Software Básico
Trabalho Prático: Montador

Antonio Cortes Rodrigues
Marcelo Luiz Harry Diniz Lemos
Pedro Henrique Martins Brito Aguiar

1 Introdução

Este trabalho consistiu em implementar um montador(*assembler*) para a máquina Swombat.

O assembler implementado leva dois passos para gerar o arquivo final. No primeiro passo, o arquivo de entrada é lido por completo, armazenando todas as *labels* e suas respectivas posições em uma tabela, para que possamos utilizá-las com facilidade no próximo passo. Em seguida, passamos mais uma vez pelo arquivo, desta vez processando as instruções e gerando o arquivo final pronto para ser utilizado na máquina Swombot.

2 Implementação

Está seção tem o objetivo de esclarecer todas as decisões de implementação tomadas pelo grupo no trabalho.

2.1 Primeiro Passo

No primeiro passo caminhamos linha a linha do arquivo fonte em busca de labels. Qualquer linha que não comece com uma label é ignorada nesta etapa, enquanto as que são iniciadas por alguma label são arquivadas em uma tabela (nome e posição) para serem utilizadas posteriormente. Isso é feito, inclusive com as pseudo-intruções .data, até que o arquivo termine. Esse primeiro passo apenas monta a tabela de símbolos, sem realizar nenhuma decodificação ou pré-decodificaçaõ.

2.2 Segundo Passo

Durante o segundo passo é realizada a decodificação das instruções. Agora que nossa tabela já está montada não temos que nos preocupar com rótulos que ainda não tinham sido vistos. Então novamente passamos linha por linha no arquivo de entrada. Primeiro verificamos qual o código de operação da instrução. Em seguida tratamos os operandos da instrução, reconhecemos os tipos e decodificamos na forma devida, completando a decodificação. Como as instruções são de 16 bits e a memória é dividida em palavras de 8 bits, as instruções são divididas em dois endereços de memória seguidos.

Após o término das instruções normais, temos as pseudo-intruções .data. Alocamos a memória de dados, inserindo os dados na mesma ordem em que eles aparecem no arquivo de entrada.

2.3 Saída

Optamos por gerar uma saída .mif pelo fato de que o grupo já estava familiarizado com o formato do arquivo.

2.4 Memória

Em relação a política de alocação, decidimos alocar todas as variáveis após o término das instruções, tendo em vista que esses dados estariam sempre nas últimas linhas do arquivo de entrada. Além disso, escolhemos alocar a memória a partir dos endereços menores para os endereços maiores.

3 Testes

Esta seção é dedicada aos testes realizados com o montador. (As imagens também podem ser vistas em um arquivo a parte, junto da documentação, para uma melhor resolução)

3.1 Teste 1

No primeiro teste realizamos diversas operações não interligadas entre si com o objetivo de testar uma grande variedade de instruções de uma única vez.

Inicialmente realizamos um loop que conta de 10 até zero. Em seguida fazemos $12/4$ e 3^2. Ao final do código executamos $3 - 9$. Durante a execução são utilizadas instruções de load, store, add, jump, entre outras. A pseudo instrução .data também é utilizada para inicializar alguns dos valores que são armazenados nos registradores.

O resultado da execução do código gerado pelo nosso montador é o mesmo da execução do código gerado pelo próprio CPUSim, como podemos ver nas imagens a seguir. A Figura 1 mostra o resultado do nosso código, enquanto a Figura 2 o do próprio CPUSim.

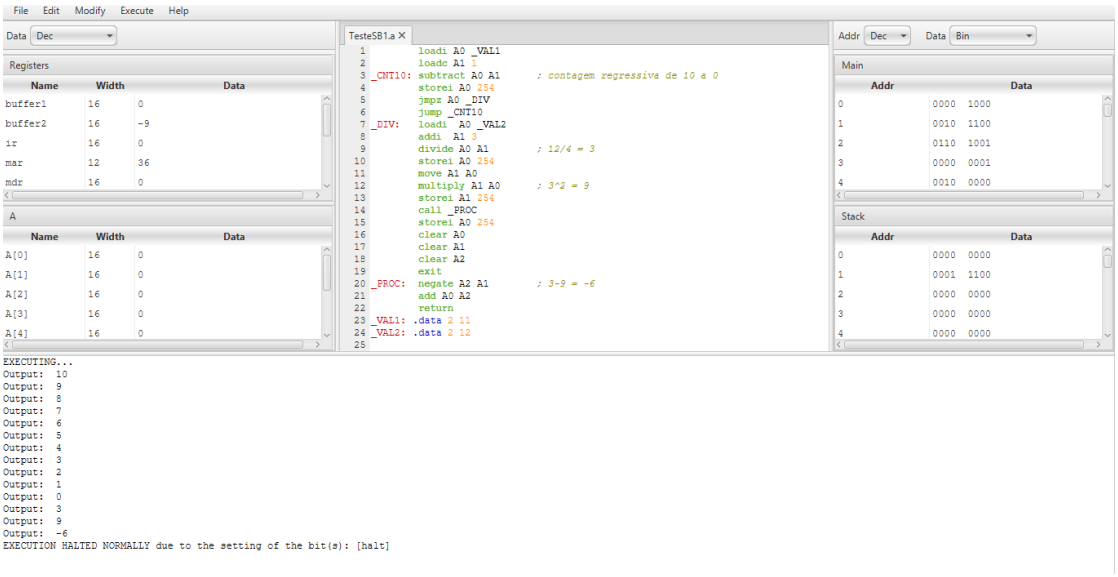

Figura 1: Resultado do primeiro teste com o código gerado pelo montador

Figura 2: Resultado do primeiro teste com o código gerado pelo CPUSim

3.2 Teste 2

Já no segundo teste, realizamos uma função específica no código: Calcular o somatório de zero até um número passado pelo usuário.

Dessa vez utilizamos principalmente as funções de pilha para chegar ao resultado final. Além disso utilizamos algumas outras instruções que não tinham sido testadas anteriormente.

Assim como no primeiro teste, os resultados de nosso montador foram iguais aos do

CPUSim, como mostrado nas próximas figuras.

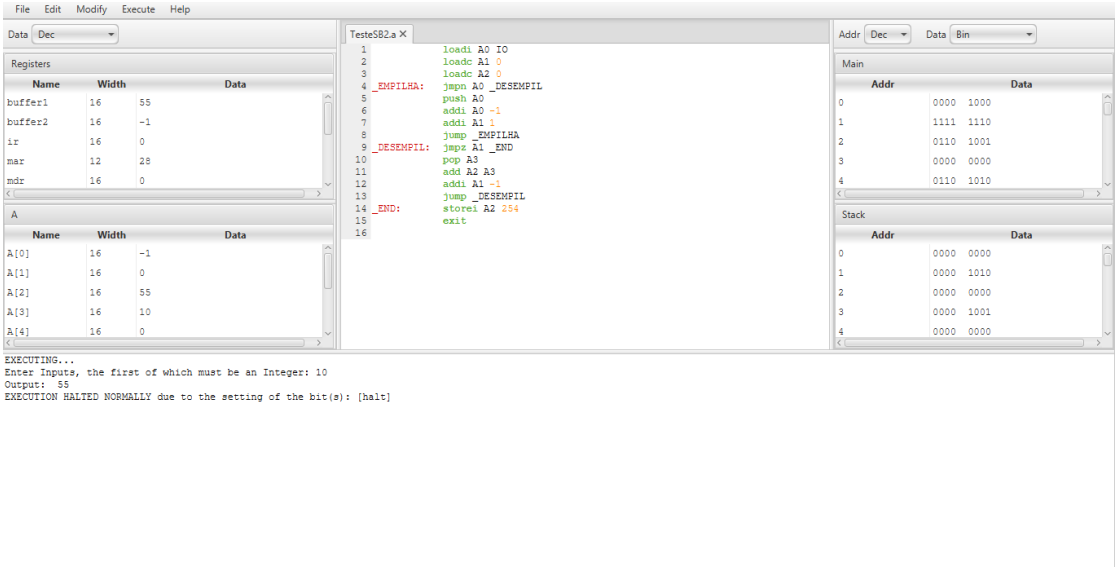

Figura 3: Resultado do segundo teste com o código gerado pelo montador

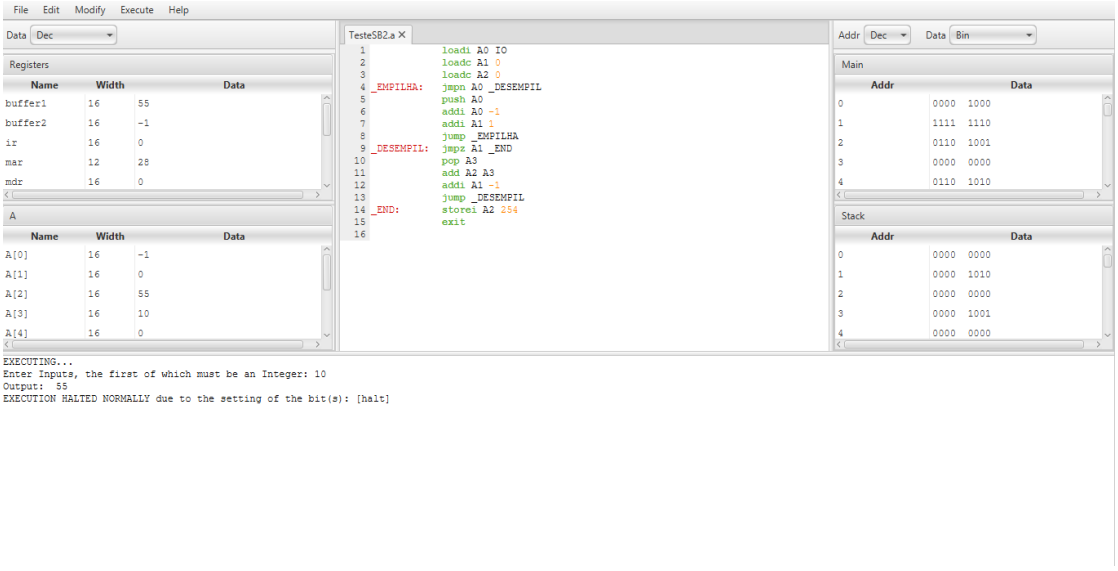

Figura 4: Resultado do segundo teste com o código gerado pelo CPUSim

4 Conclusão

Conseguimos implementar o montador sem grandes dificuldades. Os testes realizados ocorreram da forma esperada e confirmam que o montador está funcionando de forma adequada.