Organização de computadores

Última revisão em 04 de maio de 2016

Pelo que vimos até agora, sabemos que computadores são capazes de armazenar e manipular números representados em binário. Vários dos processamentos realizados pelo computador, por exemplo calcular a soma de dois números, podem ser vistos como um mapeamento de um conjunto de bits em outro conjunto de bits. No caso do exemplo citado, o primeiro conjunto representa os dois números a serem somados e o segundo conjunto representa o número que corresponde à soma dos dois números. Esses mapeamentos podem ser modelados por funções lógicas (binárias). As funções lógicas, por sua vez, podem ser expressas por meio de expressões lógicas. Dentre as diferentes expressões que expressam uma mesma função lógica, muitas vezes é do interesse encontrar uma que tenha custo mínimo (custo este definido de acordo com algum critério; por exemplo, quantidade de portas lógicas necessárias para a sua implementação direta em circuito lógico). Manipulações algébricas ou uso de algoritmos específicos podem ser úteis para, dada uma expressão, encontrar expressões equivalentes de menor custo.

Portanto, o conteúdo abordado até esta parte procurou cobrir as definições, conceitos e propriedades importantes para se entender como um computador realiza operações aritméticas (por exemplo, adição de dois números) ou operações lógicas (por exemplo, comparação de dois números). Isto naturalmente envolveu também o estudo sobre como os números são representados em um computador.

Por outro lado, embora não discutido detalhadamente, temos também uma noção de que computadores executam programas (sequências de instruções).

Neste documento tentaremos apresentar uma visão geral do funcionamento de um computador, desde o momento em que ele é ligado até o momento no qual ele é desligado. O conteúdo deste documento é em parte baseado em informação presente em `https://en.wikibooks.org/wiki/IB/Group_4/Computer_Science/Computer_Organisation`. A visão sobre o funcionamento de computadores aqui apresentada é relativamente breve, apenas introdutória, e não pretende ser detalhada nem completa (para isso, devem ser consultadas as referências bibliográficas apropriadas).

Na literatura da área, há duas terminologias relacionadas com a organização e funcionamento de computadores: "organização de computadores" e "arquitetura de computadores". De acordo com discussões relatados em alguns fórums, a distinção entre elas não é muito clara. Um dos entendimentos existentes é a seguinte:

- **Arquitetura de computadores**: refere-se a aspectos de desenho de um computador e especifica como é o funcionamento do hardware (conjunto de instruções, codificação das instruções, tipo de dados, forma de endereçamento, etc). Estabelece, portanto, um padrão que permite a comunicação entre as camadas superiores (por exemplo, sistemas operacionais) e o hardware. Os fabricantes de hardware implementam uma dada arquitetura; um sistema (software) que adere a essa arquitetura não depende dos detalhes de implementação.

- **Organização de computadores**: refere-se aos aspectos físicos do computador como o desenho dos circuitos, o tipo de memória, etc. São os aspectos que acabam influenciando custo, área do chip, eficiência, etc. A implementação de uma dada arquitetura pode considerar diferentes formas de organização de computadores.

As descrições que se seguem não foram escritas com a preocupação de indicar claramente se estão relacionadas à arquitetura ou organização no sentido definido acima.

1 O modelo de von Neumann

O modelo de computador proposto por John von Neumann distingue-se dos anteriores pela revolucionária ideia de armazenar as instruções na memória do computador em vez de as mesmas serem fornecidas ao computador (por exemplo, por meio de cartões perfurados).

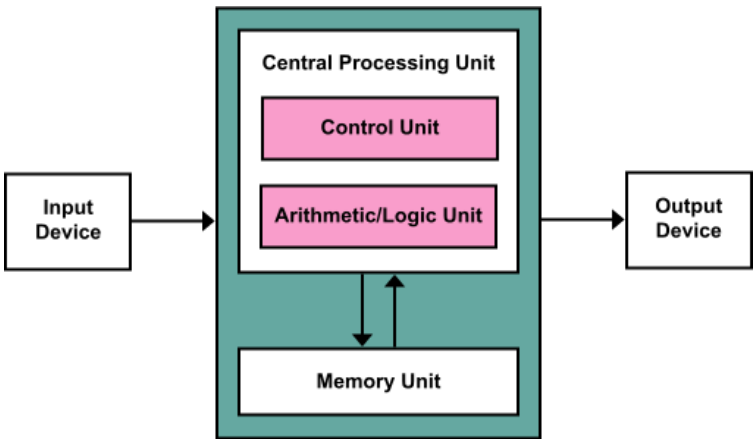

Imagem extraída de
`https://en.wikibooks.org/wiki/IB/Group_4/Computer_Science/Computer_Organisation`

1.1 Componentes do modelo de von Neumann

O modelo de von Neumann é a base dos computadores atuais, isto é, todos os computadores seguem a ideia de armazenar as instruções na memória do computador para que as mesmas sejam executadas. O modelo de von Neumann é composto de módulos, conforme ilustração acima, com as seguintes características:

Memória Módulo capaz de armazenar tanto instruções quanto dados. A memória, também conhecida por RAM (*Random Access Memory*) é uma coleção de células, identificadas por um endereço único. Podemos imaginar que essas células estão organizadas sequencialmente (linearmente), começando no endereço 0.

O tamanho da célula (em termos de número de bytes) pode variar de computador para computador. São (ou eram?) usuais células de 32 bits (4 bytes) ou de 64 bits (8 bytes). A magnitude dos números que podem ser representados no computador é definida, portanto, pelo tamanho das células (ou palavras). Por exemplo, em um computador de 32 bits, pode-se representar 2^{32} padrões binários distintos em uma célula (portanto, se considerarmos números sem sinal, podemos representar números de 0 a $2^{32} - 1$).

Os endereços são também representados com uma certa quantidade de bits; por exemplo, se considerarmos endereçamento de 8 bits, os endereços que podem ser representados variam de 0 a $2^8 - 1$. Muitos computadores atuais permitem endereçamento byte a byte. Outra possibilidade é o endereçamento célula a célula. O endereçamento está diretamente relacionado ao acesso à memória do computador, tanto para leitura como para escrita. Tipicamente, uma operação de leitura de uma posição x da memória permite que se tenha acesso aos k bytes da memória a partir da posição x, onde k é o número de bytes de uma célula.

Resumindo, a quantidade de bits das células define a magnitude dos números que podem ser representados no computador, enquanto o número de bits usados para endereçamento define a quantidade máxima de memória que pode ser endereçada.

ULA Unidade lógica-aritmética: como já vimos parcialmente, é o módulo responsável pela execução (cálculo) das operações aritméticas e lógicas.

Unidades de entrada e saída São os módulos responsáveis por permitir que os dados "entrem" e "saiam" do computador. Sem esses módulos, não seria possível que um usuário "usasse" um computador. Por exemplo, as ações executadas sobre um teclado ou com um mouse, são apropriadamente tratados por esse módulo, gerando para cada ação uma representação interna adequada para o computador.

Unidade de controle Este é o módulo responsável por executar as instruções de um programa, em um processo conhecido por *Fetch-Execute Cycle* (isto é, buscar a próxima instrução a ser executada, decodificá-la e executá-la).

A unidade de controle e a ULA estão fortemente acopladas pois a execução de várias das instruções envolve a ULA. Assim, em geral usa-se o termo Unidade de Processamento Central para se referir ao conjunto "ULA + Unidade de controle".

Os módulos acima estão conectados por canais de comunicação denominados *bus*. Há basicamente três tipos de *bus*: *data bus* para transporte de dados, *address bus* para transporte de endereços e *control bus* para o transporte de informações de controle (por exemplo, se o *data bus* está transportando da direção da memória para o processador ou vice-versa.

1.2 Fetch-Execute Cycle

Desde o momento que o computador é ligado até o momento em que ele é desligado, o processador executa incansavelmente os passos do *Fetch-Execute Cycle*. Um ciclo desses é composto de 4 ações:

1. Buscar a próxima instrução a ser executada.
2. Decodificar a instrução a ser executada.
3. Buscar dados se for necessário.
4. Executar a instrução

O endereço da próxima instrução a ser executada está armazenada em uma memória especial do processador (chamada genericamente de registrador), denominada *program counter* (PC). A instrução a ser executada é armazenada em outro registrador especial denominado *Instruction register* (IR). O endereço que está no PC é usado para fazer acesso por meio do *address bus* à célula da memória onde está a instrução, e o dado (instrução) que está na célula flui pelo *data bus*, sendo em seguida armazenado na IR. Antes da execução da instrução de um ciclo, o conteúdo do PC é incrementado de 1.

O esquema de um *Fetch-Execute Cycle* pode ser visto na figura a seguir.

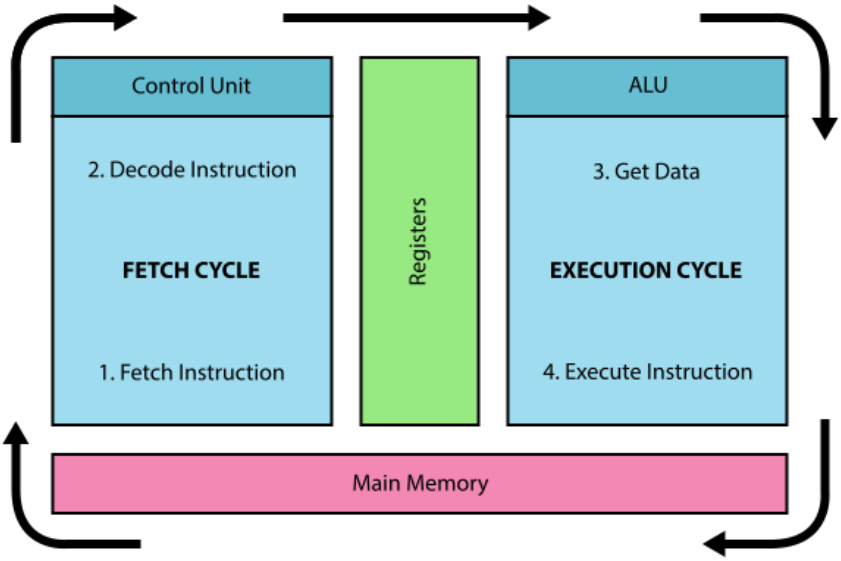

Imagem extraída de
`https://en.wikibooks.org/wiki/IB/Group_4/Computer_Science/Computer_Organisation`

2 Como funciona um computador, afinal ?

Quando um computador é ligado, alguns processamentos programados em hardware são executados. Entre esses processamentos está a ação de "carregar" e colocar em execução um sistema operacional. O sistema operacional (SO) é o programa que faz o gerenciamento do uso de recursos do computador, ou seja, garante que aplicativos (outros softwares de usuário) sejam executados harmoniosamente. Basicamente, cabe ao SO "carregar" para a memória do computador as instruções correspondentes aos diferentes programas a serem executados e colocar esses programas em uma fila de execução. Após um certo número de instruções de um programa ter sido executado, o SO altera o conteúdo do PC para que a próxima instrução a ser executada seja do próximo programa na fila de execução. Desta forma, os vários programas na fila de execução são executados em esquema de revezamento, várias vezes, até que termine. O revezamento é tão breve que o usuário não percebe que o seu programa pode ter "parado" várias vezes.

A figura a seguir mostra a hierarquia do hardware até o usuário:

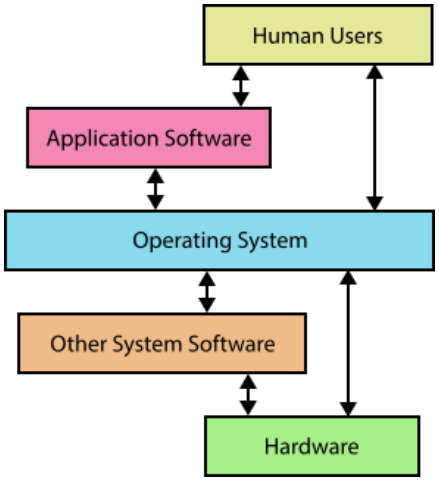

Imagem extraída de
`https://en.wikibooks.org/wiki/IB/Group_4/Computer_Science/Computer_Organisation`

3 Relação com os circuitos lógicos

Os circuitos lógicos estão diretamente relacionados com a organização dos computadores pois é esta que trata da implementação de uma arquitetura. Do ponto de vista de MAC0329, estamos interessados em entender como poderia ser uma implementação de um modelo de von Neumann enxuto.

Para implementar um modelo enxuto, precisamos implementar os principais módulos: ULA, memória e unidade de controle, bem como a comunicação entre esses módulos.

Já temos um bom conhecimento sobre o funcionamento e a implementação de uma ULA. O próximo passo será entendermos como implementar a memória. Tendo a memória, podemos especificar um conjunto de instruções e armazená-las nela. Em seguida, o desafio será fazer a unidade de controle, responsável pela execução do *Fetch-Execute Cycle*.