Arquitetura de Computadores II

Arm 9

Rodrigo Padilha
Thaiane Lopes

26 de Novembro de 2007

Introdução

Este trabalho pretende apresentar instruções de máquinas, utilizando a arquitetura Arm9. São apresentadas algumas instruções da linguagem de montagem do processador e algumas representações binárias. O Arm9 é uma arquitetura RISC, assim as instruções de cada tipo possuem um formato relativamente homogêneo, facilitando a apresentação de sua codificação binária.

Foi realizado um estudo da arquitetura ARM9, visto que tínhamos uma pequena base de ARCHC. A arquitetura ARCHC, propõe uma linguagem para a descrição de arquiteturas de processadores baseadas na linguagem SystemC. A partir do arquivo r3000, foi identificando os passos para o desenvolvimento deste trabalho. Somente assim identificamos no Arm9, os tipos de formatos, nomes das instruções, tamanho, bem como a descrição de cada instrução e a informação requerida para decodificá-la.
Para realizar a descrição deste trabalho, começaremos pelo arquivo arm9_isa.ac, que foi desenvolvido através da semelhança com o `r3000_isa.ac`.

ARM9_ISA.AC

Definiu-se os formatos das instruções, sendo elas: aritméticas, instruções de desvio condicional, loop e instruções para chamadas a procedimentos.
Para o tipo DATA PROCESSING (Type_DPI)**,** que inclui todas as instruções aritméticas, como add e sub, foi utilizado o formato abaixo:
ac_format Type_DPI = "%cond:4 %nus1:2 %if:1 %op:4 %sf:1 %rs:4 %rd:4 [%shift_op:4 %imm:8 | %rt:4]" ;
Este formato é designado às instruções add, sub, and, orr e eor que foram implementadas neste trabalho.

A seguir segue os formatos das demais instruções:

BRANCH INSTRUCTIONS - Type_BI

ac_format Type_BI = "%cond:4 %nus1:3 %lr:1 %offs1:24";

SINGLE DATA TRANSFER – Type_SDTI

ac_format Type SDTI = "%cond:4 %nus1:2 %if:1 %ppi:1 %udf:1 %bwf:1 %wbf:1 %lsf:1 %rn:4 %rd:4 %offs1:12";

NORMAL MULTIPLY - Type_NMI

ac_format Type_NMI = "%cond:4 %nus1:7 %af:1 %sf:1 %rd:4 %rs:4 %nus2:4 %rm:4";

LONG MULTIPLY - Type_LMI

ac_format Type_LMI = "%cond:4 %nus1:5 %uf:1 %af:1 %sf:1 %rdhi:4 %rdlo:4 %rs:4 %nus2:4 %rm:4";

As instruções foram relacionadas aos seus tipos de formatos. Cada uma atendendo as requisições que são relatadas como vimos anteriormente na especificação dos formatos.

Abaixo seguem as instruções relacionadas aos seus tipos:

DATA PROCESSING
ac_instr<Type_DPI> AND, EOR, SUB, RSB, ADD, ADC, SBC, RSC, TST, TEQ, CMP, CMN, ORR, MOV, BIC, MVN;

BRANCH INSTRUCTIONS
ac_instr<Type_BI> B, BLX, BX;

SINGLE DATA TRANSFER
ac_instr<Type_SDTI> LDR, STR;

NORMAL MULTIPLY
ac_instr<Type_NMI> MUL, MLA;

LONG MULTIPLY
ac_instr<Type_LMI> SMLAL, SMULL, UMLAL, UMLL;

Após relacionarmos as instruções aos seus tipos de formatos, foi designada a declaração das instruções.

DATA PROCESSING INSTRUCTIONS
ac_format Type_DPI = "%cond:4 %nus1:2 %if:1 %op:4 %sf:1 %rn:4 %rd:4 %shif_op:12";

BRANCH INSTRUCTIONS
ac_format Type_BI = "%cond:4 %nus1:3 %lr:1 %offs1:24";

SINGLE DATA TRANSFER (LOAD AND STORE) INSTRUCTIONS
ac_format Type_SDTI = "%cond:4 %nus1:2 %if:1 %ppi:1 %udf:1 %bwf:1 %wbf:1 %lsf:1 %rn:4 %rd:4 %offs1:12";

NORMAL MULTIPLY INSTRUCTIONS

ac_format Type_NMI = "%cond:4 %nus1:7 %af:1 %sf:1 %rd:4 %rs:4 %nus2:4 %rm:4";

LONG MULTIPLY INSTRUCTIONS

ac_format Type_LMI = "%cond:4 %nus1:5 %uf:1 %af:1 %sf:1 %rdhi:4 %rdlo:4 %rs:4 %nus2:4 %rm:4";

Em seguida, realizou-se as declarações das instruções com os formatos identificados anteriormente. Visto que neste trabalho, implementou-se somente as Instruções DATA PROCESSING, Type_DPI. As outras instruções foram declaradas, mas não sendo implementadas no outro arquivo, arm9_isa.cpp.
Na seqüência, especificou-se a descrição de cada instrução e a informação requerida para a decodificação. Nas instruções ADD, SUB, AND, ORR, BIC e EOR, foram calculados os opcodes, determinando e identificando cada instrução.
Exemplo:
Instrução ADD
ADD.set_asm("add %cond %rd %rs %rt");
ADD.set_decoder(**opcode=**0x04, func=0x00);

ARM9.AC

Neste arquivo, implementaram-se os formatos dos tipos de registradores. Realizou-se uma pesquisa/estudo, para identificar os formatos corretos para cada registrador de pipeline. Declarados os tipos de formatos, pôde-se definir valores aos registradores, afim de obter um endereçamento correto, para a ação do PC.

```
ac_format F_ID_EX = "%npc:32 %data1:32 %data2:32 %imm:32:s %rs:4 %rd:4
%rt: 4 %regwrite:1 %memread:1 %memwrite:1";
```

As instruções de acesso à memória no ARM são primordialmente a carga de um registrador em uma posição de memória e o armazenamento do conteúdo do registrador em uma posição da memória, conforme dita a filosofia RISC.

AMR9_ISA.CPP

Por fim, este arquivo destina-se a implementação das instruções.
Todas as instruções possuem códigos destinados a executar um procedimento quando são “chamadas”. Para cada registrador, existe um caso a ser seguido, ou seja uma execução será tomada dependo do tipo de registrador.

```
void ac_behavior( ORR ){
      .
      .
      .
case EX:
        printf("%d"EX_MEM.alures);
        if(EX_MEM.alures ==0)
                EX_MEM.alures = ex_value1 | ex_value2;

        EX_MEM.alures = ex_value1 + imm;
```

No código acima, é chamada a instrução, verificando os estágios que possui, delimitando qual regra será adotada, levando em consideração a instrução escolhida. No caso, ORR, será executada a instrução através do algoritmo, a fim de realizar tarefas do tipo DPI.

Dificuldades

Ocorreram dificuldades na implementação deste trabalho. O fator de maior impacto foi a inexperiência com a linguagem, desde seus conceitos básicos, que deveriam ser adquiridos em Arquitetura de Computadores I. Houve também a incompatibilidade dos horários disponíveis entre os integrantes do grupo, ressaltando entre eles, os horários para tirar dúvidas com o professor. Neste caso, resolve-se realizar testes domésticos, o que ocasionou em nenhum resultado, devido a falta de conhecimento para a instalação da ferramenta que nos auxiliaria.

Dentre tantas dificuldades, optou-se por realizar grande parte das implementações, em horários de aula de outras disciplinas, tentando ter um resultado mínimo para cada checkpoint realizado.

Conclusão

Com o foco na simulação de cada instrução, permitiu-se a obtenção de uma visão holística da arquitetura dos computadores. Preocupou-se com cada valor agregado aos componentes (registradores, memórias, *opcodes*, instruções, tipos e formatos) que auxiliam na execução de cada instrução.

A Arquitetura ARM (ARM9) pode ser implementada em vários processadores de 32 bits, devido à sua versatilidade de uso em diversas aplicações práticas, sendo pela boa performance apresentada ou pela constante atualização, originando novas versões.

Referências Bibliográficas

ARM Architecture Reference Manual - http://www.gpec.ucdb.br/ricrs/
http://dqsoft.blogspot.com/search/label/ARM