UENF

Universidade Estadual do Norte Fluminense Darcy Ribeiro

Curso: Ciência de Computação **Data:** 27/06/2023

Lista: 3 **Período:** 4°

Disciplina: Arquitetura de Computadores

Professor: Sanya Carvalho **Turno:** Diurno

Nome: Gabriel Costa Fassarella **Matrícula:** 20211100046

Lista 3

1) E
2) iLOAD N
   iLOAD C
   IF_ICMPEQ L1 (if C == N)
   INVOKERVIRTUAL soma
   IADD
   GOTO L2
   L1: IRETURN 0
3) Sim, pois a implementação realiza uma chamada recursiva, até que o contador seja igual ao número vezes da multiplicação.
4) O microprocessador MIC não é projetado para executar o algoritmo de multiplicação através de somas e deslocamentos diretamente. Seriam necessárias modificações significativas no hardware e no conjunto de instruções para adicionar suporte a esse algoritmo. No entanto, essa abordagem seria ineficiente e pouco prática em comparação com técnicas de multiplicação mais avançadas e eficientes disponíveis atualmente. A maioria dos microprocessadores modernos utiliza técnicas de multiplicação otimizadas, como algoritmos de multiplicação rápida, que aproveitam propriedades binárias e hardware dedicado para realizar operações de multiplicação de forma mais eficiente.
5) A metodologia para criação de projetos no nível ISA depende de inúmeros fatores, visto que cada máquina é diferente e é um caso especial, logo é necessário avaliar os caminhos de dados e a microarquitetura da máquina, assim como os seus requisitos, e logo após definir as instruções seguindo um critério para tal. Para isso, deve-se levar em conta fatores como: velocidade, custo, confiabilidade, facilidade de utilização, requisitos de energia e tamanho físico.
6) Em primeiro lugar: uma boa ISA deve definir um conjunto de instruções que pode ser implementado com eficiência em tecnologias atuais e futuras, resultando em projetos de efetivos em custo por várias gerações. Em segundo lugar: deve fornecer um alvo claro para o código compilado.
7) O compilador desempenha um papel importante na arquitetura de computadores ao traduzir o código de uma linguagem de alto nível para o nível ISA. Ele realiza análises léxicas e sintáticas, verifica a semântica do código, otimiza o programa e gera o código de máquina correspondente ao ISA do processador. Essa tradução permite que programas escritos em linguagens de alto nível sejam executados no hardware do processador de forma eficiente e correta.
8) Modos do nível ISA: Modo núcleo (kernel): deve executar programas de aplicação e não permite que certas instruções sejam executadas.

Modo usuário: deve executar programas de aplicação e não permite que certas instruções sensíveis, como as que manipulam a cache diretamente, sejam executadas.

9) E

10) Os registradores no nível ISA (Instruction Set Architecture) podem ser classificados em três categorias principais: registradores de uso geral, registradores de ponto flutuante e registradores de controle e estado. Os registradores de uso geral são usados para armazenar dados temporários, os registradores de ponto flutuante são dedicados a operações de números reais e os registradores de controle e estado gerenciam o fluxo de execução e o estado do processador. A quantidade e a organização dos registradores podem variar de acordo com a arquitetura específica do processador.

11) O nível ISA (Instruction Set Architecture) reconhece diferentes tipos de dados, incluindo inteiros, pontos flutuantes, caracteres e booleanos. Os inteiros são representados por bits e podem ter diferentes tamanhos, enquanto os números de ponto flutuante representam números reais com casas decimais. Os caracteres são usados para representar caracteres individuais, e os booleanos representam valores lógicos verdadeiros ou falsos. Além desses tipos básicos, algumas arquiteturas de ISA podem incluir tipos de dados adicionais ou especializados para fins específicos. A disponibilidade e os formatos dos tipos de dados podem variar entre as arquiteturas e implementações do ISA.

12) O formato das instruções em uma arquitetura de computadores é determinado considerando critérios como tipos de operações suportadas, codificação de operandos, tamanho das instruções, eficiência e desempenho, e complexidade versus simplicidade. Esses critérios visam garantir o suporte adequado às operações, a eficiência na execução das instruções e a facilidade de implementação do hardware e desenvolvimento de compiladores.

13) Os formatos de instrução mais comuns em arquiteturas de processadores são:

1. Formato de instrução de um único operando: Possui um campo para o operando e o código da operação. Exemplo: ADD R1, R2.
2. Formato de instrução de dois operandos: Contém dois campos para os operandos de origem e destino. Exemplo: MOV R1, R2.
3. Formato de instrução de três operandos: Possui três campos para os operandos de origem, destino e intermediário. Exemplo: ADD R1, R2, R3.
4. Formato de instrução de transferência de dados: Usado para transferir dados entre registradores e memória, com campos para os endereços de origem e/ou destino. Exemplo: LOAD R1, [R2].

Esses formatos podem variar entre as arquiteturas de processadores, mas esses exemplos são comumente encontrados.

14) Instruções de tamanho fixo têm a vantagem da simplicidade, acesso rápido à memória e alinhamento eficiente. No entanto, podem usar espaço de forma ineficiente e limitar a complexidade das instruções. Já as instruções de tamanho variável oferecem uso eficiente de espaço, flexibilidade e redução do tráfego de memória, mas têm decodificação complexa e acesso mais lento à memória. A escolha depende das necessidades da arquitetura específica.

15) O formato das instruções em uma arquitetura de computadores é determinado considerando critérios como tipos de operações suportadas, codificação de operandos, tamanho das instruções, eficiência e desempenho, e complexidade versus simplicidade. Esses critérios visam garantir o suporte adequado às operações, a eficiência na execução

das instruções e a facilidade de implementação do hardware e desenvolvimento de compiladores.

16) Existem diferentes modos de endereçamento usados em arquiteturas de processadores. Alguns exemplos incluem endereçamento imediato, direto, indireto, base-deslocamento, por registrador e relativo. Cada modo de endereçamento define como os operandos de uma instrução são especificados e acessados. A escolha do modo de endereçamento depende das necessidades da arquitetura e dos requisitos da linguagem de programação.

17) Endereçamento Imediato: O valor do operando é definido diretamente na instrução.
Endereçamento Direto: O endereço do operando é fornecido diretamente na instrução.
Endereçamento de Registrador: O operando é acessado por meio de um registrador. Conceitualmente semelhante ao endereçamento direto.
Endereçamento indireto de registrador: Endereço do dado é obtido do conteúdo da posição identificada pela instrução. Referência memória sem precisar de endereço de memória completo na instrução.
Endereçamento indexado: Endereçamento indexado é o nome que se dá ao endereçamento de memória que fornece um registrador (explícito ou implícito) mais um deslocamento constante.
Endereçamento de base indexado: Endereço do dado é obtido pela soma dos valores de dois registradores e opcionalmente uma constante.

18) C

19) E

20) B

21) a - ((A - B) + C) * D
b - (A/B) + (C/D)

22) A

23) Os quatro tipos de cópia de dados possíveis na arquitetura de computadores são:

1. Load: Transferência de dados da memória para o registrador.
2. Store: Transferência de dados do registrador para a memória.
3. Move: Transferência direta de dados entre registradores.
4. Copy: Transferência direta de dados entre posições de memória diferentes.

24) LRU: O valor que teve a sua utilização mais antiga é retirado.
FIFO: Segue a ideia da fila, o primeiro valor a entrar é o primeiro a sair.