SIMULADOR ARQUI64

Trabalho prático final da disciplina GCC117 – Arquitetura de Computadores I – 2019/1
Turma 10A – Prof. André Vital Saúde

Implementar o simulador Arqui64[1], um simulador parcial do processador ARMv8 no estado de execução de 64 bits, ISA AArch64, seguindo os requisitos descritos a seguir.

1. GRUPOS

O trabalho será realizado em grupos de no mínimo 3 e no máximo 4 pessoas. Se aparecerem pessoas sem grupo ou grupos com outro número de membros, a primeira coisa a ser feita será a tentativa de aloca-los em grupos que estejam apenas com 3 pessoas, mas se não for possível, serão retirados membros de grupos de 4 pessoas, aleatoriamente, para criar um novo grupo.

2. LINGUAGEM DE PROGRAMAÇÃO E APRESENTAÇÃO

O trabalho pode ser implementado em qualquer linguagem de programação de interesse, desde que atenda aos requisitos de teste e interação com usuário e de operações lógicas bit a bit (ver Seção 1.4 deste link https://en.wikipedia.org/wiki/Operators_in_C_and_C%2B%2B), mas preferencialmente C, C++, Java ou Python.

3. VISÃO GERAL

Este trabalho visa a implementação parcial de um simulador funcional para o conjunto de instruções AArch64 do ARMv8, um processador presente nos smartphones de última geração.

"ARM, originalmente Acorn RISC Machine, e depois Advanced RISC Machine, é uma família de arquiteturas RISC desenvolvida pela empresa britânica ARM Holdings. Tais arquiteturas são licenciadas pela ARM para outras empresas, que implementam-nas em seus próprios produtos. A ARM também desenvolve chips que utilizam tal arquitetura e que são licenciados para uso exclusivo de outras empresas em seus produtos[2]". Essa arquitetura de referência é a mais utilizada nos processadores de dispositivos móveis ou sistemas embarcados na atualidade.

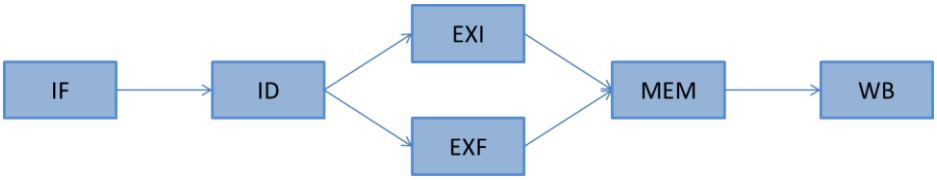
Os processadores ARM, dependendo da versão, podem implementar um ou mais desses quarto conjuntos de instruções: i) Thumb (16 bits); ii) Thumb2 ou T32 (extensão de Thumb adicionando instruções de 32 bits); iii) 32bit ARM ou AArch32; iv) 64bit ARM ou AArch64, utilizado neste trabalho.

O simulador Arqui64 a ser implementado cobre apenas parte do conjunto de instruções AArch64. Além disso, o caminho de dados a ser implementado será simplificado, usando como base os estágios do MIPS, já conhecido da disciplina, e não do ARM, visto que o ARM é muito mais complexo.

O documento de referência para o trabalho é o "ARM® Architecture Reference Manual – ARMv8, for ARMv8-A architecture profile", Partes B e C, ou seja, apenas o nível de aplicação (e não o de sistema) será de interesse. O documento de referência encontra-se no Tópico "Preparação para o Trabalho Final - Vale 5 pontos!" deste curso.

[1] Cultura inútil: existem três tipos de córtex cerebral: o paleocórtex, o arquicórtex e o neocórtex. O arquicórtex ocorre no hipocampo, estrutura cerebral do sistema límbico em partes responsável pelo armazenamento de informações na memória. O nome Arqui64 é então uma referência a três coisas: i) à disciplina de Arquitetura de Computadores I, ii) ao arquicórtex do cérebro humano, devido à sua relação com a memória e iii) à série de processadores ARM Cortex-A, presentes nos smartphones de última geração.
[2] https://pt.wikipedia.org/wiki/Arquitetura_ARM

O Arqui64 possui os mesmos registradores de uso geral do ARMv8 e um subconjunto do extenso conjunto de instruções do ARMv8. Cada instrução executada no Arqui64 demora apenas um ciclo de clock para completar, ou seja, trata-se de uma implementação Monociclo. O processador possui seis unidades funcionais denominadas: IF, ID, EXI, EXF, MEM e WB. Um ciclo de máquina utiliza 5 unidades. Se a instrução não faz operação de ponto flutuante, o ciclo é IF-ID-EXI-MEM-WB, caso contrário o ciclo é IF-ID-EXF-MEM-WB, como ilustrado na figura a seguir.



Essas unidades funcionais são descritas por:

- IF (instruction fetch): a próxima instrução a ser executada é buscada na memória e armazenada no registrador de instruções (IR);
- ID (instruction decode): a instrução sendo executada é decodificada e os operandos são buscados no banco de registradores;
- EXI (execute integer): a instrução inteira é executada e as condições dos resultados são "setados" (condition codes) para operações de ALU. Este estágio só é executado para instruções que **não exigem** operações com ponto flutuante;
- EXF (execute float): a instrução float é executada e as condições dos resultados são "setados" (condition codes) para operações de ALU. Este estágio só é executado para instruções que **exigem** operações com ponto flutuante;
- MEM (memory): loads, stores, o cálculo do endereço efetivo e a resolução de branches são feitos nesse ciclo;
- WB (write back): os resultados são escritos no banco de registradores.

Podem ser consultadas as implementações realizadas pelos alunos no semestre 2018/1, para adiantar os trabalhos. Ajudará bastante, mas as principais diferenças do trabalho de 2018/1 para este trabalho são:

1. Mudança de processador 16 bits para 64 bits,
2. Mudança do conjunto de instruções Thumb 16 para o AArch64 e
3. Alteração no ciclo da máquina, acrescentando a possibilidade de se executar operações de ponto flutuante.

4. Implementação do Simulador Arqui64

O processador tARM deverá possuir as seguintes características:

- Enderaçamento de 64 bits, palavra de 64 bits e instruções de 32 bits;
- registradores de 64 bits do ARMv8 (Seção B1.2.1);
- memória de dados e instruções (arquitetura Von Neumann, diferente do ARMv8) endereçada a nível de palavra, ou seja, cada endereço de memória deve se referir a oito bytes. No total, o processador deverá possuir 64k endereços. Então, a memória total do processador será de 512KB (64k endereços x 8 bytes).

Um dos requisitos fundamentais para se entender como implementar o simulador funcional do processador Arqui64 é a compreensão do termo "funcional''. Uma descrição funcional divide o processador nos blocos que existirão em uma implementação real. Portanto, o processador deverá conter quatro rotinas principais, uma para cada unidade funcional. Note que a única forma de comunicação entre essas quatro rotinas em um processador real é via registradores, que no caso do simulador serão implementados por estruturas de dados no programa do simulador.

Além dessas quatro rotinas, os elementos principais do processador real deverão estar encapsulados por módulos, tais como os elementos, banco de registradores, ALUs, incrementadores e memória.

5. CONJUNTO DE INSTRUÇÕES

O conjunto de instruções a ser implementado é um SUBCONJUNTO do conjunto de instruções AArch64 especificado no documento de referência.

O SUBCONJUNTO de instruções a ser implementado é composto por TODAS, mas somente essas, instruções que aparecem no programa "summation.S", **que será disponibilizada na versão 2 deste enunciado**.

6. ARQUIVO DE SAÍDA

O simulador deve executar cada instrução do programa de entrada até que ele se encerre, e deverá gerar um arquivo texto de saída contendo todos os acessos feitos à memória. Cada acesso à memória deverá ser informado em uma linha do arquivo de saída, usando um dos três formatos a seguir:

- ri 0034, que significa que foi lida uma instrução (estágio IF) do endereço 0x0034,
- rd 00A0, que significa que foi lido um dado (estágio MEM) do endereço 0x00A0, ou
- wd 009E, que significa que foi escrito um dado no endereço 0x009E.

7. TESTES

Utilize os arquivos e as instruções **que estarão disponíveis na pasta "Arquivos auxiliaries"** para testar o seu programa, usando "summation.o" como arquivo binário de entrada.

O teste final será feito pelo professor com o uso de uma variação do programa summation.S, utilizando o mesmo conjunto de instruções usado em summation.S, e será verificado o arquivo de saída.

8. AVALIAÇÃO

O trabalho será avaliado presencialmente, nas aulas de 17/06/2019 a 02/07/2019, e a versão final do trabalho, com todo o código fonte, deverá ser submetida no Campus Virtual até 04/07/2018.

Será dada nota parcial (de até 70% da nota total) para trabalhos não concluídos, mas será dada nota 0 para trabalhos não submetidos no Campus Virtual, mesmo que previamente apresentados ao professor.