Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Sistemas embarcados – 2021/1

Trabalho Prático 3 - Ligador

Nome: Lecio Charlles Barbosa
Matrícula: 2016065120

Nome: Breno de Castro Pimenta
Matrícula: 2017114809

1. INTRODUÇÃO

O objetivo deste trabalho é criar um ligador (linker) da arquitetura da máquina Swombat R3.0, para os arquivos gerados por um montador (assembler) desenvolvido no trabalho anterior, esse ligador juntará os arquivos.hex formatados corretamente, gerando assim um arquivo que pode ser executado utilizando o simulador CPUSim. Descrições mais completas do simulador e do montador podem ser encontradas na documentação do trabalho anterior.

2. IMPLEMENTAÇÃO

Nesta seção serão tratadas as decisões de implementação, desde o detalhamento da lógica à partes do código desenvolvido. O primeiro ponto notório a ser mencionado foi a escolha da linguagem, este trabalho foi implementado utilizando Python versão 3. Já as demais decisões estão descritas nos próximos subitens.

2.1. Arquivos

O projeto é dividido em três pastas, “*assembler*” onde está contido o código fonte do projeto (ver item 2.2), “*tst*” onde estão os programas de teste (ver item 3) e “*doc*” que tem como único objetivo armazenar a documentação.

2.2. Código

O arquivo de código que contém o montador é o *assembler/main.py*, já o que contém o ligador é o arquivo *assembler/ligador.py*. O ligador deve receber como entrada uma lista de arquivos montados ‘*hex*’ (ver item 2.4) e deve juntar e ligar os símbolos desses arquivos e gerar um arquivo de saída que contemple a execução de ambos. Os símbolos devem ser dispostos no código no seguinte formato:

<pseudo-instrução> <rótulo>

Onde *pseudo-instrução* indica qual é o tipo de símbolo: se referencia um módulo ou dado externo. Temos então a seguinte lista de pseudo-instruções:
.externT: referência uma tarefa externa
.externD: referência um dado externo
.globalT: tarefa que pode ser referenciada por qualquer outro módulo
.globalD: dado que pode ser referenciado por qualquer outro módulo

Dentre as pseudo-instruções acima as únicas não implementadas no código foram a .*externD* e .*globalD*, mas uma forma de implementá-las seria resolver os símbolos e em seguida caso seja .*externD*, o ligador deve buscar pelo label, indicado por .globalD em outro módulo e copiar o dado, substituindo-o na pseudo-instrução .externD com os dados que ele referencia, ao invés de realizar o jump, como no caso do texto.

2.3. Execução

Para executar o programa, pode-se utilizar o Makefile. Ao executar o comando **make** dentro da pasta assembler, ele montará e ligará os 2 programas escritos para a maquina Swombat em sequência (ver item 3), gerando arquivos resultantes em formato Intel HEX. Caso haja necessidade de se adicionar mais testes, basta criar um arquivo dentro da pasta tst com o padrão de nome “testeN.a” onde N é o numero do teste em questão e modificar o código apropriado no Makefile.

Outra forma de utilizar o programa é através do terminal, dentro da pasta assembler:

1. Monta-se qualquer módulo:

```
>python3 main.py “arquivo_assembly.a” “arquivo_de_saida.a”
```

2. Liga-se os dois módulos em um terceiro resultante:

```
>python3 ligador.py “arquivo1.hex” “arquivo2.hex” “saída.hex”
```

2.4. Entrada

Para executar o ligador a máquina deve receber como entrada no mínimo três arquivos no formato Intel HEX: o último arquivo será a saída e os primeiros serão ligados. Esses arquivos devem obedecer a algumas regas: Dado um conjunto de arquivos como entrada, ele não deve possuir símbolos .globalT/D duplicados e cada .externT/D deve possuir exatamente um .globalT/D correspondente. Para que o ligador funcione, deve haver exatamente um símbolo ***.globalT main*** indicando o início do programa. O ligador procura por esse símbolo e coloca o módulo em

primeiro lugar no arquivo de saída, então os arquivos podem ser dados na entrada em qualquer ordem.

Esses arquivos de entrada devem ter sido montados seguindo as especificações do item (2.3) de forma que ao terem sido compilados foi gerado tanto um arquivo com a extensão ".hex" (que será passado diretamente para o ligador), como também, outro arquivo com a extensão ".sym" (que o programa buscará de forma automática) o qual trata da tabela de símbolos. É comum modularizar um programa para que o código fique mais organizado e economize no tempo de compilação, para não ter necessidade de recompilar todo o código a cada mudança em partes isoladas.

2.5. Saída

Ao ligar os dois arquivos será criado um arquivo texto também com a extensão *.hex* (para mais detalhes sobre essa extensão ver documentação do trabalho anterior). Esse arquivo representa um executável que contém os arquivos de entrada ligados.

3. TESTES

Os testes estão presentes na pasta *tst e* os dois programas testados são a ligação dos 4 programas testes do trabalho anterior. Perceba que há um módulo proc.a que imprime o algarismo 6 no console e não altera o estado do program, ele é chamado dentro do código do teste1 e teste2, note que não temos que remontá-lo, uma vez já tendo o feito previamente. Abaixo há a descrição dos três programas testes disponibilizados no projeto:

O Programa **teste1.a** calcula o fatorial do número dado como entrada e imprime o valor na tela, fizemos este usando chamadas recursivas.

O Programa **teste2.a** lê inteiros da entrada ate que o 0 apareca, em seguida ele imprimira no console o maior número da entrada.

O Módulo **proc.a** imprime o valor 6 no console e não modifica o estado da máquina.

As figuras na próxima página apresentam a execução dos programas ligados desses testes junto ao simulador CPUSim. A Figura 01 apresenta a ligação do teste1 com o proc e a Figura02 apresenta a ligação do mesmo proc com com o teste2.

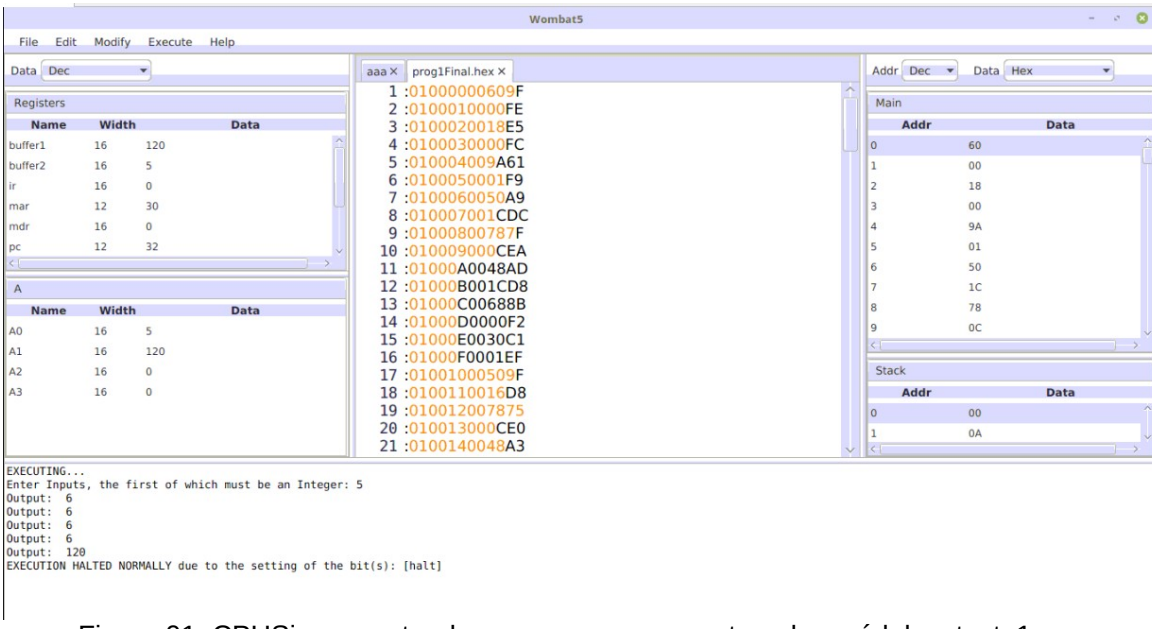

Figura 01: CPUSim executando o programa composto pelos módulos: teste1 e proc.

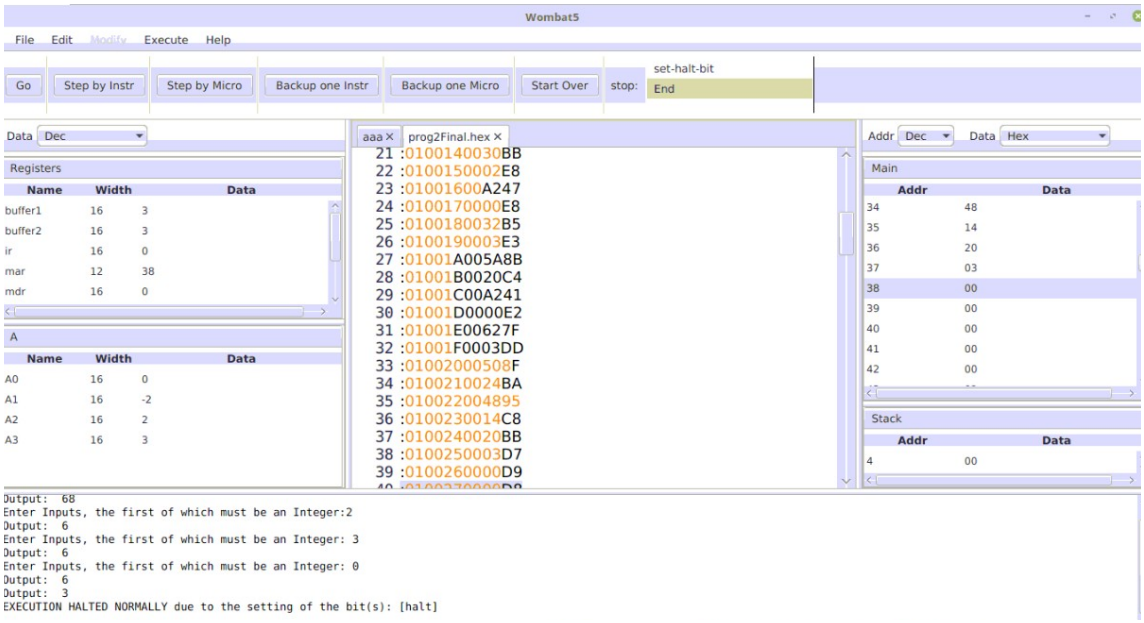

Figura 02: CPUSim executando o programa composto pelos módulos: teste2 e proc.