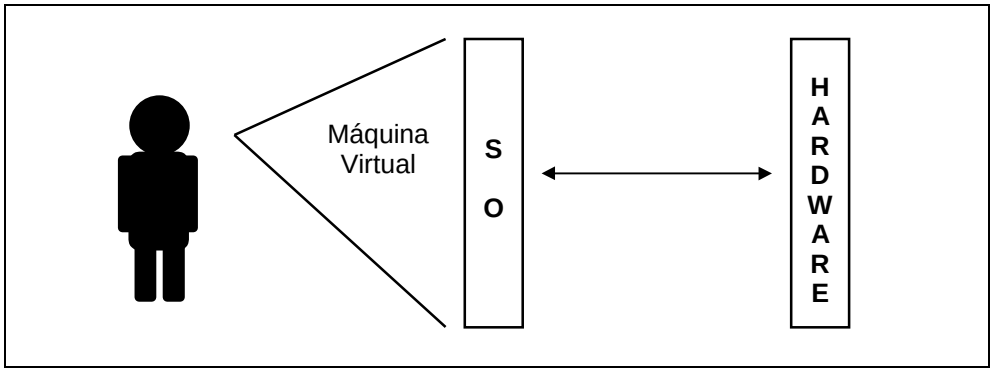
1. INTRODUÇÃO AOS SISTEMAS OPERACIONAIS

1.1 Introdução

- Programas de Aplicação: desenvolvidos pelo usuário.
- Programas Utilitários: ferramentas de auxílio às aplicações.
- Programa: conjunto de comandos com finalidade específica.
- Processo: programa em estado de execução (consumindo recursos da máquina).
- Sistema Operacional: programa que estende os recursos de hardware da máquina tornando o uso do equipamento mais fácil (transparência), econômico (compartilhamento de recursos), eficiente (maior aproveitamento do processador) e confiável (isolamento entre usuários). Sua função é controlar o funcionamento do computador, gerenciando os recursos disponíveis através de:

Facilidade de acesso aos recursos do sistema

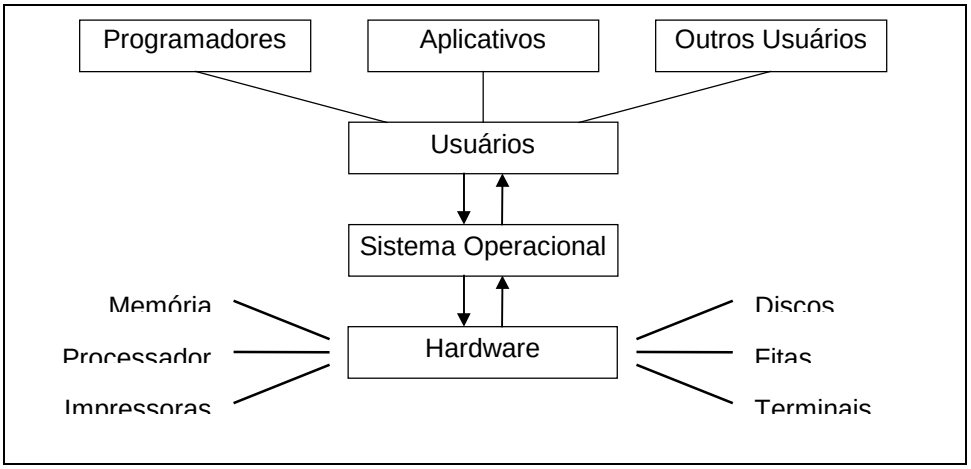
- Não nos preocupamos com a maneira como é feita a comunicação entre os programas e os dispositivos, como a leitura de um disquete: acionar a cabeça de leitura posicionando na trilha e setor desejados, transferir os dados do disco para a memória e, por fim, informar ao programa a chegada dos dados.
- *Virtual Machine*: ambiente simulado, criado pelo sistema operacional, de modo a permitir a comunicação entre o usuário e os recursos do sistema, de forma transparente, mais eficiente e com menores chances de erros. O usuário "enxerga" a máquina como sendo apenas o sistema operacional, como se o *hardware* não existisse.



- Compiladores, *linkers*, bibliotecas, depuradores e outras ferramentas semelhantes não fazem parte do sistema operacional, destinando-se a ajudar na interação do usuário com o computador.

Compartilhamento de recursos de forma organizada e protegida

- Diversos usuários compartilhando os mesmos recursos
- Chances iguais para acesso e de maneira isolada (sem interferência de um no trabalho do outro).
- Gerenciamento do acesso concorrente aos recursos, como impressoras e disco.
- Impressão de ser o único à utilizar o recurso.
- Diminuição de custos pelo compartilhamento dos recursos.



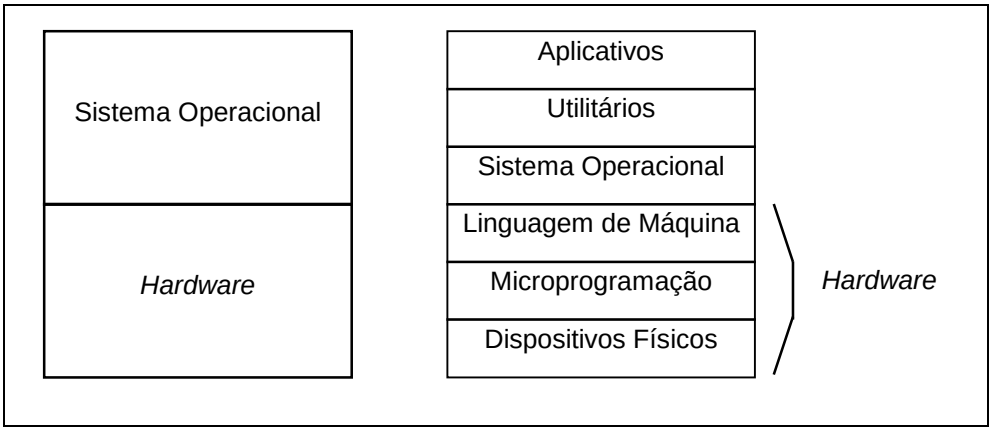
1.2 Conceitos básicos

1.2.1 Máquina de níveis

Hardware e Software são logicamente equivalentes. Operações efetuadas pelo software podem ser implementadas em hardware e instruções executadas pelo hardware podem ser simuladas via software (Tanenbaum).

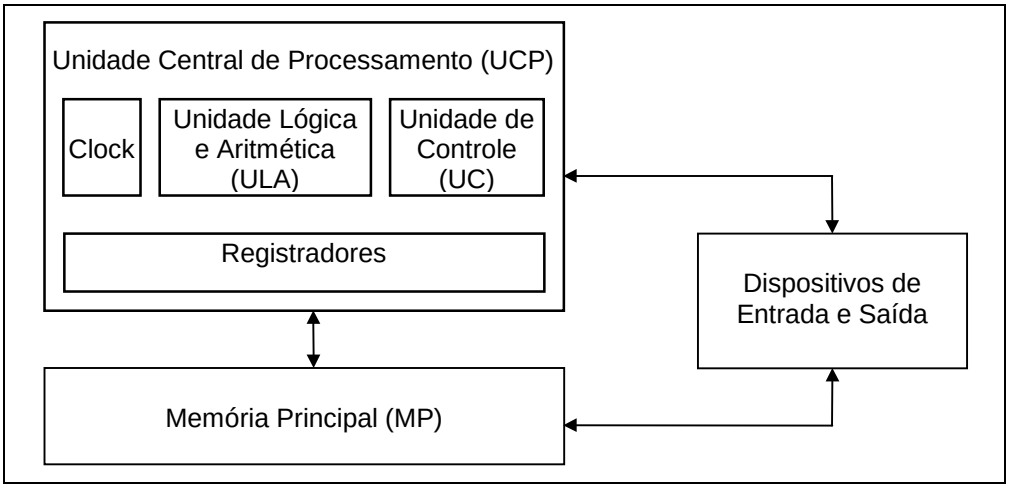
Primeiramente, o programador devia conhecer o hardware e sua linguagem de máquina (programação em painéis, através de fios). Com o sistema operacional isto tornou-se transparente para o usuário (Máquina Virtual).

O computador não possui apenas dois níveis, e sim tantos quantos o usuário necessitar para realizar suas tarefas. Atualmente, a maioria dos computadores possui a estrutura abaixo, podendo conter mais ou menos camadas. Cada nível utiliza uma linguagem própria e o usuário não necessita saber da existência das outras camadas, abaixo ou acima da sua.



1.2.2 *Hardware*

O computador é constituído por um conjunto de componentes interligados agrupados em três subsistemas básicos (unidades funcionais): unidade central de processamento, memória principal e dispositivos de entrada e saída. As implementações variam de acordo com o fabricante, porém todos estão presentes.



1.2.2.1 Unidade central de processamento

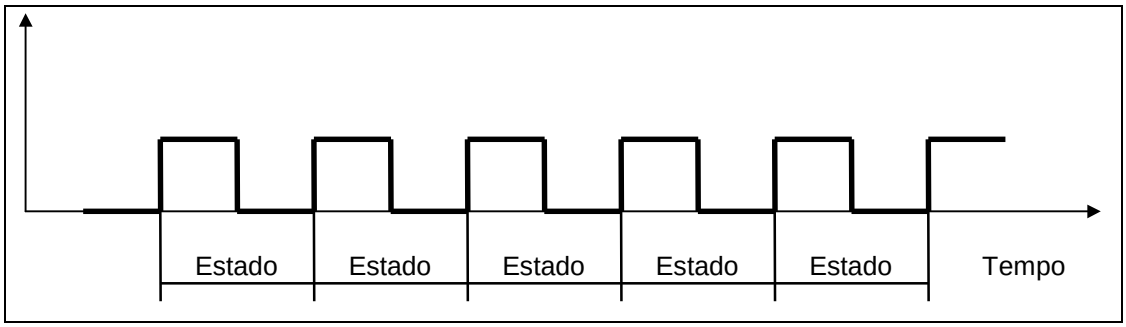
Ou simplesmente processador.

- Unifica todo o sistema controlando as funções realizadas pelas unidades funcionais.
- Executa todos os programas, sendo que estes estão na memória principal.
- A UCP busca a instrução na MP e a interpreta para executá-la.
- A UC é responsável pela emissão de sinais de controle (gravação em disco ou busca de instrução na memória)
- A ULA é responsável pela realização de operações lógicas (comparações) e aritméticas (somas e subtrações).

- A velocidade da UCP é determinada pelo número de instruções que ela executa em uma unidade de tempo (não existe padronização). MIPS (milhões de instruções por segundo - PS2/50 = 2 MIPS)

1.2.2.2 Clock

- Dispositivo da UCP que gera pulsos elétricos constantes (síncronos) em um mesmo intervalo de tempo (sinal de clock).
- Este intervalo determina a freqüência de geração dos pulsos e conseqüentemente seu período.
- A cada período do sinal de clock dá-se o nome de **estado**.
- O sinal é usado pela UC para executar as instruções.
- Como uma instrução demora vários estados, em um estado apenas parte dela é executada.
- O tempo de duração da instrução é determinado pelo seu número de estados e o tempo de duração do estado.



1.2.2.3 Registradores

- Dispositivos de alta velocidade localizados na UCP.
- Armazenam dados temporariamente.
- Alguns são de uso específico, outros de uso geral.
- O número varia de acordo com a arquitetura do processador.

Merecem destaque:

- contador de instruções (*program counter* - PC): armazena a instrução que a UCP deve executar.
- apontador da pilha (*stack pointer* - SP): contém o endereço de memória do topo da pilha (estrutura do sistema que guarda informações sobre tarefas que tiveram de ser interrompidas por algum motivo.
- registrador de estado (*program status word* - PSW): armazena informações sobre a execução do programa, como *overflow*.

1.2.2.4 Memória principal

- Composta por unidades chamadas células.
- Volátil (necessita estar energizada).
- O tamanho da célula (em bits) varia de acordo com a arquitetura do equipamento.
- O acesso é feito através de um número chamado endereço de memória.
- O endereço é especificado no registrador de endereço de memória (*memory register address* - MAR).
- Com o conteúdo desse registrador a UCP acessa o endereço.
- O conteúdo do endereço acessado é armazenado no registrador de dados de memória (*memory buffer register* - MBR).

Operação de Leitura	**Operação de Gravação**
1. O MAR recebe da UCP o endereço da célula a ser lida.	1. A UCP armazena no MAR o endereço da célula que será gravada.
2. A UCP gera um sinal de controle para a memória principal, indicando que uma operação de leitura deve ser realizada.	2. A UCP armazena no MBR a informação que deverá ser gravada.
3. O conteúdo da célula, identificada pelo endereço contido no MAR, é transferido para o MBR	3. A UCP gera um sinal de controle para a memória principal, indicando que uma operação de gravação deve ser realizada.
	4. A informação contida no MBR é transferida para a célula de memória endereçada pelo MAR.

• A capacidade de memória é limitada pelo tamanho do MAR (2^n bits, ou seja, de 0 a 2^n - 1) (IBM PS2/50 = 16 Mb).

1.2.2.5 Memória cache

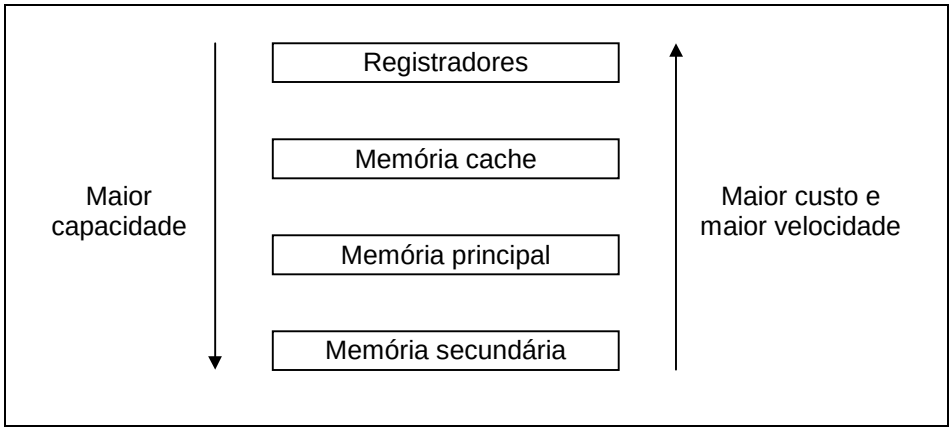
- Alta velocidade.
- Antes de acessar um dado na MP o processador busca ele na memória cache.
- Se encontra não acessa a MP, caso contrário transfere um bloco de dados a partir do dado referenciado para a cache.
- Maior performance.
- Alto custo.

1.2.2.6 Memória secundária

- Meio permanente de armazenamento (não-volátil).
- Acesso lento se comparado às outras memórias.
- Custo baixo.
- Capacidade de armazenamento muito superior à MP.
- Fitas magnéticas, discos magnéticos e óticos.

1.2.2.7 Dispositivos de entrada e saída

- Permitem comunicação entre o computador e o mundo externo.



- Dispositivos de memória secundária (discos e fitas) e de interface homem-máquina (teclados, monitores, impressoras, *plotters*).
- Interfaces mais amigáveis como *scanners*, canetas óticas, *mouses*, microfones e *touch-screens*.

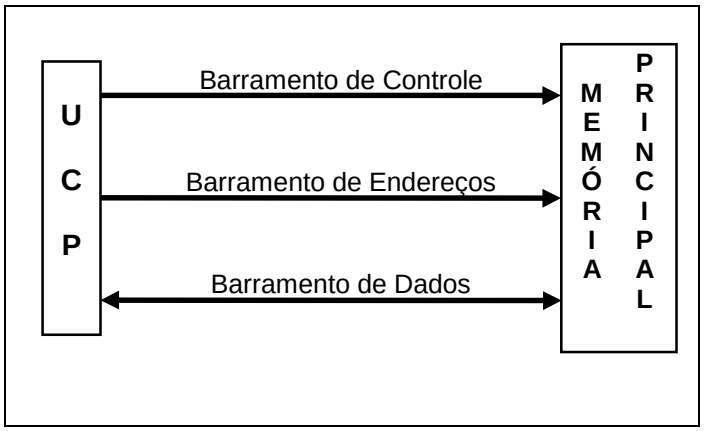
1.2.2.8 Barramento

- Linhas de comunicação entre a UCP, a MP e os dispositivos de E/S.
- De acordo com a transmissão, podem ser unidirecionais (um só sentido) ou bidirecionais (em dois sentidos).
- Exemplo: comunicação UCP / MP.

Barramento de Dados: transmite informações entre as unidades funcionais (MBR).

Barramento de Endereços: especifica o endereço da célula a ser acessada (MAR).

Barramento de Controle: sinais relativos a leitura/gravação.



1.2.2.9 Inicialização do sistema (Boot)

- SO é essencial pois disponibiliza a totalidade dos recursos disponíveis no equipamento.
- Ao ligar o computador é necessário que o SO seja transferido da memória secundária para a principal.

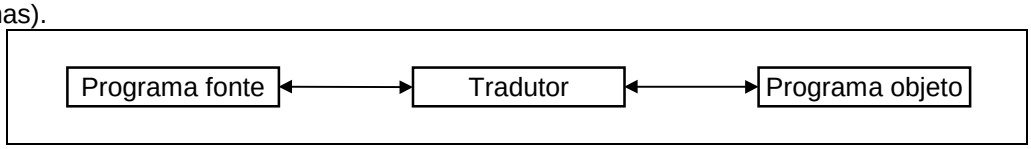
- Este processo é denominado *boot* e é realizado por um programa localizado em uma posição específica do disco (boot block, Master Boot Record).

1.2.3 *Software*

- Torna o hardware útil.
- Facilita o trabalho do usuário (conceito de camadas).

1.2.3.1 Tradutor

- Programas escritos em linguagem de alto nível
- Menor preocupação com aspectos de *hardware* (endereços de memória para carga e variáveis).
- Não executado diretamente, precisam ser "traduzidos" para um módulo chamado objeto (linguagem de máquina, mas ainda não executável na maioria dos casos, pois podem necessitar de rotinas externas).



- Dividem-se em:
  - Montador: linguagem de montagem (mnemônicos) associada à linguagem de máquina do processador. Ligada diretamente ao processador, diferente, portanto, para cada fabricante. (linguagem *assembly*).
  - Compilador: linguagem de alto nível, independente de implementação (portável).

1.2.3.2 Interpretador

- Traduz mas não gera código objeto.
- Traduz e executa a instrução logo a seguir.
- Tempo de execução elevado
- Exemplos: Basic e dBase.

1.2.3.3 Linker

- Gera, a partir de um ou mais códigos objetos, um único código executável.
- Resolve referências externas utilizando-se de bibliotecas (módulos objetos ou definições de símbolos).
- Determina a região da memória onde o programa será carregado para ser executado (relocação).
  - Código absoluto: os endereços são resolvidos em tempo de linkedição (inviável em ambientes multiprogamáveis.
  - Código relocável: os endereços serão resolvidos no momento da carga.

1.2.3.4 Loader

- Ou carregador, coloca fisicamente o programa na memória.
- O procedimento varia de acordo com o código gerado pelo *linker*.
  - Se absoluto: deve ser conhecido o endereço de memória inicial e o tamanho do módulo.
  - Se relocável: pode ser carregado em qualquer lugar da memória.
- A execução é iniciada logo após a carga.

1.2.3.5 Depurador

- Utilizado para encontrar erros de lógica.
- Permite ao usuário:
  - Acompanhar a execução do programa instrução à instrução.
  - Verificar e alterar valores de variáveis.
  - Criar pontos de parada (*breakpoints*).
  - Determinar o envio de mensagens quando da alteração de uma variável (*watchpoints*).

1.2.3.6 Linguagem de controle

- Forma mais direta de se comunicar com o SO.
- Comandos interpretados pelo *shell* (interpretador de comandos).
- Seqüência de comandos em um arquivo (tipo *batch*).

1.2.3.7 Linguagem de máquina

- Linguagem que o computador realmente consegue entender.
- Codificada em formato binário.
- Programas longos com maiores chances de erros.
- Conjunto de instruções é específico de cada processador.

1.2.3.8 Microprogramação

- Para cada instrução em linguagem de máquina, existe um microprograma associado.
- Normalmente existem 25 microinstruções básicas interpretadas pelos circuitos eletrônicos.
- Computadores microprogramáveis permitem a criação de novas instruções de máquina com novos microprogramas.
- Em PC os microprogramas estão gravados em ROM.

1.3 Histórico

Evolução diretamente ligada ao hardware.

1.3.1 1ª. Fase (1945-1955)

- Não existia o conceito de SO.
- Máquinas enormes, lentas e imprecisas (milhares de válvulas).
- Necessário profundo conhecimento de hardware.
- Programação em linguagem de máquina através de painéis.

1.3.2 2ª. Fase (1956-1965)

- Transistor (velocidade e confiabilidade).
- Memórias magnéticas (maior capacidade, mais compactos).
- Primeiras linguagens (Assembly, Fortran).
- Processamento em batch (maior utilização do processador).
- Rotinas de SO para E/S (IOCS - Input/Output Control System).

1.3.3 3ª. Fase (1966-1980)

- Circuitos integrados e microprocessadores.
- Diminuição de custos, aumento do poder de processamento.
- Multiprogramação (*time-slice* - um programa realiza I/O, outro executa).
- Spooling (alteração na ordem de execução das tarefas - não puramente seqüencial).
- Time-sharing (tempo compartilhado) programas usam o processador durante pequenos intervalos de tempo.

1.3.4 4ª. Fase (1981-1990)

- VLSI (Very Large Scale Integration) miniaturização e menores custos.
- PC's + DOS e minis + UNIX-like.
- Multiprocessamento (mais de uma UCP).
  - Processamento Vetorial.
  - Processamento Paralelo.
- Redes (heterogeneidade).

1.3.5 5ª. Fase (1991-)

- Hardware, software e telecomunicações.
- Maior capacidade de processamento e armazenamento.
- Multimídia, Inteligência Artificial e Bancos de Dados Distribuídos.
- ULSI (Ultra Large Scale Integration).
- Processamento distribuído (vários processadores e redes).
- Interfaces homem-máquina (linguagem natural, sons e imagens).

1.4 Tipos de sistemas operacionais

1.4.1 Monoprogramação

- Dedicado à um único usuário.

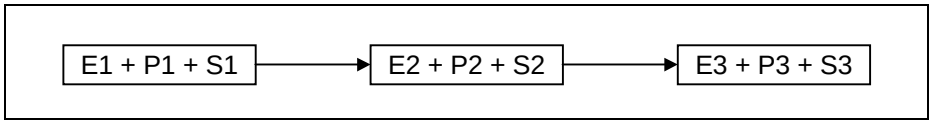
- Somente um programa residente na memória.
- Processador ocioso enquanto o programa espera interação do usuário.
- Periféricos e memória, geralmente, subutilizados.
- Implementação relativamente simples (sem preocupações com proteção).

1.4.2 Multiprogramação

- Mais complexos e eficientes (divisão de recursos entre usuários).
- Compartilhamento de memória, processador e periféricos (menor custo de utilização).
- Gerenciamento de acesso concorrente aos recursos (ordenado e protegido).
- *Workstations* (ambiente multitarefa disponível para um usuário).
- Suportam mais de um tipo de processamento (*batch, time-sharing, real-time*).

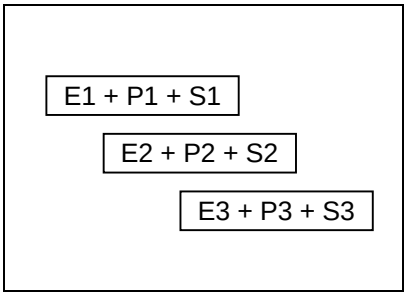
1.4.2.1 Lote (batch)

- *Jobs* (tarefas), sem interação com o usuário, executados seqüencialmente.
- *Sort*, compilações, *backup*, etc.
- Podem apresentar tempos de resposta longos (seqüenciais).



1.4.2.2 Tempo compartilhado (time-sharing)

- Respostas do SO em poucos segundos.
- Para cada usuário uma fatia de tempo (*time-slice*) é alocada pelo processador. Caso a execução exceda a fatia de tempo ele é substituído por outro e fica esperando nova oportunidade.
- Todos os recursos são compartilhados, porém o usuário tem a impressão de estar utilizando a máquina sozinho.
- Implementação complexa porém aumentam a produtividade dos usuários, reduzindo custos de utilização.



1.4.2.3 Tempo real (real-time)

- Tempo de resposta dentro de limites rígidos.
- Não existe o conceito de fatia de tempo. Um programa executa até que outro, com maior prioridade assuma o processador.
- Presentes em situações onde segurança é o principal fator (refinarias, tráfego aéreo, usinas, etc.).

1.4.3 Multiprocessamento

- Mais de um processador.
- Conceitos de multiprogramação aplicados aos vários processadores, permitindo, ainda:
  - Reconfiguração: se um processador falha o processamento continua, mesmo com menor poder.
  - Balanceamento: permite balancear a carga de processamento entre os diversos processadores envolvidos.
- Vários programas executados ao mesmo tempo ou um programa executando em vários processadores (capacidade de computação ampliada).
- Alternativa para baratear o custo do hardware (menor velocidade, porém mais processadores).

1.4.3.1 Acoplamento

Designa a forma de comunicação entre os processadores e o grau de compartilhamento da memória e dos dispositivos de I/O.

Sistemas fracamente acoplados (*loosely coupled*)

- Dois ou mais sistemas conectados por uma linha (*link*).
- Os recursos pertencem à cada sistema.
- Os sistemas operacionais podem ser diferentes.
- Possibilidade de balanceamento de carga (o processador mais livre é escolhido).
- Compartilhamento de recursos (impressoras, discos, etc.).
- Alguma tolerância à falhas (perdem-se os recursos do equipamento que falhar).

- Possibilidade ilimitada de crescimento.
- Exemplo: redes locais.

Sistemas fortemente acoplados (*tightly coupled*)
Os processadores compartilham uma única memória e são controlados por um único SO.
Assimétricos
- Um processador primário (*master/mestre*) que gerencia os demais e executa o SO.
- Processadores secundários (*slave/escravo*) apenas executam programas de usuário. Todos os serviços são solicitados ao processador primário.
- Muitas operações de I/O sobrecarregarão o processador *master* (ineficiência).
- Em caso de falha do mestre outro processador deve assumir seu lugar (reconfiguração).

Simétricos
- Todos os processadores têm as mesmas funções.
- Executam o SO independentemente.
- Acessos simultâneos à memória (resolvidos pelo hardware e pelo SO).
- Programas executados por qulaquer processador (ou mais de um ao mesmo tempo).
- Em caso de falha o processamento continua normalmente (com menor capacidade).
- Implementação complexa porém com melhor balanceamento de processamento e I/O.

1.4.3.2 Paralelismo

- Em sistemas multiprocessáveis uma tarefa pode ser dividida e executada, simultaneamente, por mais de um processador.
- O paralelismo pode ser obtido em diferentes níveis: instrução, vetor, sub-rotinas e programas.

Pipelining (nível de instrução): a instrução é dividida em unidades menores (busca, execução e armazenamento). Enquanto uma instrução está na fase de execução, outra se encontrará na fase de busca. É a técnica mais utilizada para aumento de desempenho.

Processamento vetorial (nível de vetor): manipula vetores inteiros. Os computadores que o implementam são caros e complexos, sendo utilizados em aplicações científicas e militares (problemas numéricos, processamento de imagens, meteorologia, física nuclear, etc.). Possuem, também, um processador escalar e várias unidades funcionais *pipelining*.

Processador Escalar	**Processador Vetorial**
FOR i= 1 TO N DO C[i] := A[i] + B[i]	C := A + B
N operações de busca e execução	Uma operação de busca e execução apenas

Processamento paralelo (nível de rotina/programa): uma aplicação sendo executada por mais de um processador simultaneamente. O problema é detectar se existe a possibilidade de paralelismo no programa (pode ser indicado pelo programador). É muito pouco utilizado (metodologias puramente seqüenciais, dificuldade de depuração e prova dos programas).

Operação aritmética			Operação vetorial			
x:= (a * b) + (c * a) + (d * b)			FOR i= 1 TO 100 DO Vetor[i] := 0;			
Proc. 1	Proc. 2	Proc. 3	Proc. 1	Proc. 2	...	Proc. 100
a * b	c * a	d * b	Vetor[1] := 0	Vetor[2] := 0	...	Vetor[100] := 0