

**CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS  
GERAIS –  
CEFET/MG**

Diogo Emanuel Antunes Santos (20213002091)

**AULA 3: PROGRAMANDO EM ASSEMBLY DO MIPS (INSTRUÇÕES  
COM VALORES IMEDIATOS E INSTRUÇÕES DE CONTROLE DE  
FLUXO)**

**Laboratório de Arquitetura e Organização de Computadores**

**BELO HORIZONTE  
2022**

O código foi:

Var1 = 2344

Enquanto Var1 não for igual a 1

Se Var1 > 80;

Var1 = Var1 - 2;

Caso contrário, Se Var1 <= 80;

Var1 = Var1/4;

Fim do Enquanto

Passei de assembly para MIPS:

The screenshot shows the MARS MIPS assembler interface. The main window displays the assembly code for the provided logic. The code is as follows:

```
1 #Var1 = 2344
2 #Enquanto Var1 não for igual a 1
3 #Se Var1 > 80; Var1 = Var1 - 2;
4 #Caso contrário, Se Var1 <= 80; Var1 = Var1/4;
5 # Fim do Enquanto
6
7
8 .data
9     var1: .word 2344
10    mensagem: .asciiz      #alocação do valor de vari
11
12    lw $s0, var1           #passando vari pra um registrador
13
14 Loop:
15     beq $s0, 1, Fim       #condicional de encerramento do loop
16     addi $t1, $0, 80
17     slt $t0, $t1, $s0
18     beq $t0, $0, Else
19     subi $s0, $s0, 2       #subtração para maiores de 80
20
21 Else:
22     divu $s0, $s0, 4       #divisão para menores de 80
23     j Loop
24 Fim:
25     syscall
```

The right panel shows the registers, with \$s0 containing the value 2344. The bottom panel shows the Mars Messages window, which displays the reset status.

Compilado:

The screenshot shows the MARS MIPS assembler interface with the compiled assembly code and the data segment. The main window displays the assembly code, which is the same as the previous screenshot. The right panel shows the registers, with \$s0 containing the value 2344. The bottom panel shows the Mars Messages window, which displays the reset status.

The Data Segment window shows the memory layout, with the value 2344 stored at address 0x10010000.

Agora responderei as perguntas:

**1) Quais instruções utilizadas são aritméticas? Quais são de operação de transferência de memória? Quais são de controle do fluxo de execução?**

As instruções aritméticas são:

```
addi $t1, $0, 80
```

```
slt $t0, $t1, $s0
```

```
subi $s0, $s0, 2
```

```
divu $s0, $s0, 4
```

Que realizam uma adição, comparação de menor que, subtração, e divisão.

Já para transferência de memória:

```
lw $s0, vari
```

Que faz a movimentação de dados da memória para o registrador.

Já as instruções de controle de fluxo:

```
beq $s0, 1, Fim
```

```
beq $t0, $0, Else
```

```
j Loop
```

```
j Loop
```

Os “beq” desviam para uma determinada execução caso os valores sejam os mesmos. Os “j Loop” definem o loop de repetição do código.

**2. Explique o funcionamento do código acima.**

O código funciona pegando o valor de var1 passando pro registrador \$s0, após isso executa um loop, com a condicional que se \$s0 for igual a 1 ele encerra o loop, dentro dele tem uma condicional se o var1 for maior que 80 o registrador é atribuído com ( $s0 - 2$ ), sendo igual ou menor que 80 registrador é atribuído com ( $s0 * 1/4$ ), cada passagem dessa é executada sequencialmente e repetida no loop com as mesmas condições. As instruções aritméticas, de transferência de memória e de controle de fluxo foram utilizadas para realizar essas atribuições, condicionais e loop.

**3. Este código possui algum erro ou pode ser melhorado? Justifique. (Dica: repare que o tipo da variável “Var1” não foi definido!).**

Há possibilidade de melhora, como já previsto na dica, o tipo de var 1 está indefinido, ou seja, pode trazer erros pois não se sabe se é um número inteiro. Essa declaração é importante para que as operações no código aconteçam adequadamente. Como por exemplo na imprecisão dos resultados das divisões por 4 dos inteiros menores que 80 e diferentes de 1. Dependendo da entrada poderiam causar um loop infinito já que sendo números fracionários poderiam continuar sendo divididos infinitamente entre 0 e 1 se não fossem exatamente igual a 1 em nenhum momento da execução.