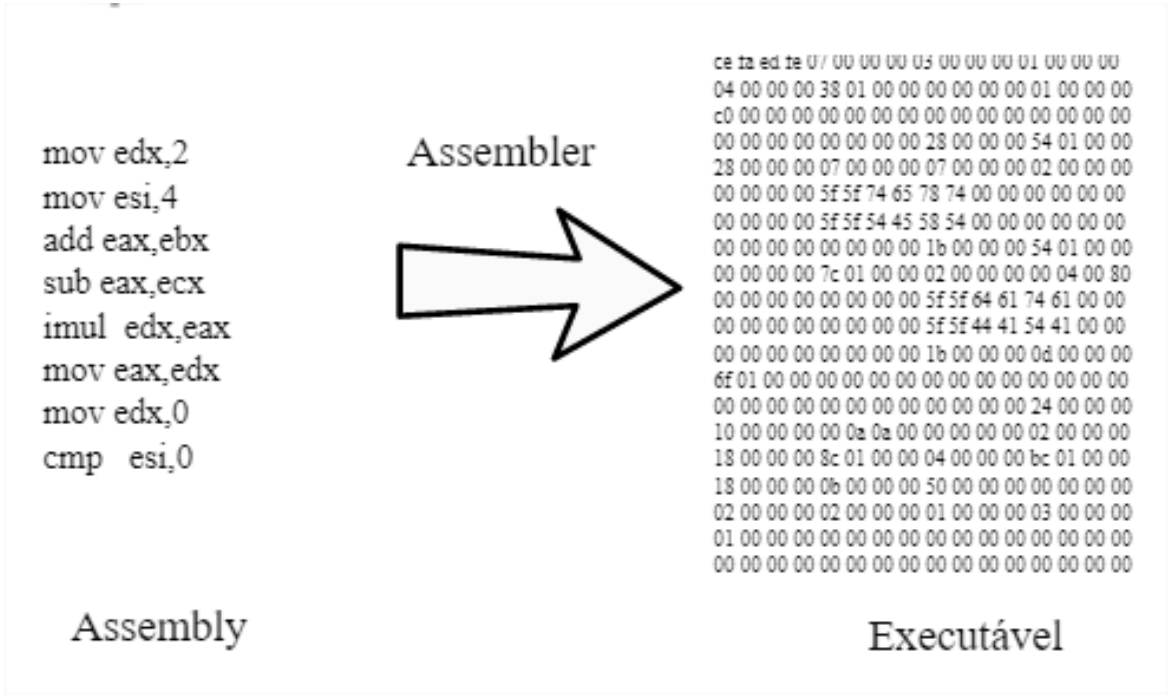
Programação em Baixo Nível (Registradores)

Porfª Mª Cleane Nascimento

RELEMBRANDO À AULA ANTERIOR...



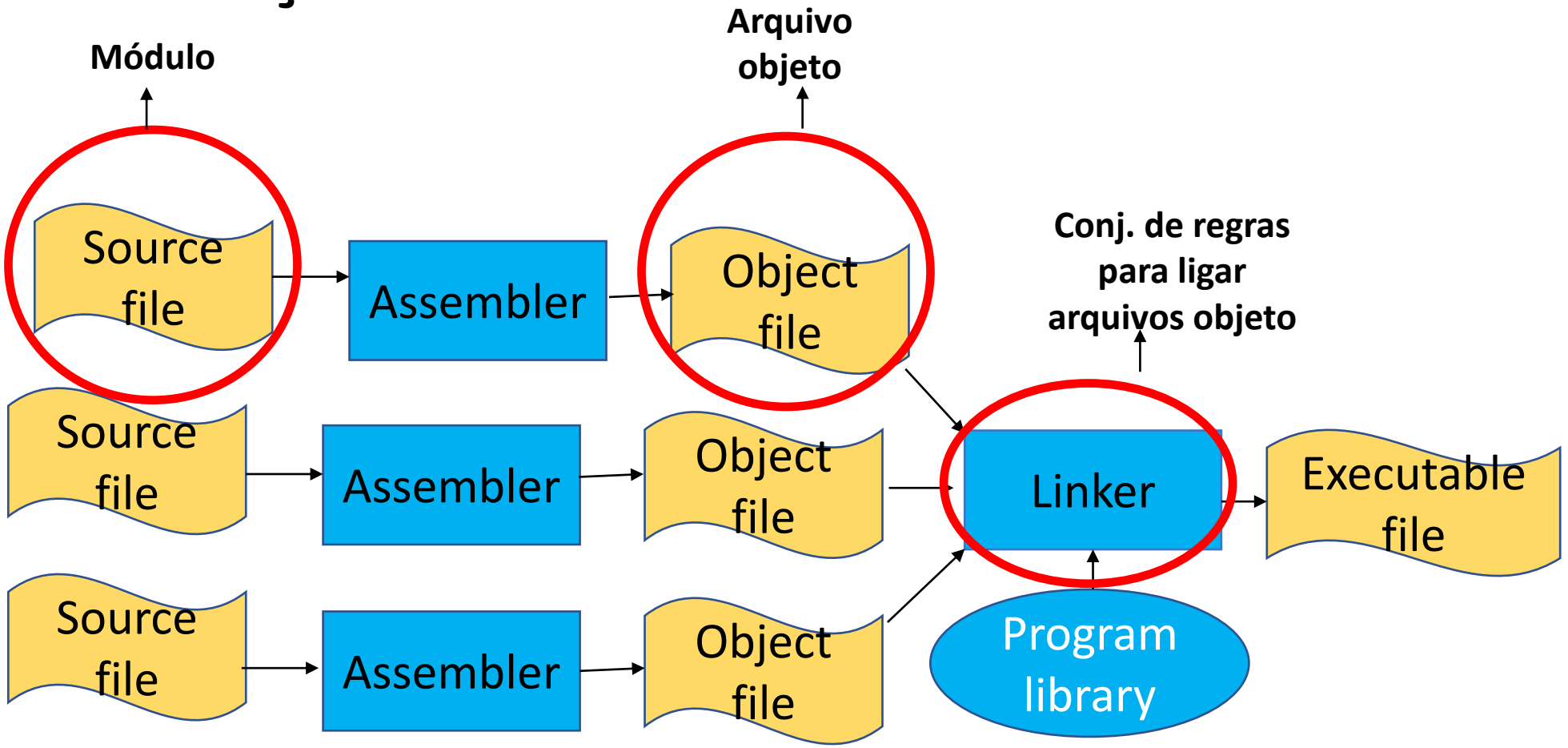
RELEMBRANDO À AULA ANTERIOR...

O que são Registradores?

- Dispositivos que armazenam dados e informações utilizadas para execução de um programa. Existem dois tipos de Registradores:
- Registradores Gerais
- Registradores de Propósitos Especiais.

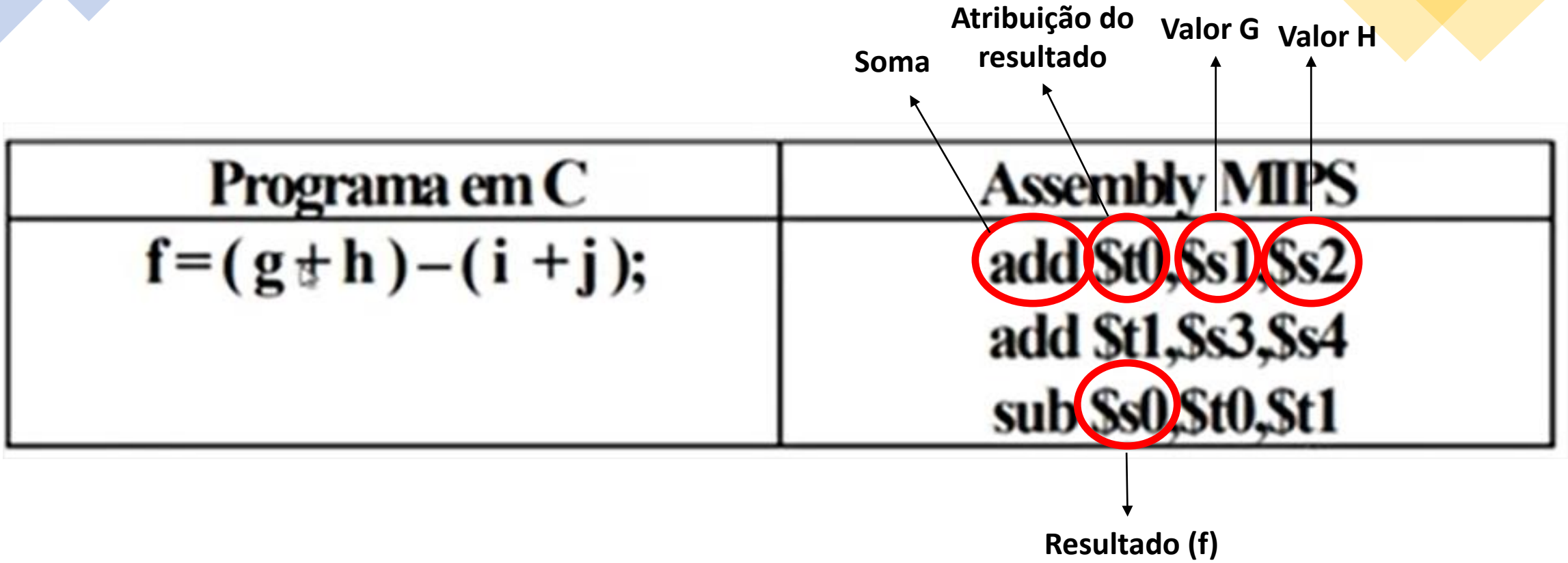
(MANZANO, 2004)

Combinação de módulos menores



CONJUNTO DE INSTRUÇÕES MIPS

- Instrução **é uma palavra da linguagem**;
- ISA MIPS → Instruções possuem até 3 operandos;
- E o que é ISA?
- Intruction Set Architecture (**Conjunto de Instruções da Arquitetura**);
- No MIPS os operandos das instruções são chamados registradores ($);
- Há 32 registradores de 32 bits;
- Cada registrador possui o símbolo $ antecedendo seu nome.



Como devo fazer para realizar a soma de (i+j)?

Load e Store

- **Load:** instrução de movimentação de dados da memória para o registrador;

Operação de **leitura da memória**.

- **Store:** instrução de movimentação de dados do registrador para a memória;

Operação **escrita na memória**.

- **Move:** instrução para passar o conteúdo de um registrador para outro;

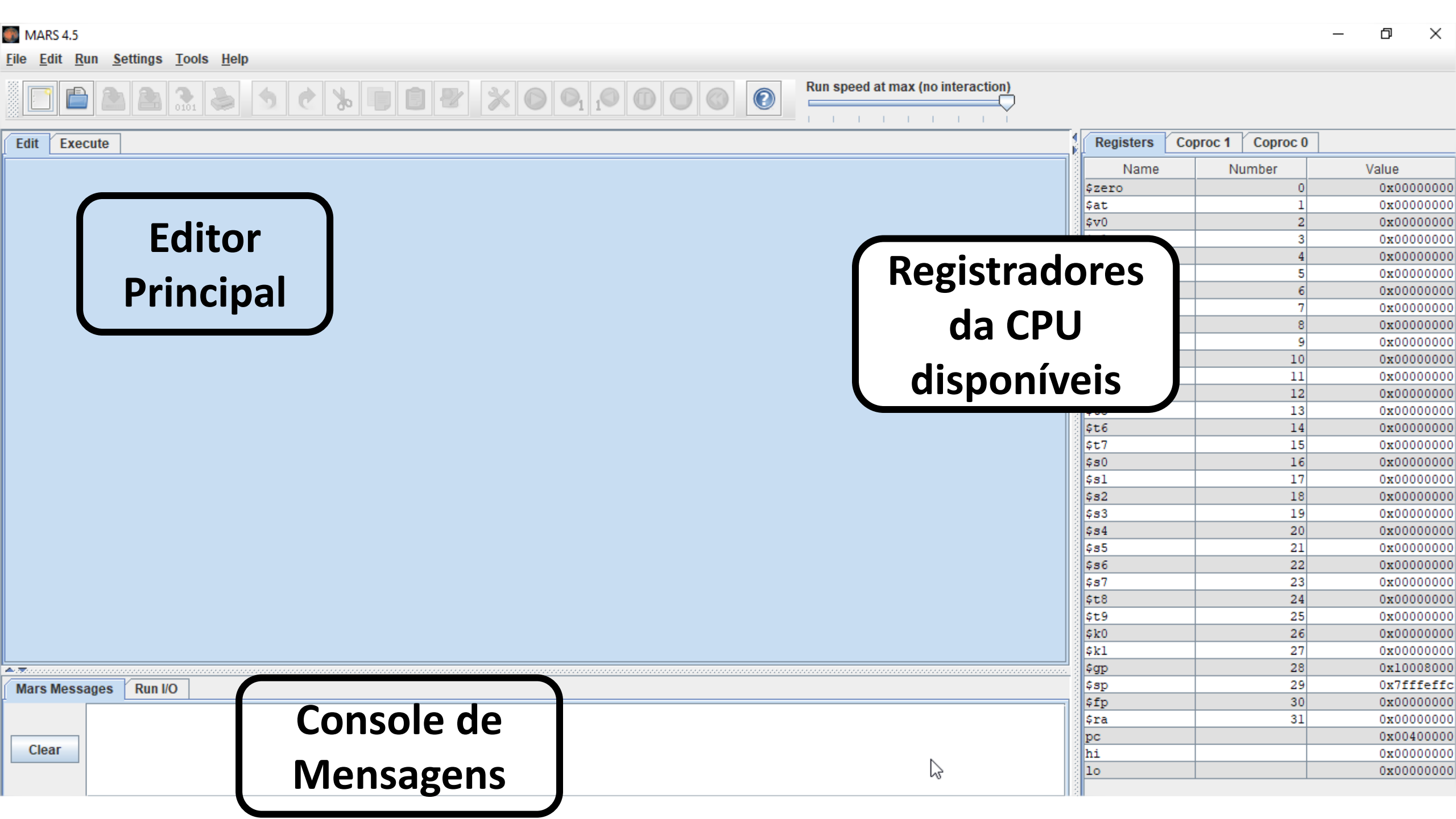
Memória RAM não é envolvida.

E qual programa utilizaremos?

- MARS – Programar e Depurar (debug);

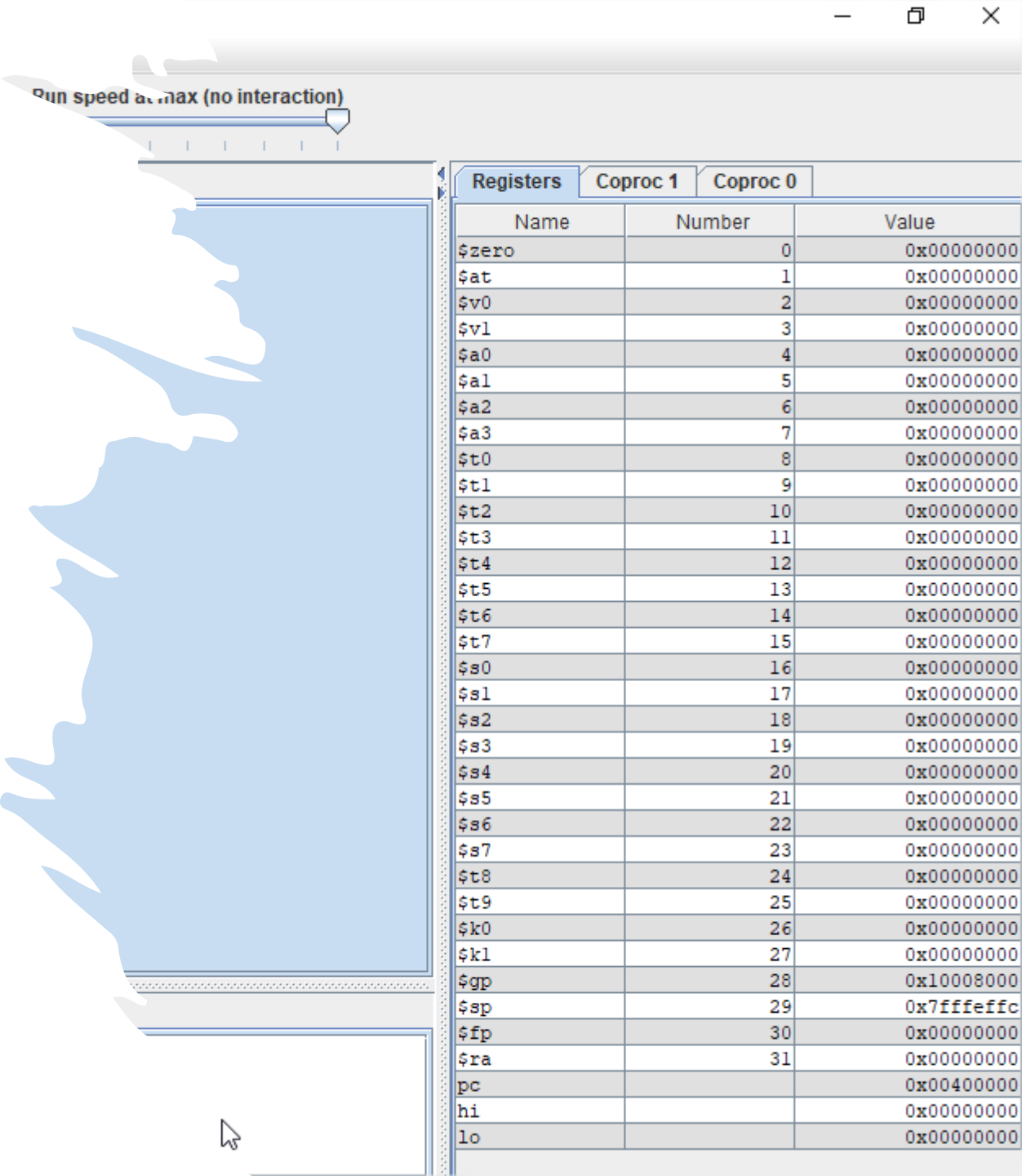
- É um arquivo .jar, portanto é necessário ter o JAVA instalado.

(O link com os arquivos de instalação se encontram no SIGAA).


MARS 4.5
File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Editor Principal
Registradores da CPU disponíveis
Registers Coproc 1 Coproc 0
Name Number Value
$zero 0 0x00000000
$at 1 0x00000000
$v0 2 0x00000000
3 0x00000000
4 0x00000000
5 0x00000000
6 0x00000000
7 0x00000000
8 0x00000000
9 0x00000000
10 0x00000000
11 0x00000000
12 0x00000000
13 0x00000000
$t6 14 0x00000000
$t7 15 0x00000000
$s0 16 0x00000000
$s1 17 0x00000000
$s2 18 0x00000000
$s3 19 0x00000000
$s4 20 0x00000000
$s5 21 0x00000000
$s6 22 0x00000000
$s7 23 0x00000000
$t8 24 0x00000000
$t9 25 0x00000000
$k0 26 0x00000000
$k1 27 0x00000000
$gp 28 0x10008000
$sp 29 0x7fffeffc
$fp 30 0x00000000
$ra 31 0x00000000
pc 0x00400000
hi 0x00000000
lo 0x00000000
Mars Messages Run I/O
Clear
Console de Mensagens

A princípio não usaremos:

- $at – assember temporary;
- $k0 e $kI – registradores do kernel;
- $gp – registradores de valores globais;
- $sp – stack pointer (aponta para o início do stack e muda progressivamente);
- $fp – frame pointer (aponta para o início da pilha e não muda até que a função seja executada).



Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00400000
hi		0x00000000
lo		0x00000000

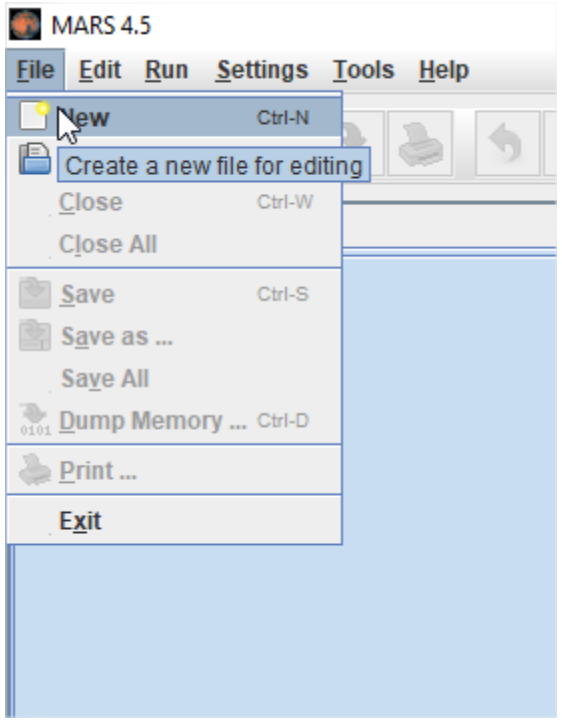
Comando li $v0

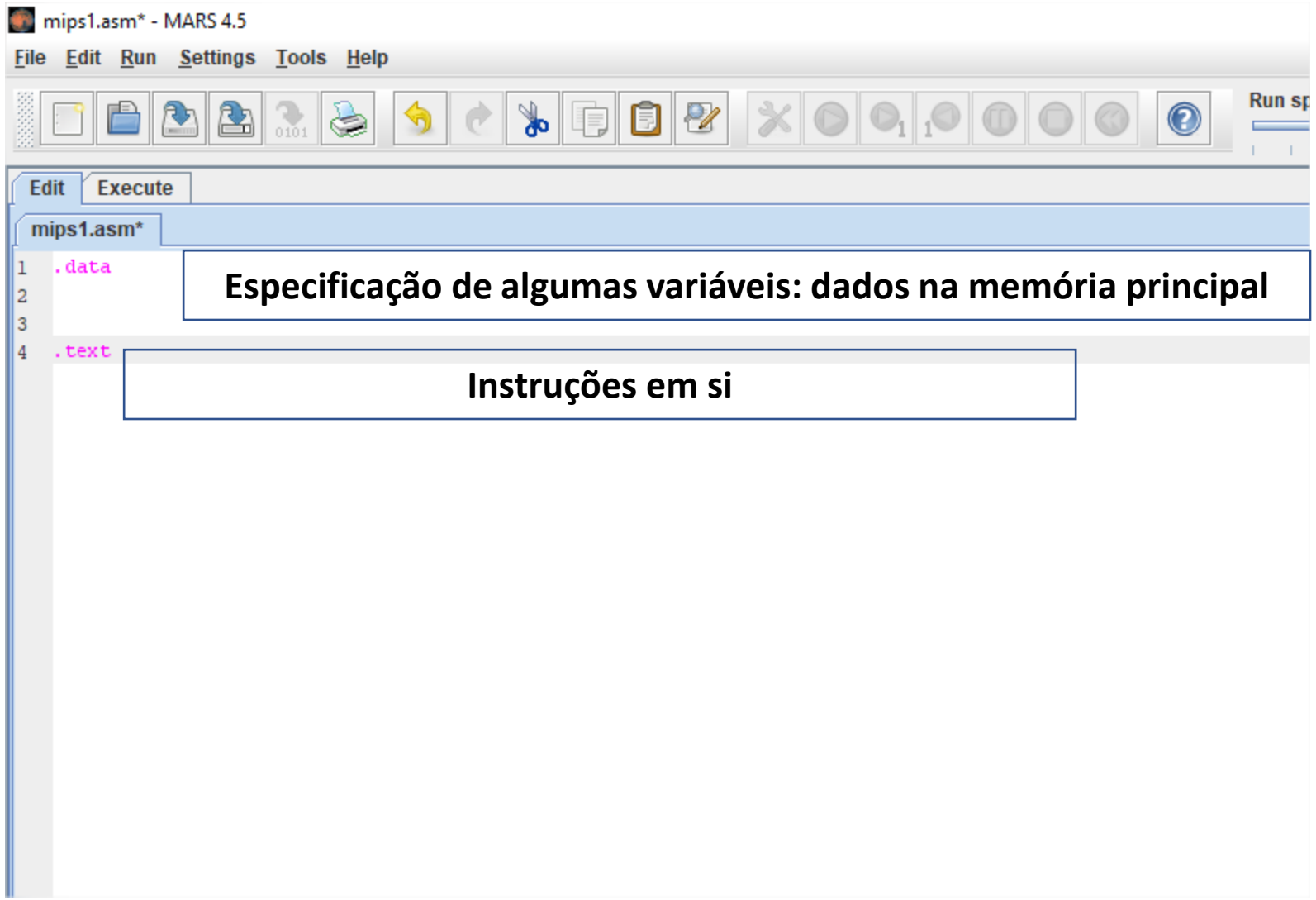
Comando	Significado
Li $v0, 1	Imprimir inteiro
Li $v0, 2	Imprimir float
Li $v0, 3	Imprimir double
Li $v0, 4	Imprimir String ou char
Li $v0, 5	Ler inteiro
Li $v0, 6	Ler float
Li $v0, 7	Ler double
Li $v0, 8	Ler String ou char
Li $v0, 10	Encerrar programa principal

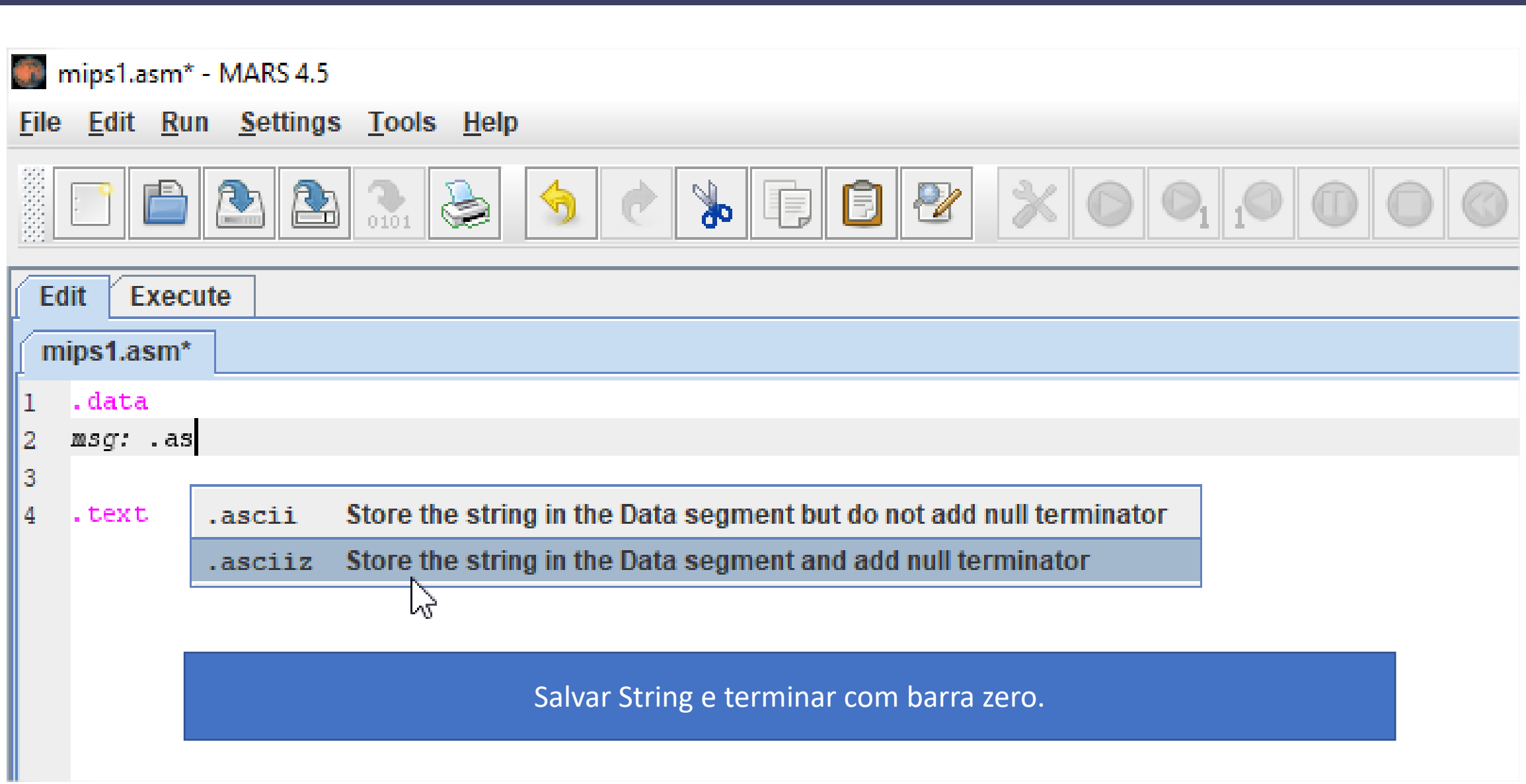
Atribuir valor inteiro à um registrador é usada a instrução li

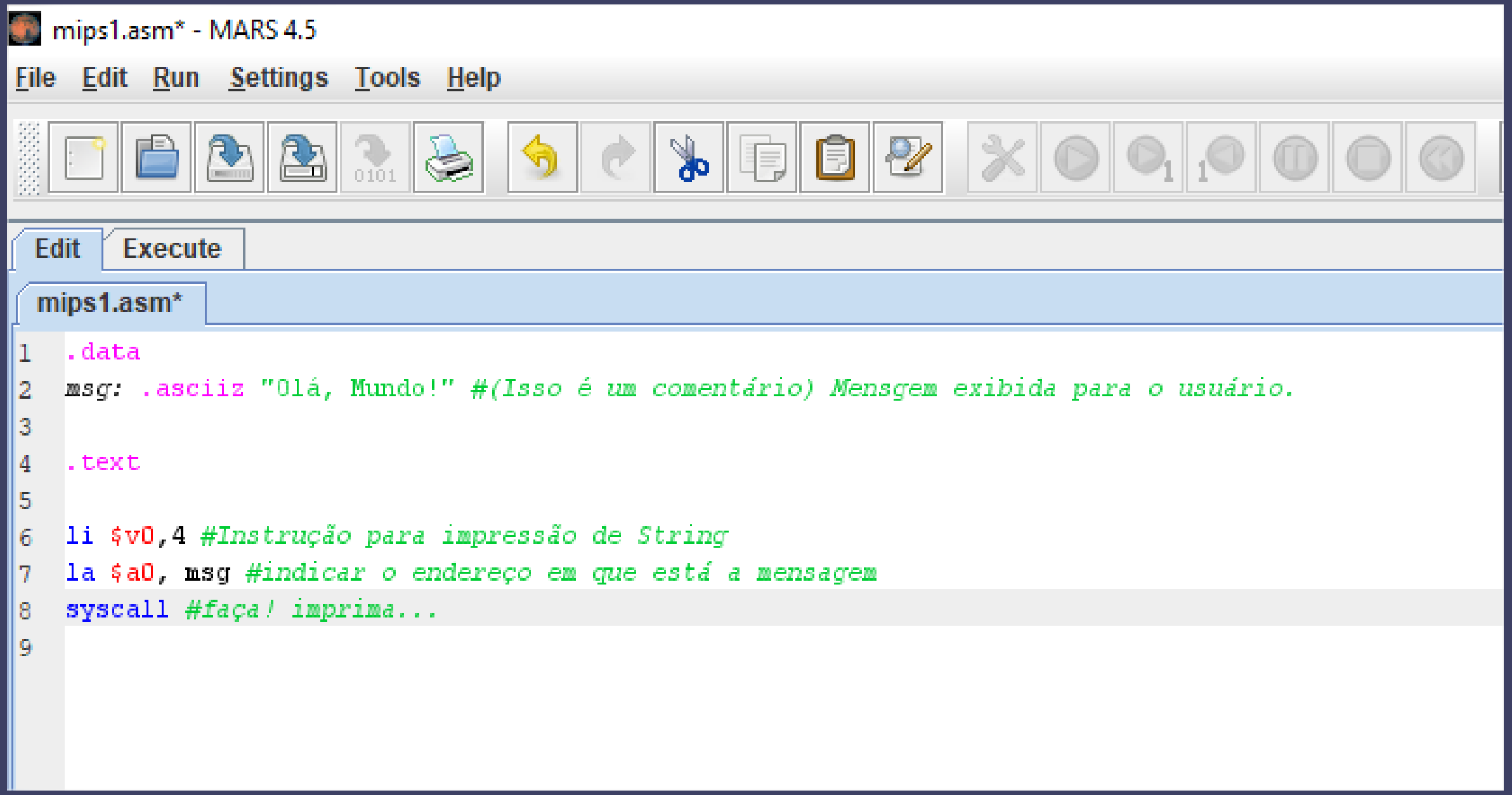
Códigos de 1 a 4 – **impressão;**
Códigos de 5 a 8 – **Leitura.**

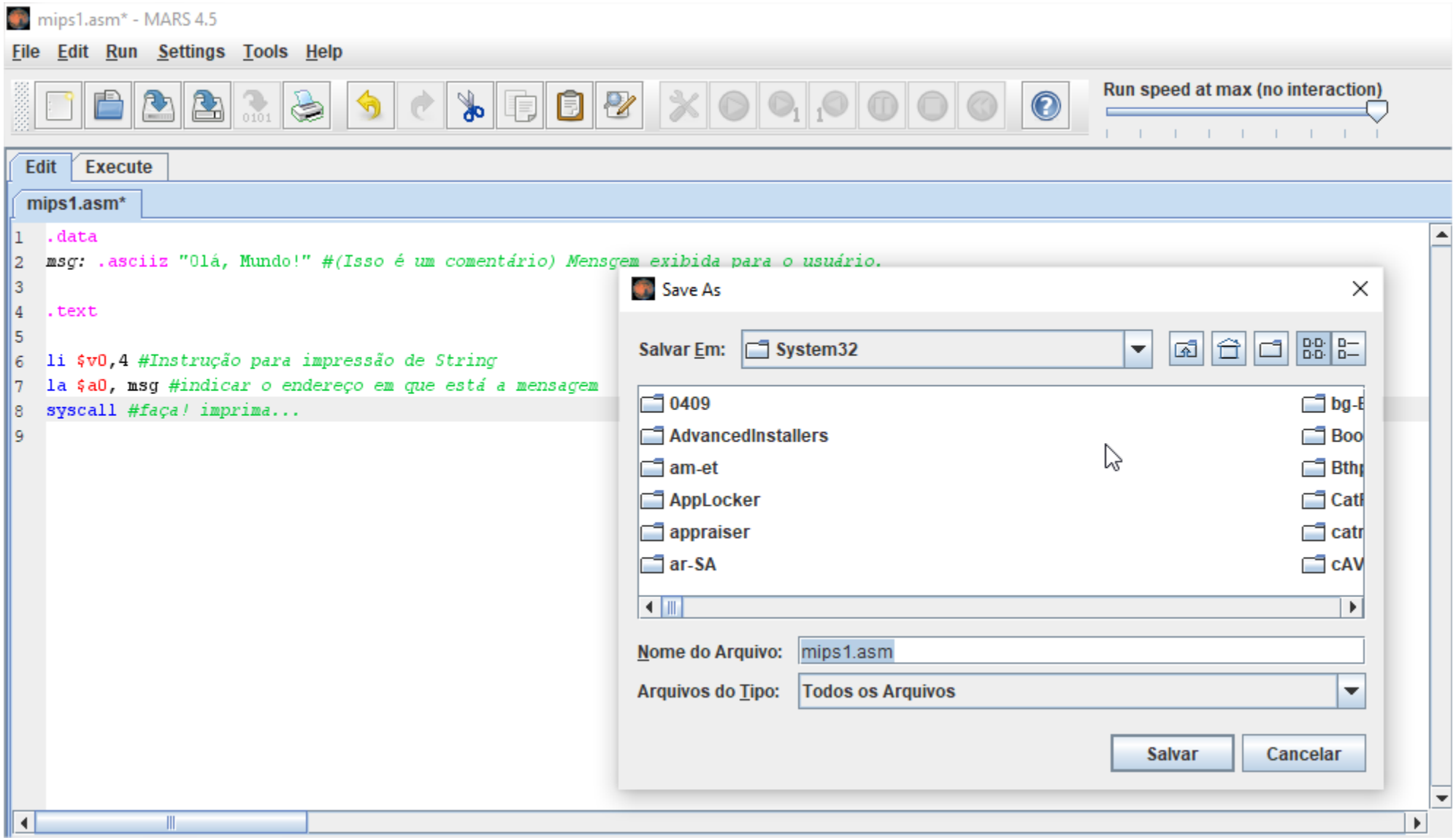
Nosso primeiro programa...

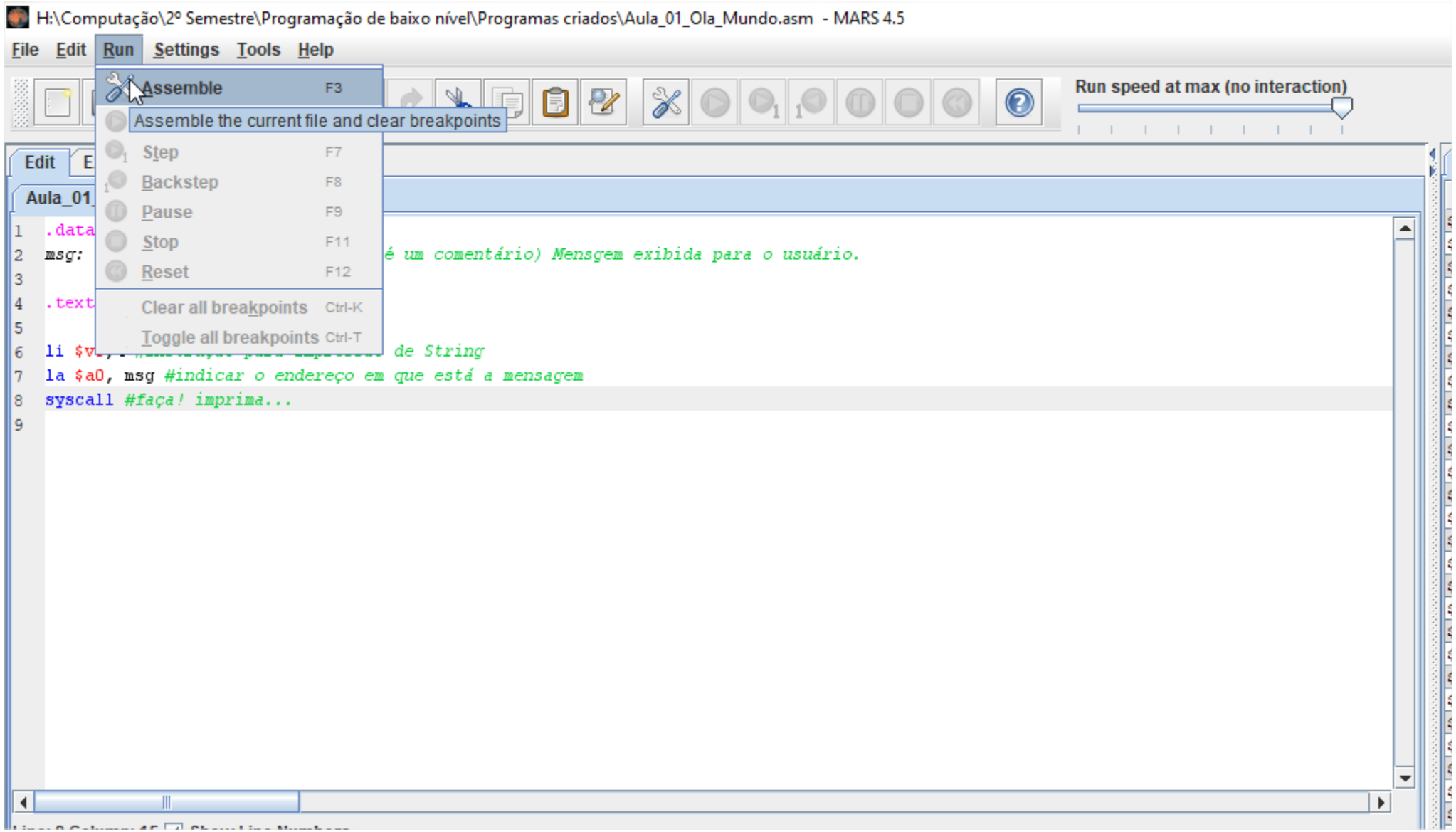





mips1.asm* - MARS 4.5
File Edit Run Settings Tools Help
Edit Execute
mips1.asm*
1 .data
2 msg: .as
3
4 .text
.ascii Store the string in the Data segment but do not add null terminator
.asciiz Store the string in the Data segment and add null terminator
Salvar String e terminar com barra zero.


mips1.asm* - MARS 4.5
File Edit Run Settings Tools Help
Edit Execute
mips1.asm*
1 .data
2 msg: .asciiz "Olá, Mundo!" #(Isso é um comentário) Mensgem exibida para o usuário.
3
4 .text
5
6 li $v0,4 #Instrução para impressão de String
7 la $a0, msg #indicar o endereço em que está a mensagem
8 syscall #faça! imprima...
9


mips1.asm* - MARS 4.5
File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
mips1.asm*
1 .data
2 msg: .asciiz "Olá, Mundo!" #(Isso é um comentário) Mensgem exibida para o usuário.
3
4 .text
5
6 li $v0,4 #Instrução para impressão de String
7 la $a0, msg #indicar o endereço em que está a mensagem
8 syscall #faça! imprima...
9
Save As
Salvar Em: System32
0409
AdvancedInstallers
am-et
AppLocker
appraiser
ar-SA
Nome do Arquivo: mips1.asm
Arquivos do Tipo: Todos os Arquivos
Salvar
Cancelar


H:\Computação\2º Semestre\Programação de baixo nível\Programas criados\Aula_01_Ola_Mundo.asm - MARS 4.5
File Edit Run Settings Tools Help
Assemble F3
Assemble the current file and clear breakpoints
Step F7
Backstep F8
Pause F9
Stop F11
Reset F12
Clear all breakpoints Ctrl-K
Toggle all breakpoints Ctrl-T
Run speed at max (no interaction)
Edit
Aula_01
1 .data
2 msg:
3
4 .text
5
6 li $v
7 la $a0, msg #indicar o endereço em que está a mensagem
8 syscall #faça! imprima...
9
é um comentário) Mensgem exibida para o usuário.
de String

Imprimir 1 caractere na tela

```
1   .data
2           caractere: .byte 'C'#caractere a ser impresso
3   .text
4   li $v0, 4 #imprimir char ou string
5   la $a0, caractere
6   syscall
7
8   li $v0, 10 #Encerrar o programa
9   syscall
10
```

Imprimir Inteiros

```
1  .data
2          idade:  .word 30
3  .text
4          li $v0, 1
5          lw $a0, idade
6          syscall
7
```

Adição de inteiros

Para somar

t0 = t1 + t2

t0 = t1 + 10

Usamos:

Add $t0, $t1, $t2

Addi $t0, $t1, 10

Se utilizarmos 2 registradores = add
Se utilizarmos 2 registradores e um inteiro = addi

A instrução fica assim:

```
1   .text
2   li $t0, 10
3   li $t1, 15
4   add $s0, $t0, $t1
5
```

Caso a gente resolva imprimir:

```
1  .data
2
3  msg1: .asciiz "Digite o primeiro número" #Variáveis Globais
4  msg2: .asciiz "Digite o segundo número"
5  resultado: .asciiz "Resultado"
6
7  .text
8
9  #Leitura do primeiro número
10         li $v0, 4 #Código para print
11         la $a0, msg1 #Argumento do print
12         syscall #Solicita que o Kernel Execute
13
```

- **Continua...**

Subtraindo2numeros.asm | somando e imprimindo_Correto.asm

```
14          li $v0, 5 #Realizar leitura
15          syscall
16
17          move $s0, $v0 #Mover o valor de V0 para outro registrador (S0
18
19          #Leitura do segundo número
20          li $v0, 4
21          la $a0, msg2
22          syscall
23
24          li $v0, 5
25          syscall
26
```

```
            move $s1, $v0

            #Operação de adição

            add $s2, $s0, $s1 #$S2 guarda o resultado da soma que s
            li $v0, 4 #Passando o código 4 para $v0
            la $a0, resultado #Passando o resultado para a0

            syscall

            li $v0, 1 #Chamada de sistema para printar um inteiro
            move $a0, $s2  #Move o valor da soma para o registrador
            syscall
```

Subtração de Inteiros

```
.text
        li $t0, 25
        li $t1, 15
        sub $t2, $t0, $t1
        subi $t3, $t2, 40
```

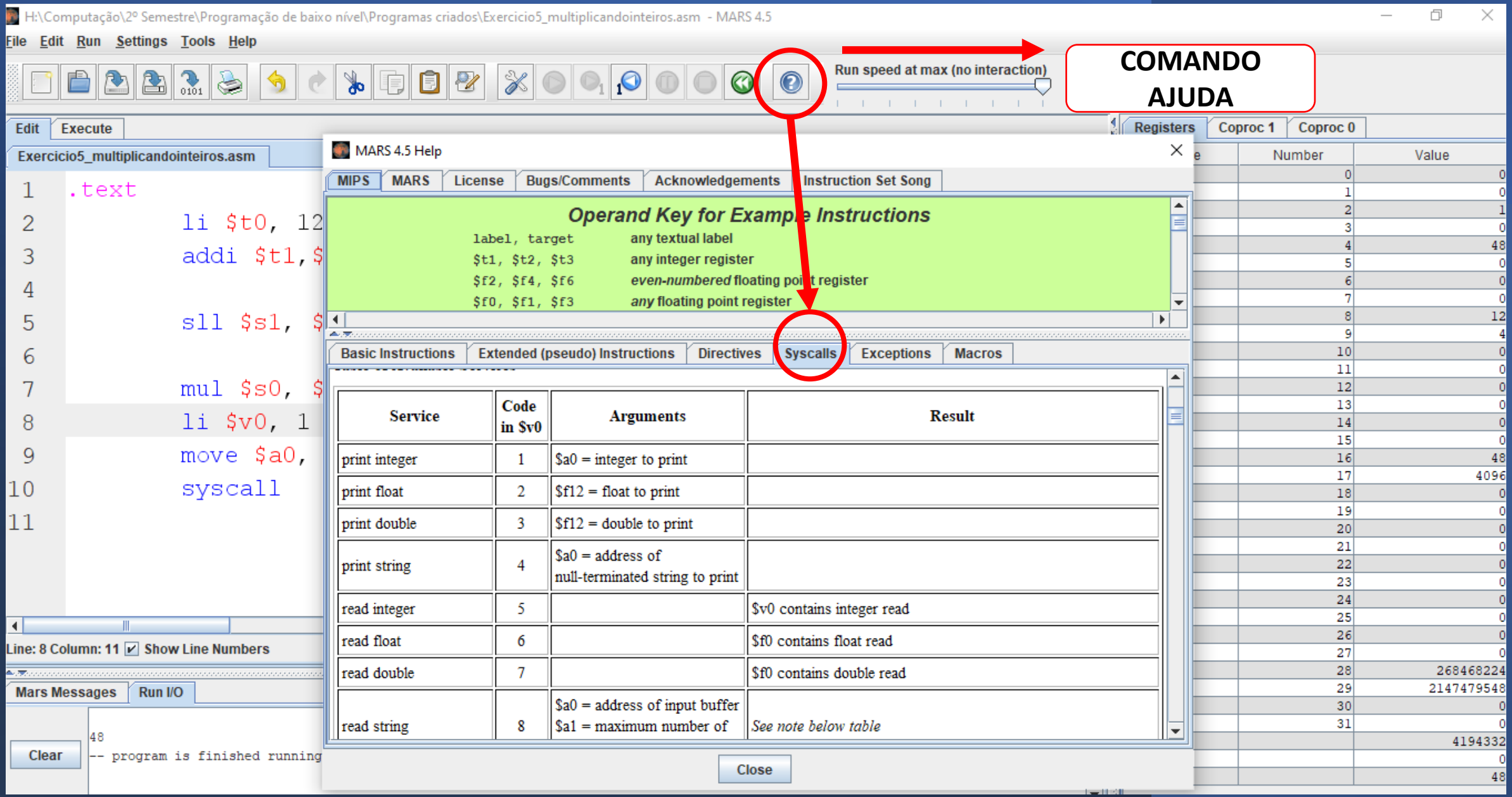
Multiplicação de inteiros

Instrução	O que representa?
mul $s0, $t0, $t1	s0= t0 * t1

Edit | Execute

Exercicio5_multiplicandointeiros.asm*

```
1   .text
2           li $t0, 12
3           addi $t1,$zero, 4
4
5       #$s0 terá o resultado da multiplicação
6           mul $s0, $t0, $t1
7           li $v0, 1
8           move $a0, $s0
9           syscall
10
```


H:\Computação\2º Semestre\Programação de baixo nível\Programas criados\Exercicio5_multiplicandointeiros.asm - MARS 4.5
File Edit Run Settings Tools Help
Run speed at max (no interaction)
COMANDO AJUDA
Edit Execute
Exercicio5_multiplicandointeiros.asm
1 .text
2 li $t0, 12
3 addi $t1,$
4
5 sll $s1, $
6
7 mul $s0, $
8 li $v0, 1
9 move $a0,
10 syscall
11
Line: 8 Column: 11 Show Line Numbers
Mars Messages Run I/O
48
-- program is finished running
Clear
MARS 4.5 Help
MIPS MARS License Bugs/Comments Acknowledgements Instruction Set Song
Operand Key for Example Instructions
label, target any textual label
$t1, $t2, $t3 any integer register
$f2, $f4, $f6 even-numbered floating point register
$f0, $f1, $f3 any floating point register
Basic Instructions Extended (pseudo) Instructions Directives Syscalls Exceptions Macros
Service | Code in $v0 | Arguments | Result
print integer | 1 | $a0 = integer to print |
print float | 2 | $f12 = float to print |
print double | 3 | $f12 = double to print |
print string | 4 | $a0 = address of null-terminated string to print |
read integer | 5 | | $v0 contains integer read
read float | 6 | | $f0 contains float read
read double | 7 | | $f0 contains double read
read string | 8 | $a0 = address of input buffer $a1 = maximum number of | See note below table
Close
Registers Coproc 1 Coproc 0
Number Value
0 0
1 0
2 1
3 0
4 48
5 0
6 0
7 0
8 12
9 4
10 0
11 0
12 0
13 0
14 0
15 0
16 48
17 4096
18 0
19 0
20 0
21 0
22 0
23 0
24 0
25 0
26 0
27 0
28 268468224
29 2147479548
30 0
31 0
4194332
0
48

MULTIPLICAÇÃO DE POTÊNCIA DE DOIS

Multiplicar números por **potências de 2** é trivial para o computador. Basta realizar à operação de ***shift left***, que significa mover os bits para à esquerda.

Se movermos os bits de um número binário uma casa para à esquerda, multiplicaremos por 2.

Se movermos os bits de um número duas casas para esquerda, multiplicaremos por 4.

Se movermos os bits de um número “n” casas para esquerda, multiplicaremos por 2^n.

Não precisa usar a instrução **mul**

Exemplo

$000110_2 = 6_{10}$

0	0	0	1	1	0

0	0	1	1	0	0

SHIFT LEFT 1 = 12 pois $6^2 = 12$

$000100_2 = 12_{10}$

0	0	0	1	0	0

0	1	1	0	0	0

SHIF LEFT LÓGICO

- **Sll $t0, $t1, n**
- **Se usarmos o shift lógico (sll) neste caso estamos pegando o valor de t1 e colocando o resultado em t0. Ou seja, t0 = $t1*2^n$.**

Utilizando Shift Left

Exercicio5_multiplicandointeiros.asm

```
1   .text
2           li $t0, 12
3           addi $t1,$zero, 4
4
5           sll $s1, $t1, 10
6
7           mul $s0, $t0, $t1
8           li $v0, 1
9           move $a0, $s0
10          syscall
11
```

Referências

MANZANO, J. A. N. G. **Fundamentos em Programação Assembly**. 1. ed. Editora Érica, 2004.

HENNESSY, J. L.; PATTERSON D. A. **Organização e Projeto de Computadores:** A Interface Hardware/Software. 4. ed. Editora Elsevier, 2013.