Trabalho Prático 2
Software Básico
Ligador Simples

João Paulo Martino Bregunci - jpbregunci@gmail.com
Ronald Davi Rodrigues Pereira - ronald.drp11@gmail.com

Novembro de 2016

1 Introdução

A tradução de código *Assembly* para código de máquina é um dos problemas mais antigos da Ciência da Computação, dada a óbvia dificuldade da programação diretamente em código de máquina. Desse problema, iniciou-se a proposta desse trabalho que será a construção de um *Ligador* simples, o qual é capaz de gerar código executável para a máquina *Wombat2*. Este dado *Ligador* em questão criado irá possibilitar a execução de funções presentes em diversos outros arquivos, retornando posteriormente ao *Program Counter* a próxima instrução depois que a função foi chamada.

O *Ligador* ao final de todo o processo é capaz de gerar um único arquivo executável, o qual contém todas as funções existentes, sendo o endereço de chamada de todas esses funções recalculado para ajustar-se a chamada de qualquer outra função presente no arquivo executável final.

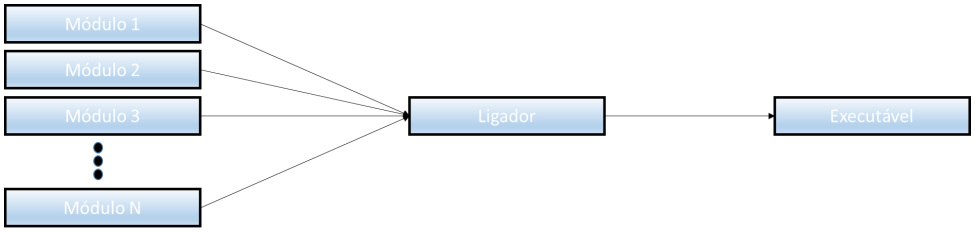

Figure 1: Diagrama que sintetiza o que o *Ligador* faz

2 Explanação do Problema

2.1 Apresentação do Problema

Como enunciado anteriormente na Introdução, o problema consiste na implementação de *Ligador* simples. Nesse contexto, notabiliza-se o maior problema o remanejamento do endereço das funções, ou seja, o recálculo do endereço de memória no qual a função se encontra depois que unem-se vários módulos ao executável principal.

2.2 Decisões para Implementação

Para a construção do ligador deve-se pontuar, primariamente, algumas condições simples para que o programa seja capaz de gerar um executável final sem que exista qualquer tipo de comportamento inesperado. Tais condições enunciadas são:

1. Não deve haver nenhum espaço (ou tabulação) antes de uma linha que contém somente comentários, ou seja, em linhas que contém exclusivamente comentários o primeiro carácter deve ser exclusivamente o ponto e vírgula.

2. Não devem haver nenhuma linha escrita no fim do programa (após os .datas), mesmo que sejam comentários, a fim de evitar comportamentos irregulares durante a execução.

3. Não devem haver nenhum rótulo (label) no meio do programa Assembly que não esteja declarado em outro local do código, a fim de evitar o evidente problema de referências para procedimentos inexistentes.

4. Labels devem começar com um *underline*, tanto para jumps ou calls, a fim de padronizar de uma maneira simples a chamada de procedimentos.

5. Labels referenciados pela pseudoinstrução .extern não devem conter um *underline*, de forma a evitar problemas com a decodificação e comparação das demais ligações.

Dito tal, o *Ligador* é implementado primariamente por meio da criação de um novo arquivo, o qual é elaborado a partir de vários acessos a um *array* com ponteiros para os arquivos dos módulos utilizados. A execução do *assembler* remete fortemente ao trabalho prático anterior, só que agora há uma execução iterativa para cada módulo incluído.

Na implementação realizada o *Program Counter* da *main* tem início no valor zero e os módulos seguem logo abaixo continuando-se a contagem do *PC*. Nessa implementação, uma ressalva muito importante que deve ser feita é que todos os *.data* são impressos ao final de toda a execução da tradução. Esse procedimento só pode ser feito quando todos os *.data* são salvos em uma lista encadeada, a

qual ao final da primeira passada, grava os respectivos *Program Counter* de todos esses *.data*.

O procedimento acima pode ser um pouco de difícil compreensão ao leitor,contudo o esquema abaixo auxília a compreender entender a configuração final do arquivo.

==========

0 : começo da main
1::30 codigo da main
31 : finalização do main
32 : começo do módulo 1
33::48 código do módulo 1
49 : finalização do módulo 1
50 : inicio do .data da main
51 : finalização do .data da main
52 : início do .data do módulo 1
53 : finalização do .data do módulo 1
END

==========

Caso seja do interesse do leitor utilizar o *Ligador*, é necessário ir a pasta *linker* e executar primeiramente o comando *make all*. Após isso é possível utilizar o programa executando o comando:

/linker.exe output.mif main.a modulo1.a modulo2.a [...] modulon.a

2.3 Detalhes de Implementação do Extern

Cada *label* e cada *.data* é atribuído um ID do arquivo de entrada que ele se encontra, o que faz acessos a esses procedimentos somente serem possíveis localmente. Isso é realizado por motivos de segurança e modularização do código, a fim de evitar acessos inesperados a tais referências.

Os *.externs* possuem uma mecânica de funcionamento de maneira similar aos *labels* e aos *.datas*. Na primeira passada são salvos na lista *"ext"* o endereço da memória de instruções e o endereço destino do *label* no seu respectivo módulo. Esse procedimento é útil, pois em cada chamada *.extern*, basta substituir a pseudoinstrução por uma instrução *call*, apontando para o endereço salvo na lista *"ext"*. Portanto, todos os *.extern* ao final da execução são traduzidos para uma instrução *call*, já que endereço de destino já foi previamente calculado para o novo arquivo.

3 Casos de Teste

Abaixo consta a imagem de todos os casos de teste para as diversas operações descritas nas especificações do trabalho. As imagens todas são referentes a simulações feitas na máquina *Wombat2*.

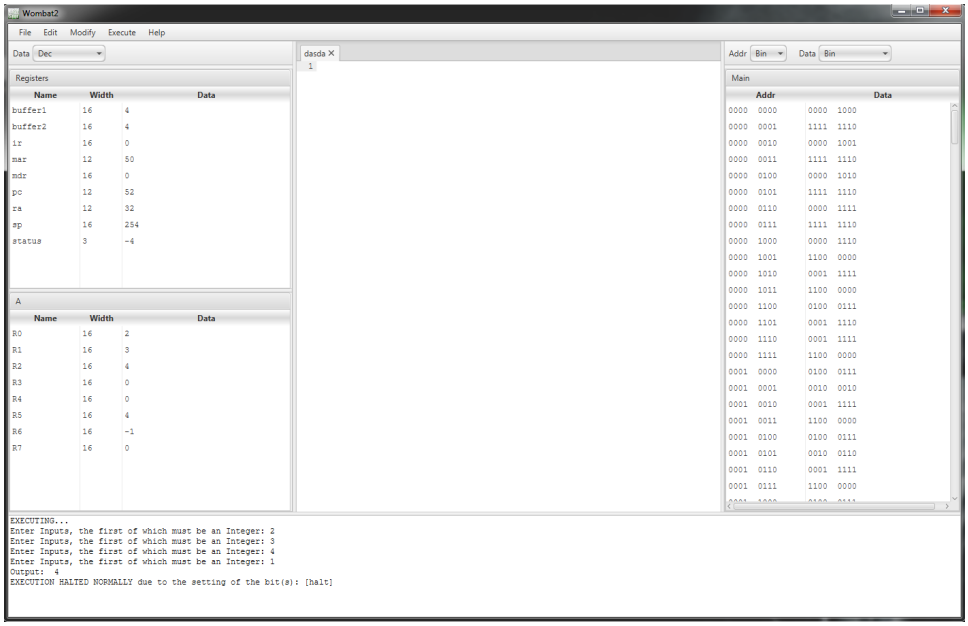

Figure 2: Imagem do programa para a operação 1 (exibir o maior entre os números)

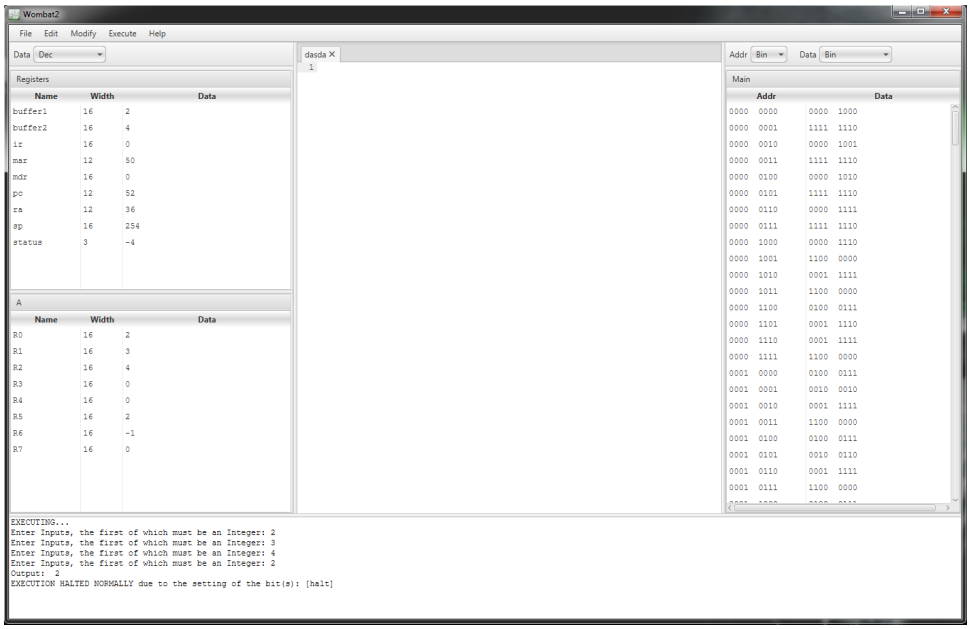

Figure 3: Imagem do programa para a operação 2 (exibir o menor entre os números)

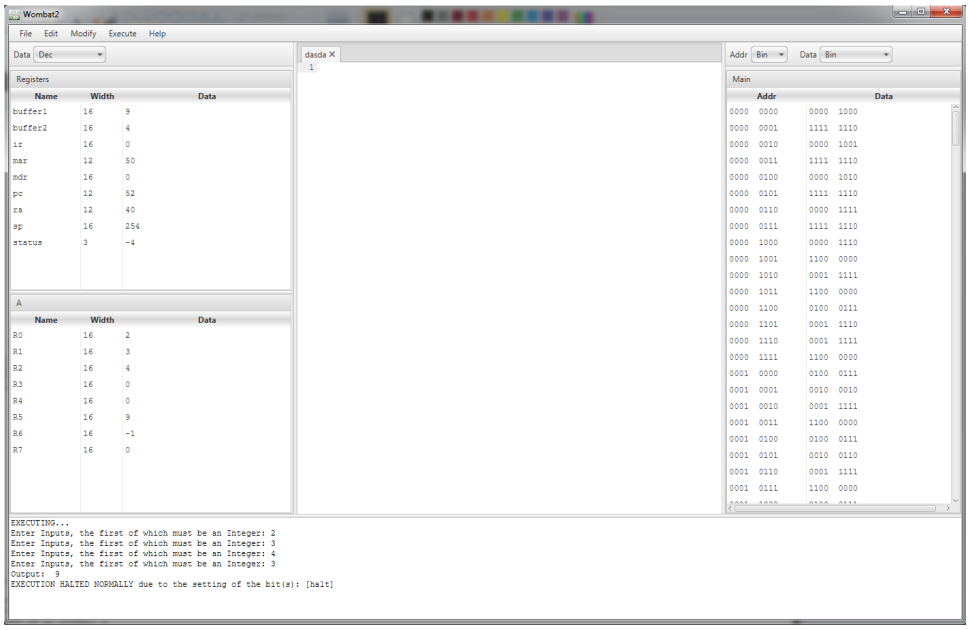

Figure 4: Imagem do programa para a operação 3 (exibir a soma entre os números)

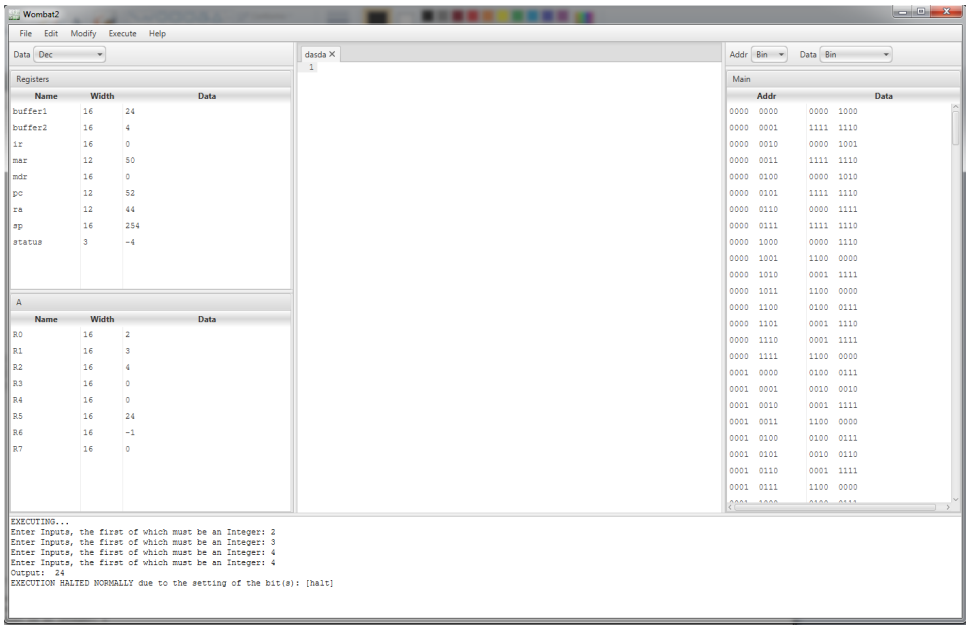

Figure 5: Imagem do programa para a operação 4 (exibir a multiplicação entre o números)

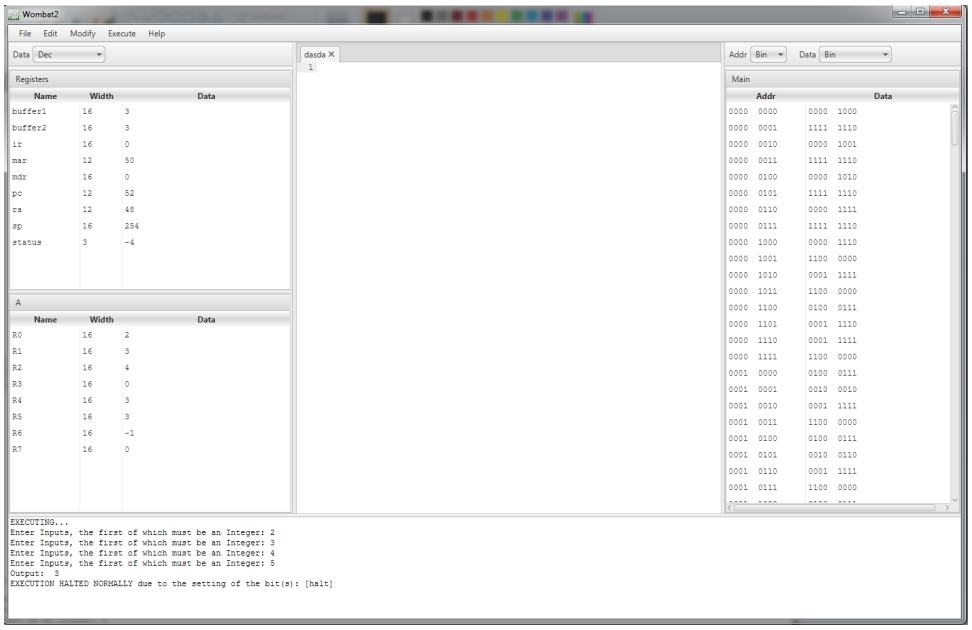

Figure 6: Imagem do programa para a operação 5 (exibir a média aritmética entre os números)

4 Conclusão

Por meio desse trabalho foi possível expandir ainda mais o conhecimento sobre a geração de código máquina, a partir de um código *Assembly*. Foi gerado ao final desse trabalho, um *Ligador* completamente funcional capaz de gerar eficientemente um código executável para a máquina *Wombat2*, utilizando para isso, diversos procedimentos de vários módulos simultaneamente.

References

[1] Colby College, Official CPUsim Website
`http://www.cs.colby.edu/djskrien/CPUSim/`