita trenutno stanje kursora
		mov  cursorInfo.bVisible, 0
		 ;// Postavlja vidljivost kursora na nevidljiv
		invoke SetConsoleCursorInfo, stdOutHandle, addr cursorInfo	;// Postavlja novo stanje kursora

		invoke SetConsoleTitle, addr winTitle
;// Postavlja title prozora
		invoke SetConsoleWindowInfo, stdOutHandle, TRUE, addr windowRect ;// Dimenzije prozora
		call clrscr

		mov eax, green + (16* black)
		call SetTextColor
		mov dx, 0040Bh
		call gotoxy
		mov edx, OFFSET txtStart1 ;//Ispisuje: MOJ BROJ
		call WriteString
		mov dx, 00509h
		call gotoxy
		mov edx, OFFSET txtStart2 ;//Ispisuje: Press any number
		call WriteString
		mov dx, 0
		call gotoxy

	_LookForKey:
			mov esi, OFFSET Formula
			movsx edi, NeedToClose
			mov  eax, 50			;//sleep, to allow OS to time slice
			call Delay			;//(otherwise, some key presses are lost)
			call ReadKey			;//look for keyboard input
			jz  _LookForKey			;//no key pressed yet
			push ax
			call clrscr
			pop ax
	ret
Menu ENDP

;//----------Igranje
PlayGame PROC
			push edx
			push ebx
			push ecx
			push esi
			push edi
```

```
mov esi, OFFSET Formula
mov ecx, 10                                    ;//toliko '(' sme max da bude

_LookForKey:
        mov esi, OFFSET Formula
        movsx edi, NeedToClose
        mov  eax, 50                                    ;//pauziraj  da
OS prihvati taster
        call Delay                                      ;//da  se
ne propusti neki simbol
        call ReadKey                                    ;//Procitaj
uneseno
        jz  _LookForKey                                 ;//nije pritisnuto
nista
                cmp     al, SPACE_KEY
        je      _space
                cmp  al, ENTER_KEY
        je      _enter
                cmp  al, ESC_KEY
        je      _exit
                cmp  dx, UP_KEY
        je      _upkey
                cmp  dx, DOWN_KEY
        je      _downkey
                cmp  dx, LEFT_KEY
        je      _leftkey
                cmp  dx, RIGHT_KEY
        je      _rightkey


;//---ENTER-----------------------
_enter:
        cmp FormulaSize, 0      ;//preskoci ako nije uneseno nista
        je _LookForKey
                jmp _end_l
;//---ESCAPE----------------------
_exit:
        push eax
        mov eax, white + (black * 16)
        call SetTextColor
        pop eax
        jmp _end_l
;//---SPACE-----------------------
_space:
                        INVOKE CheckIfUsed, Selected
                        jnz  _LookForKey                ;//Proverava  da
li je dozvoljeno da se izabere ovo polje
                        movsx edx, FormulaSize
                        cmp edx, 0
                        jz _initial             ;//skok ako je prvi izabrani
```

```
                                        add esi, edx
                                        dec esi
                                        mov al, BYTE PTR [esi]          ;//prethodno
izabrano polje
                                        cmp al, 8                       ;//Da li je bio
broj ili je bila operacija prethodni put
                                        jb _number
        ;//bio je broj
                                        cmp al, 13
                                        je _opened
        ;//bila je '('
                                        cmp al, 14
                                        jne _operation                          ;//bila
je operacija
                                        jmp _closed
        ;//bila je ')'
                        ;//prvi u izabrani
                                _initial:
                                        cmp Selected, 14
                                        ja _reset                               ;//ako je
DEL inicijalni
                                        cmp Selected, 13
                                        je _initial_open        ;//ako je '(' , inicijalni
                                ;//Inicijalni izabrani je broj

                                        INVOKE DecToBin, Selected
                                                or ax, Used             ;//broj       je
iskoriscen
                                                mov Used, ax
                                                mov ah, 0FFh    ;//menja     se     izgled
operacija
                                                mov Changed, ax

                                                mov ax, Used    ;//Dozvoli       operacije
(osim '(')
                                                mov ah, 16
                                                mov Used, ax

                                                mov al, Selected
                                                mov BYTE PTR[esi], al;//smesti    redni
broj polja broja u formulu
                                                inc FormulaSize
                                                jmp _refresh
                        ;//prethodni uzet je broj
                                _number:
                                        mov al, Selected
                                        cmp al, 14
                                        je _close                               ;//ako je ')'
                                        mov al, Selected
                                        cmp al, 9
```

```
                                        mov Changed, 0
                                        jb _refresh

                                        inc FormulaSize
        ;//smesta se operacija
                                        inc esi
                                        mov al, Selected
                                        mov BYTE PTR[esi], al

                                        mov ax, 0010111100000000b    ;//zabrane se operacije
                                        or ax, Used
                                        and ax, 1110111111111111b    ;//dozvoli se otvorena
                                        mov Used, ax
                                        mov Changed, 0011111110000000b
                                        jmp _refresh

                ;//prethodni uzet je operacija
                        _operation:
                                        cmp Selected, 13                        ;//ako je
'(' treba da se upise
                                        je _open
                                        cmp Selected, 14                        ;//ako je
')' treba da se upise
                                        je _close

                                        inc FormulaSize
                                        inc esi
                                        mov al, Selected
                                        mov BYTE PTR[esi], al

                                        INVOKE DecToBin, Selected
                                        or ax, Used             ;//broj je iskoriscen
                                        mov Used, ax
                                        mov ah, 0FFh    ;//menja se izgled operacija
                                        or al, 080h
                                        mov Changed, ax

                                        mov ax, Used    ;//Dozvoli operacije (osim '(')
                                        mov ah, 010h
                                        mov Used, ax

                                        jmp _refresh

                        ;//prethodna ')'
                        _closed:
                                        mov Changed, 0
                                        mov ax, Used    ;//Zabrani operacije
                                        mov ah, 02Fh
                                        mov Used, ax

                                        cmp Selected, 14
```

```
                                        jne _skip
                                        cmp Selected, 8
                                        jb _LookForKey
                                        dec NeedToClose
                                        cmp NeedToClose, 0FFh
                                        je _allClosed

                                        mov ax, Used    ;//Dozvoli operacije operacije (osim '(')
                                        mov ah, 010h
                                        mov Used, ax
                                _skip:  ;//preskace provere ako nije ')' ponovo

                                        inc FormulaSize
        ;//smesta se operacija
                                        inc esi
                                        mov al, Selected
                                        mov BYTE PTR[esi], al

                                        mov ah, 03Fh                    ;//menja se izgled
operacija
                                        or al, 080h
                                        mov Changed, ax

                                        jmp _refresh
                        ;//unosi se ')'
                                _close:
                                        mov Changed, 0
                                        cmp Selected, 8
                                        jb _LookForKey
                                        cmp NeedToClose, 0
                                        je _allClosed

                                        dec NeedToClose
                                        inc FormulaSize                 ;//smesta se broj
ili ')'
                                        inc esi
                                        mov al, Selected
                                        mov BYTE PTR[esi], al
                                        mov ax, 0001000011111111b       ;//dozvole se
operacije
                                        and ax, Used
                                        mov Used, ax
                                        mov Changed, 0FF80h
                                        jmp _refresh

                        ;//Prvi uneti u formulu '('
                                _initial_open:
                                        mov al, Selected
                                        mov BYTE PTR[esi], al;//smesti redni broj polja broja u
formulu
                                        inc FormulaSize
```

```
                inc NeedToClose
                inc edi
                mov ax, 0010011111111111b
                and ax, Used
                mov Used, ax
                mov Changed, 1111000010000000b
                jmp _refresh
        ;//bila je '('
        _opened:
                cmp Selected, 13
                jne _skip_opened1
                cmp NeedToClose, 6
                ja _overOpening
                inc NeedToClose
        _skip_opened1:
                inc FormulaSize                 ;//smesta se broj
ili '('
                inc esi
                mov al, Selected
                mov BYTE PTR[esi], al

                INVOKE DecToBin, Selected
                or ax, Used             ;//broj je iskoriscen
                mov ah, 02Fh
                cmp Selected, 13
                je _skip_opened2        ;//u slucaju da ponovo '(' ne
dopustaju se operacije
                mov ah, 020h
        _skip_opened2:
                mov Used, ax
                mov ah, 0FFh
                or al, 080h
                mov Changed, ax

                jmp _refresh
        ;// izabran je '('
        _open:
                cmp NeedToClose, 5
                ja _overOpening

                inc NeedToClose
                inc FormulaSize
                inc esi
                mov al, Selected
                mov BYTE PTR[esi], al                   ;//smesta se '('

                mov ax, 0010111110000000b               ;//zabrane      se
operacije
                or ax, Used
                mov Used, ax
                mov Changed, 0011111110000000b
```

```
                                        jmp _refresh
                                    _overOpening:
        ;//zabrani se dodatno otvaranje zagrada
                                        mov ax,0001000000000000b
                                        or ax, Used
                                        mov Used, ax
                                        mov Changed, 0011111100000000b      ;//menja      se
izgled operacija
                                        jmp _refresh
                                    _allClosed:
                                        mov ax , Used
                                        and ax, 1111000011111111b
                                        or ax,  0011000000000000b
                                        mov Used, ax
                                        mov Changed, 0010000010000000b      ;//menja      se
izgled operacija
                                        jmp _LookForKey
                                    _reset:
                                        mov NeedToClose, 0
                                        mov FormulaSize, 0
                                        mov Selected, 2
                                        mov Changed, 0FFFFh
                                        mov Used, 1110111110000000b
                                        mov CursorPos, 0508h
                                        jmp _refresh
                ;//---UPKEY-----------------------
                _upkey:
                                        INVOKE DecToBin, Selected
                                        mov Changed, ax
        ;//Upisujemo u Changed, koji cemo morati menjati jer je kursor promenjen od njega

                                        movsx dx, Selected
                                        ;dec dx
                                        cmp dx, 9                           ;//Da     li     se
kursor nalazi u gornjem redu
                                        jb _skokna9
                                        mov Selected, 2                     ;ne
                                        jmp _novidisplej
                                    _skokna9:
                                        mov Selected, 9                     ;da
                                        jmp _novidisplej
                ;//---DOWNKEY----------------------
                _downkey:
                                        INVOKE DecToBin, Selected
                                        mov Changed, ax
        ;//Upisujemo u Changed, koji cemo morati menjati jer je kursor promenjen od njega

                                        movsx dx, Selected
                                        ;dec dx
                                        cmp dx, 7                           ;//Da     li     se
kursor nalazi u donjem redu
```

```
                ja _skokna2
                mov Selected, 9                 ;//ne
                jmp _novidisplej
            _skokna2:
                mov Selected, 2                 ;//da
                jmp _novidisplej
        ;//---LEFTNKEY----------------------
        _leftkey:
                INVOKE DecToBin, Selected
                mov Changed, ax
    ;//Upisujemo u Changed, koji cemo morati menjati jer je kursor promenjen od njega

                movsx dx, Selected
                ;dec dx
                cmp dx, 7                       ;//Da li se kursor nalazi u donjem redu
                ja _donja1
                                                ;//ne
                cmp dx, 2
        ;// Da li je kursor skroz levo
                je _skokna7
                dec dx
            ;//ne
                mov Selected, dl
                jmp _novidisplej
            _skokna7:
            ;//da
                mov Selected, 7
                jmp _novidisplej
                                                ;//da
            _donja1:
                cmp dx, 9
        ;// Da li je kursor skroz levo
                je _skokna16
                dec dx
            ;//ne
                mov Selected, dl
                jmp _novidisplej
            _skokna16:
            ;//da
                mov Selected, 16
                jmp _novidisplej
        ;//---RIGHTKEY----------------------
        _rightkey:
                INVOKE DecToBin, Selected
                mov Changed, ax
    ;// Upisujemo u Changed, koji cemo morati menjati jer je kursor promenjen od njega

                movsx dx, Selected
                ;dec dx
```

```
                                        cmp dx, 7                       ;// Da li se kursor nalazi
u donjem redu
                                        ja _donja2
                                                                        ;//ne
                                        cmp dx, 7
                ;// Da li je kursor skroz desno
                                        je _skokna2
                                        inc dx
                        ;//ne            ('da' skace na broj skroz levi)
                                        mov Selected, dl
                                        jmp _novidisplej
                                    _donja2:                            ;//da
                                        cmp dx, 16
                ; //Da li je kursor skroz desno
                                        je _skokna9
                                        inc dx
                        ;//ne           ('da' skace na polje '+')
                                        mov Selected, dl
                                        jmp _novidisplej
                        _novidisplej:
                                        INVOKE DecToBin, Selected
                                        or ax, Changed
                                        mov Changed, ax                 ;//
Upisujemo u Changed, jer cemo menjati posto je trenutno taj selektovan
                                        jmp _refresh

                _refresh:
                                INVOKE Refresh, OFFSET Numbers
                                jmp _LookForKey

                        _end_l:

                                pop edi
                                pop esi
                                pop ecx
                                pop ebx
                                pop edx
                                ret
PlayGame ENDP


;//----------Pokretanje Igre
StartGame PROC
        INVOKE RestartVar
        INVOKE InitLogic, OFFSET Numbers
        INVOKE Refresh, OFFSET Numbers
        ret
StartGame ENDP

;//----------Razultati igre
Results PROC
```

```
push edx
push ebx
push ecx
push esi
push edi

mov edi, OFFSET Formula
mov ecx, 0
cmp FormulaSize, 1
jne _lookForOpen
        mov cl, BYTE PTR[edi]
        dec cl
        imul cx, 2
        mov esi, OFFSET Numbers
        add esi, ecx
        mov ax, WORD PTR[esi]
        push ax
jmp _out
mov ecx, 0
_lookForOpen:
                mov al , BYTE PTR[edi]
                cmp al, 13
                jne _notOpen
                mov openPar, ecx
        _notOpen:
                inc edi
                inc ecx
                cmp al , 14
                je _foundOpen
                cmp cx, FormulaSize
                je _notFoundOpen
        jmp _LookForOpen

_foundOpen:
        dec edi
        dec ecx
        mov closedPar, ecx      ;//polozaj ')'

        mov ecx, openPar
        mov edi, OFFSET Formula
        add edi, ecx
_LookForMD:
                inc ecx
                inc edi
                mov al, BYTE PTR[edi]
                cmp al, 11
                je _foundM
                cmp al, 12
                je _foundD
                cmp ecx, closedPar
```

```
                je _notFoundMD
        jmp _LookForMD

_foundM:
                dec edi
                movsx eax, BYTE PTR[edi]
                cmp eax, 1
                je _mul1
                dec ax
                imul ax,2
                mov esi, OFFSET Numbers
                add esi, eax
                movsx eax, BYTE PTR[esi]      ;//prvi cinilac
        _mul1sol:
                inc edi
                inc edi
                movsx ebx, BYTE PTR[edi]
                cmp bx, 1
                je _mul2
                dec bx
                imul bx, 2
                mov esi, OFFSET Numbers
                add esi, ebx
                movsx ebx, BYTE PTR[esi]      ;//drugi cinilac
        _mul2sol:
                imul ax, bx
                push ax

                ;//Setuj izracunato na 1, znak koji
                ;//odredjuje da je vrednost na steku
                mov BYTE PTR[edi], 1 ;// prvi cinilac
                dec edi
                mov BYTE PTR[edi], 1 ;// *
                dec edi
                mov BYTE PTR[edi], 1 ;// drugi cinilac
                mov ecx, openPar
                mov edi, OFFSET Formula
                add edi, ecx
                jmp _lookForMD
        _mul1:
                pop ax
                jmp _mul1sol
        _mul2:
                pop bx
                jmp _mul2sol

_foundD:
                dec edi
                movsx eax, BYTE PTR[edi]      ;//id kolicnika
                cmp ax, 1
                je _div1
```

```
                dec ax
                imul ax,2
                mov esi, OFFSET Numbers
                add esi, eax
                movsx eax, BYTE PTR[esi]      ;//kolicnik
                _div1sol:
                inc edi
                inc edi
                movsx ebx, BYTE PTR[edi]      ;//id delioca
                cmp bx, 1
                je _div2
                dec bx
                imul bx, 2
                mov esi, OFFSET Numbers
                add esi, ebx
                movsx ebx, BYTE PTR[esi]      ;//delilac
                _div2sol:

                mov dx, 0
                div bx
                push ax

                ;//Setuj izracunato na 1, znak koji
                ;//odredjuje da je vrednost na steku
                mov BYTE PTR[edi], 1 ;// kolicnik
                dec edi
                mov BYTE PTR[edi], 1 ;// /
                dec edi
                mov BYTE PTR[edi], 1 ;// delilac
                mov ecx, openPar
                mov edi, OFFSET Formula
                add edi, ecx
        jmp _LookForMD
                _div1:
                        pop ax
                        jmp _div1sol
                _div2:
                        pop bx
                        jmp _div2sol

_notFoundMD:
        mov ecx, closedPar
        mov edi, OFFSET Formula
        add edi, ecx
_lookForAS:
        dec ecx
        dec edi
        mov al, BYTE PTR[edi]
        cmp al, 9
        je _foundA
        cmp al, 10
```

```
je _foundS
cmp ecx, openPar
je _notFoundAS
jmp _lookForAS

_foundA:        ;//-----SABIRANJE
                dec edi
                movsx eax, BYTE PTR[edi]
                cmp ax, 1
                je _add1
                dec ax
                imul ax,2
                mov esi, OFFSET Numbers
                add esi, eax
                movsx eax, BYTE PTR[esi]      ;// prvi sabirak
        _add1sol:
                inc edi
                inc edi
                movsx ebx, BYTE PTR[edi]
                cmp bx, 1
                je _add2
                dec bx
                imul bx, 2
                mov esi, OFFSET Numbers
                add esi, ebx
                movsx ebx, BYTE PTR[esi]      ;// drugi sabirak
        _add2sol:

                add ax,bx
                push ax ;//zbir

                ;//Setuj izracunato na 1, znak koji
                ;//odredjuje da je vrednost na steku
                mov BYTE PTR[edi], 1 ;// prvi sabirak
                dec edi
                mov BYTE PTR[edi], 1 ;// *
                dec edi
                mov BYTE PTR[edi], 1 ;// drugi sabirak
                mov ecx, closedPar
                mov edi, OFFSET Formula
                add edi, ecx
                jmp _lookForAS
        _add1:
                pop ax
                jmp _add1sol
        _add2:
                pop bx
                jmp _add2sol

_foundS: ;//----ODUZIMANJE
                mov edx, edi
```

```
                inc edi
                mov esi, OFFSET Numbers
                movsx eax, BYTE PTR[edi]
                cmp ax, 1
                je _sub1
                dec ax
                imul ax,2
                add esi, eax
                movsx eax, BYTE PTR[esi]
        _sub1sol:
                dec edi
                dec edi
                mov esi, OFFSET Numbers
                movsx ebx, BYTE PTR[edi]
                cmp bx, 1
                je _sub2
                dec bx
                imul bx, 2
                add esi, ebx
                movsx ebx, BYTE PTR[esi]
        _sub2sol:
                sub bx, ax
                push bx

                ;//Setuj izracunato na 1, znak koji
                ;//odredjuje da je vrednost na steku
                mov BYTE PTR[edi], 1 ;// umanjenik
                inc edi
                mov BYTE PTR[edi], 1 ;// *
                inc edi
                mov BYTE PTR[edi], 1 ;// umanjilac
                mov ecx, closedPar
                mov edi, OFFSET Formula
                add edi, ecx
                jmp _lookForAS
        _sub1:
                pop ax
                jmp _sub1sol
        _sub2:
                pop bx
                jmp _sub2sol
_notFoundAS:
        ;//brisemo zagrade        koje smo resili
        mov edi, OFFSET Formula
        add edi, openPar
        mov BYTE PTR[edi], 1
        mov edi, OFFSET Formula
        add edi, closedPar
        mov BYTE PTR[edi], 1

        ;//da li smo zavrsili
```

```
                cmp openPar, 0
                jne _skipOut
                mov ebx, 0
                mov bx, FormulaSize
                dec bl
                cmp closedPar, ebx
                je _out
        _skipOut:
                mov edi, OFFSET FORMULA
                mov ecx, 0
                jmp _lookForOpen

        _notFoundOpen:          ;//kada smo sve zagrade resili

                mov edi, OFFSET Formula
                movsx ebx, FormulaSize
                mov closedPar, ebx
                dec closedPar
                mov ecx, 0
                mov openPar, 0
                jmp _lookForMD

_out:
        ;// ispisuje rezultat
        mov eax, black + (white*16)
        call SetTextColor
   mov  dl, 11
        mov  dh, 10
        call gotoxy
        mov edx, OFFSET txtResults1
        call WriteString
        mov  dl, 11
        mov  dh, 11
        call gotoxy
        mov edx, OFFSET txtResults2
        call WriteString

        mov eax, green + (white*16)
        call SetTextColor

        mov  dl, 30
        mov  dh, 10
        call gotoxy
        pop ax
        cmp ax, 1000
        jb _positive
        mov ax, 0               ;//Ako je negativan upisi 0
_positive:
        call WriteDec

        mov ecx, 0
```

```
	mov cx, ax
	mov edi, OFFSET Numbers
	mov bx, WORD PTR[edi]
	sub cx, bx
	cmp cx, 1000
	jb _positivResult		;//Apsolutna razlika trazenog i nadjenog
	mov cx, ax
	sub bx, cx
	mov cx, bx
	_positivResult:
	mov ax, cx

	mov  dl, 30
	mov  dh, 11
	call gotoxy
	call WriteDec

	mov  dl, 30
	mov  dh, 14
	call gotoxy
	mov eax, white + (black * 16)
	call SetTextColor

	_LookForKey:
			mov  eax, 50;//sleep, to allow OS to time slice
			call Delay;//(otherwise, some key presses are lost)
			call ReadKey;//look for keyboard input
			jz   _LookForKey;//no key pressed yet

		pop edi
		pop esi
		pop ecx
		pop ebx
		pop edx
	ret
Results ENDP

;//*********GLAVNI PROGRAM _MojBroj.asm******************
main proc
	call clrscr

_RestartGame:
	INVOKE Menu				;//Pocetni meni, ESC izlzak iz programa, ostalo pocetak
igre
		cmp al, ESC_KEY
		je _exit
	INVOKE StartGame		;//Inicijalizacija displeja i parametara
	INVOKE PlayGame			;//Proces igranja igre
		cmp al, ESC_KEY
		je _RestartGame
	INVOKE Results			;//Izracunavanje rezultata
```

```
                cmp  al, ENTER_KEY
                je      _RestartGame
                cmp  al, SPACE_KEY
                je      _RestartGame
_exit:
        INVOKE ExitProcess, 0
main endp
END main
```