MIPS (tio.run / #spim)

1) Afișare mesaj / caracter

```
.data
    m: .asciiz / .byte "Hello world"

.text
    main:
        li $v0, 4
        la $a0, m
        syscall

        li $v0, 10
        syscall
```

---

2) Afișare întreg

```
.data
    n: .word 9

.text
    main:
        li $v0, 1
        lw $a0, n
        syscall

        li $v0, 10
        syscall
```

---

3) Adunarea a 2 întregi:

```
.data
    x: .word 5
    y: .word 6
```

```
.text
main:
    lw $t0, x
    lw $t1, y
    add $t2, $t0, $t1
    li $v0, 1
    move $a0, $t2
    syscall
    li $v0, 10
    syscall
```

---

4) Interschimbarea a 2 nr:

```
.data
    x: .word 5
    y: .word 9
    str: .asciiz " "
.text
main:
    lw $t0, x
    lw $t1, y
    move $t2, $t0
    move $t0, $t1
    move $t1, $t2
    li $v0, 1
    move $a0, $t0
    syscall
    li $v0, 4
    la $a0, str
```

```
syscall
li $v0, 1
move $a0, $t1
syscall
li $v0, 10
syscall
```

5) Citire de la tastatură întreg:

```
.data
n: .space 4
.text
main:
li $v0, 5
syscall
move $t0, $v0
li $v0, 1
move $a0, $t0
syscall
li $v0, 10
syscall
```

6) Afișare numere de la 0 la n-1, cu n dat de la tastatură

```
.data
n: .space 4
str: .asciiz " "
.text
main:
```

```
li $v0,5
syscall
move $t0,$v0
li $t1,0 ⟶ pe post de i
loop: beq $t0,$t1,exit
      li $v0,1
      move $a0,$t1
      syscall
      li $v0,4
      la $a0,str
      syscall
      add $t1,$t1,1
      j loop
exit: li $v0,10
      syscall
```

dacă $x_0 = x_2$:
exit

---

7) Afișare divizori p și i ai lui n dat de la tastatură + salvare în memorie n

```
.data
    n: .space 4
    s: .asciiz " "
.text
main:
    li $v0,5
    syscall
    move $t0,$v0
    sw $t0,n
```

```
li $t1, 1

loop: bgt $t1, $t0, exit
      rem $t2, $t0, $t1
      beq $t2, $zero, afisare

continue:
      add $t1, $t1, 1
      j loop

afisare:
      move $a0, $t1
      li $v0, 1
      syscall
      la $a0, s
      li $v0, 4
      syscall
      j continue

exit:
      li $v0, 10
      syscall
```

---

8) Afisare vector din memorie:

```
.data
v: .word 1, 2, 3, 4, 5        (adrese: 1, 4, 8, 12)
n: .word 5
sp: .asciiz " "

.text
main:
      lw $t0, n
      li $t1, 0
```

```
li $t2, 0
loop: bge $t1, $t0, exit
      lw $a0, v($t2)
      li $v0, 1
      syscall
      li $v0, 4
      la $a0, sp
      syscall
      add $t1, $t1, 1
      add $t2, $t2, 4
      j loop
exit: li $v0, 10
      syscall
```

accesare vector dim 4 in 4

---

g) Citire si afisare vector cu n dat in memorie:
(+ salvare in memorie vector)

```
.data
n:  .word 5
v:  .space 20   (5*4 bytes)
sp: .asciiz " "
.text
main:
    lw $t0, n
    li $t1, 0
    li $t2, 0
```

citire: bge $t1, $t0, exit

li $v0, 5

syscall

sw $v0, v ($t2)

add $t1, 1

add $t2, 4

j citire

exit:

lw $t0, m

li $t1, 0

li $t2, 0

afisare: bge $t1, $t0, exit1

li $v0, 1

lw $a0, v($t2)

syscall

add $t1, $t1, 1

add $t2, $t2, 4

j afisare

exit1: li $v0, 10

syscall

---

10) Se dă o matrice în memorie:
Să se afișeze.

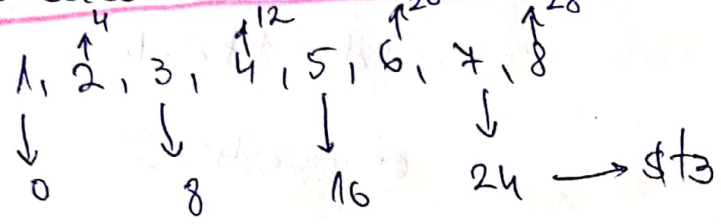
.data

v: .xlord 1, 2, 3, 4

.xlord 5, 6, 7, 8

```
n: .word 2
m: .word 4
sp: .asciiz " "
end: .asciiz "\n"
```

Așa arată în memorie:




```
.text
main:
    lw $t0,n
    lw $t1,m
    lw $t2,0
    lw $t3,0
loop: bge $t2, $t0, exit
    li $t4,0
loop1: bge $t4, $t1, exit1
    lw $a0,v($t3)
    li $v0,1
    syscall
    la $a0, sp
    li $v0,4
    syscall
    addi $t4,1
    addi $t3,4
    j loop1
```

```
exit1 : la $a0, end
        li $v0,4
        syscall
        addi $t2,1
        addi $t3,4
        j loop

exit :  li $v0,10
        syscall
```

---

11) Se citește un șir de max 99 caractere. Să se afișeze pe ecran caracterele situate pe poziții pare (indexate de la 0)

```
.data
    str: .space 100
    sp:  .asciiz " "

.text
main:
    la $a0, str
    li $a1, 99
    li $v0, 8
    syscall

    li $t0,0
    lb $t1, str($t0)
loop: beqz $t1, exit
    rem $t2, $t0, 2
    beqz $t2, 0, afisare
```

adresa de memorie

lungime maximă

cod READ STR

Pentru parcurgere șiruri de caractere

```
continue:
        addi $t0, 1
        lb   $t1, str($t0)
        j loop

afisare: lb $a0, str($t0)
         li $v0, 11
         syscall
         la $a0, sp
         li $v0, 4
         syscall
         j continue

exit:   li $v0, 10
        syscall
```

---

12) Se dă în memorie un șir de caractere. Să se adauge +1 fiecărui caracter (pe codul ascii) => modificare în memorie

ex: abc xyz → bcd!xyz

```
.data
str: .asciiz "Ana"
.text
main:
    li $t0, 0
    lb $t1, str($t0)
```

```
loop:   beqz $t1, exit
        addi $t1,1
        sb $t1, 1th($t0)
        addi $t0, 1
        lb $t1, 1th($t0)
        j loop

exit:   la $a0, 1th
        li $v0, 4
        syscall
        li $v0, 10
        syscall
```

PROCEDURI

13) Creare procedura "afis" care să afișeze un mesaj dat în memorie.

```
.data
    m: .asciiz "hello!"
.text
main:
    jal afis
    li $v0,10
    syscall
afis:
    li $v0,4
    la $a0,m
    syscall
    jr $ra
```

Funcție fără parametrii

14) Calculare suma a 2 nr x și y date în mem. Folosire proceduri cu registrii argument $a0 → $a3

```
.data
x: .word 5
y: .word 7
.text
main:
    lw $a0, x
    lw $a1, y
    jal suma
```

transmitere parametrii prin registrii argument $a0 → $a3

```
li $v0, 1
move $a0, $v1
syscall
li $v0, 10
syscall
```

$v pt rezultate și coduri sistem

```
suma:
    add $v1, $a0, $a1
    jr $ra
```

salvare rezultat de afișat în $v0 → $v1

---

15) Calculare suma a 2 nr date în memorie, folosind procedură cu lucrul pe STIVĂ.

```
.data
x: .word 5
y: .word 10
.text  # aici se pot pune procedurile
```

suma:
aici stiva este $sp: (x)(y)

MEREU
- subu $sp,4 # $sp: ()(x)(y)
- sw $fp,0($sp) # $sp: ($fp v)(x)(y)
- addi $fp,$sp,4 # $sp: ($fp v) $fp:(x)(y)

Convenție:
ca să pot să lucrez cu registrii $s0, $s1... trebuie salvați inițial pe stivă
- subu $sp,4 # $sp: ()$fp v # fp: (x)(y)
- sw $s0,0($sp) # sp: ($s0)($fp v) fp:(x)(y)
- subu $sp,4
- sw $s1,0($sp) # sp: ($s1)($s0)($fp v) fp(x)(y)

(12 = 4 + 8)

Le modific valoarea cum vreau eu
- lw $s0,0($fp) # în $s0 pe x
- lw $s1,4($fp) # $s1 pe y

În vo vreau rezultatul
- # fac suma p- ziua
- add $v0,$s0,$s1

terminare
- lw $s1,-12($fp)
- lw $s0,-8($fp)
- lw $fp,-4($fp)

Ștergere: $s0, $s1, $fp v
(le-am adăugat pe lângă lista inițială)

# șterg ce e 3 element adaug
addu $sp,12

jr $ra

main:
# încărcare y (în ord inversă)
lw $t0, y # îl bag în reg

```
subu $sp, 4    # îi fac loc pe stivă
sw  $t0, 0($sp)  # îl salvez pe stivă
# înmulțirea lui x
lw  $t0, x
subu $sp, 4
sw  $t0, 0($sp)

#apel
jal suma

# dezalocare spațiu
addu $sp, 8

# în v0 avem rezultatul de afișat
move $a0, $v0
li $v0, 1
syscall

# exit
li $v0, 10
syscall
```

16) Creare procedură care să afișeze vectorul v dat în memorie.

```
.data
    v: .word 1, 2, 3
    n: .word 3
    sp: .~~asciiz~~ byte ''
.text
```

main:

```
# incarcare n pe stiva
lw $t0, n
subu $sp, 4
sw $t0, 0($sp)  # sp : (n)

# incarcare vector pe stiva
la $t0, v
subu $sp, 4
sw $t0, 0($sp)  # sp : (v)(n)

#apel
jal afis;
#curatare stiva
addi $sp, 8

li $v0, 10
syscall
```

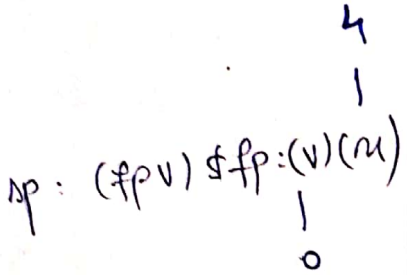
afis:

```
subu $sp, 4
sw $fp, 0($sp)
addi $fp, $sp, 4   # sp : (fp v) fp:(v)(n)

subu $sp, 4
sw $a0, 0($sp)

sub $sp, 4
sw $a1, 0($sp)
# sp : (a1)(a0)(fp v) fp:(v)(n)

lw $a0, 0($fp)
lw $a1, 4($fp)
```



```
li $t0, 0
loop: bge $t0, $s1, exit
      lw  $a0, 0 ($s0)
      li $v0, 1
      syscall
      lb $a0, sp
      li $v0, 11
      syscall
      addi $s0, 4
      addi $t0, 1
      j loop
exit :
      lw1 $ s1, -12($fp)
      lw1 $ s0, -8($fp)
      lw1 $ fp, -4($fp)
      addi $ sp, 12
      jr $ra
```

Realizat de Dima Oana Teodora 1141
Ianuarie 2020