Lógica digital e organização de computadores

Arquiteturas CISC

TADS
Tecnologia em Análise e Desenvolvimento de Sistemas

Campus de Alegre

O que é a arquitetura CISC?

- Definição: CISC = Complex Instruction Set Computer (Computador com Conjunto Complexo de Instruções).
- Contexto Histórico: Nasceu da necessidade de que o código de máquina refletisse diretamente as construções de linguagens de alto nível (Linguagens como COBOL, FORTRAN, C).
- Característica Central: O conjunto de instruções é extenso e variado, incluindo instruções de alta complexidade.

O que é a arquitetura CISC?

- Filosofia: Criar instruções poderosas (complexas) que fazem muito trabalho em uma única etapa (ex: uma única instrução lê dados da memória, realiza uma operação e armazena o resultado).
- Objetivo: Reduzir o tamanho dos programas (densidade de código) e simplificar o trabalho do compilador.
- Implementação: A complexidade não está no hardware da UAL, mas sim na Unidade de Controle (UC), que geralmente utiliza microcódigo (pequenos programas internos) para executar cada instrução complexa.

Dispositivos que utilizam

Características CISC

Conjunto Extenso de Instruções

- Variação: Possui centenas de instruções (podendo chegar a mais de 300), muitas das quais são usadas raramente.
- Instruções Complexas: Inclui instruções que realizam múltiplas microoperações em um único Código de Operação (C.Op.).
- Exemplo Histórico: Instruções para manipulação de strings inteiras ou para inicialização de vetores.
- Vantagem: O código de máquina gerado é muito denso (ocupa menos memória).

Instruções de Tamanho Variável

- Organização: As instruções não possuem um tamanho fixo (podem variar de 1 byte até 15 bytes ou mais).
- Dependência: O tamanho da instrução é determinado pelo C.Op. e pelo número/tipo de operandos.
- Exemplo: Uma instrução simples de soma pode ser curta, enquanto uma instrução que acessa memória com múltiplos modos de endereçamento pode ser longa.
- Desafio: A UCP precisa decodificar o C.Op. primeiro para saber quantos bytes da memória pertencem à instrução atual.

Operações Diretas na Memória

- Design: Muitas instruções podem operar diretamente em dados armazenados na Memória Principal (MP), sem exigir que os dados sejam carregados previamente nos registradores.
- Exemplo: ADD [Endereço A], [Endereço B] (Soma o conteúdo de B e armazena o resultado em A).
- Uso de Registradores: O número de registradores de uso geral tende a ser pequeno em comparação com RISC (ex: 8-16 registradores).
- Consequência: A latência de acesso à MP se torna um gargalo de desempenho.

Microcódigo e Unidade de Controle

- Microcódigo: Sequências de microinstruções (simples e atômicas) armazenadas em uma memória ROM/RAM interna (Memória de Controle).
- Função da UC: A Unidade de Controle traduz cada instrução CISC complexa em uma rotina de microcódigo.

Microcódigo e Unidade de Controle

- Vantagem: Permite que o hardware principal seja projetado com pouca lógica, delegando a complexidade para o microcódigo (que é mais fácil de modificar).
- Desvantagem: Adiciona um estágio de tradução/consulta (latency) na execução da instrução.

Modos de Endereçamento Complexos

- Variedade: CISC possui uma grande variedade de modos de endereçamento (mais de 10 tipos).
- Complexidade: Inclui modos que combinam múltiplos cálculos em um único estágio de busca de operando (ex: base + índice + deslocamento).
- Benefício: Um compilador pode usar um único C.Op. para traduzir uma estrutura complexa de linguagem de alto nível diretamente (ex: A[i] = B[j] + C), resultando em código compacto.

Impacto no Desempenho: Pipelining

- Dificuldade: A execução sobreposta (pipelining) é inerentemente difícil em CISC devido a:
- Tamanho Variável: A UCP não sabe onde a próxima instrução começa.
- Acesso à Memória: Instruções longas de acesso à MP bloqueiam o pipeline.
- Solução Moderna: Processadores x86 modernos traduzem as instruções CISC em micro-operações internas (μops) de tamanho fixo, que são então executadas em um pipeline interno, de estilo RISC.

Legado histórico

- Padrão de Mercado: A arquitetura CISC x86 (8086, 80386, Pentium, Athlon) dominou o mercado de computadores pessoais (PCs) por décadas.
- Compatibilidade: A principal razão para a sobrevivência do CISC x86 é a retrocompatibilidade com software e sistemas operacionais mais antigos.
- Evolução Híbrida: Atualmente, o CISC x86 é uma arquitetura híbrida – a interface para o programador é CISC, mas o core de execução é de fato RISC.

Vantagens Históricas do CISC

- Densidade de Código: Programas ocupam menos memória principal (MP) e menos espaço em disco. Isso era crucial quando a memória e o armazenamento eram caros.
- Compiladores Simples: A complexidade da tradução de linguagem de alto nível era delegada ao fabricante do hardware (microcódigo), simplificando a escrita de compiladores.
- Transição de Código: Facilidade em migrar software antigo para novos processadores.

Desvantagens Intrínsecas do CISC

- Tempo de Ciclo: O tempo para decodificar e executar uma instrução complexa pode levar muitos ciclos de clock.
- Complexidade de Design: O microcódigo e a lógica de decodificação complexa exigem mais transistores e tornam o chip mais difícil e caro de projetar e testar.
- Otimização: A dificuldade de implementar pipelining impede a execução de 1 instrução por ciclo.

O Fim da CISC Pura

- Motivação para Mudar: A latência da memória (MP) e a dificuldade de pipelining forçaram os fabricantes CISC (como a Intel) a mudar.
- O Híbrido: Processadores x86 modernos traduzem as instruções CISC em µops (micro-operações) em tempo de execução. Essas µops, de tamanho fixo e simples, são então executadas internamente em um core de estilo RISC.
- Legado: A complexidade do design foi movida para uma camada de tradução (decoder).