Natália Faria Melo
15.1.4037

Lista de Arquitetura de Computadores

1) Várias universidades e organizações comerciais já planejam usar o projeto em seus produtos e suportar a ISA RISC, algumas inclusive já os utilizam em seus projetos, como a Google, Mellanox e Oracle. Um dos fundadores do Adapteva planeja usar ISA RISC como sucessor para o seu produto acelerador multicore e Nvidia planeja usá-lo para substituir a linha de processadores Falcon em suas placas GeForce. Já é possível verficar vários projetos funcionais online usufruindo da permição de licensa BSD.
2) RISC-V é um ISA open-source desenvolvido por várias pessoas ligadas a área de Arquitetura de Computadores com experiência em diversos ramos. Formou-se então o RISC-V, um Conjunto de instruções para Computador de Propósito Geral. Ele foi produzido na Universidade da Califórnia e se tornou um conjunto limpo e modular com inteiros base em 32, 64 e 128 bits, com varias extensoes adicionais.
3) O RISC-V se iniciou como projeto academico em 2010, sendo disponibilizado em 2015. Atualmente já é utilizado em industrias como Google, Mellanox e Oracle. Um exemplo é o exemplo é o escalar de cinco estágios chamado RISC-V Rocket em Scala, utilizado para sintetização de chips.
4) O RISC-V possui inúmeras extensões, dentre elas, temos: multiplicação, que adiciona quatro instruções de multiplicação, duas de divisão e duas de manipulação de restos; sincronização A, que adiciona onze instruções de sincronização, visando conscistência e atomacidade das operações. São divididos em dois, grupos, Load Reserved e Store para operações atômicas na memória; ponto flutuante, que pode ser de precisão simples (F), precisão dupla (D) e quádrupla precisão (G) – os de maior precisão tendo os de menor como pré-requisito –, introduz 32 registradores de ponto flutuante de tamanho de 32 bits, inclui um registrador de 5 bits para exceções (para que não gere interrupção). As instruções load/store usam o endereçamento base+offset; compressão de tamanho de código (C), codifica instruções inteiras para salvar espaço e reduzir o footprint. Para tal, criam-se instruções compressadas em 16 bits. É disponível tanto para bases inteiras como para pontos flutuantes. Foi feito pensando em sistemas embarcados, e se mostrou 20% menor que um codigo x86 e MIPS comprimido e 2% maior queum ARM Thumb-2.
5) Porções do espaço de codificação de instruções já foram reservadas para uso futuro, portanto, extensões podem ser adicionadas e utilizadas.
6) O RISC-V possui 32 registradores inteiros e 32 opcionais para ponto flutuante (que são incluídos junto a extensão de ponto flutuante). Existe também uma versão de pequeno porte com apenas 16 registradores.
7) Várias otimizações foram feitas no projeto do RISC-V, entre elas: colocar os bits mais significantes numa posição fixa e uma disposição de bits feita para reduzir o número de multiplexadores no chipem sua implementação.
8) O RISC-V é uma máquina load-store, o que significa que apenas essas duas instruções possuem acesso à memória principal.
9) Ela é utilizada para reduzir o tamanho do código binário, reduzir gasto de energia e custo para computadores pequenos, visando sistemas embarcados.
10) Tanto o RV32I quanto o RV64I são variantes do RISC. Já a Extensão C é uma extensão adicional complementar ao RISC.
    A extensão C e o RV32I possuem objetivos similares: permitir que o RISC seja utilizado em aparelhos embarcados e de baixa potencia. A extensão reduz o tamanho do código

gerado, comprimindo instruções. O RV32I, por ser uma variante, também permite redução no tamanho final do chip, viabilizando ainda mais as utilizações. O RV64I é apenas uma variação do RISC original com suporte para 64 bits. Ele também introduz instruções novas para operar com dados menores que 32bits.

11) O RISC-V não possui o condicional if. Para equivalencia foram utilizados comparações nos jumps condicionais. Isso foi feito para simplificar o desenvolvimento da CPU, minimizando inteterações entre instruções.
12) Os registradores se mantem todos na mesma posição mesmo em instruções diferente para facilitar a decodificação, não necessitando criar formatos diferentes para diferentes instruções.
13) A instrução JUMP executa um salto, saindo da instrução atual e indo para a indicada. Já a instrução JUMP AND LINK, além de executar esse salto, armazena em um registrador específico o endereço da instrução que chamou o jump, permitindo retorno após a execução para que continue normalmente a leitura e execução das instruções.
14) Rocket-Chip é um escalar de cinco estágios utilizado para sintetização de chips. Ele permite gerar diferentes configurações de SoC, que são especificadas por meio de parametros
15) O Rocket-chip tem diferentes modulos, cada um é responsável por uma função. As configurações de chip são atingidas manipulando e alterando esses modulos, de forma a alcançar diferentes objetivos. Os rockets são gerados junto de suas chaches que interarem com itens externos e compartilhados. Após, é possível gerar um programa em C++ onde itens podem ser feitos com base nas configurações previamente definidas no rocket. Com base na configuração do rocket e a junção feita com o software gerado, é possível então, finalmente, sintetizar a placa.