



**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS**

Instituto de Ciências Exatas e Informática – ICEI

Departamento de Ciência da Computação

## **TRABALHO PRÁTICO 06**

Belo Horizonte

2022

# **PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS**

Ciência da Computação

Hernane Velozo Rosa

[hernane.rosa@sga.pucminas.br](mailto:hernane.rosa@sga.pucminas.br)

Este trabalho apresenta os códigos dos  
exercícios práticos propostos.

Professor: Dr. Romanelli Lodron Zuim

Disciplina: Arquitetura de Computadores II

Palavras-chaves: Instruções, ULA,  
programação de baixo-nível e MIPS.

# RELATÓRIO: CÓDIGOS E EXECUÇÃO

## Programa 09 – Código

```
# Programa 09 - By Hernane V | Data: 04/12/2022
# Arquitetura de Computadores II - Romanelli
.text
.globl main

main:
    addi $t0, $zero, 0x1001    # $t0 = 0x1001
    sll $t0, $t0, 16           # $t0 = 0x10010000

    lw $s0, ($t0)              # $s0 = Mem[10010000]
    lw $s1, 4($t0)              # $s1 = Mem[10010000 + 4]
    lw $s2, 8($t0)              # $s2 = Mem[10010000 + 8]
    lw $s3, 12($t0)             # $s3 = Mem[10010000 + 12]
    lw $s4, 16($t0)             # $s4 = Mem[10010000 + 16]

    add $t1, $s0, $s1           # $t1 = $s0 + $s1
    add $t1, $t1, $s2           # $t1 = $t1 + $s2
    add $s4, $t1, $s3           # $t1 = $t1 + $s3

    sw $s4, 16($t0)             # Mem[10010000 + 16] = $s4

.data
x1: .word 15
x2: .word 25
x3: .word 13
x4: .word 17
soma: .word -1
```

## Programa 09 – Execução via MARS

The screenshot displays the MARS MIPS simulator interface. The main window shows the assembly code for Program 09, with the following instructions highlighted in yellow:

```
addi $t0, $zero, 0x1001
sll $t0, $t0, 16
lw $s0, ($t0)
lw $s1, 4($t0)
lw $s2, 8($t0)
lw $s3, 12($t0)
lw $s4, 16($t0)
add $t1, $s0, $s1
add $t1, $t1, $s2
add $s4, $t1, $s3
sw $s4, 16($t0)
```

The register window on the right shows the values of the registers. The \$t0 register contains 0x10010000, and the \$s4 register contains 0x10010000. The data segment window at the bottom shows the memory layout, with the address 0x10010000 containing the value 15, 0x10010004 containing 25, 0x10010008 containing 13, and 0x1001000C containing 17. The message window at the bottom shows the message "Reset: reset completed."

## Programa 10 – Código

```
# Programa 10 - By Hernane V | Data: 04/12/2022
# Arquitetura de Computadores II - Romanelli
```

```
# y = 127x - 65z + 1
# x ? $s0
# y ? $s1
# z ? $s2
```

```
.text
.globl main
```

```
main:
    addi $t0, $zero, 0x1001    # t0 = 0x1001
    sll $t0, $t0, 16           # t0 = 0x10010000
    lw $t1, 0($t0)             # t1 = Mem[10010000 + 0]
    lw $t2, 4($t0)             # t2 = Mem[10010000 + 4]

    sll $t3, $t1, 7            # t3 = 128x
    sub $t3, $t3, $t1          # t3 = 127x
    sll $t4, $t2, 6            # t4 = 64z
    add $t4, $t4, $t2          # t4 = 65z
    addi $t5, $zero, 1         # t5 = 1
    sub $t6, $t3, $t4          # t6 = 127x - 65z

    add $s2, $t6, $t5          # y = 127x - 65z + 1
    sw $s2, 8($t0)             # Mem[10010000 + 8] = 127x - 65z + 1
```

```
.data
x: .word 5
z: .word 7
y: .word 0
```

## Programa 10 – Execução via MARS

The screenshot shows the MARS MIPS simulator interface. The main window displays the assembly code for Program 10, with comments explaining the operations. The right panel shows the registers, and the bottom panel shows the data segment and memory values.

**Registers:**

Register	Name	Number	Value
\$zero		0	0
\$a0		1	0
\$a1		2	0
\$a2		3	0
\$a3		4	0
\$a4		5	0
\$a5		6	0
\$a6		7	0
\$a7		8	0
\$s0		9	268500992
\$s1		10	5
\$s2		11	7
\$s3		12	635
\$s4		13	455
\$s5		14	1
\$s6		15	180
\$s7		16	0
\$s8		17	0
\$s9		18	181
\$s10		19	0
\$s11		20	0
\$s12		21	0
\$s13		22	0
\$s14		23	0
\$s15		24	0
\$s16		25	0
\$s17		26	0
\$s18		27	0
\$s19		28	268460324
\$s20		29	2147479548
\$s21		30	0
\$s22		31	0
\$t0		32	4194352
\$t1		33	0
\$t2		34	0

**Data Segment:**

Address	Value (+0)	Value (+0)	Value (+0)	Value (+0)	Value (+0)	Value (+0)	Value (+0)	Value (+0)
0x10010000	5	7	181	0	0	0	0	0
0x10010004	0	0	0	0	0	0	0	0
0x10010008	0	0	0	0	0	0	0	0
0x1001000C	0	0	0	0	0	0	0	0
0x10010010	0	0	0	0	0	0	0	0
0x10010014	0	0	0	0	0	0	0	0
0x10010018	0	0	0	0	0	0	0	0
0x1001001C	0	0	0	0	0	0	0	0
0x10010020	0	0	0	0	0	0	0	0
0x10010024	0	0	0	0	0	0	0	0
0x10010028	0	0	0	0	0	0	0	0
0x1001002C	0	0	0	0	0	0	0	0
0x10010030	0	0	0	0	0	0	0	0
0x10010034	0	0	0	0	0	0	0	0
0x10010038	0	0	0	0	0	0	0	0
0x1001003C	0	0	0	0	0	0	0	0

**Messages:**

```
Reset: reset completed.
-- program is finished running (dropped off bottom) --
-- program is finished running (dropped off bottom) --
```

## Programa 11 – Código

```
# Programa 11 - By Hernane V | Data: 04/12/2022
# Arquitetura de Computadores II - Romanelli

    # y = x-z+300000

.text
.globl main

main:
    addi $t0, $zero, 0x1001    # $t0 = 0x1001
    sll  $t0, $t0, 16          # $t0 = 0x10010000

    lw   $s0, ($t0)            # x = 100000
    lw   $s1, 4($t0)           # z = 200000

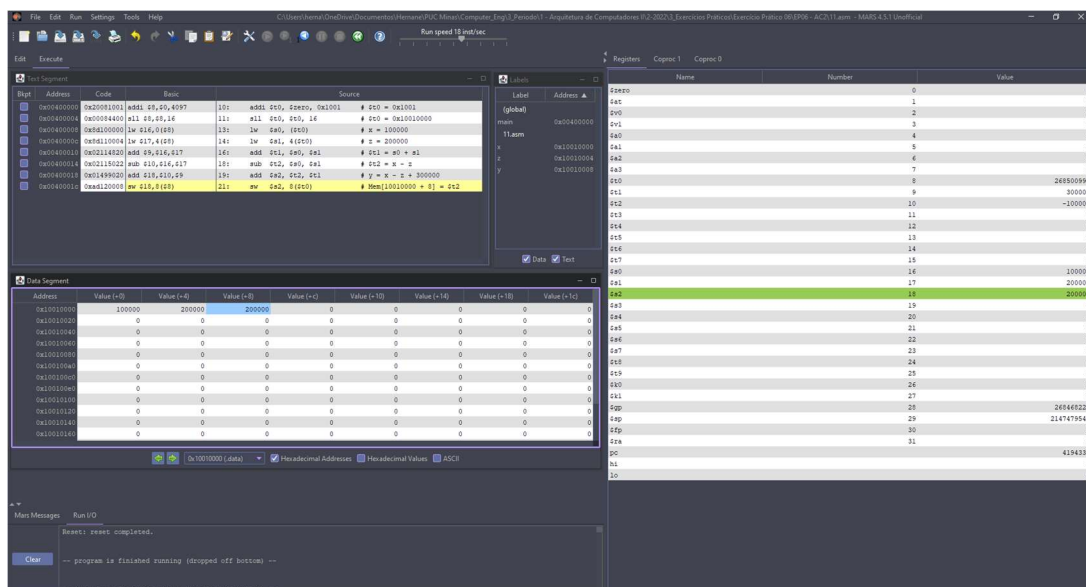
    add  $t1, $s0, $s1         # $t1 = s0 + s1

    sub  $t2, $s0, $s1         # $t2 = x - z
    add  $s2, $t2, $t1         # y = x - z + 300000

    sw   $s2, 8($t0)           # Mem[10010000 + 8] = $t2

.data
x: .word 100000
z: .word 200000
y: .word 0                    # valor a ser sobrescrito
```

## Programa 11 – Execução via MARS



## Programa 12 – Código

```
# Programa 12 - By Hernane V | Data: 04/12/2022
# Arquitetura de Computadores II - Romanelli

.text
.globl main

main:
    ori $t0, $zero, 0x1001
    sll $t0, $t0, 16
    sw $t0, 0($t0)
    ori $t1, $zero, 0x1001

    sll $t1, $t1, 16
    ori $t1, $t1, 0x004

    sw $t1, 0($t1)
    ori $t2, $zero, 0x1001

    sll $t2, $t2, 16
    ori $t2, $t2, 0x0008
    sw $t2, 0($t2)

    or $s0, $zero, $t2
```

## Programa 12 – Execução via MARS

The screenshot displays the MARS MIPS simulator interface. The main window shows the assembly code being executed, with the following instructions visible in the Source panel:

Inst	Address	Code	Basic	Source
0x00400000	0x00400000	ori \$t0, \$zero, 0x1001	T0: ori \$t0, \$zero, 0x1001	
0x00400004	0x00400004	sll \$t0, \$t0, 16	S1: sll \$t0, \$t0, 16	
0x00400008	0x00400008	sw \$t0, 0(\$t0)	S2: sw \$t0, 0(\$t0)	
0x0040000c	0x0040000c	ori \$t1, \$zero, 0x1001	T1: ori \$t1, \$zero, 0x1001	
0x00400010	0x00400010	sll \$t1, \$t1, 16	S3: sll \$t1, \$t1, 16	
0x00400014	0x00400014	ori \$t1, \$t1, 0x004	S4: ori \$t1, \$t1, 0x004	
0x00400018	0x00400018	sw \$t1, 0(\$t1)	S5: sw \$t1, 0(\$t1)	
0x0040001c	0x0040001c	ori \$t2, \$zero, 0x1001	T2: ori \$t2, \$zero, 0x1001	
0x00400020	0x00400020	sll \$t2, \$t2, 16	S6: sll \$t2, \$t2, 16	
0x00400024	0x00400024	ori \$t2, \$t2, 0x0008	S7: ori \$t2, \$t2, 0x0008	
0x00400028	0x00400028	sw \$t2, 0(\$t2)	S8: sw \$t2, 0(\$t2)	
0x0040002c	0x0040002c	or \$s0, \$zero, \$t2	S9: or \$s0, \$zero, \$t2	

The Registers panel on the right shows the state of the MIPS registers. The \$s0 register is highlighted, showing its value as 0x10010004. The Data Segment panel at the bottom shows the memory layout, with the address 0x10010000 highlighted. The Messages panel at the bottom shows the status of the program execution, indicating that the program is finished running.

## Programa 13 – Código

```
# Programa 13 - By Hernane V | Data: 04/12/2022
# Arquitetura de Computadores II - Romanelli

.text
.globl main

    # A → $s0

main:
    lui    $t0, 0x1001          # $t0 = 0x10010000
    lw     $s0, 0($t0)          # $s0 = A

    slt    $t1, $s0, $zero      # se A < 0 → $t1 = 1 então $t1 = 0
    beq    $t1, $zero, fim      # se A < 0 → A=A*(-1)

    addi   $t2, $zero, -1       # $t2 = -1
    mult   $s0, $t2             # lo = A*(-1)
    mflo   $s0                 # A = lo

fim:
    sw     $s0, 0($t0)          # A = $s0

.data
a: .word -359                  # 0x10010000
```

## Programa 13 – Execução via MARS

The screenshot displays the MARS MIPS simulator interface. The main window shows the assembly code for Program 13, which calculates the absolute value of a constant -359 and stores it in register \$s0. The code is as follows:

```
# Programa 13 - By Hernane V | Data: 04/12/2022
# Arquitetura de Computadores II - Romanelli

.text
.globl main

    # A → $s0

main:
    lui    $t0, 0x1001          # $t0 = 0x10010000
    lw     $s0, 0($t0)          # $s0 = A

    slt    $t1, $s0, $zero      # se A < 0 → $t1 = 1 então $t1 = 0
    beq    $t1, $zero, fim      # se A < 0 → A=A*(-1)

    addi   $t2, $zero, -1       # $t2 = -1
    mult   $s0, $t2             # lo = A*(-1)
    mflo   $s0                 # A = lo

fim:
    sw     $s0, 0($t0)          # A = $s0

.data
a: .word -359                  # 0x10010000
```

The simulator shows the following state:

- Registers:** \$s0 contains the value -359 (0xFFFFFFFF).
- Data Segment:** The word at address 0x10010000 contains the value -359.
- Console:** The console shows the message "Reset: reset completed."

### Programa 14 – Código

[illegible]

## Programa 14 – Execução via MARS

[illegible]



## Programa 15 – Código e quantidade de Instruções conforme o tipo

```
# Programa 15 - By Hernane V | Data: 04/12/2022
# Arquitetura de Computadores II - Romanelli
.text
.globl main

# $s0 → V[0]
# $s1 → i
# $s2 → valor fixo de 4

main:
    ori $s0, $zero, 0x1001    # $s0 = 0x1001
    sll $s0, $s0, 16          # $s0 = $s0 << 16

    add $s1, $zero, $zero     # $s1 = 0
    ori $s2, $zero, 4         # $s2 = 4

while:
    sll $t1, $s1, 1           # $t1 = s1 * 2
    addi $t1, $t1, 1          # $t1 = t1 + 1

    mult $s2, $s1             # multiplica 4 vezes a posição no vetor
    mflo $t2                  # $t2 = 4 * $s1 + $s0, o endereço do elemento na memória
    add $t2, $t2, $s0         # armazena $t1 na memória
    sw $t1, 0($t2)

    addi $s1, $s1, 1          # incrementa o contador mais 1
    slti $t3, $s1, 100        # compara a posição do vetor com a última posição
    beq $t3, $zero, fim       # se a posição de s1 chegar a 100, encerra
    j while

fim:

.data
```



## Programa 15 – Execução via MARS

The screenshot displays the MARS MIPS simulator interface. The main window is divided into several panes:

- Code Segment:** Shows the assembly code with addresses, hex codes, and comments. The code is the same as in the previous block.
- Registers:** A table showing the state of MIPS registers. The \$s0 register is highlighted in green, indicating it is the current register being used. The \$s0 register contains the value 0x1001.
- Data Segment:** A table showing the state of memory. The memory is initialized with zeros. The \$s0 register is highlighted in green, indicating it is the current register being used.
- Registers Table:** A table showing the state of MIPS registers. The \$s0 register is highlighted in green, indicating it is the current register being used. The \$s0 register contains the value 0x1001.
- Registers Table:** A table showing the state of MIPS registers. The \$s0 register is highlighted in green, indicating it is the current register being used. The \$s0 register contains the value 0x1001.

The bottom pane shows the MARS Messages window, which displays the status of the program execution. The message "Reset: reset completed." is visible.

## Programa 16 – Código

```
# Programa 16 - By Hernane V | Data: 04/12/2022
# Arquitetura de Computadores II - Romanelli

.text
.globl main

main:
    lui $t0, 0x1001
    addi $t0, $t0, 0x0000
    lw $s0, 0($t0)      # $s0 = x
    lw $s1, 4($t0)      # $s1 = y
    lw $s2, 8($t0)      # $s2 = z

    div $s0, $s2        # x/z
    mflo $t1            # $t1 = x/z
    mult $t1, $s1        # $t1*y
    mflo $t1            # $t1=t1*y

.data
x: .word 0x186A00      # 1600000
y: .word 0x13880       # 80000
z: .word 0x61A80       # 400000
```

## Programa 16 – Execução via MARS

The screenshot displays the MARS MIPS simulator interface. The main window shows the assembly code for Program 16, with the following instructions highlighted in yellow:

```
lui $t0, 0x1001
addi $t0, $t0, 0x0000
lw $s0, 0($t0)      # $s0 = x
lw $s1, 4($t0)      # $s1 = y
lw $s2, 8($t0)      # $s2 = z

div $s0, $s2        # x/z
mflo $t1            # $t1 = x/z
mult $t1, $s1        # $t1*y
mflo $t1            # $t1=t1*y
```

The registers window on the right shows the values of the registers after execution. The registers \$s0, \$s1, and \$s2 contain the values 1600000, 80000, and 400000, respectively. The register \$t1 contains the value 320000, which is the result of the calculation  $(x/z) * y$ .

The data segment window at the bottom shows the memory layout of the program, with the variables x, y, and z stored at addresses 0x186A00, 0x13880, and 0x61A80, respectively.

The console window at the bottom shows the message "Reset: reset completed." and "program is finished running (dropped off bottom)".

## Programa 17 – Código

```
# Programa 17 - By Hernane V | Data: 04/12/2022
# Arquitetura de Computadores II - Romanelli
.text
.globl main

main:
    lui $t0, 0x1001
    lw $s0, 0($t0)          # $s0 = x
    lw $s1, 4($t0)          # $s1 = y
    lw $s2, 8($t0)          # $s2 = z
    addi $t1, $zero, 0      # counter

loop:
    add $s2, $s2, $s0        # $s2 = $s2 + $s0
    addi $t1, $t1, 1         # $t1 = $t1 + 1
    bne $t1, $s1, loop

    sw $s2, 8($t0)          # guarda na memória

.data
x: .word 3
y: .word 4
k: .word 0
```

## Programa 17 – Execução via MARS

The screenshot displays the MARS application window. The main assembly editor shows the following code:

```
1: lui $t0, 0x1001
2: lw $s0, 0($t0)
3: lw $s1, 4($t0)
4: lw $s2, 8($t0)
5: addi $t1, $zero, 0
6:
7: loop:
8: add $s2, $s2, $s0
9: addi $t1, $t1, 1
10: bne $t1, $s1, loop
11:
12: sw $s2, 8($t0)
13:
14: .data
15: x: .word 3
16: y: .word 4
17: k: .word 0
```

The 'Registers' panel on the right shows the following values:

Register	Value
\$s0	3
\$s1	4
\$s2	12
\$s3	0
\$s4	0
\$s5	0
\$s6	0
\$s7	0
\$s8	0
\$s9	0
\$s10	0
\$s11	0

The 'Data Segment' panel at the bottom shows the following memory layout:

Address	Value
0x10010000	3
0x10010004	4
0x10010008	12
0x1001000C	0
0x10010010	0
0x10010014	0
0x10010018	0
0x1001001C	0
0x10010020	0
0x10010024	0
0x10010028	0
0x1001002C	0
0x10010030	0
0x10010034	0
0x10010038	0
0x1001003C	0
0x10010040	0
0x10010044	0
0x10010048	0
0x1001004C	0
0x10010050	0
0x10010054	0
0x10010058	0
0x1001005C	0
0x10010060	0
0x10010064	0
0x10010068	0
0x1001006C	0
0x10010070	0
0x10010074	0
0x10010078	0
0x1001007C	0
0x10010080	0
0x10010084	0
0x10010088	0
0x1001008C	0
0x10010090	0
0x10010094	0
0x10010098	0
0x1001009C	0
0x100100A0	0
0x100100A4	0
0x100100A8	0
0x100100AC	0
0x100100B0	0
0x100100B4	0
0x100100B8	0
0x100100BC	0
0x100100C0	0
0x100100C4	0
0x100100C8	0
0x100100CC	0
0x100100D0	0
0x100100D4	0
0x100100D8	0
0x100100DC	0
0x100100E0	0
0x100100E4	0
0x100100E8	0
0x100100EC	0
0x100100F0	0
0x100100F4	0
0x100100F8	0
0x100100FC	0
0x10010100	0
0x10010104	0
0x10010108	0
0x1001010C	0
0x10010110	0
0x10010114	0
0x10010118	0
0x1001011C	0
0x10010120	0
0x10010124	0
0x10010128	0
0x1001012C	0
0x10010130	0
0x10010134	0
0x10010138	0
0x1001013C	0
0x10010140	0
0x10010144	0
0x10010148	0
0x1001014C	0
0x10010150	0
0x10010154	0
0x10010158	0
0x1001015C	0
0x10010160	0
0x10010164	0
0x10010168	0
0x1001016C	0
0x10010170	0
0x10010174	0
0x10010178	0
0x1001017C	0
0x10010180	0
0x10010184	0
0x10010188	0
0x1001018C	0
0x10010190	0
0x10010194	0
0x10010198	0
0x1001019C	0
0x100101A0	0
0x100101A4	0
0x100101A8	0
0x100101AC	0
0x100101B0	0
0x100101B4	0
0x100101B8	0
0x100101BC	0
0x100101C0	0
0x100101C4	0
0x100101C8	0
0x100101CC	0
0x100101D0	0
0x100101D4	0
0x100101D8	0
0x100101DC	0
0x100101E0	0
0x100101E4	0
0x100101E8	0
0x100101EC	0
0x100101F0	0
0x100101F4	0
0x100101F8	0
0x100101FC	0
0x10010200	0
0x10010204	0
0x10010208	0
0x1001020C	0
0x10010210	0
0x10010214	0
0x10010218	0
0x1001021C	0
0x10010220	0
0x10010224	0
0x10010228	0
0x1001022C	0
0x10010230	0
0x10010234	0
0x10010238	0
0x1001023C	0
0x10010240	0
0x10010244	0
0x10010248	0
0x1001024C	0
0x10010250	0
0x10010254	0
0x10010258	0
0x1001025C	0
0x10010260	0
0x10010264	0
0x10010268	0
0x1001026C	0
0x10010270	0
0x10010274	0
0x10010278	0
0x1001027C	0
0x10010280	0
0x10010284	0
0x10010288	0
0x1001028C	0
0x10010290	0
0x10010294	0
0x10010298	0
0x1001029C	0
0x100102A0	0
0x100102A4	0
0x100102A8	0
0x100102AC	0
0x100102B0	0
0x100102B4	0
0x100102B8	0
0x100102BC	0
0x100102C0	0
0x100102C4	0
0x100102C8	0
0x100102CC	0
0x100102D0	0
0x100102D4	0
0x100102D8	0
0x100102DC	0
0x100102E0	0
0x100102E4	0
0x100102E8	0
0x100102EC	0
0x100102F0	0
0x100102F4	0
0x100102F8	0
0x100102FC	0
0x10010300	0
0x10010304	0
0x10010308	0
0x1001030C	0
0x10010310	0
0x10010314	0
0x10010318	0
0x1001031C	0
0x10010320	0
0x10010324	0
0x10010328	0
0x1001032C	0
0x10010330	0
0x10010334	0
0x10010338	0
0x1001033C	0
0x10010340	0
0x10010344	0
0x10010348	0
0x1001034C	0
0x10010350	0
0x10010354	0
0x10010358	0
0x1001035C	0
0x10010360	0
0x10010364	0
0x10010368	0
0x1001036C	0
0x10010370	0
0x10010374	0
0x10010378	0
0x1001037C	0
0x10010380	0
0x10010384	0
0x10010388	0
0x1001038C	0
0x10010390	0
0x10010394	0
0x10010398	0
0x1001039C	0
0x100103A0	0
0x100103A4	0
0x100103A8	0
0x100103AC	0
0x100103B0	0
0x100103B4	0
0x100103B8	0
0x100103BC	0
0x100103C0	0
0x100103C4	0
0x100103C8	0
0x100103CC	0
0x100103D0	0
0x100103D4	0
0x100103D8	0
0x100103DC	0
0x100103E0	0
0x100103E4	0
0x100103E8	0
0x100103EC	0
0x100103F0	0
0x100103F4	0
0x100103F8	0
0x100103FC	0
0x10010400	0
0x10010404	0
0x10010408	0
0x1001040C	0
0x10010410	0
0x10010414	0
0x10010418	0
0x1001041C	0
0x10010420	0
0x10010424	0
0x10010428	0
0x1001042C	0
0x10010430	0
0x10010434	0
0x10010438	0
0x1001043C	0
0x10010440	0
0x10010444	0
0x10010448	0
0x1001044C	0
0x10010450	0
0x10010454	0
0x10010458	0
0x1001045C	0
0x10010460	0
0x10010464	0
0x10010468	0
0x1001046C	0
0x10010470	0
0x10010474	0
0x10010478	0
0x1001047C	0
0x10010480	0
0x10010484	0
0x10010488	0
0x1001048C	0
0x10010490	0
0x10010494	0
0x10010498	0
0x1001049C	0
0x100104A0	0
0x100104A4	0
0x100104A8	0
0x100104AC	0
0x100104B0	0
0x100104B4	0
0x100104B8	0
0x100104BC	0
0x100104C0	0
0x100104C4	0
0x100104C8	0
0x100104CC	0
0x100104D0	0
0x100104D4	0
0x100104D8	0
0x100104DC	0
0x100104E0	0
0x100104E4	0
0x100104E8	0
0x100104EC	0
0x100104F0	0
0x100104F4	0
0x100104F8	0
0x100104FC	0
0x10010500	0
0x10010504	0
0x10010508	0
0x1001050C	0
0x10010510	0
0x10010514	0
0x10010518	0
0x1001051C	0
0x10010520	0
0x10010524	0
0x10010528	0
0x1001052C	0
0x10010530	0
0x10010534	0
0x10010538	0
0x1001053C	0
0x10010540	0
0x10010544	0
0x10010548	0
0x1001054C	0
0x10010550	0
0x10010554	0
0x10010558	0
0x1001055C	0
0x10010560	0
0x10010564	0
0x10010568	0
0x1001056C	0
0x10010570	0
0x10010574	0
0x10010578	0
0x1001057C	0
0x10010580	0
0x10010584	0
0x10010588	0
0x1001058C	0
0x10010590	0
0x10010594	0
0x10010598	0
0x1001059C	0
0x100105A0	0
0x100105A4	0
0x100105A8	0
0x100105AC	0
0x100105B0	0
0x100105B4	0
0x100105B8	0
0x100105BC	0
0x100105C0	0
0x100105C4	0
0x100105C8	0
0x100105CC	0
0x100105D0	0
0x100105D4	0
0x100105D8	0
0x100105DC	0
0x100105E0	0
0x100105E4	0
0x100105E8	0
0x100105EC	0
0x100105F0	0
0x100105F4	0
0x100105F8	0
0x100105FC	0
0x10010600	0
0x10010604	0
0x10010608	0
0x1001060C	0
0x10010610	0
0x10010614	0
0x10010618	0
0x1001061C	0
0x10010620	0
0x10010624	0
0x10010628	0
0x1001062C	0
0x10010630	0
0x10010634	0
0x10010638	0
0x1001063C	0
0x10010640	0
0x10010644	0
0x10010648	0
0x1001064C	0
0x10010650	0
0x10010654	0
0x10010658	0
0x1001065C	0
0x10010660	0
0x10010664	0
0x10010668	0
0x1001066C	0
0x10010670	0
0x10010674	0
0x10010678	0
0x1001067C	0
0x10010680	0
0x10010684	0
0x10010688	0
0x1001068C	0
0x10010690	0
0x10010694	0
0x10010698	0
0x1001069C	0
0x100106A0	0
0x100106A4	0
0x100106A8	0
0x100106AC	0
0x100106B0	0
0x100106B4	0
0x100106B8	0
0x100106BC	0
0x100106C0	0
0x100106C4	0
0x100106C8	0
0x100106CC	0
0x100106D0	0
0x100106D4	0
0x100106D8	0
0x100106DC	0
0x100106E0	0
0x100106E4	0
0x100106E8	0
0x100106EC	0
0x100106F0	0
0x100106F4	0
0x100106F8	0
0x100106FC	0
0x10010700	0
0x10010704	0
0x10010708	0
0x1001070C	0
0x10010710	0
0x10010714	0
0x10010718	0

## Programa 18 – Código e quantidade de Instruções conforme o tipo

```
# Programa 18 - By Hernane V | Data: 04/12/2022
# Arquitetura de Computadores II - Romanelli
.text
.globl main

main:
    lui $t0, 0x1001
    lw $s0, 0($t0)          # $s0 = x
    lw $s1, 4($t0)          # $s1 = y
    lw $s2, 8($t0)          # $s2 = z
    addi $t1, $zero, 0      # counter

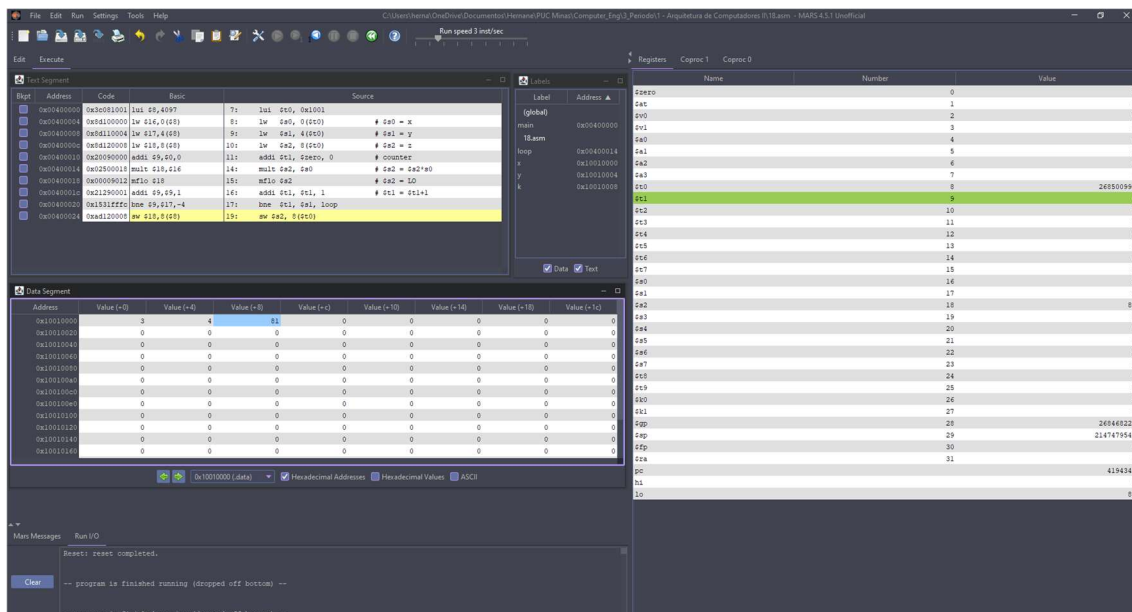
loop:
    mult $s2, $s0           # $s2 = $s2*$s0
    mflo $s2                # $s2 = LO
    addi $t1, $t1, 1        # $t1 = $t1+1
    bne $t1, $s1, loop

    sw $s2, 8($t0)

.data
x: .word 3
y: .word 4
k: .word 1
```



## Programa 18 – Execução via MARS



## REPOSITÓRIO DE PMG-AC2

Disponível em: <https://github.com/hernanevelozo/PMG-AC2>

## REFERÊNCIAS:

**HENNESSY, John L.** *Arquitetura de computadores uma abordagem quantitativa*. Rio de Janeiro **GEN LTC 2019 1** recurso online ISBN 9788595150669.

**SPRINGERLINK (ONLINE SERVICE).** *The MIPS-X RISC Microprocessor*. 1st ed. 1989. **XXIV**, 232 p ISBN9781475767629.