

UNIVERZITET U BEOGRADU ELEKTROTEHNIČKI FAKULTET

Katedra za elektroniku

Predmet: Računarska elektronika



Projekat 19: Igra memorije

Projekat radili:

Prezime	Ime	Broj indeksa
Mijušković	Filip	2013/0156
Lilić	Dimitrije	2013/0352

Predmetni profesor: prof. Dr Milan Prokin

Predmetni asistent: asis. Aleksandra Lekić

Sadržaj

Tekst zadatka	3
Opis projektnog koda	4
irvine32.inc.....	4
.data	4
.code.....	5
glavni program i ostale procedure.....	5
Izgled igrice	6
Projektni kod	7

Tekst projektnog zadatka

Projekat 19 - Igra memorije

Potrebno je napraviti igru memorije koja ima sledeće funkcionalnosti: igrač na početku bira da li će tabla biti 4x5 ili 5x4. Zatim se nasumično postavljaju karakteri (poželjno je da to budu brojevi) i prikazuju se tabla sa raspoređenim znakovima. Takva tabela ostaje vidljiva 15 sekundi i onda se prikazuje sakrivena tabela. Korisnik unosi željenu koordinatu (npr. AA, ukoliko želi da proveri prvu koordinatu) a zatim i drugu i ukoliko je par pronađen on se zadržava otkriven na tabli a ukoliko nije, sakriva se nakon 5 sekundi. Igra je gotova kada se svi parovi pogode.

	A	B	C	D	E
A	1	3	8	7	5
B	2	0	6	4	1
C	5	7	8	9	0
D	4	9	2	3	6

Slka 19.1 4x5 tabela sa otkrivenim brojevima

	A	B	C	D	E
A	♥	♥	♥	♥	♥
B	♥	♥	♥	♥	♥
C	♥	♥	♥	♥	♥
D	♥	♥	♥	♥	♥

Slka 19.2 4x5 tabela sa skrivenim znakovima

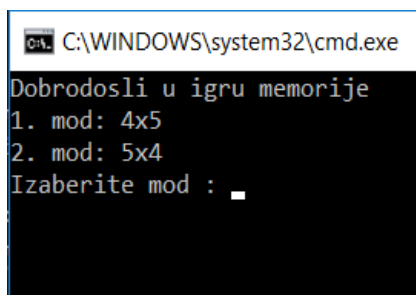
Napomena: Zbog toga što znak srce ne postoji u ASCII kodnoj tabeli, umesto njega koriscen je znak 177 koji podseca na poledjinu karte.

Opis projektnog koda

- ❖ Na početku smo uključili biblioteku **irvine32.inc** kao i **macros.inc** i iz njih smo koristili sledeće funkcije:
 - *Randomize* : Postavlja novi seed u generatoru nasumičnog broja sa trenutnim vremenom u hiljaditim delovima sekunde;
 - *RandomRange* : Generiše pseudo-nasumični neoznačeni 32-bitni integer u opsegu od 0 do n-1, gde je n broj koji mi funkciji prosleđujemo kao parametar;
 - *Clrscr* : Briše ekran tako što upisuje sve blanko karaktere na svim pozicijama;
 - *Clrf* : Upisuje sekvencu (0Dh,0Ah) na standardni izlaz – novi red;
 - *ReadChar* : Čita samo jedan karakter sa standardnog ulaza i vraća znak u AL registar;
 - *WriteChar* : Ispisuje jedan karakter iz AL registra na standardni izlaz;
 - *Delay* : Pauzira trenutni process za zadati broj milisekundi;
 - *ReadString* : Čita string od ECX nenultih karaktera sa standardnog ulaza, i zasutavlja se kada se pritisne Enter;
 - *WriteDec* : Ispisuje neoznčeni 32-bitni decimalni broj na standardni izlaz u decimalnom format;
 - *mWrite* : ova makro funkcija ispisuje string koji joj se prosledi na izlaz bez dodavanja NULL karaktera na kraj stringa;
- ❖ **.data** sadrži inicijalizaciju svih promenljivih koje su potrebne za projekat. Kao i string u kojem su sadžana slova za ispis od A do E, i matricu parova brojeva koji se pogadjaju od 0 do 9.
- ❖ **.code** sekcija
 - Glavni program - main PROC startuje odmah na početku, u kojoj se zatim poziva randarray PROC u kojoj je potrebno odraditi mešanje brojeva/karata 20 puta kako bismo dobili nasumičan raspored karata. Zatim se ispisuje na standardnom izlazu tekst koji nudi da se izabere mod 1 ili mod 2 koji predstavljaju rasporede karata 4x5 ili 5x4. Zatim se vrši komparacija unetog broja i skače na delove koda koji će formirati tabele u zavisnosti od toga koji je mod izabran. Nakon toga se prikazuju svi promešani brojevi 15 sekundi kao što je zahtevano u

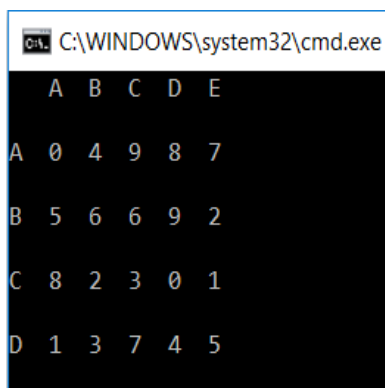
projektu. Nakon toga je potrebno iscrtati ponovo ekran koji je sada sa zatvorenim brojevima/kartama. Ovo crtanje se vrši red po red. Nakon toga se započinje igra, gde je potrebno uneti koordinate koje će biti proveravane da li su unete ispravno i u slučaju neispravnog unosa se ispisuje koja je greška u pitanju, da li je nedozvoljena koordinata ili je već uneta/pogođena koordinata. Nakon unošenja prve dve koordinate, na ekranu se ispisuje rezultat koji je trenutno aktuelan. U slučaju da igrač ne pogodi dva broja, omogućen je prikaz tih brojeva narednih 5 sekundi. Kada je igra završena postavlja se pitanje da li igrač želi da igra ponovo, ako odgovori sa D nova partija se startuje, a ako odgovori sa N onda smo došli do kraja igre i igra se gasi.

Izgled igrice



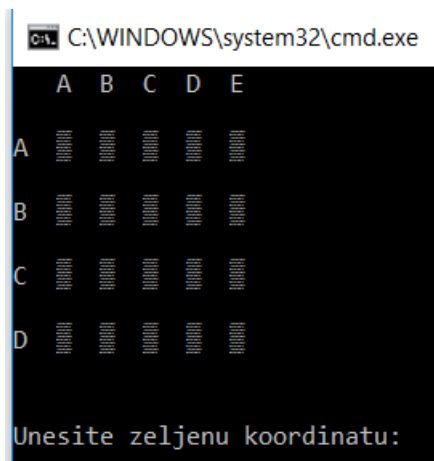
```
C:\WINDOWS\system32\cmd.exe
Dobrodosli u igru memorije
1. mod: 4x5
2. mod: 5x4
Izaberite mod : _
```

Slika 1 – Početni ekran

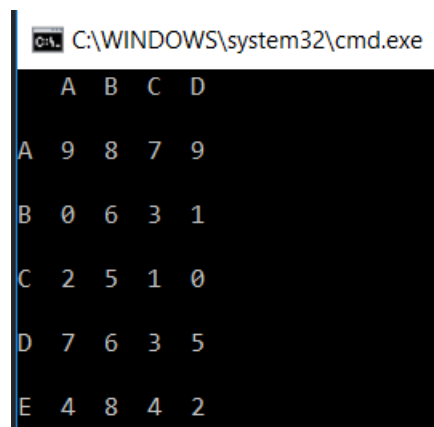


```
C:\WINDOWS\system32\cmd.exe
  A  B  C  D  E
A  0  4  9  8  7
B  5  6  6  9  2
C  8  2  3  0  1
D  1  3  7  4  5
```

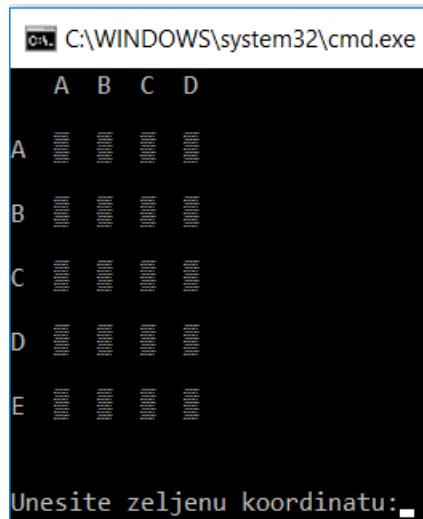
Slika 2 – prvih 15 sek otvorene igrice dimenzija 4x5



Slika 3 – nastavak igranja igrice dimenzija 4x5



Slika 4 – prvih 15 sek otvorene igrice dimenzija 5x4



Slika 5 – nastavak igranja igrice dimenzija 5x4

Projektni kod

```
TITLE Program Template(template.asm)
```

```
INCLUDE Irvine32.inc
```

```
INCLUDE macros.inc
```

```
.data
```

```
rply byte 0
```

```
md dword ?
```

```
score1 dword 0
```

```
randval dword 0
```

```
cor2 dword 0
```

```
cor1 dword 0
```

```
actual byte 20 dup(0)
```

```
coor dword ?
```

```
temp dword ?
```

```
letter byte 32, 65, 66, 67, 68, 69
```

```
arraych byte 49, 50, 51, 52, 53,
```

```
49, 50, 51, 52, 53,
```

```
54, 55, 56, 57, 48,
```

```
54, 55, 56, 57, 48
```

```
select byte 0
```

```
input1 byte 128 dup(0)
```

```
input2 byte 128 dup(0)
```

```
.code
```

```
main PROC
```

```
call randarray
_again :
call clrscr
mwrite "Dobro dosli u igru memorije"
call crlf
mwrite "1. mod: 4x5"
call crlf
mwrite "2. mod: 5x4"
call crlf
_begin :
mwrite "Izaberite mod : "
call readchar
mov select, al
call writechar
call crlf
cmp al, 49
je _4x5
cmp al, 50
je _5x4
```

```
mwrite "Nepostojeci mod!"
mov al, 7
call writechar
call crlf
jmp _begin
```

_5x4 :

```
mov md, 2
```

```
mov eax, 500
call delay
call clrscr
mov ebx, 0
```

_prvi_prikaz:

```
mov actual[ebx], 1
inc ebx
cmp ebx, 20
jne _prvi_prikaz
call screen_5x4
mov eax, 15000
call delay
mov ebx, 0
```


_završen_prvi_prikaz :

```
mov actual[ebx], 0
inc ebx
cmp ebx, 20
jne _završen_prvi_prikaz
mov eax, 500
call delay
call clrscr
call screen_5x4
call crlf
mwrite "Unesite zeljenu koordinatu:"
```

jmp _startup2

_4x5 :

```
mov md, 1
```

```
mov eax, 500
call delay
call clrscr
mov ebx, 0
```

_prvi_prikaz2:

```
mov actual[ebx], 1
inc ebx
cmp ebx, 20
jne _prvi_prikaz2
call screen_5x4
mov eax, 15000
call delay
mov ebx, 0
```

_završen_prvi_prikaz2 :

```
mov actual[ebx], 0
inc ebx
cmp ebx, 20
jne _završen_prvi_prikaz2
mov eax, 500
call delay
call clrscr
call screen_5x4
```

```

call crlf
mwrite "Unesite zeljenu koordinatu:"

_startup2 :

jmp _startup

_inval1 :

mwrite "Nije validan karakter!"
call crlf
mov al, 7
call writechar
call crlf
mwrite "Unesite zeljenu koordinatu:"
jmp _startup

_ara2 :

mwrite "Nije validan karakter!"
call crlf
mov al, 7
call writechar
call crlf
mwrite "Unesite zeljenu koordinatu:"

_startup :

mov ecx, sizeof input1
mov edx, offset input1
call readstring
cmp input1[0], 0
je _ara2
cmp input1[1], 0
je _ara2
cmp input1[2], 0
jne _ara2

mov esi, 0
mov edi, 0

cmp input1[edi], 'a'
je _A
cmp input1[edi], 'A'
je _A

```

```
cmp input1[edi], 'b'
je _B
cmp input1[edi], 'B'
je _B
cmp input1[edi], 'c'
je _C
cmp input1[edi], 'C'
je _C
cmp input1[edi], 'd'
je _D
cmp input1[edi], 'D'
je _D
cmp md, 2
je _greska
cmp input1[edi], 'e'
je _E
cmp input1[edi], 'E'
je _E
```

```
_greska :
jmp _ara2
```

```
_A :
add esi, 0
jmp _cont
```

```
_B :
add esi, 1
jmp _cont
```

```
_C :
add esi, 2
jmp _cont
```

```
_D :
add esi, 3
jmp _cont
```

```
_E :
add esi, 4
jmp _cont
```

```
_cont :
mov edi, 1
```

```
cmp input1[edi], 'a'
je _A1
cmp input1[edi], 'A'
je _A1
cmp input1[edi], 'b'
je _B1
cmp input1[edi], 'B'
je _B1
cmp input1[edi], 'c'
je _C1
cmp input1[edi], 'C'
je _C1
cmp input1[edi], 'd'
je _D1
cmp input1[edi], 'D'
je _D1
cmp md, 1
je _greska1
cmp input1[edi], 'e'
je _E1
cmp input1[edi], 'E'
je _E1
```

```
_greska1 :
jmp _invalid
```

```
_A1 :
```

```
add esi, 0
jmp _final1
```

```
_B1 :
```

```
add esi, 4
add esi, 2
sub esi, md
jmp _final1
```

```
_C1 :
```

```
add esi, 8
add esi, 2
sub esi, md
add esi, 2
sub esi, md
jmp _final1
```

```

_D1 :

add esi, 12
add esi, 2
sub esi, md
add esi, 2
sub esi, md
add esi, 2
sub esi, md
jmp _final1

_E1 :

add esi, 16
jmp _final1

_final1 :

mov cor1, esi
mov eax, cor1

cmp actual[esi], 1
je _err
mov actual[esi], 1
call screen_5x4
call crlf
mwrite "Unesite zeljenu koordinatu:"
jmp _secondary

_err :

mwrite "Vec ste uneli ovu koordinatu!"
mov al, 7
call writechar
call crlf
mwrite "Unesite zeljenu koordinatu:"
jmp _startup

_inval2 :

mwrite "Nije validan karakter!"
mov al, 7
call writechar
call crlf
mwrite "Unesite zeljenu koordinatu:"

```

_secondary :

```
mov edx, offset input2
mov ecx, sizeof input2 - 1
```

```
call readstring
```

```
cmp input2[0], 0
je _inval2
cmp input2[1], 0
je _inval2
cmp input2[2], 0
jne _inval2
```

```
mov esi, 0
mov edi, 0
```

```
cmp input2[edi], 'a'
je _A2
cmp input2[edi], 'A'
je _A2
cmp input2[edi], 'b'
je _B2
cmp input2[edi], 'B'
je _B2
cmp input2[edi], 'c'
je _C2
cmp input2[edi], 'C'
je _C2
cmp input2[edi], 'd'
je _D2
cmp input2[edi], 'D'
je _D2
cmp md, 2
je _greska2
cmp input2[edi], 'e'
je _E2
cmp input2[edi], 'E'
je _E2
```

```
_greska2 :
jmp _inval2
```

_A2 :

```
add esi, 0
jmp _cont2
```

```
_B2 :
```

```
add esi, 1
jmp _cont2
```

```
_C2 :
```

```
add esi, 2
jmp _cont2
```

```
_D2 :
```

```
add esi, 3
jmp _cont2
```

```
_E2 :
```

```
add esi, 4
jmp _cont2
```

```
_cont2 :
mov edi, 1
```

```
cmp input2[edi], 'a'
je _A3
cmp input2[edi], 'A'
je _A3
cmp input2[edi], 'b'
je _B3
cmp input2[edi], 'B'
je _B3
cmp input2[edi], 'c'
je _C3
cmp input2[edi], 'C'
je _C3
cmp input2[edi], 'd'
je _D3
cmp input2[edi], 'D'
je _D3
cmp md, 1
je _greska3
cmp input2[edi], 'e'
```

```
je _E3
cmp input2[edi], 'E'
je _E3
```

```
_greska3 :
jmp _invalid
```

```
_A3 :
```

```
add esi, 0
jmp _final2
```

```
_B3 :
```

```
add esi, 4
add esi, 2
sub esi, md
jmp _final2
```

```
_C3 :
```

```
add esi, 8
add esi, 2
sub esi, md
add esi, 2
sub esi, md
jmp _final2
```

```
_D3 :
```

```
add esi, 12
add esi, 2
sub esi, md
add esi, 2
sub esi, md
add esi, 2
sub esi, md
jmp _final2
```

```
_E3 :
```

```
add esi, 16
jmp _final2
```

```
_final2 :
```



```

mov cor2, esi
mov eax, cor2
cmp actual[esi], 1
je _err2
mov actual[esi], 1
call screen_5x4
jmp conti

_err2 :

mwrite "Vec ste uneli ovu koordinatu!"
mov al, 7
call writechar
call crlf
mwrite "Unesite zeljenu koordinatu:"
jmp _secondary

conti :

mov esi, cor1
mov edi, cor2
mov al, arraych[esi]
cmp al, arraych[edi]
je _yey
mov actual[esi], 0
mov actual[edi], 0
jmp _scor

_yey :

inc score1
jmp _skor

_scor :
mov eax, 4500
call delay

_skor :
mov eax, 500
call delay

call clrscr
call screen_5x4
call crlf
mwrite " Rezultat = "
mov eax, score1

```

```

call writedec
call crlf

cmp score1, 9
jg _ex1
mwrite "Unesite zeljenu koordinatu:"
jmp _startup2

_ex1 :

mwrite " Pobedili ste !!"
call _restart
cmp rply, 1
jne _qu
jmp _again

_qu :

exit
main ENDP

_restart PROC
call crlf
mwrite "Da li zelite da igrate ponovo? (D/N)"
call readchar

_h1 :

cmp al, 68
je _D
cmp al, 100
je _D

cmp al, 78
je _N
cmp al, 110
je _N
call crlf
mwrite "Nije validan karakter!"
call crlf
jmp _h1

_D :

mov rply, 1

```

```

mov score1, 0
mov edi, 0
mov ecx, 20

_lp1:

mov actual[edi], 0

loop _lp1

jmp _quit

_N :

mov rply, 0
call crlf
mwrite "Hvala sto ste igrali..."

_quit :
ret
_restart ENDP

screen_5x4 PROC

call clrscr

mwrite "Ucitavanje..."
mov eax, 500
call delay
call clrscr
mov esi, 0
mov edi, 0

cmp md, 2
je _mod2
mov ecx, 6
jmp _mod1

_mod2 :

mov ecx, 5

_mod1 :

```

```

mov al, 32
call writechar

ispis_5x4 :

mov al, letter[esi]
call writechar
mov al, 32
call writechar
cmp esi, 0
je _preskok
mov al, 32
call writechar

_preskok:

inc esi
loop ispis_5x4

call crlf
call crlf

cmp md, 2
je _mode2
mov ecx, 4
jmp _mode1

_mode2 :

mov ecx, 5

_mode1 :

mov edi, 1
mov esi, 0

14:

mov temp, ecx
mov al, letter[edi]
call writechar
mov al, 32
call writechar
inc edi

cmp md, 2

```

```

je _md2
mov ecx, 5
jmp _md1

_md2 :

mov ecx, 4

_md1 :

13 :

mov al, 32
call writechar
cmp actual[esi], 1
jne _not_keep
mov al, arraych[esi]
call writechar
mov al, 32
call writechar
jmp _after

_not_keep :

mov al, 177
call writechar
mov al, 32
call writechar

_after :

inc esi
loop 13
call crlf
mov ecx, temp
call crlf
loop 14

ret

screen_5x4 ENDP

randarray PROC
mov ebx, 0
mov edi, 0

```

```
call randomize
```

```
_mesanje :
```

```
mov eax, 20
```

```
call RandomRange
```

```
mov randval, eax
```

```
mov esi, randval
```

```
mov dl, arraych[esi]
```

```
mov cl, arraych[edi]
```

```
mov arraych[edi], dl
```

```
mov arraych[esi], cl
```

```
inc edi
```

```
inc ebx
```

```
cmp ebx, 20
```

```
jne _mesanje
```

```
ret
```

```
randarray ENDP
```

```
END main
```