Organização de Computadores

Prof. Robson de Souza

Processadores

Como já foi explicado anteriormente, a função de uma CPU é executar programas armazenados na memória principal buscando suas instruções, examinando-as e então executando-as uma após a outra. Também sabemos que a CPU é composta por várias partes distintas. A unidade de controle é responsável por buscar instruções na memória principal e determinar seu tipo, a unidade de aritmética e lógica (ALU ou ULA) efetua operações como adição e AND (E) booleano para executar as instruções e, por fim, existem alguns registradores (memória de alta velocidade interna a CPU) importantes, sendo eles o Contador de Programa (PC) e o Registrador de Instrução (IR), que mantém a instrução que está sendo executada no momento em questão.

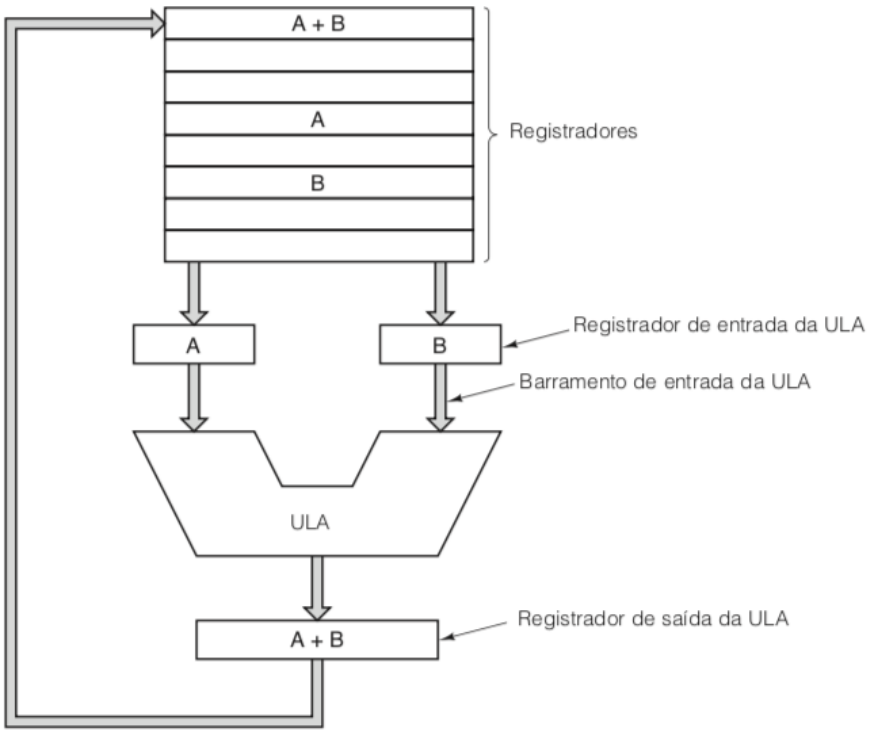
Tendo esses conhecimentos em mente, é possível começar a entender como funciona a organização de uma CPU.

Organização da CPU

A organização interna de parte de uma típica CPU de von Neumann é mostrada na Figura abaixo com mais detalhes. Essa parte é denominada **caminho de dados** e é composta por registradores (em geral 1 a 32), da ULA (unidade lógica e aritmética) e por diversos barramentos que conectam as partes.

Os registradores alimentam dois registradores de entrada da ULA, representados por A e B na figura. Eles contêm a entrada da ULA enquanto ela está executando alguma operação de computação.

O caminho de dados de uma típica máquina de von Neumann.



Fonte: (Tanenbaum e Austin, 2013)

A ULA efetua adição, subtração e outras operações simples sobre suas entradas, produzindo assim um resultado no registrador de saída, o qual pode ser armazenado em um registrador. Mais tarde, ele pode ser escrito (isto é, armazenado) na memória, se desejado.

Grande parte das instruções pode ser dividida em uma de duas categorias: registrador-memória ou registrador-registrador.

Registrador-memória → Permitem que palavras de memória sejam buscadas em registradores, onde podem ser usadas como entradas de ULA em instruções subsequentes. (“Palavras” são as unidades de dados movimentadas entre memória e registradores. Uma palavra pode ser um número inteiro por exemplo) Outras instruções registrador-memória permitem que registradores voltem à memória para armazenagem.

Registrador-registrador → Busca dois operandos nos registradores, traz os dois até os registradores de entrada da ULA, efetua alguma operação com eles (por exemplo, adição ou AND booleano) e armazena o resultado em um dos registradores. O processo de passar dois operandos pela ULA e armazenar o resultado é denominado ciclo do caminho de dados. Quanto mais rápido for o ciclo do caminho de dados, mais rápido será o funcionamento da máquina.

Execução de instrução

A CPU executa cada instrução em uma série de pequenas etapas. Em termos simples, as etapas são as seguintes:

1. Trazer a próxima instrução da memória até o registrador de instrução.
2. Alterar o contador de programa para que aponte para a próxima instrução.
3. Determinar o tipo de instrução trazida.
4. Se a instrução usar uma palavra na memória, determinar onde essa palavra está.
5. Trazer a palavra para dentro de um registrador da CPU, se necessário.
6. Executar a instrução.
7. voltar à etapa 1 para iniciar a execução da instrução seguinte.

Tal sequência de etapas costuma ser denominada ciclo buscar-decodificar-executar.

Após a especificação da linguagem de máquina, L, para um novo computador, a equipe de projeto pode decidir se quer construir um processador de hardware para executar programas em L diretamente ou se quer escrever um interpretador para interpretar programas em L. Se a equipe preferir escrever um interpretador, também deve providenciar alguma máquina de hardware para executá-lo.

Um interpretador subdivide as instruções da máquina em pequenas etapas. Logo, a máquina na qual o interpretador roda deve ser muito mais simples e menos cara do que seria um processador de hardware para a máquina citada. a economia vem do fato de que o hardware está sendo substituído por software (o interpretador) e custa mais reproduzir hardware do que software.

Os primeiros computadores tinham conjuntos de instruções pequenos, simples. Mas a procura por equipamentos mais poderosos levou, entre outras coisas, a instruções individuais mais poderosas. Logo se descobriu que instruções mais complexas muitas vezes levavam à execução mais rápida do programa, obviamente que uma instrução mais complexa é mais cara.

A interpretação foi a solução encontrada para que computadores de **baixo custo**, com poucas instruções, pudessem executar instruções mais complicadas e de alto desempenho, porque a implementação de hardware direto (isto é, não interpretado) com instruções complexas era usada somente em computadores mais caros.

Além do custo mais baixo, computadores simples com instruções interpretadas também tinham outros benefícios, como por exemplo:

1. A capacidade de corrigir, em campo, instruções executadas incorretamente ou até compensar deficiências

de projeto no hardware básico.
2. A oportunidade de acrescentar novas instruções a um custo mínimo, mesmo após a entrega da máquina.
3. Projeto estruturado que permitia desenvolvimento, teste e documentação eficientes de instruções complexas.

Arquitetura RISC vs Arquitetura CISC

Durante o final da década de 70, houve experiências com instruções muito complexas que eram possibilitadas pelo interpretador. O objetivo era atender a demanda entre o que as máquinas podiam fazer e o que as linguagens de programação de alto nível requeriam. Quase ninguém pensava em projetar máquinas mais simples.

Houve um grupo que se opôs a essa tendência e começou a projetar chips para CPUs que não usavam interpretação. Eles cunharam o termo **RISC** para esse conceito. Na época em que o projeto desses processadores estava no início, a característica que chamou a atenção de todos era o número relativamente pequeno de instruções disponíveis quando comparados a CPUs que utilizavam instruções complexas. Inclusive, RISC quer dizer **Reduced Instruction Set Computer** (computador com conjunto de instruções reduzido).

A arquitetura **CISC** é exatamente o oposto, o acrônimo CISC significa **Complex Instruction Set Computer** (computador com conjunto de instruções complexo).

Os defensores do RISC afirmavam que o melhor modo de projetar um computador era ter um pequeno número de instruções simples que executassem em um só ciclo do caminho de dados, ou seja, buscar dois registradores, combiná-los de algum modo (por exemplo, adicionando-os ou fazendo AND) e armazenar o resultado de volta em um registrador. O argumento desses pesquisadores era de que, mesmo que uma máquina RISC precisasse de quatro ou cinco instruções para fazer o que uma CISC fazia com uma só, se as instruções RISC fossem dez vezes mais rápidas (porque não eram interpretadas), o RISC vencia.

Outro ponto importante é que, naquele tempo, a velocidade de memórias principais tinha alcançado a velocidade de memórias de controle somente de leitura, de modo que a penalidade imposta pela interpretação tinha aumentado demais, o que favorecia muito as máquinas RISC.

Mesmo com essas vantagens, o RISC não sobressaiu sobre o CISC, por alguns motivos:

1 – A questão da compatibilidade, ou seja, algumas empresas já haviam investido bilhões de dólares em software para uma linha de processadores que utilizavam CISC.

2 - A intel conseguiu empregar as mesmas ideias, mesmo em uma arquitetura CISC. A partir do 486, as CPUs da intel contêm um núcleo RISC que executa as instruções mais simples em um único ciclo do caminho de dados, enquanto interpreta as mais complicadas no modo CISC de sempre.

Princípios de projetos para computadores modernos

Há um conjunto de princípios de projeto, às vezes denominados princípios de projeto RISC, que os arquitetos de CPUs de uso geral se esforçam por seguir:

1 - Todas as instruções comuns são executadas diretamente pelo hardware, isso porque as instruções executadas por hardware são mais rápidas que as instruções interpretadas.

2 - Maximizar a taxa de execução das instruções.

3 – Instruções devem ser fáceis de decodificar. Isso inclui fazer instruções regulares, de comprimento fixo, com um pequeno número de campos. Quanto menor o número de formatos diferentes para as instruções, melhor.

4 - Somente instruções LOAD e STORE devem referenciar a memória. Todas as outras devem operar apenas

em registradores.

5 - Providenciar muitos registradores de modo que, assim que uma palavra for buscada, ela possa ser mantida em um registrador até não ser mais necessária. Esgotar os registradores e ter de descarregá-los de volta à memória só para ter de recarregá-los mais tarde é indesejável e deve ser evitado o máximo possível.

Referências bibliográficas:

TANENBAUM, Andrew S. Organização Estruturada de Computadores, 2007, 5ª Edição.

TANENBAUM, Andrew S. AUSTIN, Todd; Organização Estruturada de Computadores, 2013, 6ª Edição.