UNIVERSIDADE FEDERAL DE OURO PRETO

Lista: RISC-V – 24 de janeiro de 2017

Arquitetura de Computadores - BCC 236

Bernardo Marotta de Rezende – 15.1.4292

Questão 1.

ISA RISC (Reduced Instructions Set Computer) são conjuntos de instruções que são atualmente usadas nos dispositivos computacionais mais modernos como videogames, projetos embarcados de pequeno porte até servidores ou celulares smartfones.

Questão 2.

O projeto RISC-V é um conjunto de instruções moderno e de alta qualidade, que foi desenvolvido por profissionais da área de Arquitetura de Computadores, na Universidade de Berkeley, CA. A meta do RISC-V é criar um conjunto de instruções universal open-source, que pode ser utilizado tanto de forma acadêmica quanto comercial.

Questão 3.

Não, seu propósito de mercado é amplo, seu conjunto de instruções provém tudo que é necessário para suportar qualquer projeto comercial. Outro motivo que o coloca no mercado é o fato de ser aberto, podendo assim ser usado em fabricações e vendas.

Questão 4.

Multiplicação M: Adiciona 4 instruções de multiplicação, duas de divisão, e duas de manipulação de restos.

Sincronização A: Adiciona 11 instruções de sincronização visando consistência e atomicidade da operação.

Ponto flutuante: Possui extensão de precisão simples F, dupla D e quádrupla Q.

Compressão de tamanho de código C: Não adiciona novas funções, mas codifica as instruções para salvar espaço.

Questão 5.

Sim, o RISC-V possui suporte para extensões criadas.

Questão 6.

O RISC-V possui 32 registradores inteiros, e 32 registradores opcionais para float, que são incluidos juntos com a extensão de ponto flutuante.
Também existe uma variante de pequeno porte do RISC-V com só 16 registradores inteiros.

Questão 7.

O conjunto de instruções RISC-V foi desenvolvido com foco na praticidade das implementações, com características que aumentam a velocidade computacional e reduz seu custo de energia. Duas otimizações significativas foram colocar os bits mais significativos em uma posição fixa e uma disposição que reduz o número de multiplexadores no chip.

Questão 8.

Como todos os designs de RISC's, RISC-V é uma máquina de load-store. Somente estas duas instruções acessam a memória principal e todas as operações lógico-aritméticas ocorrem entre registradores.

Questão 9.

A extensão C tem a função de salvar espaço e com isso reduzir o tamanho do footprint. O propósito de reduzir o tamanho do código binário, energia e custo para pequenos computadores, é visando sistemas embarcados. Pesquisas com C mostram que um código pode ficar 20% menor que um x86 em MIPS Comprimido e 2% maior que um ARM Thumb-2.

Questão 10.

As variantes de RISC-V são baseadas em RV32I. A variante RV32E é usada em implementações de pequeno porte, e as variações RV64I e RV128I são apenas extensões do tamanho de bits para 64 e 128 respectivamente.

Questão 11.

Não, o RISC-V intencionalmente não possui estruturas condicionais, a fim de simplificar o desenvolvimento da CPU, minimizando interações entre as instruções. Foram construídos operandos de comparação dentro dos jumps condicionais para contornar isso.

Questão 12.

Os registradores são colocados em uma mesma disposição com o objetivo de

facilitar na decodificação.

Questão 13.

Jump and link é uma função jump com endereço de retorno.

Questão 14.

Rocket-Chip é uma ferramenta capaz de gerar diferentes configurações de RISC-V para diferentes ambientes SoC (System on Chip). Possui 5 estados.

Questão 15.

Utiliza-se um emulador C++ de RTL (Register transfer level).
Entre os vários diretórios de pastas do projeto, o emulador C++ RTL é executado do diretório /emulator. Um comando de execução default já é conhecido e possui a seguinte sintaxe

```
make run-asm-tests.
```

Com este comando, o emulador realizará emulações sobre o assembly realizando testes assim como é possível realizar em benchmarks usando o comando

```
make run-bmark-tests.
```

Desta forma, será executado uma emulação de um projeto que esteja em suas configurações padrões sobre o comando escolhido.

Podemos também construir itens com configurações de pequeno porte utilizando

```
make CONFIG=ExampleSmallConfig.
```

E para testar, utiliza-se

```
make CONFIG=ExampleSmallConfig run-asm-tests.
```