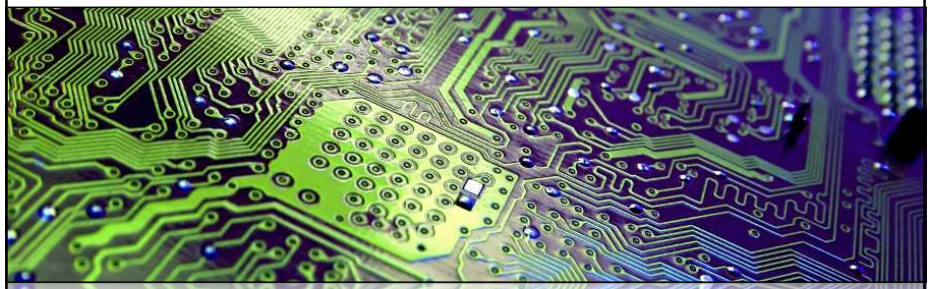
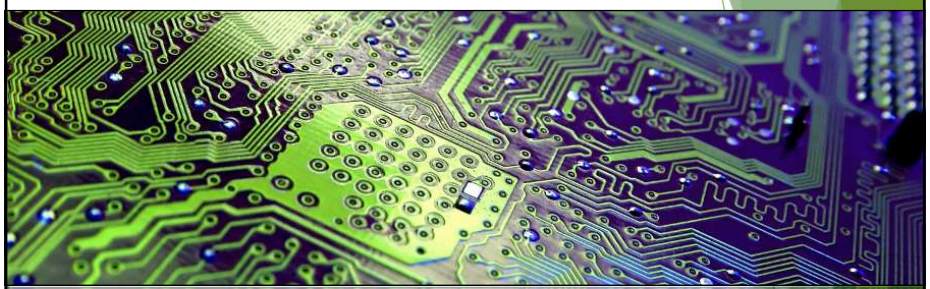


Organização e Arquitetura de Computadores

Começamos às 09:00, aguardem!

UNIFG
LAUREATE INTERNATIONAL UNIVERSITIES

1



Organização e Arquitetura de Computadores

Evolução dos Computadores RISC/CISC

Prof. Marcelo Rabello
marcelo.rabello@unifg.edu.br

UNIFG
LAUREATE INTERNATIONAL UNIVERSITIES

2

Objetivos de aprendizagem

Distinguir, comparar e aplicar as instruções

3

Processador

- ✓ Circuito integrado que realiza as funções de cálculo e tomada de decisão de um computador e é considerado o cérebro;
- ✓ Chamado de Unidade Central de Processamento (em inglês CPU: Central Processing Unit);
- ✓ Os processadores trabalham apenas com linguagem de máquina (lógica booleana). Realizam as seguintes tarefas:
  - ✓ ***Busca e execução de instruções existentes na memória.***
- ✓ Os programas e os dados que ficam gravados no disco (disco rígido), são transferidos para a memória. Uma vez na memória, o processador pode executar os programas e processar os dados;

4

Processador – Frequência de Operação

- ✓ O relógio do sistema (Clock) é um circuito oscilador a cristal (efeito piezoelétrico) que tem a função de sincronizar e ditar a medida de tempo de transferência de dados no computador. ***Esta frequência é medida em ciclos por segundo, ou Hertz.***
- ✓ A capacidade de processamento não está relacionada exclusivamente à frequência do relógio, mas também a outros fatores como: largura dos barramentos, quantidade de memória cache, arquitetura do processador, tecnologia de coprocessamento, tecnologia de previsão de saltos (branch prediction), tecnologia de pipeline, conjunto de instruções etc.
- ✓ O aumento da frequência de operação nominal do processador é denominado Overclocking.

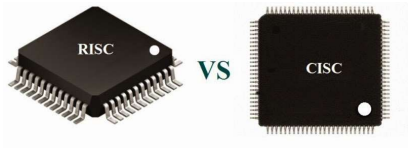
5

Parâmetros de desempenho do processador

- ❑ Clock – Frequência em (Hz);
- ❑ Fabricante e modelo do processador;
- ❑ Arquitetura de x86 ou x64;
- ❑ Núcleos do processador;
- ❑ Memória Cache;
- ❑ RISC ou CISC.

6

Arquitetura RISC e CISC



7

CISC – Complex Instruction Set Computing

- ✓ Usada em processadores **Intel e AMD**;
- ✓ Possui um grande conjunto de instruções (**tipicamente centenas**) que são armazenadas em uma pequena memória não-volátil interna ao processador. Cada posição desta memória contém as ***micro instruções***, ou seja, os passos a serem realizados para a execução de cada instrução.

8

CISC – Complex Instruction Set Computing

São fáceis de programar e permitem um uso **eficiente de memória**. A pouco tempo atrás as máquinas eram programadas única e exclusivamente em linguagem Assembly (**linguagem de máquina**), e as memórias eram lentas e caras, o que justificou a filosofia CISC. Assim, projetos de microprocessadores clássicos, tais como o **Intel 8086 e o Motorola 68K series**, seguiram a filosofia CISC.

9

CISC – Aspectos básicos

- ❑ Menos uso de microcódigo;
- ❑ Construção de conjuntos com instruções completas e eficientes;
- ❑ Criação de instruções de máquina de "**alto nível**", ou seja, com complexidade semelhante à dos comandos de alto nível.

10

CISC – Vantagens

Das vantagens existentes em se usar arquiteturas CISC podemos dizer que os ***programas ficam menores e mais simples***, porém há um ***custo de performance***. Outro ponto que **decrementa** a performance é um **baixo uso dos registradores e logo maior acesso externo ao processador**.

Na arquitetura CISC o uso da técnica de pipeline é de **moderado a nenhum**, tendo em vista que as ***instruções não têm tamanho fixo*** e, portanto, o tempo de execução não é homogêneo, logo o uso desta técnica se torna algo difícil.

11

RISC – Reduced Instruction Set Computing

Usada em processadores PowerPC (da Apple, Motorola e IBM) e SPARC (SUN); possui um ***conjunto pequeno de instruções*** (tipicamente algumas dezenas) implementadas diretamente em hardware;

- ✓ Nesta técnica não é necessário realizar a leitura em uma memória e, por isso, a execução das instruções é muito rápida (normalmente um ciclo de clock por instrução). Por outro lado, as instruções são muito simples e para a realização de certas tarefas são necessárias mais instruções que no modelo CISC.

12

RISC – Reduced Instruction Set Computing

- ✓ Arquiteturas RISC são do tipo **LOAD/STORE**, ou seja, as únicas operações de acesso a memória são de leitura e escrita. As operações lógicas e aritméticas são ***realizadas apenas entre registradores***. Isto é feito para que o acesso externo seja ***minimizado e portanto uma velocidade maior seja alcançada***, a aceleração será obtida pois as conexões internas ao processador tendem a ser mais rápidas, dado o distanciamento entre as partes e as frequências elevadas.

13

RISC – Reduced Instruction Set Computing

RISC tem como característica funcionar através de combinação de instruções simples para formar instruções complexas e por este motivo os **programas ficam maiores** dos que os escritos em arquitetura CISC, ocupando assim mais espaço na memória. Como as instruções são simples, de tamanho fixo e pequenas, elas são ***executadas frequentemente em um ciclo de clock***, portanto esta simplicidade traz a capacidade e a facilidade de fazer o uso intenso de pipeline.

14

CISC e RISC – Principais caraterísticas

- ✓ Um ponto que sempre será crucial para evolução da área da computação é o incremento de **velocidade na execução de uma instrução.**
- ✓ A compreensão da máquina multinível pode auxiliar no entendimento das arquiteturas RISC e CISC.
- ✓ Característica do hardware podemos ressaltar que as arquiteturas RISC admitem **frequências (clock) mais alta**, tendo em vista que arquitetura é mais simples e exige menos componentes de hardware e assim o aquecimento é menor, permitindo frequências maiores.

15

CISC e RISC - Comparativo

CISC	RISC
Único conjunto de registradores de uso geral, geralmente entre 6 e 16 registradores	Múltiplos conjuntos de registradores (pode superar 256)
Um ou dois operandos de registradores permitidos por instrução	Três operandos permitidos por instruções
Instruções de múltiplos ciclos	Instruções de um único ciclo (load e store)
Controle **micro programado**	Controle **diretamente no hardware**
Muitos instruções complexas	Instruções simples e em número reduzido
Instruções de tamanho variável	Instruções de **tamanho fixo**
Complexidade no **código**	Complexidade no **compilador**
Muitas instruções acessam a memória	**Apenas** instruções de **Load e Store** acessam a **memória**
Muitos modos de endereçamento	Poucos modos de endereçamento

16

Pipeline

É uma técnica de hardware que permite que a CPU realize a busca de uma ou mais instruções além da próxima a ser executada.
Estas instruções são colocadas em uma fila de memória dentro do processador (CPU) onde aguardam o momento de serem executadas: assim que uma instrução termina o primeiro estágio e parte para o segundo, a próxima instrução já ocupa o primeiro estágio.

A segmentação da instrução é utilizada para acelerar a velocidade de operações da CPU (*Throughput*). Uma vez que o processamento é realizado através de fases, é possível otimizar os recursos da CPU.

17

Pipeline

Até o 386, os processadores da família x86 eram capazes de processar apenas uma instrução de cada vez. Uma instrução simples podia ser executada em apenas um ciclo de clock, enquanto instruções mais complexas demoravam vários ciclos de clock para serem concluídas. Seria mais ou menos como montar um carro de maneira artesanal, peça por peça.

Para melhorar o desempenho do **486**, a Intel resolveu usar o pipeline, uma técnica inicialmente usada em processadores RISC, que consiste em dividir o processador em vários estágios distintos.

18

Pipeline

O 486, possui um pipeline de 5 níveis, ou seja, é dividido em 5 estágios.

IF (INSTRUCTION FETCH)	BUSCA DE INSTRUÇÃO NA MEMÓRIA.
ID (INSTRUCTION DECODE)	DECODIFICAÇÃO DE INSTRUÇÃO E LEITURA DE REGISTRADORES.
EX (EXECUÇÃO)	EXECUTA OPERAÇÃO OU CALCULA ENDEREÇO.
MEM (MEMÓRIA)	ACESSO O OPERANDO NA MEMÓRIA.
WB (ESCREVER)	ESCREVE RESULTADO NO REGISTRADOR.

19

Pipeline - Uso Geral

- Algumas CPUs incluem conceitos muito mais avançados de pipeline:
- Pré-decodificação: a CPU pode iniciar a decodificação de diversas instruções (paralelamente) e antes do momento das mesmas serem executadas.

20

Como aprimorar o desempenho?

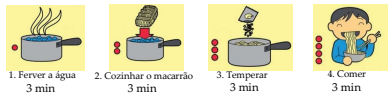
- Lei de Moore (1965): "*O número de transistores presentes nos chips dobra, pelo mesmo custo, a cada período de 18 meses*".
  - Ganho em desempenho (velocidade) baseado no desenvolvimento do *hardware*.
- *Multicore*: vários núcleos de processamento executando instruções simultaneamente.
- *Pipeline*: técnica que permite a execução paralela de instruções de máquina na mesma CPU.

21

Pipeline

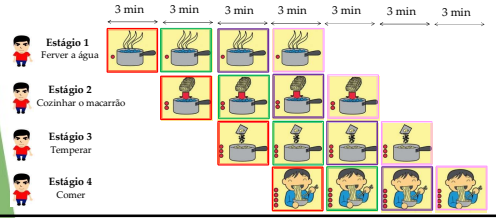
Inspiração → "Linha de produção"

Exemplo: Como fazer miojo?



TEMPO TOTAL
12 min

E se você precisasse fazer 4 miojos, quanto tempo levaria? Solução: Dividir em estágios!



TEMPO TOTAL
21 min
5,25 min/miojo

22

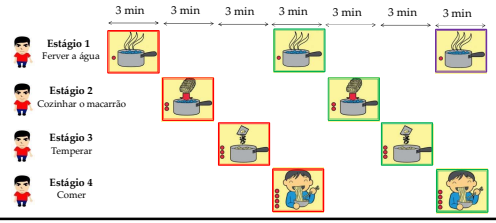
Pipeline

Inspiração → "Linha de produção"

E se eu tiver apenas uma panela?

HAZARD

"Risco" – Condições que impedem o *pipelining*



TEMPO TOTAL
39 min
9,75 min/miojo

23

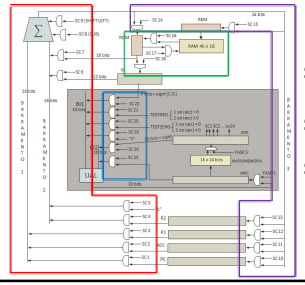
Pipeline: Definição e Requisitos

Definição: É o processo pelo qual uma instrução é subdividida em etapas, e cada uma destas etapas é executada por uma parte especializada da CPU, sendo possível colocar instruções em execução simultânea.

Requisitos:

1. Organização do *hardware*

Vejamos o EA869M:



SC 1 a 9: Estágio "Leitura dos registradores e operações na ULA"

SC 10 a 15: Estágio "Escrita nos registradores"

SC 16 a 18: Estágio "Acesso à memória"

SC 19 a 24: Estágio "End da próxima microinstrução"

O *hardware* precisa possuir partes especializadas (estágio) que executem separadamente etapas da instrução.

24

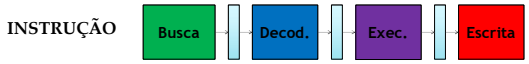
Pipeline - Requisitos

Requisitos:
2. A execução das **Instruções** deve ser organizada **em relação ao acesso a estes estágios.**

- O primeiro estágio deve sempre realizar a **busca** da instrução.
- O segundo estágio deve realizar a sua **decodificação.**
- O terceiro estágio deve realizar a sua **execução.**
- O estágio final deve realizar a **escrita** dos resultados.

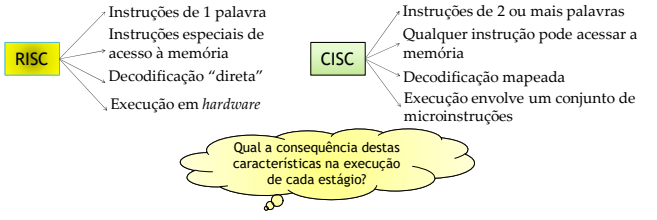
Nossa arquitetura deve, portanto, possuir ao menos 4 regiões especializadas, uma para cada estágio da instrução.

Entre cada estágio, é necessária a existência de um *buffer* para armazenar o conteúdo final e liberar a área para a próxima instrução.

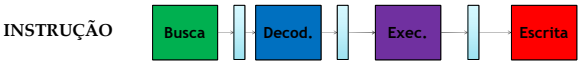


25

Pipeline



Qual a consequência destas características na execução de cada estágio?



RISC	Apenas um ciclo de acesso à memória	Do RI direto para execução	Instruções sempre levarão o mesmo tempo de execução	Com exceção das instruções de acesso a memória, envolve apenas escrita dos resultados em registradores
CISC	Pode haver mais de um ciclo de acesso à memória		Tempo depende da instrução e pode variar muito	Pode envolver a memória ou registradores em qualquer instrução

26

Pipeline

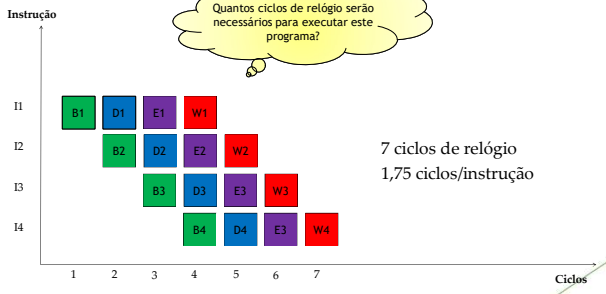
- Percebemos, portanto, que a ideia de *pipeline* combina naturalmente com as características de processadores RISC.
- Não obstante, processadores CISC, especialmente os da família Intel x86, também possuem execução baseada em *pipeline*.
- Porém, as instruções CISC demandam muitos estágios para executar a *pipeline*.
  - **Pentium 4:** 20 estágios de pipeline para execução completa de uma instrução.

27

Pipeline

- Vamos considerar um processador com *pipeline* de 4 estágios.
- Cada estágio é executado em 1 ciclo de relógio.
- Considere que desejemos executar uma sequência de 4 instruções.

Quantos ciclos de relógio serão necessários para executar este programa?

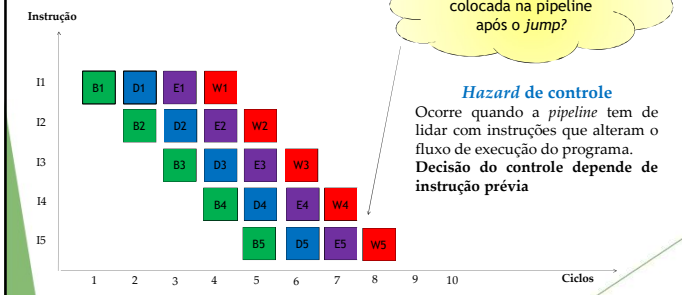


7 ciclos de relógio
1,75 ciclos/instrução

28

Pipeline

- Vamos considerar que um bloco de cinco instruções esteja pronto para ser executado. A quarta, em particular, se trata de um desvio incondicional (*jump*) para um endereço identificado pelo rótulo L.



Qual instrução deve ser colocada na pipeline após o *jump?*

Hazard de controle

Ocorre quando a *pipeline* tem de lidar com instruções que alteram o fluxo de execução do programa.
Decisão do controle depende de instrução prévia

29

Pipeline

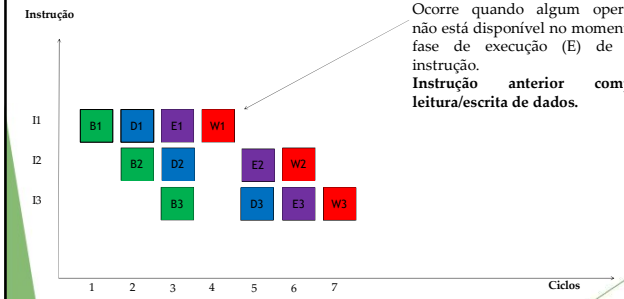
- ***Hazard* de controle:**
  - Surge por causa da necessidade de tomar uma decisão baseada em resultados de uma instrução enquanto outras estão em execução.
  - Ou seja, está ligado a instruções de desvio (*branches*).
  - **Problema:**
    - A *pipeline* inicia a busca da instrução subsequente ao desvio no próximo ciclo de relógio.
    - Porém, não há como a *pipeline* saber qual é a instrução correta a ser buscada, uma vez que acabou de receber o próprio desvio da memória.
    - Em outras palavras, a *pipeline* ainda não descobriu que se trata de um desvio, tampouco conhece seu resultado – se ele deve ser tomado ou não – e o eventual endereço de destino, mas precisa decidir a próxima instrução a ser lida.

30

Pipeline

- Vamos considerar agora que as seguintes instruções estejam na *cache*:

ADD R1, R2, R3
MOV R2, R1
ADD R4, R5, R6



Hazard de dados

Ocorre quando algum operando não está disponível no momento da fase de execução (E) de uma instrução.
Instrução anterior completa leitura/escrita de dados.

31

Pipeline

- ***Hazard* de dados:**
  - Ocorrem quando uma instrução depende da conclusão de uma instrução prévia que ainda esteja na *pipeline* para realizar sua operação e/ou acessar um dado.
  - **Exemplo:**

    add r0, r0, r1

    sub r2, r0, r3

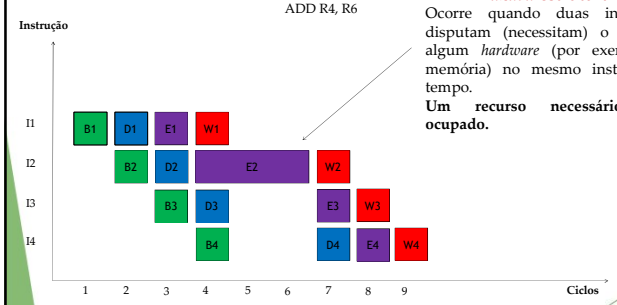
    - A instrução *add* somente escreve seu resultado no final do 4º estágio da *pipeline*.
    - Logo, teríamos que desperdiçar um ciclo de relógio aguardando até que o resultado correto (r0) pudesse ser lido pela instrução *sub*.

32

Pipeline

- Considere que a etapa de execução da instrução MUL precisa de 3 ciclos de relógio.
- As seguintes instruções estão carregadas na *cache*.

ADD R2, R1
MUL R5, R3
ADD R4, R6



***Hazard* estrutural**
Ocorre quando duas instruções disputam (necessitam) o uso de algum *hardware* (por exemplo, a memória) no mesmo instante de tempo.
Um recurso necessário esta ocupado.

33

Pipeline

- ***Hazard* estrutural:**
  - Situação de conflito pelo uso (simultâneo) de um mesmo recurso de *hardware*.
  - Ocorre quando duas instruções precisam utilizar o mesmo componente de *hardware* - para fins distintos ou com dados diferentes - no mesmo ciclo de relógio.
  - **Exemplo:** única memória sendo acessada no mesmo ciclo para finalidades diferentes (e.g., para busca de instrução e para carregamento de um valor em um registrador).

34

Conclusões

- *Pipeline* é um recurso amplamente utilizado na implementação de processadores e consiste em dividir a execução de uma instrução em estágios independentes, permitindo a exploração do paralelismo no nível de instruções.
- O uso de *pipeline* traz um ganho significativo de desempenho, particularmente no tocante à **taxa de execução de instruções por unidade de tempo**.
- **RISC:** instruções de 1 palavra + decodificação direta em *hardware* + tempo de execução de instrução com pouca variação => facilita a otimização da *pipeline*.
- **CISC:** demanda muitos estágios para a *pipeline*.
- **Desafios:** tratamento dos *hazards*.

35

Atividade Extraclasse

Ler o artigo **"RISC e CISC: Comparação de suas técnicas"**
http://ww2.deinfo.ufrpe.br/sites/ww2.deinfo.ufrpe.br/files/artigos_aoc/RISCxCISCv2.pdf

Processador CISC e RISC
https://www.youtube.com/watch?v=m1uKe6GdjOE

Organizações paralelas (pipeline) do processador
https://www.youtube.com/watch?v=tiVY7TAIiSI

36

Dúvidas? Sugestões?

37

"Back on my feet again
Eyes open to real world
Metropolis surrounds me
The mirror's shattered the girl

Why is this other life
Haunting me everyday
I`d break through the other side
If only I`d find the way"

https://www.youtube.com/watch?v=LWLrF1LpfqQ

Scenes From a Memory by Dream Theater

"Hey ho, let's go! Hey ho, let's go!
Hey ho, let's go! Hey ho, let's go!

They're forming in straight line
They're going through a tight wind
The kids are losing their minds
The blitzkrieg bop"

https://www.youtube.com/watch?v=iymtpePP8I8

The Blitzkrieg bop by Ramones

38