Trabalho Prático 2 - Ligador

Fernanda de Oliveira Ramalho Gustavo Henrique Oliveira Costa
Gabriel Henrique Souto Pires Raydan Elias Gaspar

1 Introdução

No primeiro trabalho prático foi implementado um montador para a máquina Swombat que basicamente traduzia um programa em assembly para linguagem de máquina. Mas o montador conseguia apenas traduzir programas constituídos de apenas um arquivo, programas modulares não funcionariam no montador antigo. Para resolver esse problema, nesse trabalho foi implementado um *ligador* que faz o trabalho de unir os vários módulos que o montador traduz separadamente. Para conseguir ligar os módulos, o montador fornece ao ligador informações adicionais sobre cada módulo como a lista de símbolos contidos no módulo, o tamanho do módulo, e qualquer outra informação que ajude a realocar os endereços de memória usados em cada módulo, além do código binário do programa montado.

2 Desenvolvimento

2.1 Montador

O primeiro passo para desenvolver o ligador foi adaptar o montador do primeiro trabalho para fornecer informações necessárias para unir os módulos. Essa adaptação é necessária, uma vez que sem ela cada módulo de um programa seria montado como um programa independente e teríamos problemas caso um módulo tentasse usar uma função que é declarada em outro módulo, já que o endereço dessa função não seria conhecido.

Além de traduzir o programa como já fazia no primeiro trabalho, o montador agora imprime em um arquivo algumas informações que são lidas pelo ligador na hora de fazer a ligação e relocação. A primeira parte do módulo é constituída pelas informações que o ligador usará para vincular os módulos. Cada módulo começa com a string "MODULE_START" que é usada na hora da ligação para separar o início de cada módulo já que todos os módulos são unidos no mesmo arquivo e depois lidos sequencialmente. Depois temos o nome do arquivo onde o módulo está contido (apenas para fins de debug, esse nome não é usado no ligador) e o tamanho do módulo que é usado mais tarde para atualizar a constante de relocação, essa variável representa quanto de memória cada módulo utiliza. Depois todas as *Labels* e constantes do módulo são escritas uma por linha acompanhadas

por vários números inteiros, o primeiro desses números é o seu endereço na memória e os números seguintes são os endereços de todas as instruções que utilizam a label ou a constante. Um "X" sinaliza o fim dos endereços. Logo abaixo das labels temos uma lista com todas as labels declaradas com *.extern* do módulo seguindo o mesmo padrão para os endereços. As listas de *Labels* e *Externs* são delimitadas pelas strings "END_LABELS" e "END_EXTERNS" respectivamente.

Na segunda parte do módulo temos o código binário já devidamente traduzido. O final do módulo é marcado pela string "MODULE_END".

2.2 Ligador

Ao receber os módulos gerados pelo montador, o ligador primeiro lê todos os arquivos de entrada na função *recebe_entrada()* e concatena todos em um arquivo temporário chamado "ligacao.temp" que serve de guia para o resto do programa. Depois na função *ligacao()* o ligador basicamente desfaz todo o trabalho do montador ao ler todas as *labels* e *externs* de todos os módulos e colocá-los em uma lista juntamente com seus respectivos endereços já somados à constante de relocação. Nesta etapa o ligador também escreve no vetor de memória o código binário de todos os módulos traduzidos pelo montador. Caso exista alguma label repetida (duas labels listadas com o mesmo nome), elas são juntadas em uma só na função *junta_labels()* por motivos de organização.

Depois na função *localiza_externs()* cada extern é localizado na lista de labels para que seu endereço seja alterado e se possa fazer a devida alteração na memória posteriormente.

Com todos os externs localizados e cada label com seus respectivos endereços de memória, os valores de memória escritos pelo montador são recalculados usando a lista de labels como referência para que todos os módulos funcionem como um programa só. Por fim a memória é escrita no formato mif.

3 Programas testados

O principal teste utilizado foi o programa que deveria ser implementado como parte do trabalho, que consiste em cinco operações diferentes, cada uma declarada em um módulo separado.

Inicialmente o programa recebe os 4 dados de entrada, A, B, C e OP, e guarda o valor 1 dentro do registrador A7. Em seguida ocorre uma série de subtrações e comparações para saber qual operação será feita sobre os valores de entrada. Se o valor da operação for 1, será chamado o procedimento _opp1, que retorna o maior número entre 3 três. Caso esse procedimento não seja chamado, o valor da operação é subtraído de 1 e, se for chamado o procedimento _opp2, será retornado o menor de 3 números. Caso o procedimento _opp2 não for chamado, será subtraído 1 do valor da operação e, se for chamado o procedimento _opp3, será retornado a soma dos 3 números. Esse processo de subtração e comparação do valor subtraído com o valor 0 é feito para as 5 chamadas de procedimentos possíveis. Caso o número da operação seja diferente de 1, 2, 3, 4 e 5, o programa não fará nada.

Se o procedimento _opp1 for chamado, será retornado o maior valor entre A, B e C. Esse procedimento chamará outro procedimento, chamado _op1 que se encontra dentro do módulo op1_maior_3.a . Inicialmente, em _op1, move-se o primeiro valor, no caso A, para o registra-

dor A4. Em seguida é feita a subtração de A pelo valor de B, que se encontra armazenado em A2. Se o resultado da subtração for menor que 0, significa que B é maior que A, e nesse caso é chamado outro procedimento, chamado _B_maior_A; porem, se o resultado da substração for maior que 0, significa que A é maior que B, e será chamado o procedimento _A_maior_b. Caso o procedimento _B_maior_A for chamado, esse procedimento chamará outro procedimento chamado _BmaiorA. Inicialmente, o valor de B é copiado para o registrador A5. Em seguida é feita a subtração de B pelo valor de C. Se o resultado da subtração for menor que 0, significa que C é maior que B e, consequentemente, C será o maior valor entre os 3. Para retornar o valor de C, será chamado o procedimento _C_maior que imprime na tela o valor de C. Caso o valor da subtração seja maior que 0, significa que B é maior que C e, consequentemente, B será o maior dentre os 3. Voltando ao caso em que o procedimento _A_maior_B é chamado, esse procedimento chamará outro procedimento chamado _AmaiorB. Inicialmente, o valor de A é copiado para A5 e em seguida é feita a subtração de A por C. Se o valor da subtração é menor que 0, C será o maior entre os 3, se o resultado for maior que 0, nessa caso A será o maior valor dos 3.

Se procedimento _opp2 for chamado, será retornado o menor valor entre 3 números. Esse procedimento chamará outro procedimento, chamado _op2 que se encontra dentro do módulo op2_menor_3.a . Inicialmente, o valor de A é guardado em A4 e em seguida é feita a subtração de A por B. Se o valor da subtração for menor que 0, significa que A é menor que B e será chamado outro procedimento, chamado _A_menor_B. Caso o valor da subtração seja maior que 0, significa que B é menor que A e será chamado outro procedimento, chamado _B_menor_A. Caso o procedimento _A_menor_B for chamado, ele chamará outro procedimento, chamado _AmenorB. Inicialmente o valor de A é copiado para A5 e em seguida é feita a subtração de A por C. Se o resultado for menor que 0, significa que A menor que C e, consequentemente, será o menor entre os 3. Será chamado o procedimento _A_menor que imprime na tela o valor de A. Se o resultado da subtração for maior que 0, significa que C é menor que A e, consequentemente, será o menor dos 3 e seu valor será impresso na tela. Voltando ao caso em que é chamado o procedimento _B_menor_A, esse procedimento chamará outro procedimento, chamado _BmenorA. Inicialmente, o valor de B é copiado para A5 e em seguida é feita a subtração de B por C. Se o resultado da subtração for menor que 0, significa que B é menor que C e, consequentemente, será o menor dos 3. Será chamado o procedimento _B_menor que imprimirá o valor de B na tela. Caso o valor da subtração seja maior que 0, significa que C será menor B e, consequentemente, terá o seu valor impresso na tela.

Se o procedimento _opp3 for chamado, será retornado a soma entre 3 números. Esse procedimento chamará outro procedimento, chamado _op3 que se encontra dentro do módulo op3_soma_3.a . Inicialmente, faz se a soma entre A e B e o resultado é guardado em A. Em seguida esse resultado é somado a C e será guardado em A. Por fim o valor armazenado em A será impresso na tela.

Se o procedimento _opp4 for chamado, será retornado a multiplicação entre 3 números. Esse procedimento chamará outro procedimento, chamado _op4 que se encontra dentro do módulo op4_multiply_3.a . Inicialmente, faz-se a multiplicação entre A e B e o resultado é guardado dentro de A. Em seguida, esse resultado é multiplicado por C e o novo resultado é guardado dentro de A. Por fim o resultado armazenado em A será impresso na tela.

Se o procedimento _opp5 for chamado, será retornado a média entre 3 números. Esse procedimento chamará outro procedimento, chamado _op5 que se encontra dentro do módulo op5_media_3.a . Inicialmente, é feita a soma de A com B e o resultado é guardado dentro de A. Em seguida, esse

resultado é somado com C e o novo resultado é guardado dentro de A. O valor 3 é guardado dentro de A4 e por fim, será feita a divisão da soma dos 3 números por 3. Esse valor será guardado dentro de A que terá seu valor impresso na tela.

4 Testes e simulações

Para testar a saída do ligador carregamos o arquivo *mif* que o ligador gera no CPUSim e rodamos uma vez para cada operação. O primeiro número da entrada é a operação a ser executada e os 3 últimos são os números que são usados nas operações. A saída dos testes segue abaixo:

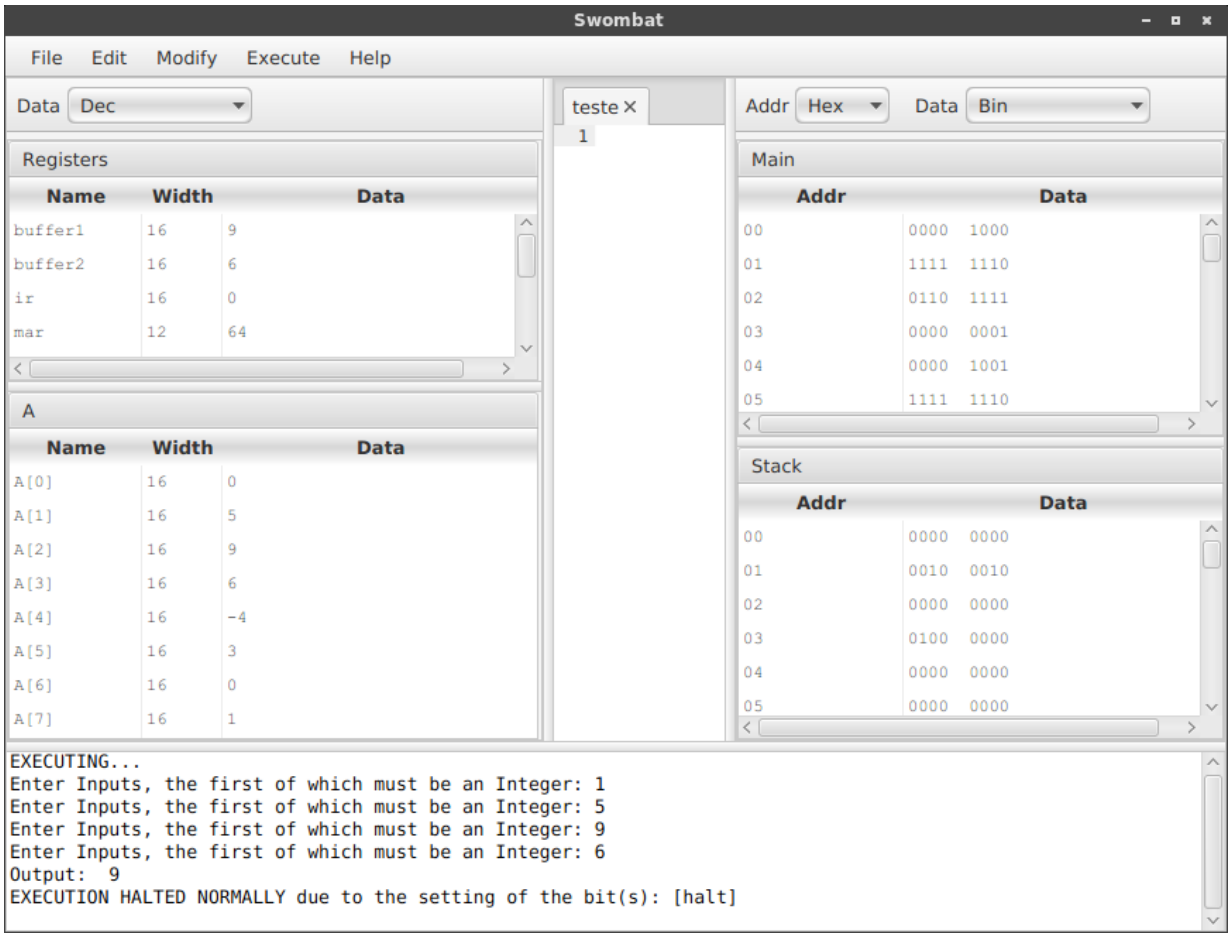

Figure 1: Teste da operação 1 (maior dos números)

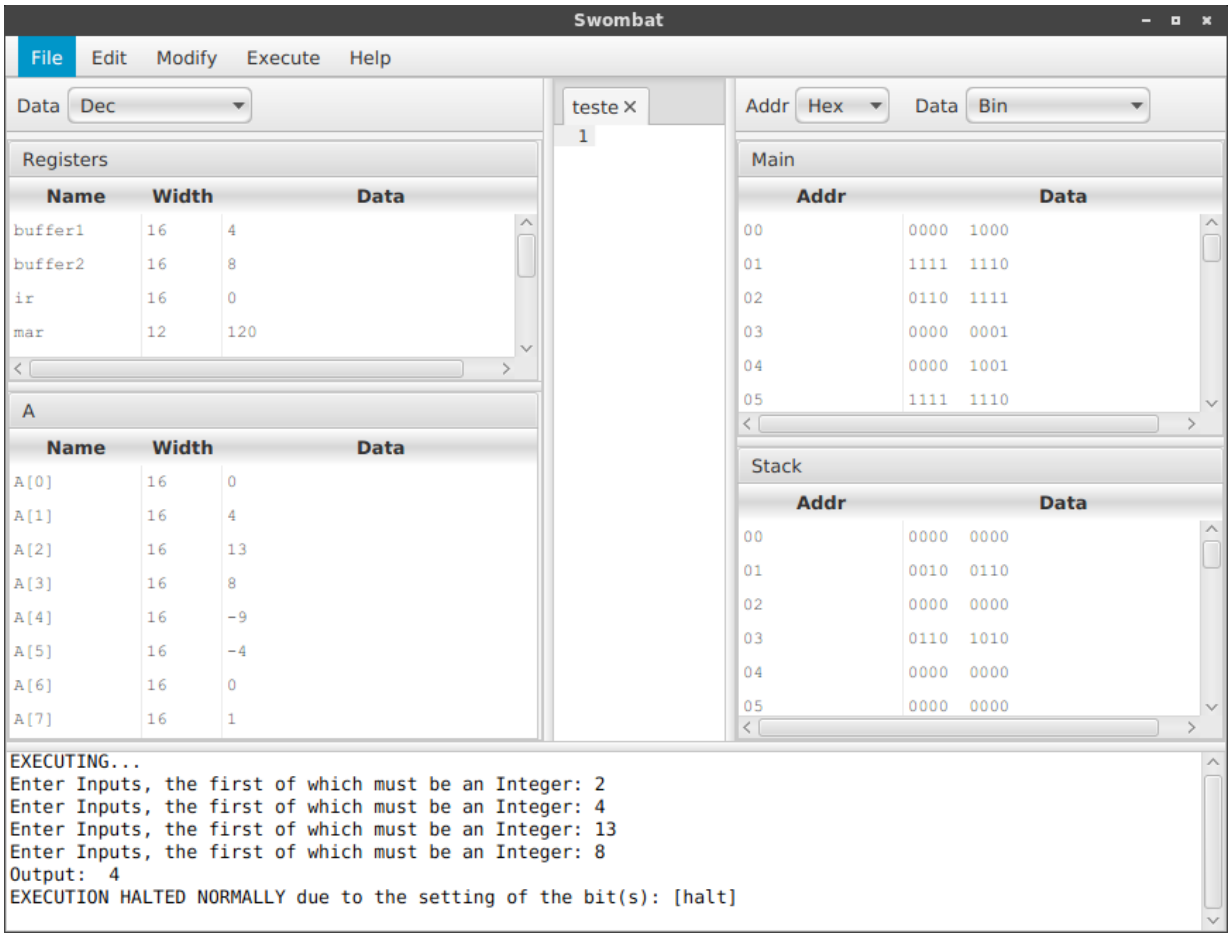

Figure 2: Teste da operação 2 (menor dos números)

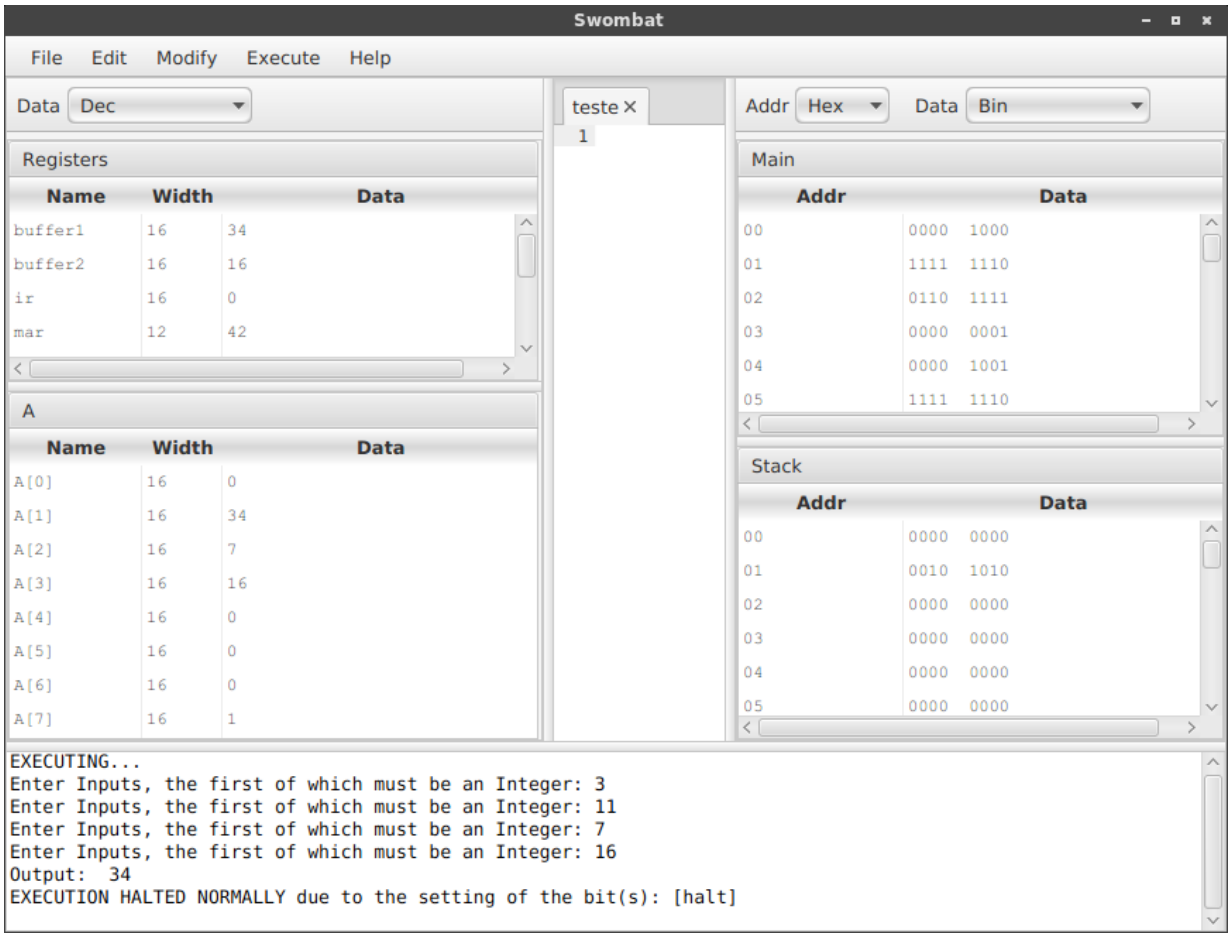

Figure 3: Teste da operação 3 (soma dos números)

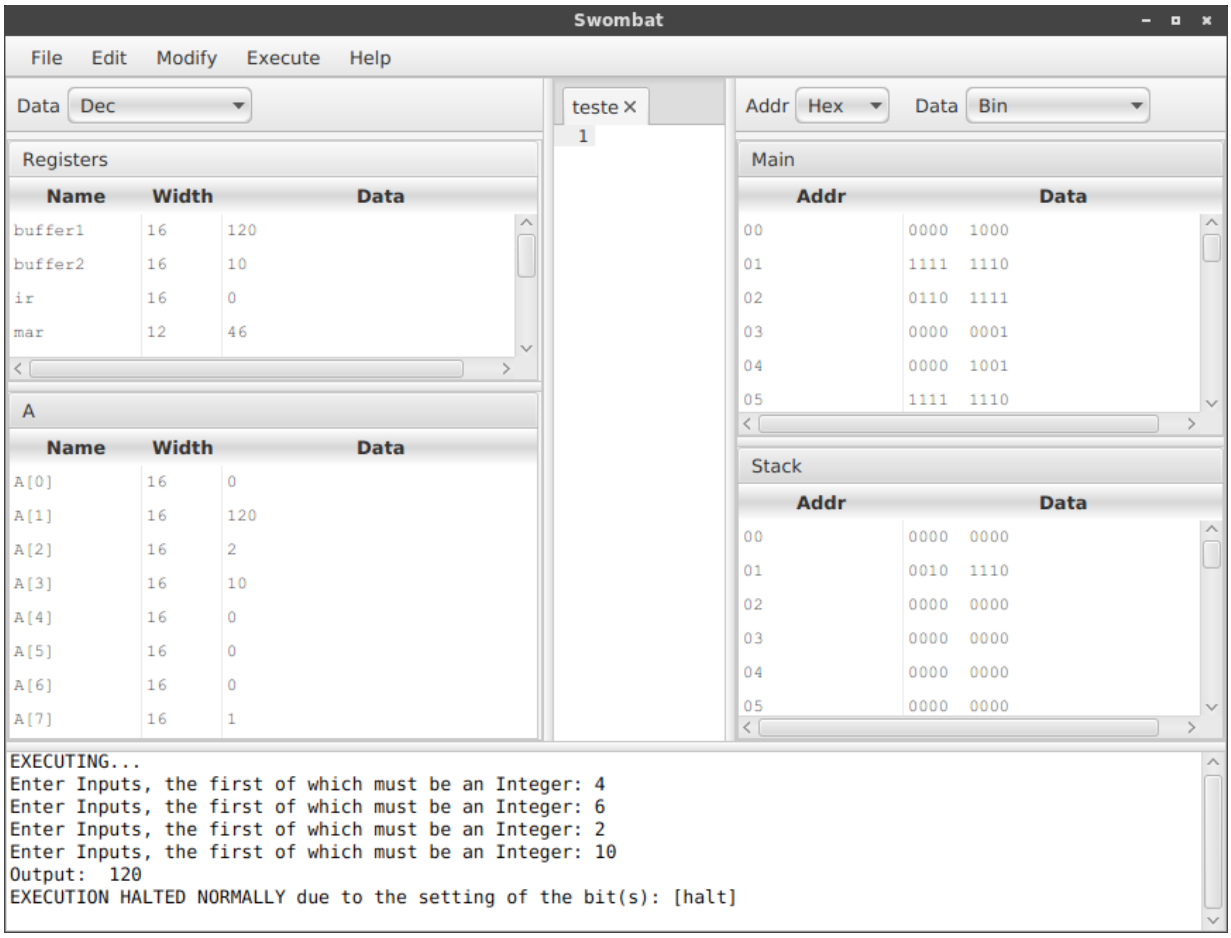

Figure 4: Teste da operação 4 (produto dos números)

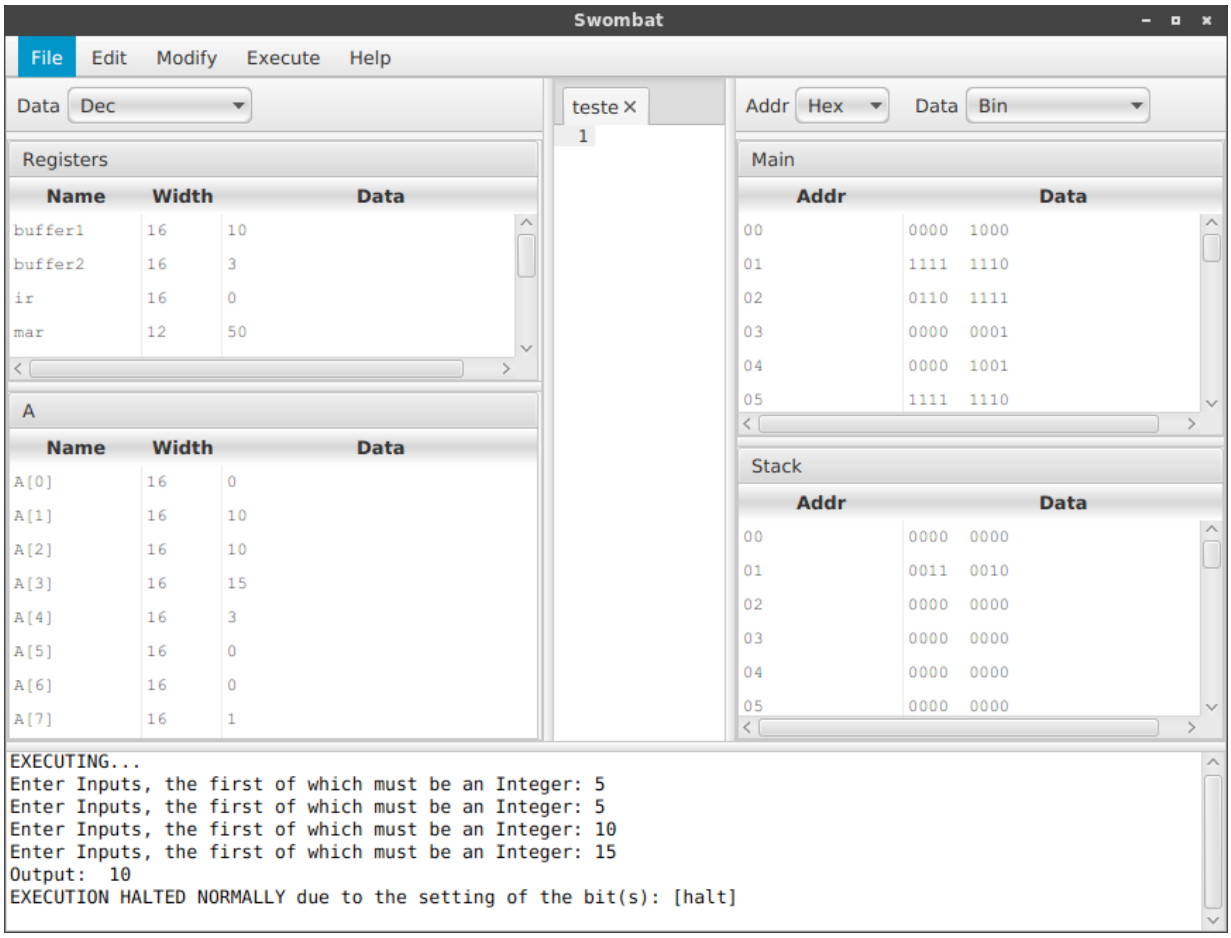

Figure 5: Teste da operação 5 (média dos números)

5 Conclusão

No processo de desenvolvimento do ligador, a parte mais importante acabou se mostrando ser a adaptação do montador. Desenvolver o ligador em si envolveu basicamente criar um método para ler as informações que o montador disponibilizou, o que pode ser interpretado como o trabalho inverso do montador. Talvez criar tudo como um programa só fosse mais eficiente, uma vez que o processo de ligação poderia ser feito direto em tempo de execução e não seria necessário criar arquivos de saída para cada módulo com exceção da saída principal do programa. A dificuldade principal encontrada foi achar um meio de escrever todas as informações necessárias de modo que fosse fácil para o ligador decodificá-las, mas seguindo um padrão próximo ao apresentado no livro da disciplina esse problema pôde ser contornado.