

2º. Trabalho Prático
MC404 - Organização Básica de Computadores e
Linguagens de Montagem
Prof. Ricardo Pannain
2º. Semestre de 2015

1. Introdução

O trabalho consiste em implementar um simulador IAS na linguagem de montagem ARM.

2. Método

Um simulador computacional de uma arquitetura é um programa capaz de executar software escrito para uma arquitetura "hóspede" (ou simulada) numa arquitetura "hospedeira", sendo idealmente capaz de ler, carregar e executar tais programas de modo transparente no hospedeiro. Formalmente, um simulador desse tipo é denominado uma máquina virtual.

Um simulador busca se comportar de modo bem próximo a uma máquina real (física), e em geral, possui módulos simulando a memória, dispositivos de entrada/saída e o processador da arquitetura hóspede. Da mesma forma que um computador real, a máquina virtual executa instrução por instrução do programa que está sendo simulado e para tanto, geralmente, implementa todos os passos para executar uma instrução, isto é, busca, decodificação, execução, etc. Além disso é comum que um simulador disponha de uma interface para permitir a inspeção, em tempo de execução, do programa sendo simulado e do estado atual da máquina virtual. Note que o simulador deve executar as instruções, contudo não é obrigatório que internamente ele se comporte exatamente como o hardware original da arquitetura hóspede, basta que a arquitetura seja simulada fielmente, a microarquitetura não precisa ser simulada de modo preciso (embora, como dito acima, geralmente os simuladores também são fiéis à microarquitetura).

Nesse trabalho você irá implementar um simulador da arquitetura IAS (arquitetura hóspede) escrito completamente em linguagem de montagem do ARM (arquitetura hospedeira). Tal simulador deverá ser capaz de executar as instruções da arquitetura do IAS e, além disso, produzir informações que permitam ao usuário visualizar o estado da máquina virtual.

3. Especificação

O arquivo de entrada do simulador é um mapa de memória que contém linhas no formato: AAA DD DDD DD DDD

Este arquivo deve ser passado como entrada padrão para o simulador, seguindo o exemplo abaixo:

```
usuario@maquina$ ./simulador < arquivo.hex
```

Desta forma, o simulador consegue ler o conteúdo do arquivo sem precisar abri-lo explicitamente, bastando usar as rotinas de leitura/escrita da biblioteca padrão da linguagem C (scanf, printf).

Apenas estas funções podem ser usadas.

Como anteriormente especificado, AAA é uma sequência de 3 dígitos hexadecimais que representa o endereço de memória, totalizando 12 bits. Já DD DDD DD DDD é uma sequência de 10 dígitos hexadecimais, que totaliza 40 bits e representa um dado ou duas instruções do IAS, conforme já visto em aula. Note que existem caracteres de espaço na linha, num total de exatos 4 espaços. Apesar de outras representações serem usadas para o mapa de memória, é importante que seu simulador respeite esse formato para permitir a execução dos casos de teste. Não deve haver caracteres extras ou linhas em branco, apenas linhas no formato acima.

O trabalho deve ser implementado em linguagem de montagem do ARM, sendo permitindo somente o uso da função printf e scanf, da biblioteca padrão C. A saída do simulador deve ser idêntica à do simulador IAS utilizado na primeira metade da disciplina, de forma a simplificar a correção. Ou seja, o simulador implementado na linguagem de montagem ARM deve imprimir a instrução atual e estado de execução na medida que progride com a execução do programa. O simulador IAS escrito em C, cujo código-fonte encontra-se na pasta compartilhada do Google Drive, pode ser utilizado como referência, entretanto é exigida a confecção inteiramente manual do simulador na linguagem ARM. Para simplificação, a simulação pode considerar que os registradores IAS possuem apenas 32 bits de precisão, o que simplifica a implementação das instruções aritméticas. A implementação da aritmética de 40 bits será tratada como bonificação.

4. Entrega e Avaliação

A linguagem de programação a ser utilizada para desenvolver o simulador devem ser obrigatoriamente a linguagem de montagem ARM. Não serão aceitos trabalhos que façam uso de outra função que não seja printf e scanf da biblioteca padrão do gcc. Apenas as funções printf e scanf são permitidas. O trabalho pode ser entregue nas modalidades individual ou em dupla, mas trabalhos em dupla podem ter arguição adicional. Caso haja qualquer tentativa de fraude, como plágio, todos os envolvidos receberão nota 0 na média da disciplina. O prazo de entrega do trabalho é 16 de novembro de 2015 até às 12:00hs, impreterivelmente. A entrega consistirá em:

- Código-fonte completo e comentado do simulador, além de um arquivo Makefile que gere o executável como regra padrão que deverá ser submetido na pasta a ser criada no Google Drive;
- O arquivo devem ter o nome raXXXXXX.s, em que XXXXXX é o número de seu RA.
- O executável do simulador, que será gerado pelo Makefile, deve ser nomeado raXXXXXX (sem extensão mesmo!). Em caso de trabalho feito em dupla, utilize a convenção raXXXXXX_raYYYYYY
- Programas de exemplo que demonstrem o funcionamento correto do simulador.

5. Bônus

Precisão de 40 bits

Trabalhos que implementem corretamente aritmética com precisão de 40 bits, incluindo instruções de multiplicação e divisão, receberão bonificação na correção. A implementação da aritmética não precisa ser particularmente eficiente, mas precisa estar absolutamente correta para concorrer ao ponto adicional.

6. Dicas

Faça o melhor proveito possível das funções printf e scanf, da biblioteca padrão da linguagem de programação C. Para isso, estude cuidadosamente como utilizar os mecanismos de chamada de procedimento para fazer uso dessas rotinas. Os trabalhos serão executados em uma das plataformas ARM disponíveis para produção do trabalho.