**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)

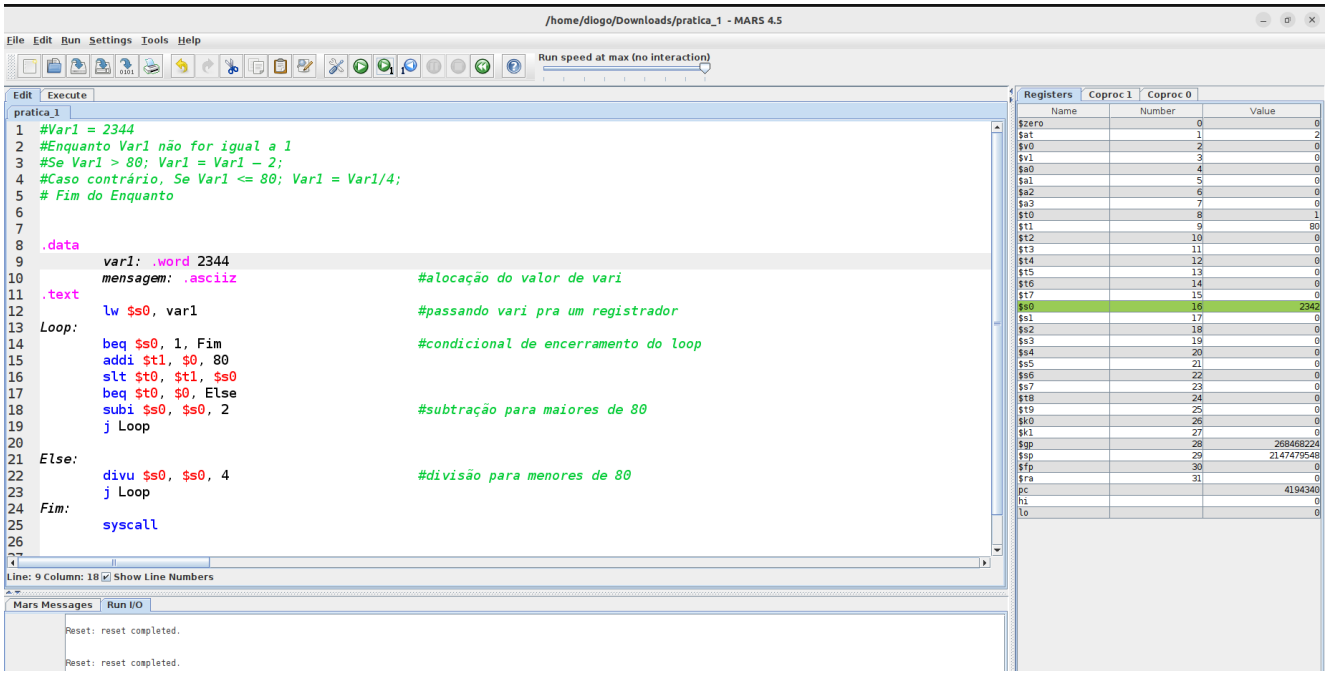
Laboratório de Arquitetura e Organização de Computadores

**BELO HORIZONTE
2022**

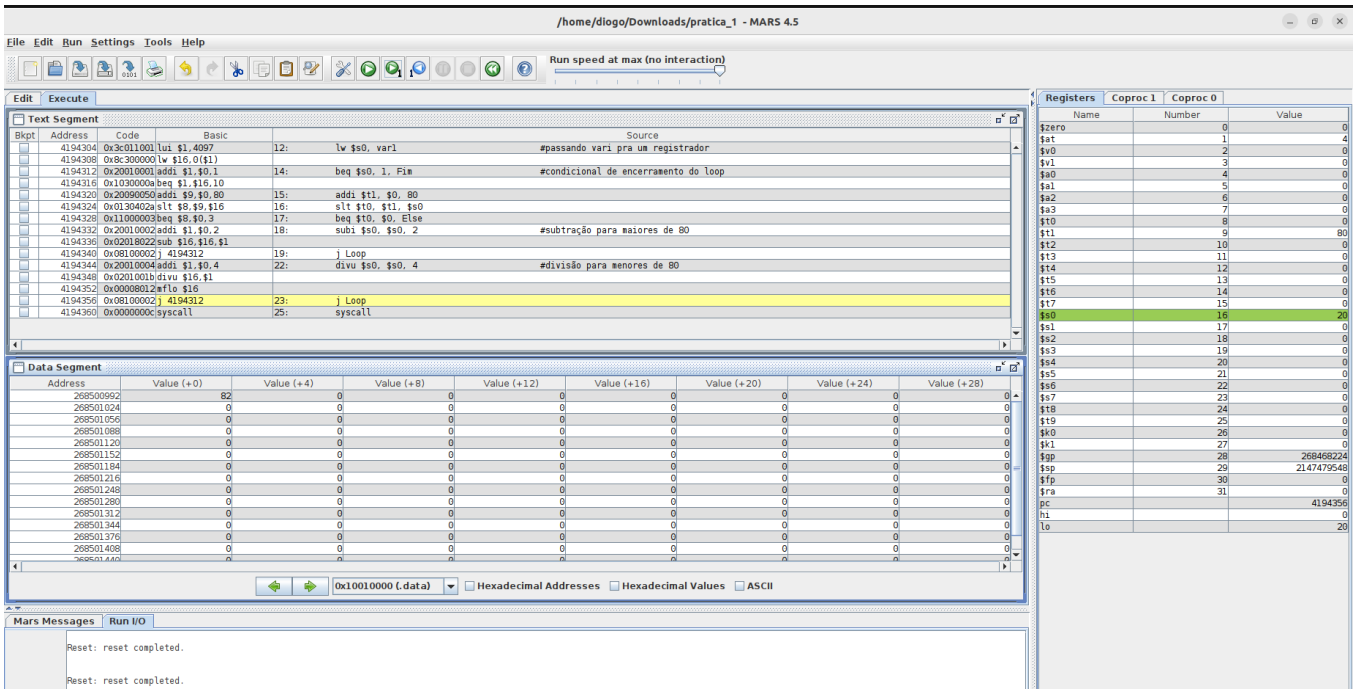
O código foi:

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:



Compilado:



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.