Fundamentos da Arquitetura de Computadores

CPU

Prof. André Renato
1º Semestre / 2012

CPU

- A CPU é o elemento central dos sistemas de computação, responsável pelas operações de processamento e controle, durante a execução de um programa;
- Um programa, para ser efetivamente executado pelo processador, deve ser constituído de uma série de instruções de máquina. Para que a execução tenha início, as instruções devem ser armazenadas na memória principal;

CPU

- As funções da CPU são:
  - Buscar na memória a instrução a ser executada;
  - Interpretar que operação a instrução está explicitando;
  - Buscar os dados onde estiverem armazenados;
  - Executar efetivamente a operação com os dados e armazenar o resultado no local definido pela instrução;
  - Reiniciar o processo, buscando a próxima instrução.

CPU

- Estas etapas compõem o que se denomina um *ciclo de instrução.* Este ciclo se repete indefinidamente até que o sistema seja desligado, ou ocorra algum tipo de erro, ou seja encontrada uma instrução de parada. Em outras palavras, a CPU é projetada e fabricada com o propósito único de executar sucessivamente pequenas operações, na ordem definida pela organização do programa.

CPU

- Entre os operações de processamento de dados, podemos citar:
  - Operações aritméticas
  - Operações lógicas
  - Movimentação de dados
  - Desvios
  - Operações de entrada ou saída

CPU

- Estas operações são realizada por um componente chamado de Unidade Lógico-Aritmética (ULA);
- A ULA é um conjunto de circuitos digitais que normalmente recebe um dois operandos e produz um resultado;

CPU

- Outra tarefada CPU é realizar o controle dos demais dispositivos de acordo com a instrução que está sendo executada;
- Ex: uma instrução de soma vai ativar o circuito somador da ULA; uma instrução de entrada/saída vai sinalizar ao dispositivo o que deve ser feito;

CPU

- Para que a CPU possa processar os dados e movê-los corretamente, ele precisará de uma pequena memória para manter os dados enquanto estão sendo manipulados;
- Esta memória é formada por pequenos componentes chamados de registradores;

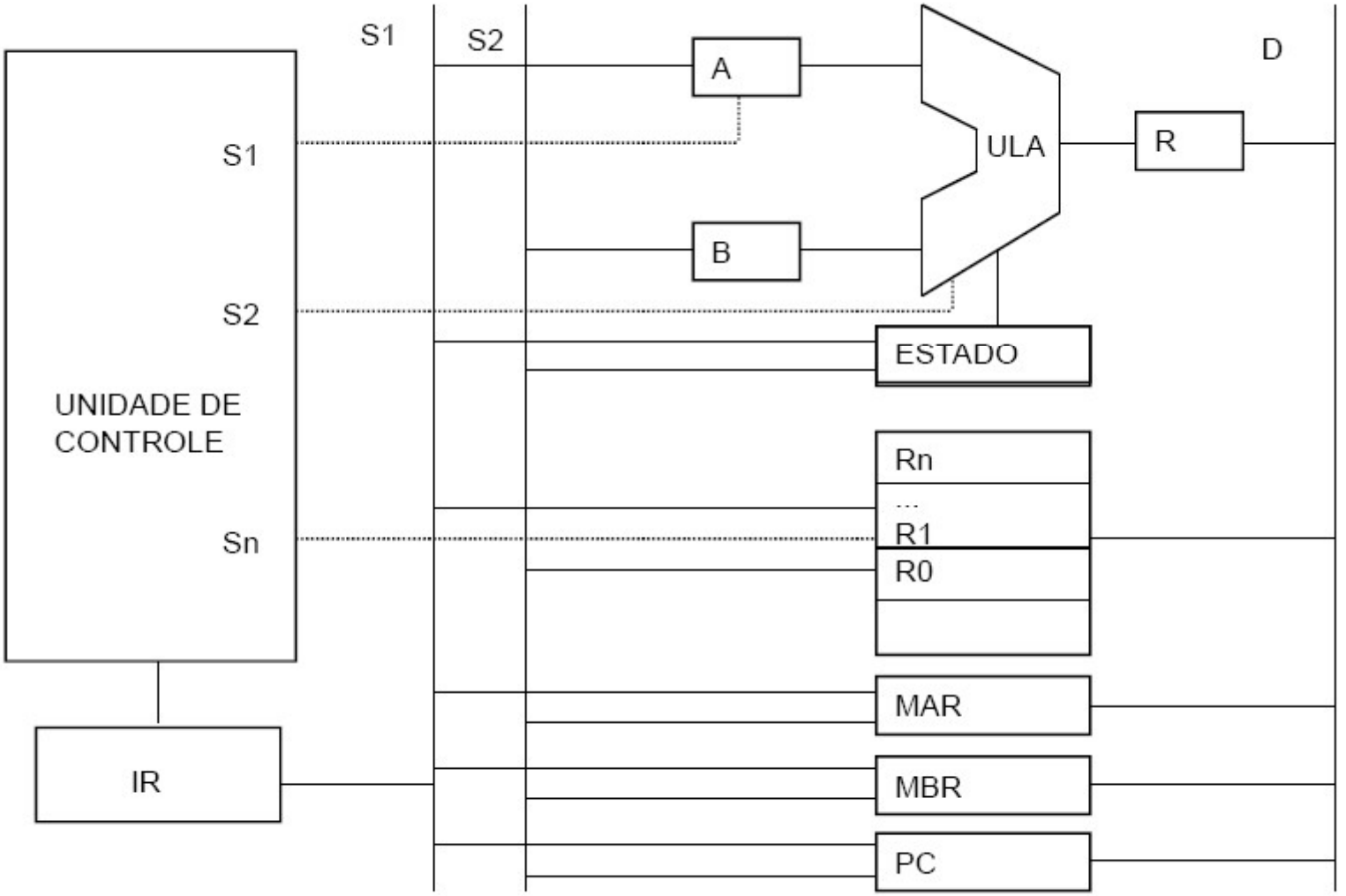
CPU

- Registradores:
  - A, B, R: registradores temporários que armazenam, respectivamente, os valores a serem operados e o valor resultante
  - R0 ... Rn: registradores de dados MAR: endereço da locação de memória onde será feito o acesso
  - MBR: armazena temporariamente a informação transferida de ou para a locação de memória endereçada pelo MAR
  - PC: contador de programa, contém o endereço da locação de memória onde se encontra a próxima instrução a ser executada
  - IR: registrador de instruções
  - ESTADO: guarda informações sobre o resultado produzido pela ULA. Exemplo: o bit n é 1 se o resultado for nulo e 0 se for não-nulo

CPU

- Uma arquitetura de processador é uma arquitetura de n bits quando todas as operações da ULA podem ser realizadas sobre operandos de até *n bits;*
- Normalmente em uma arquitetura de *n bits, os* registradores de dados e os barramentos internos também são de n bits, de forma a permitir que os dados sejam armazenados e transferidos de forma eficiente;

CPU


S1
S2
A
D
S1
ULA
R
B
S2
ESTADO
UNIDADE DE
CONTROLE
Rn
...
Sn
R1
R0
MAR
IR
MBR
PC

CPU

- Unidade de controle:
  - Todas as operações básicas que ocorrem dentro da seção de processamento são comandadas pela seção de controle. Ao efetuar a busca da instrução, a unidade de controle interpreta a instrução de modo a identificar quais as operações básicas que devem ser realizadas e ativa sinais de controle (S1, S2, ..., S*n) que fazem uma operação básica de fato* acontecer
  - Em outras palavras, a seção de controle é projetada para entender o quê fazer, como fazer e comandar quem vai fazer no momento adequado.

CPU

- É o dispositivo mais complexo da CPU;
- Além de possuir a lógica necessária para realizar a movimentação de dados e instruções de e para a CPU, esse dispositivo controla a ação da ULA;
- Os sinais de controle emitidos pela UC ocorrem em vários instantes durante o período de realização de um ciclo de instrução e, de modo geral, todos possuem uma duração fixa e igual, dada pelo clock;

CPU

- O conjunto de instruções que a CPU pode executar é um elemento crucial do desenho e na implementação dos circuitos da ULA;
- O conjunto de instruções utilizadas afeta não somente o projeto da seção de processamento: a estrutura e complexidade da unidade de controle é determinada diretamente pelas características do conjunto de instruções

CPU

- Como as instruções são criadas?
  - As instruções são geradas a partir dos comandos escritos pelo programador;
  - Uma soma seguida de uma atribuição devem ser convertidas em uma sequencia de instruções que fazem a busca dos dados em memória, a operação de soma e a cópia do resultado para a variável (posição de memória) correspondente;

CPU

- Como o programador sabe a sequencia de instruções que deve ser realizada pela CPU?
  - Não precisa saber. A conversão dos comandos de soma, atribuição, comparação e repetição, por exemplo, em instruções de máquina é feita pelo compilador, através de uma linguagem de alto-nível;

CPU

- Linguagem de alto nível
  - objetivo: tornar a comunicação com o computador mais simples e com menos instruções do que a linguagem de montagem mais distante da máquina
  - o programador não precisa se preocupar com o tipo de CPU ou de memória onde o programa será executado
  - Exs.: Fortran, Pascal, C

CPU

- Em geral, os programas são desenvolvidos em uma linguagem de alto nível, com Pascal, C, ou Java. O compilador traduz o programa de alto nível em uma sequência de instruções de processador
- Desta tradução resulta o programa em linguagem de montagem (*assembly language).*
- A linguagem de montagem é uma forma de representar textualmente as instruções oferecidas pela arquitetura, cada uma com uma linguagem de

CPU

- No programa em linguagem de montagem, as instruções são representadas através de “abreviações”, chamadas de mnemônicos, que associam o nome da instrução à sua função, como por exemplo:
  - **ADD: Adição**
  - **SUB: Subtração**
  - **MPY: Multiplicação**
  - **DIV: Divisão**
  - **LOAD: Carregar dados da memória**
  - **STOR: Armazenar dados na memória**

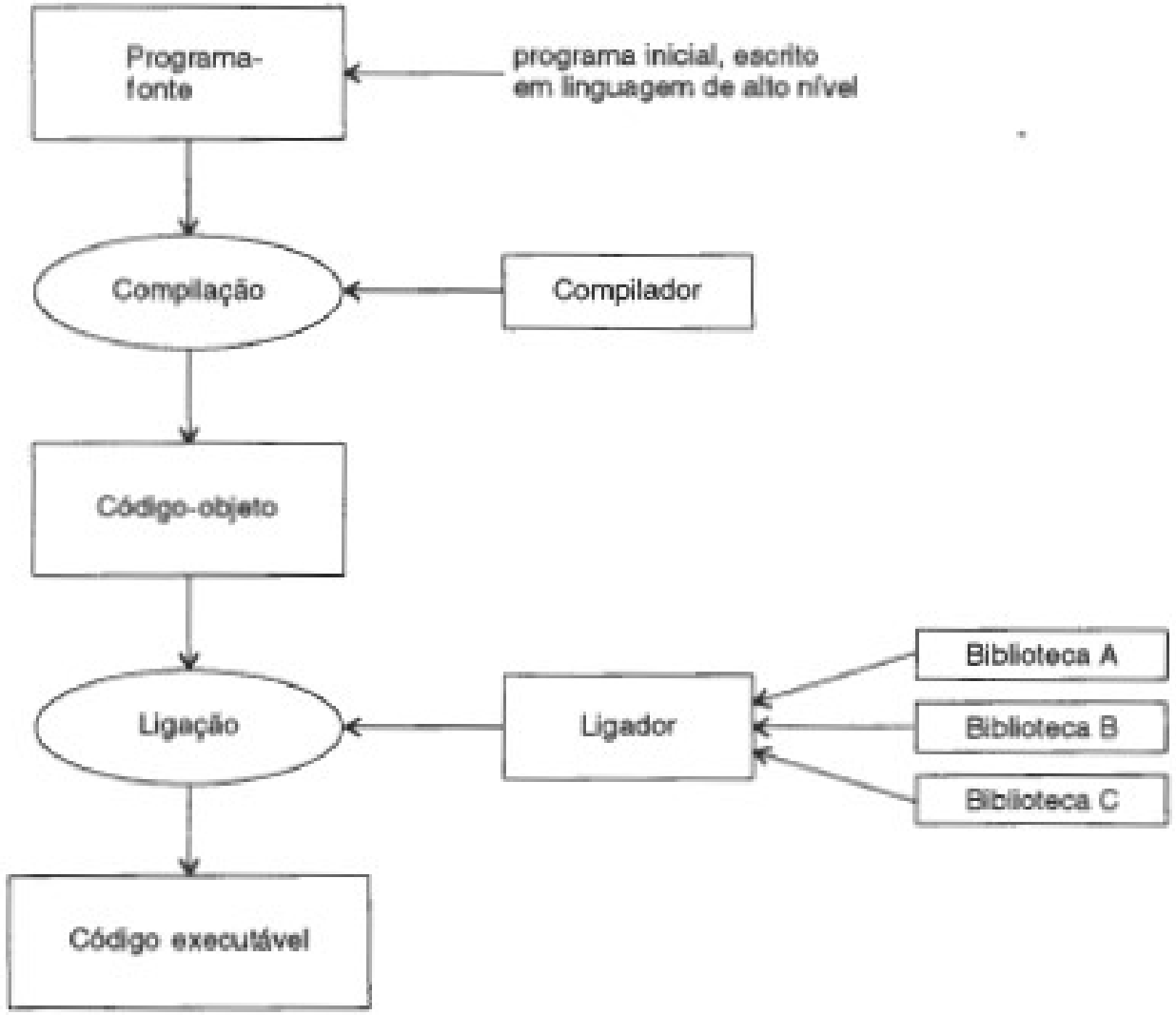
CPU

- O programa em linguagem de montagem é convertido para um programa em código objeto pelo montador (*assembler). O montador* traduz diretamente uma instrução da forma textual para a forma de código binário. É sob a forma binária que a instrução é carregada na memória e interpretada pelo processador

CPU

- Programas complexos são normalmente estruturados em módulos. Cada módulo é compilado separadamente e submetido ao montador, gerando diversos módulos em código objeto. Estes módulos são reunidos pelo ligador (*linker),* resultando finalmente no programa executável que é carregado na memória

CPU


Programa-fonte
programa inicial, escrito em linguagem de alto nível
Compilação
Compilador
Código-objeto
Ligação
Ligador
Biblioteca A
Biblioteca B
Biblioteca C
Código executável

CPU

- A compilação não é o único método de execução de programas;
- O método de interpretação consiste em realizar as três etapas (compilação, ligação, execução) de uma vez só, comando a comando;
- O código-fonte é analisado por um programa chamado de interpretador que realiza diretamente as operações e produz os resultados;

CPU

- Algumas linguagens apresentam características estruturais que são típicas da compilação como Fortran, Pascal, C;
- A linguagem Basic (hoje, Visual Basic) foi durante muito tempo apenas interpretada, mas atualmente existe processo de compilação para ela;
- A linguagem Java é interpretativa (JVM);

CPU

- Vantagens e desvantagens:
  - A principal vantagem da interpretação é a possibilidade de indicar erros já no código-fonte;
  - Uma desvantagem da interpretação é que certas partes do programa podem ser interpretadas tantas vezes quanto forem definidas nas estruturas de repetição;

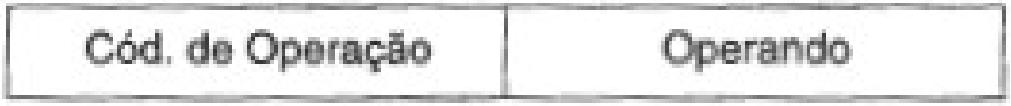
CPU

- Uma grande desvantagem é o consumo de memória:
  - O compilador só utiliza a memória enquanto ele mesmo estiver executando;
  - o interpretador consome memória o tempo todo em que o programa estiver executando;

CPU

- Formato das instruções:
  - Para que a CPU seja capaz de executar uma instrução, é preciso definir a quantidade de bits que será utilizada para representar a operação em si e a quantidade de bits que será utilizada para os operandos:

Instrução de máquina de um operando



Cód. de Operação - indica o tipo da operação a ser realizada
Operando - endereço do dado

CPU

- Toda instrução de máquina possui um código de operação específico e único para a tarefa que deseja executar;
- Este código, após ser decodificado durante o ciclo execução de instrução, permitirá que a UC envie os sinais necessários, e previamente programados, para se realizar a operação;

CPU

- Além do código de operação, cada instrução pode ter um conjunto de um ou mais operandos, cada um contendo um dado que deve ser utilizado durante a operação;

CPU

- Exemplos:
  - ADD A,B,X (X recebe A+B)
  - MUL R1,R2,R3 (R3 recebe R1*R2)
  - ADD A,B (A recebe A+B)
  - SUB R4,R2 (R4 recebe R4 – R2)
  - ADD A (Acc recebe Acc+A)
  - DIV R5 (Acc recebe Acc+R5)

CPU

- Modo de endereçamento:
  - Nem sempre e valor do operando é o que deve ser utilizado para a realização da operação;
  - Às vezes, o valor do operando significa onde o dado real se encontra na memória principal do computador;
  - É preciso definir qual o modo de endereçamento dos dados;

CPU

- Endereçamento imediato:
  - O valor do operando é exatamente o que deve ser utilizado pela CPU;
  - Ex: ADD 3
  - (Acc recebe Acc+3)
  - Vantagem: como o dado já está disponível, não é necessário buscá-lo na memória, diminuindo o tempo de processamento;
  - Desvantagem: a quantidade de bits geralmente será pequena, pois é necessário reservar alguns bits para o código de operação

CPU

- Endereçamento direto:
  - O valor do operando indica o endereço da memória onde está efetivamente o dado a ser operado;
  - Ex: MUL A,B
  - (célula A recebe o valor da célula A vezes o valor da célula B)
  - Vantagem: Pode utilizar dados de tamanho maior (tamanho da célula), além de ser rápido (apenas um acesso);
  - Desvantagem: pode representar uma quantidade pequena de memória

CPU

- Endereçamento indireto:
  - O valor do operando indica onde está na memória o endereço do dado real;
  - Ex: DIV A
  - (Acc recebe o valor da divisão de Acc pelo valor contido no endereço A da memória);
  - Vantagem: não há limitação tão restrita do espaço de memória;
  - Desvantagem: é preciso acessar a memória duas vezes;

CPU

- Existe como melhorar ainda mais o desempenho da CPU?
  - Será preciso analisar a forma como cada componente dela executa sua tarefa específica para poder dar a resposta;
  - Vamos pensar no funcionamento da CPU durante a execução de uma instrução...

CPU

- Nós vimos que para executar uma instrução, a CPU realiza diversas etapas;
  - Vai à memória (cache) buscar a instrução;
  - Decodifica a instrução;
  - Busca os parâmetros (operandos);
  - Realiza a instrução;
  - Salva o resultado;
  - Muda para próxima instrução;
  - etc

CPU

- Cada uma dessas etapas é realizada por um circuito (componente) distinto;
- O circuito que decodifica a instrução não busca os dados na memória, por exemplo;
- O que acontece com o circuito que decodifica a instrução enquanto os dados estão sendo buscados na memória?

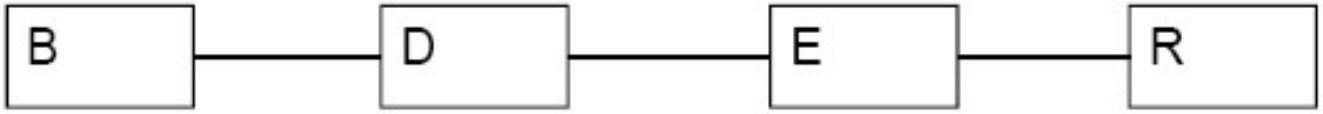
CPU

- O circuito poderia fazer outra coisa para “adiantar” trabalhos futuros;
- Antigamente, a CPU só começava a executar uma instrução quando a anterior tivesse sido completamente finalizada;
- Muitos circuitos ficavam ociosos....

CPU

- Na técnica de **pipeline**, permite-se que várias instruções sejam executadas simultaneamente, pois os passos da execução são realizados por unidades independentes, denominadas estágios do **Pipeline***;*
- O exemplo a seguir representa um pipeline de quatro estágios

CPU



Estágios:

B: busca
D:decodificação
E: execução
R: resultado

A execução de uma instrução inicia-se pelo estágio B, sendo completada no estágio R. Em condições normais, uma instrução avança para o estágio seguinte a cada novo ciclo de clock.

CPU

- No ciclo c1, a instrução i1 é buscada no estágio B
- No ciclo c2, a instrução i1 é decodificada no estágio D, enquanto o estágio B busca uma nova instrução, i2
- No ciclo c3, o estágio E executa a instrução i1, ao mesmo tempo que o estágio D decodifica a instrução i2 e o estágio B busca a instrução i3
- No ciclo c4, o resultado da instrução i1 é armazenado pelo estágio R, as instruções i2 e i3 avançam para o próximo estágio e o estágio B busca a instrução i4

CPU

- Novas instruções entram no pipeline antes que a execução das instruções anteriores seja completada
- Quando o pipeline encontra-se cheio, várias instruções estão sendo executadas em paralelo, uma em cada estágio do pipeline
- Na realidade, o aspecto mais importante na técnica de pipeline é que uma instrução seja completada em um ciclo de clock. Em uma arquitetura sequencial, a execução de uma instrução consome vários ciclos de clock, fazendo que o número médio de ciclos esteja bem acima da média de 1 ciclo/operação obtida com o uso da técnica de pipelining

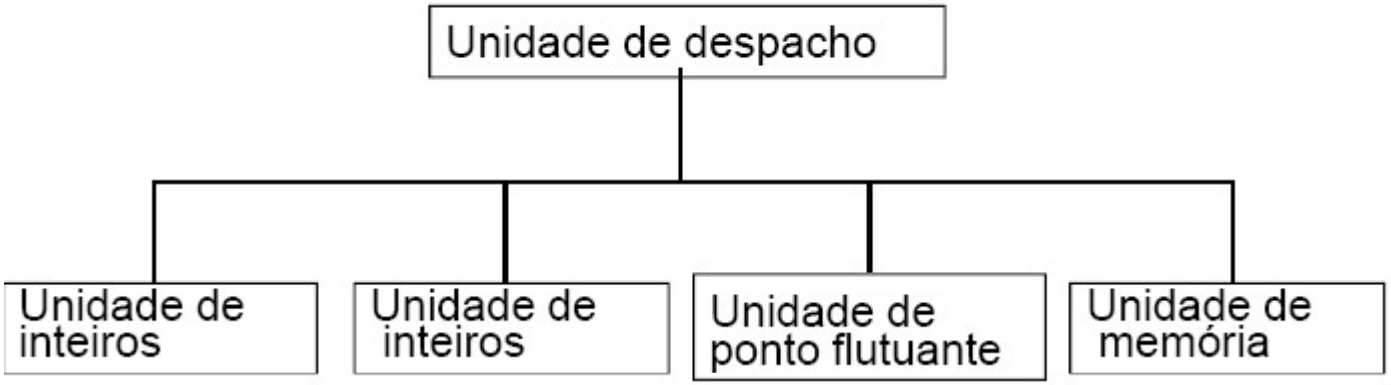
CPU

- Logo, as arquiteturas modernas voltadas para aplicações de alto desempenho utilizam a técnica de pipelining, sendo que suas instruções são implementadas de forma a apresentarem um fator de 1 ciclo/instrução
- Apesar de conceitualmente simples, o uso de pipelining encontra alguns problemas na prática.
- A técnica funciona se houver uma continuidade no fluxo de instruções, o que muitas vezes não acontece, por exemplo, quando existe uma dependência de dados entre duas instruções

CPU

- Com o *pipelining,* apenas uma instrução é completada por ciclo, resultando em um ipc máximo de 1 ciclo/instrução. Seria possível aumentar o desempenho de um sistema se o fator ipc fosse elevado para acima desta média
- Uma arquitetura super-escalar opera de forma que mais de uma instrução possa ser completada a cada ciclo, através de múltiplas unidades funcionais independentes, que executam instruções em paralelo
- A cada ciclo, múltiplas instruções podem ser enviadas (despachadas) para a execução nestas unidades funcionais

CPU



- Neste modelo teórico, até quatro instruções podem ser completadas por ciclo. A cada ciclo, a unidade de despacho busca e decodifica um certo número de instruções, e verifica quais destas instruções podem ser despachadas para as unidades funcionais

CPU

- Se hoje em dia as memórias ainda não estão no mesmo nível de velocidade do processador, o problema era ainda maior antigamente;
- Programas grandes demoravam muito porque muitas instruções precisavam ser buscadas na memória;
- Eram desejáveis programa pequenos;

CPU

- Linguagens de alto nível eram consideradas ineficientes em termos de espaço e tempo de execução, porém, a complexidade das aplicações foi tornando proibitivo o uso de programação *assembly*
- Este foi o motivo para o desenvolvimento de arquiteturas que suportariam o uso de linguagens de alto nível, sendo que sua execução seria tão eficiente quanto a de programas escritos em *assembly*

CPU

- Surgimento das arquiteturas para linguagens de alto nível (HLLC –*High-Level Language Computer architectures*)
  - Ex: arquitetura CISC (*Complex Instruction Set Computers*)
- O conjunto de instruções de diferentes arquiteturas baseiam-se no nível de funcionalidade; nas de alto nível, cada instrução realiza um grande número de operações

Alto nível de funcionalidade:

```
ADD A, B, C
```

Baixo nível de funcionalidade:

```
LOAD    A, R1
LOAD    B, R2
ADD     R1, R2, R3
STORE   R3, C
```

CPU

- Ambos os casos implementam a operação C = A + B, em diferentes níveis de funcionalidade
- Em uma linguagem de baixo nível de funcionalidade, são realizadas quatro operações: o carregamento das variáveis A e B em registradores, a adição propriamente dita e o armazenamento do resultado em C; em uma linguagem de alto nível de funcionalidade, os operandos são acessados diretamente na memória, é feita a adição e o resultado já é armazenado na memória

CPU

- Uma arquitetura com um pequeno conjunto de instruções com alto nível de funcionalidade apresenta dois benefícios:
  - Programas mais eficientes em termos de espaço ocupado na memória (menor número de instruções)
  - Comandos de alto nível com um menor número de instruções resultam em um menor número de acessos a memória para busca de instruções

CPU

- Na prática, contudo, não é bem assim:
  - Eficácia dos programas: em alguns casos, instruções complexas possuíam um tempo de execução elevado, até mesmo maior do que uma sequência de operações simples que realiza a mesma tarefa
  - Utilização de instruções: apenas uma pequena parcela das instruções oferecidas era realmente utilizada (ex.: IBM 370 tinha 183 instruções disponíveis; em 99% dos casos eram utilizadas apenas 48)
  - Efeito sobre o desempenho: o aumento do nível de funcionalidade possui um efeito negativo sobre o desempenho, por exemplo, é extremamente difícil implementar uma UC

CPU

- Abriu-se espaço para o surgimento de uma novo filosofia RISC (Reduced Instruction Set Computers):
  - Simplicidade das instruções;
  - Uso frequente na codificação de programas de alto-nível;
  - Novas tecnologias de fabricação das memórias, reduzindo o custo e o tempo de acesso;

CPU

- São características comuns a todas as arquiteturas RISC:
  - *Pipeline de instruções*
  - Arquitetura registrador-registrador: todos os operandos a serem utilizados em operações aritméticas e lógicas encontram-se em registradores, o que diminui a complexidade da unidade de controle
  - Regularidade no formato das instruções: todos os códigos possuem o mesmo tamanho, igual ao de uma palavra da memória, o que permite o acesso em um único ciclo a uma instrução completa

CPU

- Todos os modelos de sistemas computacionais que vimos são compostos por uma CPU e uma memória;
- Em sistemas paralelos, pode haver mais de um destes elementos;
- M. J. Flynn criou um esquema de classificação para estes sistemas;

CPU

- A classificação de Flynn baseia-se em dois fatores: o número de fluxos de instruções e de fluxos de dados existentes no sistema
  - Um fluxo de instrução é uma sequência de instruções endereçadas pelo contador de programa de um elemento processador. Se um sistema possui *n* contadores de programa, este sistema é capaz de processar *n* fluxos de instrução distintos
  - Um fluxo de dados corresponde a um conjunto de dados que é manipulado por um elemento processador

CPU

- SISD – *Single Instruction, Single Data Stream: um* único processador executa uma única sequência de instruções, utilizando dados armazenados em uma única memória.
- SIMD – *Single Instruction, Multiple Data Stream:* uma única instrução de máquina controla a execução simultânea de um certo número de elementos de processamento, cada elemento operando sobre um dado pertencente a um fluxo de dados diferente
  - Ex.: supercomputadores vetoriais, que realizam em paralelo uma mesma operação sobre múltiplos elementos de um vetor

CPU

- MISD – *Multiple Instruction, Single Data: uma* única sequência de dados é transmitida para um conjunto de processadores, cada um dos quais executa uma sequência de operações diferentes.
  - Na prática, ainda não existe nenhuma estrutura deste tipo implementada
- MIMD – *Multiple Instruction, Multiple Data: um* conjunto de processadores executa simultaneamente sequências diferentes de instruções, sobre conjuntos de dados distintos.
  - Sistemas de memória compartilhada ou distribuída (como clusters) são classificados