Катедра за рачунарску технику и информатику

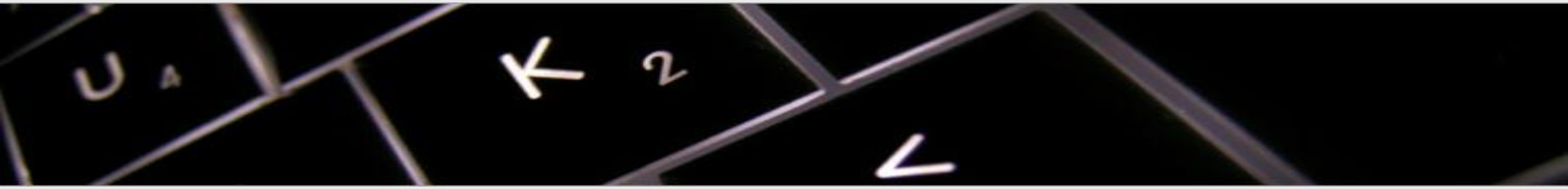
Програмирање 1

Прва година студијских програма

- **Електротехника и рачунарство**
- **Софтверско инжењерство**

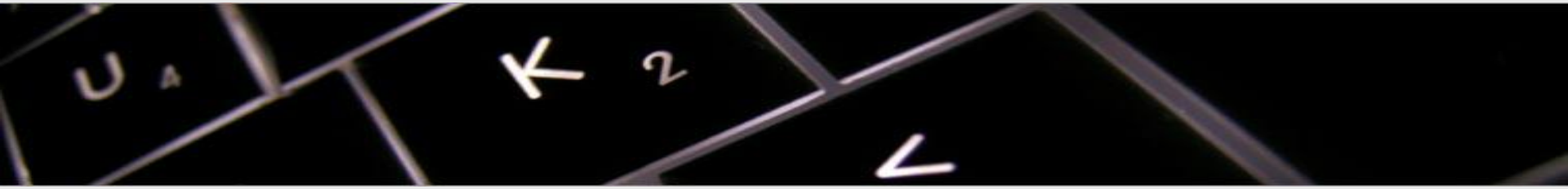
picoComputer

САДРЖАЈ

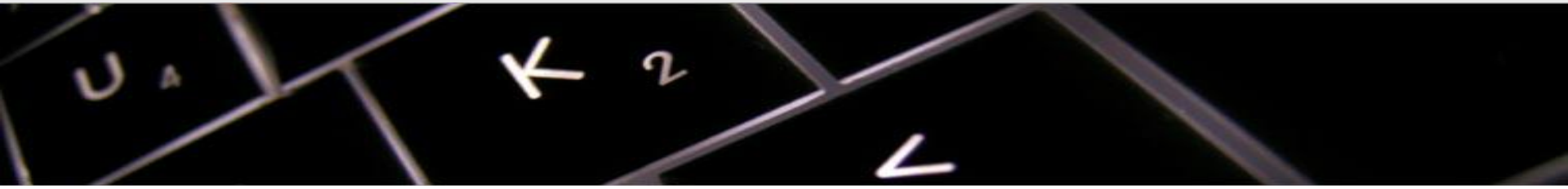


- **ПРЕГЛЕД**
- **АРХИТЕКТУРА И ОРГАНИЗАЦИЈА**
- **СКУП ИНСТРУКЦИЈА**
- **СИНТАКСА ИНСТРУКЦИЈА**
- **ПРИМЕРИ**

Преглед

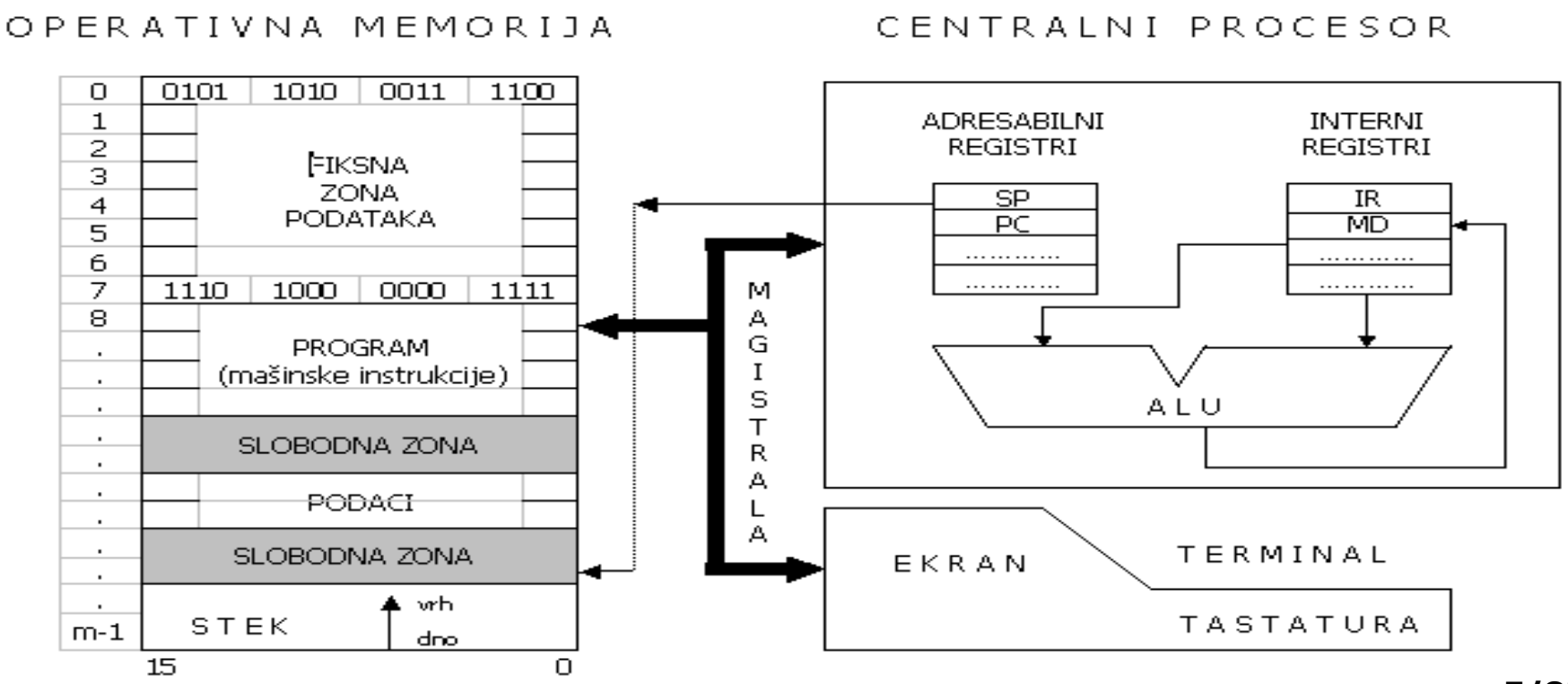


- Троадресни формат
- 16-то битне (и 32-битне)инструкције
- Меморија - 64К 16-битних речи
- 16-то битне адресе



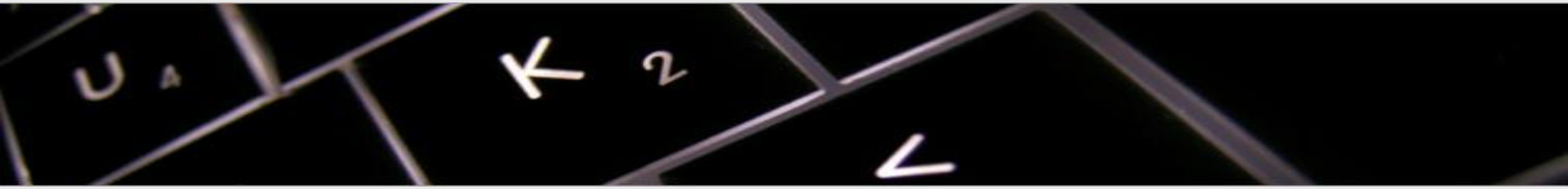
PICOCOMPUTER

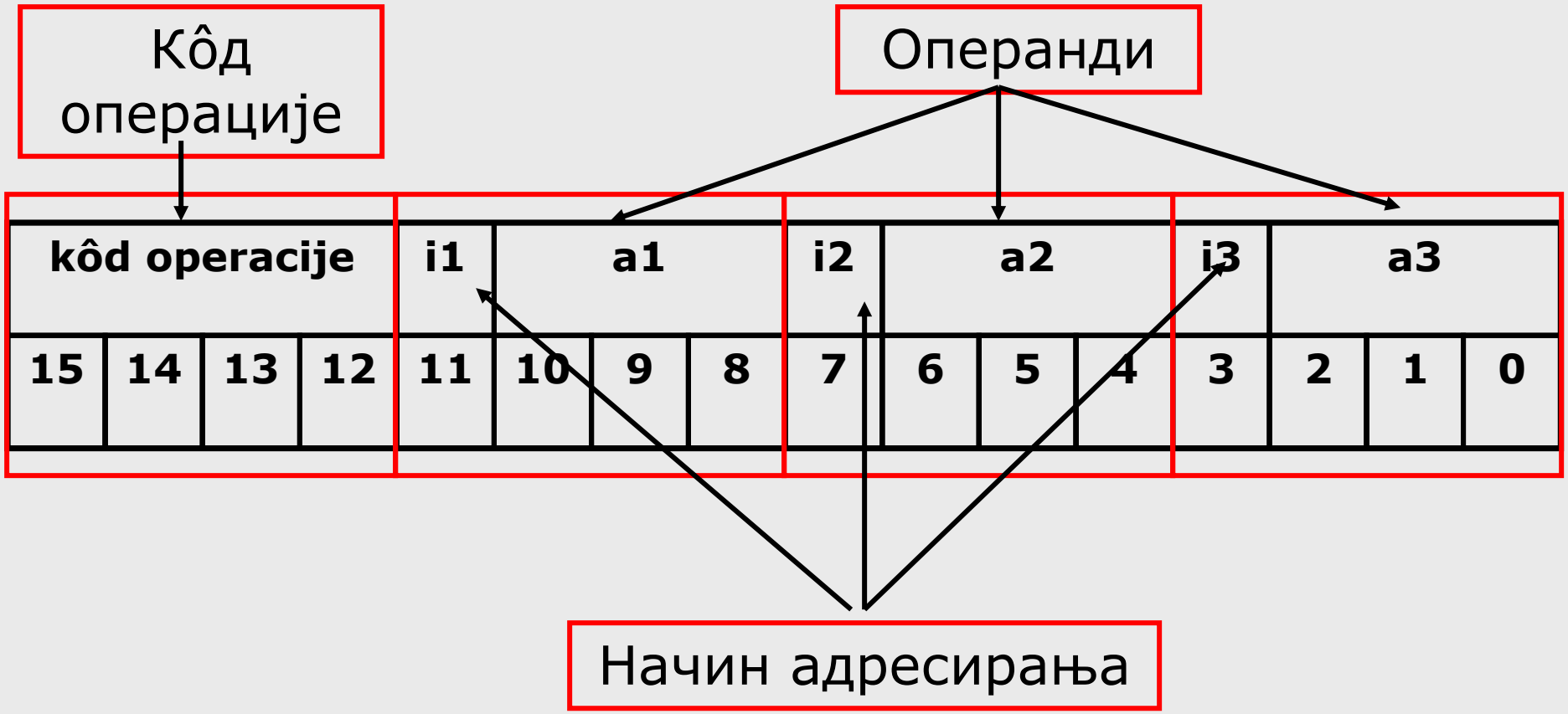
Referentni priručnik za picoComputer



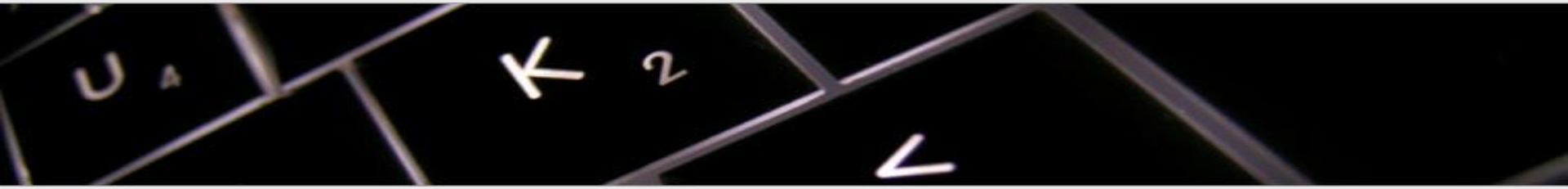
VELIČINA MEMORIJE: m ≤ 65536 (REČI SU DUŽINE 16b)

Структура машинских инструкција

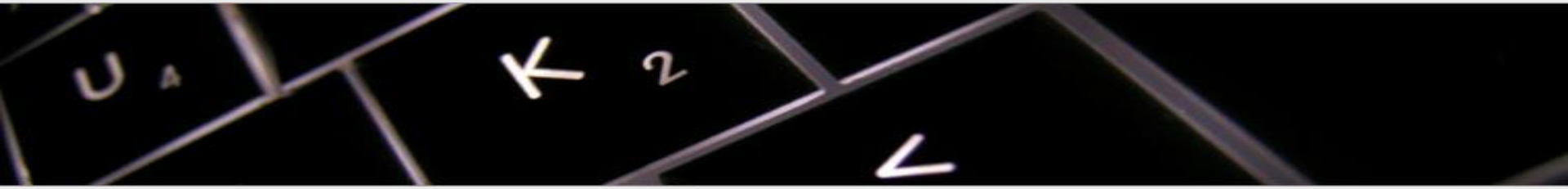




Структура машинских инструкција

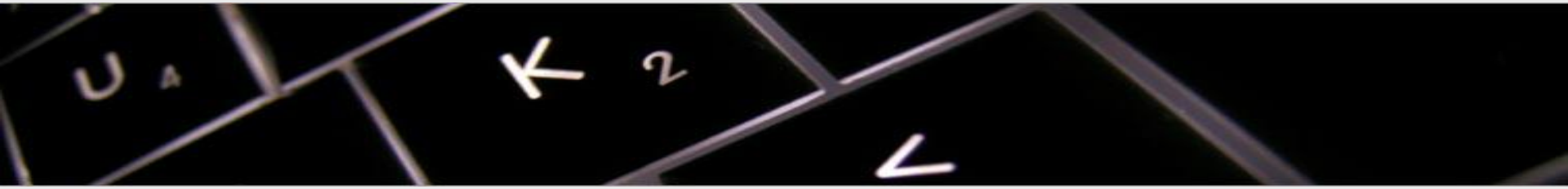


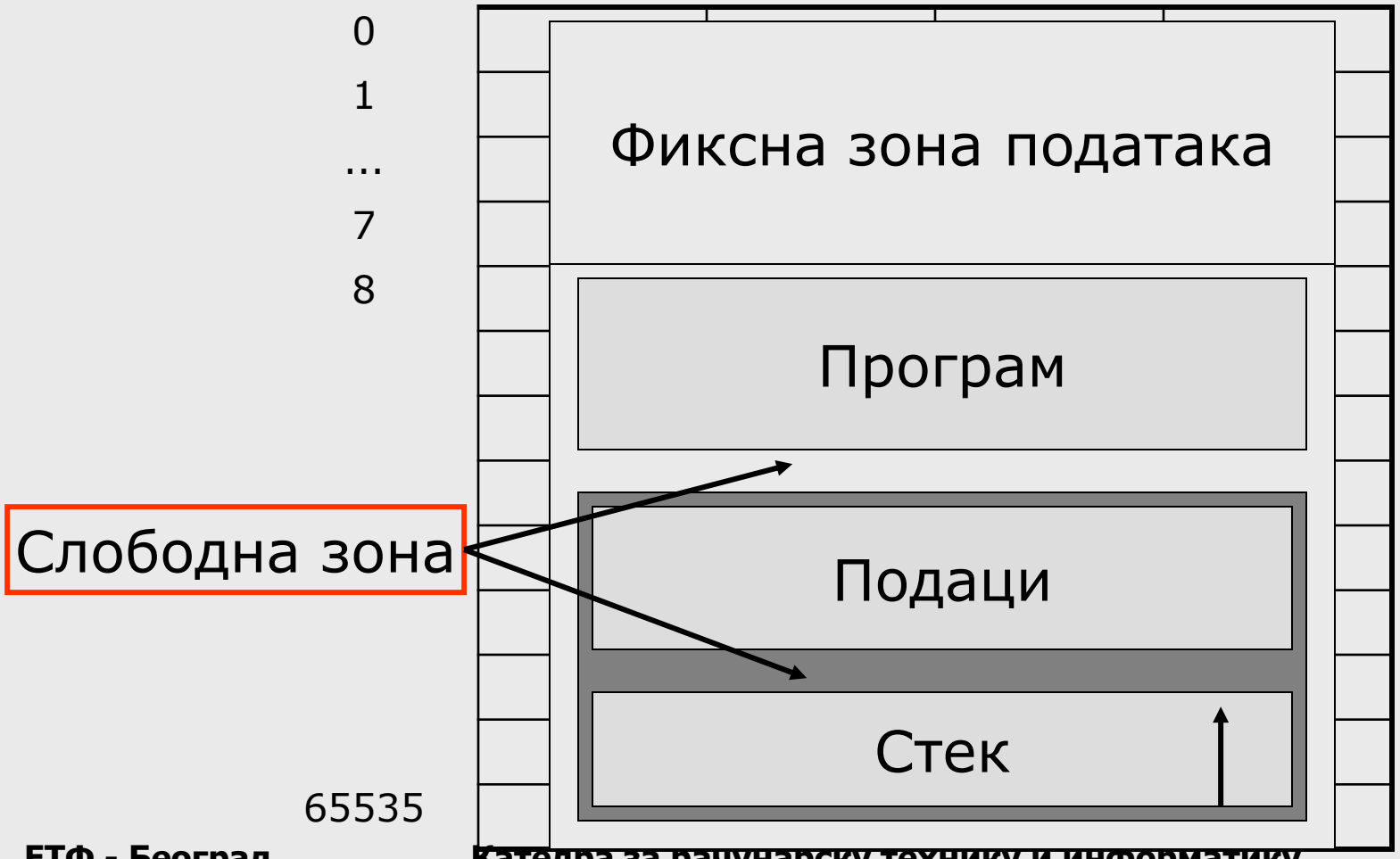
Начин адресирања



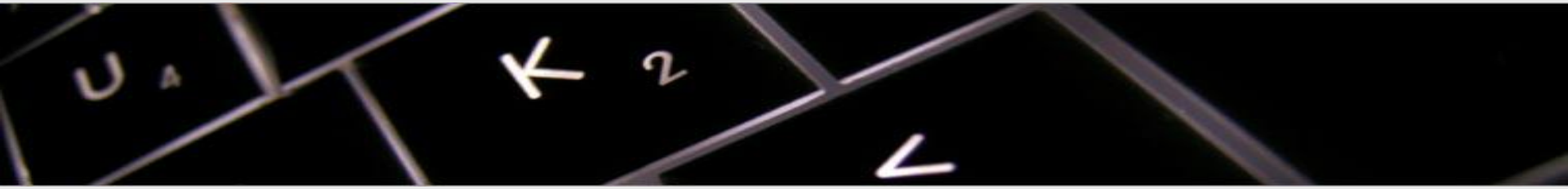
- **Меморијско директно адресирање**
  (операнд у фиксној зони)
  i = 0, adr(A) = a
- **Меморијско индиректно адресирање**
  (адреса операнда у фиксној зони,
  сам операнд било где)
  i = 1, adr(A) = val(a)

Меморија



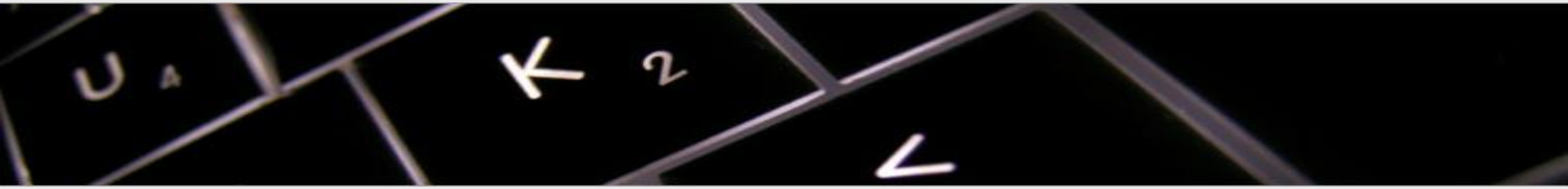

0
1
...
7
8
Фиксна зона података
Програм
Слободна зона
Подаци
Стек
65535

Типови инструкција



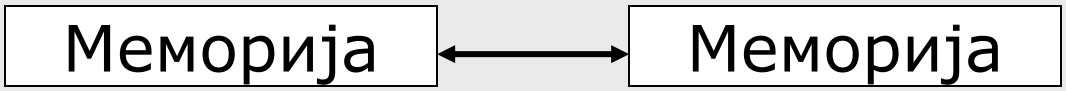
- инструкције за пренос података
- аритметичке инструкције
- контролне инструкције
- улазно-излазне инструкције

Инструкције преноса података

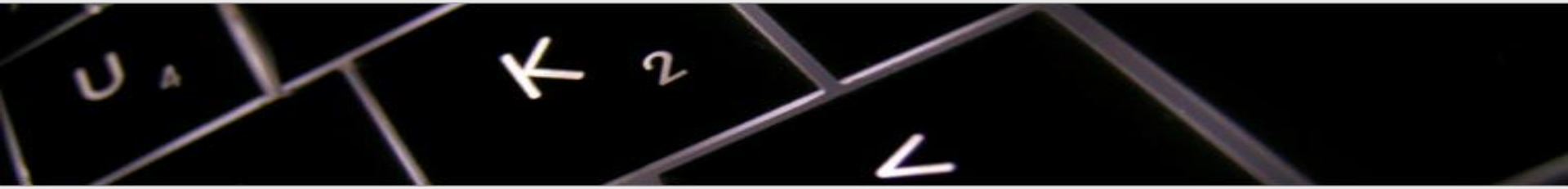


Пренос (копирање)

- скаларни
  (константа или меморија у меморију)
- векторски пренос (један блок података у меморији у други блок)

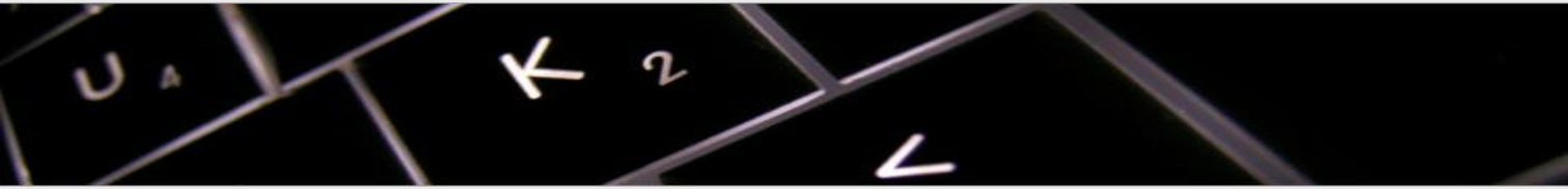


Аритметичке инструкције



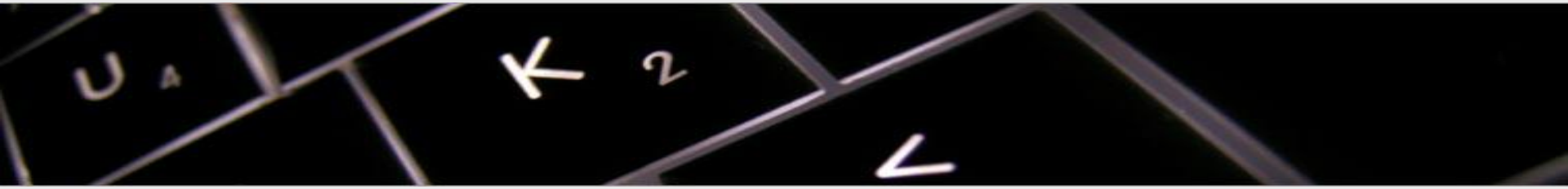
- Сабирање
- Одузимање
- Множење
- Дељење

Контролне инструкције



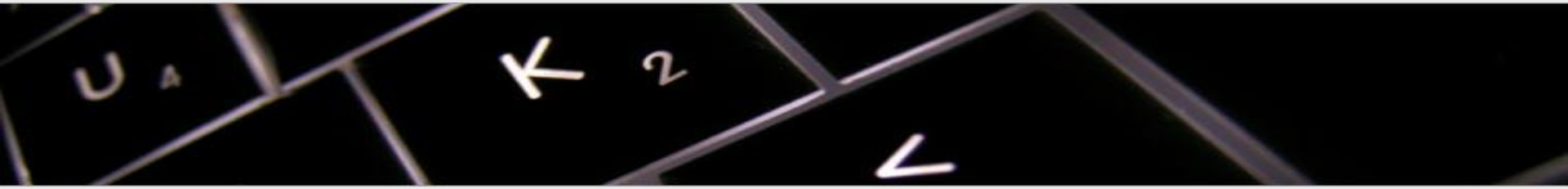
- Условни скокови – услов:
  - Једнакост аргумената
  - Први аргумент већи од другог
- Рад са потпрограмима
  - Скок у потпрограм
  - Повратак из потпрограма
- Крај рада

Улазно-излазне инструкције



- Улаз података
- Излаз података

Извршавање инструкције



MA := val(PC);

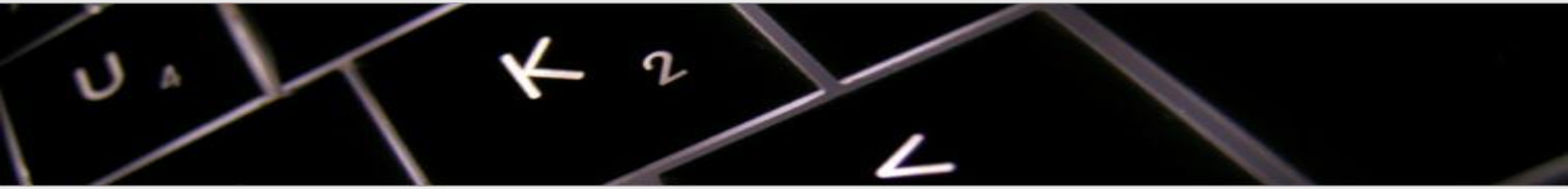
MD := val(val(MA));

IR := val(MD);

PC := val(PC) + 1;

Memory Address register
Memory Data register
Instruction Register
Program Counter

Нотација



kkkk – Код операције

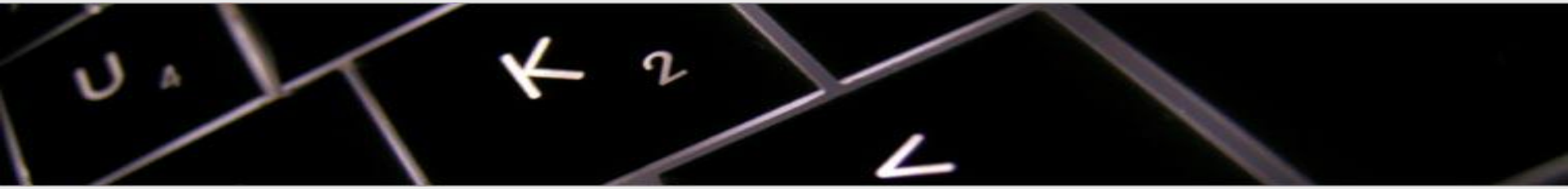
xxxx – Адресно поље (4 бита) за променљиву X

0nnn – Адресно поље (4 бита) за променљиву N у фиксној зони података

???? – Поље које се не користи

cc…c – бинарно кодирана целобројна константа C

Симболички начини адресирања



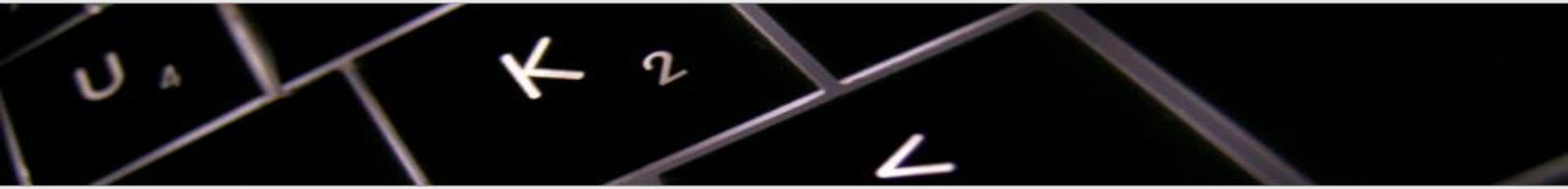
- Директно адресирање

  X – val(X)
- Индиректно адресирање

  (X) – val(val(X))
- Непосредно адресирање

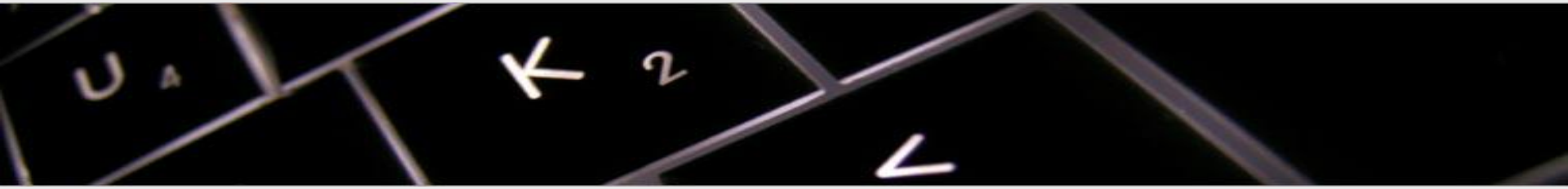
  #X - adr(X)

Симболички начини адресирања



- val – Генеричко име функције
- #X – Симболичка константа (ако је изван опсега 0...7), а не само адреса

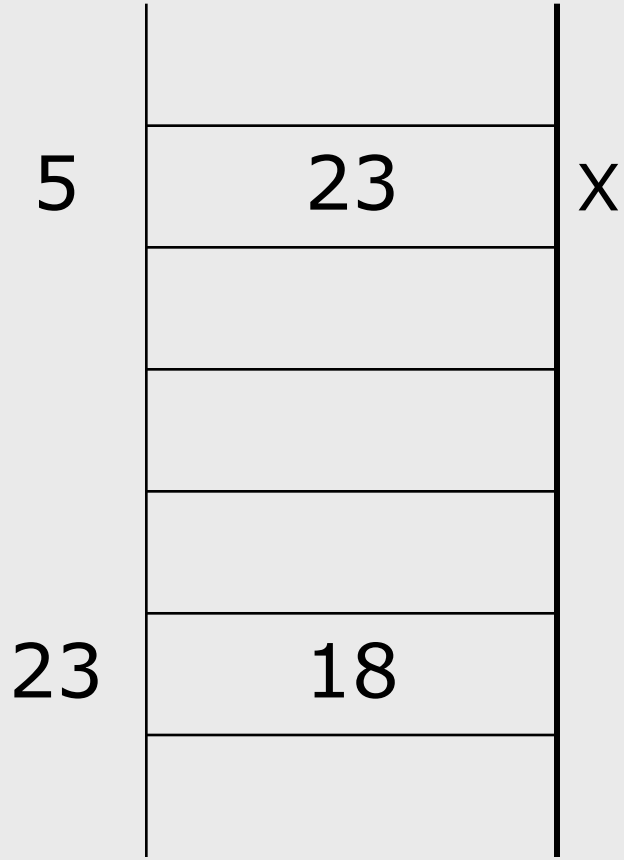
Пример



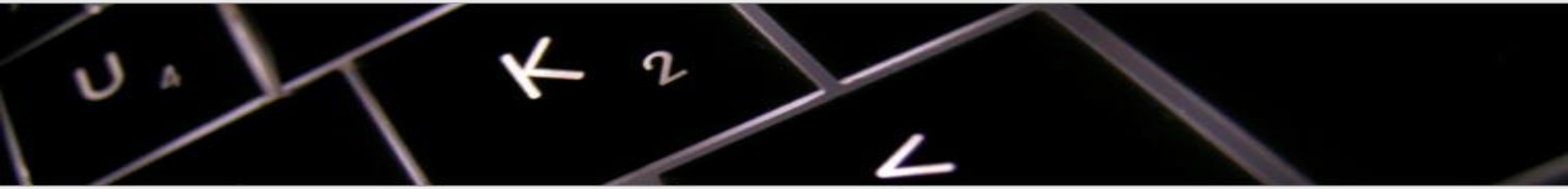
X = val(X) : 23

#X = adr(X) : 5

(X) = val(val(X)) : 18



Вектори

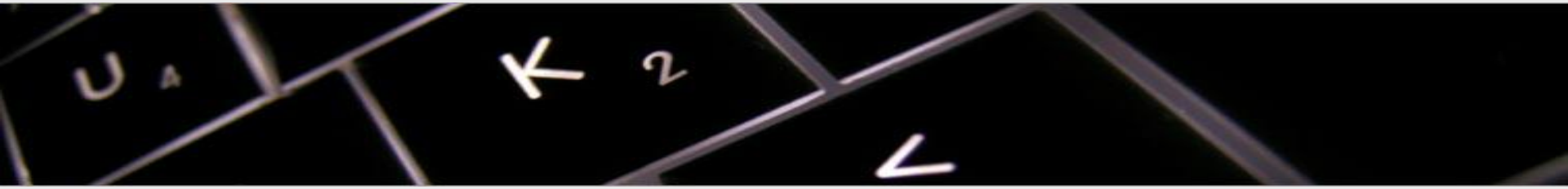


A[0], A[1], A[2], …

adr(A[j]) = adr(A) + j, j >=0

A[j] = val(adr(A) + j), j >= 0

Пример



$n = 5$

$adr(A[0]) = adr(A)$

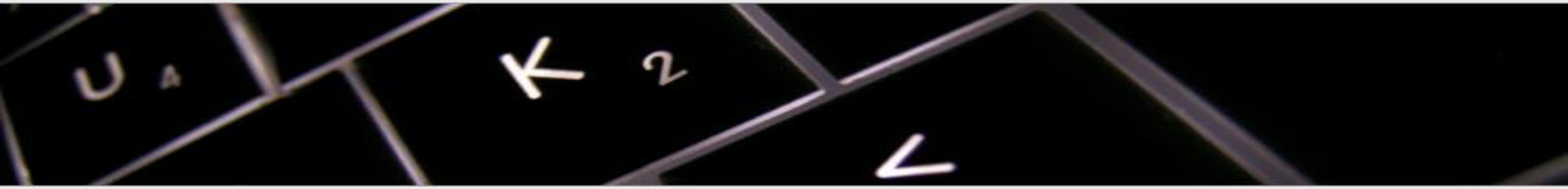
$adr(A[1]) = adr(A) + 1$

$adr(A[n-1]) = adr(A) + n - 1$

…	50	51	52	53	54	…
…	**23**	**20**	**15**	**30**	**18**	…
…	A[0]	A[1]	A[2]	A[3]	A[4]	…

50:A[0]	23
51:A[1]	20
52:A[2]	15
53:A[3]	30
54:A[4]	18

Машински језик



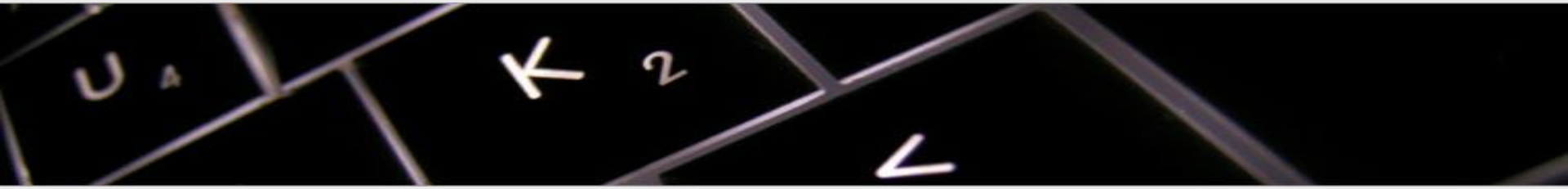
Предност:

- Директно управљање хардвером
- Оптимизација програма
- Контрола заузећа меморије

Мане:

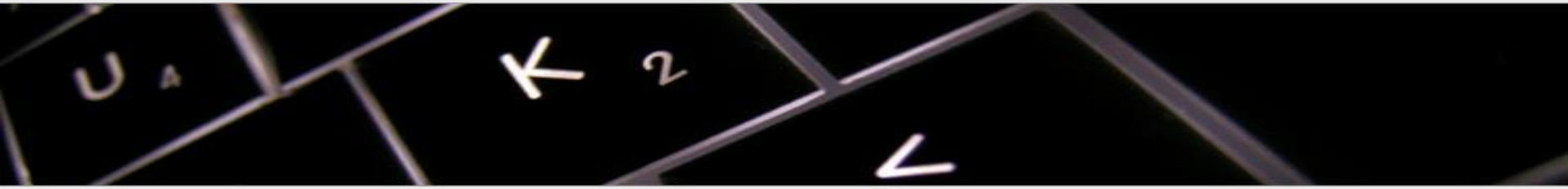
- изузетно непогодно за писање програма
- неразумљив програм

Симболички машински језик



- Симболички кодови операција
- Симболичка имена променљивих
- Симболичко означавање начина адресирања
- Симболичке ознаке инструкција

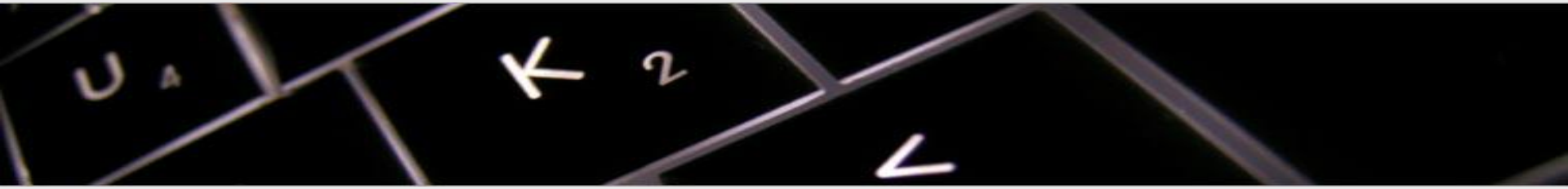
Симболичке ознаке за инструкције



0	MOV
1, 9	ADD
2, A	SUB
3, B	MUL
4, C	DIV
5	BEQ

6	BGT
7	IN
8	OUT
D	JSR
E	RTS
F	STOP

Симболичке ознаке за променљиве и константе



Иницијализација табеле симбола

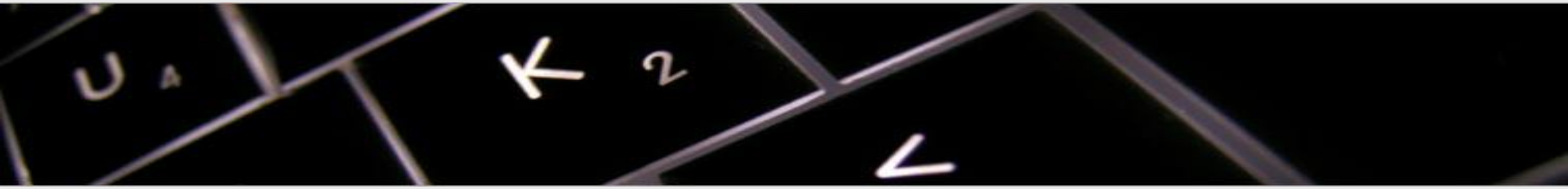
simbol = vrednost

пример:

X = 3

Симбол	Вредност
X	3

Симболичке ознаке за променљиве и константе



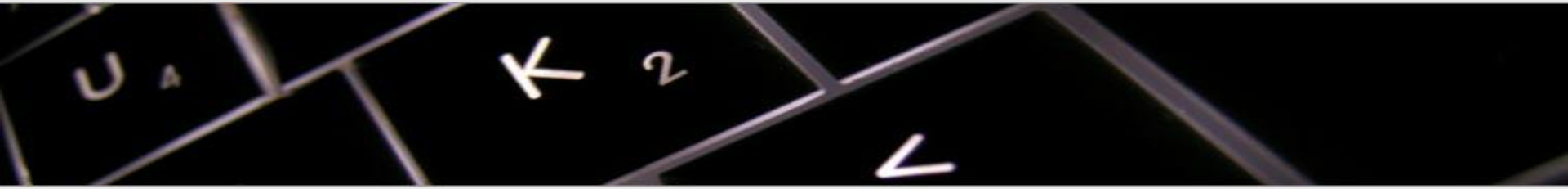
Коришћење:

X – Вредност променљиве - val(3)

#X – Вредност симбола - 3

(X) – Вредност променљиве чија је адреса X - val(val(3))

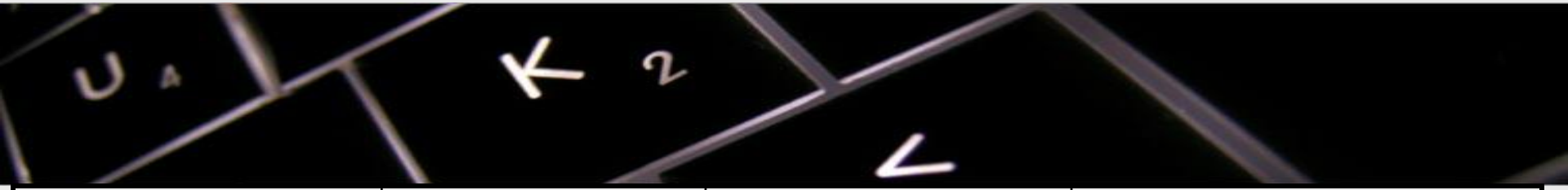
Начин записа константи



Нумерички MOV X, 65

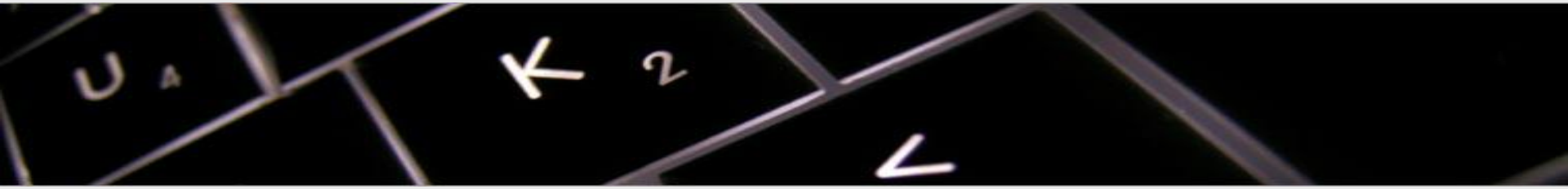
Симболички MOV X, #CNST

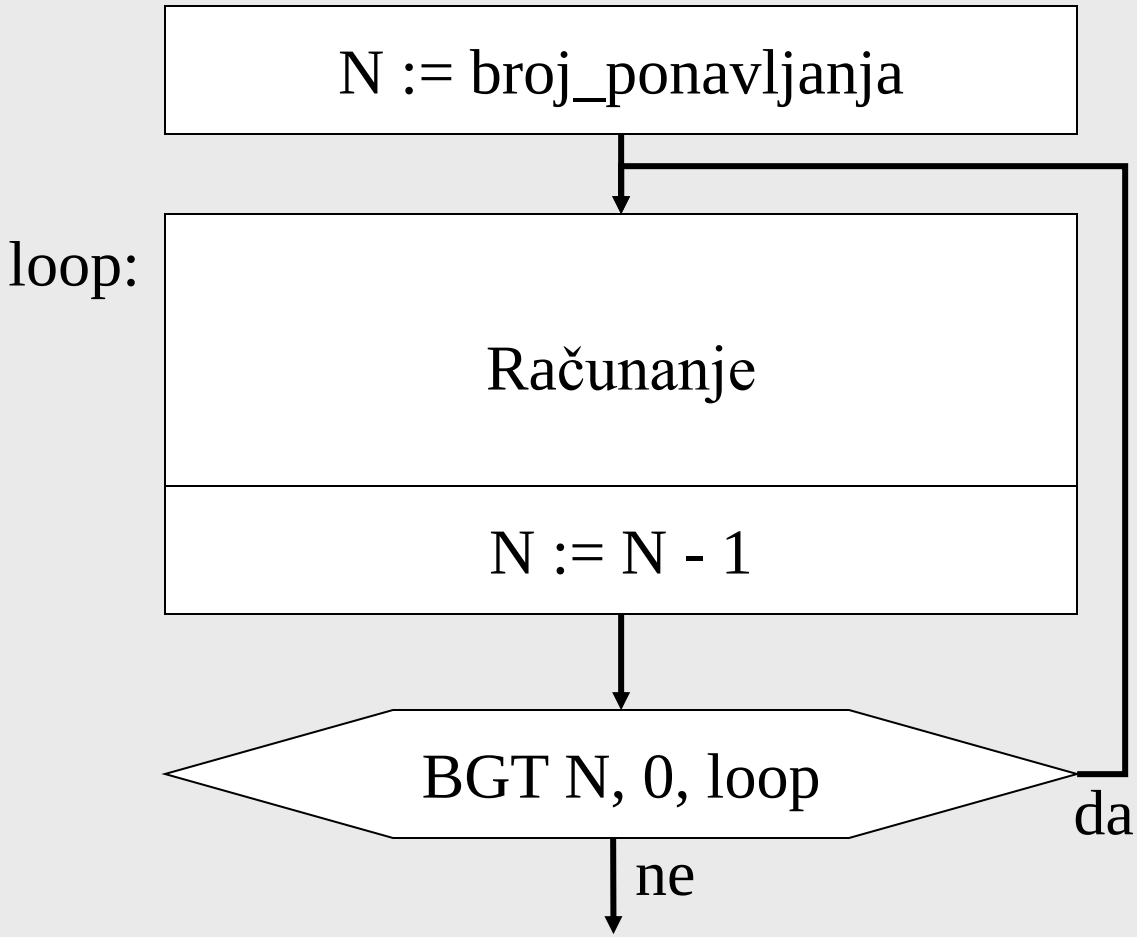
Превођење константи



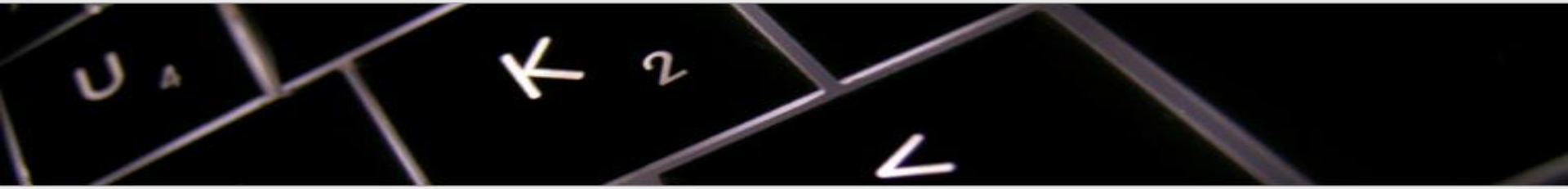
Поље симболичке адресе	Садржај адресних поља (bin)	Садржај друге речи (hex)	Коментар
65	-	0041	Непосредно
#X	-	0003	Симболичка константа или вредност симбола
X	0011		Директно
(X)	1011		Индиректно

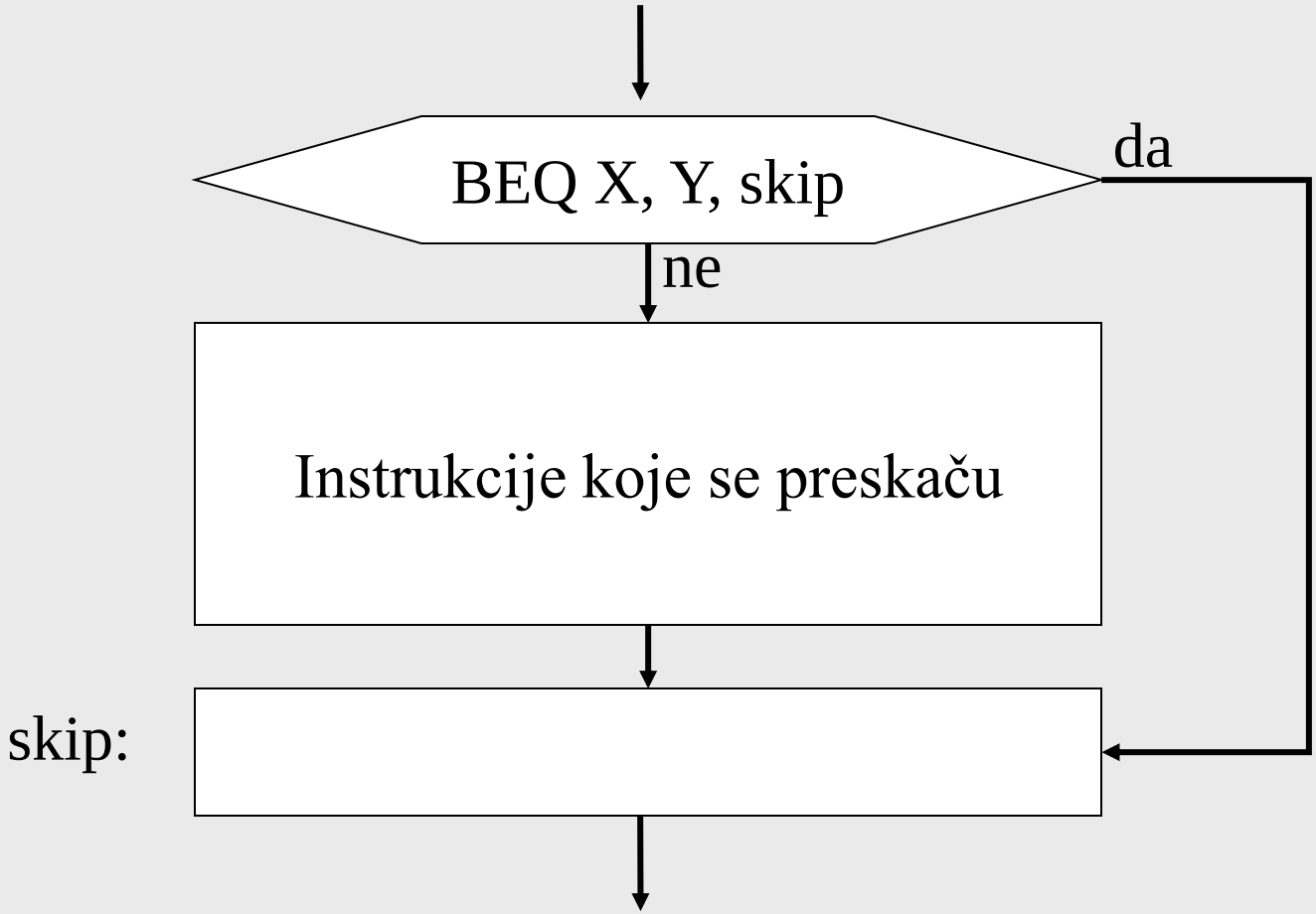
Лабеле



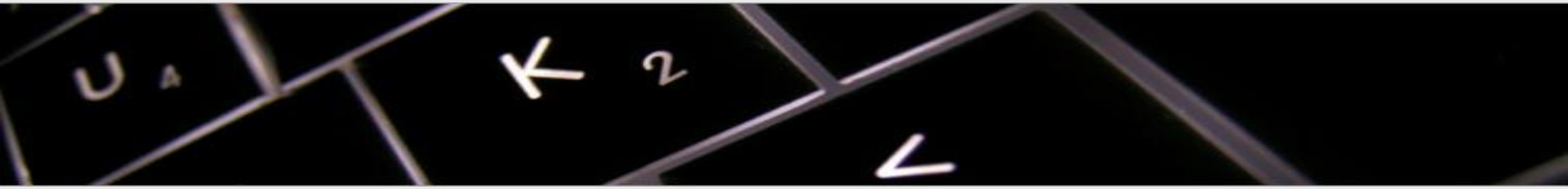


Лабеле





Синтакса - Директиве



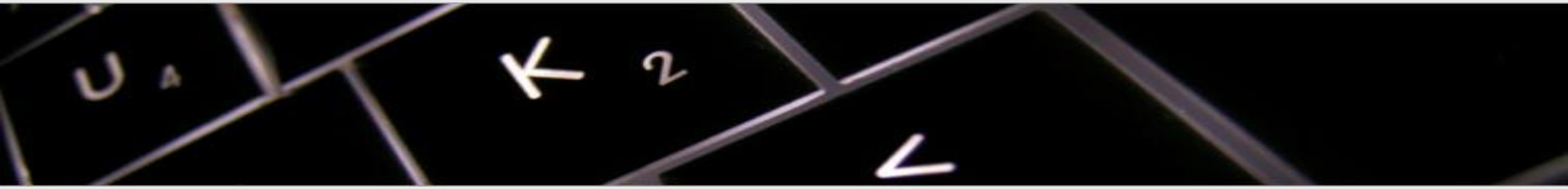
- Спецификација симбола

  simbol = konstanta [; komentar]

- Спецификација почетка програма

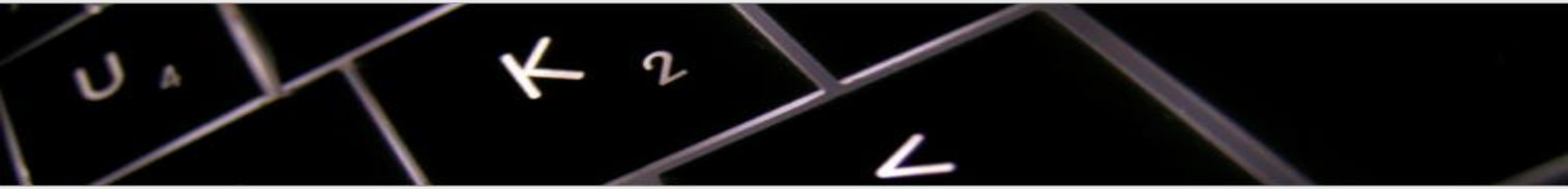
  ORG konstanta [; komentar]

Преглед структуре машинских инструкција MOV

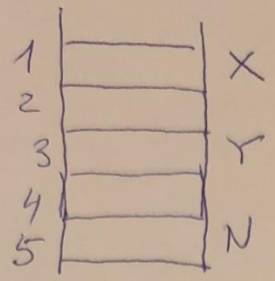


КÔД ОПЕРАЦИЈЕ	МАШИНСКА ИНСТРУКЦИЈА	ОПИС АКТИВНОСТИ, КОЈУ ИНСТРУКЦИЈА РЕАЛИЗУЈЕ
MOV 0000	0000 xxxx yyyy 0000	X:= Y, i3=0, a3=0
	0000 xxxx ???? 1000 cccc cccc cccc cccc	X:= C, i3=1, a3=0
	0000 xxxx yyyy 0nnn	X[j]:= Y[j], j=0, ..., N–1, i3=0, a3>0
	0000 xxxx yyyy 1111 cccc cccc cccc cccc	X[j]:= Y[j], j=0, ..., C–1, i3=1, a3=7

Преглед структуре машинских инструкција MOV

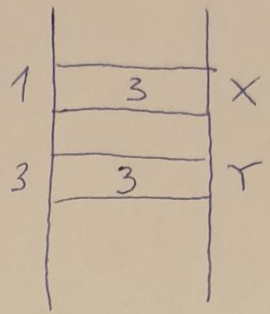


X = 1
Y = 3
N = 5



② MOV X, 3

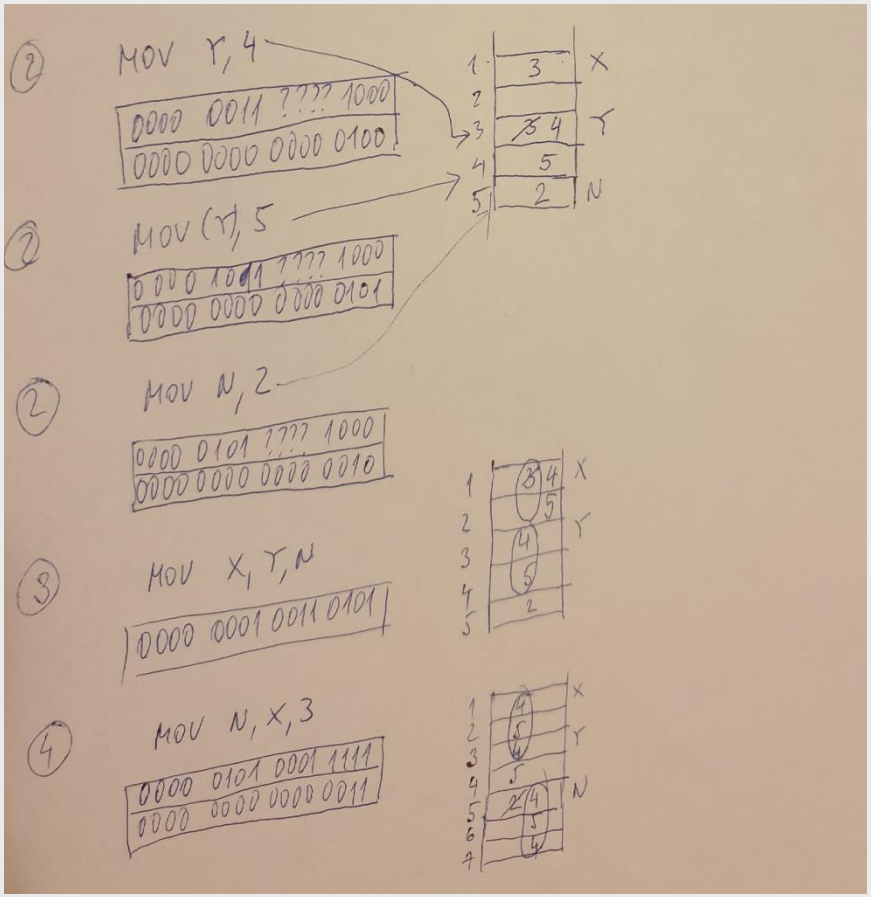
0000 0001 ???? 1000
0000 0000 0000 0011



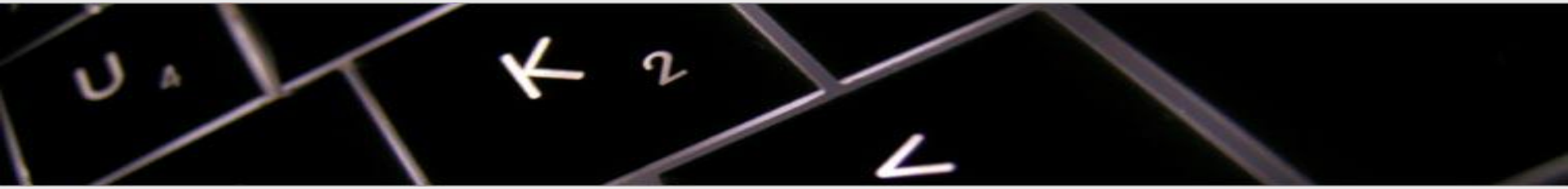
① MOV Y, X

0000	0011	0001	0000

Преглед структуре машинских инструкција MOV

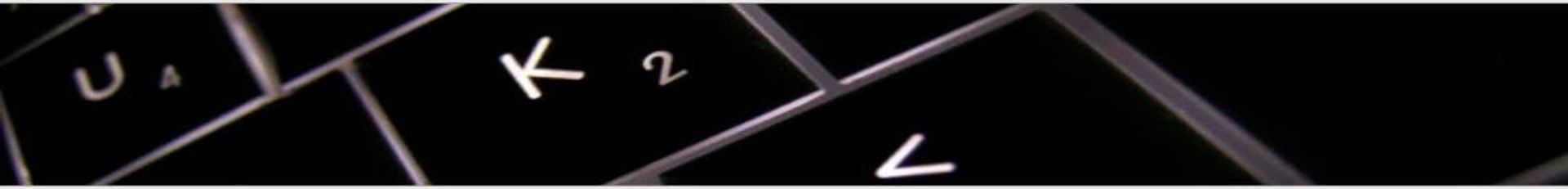


Основне машинске инструкције
MOV



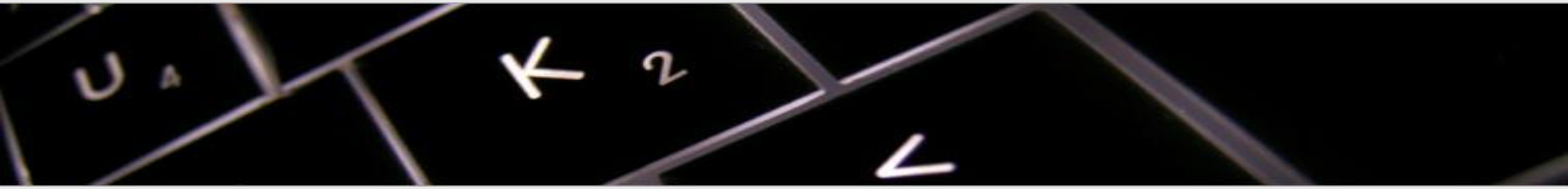
НАЗИВ ИНСТРУКЦИЈЕ	БИНАРНИ И СИМБОЛИЧКИ КÔД ОПЕРАЦИЈЕ	ОПИС АКТИВНОСТИ, КОЈУ РЕАЛИЗУЈЕ ИНСТРУКЦИЈА
ПОМЕРАЊЕ	0000 MOV	i3=0, a3=0 $\Rightarrow$ A1:=A2 i3=1, a3=0 $\Rightarrow$ A1:=val(val(PC)) i3=0, a3>0 $\Rightarrow$ A1[j]:=A2[j], j=0, 1, …,val(A3)–1 i3=1, a3=7 $\Rightarrow$ A1[j]:=A2[j], j=0, 1, …,val(val(PC))–1

Синтакса - Спецификација извршних инструкција MOV



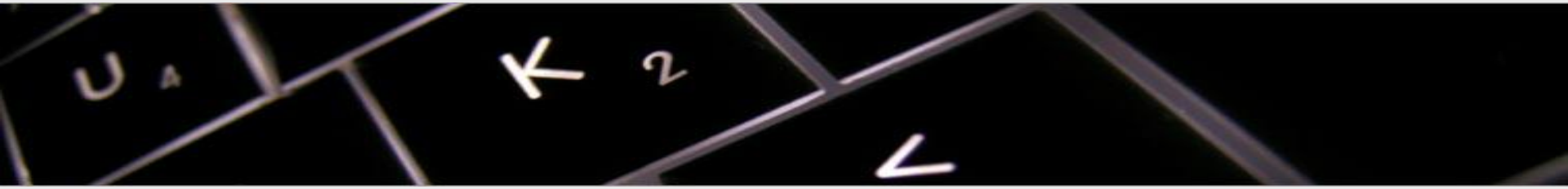
$$[simbol:]MOV \left\{\begin{matrix} simbol \\ (simbol) \end{matrix}\right\}, \left\{\begin{matrix} \left\{\begin{matrix} \text{konstanta} \\ \#simbol \\ simbol \\ (simbol) \end{matrix}\right\} \\ \left\{\begin{matrix} simbol \\ (simbol) \end{matrix}\right\}, \left\{\begin{matrix} \text{konstanta} \\ \#simbol \\ simbol \end{matrix}\right\} \end{matrix}\right\} [;komentar]$$

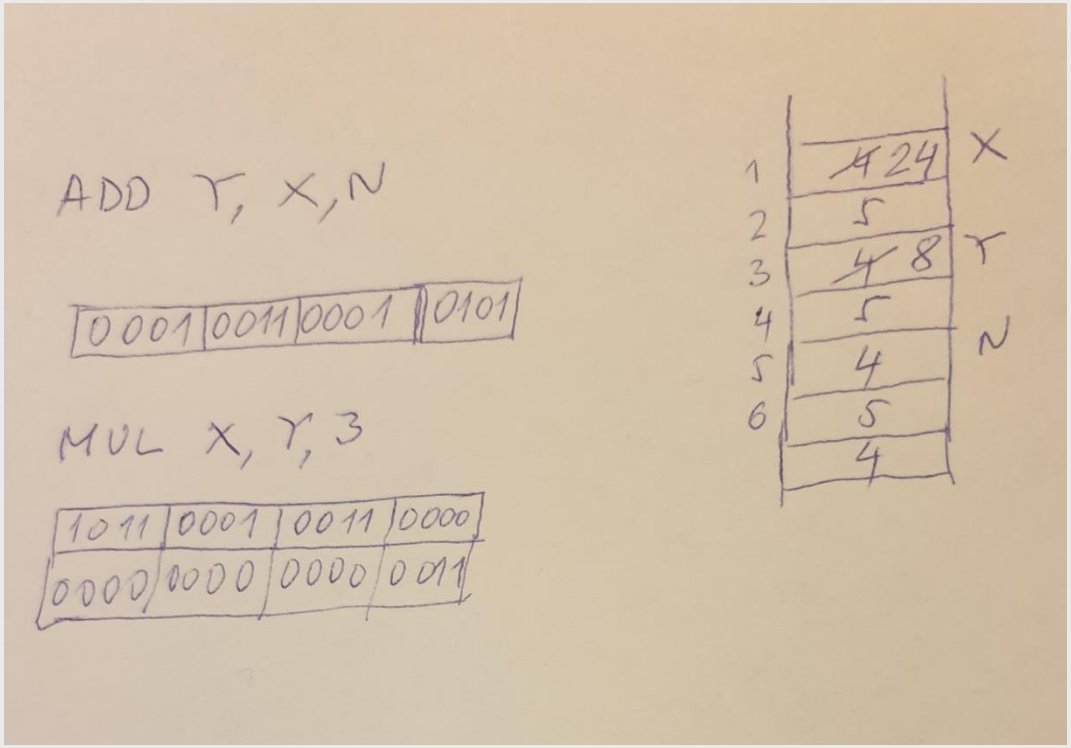
Преглед структуре машинских инструкција
ADD, SUB, MUL, DIV



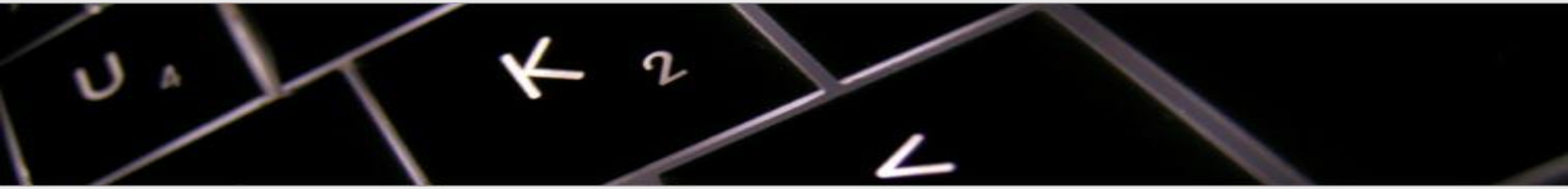
ADD b001 SUB b010	0kkk xxxx yyyy zzzz	X : = Y ⊕ Z, ⊕ ∈ { +, –, *, / }
MUL b011 DIV b100	1kkk xxxx 0000 zzzz cccc cccc cccc cccc	X : = C ⊕ Z (zzzz>0)

Преглед структуре машинских инструкција
ADD, SUB, MUL, DIV



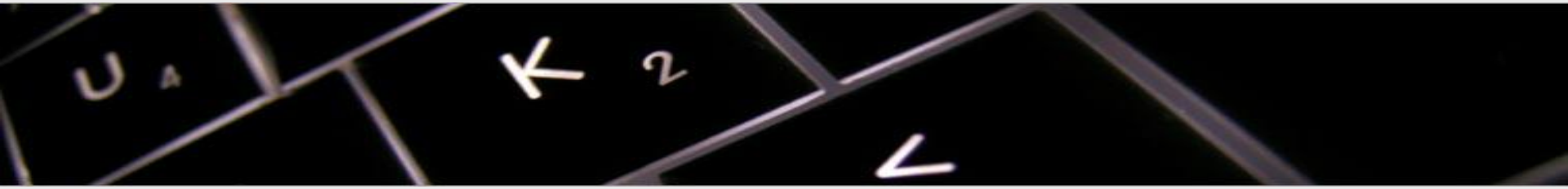


Основне машинске инструкције
ADD, SUB, MUL, DIV



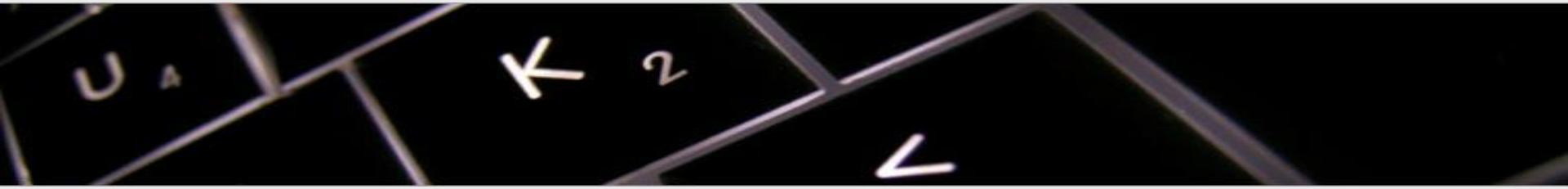
САБИРАЊЕ	0001 ADD 1001 ADD	A1:=A2+A3 (сви аргументи су променљиве) A1:=A2+val(val(PC)), или A1:=val(val(PC))+A3
ОДУЗИМАЊЕ	0010 SUB 1010 SUB	A1:=A2–A3 (сви аргументи су променљиве) A1:=A2–val(val(PC)), или A1:=val(val(PC))–A3
МНОЖЕЊЕ	0011 MUL 1011 MUL	A1:=A2*A3 (сви аргументи су променљиве) A1:=A2*val(val(PC)), или A1:=val(val(PC))·A3
ДЕЉЕЊЕ	0100 DIV 1100 DIV	A1:=⌊A2/A3⌋ (сви аргументи су променљиве) A1:=⌊A2/val(val(PC))⌋, или A1:=⌊val(val(PC))/A3⌋ За кодове > 8 ⇒ мора бити adr(A2)>0 ∧ adr(A3)>0

Синтакса - Спецификација извршних инструкција ADD, SUB, MUL, DIV



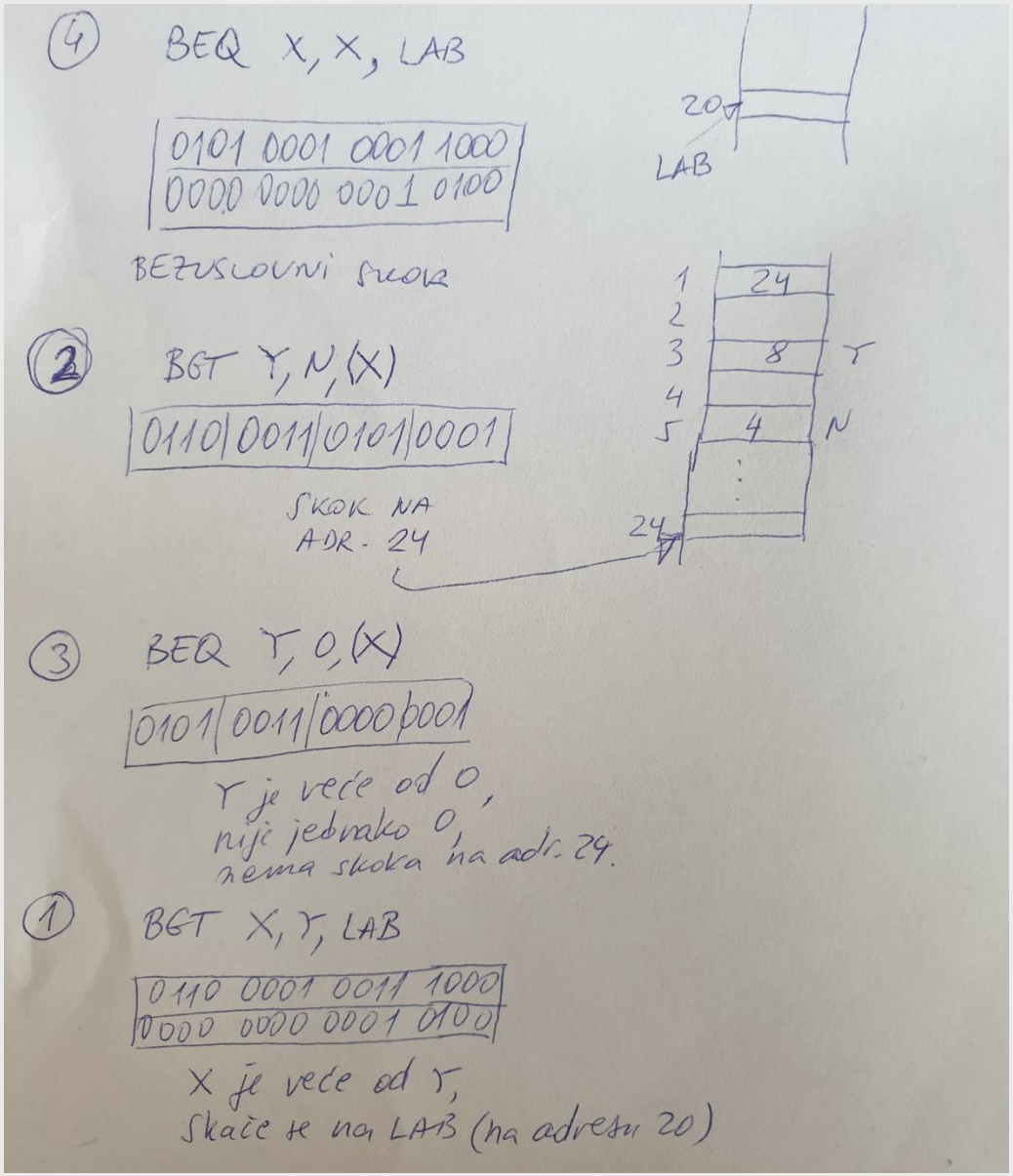
$$[simbol:]\begin{Bmatrix}ADD\\SUB\\MUL\\DIV\end{Bmatrix}\begin{Bmatrix}simbol\\(simbol)\end{Bmatrix},\begin{Bmatrix}\begin{Bmatrix}simbol\\(simbol)\end{Bmatrix},\begin{Bmatrix}\text{konstanta}\\\#simbol\\simbol\\(simbol)\end{Bmatrix}\\\begin{Bmatrix}\text{konstanta}\\\#simbol\end{Bmatrix},\begin{Bmatrix}simbol\\(simbol)\end{Bmatrix}\end{Bmatrix}[;komentar]$$

Преглед структуре машинских инструкција
BEQ, BGT

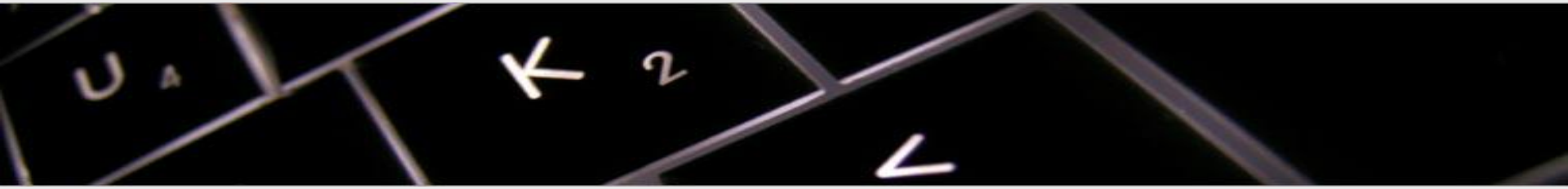


BEQ 0101 BGT 0110	kkkk xxxx yyyy 1000 cccc cccc cccc cccc	if X ® Y then PC:=ccccccccccccccc (xxxx>0, yyyy>0)
	kkkk xxxx yyyy 0nnn	if X ® Y then PC:= val(nnn) (xxxx>0)
	kkkk xxxx 0000 0nnn	if X ® 0 then PC:= val(nnn) (xxxx>0)
	0101 xxxx xxxx 1000 cccc cccc cccc cccc	PC:= ccccccccccccccc (xxxx>0) (GOTO C)

Преглед структуре машинских инструкција BEQ, BGT

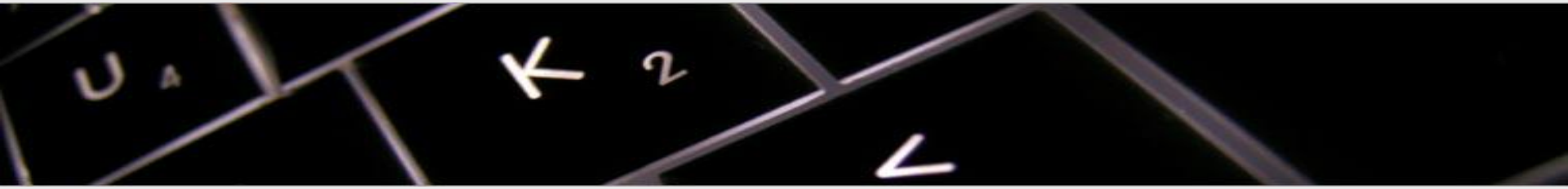


Основне машинске инструкције
BEQ, BGT



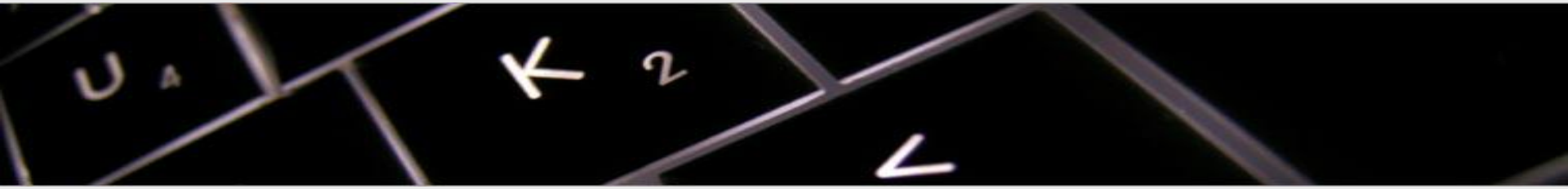
УСЛОВНИ СКОК, АКО ЈЕ ПРВИ АРГУМЕНТ ЈЕДНАК ДРУГОМ	0101 BEQ	Условни скокови реализују операцију: ако је Uslov_Ispunjen $\Rightarrow$ PC:=Adresa_Skoka. Могуће је користити три врсте услова: (1) A1=A2, или A1>A2 (kodirano: a1>0 $\wedge$ a2>0) (2) A1=0, или A1>0 (кодирано: a1>0 $\wedge$ i2=a2=0) (3) 0=A2, или 0>A2 (кодирано: i1=a1=0 $\wedge$ a2>0)
УСЛОВНИ СКОК, АКО ЈЕ ПРВИ АРГУМЕНТ ВЕЋИ ОД ДРУГОГ	0110 BGT	Услови: adr(A1)>0 $\wedge$ adr(A2)>0 Дозвољене су две адресе скока: (1) i3=0 $\wedge$ 0$\leq$a3$\leq$7 $\Rightarrow$ Adresa_Skoka=val(a3) (2) i3=1 $\wedge$ a3=0 $\Rightarrow$ Adresa_Skoka=val(val(PC))

Синтакса - Спецификација извршних инструкција BEQ, BGT



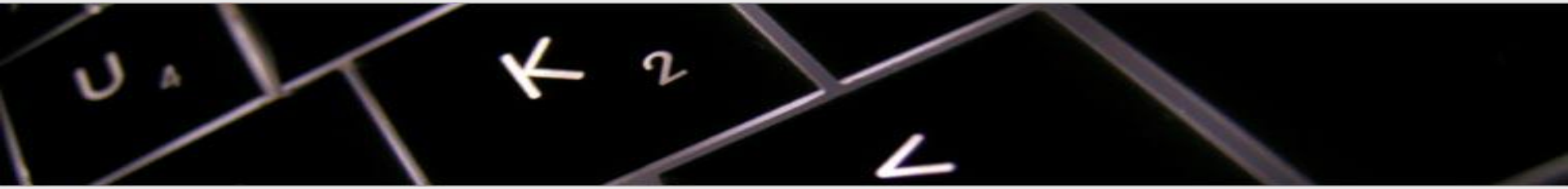
$$[simbol:]\ \left\{\begin{matrix} BEQ \\ BGT \end{matrix}\right\}\ \left\{\begin{matrix} \left\{\begin{matrix} simbol \\ (simbol) \end{matrix}\right\}, \left\{\begin{matrix} simbol \\ (simbol) \end{matrix}\right\} \\ \left\{\begin{matrix} simbol \\ (simbol) \end{matrix}\right\}, 0 \\ 0, \left\{\begin{matrix} simbol \\ (simbol) \end{matrix}\right\} \end{matrix}\right\}, \left\{\begin{matrix} simbol \\ (simbol) \end{matrix}\right\}\ [;komentar]$$

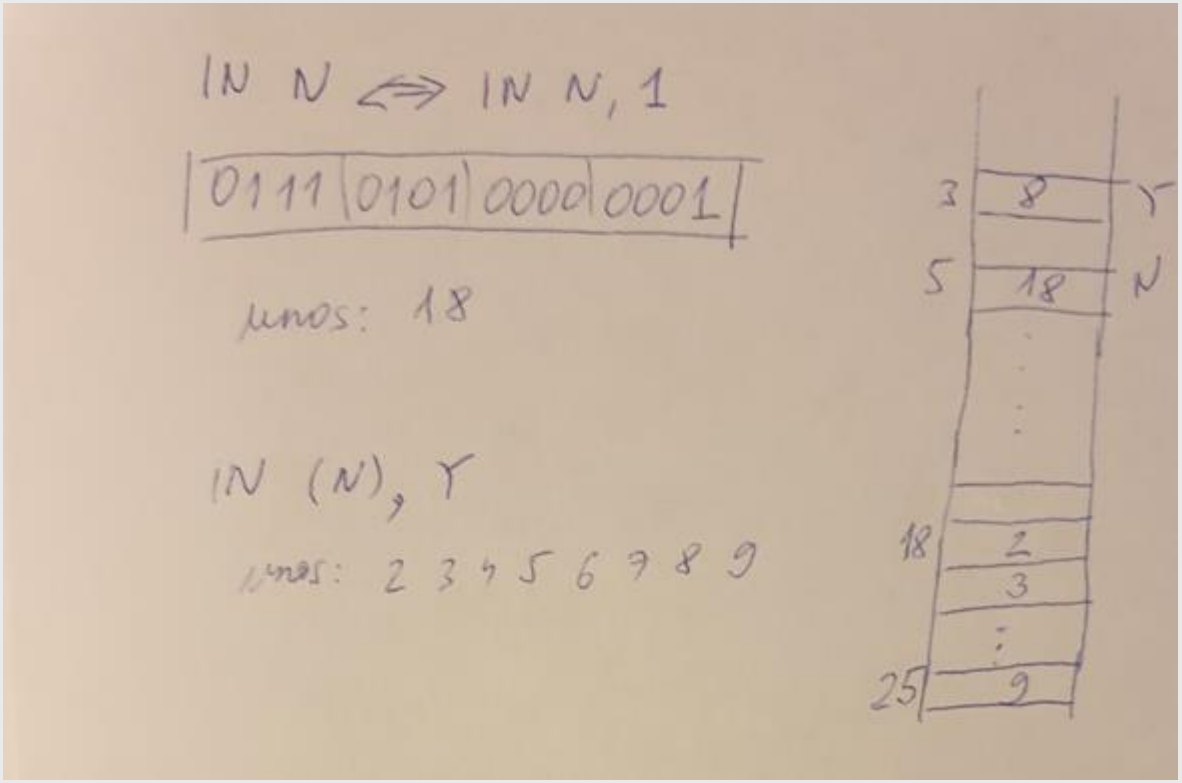
Преглед структуре машинских инструкција IN, OUT



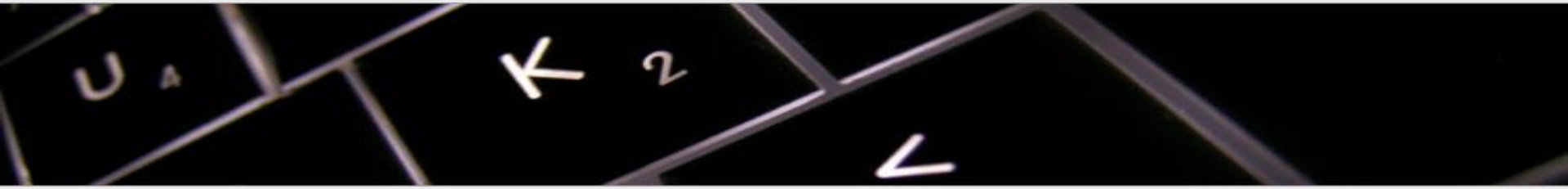
IN 0111 OUT 1000	kkkk xxxx 1000 nnnn kkkk xxxx 0ccc cccc	Улаз/излаз за X[j], j=0, ..., N–1 Улаз/излаз за X[j], j=0, ..., C–1

Преглед структуре машинских инструкција IN, OUT



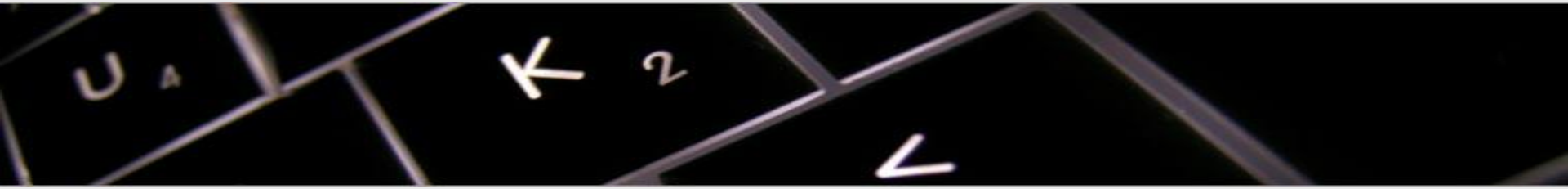


Основне машинске инструкције
IN, OUT



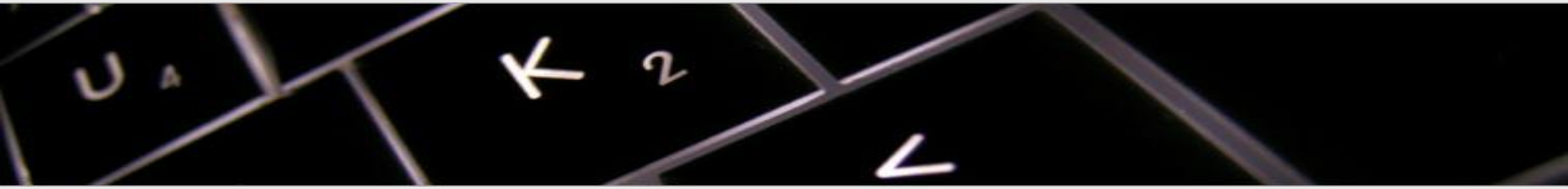
УЛАЗ ПОДАТАКА	0111 IN	Пренос n целих бројева са тастатуре, на локације A1[j], J=0, ..., n-1. Ако је i2=0 $\Rightarrow$ битови 0–6 десне полуречи садрже n, па је n $\leq$ 127. Ако је i2=1 $\wedge$ a2=0 $\Rightarrow$ n=A3. Увек мора бити n>0.
ИЗЛАЗ ПОДАТАКА	1000 OUT	Пренос n целих бројева, са локација A1[j], j=0, …, n–1 на терминал. Интерпретација n је иста, као у инструкцији IN.

Синтакса - Спецификација извршних инструкција IN, OUT



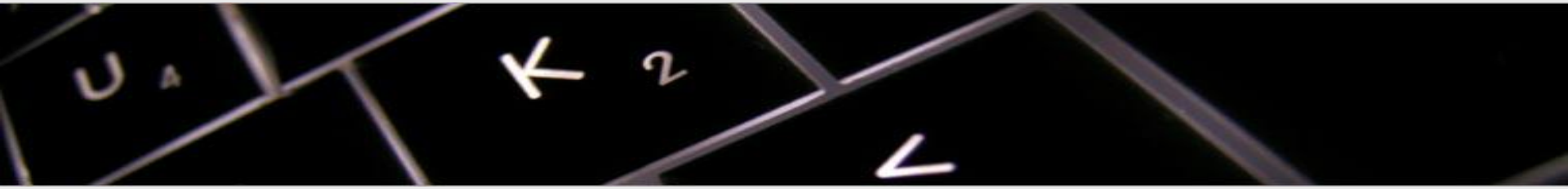
$$[simbol:]\ \left\{\begin{matrix} IN \\ OUT \end{matrix}\right\}\ \left\{\begin{matrix} simbol \\ (simbol) \end{matrix}\right\}\ \left[\ ,\ \left\{\begin{matrix} \text{konstanta} \\ \#simbol \\ simbol \\ (simbol) \end{matrix}\right\}\ \right]\ [;\ komentar]$$

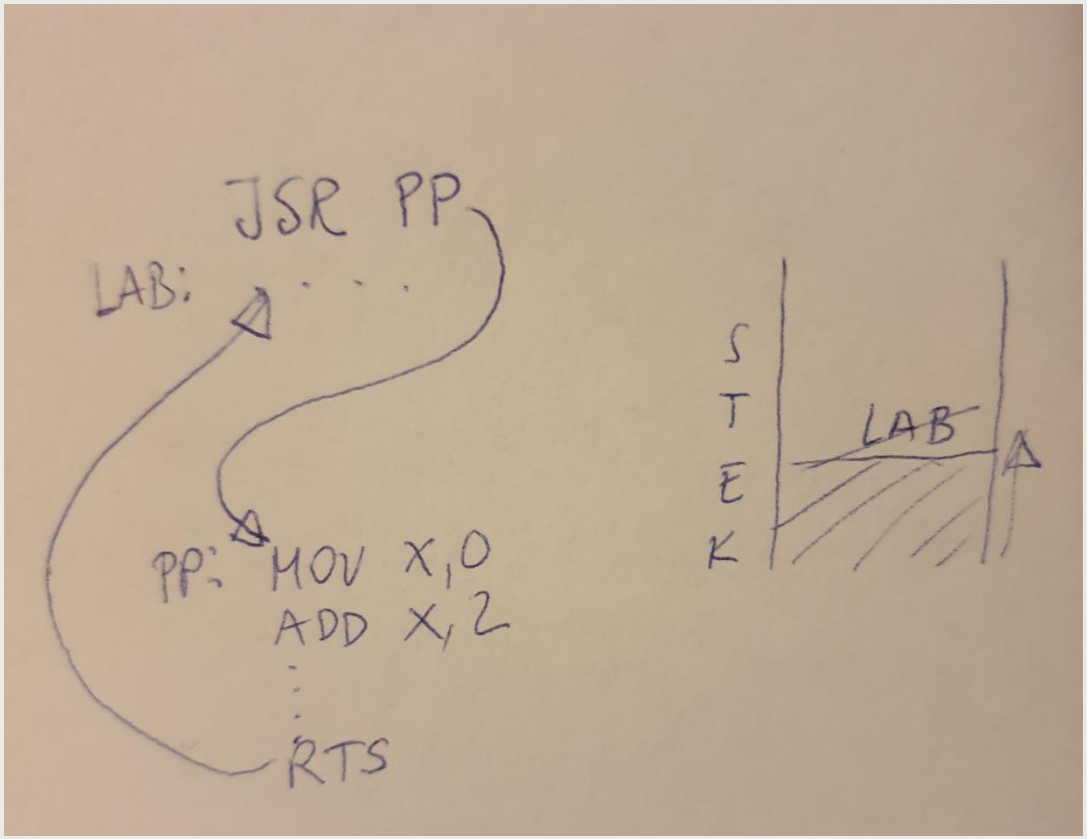
Преглед структуре машинских инструкција
JSR, RTS



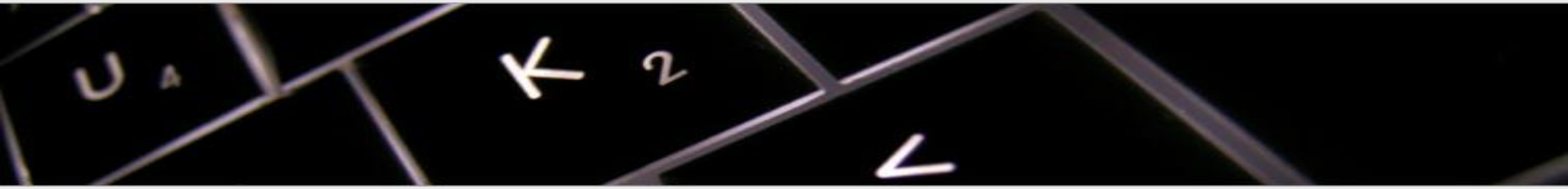
JSR 1101	1101 ???? ???? ???? cccc cccc cccc cccc	Stek:=PC+1, PC:=ccccccccccccccc
RTS 1110	1110 ???? ???? ????	PC:=stek

Преглед структуре машинских инструкција
JSR, RTS



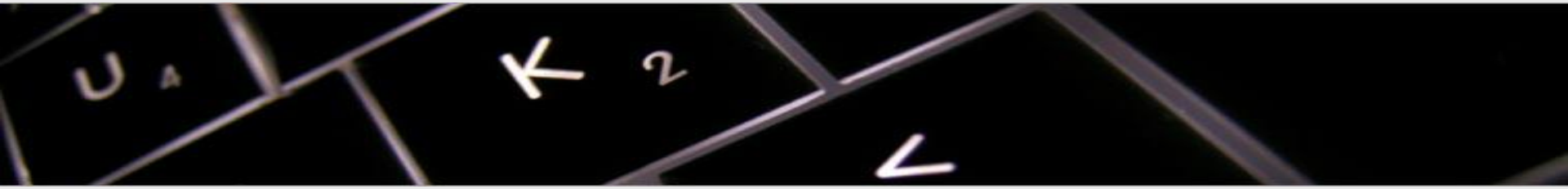


Основне машинске инструкције
JSR, RTS



СКОК У ПОТПРОГРАМ	1101 JSR	stek := PC+1; PC:=val(val(PC)) . Вредност PC након скока је адреса друге речи инструкције.
ПОВРАТНИ СКОК ИЗ ПОТПРОГРАМА	1110 RTS	PC:=stek програмски бројач се пуни вредношћу повратне адресе узете са стека

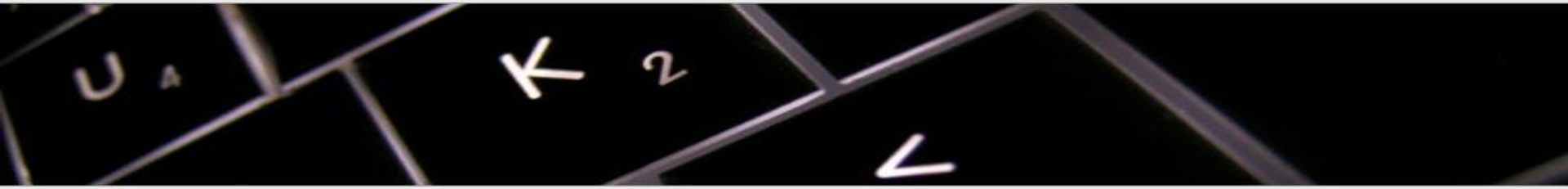
Синтакса - Спецификација извршних инструкција JSR, RTS



[*simbol*:] *JSR* *simbol* [; *komentar*]

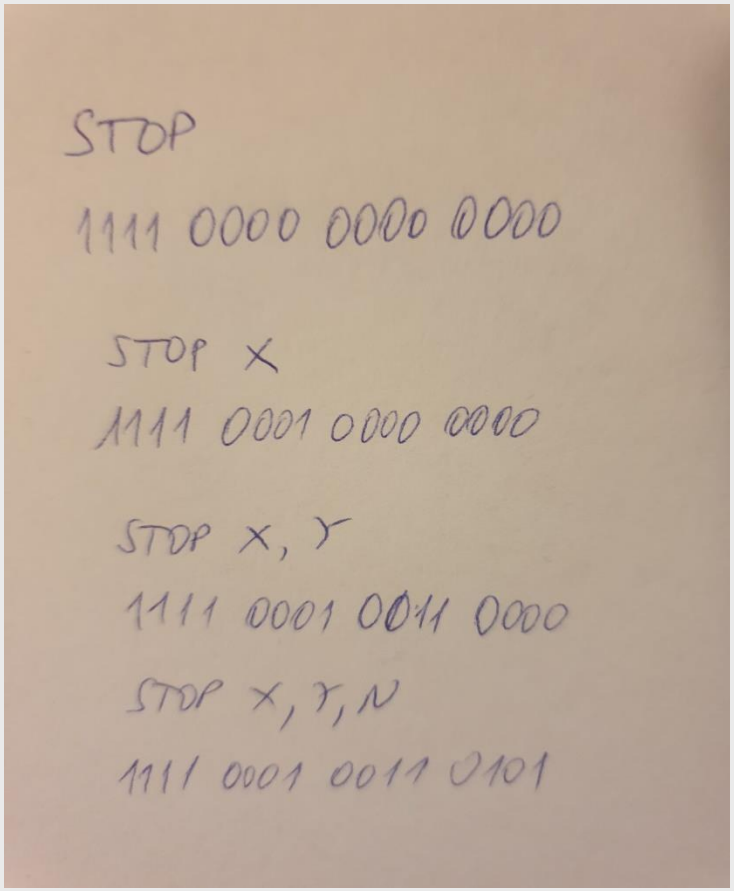
[*simbol*:] *RTS* [; *komentar*]

Преглед структуре машинских инструкција
STOP

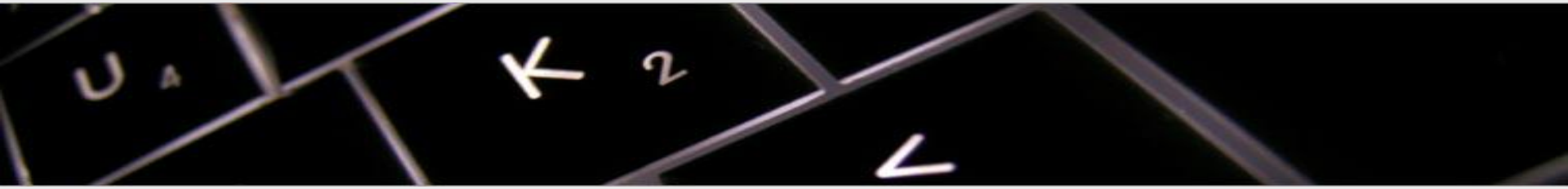


STOP 1111	1111 0000 0000 0000 1111 xxxx yyyy zzzz	Крај рада Опциони приказ X, Y, Z (xxxx>0, yyyy>0, zzzz>0)

Преглед структуре машинских инструкција
STOP

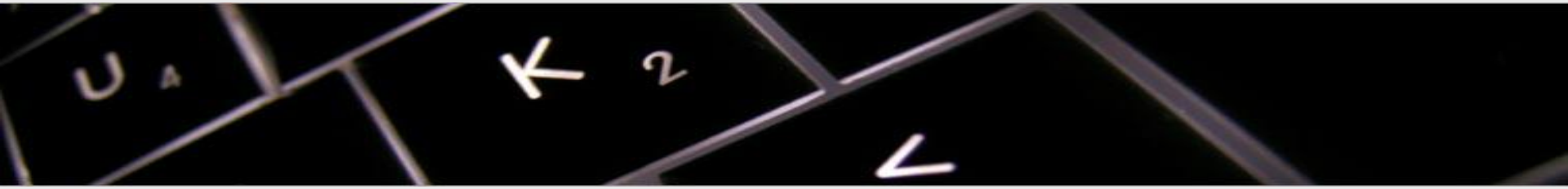


Основне машинске инструкције
STOP



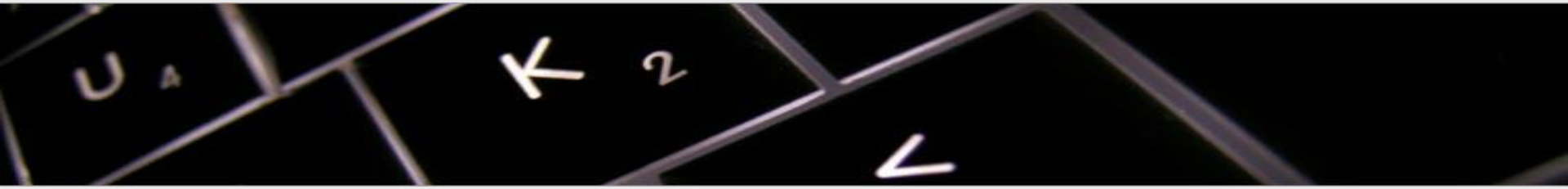
КРАЈ РАДА	1111 STOP	Престанак инкрементирања PC-а. Ако је садржај адресних поља различит од 0000, онда ће, на екрану терминала, пре завршетка рада, бити приказане крајње вредности одговарајућих аргумената (овако није могуће приказати крајњи садржај локације 0).

Синтакса - Спецификација извршних инструкција STOP



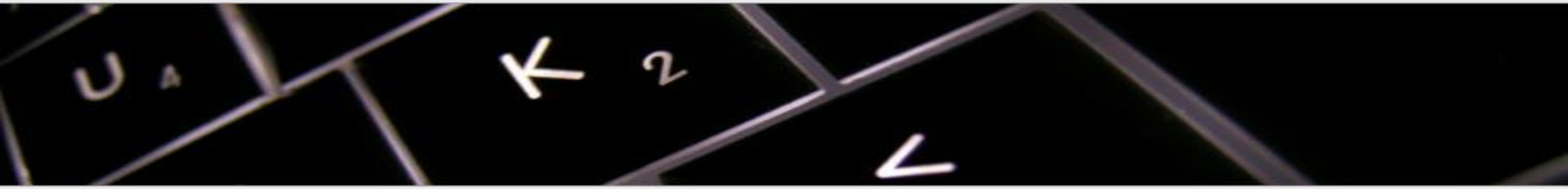
$$[simbol:]\ STOP\ \left[\left\{\begin{matrix} simbol \\ (simbol) \end{matrix}\right\}\left[,\left\{\begin{matrix} simbol \\ (simbol) \end{matrix}\right\}\left[,\left\{\begin{matrix} simbol \\ (simbol) \end{matrix}\right\}\right]\right]\right]\ [;komentar]$$

Пример – Еуклидов алгоритам

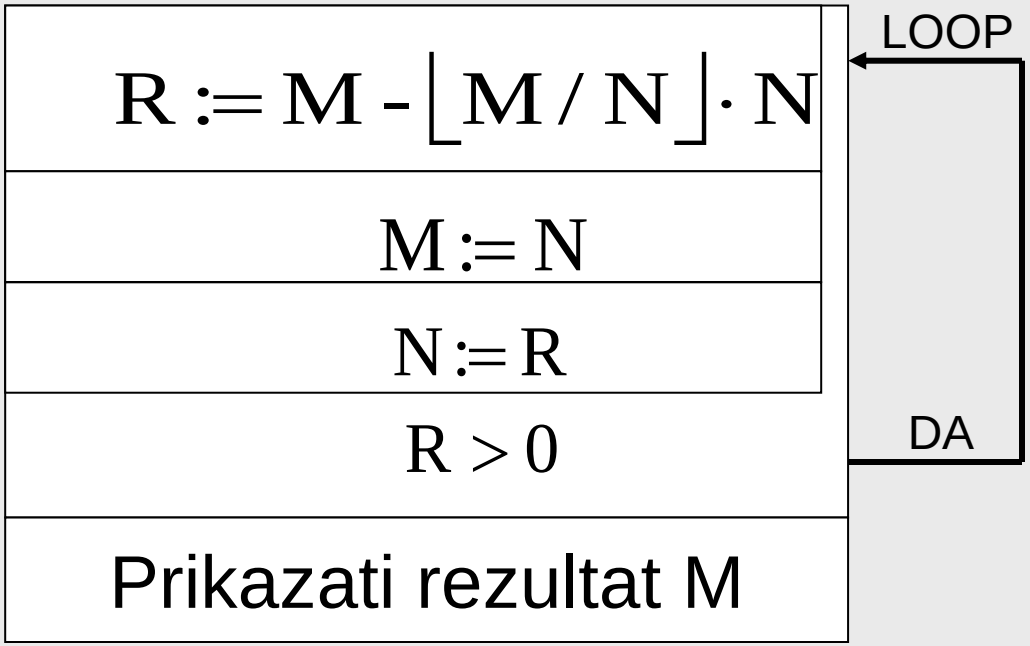


Написати програм који налази највећи заједнички фактор два различита цела броја M и N, где је M > N користећи Еуклидов алгоритам

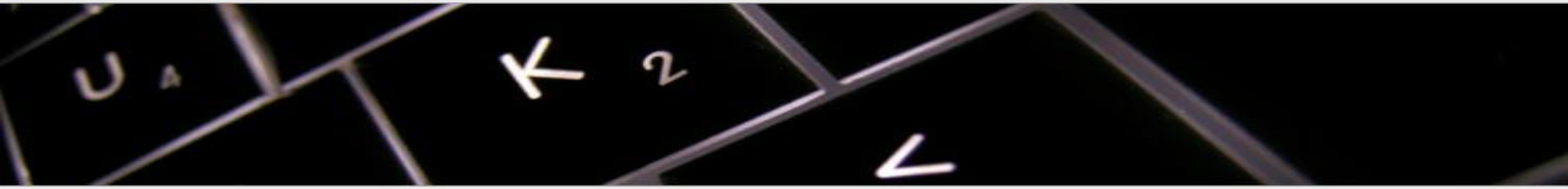
Пример – Еуклидов алгоритам



M	N	R
25	10	5
10	5	0
5	0	

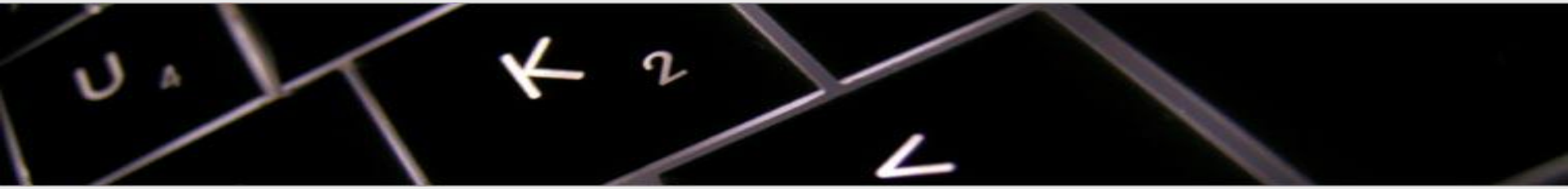


Пример – Еуклидов алгоритам



```
        M = 1               ; Adresa za M
        N = 2               ; Adresa za N
        R = 3               ; Adresa za R
        ORG 8               ; Pocetak programa
        IN    M, 2          ; Ucitavanje M i N
LOOP:DIV   R, M, N          ; R := |_ M / N _|
        MUL  R, R, N        ; R := |_ M / N _|*N
        SUB  R, M, R        ; R := M mod N
        MOV M, N            ; M := N
        MOV N, R            ; N := R
        BGT R, 0, LOOP      ; ako R>0 idi na LOOP
        STOP M
```

Пример



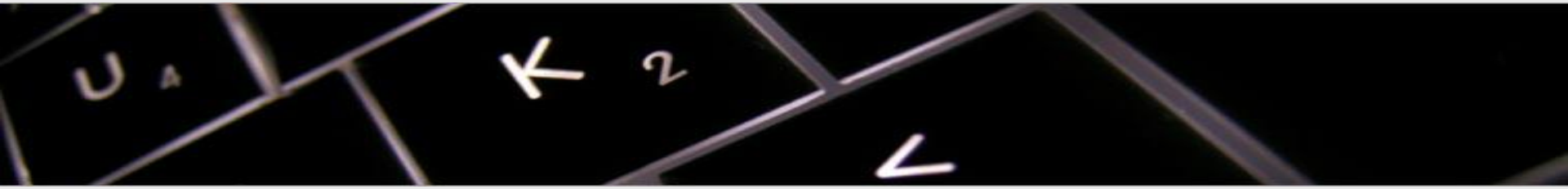
Шта треба да стоји уместо *** да би приложени програм за picoComputer исписао **1 3**?

```
A=4     MOV B, #C          ;(1)
B=6     MOV D, #A          ;(2)
C=2     MOV (B), D         ;(3)
D=5     MOV (C), #B        ;(4)
E=1     DIV (A), A, B      ;(5)
F=3     SUB (B), C, B      ;(6)
G=0     SUB (C), (C), (A)  ;(7)
ORG 8   ***
```

0		G
1		E
2	4 **(3)**	C
3	1 **(6)**	F
4	~~6 **(4)**~~ 3 **(7)**	A
5	4 **(2)**	D
6	~~2 **(1)**~~ 3 **(5)**	B

(A) OUT (A), #C **B) OUT F, #F** **(C) OUT (B), 2**

Пример



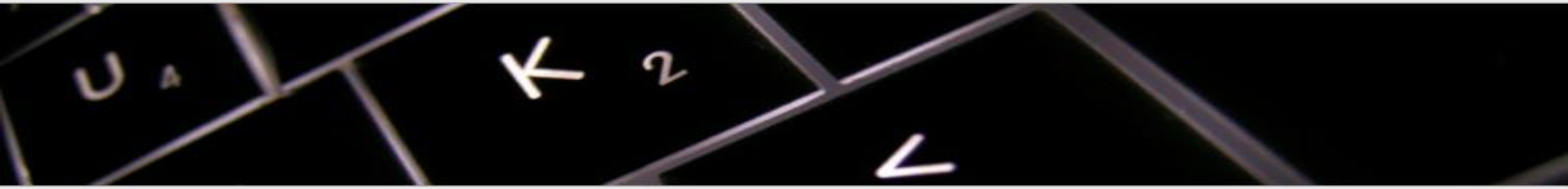
Приложени програм за picoComputer исписује произвољан учитани низ дужине N>1:

```
N = 1       IN    N
adrA = 2    MOV   adrA, #A
adrK = 3    IN    (adrA), N
adrP = 4    MOV   adrP, adrA
T = 5       ADD   adrK, adrA,N
A = 100
ORG  8
```

```
P:SUB adrK, adrK, 1
    MOV T,(adrK)
    MOV (adrK), (adrP)
    MOV (adrP), T
    ADD adrP, adrP, 1
    BGT adrK, adrP, P
    OUT (adrA), N
    STOP
```

A) сортиран **B) неизмењен** **(C) обрнутим редом**

Пример

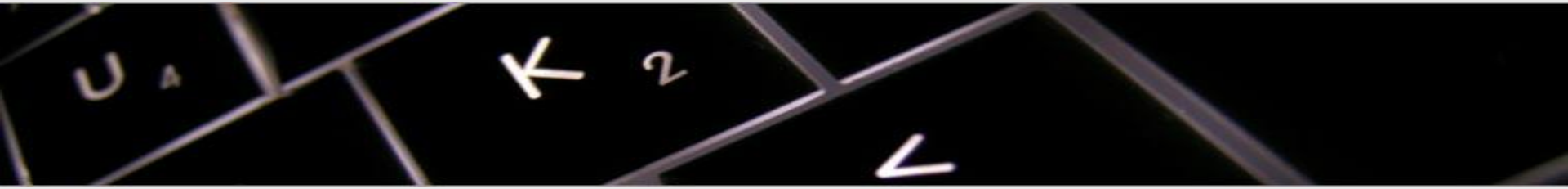


```
N = 1
adrA = 2
adrK = 3
adrP = 4
T = 5
A = 100
ORG  8
IN      N
MOV     adrA, #A
IN      (adrA), N
MOV     adrP, adrA
ADD     adrK, adrA,N
P:  SUB adrK, adrK, 1
    MOV T,(adrK)
    MOV (adrK), (adrP)
    MOV (adrP), T
    ADD adrP, adrP, 1
    BGT adrK, adrP, P
    OUT (adrA), N
    STOP
```

Unos: 3 5 2 8
Ispis: 8 2 5

0		
1	3	N
2	100	adrA
3	~~103~~ ~~102~~ 101	adrK
4	~~100~~ ~~101~~ 102	adrP
5	~~8~~ 2	T
…	…	
100	~~5~~ 8	A
101	2	
102	~~8~~ 5	

Пример

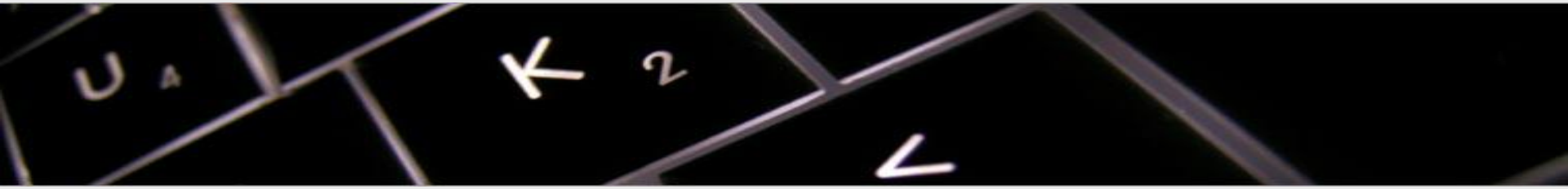


Следећи програм на симболичком машинском језику за picoComputer, за позитивне целобројне вредности A и B, израчунава и исписује

```
    A=1                          JSR   Q
    B=2                          STOP  X
    X=3                    Q:    SUB   N,N,1
    N=4                          BEQ   N,0,KRAJ
    ORG 8                        MUL   X,X,B
    IN    A,2                    JSR   Q
    MOV   X,1                    KRAJ: RTS
    MOV   N,A
```

A) $A*B$ B) A^B C) B^A

Пример



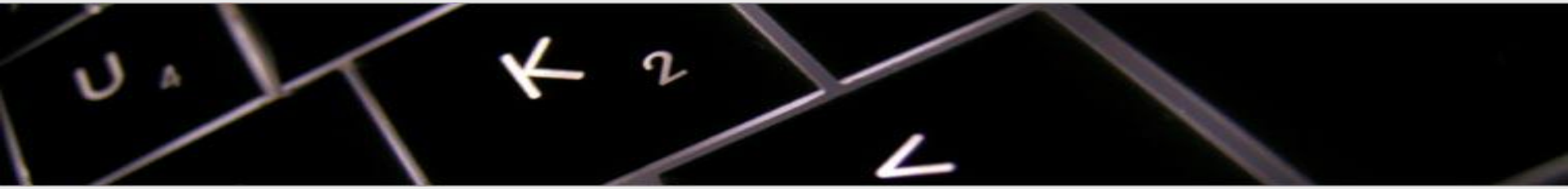
```
A=1
B=2
X=3
N=4
ORG 8
IN    A,2
MOV   X,1
MOV   N,A
      JSR   Q
POV:  STOP  X
Q:    SUB   N,N,1
      BEQ   N,0,KRAJ
      MUL   X,X,B
      JSR   Q
KRAJ: RTS
```

Unos: 3 4
Ispis: 16
$16=4x4=BxB=B^{A-1}$

STEK

0		
1	3	A
2	4	B
3	~~1~~ ~~4~~ 16	X
4	~~3~~ ~~2~~ ~~1~~ 0	N
…	…	
	~~KRAJ~~	
	~~KRAJ~~	
	~~POV~~	

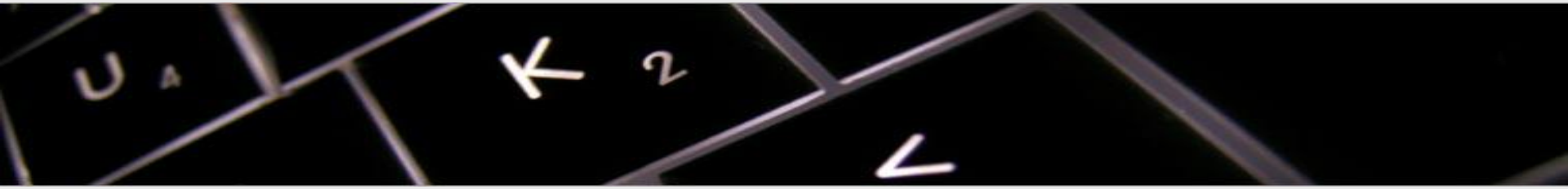
Упутство за коришћење алата MessyLab



Почетак рада: треба инсталирати алат MessyLab. Алат садржи преводилац за симболички машински језик pC-а и симулатор pC-а. Принцип рада: сличан модерним развојним окружењима, попут MS Visual Studio-а. Алат се налази на сајту http://messylab.com/

Једноставно Веб окружење: https://picosim.app/

Архитектура и први pCas



UNIVERSITY UPDATES **Modified campus operations continue**

Read campus messages and details.

San Francisco State University | A–Z | Calendar | Login | Search

Department of Computer Science

People Research Resources Employment

Jozo Dujmovic

Contact Information

Phone: (415) 338-2207
Email: jozo@sfsu.edu
Building: Thornton Hall
Room Number: 946

Office hours will be posted on the respective iLearn page and held through Zoom. Please contact the office if you are unable to find instructions for office hours for a particular class.

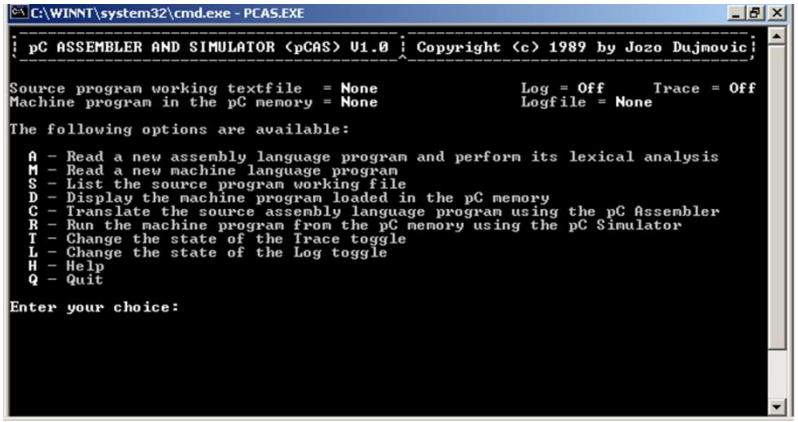
Advising Hours

Friday: 2:00 p.m. - 3:00 p.m.

https://sfsu.zoom.us/meeting/register/tJcrcOmoqT8uHNzsAz0vT3dwSw8xIBjyJsdE

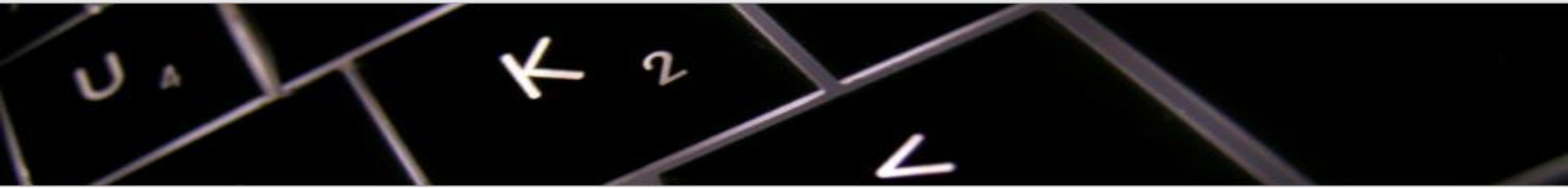
Areas of Expertise

- Soft Computing
- Decision Support Systems
- Benchmarking
- Software Metrics
- Computer Performance Evaluation



Professor
Undergraduate Advisor

Алат MessyLab



oo1p1 - Messy Lab

File Edit View Project Debug Help

Project Explorer

Add Remove

Main	Filename
*	Main.pca

Breakpoints

New Breakpoint Delete

Description	Flags

Main.pca

```
1          z = 0
2          a = 1
3          b = 2
4          c = 3
5          d = 4
6          e = 5
7          f = 6
8          g = 7
9
10         ORG 8 ; TODO: Adjust to make room for static data.
11
12         ; Program entry point
13
14         stop
15
```

8: F000

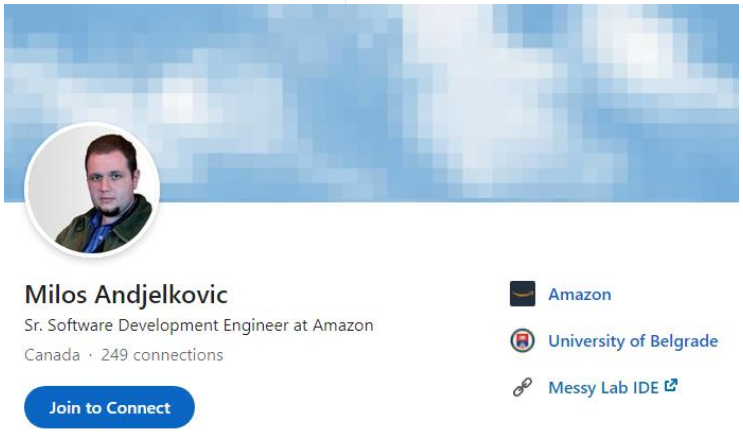
Hardware

Item	Value
Registers	
PC	?
SP	?
Current Instruction	
Symbolic	?
Instruction	?
Constant	?

Watch

New Watch Delete

Name	Value	Type



Auto-assemble Ln 1 Col 1 INS

Task List Error List Console

Веб окружење picosim

https://picosim.app/

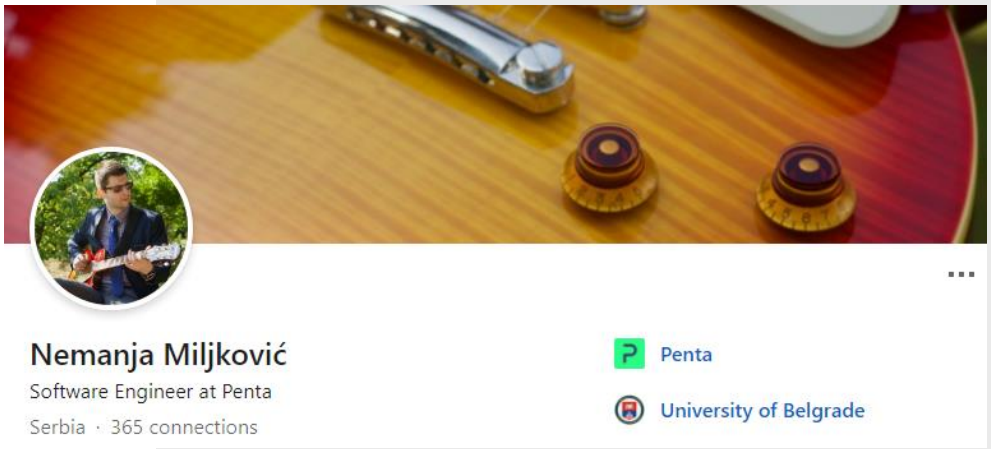
picoComputer Web IDE

EndNote 3.13 Capture Help

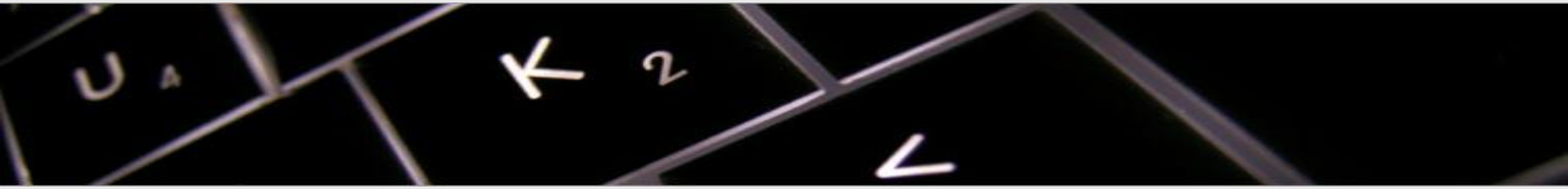
Page Safety Tools

pico

```
1        X = 2
2        N = 3
3        P = 4
4
5
6        ORG 8
7        IN X
8        SUB N,N,N
9    L1: DIV P,X,2
10       MUL P,P,2
11       SUB P,X,P
12       DIV X,X,2
13       BEQ P,0,L2
14       ADD N,N,1
15       BEQ X,X,L1
16   L2: STOP N
17
```



MessyLab - Главни MENU



File

Edit

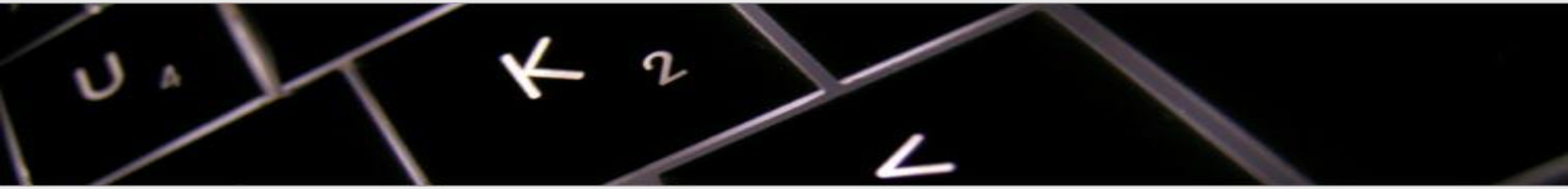
View

Project

Debug

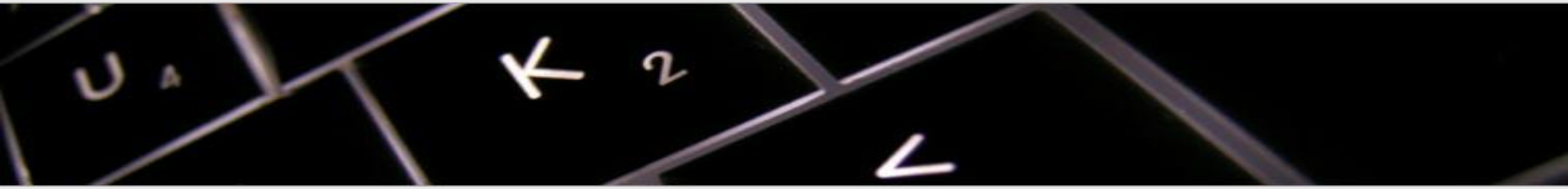
Help

Главни MENU – опција File



Учитавање програма за pC (симболички машински језик).
PCA је стандардни тип за датотеке са изворним текстом (ime_prog.PCA).
New Project, Open Project, Close, Close Project, Save, Save All, Print, Recent Projects, Exit

Главни MENU – опција Edit

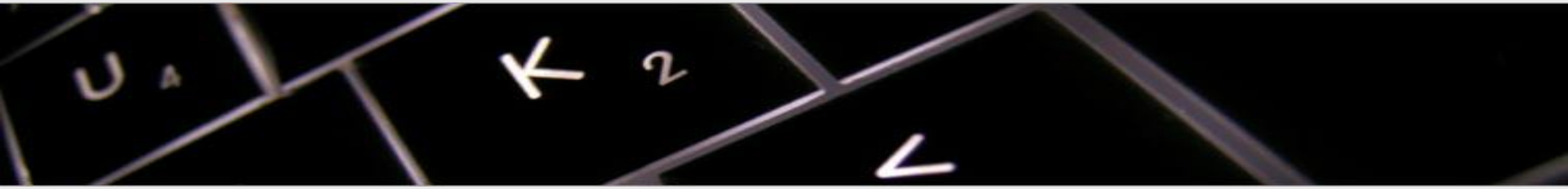


Undo, Redo, Cut, Copy, Paste

Select All, Find and Replace

Go to, Advanced, Options

Главни MENU – опција View



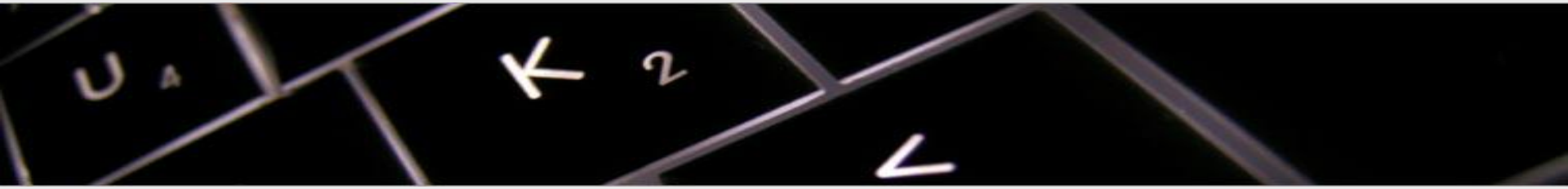
Project Explorer

Error List, Task List

Watch, Hardware

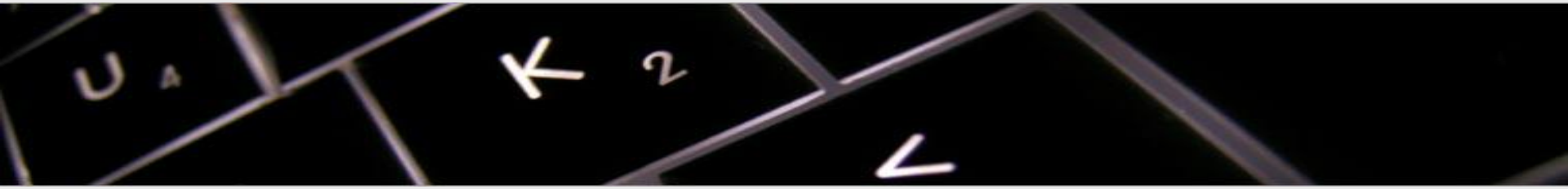
Breakpoints, Console

Формат датотеке



aaaa	Почетна адреса програма (hex)
iiii	Сукцесивне машинске речи програма (hex)
...	
iiii	

Главни MENU – опција Project

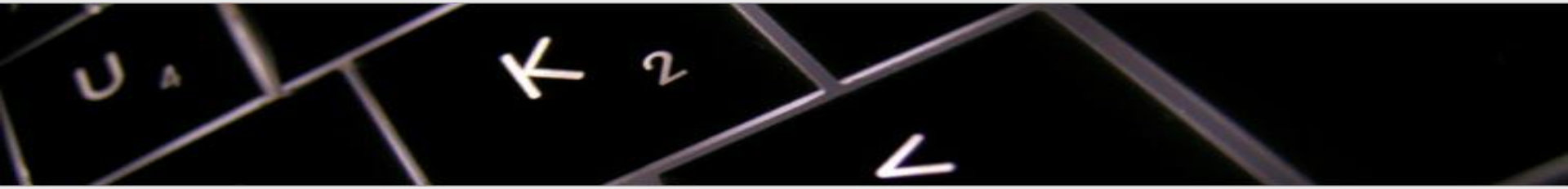


Add New File, Add Existing File

Build, Run

Properties

Главни MENU – опција Debug



Debug, Pause, Stop, Restart

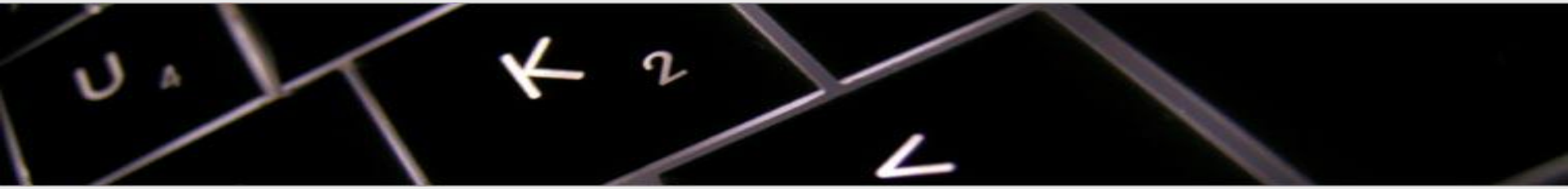
Step Into, Step Over, Step Out

Step Back Into, Step Back Over,

Step Back Out

Toggle Breakpoint

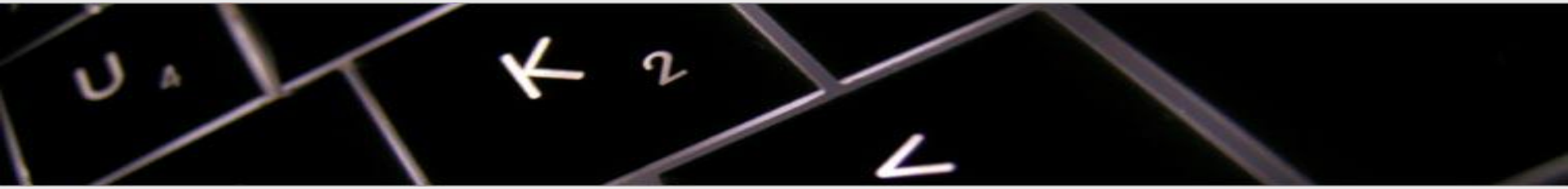
Главни MENU – опција Help



Online Docs

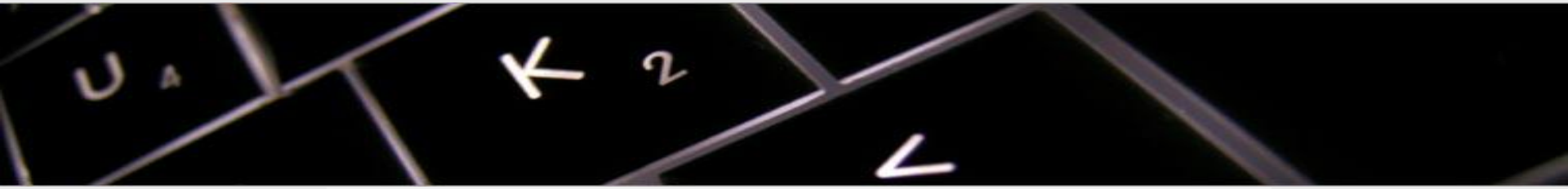
About

Уобичајени редослед



MessyLab	СТАРТОВАЊЕ АЛАТА
File, New Project	УНОШЕЊЕ ИЗВОРНОГ КОДА
Project, Build	ПРЕВОЂЕЊЕ ИЗВОРНОГ ПРОГРАМА
Debug	ДЕБАГОВАЊЕ ПРОГРАМА
Project, Run	ИЗВРШАВАЊЕ ПРЕВЕДЕНОГ ПРОГРАМА
File, Exit	КРАЈ РАДА (КРАЈ КОРИШЋЕЊА MessyLab-А)

Пример

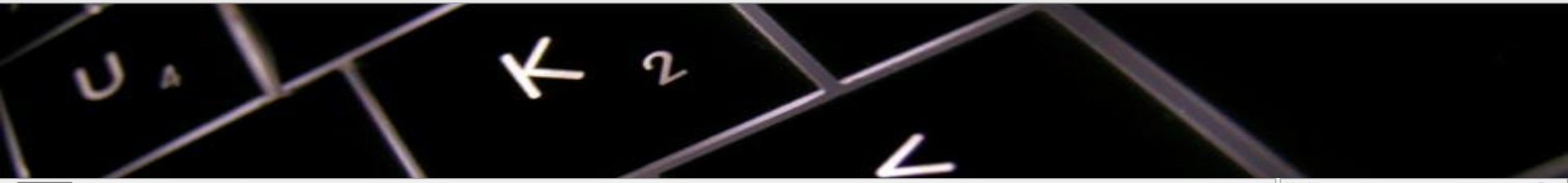


Main.pca

```
 8: 7201
 9: 2333
10: C420 0002
12: B440 0002
14: 2424
15: C220 0002
17: 5408 0017
19: 9330 0001
21: 5228 000A
23: F300
```

```
1          X = 2
2          N = 3
3          P = 4
4
5          ORG 8 ;
6
7          IN X
8          SUB N, N, N
9    L1:   DIV P, X, 2
10         MUL P, P, 2
11         SUB P, X, P
12         DIV X, X, 2
13         BEQ P, 0, L2
14         ADD N, N, 1
15         BEQ X, X, L1
16   L2:   STOP N
17
18
```

Пример



Main.pca

```
 8: 7201
 9: 2333
10: C420 0002
12: B440 0002
14: 2424
15: C220 0002
17: 5408 0017
19: 9330 0001
21: 5228 000A
23: F300
```

```
1         X = 2
2         N = 3
3         P = 4
4
5         ORG 8 ;
6
7         IN X
8         SUB N, N, N
9   L1:   DIV P, X, 2
10        MUL P, P, 2
11        SUB P, X, P
12        DIV X, X, 2
13        BEQ P, 0, L2
14        ADD N, N, 1
15        BEQ X, X, L1
16  L2:   STOP N
17
18
```

Hardware

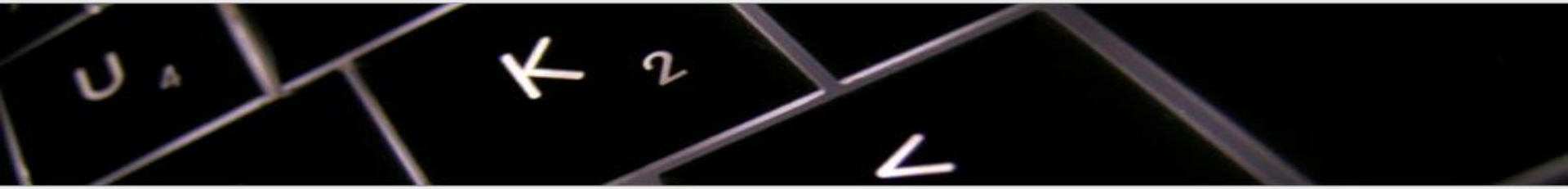
Item	Value
Registers	
PC	10
SP	65535
Current Instruction	
Symbolic	DIV 4, 2, #2
Instruction	1100 0100 0010 0000
Constant	0000000000000010

Watch

New Watch Delete

Name	Value	Type
P	0	Signed

Пример



An integer value for location #2: 20

The contents of memory location #3 = 0

Пример 2019

Zadatak

Napisati program na simboličkom mašinskom jeziku za picoComputer vrši proveru da li drugi učitani niz predstavlja prvi učitani niz u obrnutom poretku. Program najpre učitava dužinu svakog od nizova ($0 \leq N1, N2 \leq 100$), a zatim i elemente oba niza. Ukoliko se za dužinu nekog od nizova unese nekorektna vrednost, program treba da završi svoje izvršavanje. Nakon unosa program vrši zahtevanu obradu i ispisuje vrednost 1, ukoliko drugi učitani niz predstavlja prvi učitani niz u obrnutom poretku, vrednost 0 ukoliko ne predstavlja, a vrednost 2 ukoliko dužine učitanih nizova nisu jednake. Proveru da li jedan niz predstavlja drugi niz u obrnutom poretku realizovati kao zaseban potprogram.

```
n1 = 1
n2 = 2
start1 = 100
start2 = 200
tmp = 3
a1 = 4
a2 = 5
result = 6
i = 7
org 8
in n1
in n2
bgt 0, n1, error
bgt 0, n2, error
mov tmp, 100
bgt n1, tmp, error
bgt n2, tmp, error
mov a1, #start1
mov a2, #start2
in (a1), n1
in (a2), n2
mov result, 2
bgt n1, n2, print
bgt n2, n1, print
sub result, result, 1
jsr pp
print:  out result
error:  stop
pp:     mov i, n1
        add a2, a2, n1
        sub a2, a2, 1
loop:   bgt (a1), (a2), end
        bgt (a2), (a1), end
        add a1, a1, 1
        sub a2, a2, 1
        sub i, i, 1
        bgt i, 0, loop
end:    beq i, 0, ret
        sub result, result, 1
ret:    rts
```

Пример - поправни 2019

Zadatak

Napisati program na simboličkom mašinskom jeziku za picoComputer koji vrši određenu obradu nad nizom celih brojeva. Program treba najpre da učita dužinu niza N ($0 < N \leq 100$), a zatim i same elemente niza. Ukoliko se za dužinu niza unese nekorektna vrednost, program treba da završi svoje izvršavanje. Za svaki element niza izračunava se proizvod njegovih cifara (npr. za broj 357 proizvod cifara je 3*5*7=105). Zatim se izračunata vrednost upisuje u niz umesto elementa za koji je izračunat proizvod (npr. proizvod 105 se upisuje na mesto broja 357). Nakon obrade, ispisuje se izmenjeni niz. Izračunavanje proizvoda cifara decimalnog broja realizovati kao zaseban potprogram.

Primer ulaznog niza za N = 6	Primer izlaza
357 12 9 4321 503 89	105 2 9 24 0 72

```
A = 100
N = 1
adrA = 2
T = 3
I = 4
P = 5
K = 6
O = 7
ORG 8
IN N
MOV adrA, #A
IN (adrA), N
MOV T, 1
BGT T, N, KRAJ
MOV T, 100
BGT N, T, KRAJ
MOV I, 0
PETLJA_1:   JSR PP
            MOV (adrA), P
            ADD adrA, adrA, 1
            ADD I, I, 1
            BGT N, I, PETLJA_1
            MOV adrA, #A
            OUT (adrA), N
KRAJ:       STOP

PP:     MOV P, 1
            MOV K, (adrA)
PETLJA_2:   DIV T, K, 10
            MUL T, T, 10
            SUB O, K, T
            MUL P, P, O
            DIV K, K, 10
            BGT K, 0, PETLJA_2
            RTS
```