Hardware e Software Paralelos

Computação Paralela

Prof. Mayk F. Choji

Sumário

Background

Background

Hardware e *software* paralelos foram desenvolvidos a partir de *hardware* e *software* seriais: aqueles que rodam (mais ou menos) um único *job* por vez. Assim, antes de partir para o estudo de sistemas paralelos atuais, vamos relembrar alguns pontos sobre sistemas seriais.

Arquitetura de von Neumann

- Memória principal;
- Unidade central de processamento (CPU) ou processador ou núcleo (*core*);
- Interconexão entre memória e CPU.

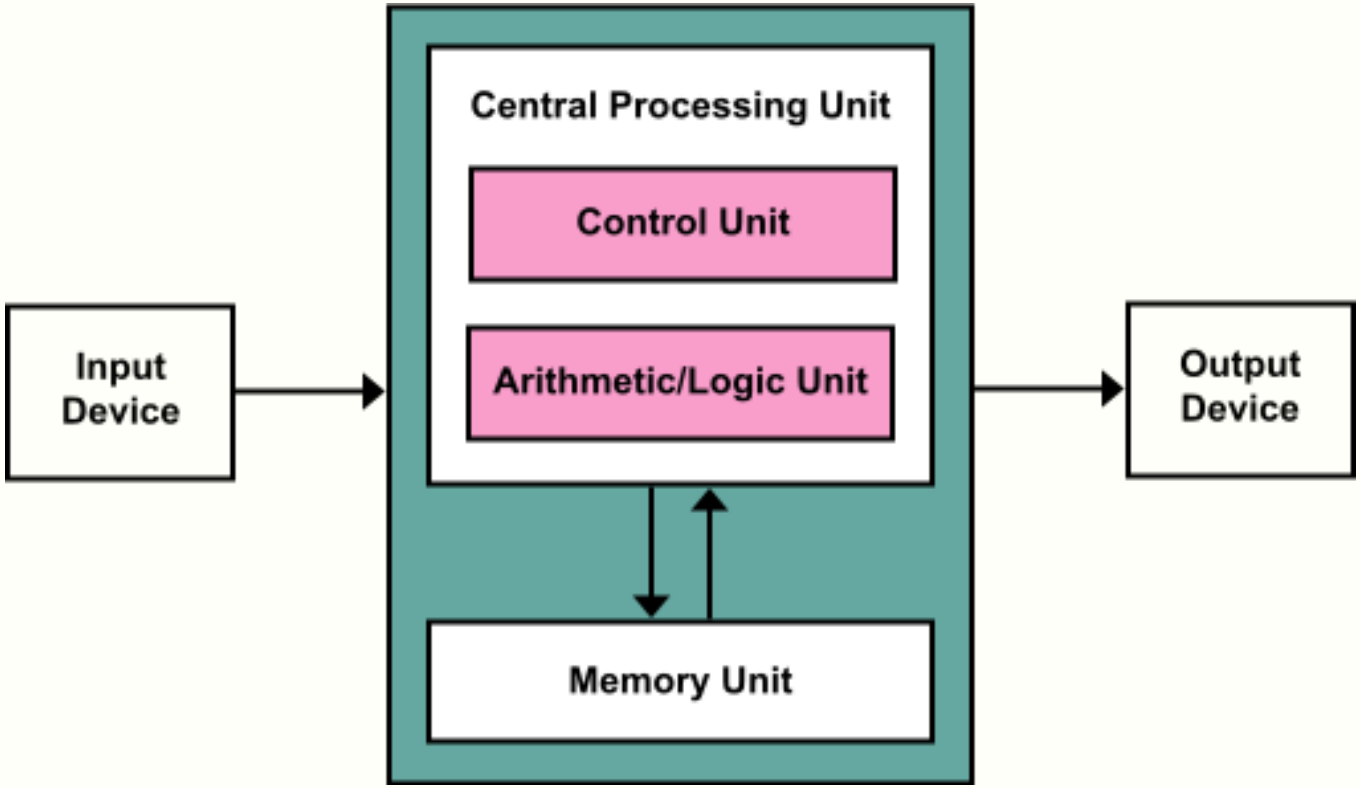
Arquitetura de von Neumann

- Memória principal;
- Unidade central de processamento (CPU) ou processador ou núcleo (*core*);
- Interconexão entre memória e CPU.

Arquitetura de von Neumann

- Memória principal;
- Unidade central de processamento (CPU) ou processador ou núcleo (*core*);
- Interconexão entre memória e CPU.

Arquitetura de von Neumann



Arquitetura de von Neumann. Fonte: Kapooht / https://is.gd/ENsqUE / CC-BY-SA-3.0

Memória

- Coleção de locais
  - Cada local armazena instruções e dados e consiste de um **endereço** e conteúdo.

CPU

Dividida em:

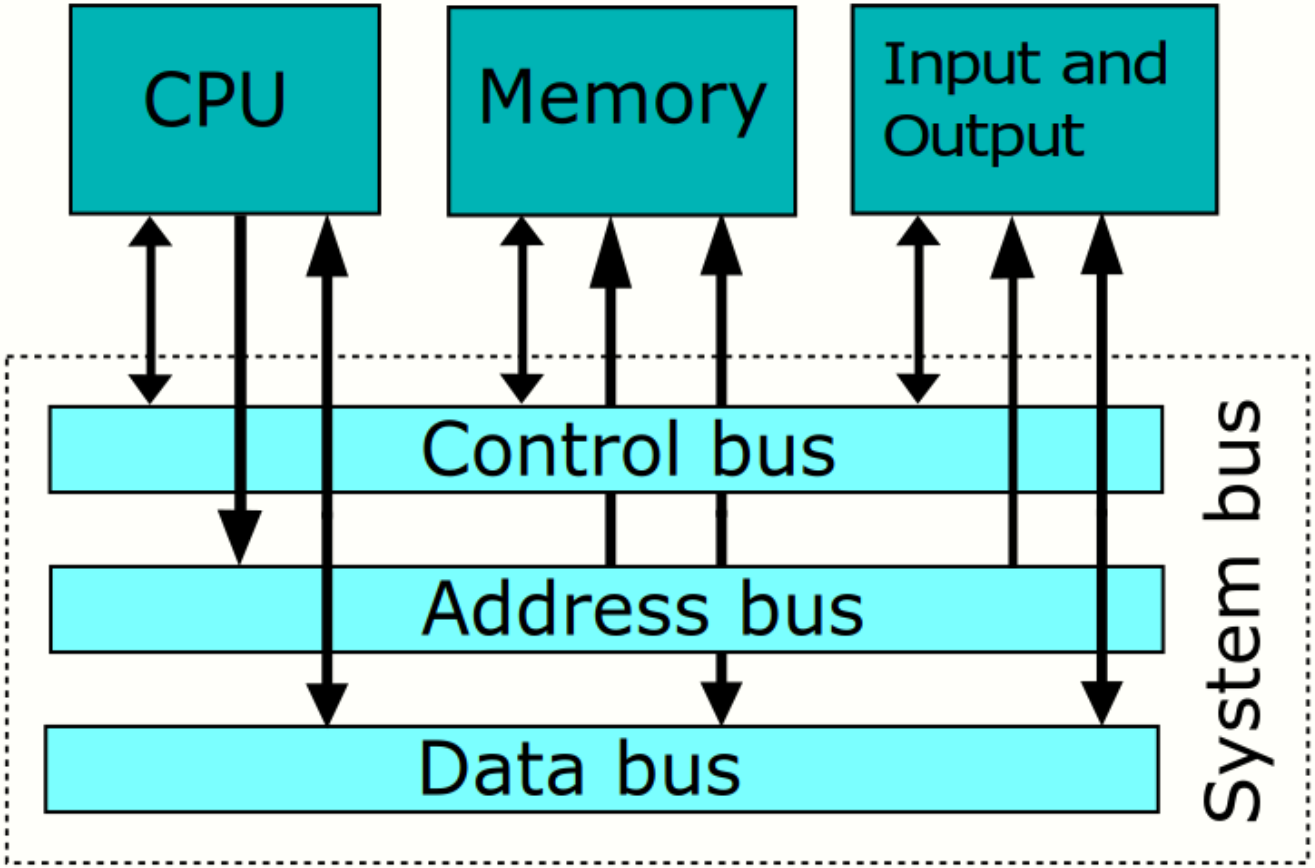
- Unidade de controle
  - Decide quais instruções de um programa devem ser executadas.
- Unidade lógica e aritmética (ALU)
  - Executa as instruções.

CPU

- Dados na CPU e informações sobre o estado de programas em execução são armazenados em registradores—armazenamento especial de alta velocidade;
- Contador de programa (*program counter*): registrador especial da unidade de controle
  - Armazena o endereço da próxima instrução a ser executada.
- Instruções e dados são transferidos entre memória e CPU via interconexão (tradicionalmente um barramento–coleção de fios paralelos e algum *hardware* controlando o acesso aos fios).

Barramento

- Um barramento é um sistema de comunicação que transfere dados entre componentes em um computador ou entre computadores;
- O termo barramento de sistema é utilizado para descrever um barramento de computador único que conecta os principais componentes de um sistema, combinando três funções:
  - Barramento de dados para transferir informação;
  - Barramento de endereço para determinar o destino;
  - Barramento de controle para determinar a operação.



Exemplo de barramento de sistema. Fonte: W Nowicki / https://is.gd/EtVSiY / CC-BY-SA-3.0

- A técnica de barramento de sistema foi desenvolvida para reduzir custos e aumentar modularidade
  - Mais popular nas décadas de 1970 e 1980;
  - Porém ainda presentes em microprocessadores embarcados;
- Computadores modernos utilizam tecnologias de interconexão de alta performance como
  - HyperTransport (2001);
  - Intel QuickPath Interconnect (2008);
  - Intel Ultra Path Interconnect (2017).

- A separação da memória e da CPU é geralmente chamada de **gargalo de von Neumann**
  - A interconexão determina a taxa na qual instruções e dados podem ser acessados;
  - Em 2010, CPUs eram capazes de executar instruções cem vezes mais rápidas do que podiam buscar itens da memória principal;
  - Para endereçar este problema (*i.e.* melhorar o desempenho da CPU), pesquisadores têm buscado várias modificações na arquitetura básica de von Neumann.

Processos, multitarefas e *threads*

- O Sistema Operacional (SO) é a parte principal de *software* cujo propósito é gerenciar recursos de *hardware* e *software* no computador
  - O SO determina **quais** programas podem rodar e **quando** eles podem rodar;
  - Controla a alocação de memória para programas em execução e o acesso a dispositivos periféricos (*e.g.* HDs e NICs).
- Quando um usuário executa um programa, o sistema operacional cria um processo—uma instância de um programa de computador que está sendo executado.

Processo

Um processo consiste de várias entidades:

- O programa em linguagem de máquina executável;
- Um bloco de memória que incluirá, entre outras coisas:
  - O código executável;
  - Uma pilha de chamadas que mantém registro das funções ativas;
  - Uma *heap*.
- Descritores de recursos alocados pelo SO (*e.g.* descritores de arquivos);
- Informação de segurança (*e.g.* quais recursos o processo pode acessar);
- Informação sobre o estado do processo (*e.g.* se o processo está pronto para ser executado ou está aguardando algum recurso, conteúdo de registradores *etc*).

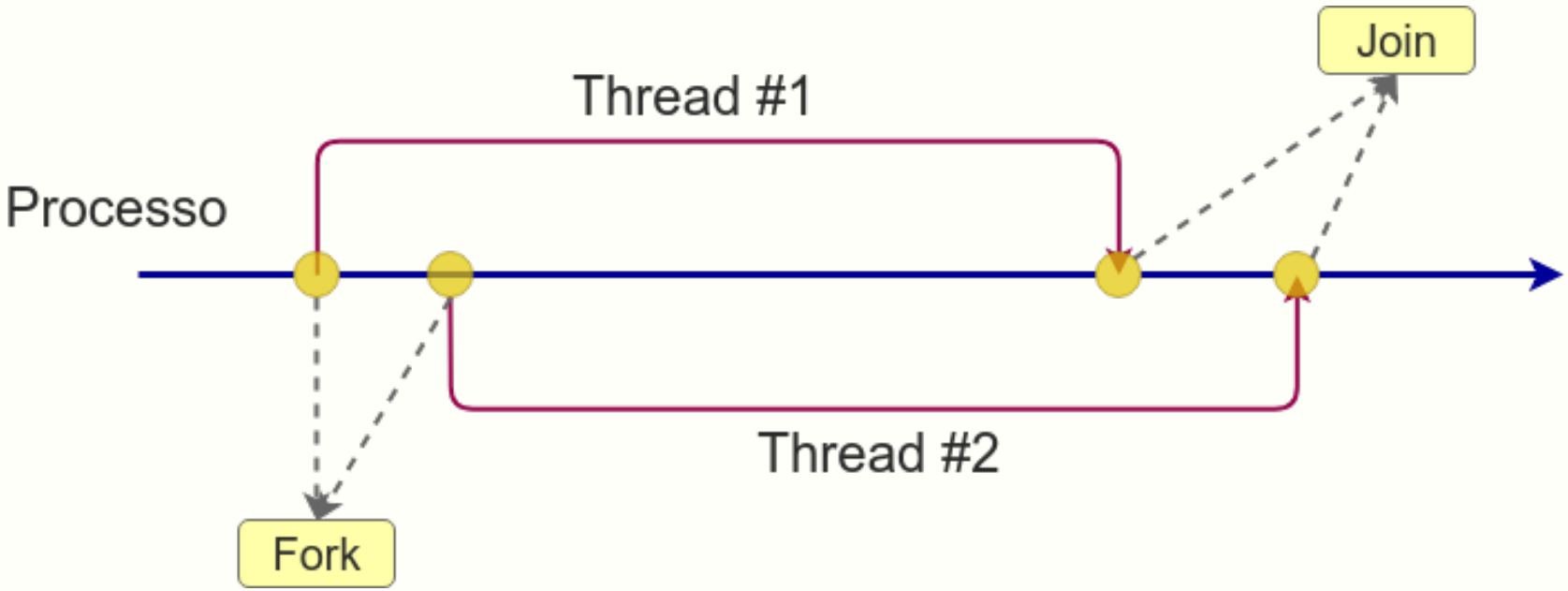
Multitarefa

- O sistema operacional provê suporte para a aparente execução simultânea de múltiplos programas;
- Possível mesmo em sistemas *single core*
  - Cada processo roda por um pequeno intervalo de tempo (geralmente alguns milisegundos).
- Um SO multitarefa pode mudar o processo em execução muitas vezes em um minuto, embora essa troca possa tomar muito tempo;
- Se um processo precisar aguardar por um recurso, ele será bloqueado
  - Irá parar de ser executado e o SO poderá executar outro processo.

Threads

- Provê um mecanismo para programadores dividirem seus programas em tarefas mais ou menos independentes
  - Uma *thread* pode estar rodando enquanto outra está bloqueada.
- A troca entre *threads* é muito mais rápida do que entre processos;
- São contidas em processo, então podem usar o mesmo executável, e geralmente compartilham a mesma memória e os mesmos dispositivos de E/S;
- Precisam de registros de contadores de programa e pilhas de chamadas próprias;
- Utilizam conceitos de *fork* e *join*.

Threads



Um processo e duas *threads*.

Modificações no Modelo de von Neumann

Modificações no Modelo de von Neumann

- Os primeiros computadores foram desenvolvidos nos anos de 1940;
- Muitas melhorias foram feitas desde então na arquitetura básica de von Neumann
  - Reduzir o problema do gargalo de von Neumann;
  - Tornar a CPU mais rápida.
- Algumas melhorias:
  - *Caching*;
  - Memória virtual;
  - Paralelismo em baixo nível.

Cache

- *Caching* é um dos métodos mais usados para endereçar o gargalo de von Neumann;
- Em geral, uma *cache* é uma coleção de locais de memória que podem ser acessadas em menos tempo que outros locais;
- Quando falamos em *cache*, geralmente nos referimos à *cache* da CPU
  - Conjunto de locais de memória que a CPU pode acessar mais rapidamente que a memória principal;
  - Pode ser localizada no mesmo *chip* que a CPU ou em um *chip* separado que possa ser acessado muito mais rápido que um *chip* de memória comum.

Memória Virtual

- Ao se executar um programa muito grande, ou manipular uma grande quantidade de dados, as instruções ou dados podem não caber na memória principal;
- Mais comum em sistemas multi-tarefas;
- Compartilhamento da memória principal pelas diversas tarefas em execução;
- Cada processo enxerga um espaço de memória próprio;
- Operação sobre blocos de dados e instruções conhecidos como **páginas** (4 a 15KB, geralmente);
- Tradução de páginas (**endereços virtuais**) para endereços físicos feita por meio de **tabela de páginas**.

Hardware Paralelo

Taxonomia de Flynn

- Em computação paralela, a taxonomia de Flynn é frequentemente usada para classificar arquiteturas de computadores;
- Classifica um sistema de acordo com o número de fluxo de instruções e o número de fluxo de dados que ele pode gerenciar simultaneamente;

Taxonomia de Flynn

- Abrange quatro classes de arquiteturas de computadores:

  SISD (*Single Instruction Single Data*) fluxo único de instruções sobre um único conjunto de dados;

  SIMD (*Single Instruction Multiple Data*) fluxo único de instruções em múltiplos conjuntos de dados;

  MISD (*Multiple Instruction Single Data*) fluxo múltiplo de instruções em um único conjunto de dados;

  MIM (*Multiple Instruction Multiple Data*) fluxo múltiplo de instruções sobre múltiplos conjuntos de dados.

- Um sistema clássico de von Neumann é, portanto, um sistema SISD;
- SIMD está relacionado com paralelismo de dados.
- Na década de 1990, os únicos sistemas SIMD amplamente produzidos eram processadores vetoriais. Recentemente, GPUs e CPUs de *desktop* estão fazendo uso de aspectos da arquitetura SIMD. (Pacheco 2011).

GPU

- APIs de interface gráfica utilizam pontos, linhas e triângulos para representar superfícies de objetos;
- Usam *pipeline de processamento gráfico* para converter representação interna em *pixels* na tela;
- Comportamento dos estágios definidos por funções ***shader***;
  - Funções implicitamente paralelas;
  - Uso de paralelismo SIMD;
- GPUs não são totalmente SIMD. Novas gerações com dezenas de *cores* permitem execução de instruções independentes (Pacheco 2011).

Referências

Pacheco, Peter (2011). *An introduction to parallel programming*. Elsevier.

Mayk F. Choji, 2019