Univerzitet u Beogradu
Elektrotehnički fakultet

Predmet:
Računarska elektronika

Tema projektnog zadatka:

# Igrica Battleship u x86 asembleru

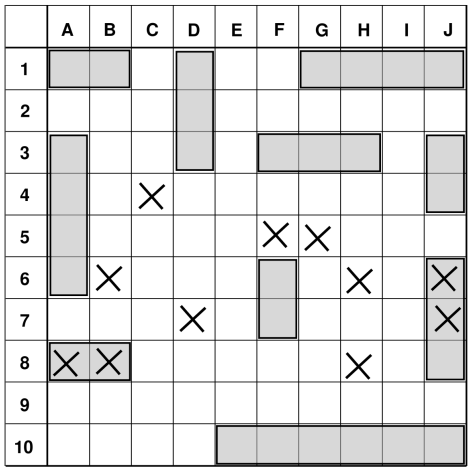
Ksenija Josipović 124/14
jk140124d@student.etf.bg.ac.rs

Mihajlo Karličić 491/14
km140491d@student.etf.bg.ac.rs

31. maj 2017

# Tekst projektrnog zadatka

Potrebno je realizovati igru potapanje brodića (Battleship). Standardni izgled table za brodiće prikazan je na slici 1.




Slika 1: Igrica Battleship

Potrebno je u aplikaciji najpre omogućiti igračima da unesu imena fajlova u kojima se nalaze njihove mape rasporeda brodića a koji treba da sadrže 10 redova sa po 10 karaktera razdvojenih razmakom u sledećem formatu:

```
1 1 - 2 - - 3 3 3 3
- - - 2 - - - - - -
6 - - 2 - 4 4 4 - 5
6 - - - - - - - - 5
6 - - - - - - - - -
6 - - - - 9 - - - 0
- - - - - 9 - - - 0
7 7 - - - - - - - 0
- - - - - - - - - -
- - - - 8 8 8 8 8 -
```

Ovaj fajl bi predstavljao opis mape kao na slici 1. Mapa sadrži 10 brodića numerisanih ciframa 0-9, i to 1 brodć od 5 polja, 2 brodića od 4 polja, 3 brodića od 3 polja i 4 brodića od 2 polja, koji mogu biti postavljeni horizontalno ili vertikalno. Nakon što igrači unesu imena fajlova u kojima se nalaze njihove mape, potrebno je iscrtati dve mape koje odgovaraju svakom pojedinačnom igraču:

```
      Player 1                    Player 2
  A B C D E F G H I J          A B C D E F G H I J
0 - - - - - - - - - -        0 - - - - - - - - - -
1 - - - - - - - - - -        1 - - - - - - - - - -
2 - - - - - - - - - -        2 - - - - - - - - - -
3 - - - - - - - - - -        3 - - - - - - - - - -
4 - - - - - - - - - -        4 - - - - - - - - - -
5 - - - - - - - - - -        5 - - - - - - - - - -
6 - - - - - - - - - -        6 - - - - - - - - - -
7 - - - - - - - - - -        7 - - - - - - - - - -
8 - - - - - - - - - -        8 - - - - - - - - - -
9 - - - - - - - - - -        9 - - - - - - - - - -
```

Borba se odvija tako što igrači sa standardnog ulaza unose koordinate polja koje žele da gađaju u formi slovo pa broj, koji nisu razdvojeni razmakom (A5, F3, ...). Ukoliko je igrač promašio protivnički brod, potrebno je na tom polju upisati x i omogućiti drugom igraču da pogađa, dok u slučaju da je igrač pogodio protivnički brod, na tom polju potrebno je iscrtati blok karakter (DBh) i omogućiti istom igraču da pogadja ponovo. Pobednik je onaj igrač koji prvi pogodi sve protivničke brodiće.

# Postupak rešavanja zadatka

Zadatak smo rešavali metodom zavadi pa vladaj — problem je najpre podeljen na dva jednostavnija problema:

- Učitavanje fajla i provera poštovanja pravila
- Igranje i ispis u konzolu

U pri rešavanju prvog problema smo koristili postojeći kod iz primera `07_ReadFile.asm` sa vežbi. Nadogradnjom tog primera su učitani podaci o poziciji brodića iz ulaznih fajlova. Jedostavnom proverom da li su na očekivanim pozicijama znaci za kraj reda i za novi red vrši se provera da li je korisnik prosledio bar donekle validan fajl. Zatim se otklonjaju svi blanko znaci (space, kraj reda i novi red) i proveri se da li ima tačno 100 preostalih znakova. U daljem kodu se vrši provera da li su ti preostali znaci cifre (0-9) ili crtica (2Dh).
Nakon što je utvrđeno da je korisnik uneo validan fajl potrebno je proveriti da li su ispoštovana pravila igre. Brodići se ne smeju dodirivati niti savijati. To je urađeno tako što je za svako polje gde je neki brodić provereno da li se u njegovoj okolini nalazi neki drugi brodić (obeležen drugom cifrom) i da li su susedni delovi istog brodića samo vertikalno ili samo horizontalno postavljeni. Pošto se utvrdi da je korisnik ispoštovao i ta pravila vrši se provera da li je postavljen broj brodića onako kako je određeno tekstom zadatka. To se proerava tako što se prebroji koliko je polja obeleženo kojom cifrom, gde svakom cifrom mora biti obeležen tačno jedan brodić. Rezultat tog brojanja inkrementira određeni brojač kojim se broji koliko puta se pojavljuje brodić određene veličine (dva, tri, četiri ili pet polja). Na kraju se samo proveri da li su ti brojači jednaki onome što pravila diktiraju.
U drugom delu projekta bilo je potrebno napisati deo koda koji se bavio proverom da li se na datom pomeraju nalazi brodić ili ne, a zatim iscrtava tabelu u kojoj stoji blok karakter (DBh) ako je došlo do pogotka ili 'x' ako je došlo do promašaja.
Prvo se od igrača zahteva unos koordinate na koju sumnja da krije neprijateljski brodić. Koordinata se unosi u formatu SlovoBroj, (tj. A5, F9) s tim da je ograničavajći faktor da slovo mora da bude veliko. Brojevi se kreću u opsegu 0-9 a slova A-J. Kada se unese koordinata potrebno je od nje napraviti linearni pomeraj kako bi se testirao protivnički niz na postojanje brodića na tom polju. To se rešava tako što se uneto slovo umanji za 64 (decimalno) i od njega se dobije broj. Tako bi A postalo 1, B 2 itd.. Broj se umanjuje za 48 I prebacuje se u decimalnu vrednost. Zatim se tako dobijeni broj množi sa 10 i sabira sa vrednošću slova i tako se dobija linarni pomeraj.
Nakon toga, bilo je potrebno proveriti da li se u datom nizu nalazi '-' ili broj, kako bi se znalo da li je igrač pogodio ili promašio. Potom se izračunava pomeraj za iscrtavanje tabele, koji se razlikovao od pomeraja za pretragu niza jer se niz za iscrtavanje realizovao kao:

```
crtice1 BYTE 10 DUP(1 DUP('5'), 10 DUP('-'), 1 DUP('␣␣␣␣'))
```

To znači da ovaj niz ima 120 karaktera ili 12 karaktera po redu tako da se vrednost promenljive ***i*** morala promeniti u 12 (u našem slučaju se koristila nova promenljiva koje se zove ***idvan***.
U zavisnosti od toga da li je došlo do promašaja ili pogotka odlazi se u različite delove koda. Ukoliko je pogođeno, onda se dekrementira brojač za 1 i upoređuje se sa nulom. Ako je pao na nulu onda se skače u kraj igre i obaveštava se igrač koji je pobedio. Ako nije kraj, vraća se na istog igrača kako bi mu se omogućilo još jednom da igra.

Ukoliko je brodić promašen, onda se brojač ne umanjuje i pogađanje se prebacuje drugom igraču U kodu se primećuje pozivanje procedure ***paralelnaStampa***. Ona predstavlja proceduru za ispis na konzolu.
Ispis se vrši na osnovu setovanih promenljivih gde se na odgovarajuće mesto upisuje blok karakter ili 'x'. Zatim se dodaju brojevi koji označavaju kolonu tabele (brojevi 0-9) na prvo mesto u svakom redu. Onda se pristupa paralelnoj štampi, gde se štampa najpre red iz prve tabele pa ako se naleti na TAB karakter prelazi se na štampanje drugog reda. Ako se u drugom redu naiđe na TAB, upisuje se karakter za prelazak u novi red i postupak se ponavlja dok se ne ispiše sve.
Takođe, postoji procedura ***stampa_blank*** koja samo štampa početne prazne tabele sa uvodnom porukom dobrodošlice.
U kodu postoji I deo koji proverava da li je neka određena koordinata koja je već "pokušana" ponovo uneta, kako se brojač preostalih brodića ne bi iznova dekrementirao I kako neki od igrača ne bi na nesportski način dobio igru.

# Kod

```
INCLUDE Irvine32.inc
INCLUDE macros.inc

BUFFER_SIZE = 209
VALID_SIZE = 64h

.data
buffer BYTE BUFFER_SIZE DUP(?) ; Buffer za ucitavanje iz ulaznog fajla
cnt BYTE ?
filename    BYTE 80 DUP(0)
fileHandle  HANDLE ?

array1 BYTE VALID_SIZE DUP(?) ; Niz sa informacijama o polozaju brodica prvog igraca
array2 BYTE VALID_SIZE DUP(?) ; Niz sa informacijama o polozaju brodica drugog igraca
allowedCharacters BYTE '-1234567890' ; dozvoljeni karakteri u tabeli
i DWORD ?
j DWORD ?
h BYTE '-'

; brojaci za proveru broja unetih brodica
numFive BYTE 0
numFour BYTE 0
numThree BYTE 0
numTwo BYTE 0

; pomocne promenljive
player BYTE 0
distance BYTE 0

endl EQU <0dh,0ah>                          ; end of line sekvenca
message6 BYTE "Igrac 1:", 0 ;Labele za oznaku koji igrac je trenutno na potezu
message7 BYTE "Igrac 2:", 0

message8 BYTE "Dobrodosli u potapanje podmornica! Igrac1 je prvi na potezu.
              Unesite koordinate:", 0

;Poruke o rezultati odigranog poteza i kraju igrice
message BYTE "Promasili ste, sada igra drugi igrac" , 0
messageSize DWORD ($-message)

prompt BYTE "Pogodili ste, igrajte ponovo", 0
promptSize DWORD ($-prompt)

message2 BYTE "Neispravan unos, pokusaj ponovo"
message2Size DWORD ($-message2)

message3 BYTE "Pobedio je Igrac 1, cestitamo!", 0
message3Size DWORD ($-message3)

message4 BYTE "Pobedio je Igrac 2, cestitamo!", 0
message4Size DWORD ($-message4)

message5 LABEL BYTE   ; zaglavlje tabele za podmornice
        BYTE "      Podmornice 1                Podmornice2" , endl
        BYTE "   A B C D E F G H I J      A B C D E F G H I J", endl
message5Size DWORD ($-message5)

;Niz koji omogucava ispis rednog broja reda tabele
```

```
array BYTE '0', '1', '2', '3', '4', '5', '6', '7', '8', '9'
arraySize EQU SIZEOF array

crtice1 BYTE 10 DUP(1 DUP('5'), 10 DUP('-'), 1 DUP('    ')) ;nizovi za stampanje
crtice1Size EQU SIZEOF crtice1

crtice2 BYTE 10 DUP(1 DUP('5'), 10 DUP('-'), 1 DUP('    '))
crtice2Size EQU SIZEOF crtice2

jind WORD 0 ;brojaci za upis rednog broja u nizove crtice1 i crtice2
ind WORD 0


k WORD 0

MAX = 4   ; promenljive za proveru koliko je karaktera uneto za koordinatu brodica
unos BYTE MAX+1 dup(?)

;i BYTE 10 ;pomeraj za trazenje brodica u nizu
idvan BYTE 12 ; pomeraj za stampanje na konzolu

broj BYTE 0 ; broj koji je unet za koordinatu
slovo BYTE 0 ; slovo koje je uneto za koordinatu

brojac1 BYTE 30 ; brojac za 1.brodice
brojac2 BYTE 30 ; brojac za 2.brodice

consoleHandle HANDLE 0      ; handle za standardni izlaz(konzolu)
bytesWritten  DWORD ?       ; broj bajtova koji je ispisan

igra1 BYTE 0 ; promenljive koje odredjuju koji igrac igra (igra1 igra2)
igra2 BYTE 0 ; i da li se desio promasaj ili nije (prom1 prom2)
prom1 BYTE 0
prom2 BYTE 0

.code
paralelnaStampa PROC
        call Clrscr
  ; Get the console output handle:
        INVOKE GetStdHandle, STD_OUTPUT_HANDLE
        mov consoleHandle,eax
  ; Write a string to the console:
        INVOKE WriteConsole,
          consoleHandle,                ; konzolni output handle
          ADDR message5,          ; pokazivac na string
          message5Size,                 ; duzina stringa
          ADDR bytesWritten,     ;vraca broj ispisanih bajtova
          0                                     ; ne koristi se
        mov ecx, arraySize ; u ecx se smesta duzina niza koji odredjuje kolone
        mov ind, 0 ;indeks za upis kolone brojeva u prvu tabelu
        mov jind, 0 ; indeks za upis kolone brojeva u drugu tabelu
        cmp prom1, 0 ; da li je prvi igrac pogodio ili promasio brodic
        jne omas1
        cmp prom2, 0 ; da li je drugi igrac pogodio ili omasio brodic
        jne omas2
        cmp igra1, 0 ; igrac1 je pogodio
        je igrac1
        cmp igra2, 0 ;igrac2  je pogodio
        je igrac2
```

```
igrac1: ; upis blok karaktera u niz i produzetak na dodavanje brojeva u kolonu
        mov al, 254
        mov crtice2[ebx], al
        xor ebx, ebx
        xor eax, eax
        xor edx, edx
        jmp dodajbr1

igrac2: ; upis blok karaktera u niz i produzetak na dodavanje brojeva u kolonu
        mov al, 254
        mov crtice1[ebx], al
        xor ebx, ebx
        xor eax, eax
        xor edx, edx
        jmp dodajbr1

omas1:  ; upis X karaktera zbog omaske i produzetak na dodavanje brojeva u kolonu
        mov al, 'x'
        mov crtice2[ebx], al
        xor ebx, ebx
        xor eax, eax
        xor edx, edx
        jmp dodajbr1

omas2: ; upis X karaktera zbog omaske i produzetak na dodavanje brojeva u kolonu
        mov al, 'x'
        mov crtice1[ebx], al
        xor ebx, ebx
        xor eax, eax
        xor edx, edx

dodajbr1: ;dodatak brojeva u kolonu za prvu tabelu
        mov bx, ind
        mov dl, array[bx]
        mov ax, 12
        mov bl, dl
        mul ind
        mov dl, bl
        mov bx, ax
        mov al, dl
        mov crtice1[bx], dl;
        inc ind
        loop dodajbr1

        xor eax, eax
        xor ebx, ebx
        xor edx, edx
        mov ecx, arraySize

dodajbr2: ;dodatak brojeva u kolonu za drugu tabelicu
        mov bx, jind
        mov dl, array[bx]
        mov ax, 12
        mov bl, dl
        mul jind
        mov dl, bl
        mov bx, ax
        mov al, dl
        mov crtice2[bx], dl;
        inc jind
```

```
        loop dodajbr2

        mov esi, OFFSET crtice2 ;setuje se na pocetak tabele za ispis drugog igraca
        mov ecx, crtice1Size ;ecx uzima vrednost velicine tabela
        xor edx, edx ; ciscenje registara od zaostalih informacija
        xor eax, eax
        xor ebx, ebx

        mov edi, OFFSET crtice1 ;edi se postavlja na pocetak tabele

istampaj: ; stampanje
        mov al, '␣' ; upis razmaka radi preglednosti
        call writechar ;procedura iz Irvin biblioteke za ispis jednog karaktera
        mov al, [edi+edx] ; dodatak prvog clana iz tabele prvog igraca
        call writechar
        cmp al, '␣␣␣␣␣␣␣␣' ; provera da li je poslednji karaktera TAB
                                ; i onda se prelazi na stampanje druge tabele
        je predji ; prelazi se na stampanje druge tabele
        inc edx ; predji na sledeci karakter
        cmp edx, 120 ; provera da li je kraj tabele
        je kraj
        loop istampaj ; povratak na stampanje ako nije kraj tabele

    jmp kraj
predji:
        inc edx ;edx se uvecava da bi se dohvatio prvi naredni karakter iz 1.tabele
        jmp istampaj2 ; prelazi se na stampanje druge tabele
lupiendl: ; stampanje end of line karaktera da se spusti u drugi red
        mov al, 0Dh
        call writechar
        mov al, 0Ah
        call WriteChar
        cmp ecx, 10 ; provera ecx, kako se ne bi stampao dodatni red
        je kraj
        jmp istampaj ; povratak na stampanje prve tabele

istampaj2:
        mov al, '␣' ;ponavlja se isti postupak za stampanje 2. tabele kao i za prvu
        call writechar ; samo se koriste registri ebx i esi
        mov al, [esi+ebx]
        inc ebx
        cmp al, '␣␣␣␣␣␣␣␣'
        call writechar
        je lupiendl ; obavezno spustanje u novi red nakon kraja reda u drugoj tabeli
        jne istampaj2 ; ako nije kraj reda stampaj jos karatkera


kraj: ; kraj procedure paralelna stampa
        ret
paralelnaStampa endp

stampa_blank PROC ; procedura koja predstavlja pocetak igrice gde se
                        ; ispisuju prazne tabele i ispisuje ulazna poruka
        mov edx, OFFSET message8 ; ulazna poruka dobrodoslice
        call WriteString
        call Crlf
  ; Get the console output handle:
        INVOKE GetStdHandle, STD_OUTPUT_HANDLE
        mov consoleHandle,eax
  ; Write a string to the console:
```

```
        INVOKE WriteConsole,
          consoleHandle,                ; handle izlazne konzole
          ADDR message5,          ; pokazivac na string
          message5Size,                 ; duzina stringa
          ADDR bytesWritten,      ; broj ispisanih bajtova
          0                                     ; ne koristi se
        mov ecx, arraySize ; duzina niza kolone sa brojevima u tabeli
        mov ind, 0 ;promenljive koje sluze za iteraciju kroz array
        mov jind, 0
        xor ebx, ebx
        xor edx, edx

dodajbr1: ;dodatak brojeva u prvu tabelu
        mov bx, ind
        mov dl, array[bx]
        mov ax, 12
        mov bl, dl
        mul ind
        mov dl, bl
        mov bx, ax
        mov al, dl
        mov crtice1[bx], dl;
        inc ind
        loop dodajbr1

        xor eax, eax ;reset vrednosti
        xor ebx, ebx
        xor edx, edx
        mov ecx, arraySize

dodajbr2: ;dodatak brojeva u drugu tabelu
        mov bx, jind
        mov dl, array[bx]
        mov ax, 12
        mov bl, dl
        mul jind
        mov dl, bl
        mov bx, ax
        mov al, dl
        mov crtice2[bx], dl;
        inc jind
        loop dodajbr2

        mov esi, OFFSET crtice2 ;podesavanje parametara za paralelnu stampu
        mov ecx, crtice1Size
        xor edx, edx
        xor eax, eax
        xor ebx, ebx

        mov edi, OFFSET crtice1

istampaj: ;isti postupak kao u procedtu paralelnaStampa
        mov al, '␣'
        call writechar
        mov al, [edi+edx]
        call writechar
        cmp al, '␣␣␣␣␣␣␣'
        je predji
        inc edx
        cmp edx, 120
```

```
        je kraj
        loop istampaj

    jmp kraj
predji:
        inc edx
        jmp istampaj2
lupiendl:
        mov al, 0Dh
        call writechar
        mov al, 0Ah
        call WriteChar
        cmp ecx, 10
        je kraj
        jmp istampaj

istampaj2:
        mov al, ' '
        call writechar
        mov al, [esi+ebx]
        inc ebx
        cmp al, '        '
        call writechar
        je lupiendl
        jne istampaj2


kraj: ;kraj procedute stampa_blank
        ret
stampa_blank endp

main PROC
; Unos imena fajlova igraca i provera ispravnosti fajlova

; Unos za prvog igraca
        mWrite "Unesite ime fajla za 1. igraca: "
        mov     edx,OFFSET filename
        mov     ecx,SIZEOF filename
        call    ReadString
        jmp openFile

secondPlayerFile:
; Unos za drugog igraca
        mWrite "Unesite ime fajla za 2. igraca: "
        mov     edx,OFFSET filename
        mov     ecx,SIZEOF filename
        call    ReadString

openFile:
; Otvaranje fajla
        mov     edx,OFFSET filename
        call    OpenInputFile
        mov     fileHandle,eax

; Provera greski pRi ucitavanju fajla - da li postoji fajl
        cmp     eax,INVALID_HANDLE_VALUE
        jne     file_ok
        mWrite <"Fajl ne moze da se otvori!",0dh,0ah>
        jmp     quit
file_ok:
```

```
; Ucitavanje ulaznog fajla u buffer
        mov     edx,OFFSET buffer
        mov     ecx,BUFFER_SIZE
        call    ReadFromFile
        jnc     check_buffer_size                       ; greska pri citanju?
        mWrite <"Greska prilikom citanja fajla!",0dh,0ah>
        call    WriteWindowsMsg
        jmp     close_file

check_buffer_size:
        cmp     eax,BUFFER_SIZE                 ; da li je bafer dovoljno veliki?
        jb      close_file
        mWrite <"Fajl nije odgovarajuce duzine!",0dh,0ah>
        jmp     quit

close_file:
        mov     eax,fileHandle
        call    CloseFile

; Provera da li je ulazni fajl u redu
        mov esi, offset buffer
        mov ebx, 19
        mov ecx,9
inputLoop:
        mov al, [esi + ebx]
        mov ah, [esi + ebx + 1]
        cmp al, 0dh
        jne inputFileError
        cmp ah, 0ah
        jne inputFileError
        add ebx,21
        loop inputLoop

        jmp inputFileOk
inputFileError:
        mWrite <"Greska u ulaznom fajlu!",0dh,0ah,0dh,0ah>
        jmp quit

inputFileOk:
        cmp player, 0
        jne secondPlayer

        mov edi, offset array1
        jmp removal

secondPlayer:
        mov edi, offset array2

; Ulazni fajl je u redu
removal:
; Otklanjanje blanko znaka iz ulaznog bafera
        mov esi, offset buffer
startOfRemoval:
        mov al, [esi]
        inc esi
        cmp al, 0
        je endOfRemoval
        cmp al, 32
        je startOfRemoval
```

```
        cmp al, 0dh
        je startOfRemoval
        cmp al, 0ah
        je startOfRemoval
        mov [edi],al
        inc edi
        jmp startOfRemoval

endOfRemoval:
        cmp player,0
        jne secondPlayerRemoval
        mov eax, lengthof array1
        jmp continueRemoval

secondPlayerRemoval:
        mov eax, lengthof array2

continueRemoval:
        cmp eax, VALID_SIZE
        jne inputFileError

        cmp player, 0
        jne secondPlayerIC

        mov esi, offset array1
        jmp illegalCharacters

secondPlayerIC:
        mov esi, offset array2

; Provera postovanja pravila
; Provera da li su korisceni samo dozvoljeni znaci (-0123456789)
illegalCharacters:
        mov cnt, VALID_SIZE-1
searchLoop:
        mov edi, offset allowedCharacters
        mov ecx, 11
        mov eax, [esi]
        repne scasb
        jne invalidCharacterFound
        inc esi
        dec cnt
        je continue
        jmp searchLoop

invalidCharacterFound:
        mWrite <"Nedozvoljeni znaci u ulaznom fajlu!",0dh,0ah,0dh,0ah>
        jmp quit

continue:
        cmp player, 0
        jne secondPlayerPC

        mov esi, offset array1
        jmp placementCheck

secondPlayerPC:
        mov esi, offset array2

placementCheck:
```

```
        mov numTwo, 0
        mov numThree, 0
        mov numFour, 0
        mov numFive, 0

; Provera rasporeda brodica
; Proverava da li se brodici dodiruju, ako se dodiruju - krsenje pravila!
        mov ecx, 10
        mov i,0
        mov j,0
cheatingLoop:
        cmp i, ecx ; ako je i = 10 uvecava se j
        je incJ
        mov ebx, i
        imul eax, j, 10
        add ebx, eax
        mov al, [esi+ebx]
        mov dl, 45 ; dl = -
        cmp al, dl
        je hyphen
        cmp j, 0
        je topRow
        cmp j, 9
        je bottomRow
        cmp i, 0
        je leftColumn
        cmp i, 9
        je rightColumn

        cmp al, [esi+ebx+1]
        je checkLeft
        cmp dl, [esi+ebx+1]
        jne cheatingFound
        cmp al, [esi+ebx-1]
        je horizontalShip
        cmp dl, [esi+ebx-1]
        jne cheatingFound
        jmp verticalShip

checkLeft:
        cmp al, [esi+ebx-1]
        je horizontalShip
        cmp dl, [esi+ebx-1]
        jne cheatingFound
        jmp horizontalShip

verticalShip:
        cmp al, [esi+ebx-10]
        je checkBottom
        cmp dl, [esi+ebx-10]
        jne cheatingFound
        cmp al, [esi+ebx+10]
        jne cheatingFound
        jmp diagonal

checkBottom:
        cmp al, [esi+ebx+10]
        je diagonal
        cmp dl, [esi+ebx+10]
        jne cheatingFound
```

```
        jmp diagonal

horizontalShip:
        cmp dl, [esi+ebx+10]
        jne cheatingFound
        cmp dl, [esi+ebx-10]
        jne cheatingFound
diagonal:
        cmp dl, [esi+ebx-9]
        jne cheatingFound
        cmp dl, [esi+ebx-11]
        jne cheatingFound
        cmp dl, [esi+ebx+9]
        jne cheatingFound
        cmp dl, [esi+ebx+11]
        jne cheatingFound
        inc i
        jmp cheatingLoop

hyphen:
        inc i
        jmp cheatingLoop

incJ:
        cmp j,9
        je notCheating
        mov i,0
        inc j
        jmp cheatingLoop

topRow:
        cmp i,0
        je topLeftCorner
        cmp i,9
        je topRightCorner
        cmp al, [esi+ebx+1]
        je checkLeftTR
        cmp dl, [esi+ebx+1]
        jne cheatingFound
        cmp al, [esi+ebx-1]
        je horizontalShipTR
        cmp dl, [esi+ebx-1]
        jne cheatingFound
        jmp verticalShipTR

checkLeftTR:
        cmp al, [esi+ebx-1]
        je horizontalShipTR
        cmp dl, [esi+ebx-1]
        jne cheatingFound
        jmp horizontalShipTR

verticalShipTR:
        cmp al, [esi+ebx+10]
        je diagonalTR
        jne cheatingFound

horizontalShipTR:
        cmp dl, [esi+ebx+10]
        jne cheatingFound
```

```
diagonalTR:
        cmp dl, [esi+ebx+9]
        jne cheatingFound
        cmp dl, [esi+ebx+11]
        jne cheatingFound
        inc i
        jmp cheatingLoop

topLeftCorner: ;TLC
        cmp al, [esi+ebx+1]
        jne verticalShipTLC
horizontalShipTLC:
        cmp dl, [esi+ebx+10] ;sa hyphen poredimo
        jne cheatingFound
        jmp diagonalTLC
verticalShipTLC:
        cmp al, [esi+ebx+10]
        jne cheatingFound
        cmp dl, [esi+ebx+1]
        jne cheatingFound
diagonalTLC:
        cmp dl, [esi+ebx+11]
        jne cheatingFound
        inc i
        jmp cheatingLoop

topRightCorner: ;TRC
        cmp al, [esi+ebx-1]
        jne verticalShipTRC
horizontalShipTRC:
        cmp dl, [esi+ebx+10] ;sa hyphen poredimo
        jne cheatingFound
        jmp diagonalTRC
verticalShipTRC:
        cmp al, [esi+ebx+10]
        jne cheatingFound
        cmp dl, [esi+ebx-1]
        jne cheatingFound
diagonalTRC:
        cmp dl, [esi+ebx+9]
        jne cheatingFound
        inc i
        jmp cheatingLoop

leftColumn:
        cmp al, [esi+ebx-10]
        je checkDownLC
        cmp dl, [esi+ebx-10]
        jne cheatingFound
        cmp al, [esi+ebx+10]
        je verticalShipLC
        cmp dl, [esi+ebx+10]
        jne cheatingFound
        jmp horizontalShipLC

checkDownLC:
        cmp al, [esi+ebx+10]
        je verticalShipLC
        cmp dl, [esi+ebx+10]
        jne cheatingFound
```

```
        jmp verticalShipLC

horizontalShipLC:
        cmp al, [esi+ebx+1]
        je diagonalLC
        jne cheatingFound

verticalShipLC:
        cmp dl, [esi+ebx+1]
        jne cheatingFound
diagonalLC:
        cmp dl, [esi+ebx-9]
        jne cheatingFound
        cmp dl, [esi+ebx+11]
        jne cheatingFound
        inc i
        jmp cheatingLoop

rightColumn:
        cmp al, [esi+ebx-10]
        je checkDownRC
        cmp dl, [esi+ebx-10]
        jne cheatingFound
        cmp al, [esi+ebx+10]
        je verticalShipRC
        cmp dl, [esi+ebx+10]
        jne cheatingFound
        jmp horizontalShipRC

checkDownRC:
        cmp al, [esi+ebx+10]
        je verticalShipRC
        cmp dl, [esi+ebx+10]
        jne cheatingFound
        jmp verticalShipRC

horizontalShipRC:
        cmp al, [esi+ebx-1]
        je diagonalRC
        jne cheatingFound

verticalShipRC:
        cmp dl, [esi+ebx-1]
        jne cheatingFound
diagonalRC:
        cmp dl, [esi+ebx+9]
        jne cheatingFound
        cmp dl, [esi+ebx-11]
        jne cheatingFound
        inc i
        jmp cheatingLoop

bottomRow:
        cmp i,0
        je bottomLeftCorner
        cmp i,9
        je bottomRightCorner
        cmp al, [esi+ebx+1]
        je checkLeftBR
        cmp dl, [esi+ebx+1]
```

```
        jne cheatingFound
        cmp al, [esi+ebx-1]
        je horizontalShipBR
        cmp dl, [esi+ebx-1]
        jne cheatingFound
        jmp verticalShipBR

checkLeftBR:
        cmp al, [esi+ebx-1]
        je horizontalShipBR
        cmp dl, [esi+ebx-1]
        jne cheatingFound
        jmp horizontalShipBR

verticalShipBR:
        cmp al, [esi+ebx-10]
        je diagonalBR
        jne cheatingFound

horizontalShipBR:
        cmp dl, [esi+ebx-10]
        jne cheatingFound
diagonalBR:
        cmp dl, [esi+ebx-9]
        jne cheatingFound
        cmp dl, [esi+ebx-11]
        jne cheatingFound
        inc i
        jmp cheatingLoop

bottomLeftCorner: ;BLC
        cmp al, [esi+ebx+1]
        jne verticalShipBLC
horizontalShipBLC:
        cmp dl, [esi+ebx-10] ;sa hyphen poredimo
        jne cheatingFound
        jmp diagonalBLC
verticalShipBLC:
        cmp al, [esi+ebx-10]
        jne cheatingFound
        cmp dl, [esi+ebx+1]
        jne cheatingFound
diagonalBLC:
        cmp dl, [esi+ebx-9]
        jne cheatingFound
        inc i
        jmp cheatingLoop

bottomRightCorner: ;BRC
        cmp al, [esi+ebx-1]
        jne verticalShipBRC
horizontalShipBRC:
        cmp dl, [esi+ebx-10] ;sa hyphen poredimo
        jne cheatingFound
        jmp diagonalBRC
verticalShipBRC:
        cmp al, [esi+ebx-10]
        jne cheatingFound
        cmp dl, [esi+ebx-1]
        jne cheatingFound
```

```
diagonalBRC:
        cmp dl, [esi+ebx-11]
        jne cheatingFound
        inc i
        jmp cheatingLoop

jmp notCheating

; Pronadjeno krsenje pravila
cheatingFound:
        mWrite <"Varas, varalice!",0dh,0ah,0dh,0ah>
        jmp quit

notCheating:

        cmp player, 0
        jne secondPlayerNum

        mov edi, offset array1
        jmp numCheck

secondPlayerNum:
        mov edi, offset array2

numCheck:
; Provera broja brodova
; Proverava da li su koriscene sve cifre od 0-9, ako jesu onda proverava
; da li odgovarajuci broj brodova.
        mov esi, offset allowedCharacters
        mov ebx, 1 ; polazimo od 1
countingLoop:
        cmp ebx, 11
        je finishedCounting
        mov ecx,0
        mov cnt,0
        mov al, [esi+ebx]
        mov distance, -100
arrayIteration:
        cmp ecx, VALID_SIZE ; uslov izlaska iz petlje
        je endOfString
        mov ah, [edi+ecx]
        inc ecx
        inc distance
        cmp al,ah
        je incCnt
        jmp arrayIteration

incCnt:
        inc cnt
        cmp distance, 10
        jg wrongNumber
        mov distance, 0
        jmp arrayIteration

endOfString:
        inc ebx
        cmp cnt,2
        je incTwo
        cmp cnt,3
        je incThree
```

```
        cmp cnt,4
        je incFour
        cmp cnt,5
        je incFive
        jmp wrongNumber

incTwo:
        inc numTwo
        jmp countingLoop

incThree:
        inc numThree
        jmp countingLoop

incFour:
        inc numFour
        jmp countingLoop

incFive:
        inc numFive
        jmp countingLoop

finishedCounting:
        cmp numFive, 1
        jne wrongNumber
        cmp numFour, 2
        jne wrongNumber
        cmp numThree, 3
        jne wrongNumber
        cmp numTwo,4
        jne wrongNumber
        jmp allIsWell

wrongNumber:
        mWrite <"Niste uneli odgovarajuce brodice!",0dh,0ah,0dh,0ah>
        jmp quit

allIsWell:
        inc player
        cmp player,1
        je secondPlayerFile

        mov i, 10
        xor eax, eax

        call stampa_blank ;ispis pocetnog ekrana
igrac1:
mov edx, OFFSET message6 ;dodatak labele da bi se znalo koji je igrac na potezu
call WriteString
call Crlf
unesi1:
        mov edx, OFFSET unos ; ucitavanje koordinate na koju sumnjamo da krije brodic
        mov ecx, MAX
        call ReadString
ucitaj1:
        xor ecx, ecx
        cmp eax, 2 ; poredjenje duzine upisanog podatka sa 2,
                        ; kako bi znali da li je upisana odgovarajuca vrednost
        jne greska1 ; ako nije prijavljuje se greska
        mov ah, [edx] ;pravljenje pomeraja od unetog slova i broja
```

```
        inc edx
        cmp ah, 'A' ;poredjenje da li je slovo u odgovarajucem opsegu,
                                ;ako nije prijavi gresku i zatrazi ponovni upis
        jl greska1
        cmp ah, 'J'
        jg greska1
        sub ah, 64 ; pravljenje decimalne vrednosti od slova(preko ASCII koda)
        mov al, [edx] ;ucitavanje broja
        cmp al, '0' ;provera da li je broj u odgovarajucem opsegu,
                                ; ako nije prijavi gresku
        jl greska1
        cmp al, '9'
        jg greska1
        sub al, 48 ; napravi decimalnu vrednost od karaktera koji predstavlja broj
        mov broj, al
        mov slovo, ah
        mov al, broj
        mul i ;mnozenje sa 10 kako bi se dobio odgovarajuci linearni pomeraj
        add cl, slovo
        mov ah, 0
        add ax, cx ;krajnji pomeraj


        xor ebx, ebx ;ciscenje registara od potencijalnih zaostalih vrednosti
        xor edx, edx
        mov bx, ax
        dec ax ; odgovarajuci pomeraj mora da se umanji za 1 zbog nacina indeksiranja
        mov bx, ax
        cmp array2[ebx], '-' ;kako bismo znali da li se desio pogodak
        je promasaj1 ;ako nije skoci u promasaj

        ;prebaci na pomeraj za stampu
        xor ecx, ecx
        xor eax, eax
        mov al, broj
        mul idvan ;pomeraj za stampu - izlazna tabela ima 12 karaktera po redu, ne 10
        add cl, slovo ;otud je idvan=12
        add ax, cx
        xor ebx, ebx
        mov bx, ax

        cmp crtice2[ebx], '-'
        jne vecIgrano1 ; ako smo vec odigrali neki potez da se ne ponavljamo

        mov igra1, 0 ; setuj promenljivu igra1 na 0
                                ; igra prvi igrac ponovo jer se desio pogodak
        xor eax, eax
        sub brojac2, 1 ; umanji borjac brodica2 jer se desio pogodak
        cmp brojac2, 0
        je kraj1 ;uporedi brojac sa nulom da znamo da li je kraj igre i skoci u kraj1
        xor eax, eax
        mov prom1, 0 ; setuj flegove za promasaj na 0
        mov prom2, 0
        call paralelnaStampa ;pozovi ispis tabela
        mov edx, OFFSET prompt ;prikazi poruku odobravanja
        call WriteString
        call Crlf
        jmp igrac1 ;vrati se na pocetak petlje jer opet igra prvi igrac

greska1:
```

```
        mov edx, OFFSET message2 ;izbaci poruku da je doslo do pogresnog unosa
        call WriteString
        call Crlf
        jmp unesi1 ;vrati se na ponovni unos
vecIgrano1:
        mWrite <"Vec ste pokusali ovu koordinatu, unesite neku drugu", 0Dh, 0Ah>
        jmp igrac1


promasaj1:
;prebaci na pomeraj za stampu
        xor ecx, ecx
        xor eax, eax
        mov al, broj
        mul idvan
        add cl, slovo
        add ax, cx
        xor ebx, ebx
        mov bx, ax
        cmp crtice2[ebx], '-'
        jne vecIgrano1 ;ako smo odigrali neki potez da se ne ponavljamo
        mov prom2, 0 ;setuj flegove za promasaj 1. igraca i omoguci igracu 2 da igra
        mov igra2, 0 ; komandom mov igra2, 0 i mov igra1, 0
        mov prom1, 1
        mov igra1, 1
        call paralelnaStampa ; pozovi stampu tabele
        mov edx, OFFSET message ;prikazi poruku neodobravanja
        call WriteString
        call Crlf ;procedura iz IRVIN biblioteke za dodatak novog reda
        jmp igrac2 ;predji na igraca 2


igrac2:
mov edx, OFFSET message7 ;prikazi poruku da igra drugi igrac
call WriteString
call Crlf
unesi2: ;ista provera kao za igraca 1
        mov edx, OFFSET unos
        mov ecx, MAX
        call ReadString ;unos zeljene koordinate
ucitaj2:
        cmp eax, 2 ;provera da li je unet string odgovarajuce duzine
        jne greska2 ; ako nije skoci u gresku
        xor ecx, ecx
        mov ah, [edx] ;dohvatanje prvog karaktera u nizu
        inc edx
        cmp ah, 'A' ; provera da li je prvi karakter u dozvoljenom opsegu
        jl greska2
        cmp ah, 'J'
        jg greska2 ; ako nije prijavi gresku
        sub ah, 64
        mov al, [edx] ;dohvatanje novog karaktera koji treba da predstavlja broj reda
        cmp al, '0'
        jl greska2 ;provera da li je u odgovarajucem opsegu
        cmp al, '9'
        jg greska2 ;ako nije prijavi gresku
        sub al, 48
        mov broj, al
        mov slovo, ah
        mov al, broj
```

```
        mul i ;pravljenje linearnog pomeraja za pretragu po ulaznom nizu
        mov ah, 0
        add cl, slovo
        add ax, cx

        dec ax
        xor ebx, ebx
        xor edx, edx
        mov bx, ax
        cmp array1[ebx], '-' ;da li je brodic pogodjen
        je promasaj2 ; ako nije idi u promasaj
        mov igra2, 0 ;setuj odgovarajuce flegove kako bi igrac2 ponovo igrao

        ;prebaci na pomeraj za stampu
        xor ecx, ecx
        xor eax, eax
        mov al, broj
        mul idvan ;mnozenje sa 12 jer je pomeraj drugaciji u tabeli za prikaz
        add cl, slovo
        add ax, cx
        xor ebx, ebx
        mov bx, ax

        cmp crtice1[ebx], '-'
        jne vecIgrano2 ; ako smo odigrali potez da se ne ponavljamo

        xor eax, eax
        sub brojac1, 1 ; kako bisimo znali da kad je kraj
        cmp brojac1, 0
        je kraj2 ;skoci u kraj2 ako nema vise brodica
        xor eax, eax
        mov prom2, 0 ;setuj flegove kako bi igrac2 mogao ponovo da igra
        mov prom1, 0
        call paralelnaStampa ;prikazi tabele
        mov edx, OFFSET prompt ;izbaci poruku odobravanja
        call WriteString
        call Crlf
        jmp igrac2 ; vrati se na pocetak kako bi igrac2 ponovo igrao

greska2:
        mov edx, OFFSET message2 ;prikazi poruku greske
        call WriteString
        call Crlf
        jmp unesi2 ;vrati se na ponovni unos

vecIgrano2:
        mWrite <"Vec ste pokusali ovu koordinatu, unesite neku drugu", 0Dh, 0Ah>
        jmp igrac2


promasaj2:
;prebaci na pomeraj za stampu
        xor ecx, ecx
        xor eax, eax
        mov al, broj
        mul idvan
        add cl, slovo
        add ax, cx
        xor ebx, ebx
        mov bx, ax
```

```
        cmp crtice1[ebx], '-'
        jne vecIgrano2
        mov prom1, 0 ; igrac2 je promasio - setuj flag
        mov igra1, 0 ; i da sada igra igrac 1
        mov prom2, 1
        mov igra2, 1
        call paralelnaStampa ; pozovi prikaz tabela na konzoli
        mov edx, OFFSET message ;prikazi poruku neodobravanja
        call WriteString
        call Crlf
        jmp igrac1

kraj1: ;kraj u kome je igrac 1 pobednik
        call paralelnaStampa ;stampa personalizovane poruke
        mov edx, OFFSET message3
        call WriteString
        call Crlf
        INVOKE ExitProcess,0

kraj2: ;kraj u kome je igrac 2 pobednik
        call paralelnaStampa
        mov edx, OFFSET message4 ;stampa personalizovane poruke
        call WriteString
        call Crlf
        INVOKE ExitProcess,0

quit:
        exit
main ENDP

END main
```