Disciplina: Fundamentos e Arquitetura de Computadores (FAC)
Professor: Tiago Alves
Trabalho 2 - Fundamentos de Arquitetura de Computadores

Alunos	Matrícula	Sistema Operacional
Djorkaeff Alexandre	16/0026822	macOS Sierra
Daniel Maike	16/0117003	Windows 10

O ambiente de desenvolvimento utilizado pelos dois alunos foi o MARS v4.5.

Instruções de uso:

Para simular os arquivos de resposta para as questões deste trabalho foi usado o aplicativo Mars versão 4.5 que pode ser baixado através de (http://courses.missouristate.edu/kenvollmar/mars/).

Após iniciar o aplicativo Mars, devemos abrir o arquivo de resposta da questão, na pasta zipada entregue neste trabalho temos um arquivo, com o nome " *exp_mod.asm* ", estes arquivos são gerados pelo Mars e podem ser lidos pelo mesmo.

Após abrir um dos arquivos será lhe apresentado o código MIPS do mesmo.

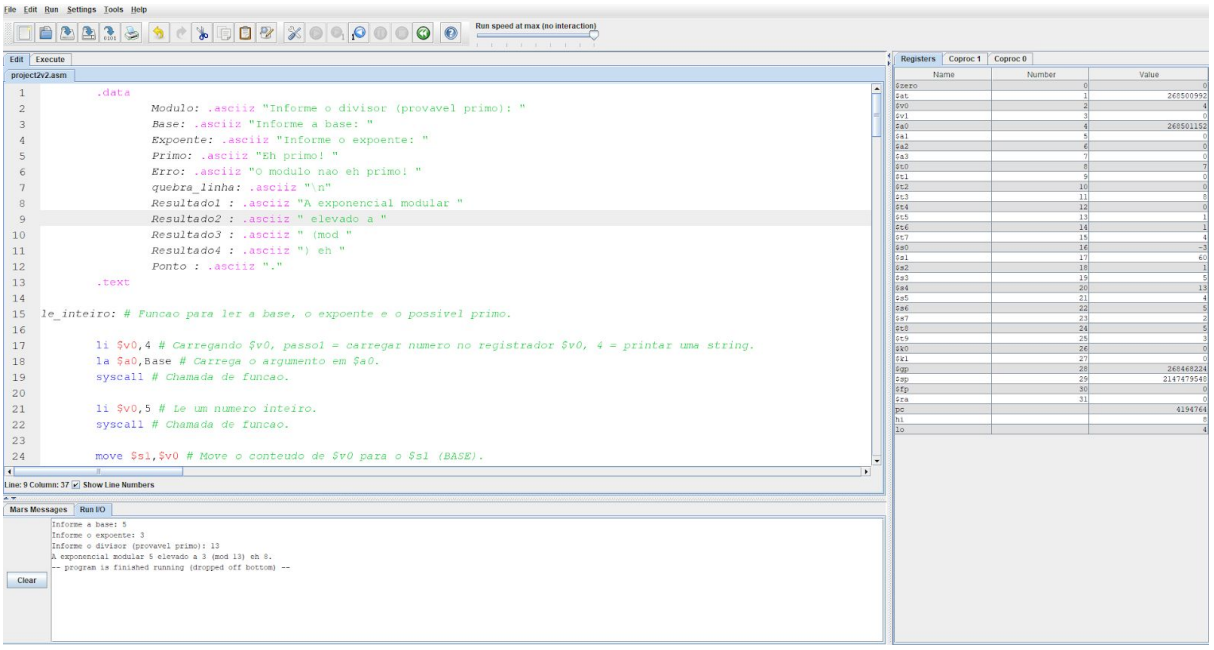

Figura 1 - Abertura de arquivo .asm no simulador Mars v4.5 código "*exp_mod.asm*

Após aberto o arquivo *.asm* devemos montá-lo (simulação), para isso selecionamos no menu superior a opção Run > Assemble.

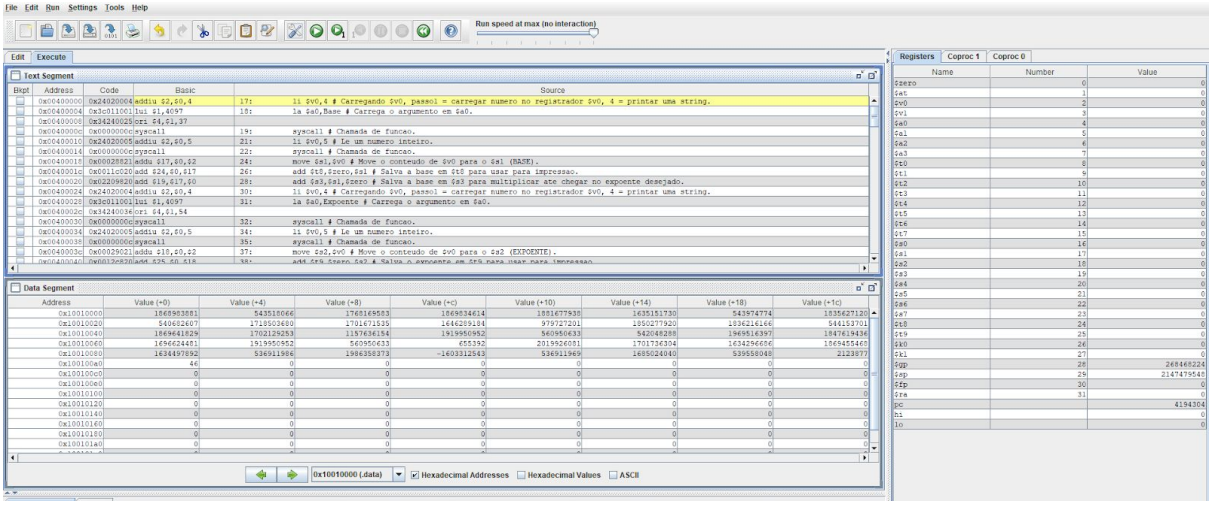

Figura 2 - Lista de comandos executados com o arquivo presente em *Text Segment*

Após clicar na opção Assemble presente na aba Run do menu superior, será apresentada a tela de Text Segment onde estarão presentes as instruções que serão realizadas por este arquivo sobre a memória simulada.

Deve-se clicar no botão indicado ao lado dentro do aplicativo Mars para que as instruções sejam simuladas.

Figura 3 - Saídas produzidas nos registradores pelo código

As entradas para as saídas representadas na figura 3 foram 5, 4 e 2.

Após clicar no botão que simula as instruções são geradas saídas nos registradores, os registradores.

Funcionamento do código:

O código primeiramente irá verificar se o divisor é primo para poder fazer o módulo.
Se for igual a 1 ele irá imprimir o erro diretamente e se for igual a 2 ele irá direto para o cálculo da exponenciação modular.
Para verificar se é primo, o código irá tirar a raiz quadrada do divisor (possível

primo) inserido pelo usuário do programa, se a raiz quadrada dele não for exata, ele irá pegar o número inteiro após o número com casas decimais (por exemplo: raiz quadrada de 6 ele irá pegar o 3), isso irá otimizar o código para que não precise percorrer todo o número digitado (faz com que possamos colocar números de até 31 bits, sendo o último o 2147483647 que é o último primo de 31 bits).

Ele irá percorrer a raíz quadrada do número inserido pelo usuário e fará divisões e quando o resto for igual a zero, ele irá incrementar um registrador utilizado para isso, após percorrer o número todo, se o registrador que incrementa quando o resto é igual a zero for maior que 1 (mais de um número teve resto igual a zero), ele irá imprimir o erro, caso contrário ele irá continuar para o cálculo da exponenciação modular.

Para tirar a exponenciação modular sem que estoure os bits, o código irá verificar se a base é maior que o número primo, se for ele irá calcular o módulo da base e irá voltar para o calc_exp, se não for ele irá verificar se o expoente é par, se for par ele irá dividir por 2 o expoente e irá elevar ao quadrado a base, reduzindo o expoente, caso o expoente seja ímpar ele irá subtrair 1 do expoente e irá pegar a base e multiplicar em um registrador temporário para guardá-lo para poder dividir o expoente por 2 e elevar a base ao quadrado, ele irá fazer esse loop até que o expoente seja igual a 1, com isso irá multiplicar o temporário e o restante da base e fazer o módulo final.

Limitações conhecidas:

Os números deverão ter 31 bits, entre 0 e 2.147.483.647, tanto o possível primo quanto a base elevada ao expoente, com isso o programa rodará corretamente seus resultados.