SISTEMAS COMPUTACIONAIS
E
SEGURANÇA

PROF. RICARDO SCHWACH
MSC, MBA, PMP, PSM, ITIL, COBIT

Sistemas Computacionais e Segurança

Arquiteturas RISC e CISC

Macro arquitetura

- O modelo que o programador enxerga da máquina é representado pelo conjunto de instruções e os recursos de hardware a ele associado.
- Historicamente este nível é desenvolvido antes dos demais. É a interface entre o hardware e o software.
- Programas em linguagens de nível superior são transformados em programas no formato ISA (instruction set architecture).
- Software executa no nível ISA.

Arquiteturas CISC e RISC

- As arquiteturas RISC e CISC possuem características muito importantes, que são determinantes para a velocidade e o desempenho das máquinas.

  - Faça a leitura indicada a seguir para compreender essas características, suas vantagens e desvantagens. Sugestão: faça anotações sobre as duas arquiteturas para facilitar o entendimento dos aspectos relevantes de cada uma.

Leia os tópicos 11.2 Características das arquiteturas CISC e 11.3 Características das arquiteturas RISC (p. 378-382), disponíveis no link:

https://integrada.minhabiblioteca.com.br/#/books/978-85-216-1973-4/cfi/390!/4/2@100:0.00

MONTEIRO, M. A. Introdução à organização de computadores. 5. ed. Rio de Janeiro: LTC, 2007.

Macro arquitetura

Nível ISA inclui:

- Instruction set
- Memória (modelo)
- Registradores acessíveis pelo programador

➢ Este nível define a Macro Arquitetura do sistema

Macro arquitetura

A Macro arquitetura define

- Set de instruções
- Formato de instruções
- Formato de memória
- Endereçamento de memória
- Acessos à memória
- Quantidade e uso de registradores

Macro arquitetura

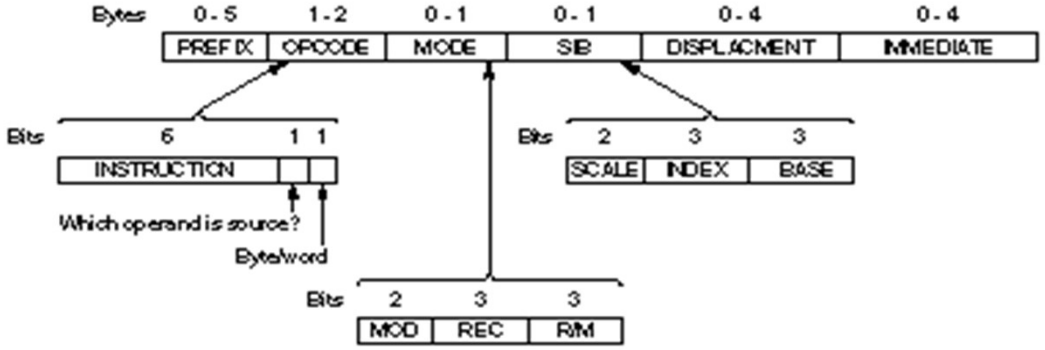
Exemplo: diferença de endereçamento de memória

Addressing mode	Pentium II	UltraSPARC II	JVM
Immediate	×	×	×
Direct	×		
Register	×	×	
Register indirect	×		
Indexed	×	×	×
Based-indexed		×	
Stack			×

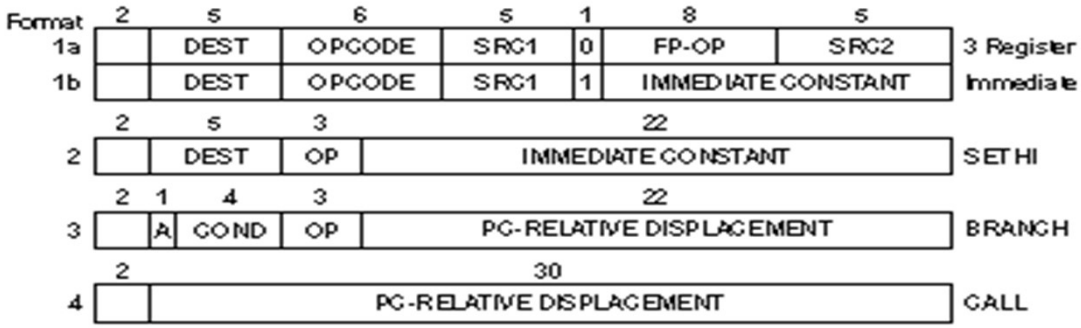
Macro arquitetura

- Exemplo: diferença de formato de instrução.

Pentium II



Ultra Sparc II



Instruções

- Cada arquitetura tem um conjunto de instruções, que são operações que o microprocessador consegue fazer
- Essas instruções comumente são do tipo:
  - Controle (habilitar ou desabilitar interrupções, NOP – no operation...)
  - Lógica (comparação, e, ou, vai-um...)
  - Ramificação (pular se zero, pular se ímpar, pular se vai-um, retornar se negativo...)
  - Aritmética (somar, subtrair, incrementar, decrementar, multiplicar...)
  - Transferência (copiar, mover, escrever endereços de memória ou registradores...)

Instruções

- Somar dois números inteiros de 8 bytes, armazenados no endereço 3005H e 3006H da memória
  - LXI H 3005H : "registrador HL aponta para 3005H"
  - MOV A, M : "Obter o primeiro número"
  - INX H : "HL aponta para 3006H"
  - ADD M : "Adicionar o segundo número"
  - INX H : "HL aponta para 3007H"
  - MOV M, A : "Guardar o resultado em 3007H"
  - HLT : "Sair do programa"

https://www.youtube.com/watch?v=dVHD2BKZklc

Conjunto de instruções

- **Quanto mais instruções**, **mais fácil** é escrever um compilador
- Primeiras arquiteturas x86 começaram então a adicionar um conjunto crescente de instruções a cada geração: CISC
- Nesta técnica há instruções de complexidade bastante variadas, ou seja, algumas podem demorar bem mais que outras para executar
- Quando as instruções tem tamanhos semelhantes, é mais fácil otimizar o processador, como no pipeline por exemplo

A técnica de pipelining permite que o processador tire proveito máximo de suas estruturas internas, dividindo o ciclo de processamento em tarefas menores.

Exemplos de Arquiteturas: x86, ARM

Existem duas arquiteturas dominantes no projeto de processadores, embora atualmente os processadores usarem um misto de ambas:

- Microprocessadores CISC (*Complex Instruction Set Computer*)
- Microprocessadores RISC (*Reduced Instruction Set Computer*)

- x86 é uma arquitetura CISC
- ARM é uma arquitetura RISC

Hoje em dia, a arquitetura x86 se chama AMD64 ou x86-64 e é um híbrido com instruções CISC e RISC

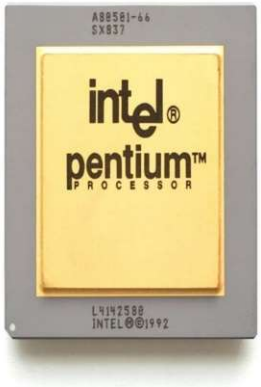
Exemplos de Arquiteturas: CISC

8086

80286

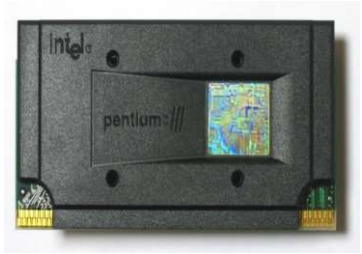
80386

80486



Pentium I

Pentium II



Pentium III

Arquiteturas: CISC

- Microprocessadores CISC (*Complex Instruction Set Computer*) são fáceis de programar (com mais instruções é mais fácil escrever compiladores)
- Permitem um uso eficiente de memória (programas são menores pois há mais instruções de mais alto nível)
- 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 80x86* e o *Motorola 68K series*, seguiram a filosofia CISC

Arquiteturas: CISC

- Mudanças recentes na tecnologia de software e hardware forçou uma reavaliação em termos de arquitetura.
- Assim, muitos processadores CISC mais modernos têm implementado alguns princípios RISC (*Reduced Instruction Set Computer*), tornando-se arquiteturas híbridas mais convenientes.
- No fundo, a ideia CISC ganhou força devido ao fato de ela permitir a existência de compiladores mais simples, uma vez que muitos comandos tem instruções equivalentes pelo próprio processador.

Atributos de CISC

- Possui um formato de 2 operandos, onde as instruções têm uma fonte e um destino, permitindo
  - Comandos de registrador-registrador, registrador-memória, e memória-registrador; e
  - Múltiplos modos de endereçamento para acesso à memória.
- Instruções de tamanho variável de acordo com o modo de endereçamento
- Instruções que requerem múltiplos ciclos de clock para executar

Características de Processadores CISC

- Possuem uma lógica de decodificação de instrução complexa, originada pela necessidade de suportar modos de endereçamento múltiplos
- Possuem um número pequeno de registradores de propósito geral, devido ao fato de as instruções poderem operar diretamente na memória, além de uma quantidade limitada de espaço em chip não dedicada
- Possuem muitos registradores de propósito específico tais como, apontadores de pilha, tratadores de interrupção etc.
- Possuem um registrador "Condition code" alterado de acordo com efeitos causados por algumas instruções (menor do que, é igual a, maior do que, e gravação de ocorrência de certas condições de erro)

Desvantagens de CISC

- Como as novas gerações de uma família de processador geralmente envolve a geração antecessora, tanto o conjunto de instruções quanto o hardware do novo chip tornam-se mais complexos
- Instruções diferentes levam quantidades diferentes de período clock para executar, o que pode tornar a máquina excessivamente lenta
- Instruções muito especializadas não são usadas com a frequência suficiente a ponto de justificar sua existência – aproximadamente 20% das instruções disponíveis são usadas em um programa típico (regra de Pareto)
- Instruções CISC típicas definem “condition codes”, o que demanda tempo de execução, além do fato de os programadores terem um esforço extra em lembrar de examiná-las

Micro programação em CISC

- Objetivo: otimizar o desempenho da máquina
- Micro programação é tão fácil de implementar quanto o Assembly, e muito menos caro do que hardwarizar uma unidade de controle
- A facilidade de novas instruções de microcódigos permitiu a projetistas tornar as versões mais recentes de máquinas CISC compatíveis com as mais antigas: um novo computador pode executar um mesmo programa executado em um computador antigo, pois o novo contém o super conjunto das instruções do antigo
- Como cada instrução torna-se mais capaz, menos instruções podem ser usadas para implementar uma dada tarefa
- Conjuntos de instruções de micro programas podem ser escritos de modo a compartilhar os alicerces de linguagens de alto nível, reduzindo a complexidade de compiladores

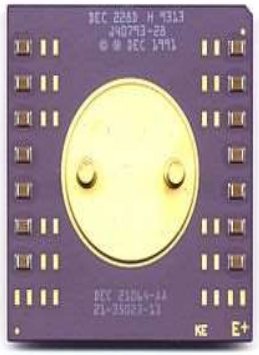
Arquitetura RISC

- Microprocessadores **RISC** (*Reduced Instruction Set Computer*) são aqueles que **utilizam um pequeno conjunto de instruções altamente otimizado**.
  - Poucas instruções
  - Instruções **mais simples e rápidas**
  - Facilitam a otimização das etapas do processamento

- Os primeiros projetos RISC foram desenvolvidos nos anos 70 e 80 pelas universidades de Stanford e Berkeley, respectivamente.

Arquitetura RISC

PowerPC
Apple



DEC Alpha

SPARC

ARM
Celulares
PDA

Arquitetura RISC

Algumas características RISC importantes são:

- **Execução em um ciclo de *clock***. Esta característica é resultado da otimização de cada instrução, aliada a uma técnica chamada de *Pipelining* - ***Pipelining* é uma técnica que permite execução simultânea de partes**, ou estágios, de instruções, tornando o processo mais eficiente;
- **Grande número de registradores** para evitar uma quantidade elevada de interações com a memória.
- Conjunto reduzido de instruções
- **Instruções menos complexas**
- Unidade de controle (responsável pela busca (fetch), decodificação e execução das instruções) “hardwarizada” (sem micro programação, como uma máquina de estados finita)
- Baixa capacidade de endereçamento para operações de memória, com apenas duas instruções básicas, LOAD e STORE

Uso de um grande banco de Registradores:

- O Banco de Registradores atua como uma Memória Cache, armazenando de forma temporária e de acesso rápido às informações.
- Surge portanto a questão de saber se não seria melhor e mais simples utilizar uma Memória Cache e o pequeno conjunto tradicional de registradores.

Uso de um grande banco de Registradores:

A **importância de se usar registradores** é que entre os dispositivos de armazenamento disponíveis, os registradores constituem os dispositivos que oferecem **acesso mais rápido que a memória principal ou a memória cache**.

Um banco de registradores é fisicamente pequeno, fica contido na Unidade Lógica e Aritmética (ULA) e na Unidade de Controle e usa endereços de tamanho muito menor que endereços da memória cache e memória principal. Os registradores armazenam os operandos mais utilizados minimizando operações de transferência de dados entre eles e a memória.

Uso de um grande banco de Registradores:

A escolha entre Banco de Registradores ou Memória Cache não parece óbvio. Entretanto existe uma característica em que a abordagem de banco de registradores é claramente superior do que a memória cache que é em **relação ao mecanismo de endereçamento**.

Portanto para melhora do desempenho o uso de bancos de registradores baseados em janelas é superior ao uso de cache para armazenar variáveis escalares locais. **Um desempenho maior pode ser obtido pela adição de uma cache apenas para instruções**.

Desvantagens da Arquitetura RISC

- É fato que máquinas RISC são mais baratas e mais rápidas do que as CISC, o que pode nos **induzir a pensar que elas são as máquinas do futuro**.

- Entretanto, **o custo de um hardware mais simples é a necessidade de um software mais complexo**.

- Isto é bom ou ruim?

Comparação entre as arquiteturas CISC e RISC

O quadro a seguir exibe uma comparação dos recursos das duas arquiteturas:

RISC	CISC
Instruções simples e em número reduzido	Muitas instruções complexas
Altamente paralelizado (recurso *pipeline*)	Fracamente paralelizado
Passagem eficiente de parâmetros por registradores no chip (processador)	Passagem de parâmetros ineficiente por meio da memória
Instruções de tamanho fixo	Instruções de tamanho variável
Complexidade no compilador	Complexidade no código
Poucos modos de endereçamento	Muitos modos de endereçamento
Apenas as instruções *load* e store podem acessar a memória	Muitas instruções podem acessar a memória
Instruções de um único ciclo	Instruções de múltiplos ciclos
Controle embutido no hardware	Controle microprogramado

Comparação RISC / CISC

Características	CISC	RISC
Arquitetura	Registrador-Memória	Registrador-Registrador
Tipos de Dados	Muito variada	Pouca variedade
Tamanho das Instruções	Instruções com muitos endereços	Instruções poucos endereços
Modo de Endereçamento	Muita variedade	Pouca variedade
Estágios de Pipeline	Entre 20 e 30	Entre 4 e 10
Acesso aos dados	Via memória	Via registradores

CISC x RISC

- Máquinas **RISC conseguem maior MIPS** (“processamento”) que máquinas CISC
- Para instruções de **ponto flutuante, as máquinas RISC necessitam de hardware especial** para terem desempenho equivalente às CISC.
- Múltiplos conjuntos de registradores das RISC contribuem para um maior desempenho.
- Alguns programas criados em linguagens de alto nível precisam de uma biblioteca de procedimentos para rodar eficientemente em máquinas RISC, o que pode ser realizado via microcódigo nas CISC.
- As arquiteturas **RISC são mais fáceis de produzir** devido à sua simplicidade e menor número de transistores.
- Compiladores para RISCs são mais complexos pois precisam usar eficientemente os recursos de pipeline e de alocação de registradores.

CISC x RISC

RISC	CISC
Instruções simples durando 1 ciclo	Instruções complexas durando vários ciclos
Apenas LOAD/STORE referencia a memória	Qualquer instrução pode referenciar a memória
Alto uso de Pipeline	Baixo uso de Pipeline
Instruções executadas pelo hardware (*Hardwired*)	Instruções interpretadas pelo microprograma (Microprogramação)
Instruções com formato fixo	Instruções de vários formatos
Poucas instruções e modos de endereçamento	Muitas instruções e modos de endereçamento
Múltiplos conjuntos de registradores	Conjunto único de registradores
A complexidade está no compilador	A complexidade está no microprograma

Pipelining

A técnica de pipelining permite que o processador tire proveito máximo de suas estruturas internas, dividindo o ciclo de processamento em tarefas menores.

A execução destas tarefas em sequência é que permitem o melhor aproveitamento do hardware.

Para que ocorra pipelining é necessário:

- Instruções com tempo de execução próximos e previsíveis (**execução em um único ciclo**)
- Baixo número de instruções.

Pipelining

- Melhora o throughput (taxa de transferência), não a latência
- Várias tarefas **operando em paralel**o utilizando recursos diversos
- **Saída do pipeline é limitada pelo estágio mais lento**
- **Desequilíbrios na duração dos estágios** reduzem a aceleração
- Tempo para encher e esvaziar podem reduzir a aceleração
- **Pode parar por dependências** entre as tarefas

Superescalar

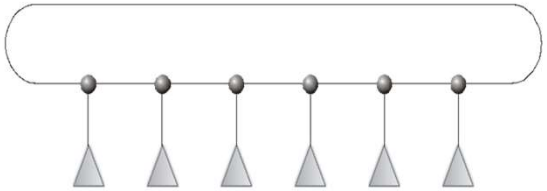
- Processador com vários pipelines
- Necessita alguns componentes especiais, como:
  - **Unidade de Busca de Instruções**: capaz de buscar mais de uma instrução por ciclo.
    - Possui também um preditor de desvios, que deve ter alta taxa de acerto, para poder buscar as instruções sem ter que esperar pelo resultados dos desvios.
  - **Unidade de Decodificação**: capaz de ler vários operandos do banco de registradores a cada ciclo.
  - **Unidades Funcionais Inteiras e de Ponto Flutuante**: em número suficiente para executar as diversas instruções buscadas e decodificadas a cada ciclo.

Arquiteturas paralelas

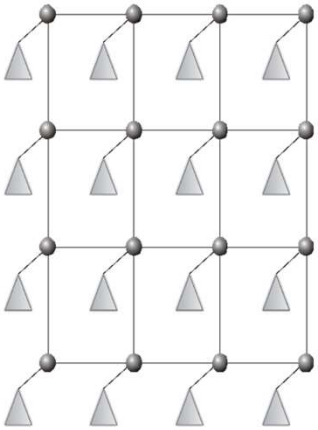
- Classificação quanto ao grau de paralelismo:
  - **SISD** (Single Instruction Single Data) - suportam **uma única** sequência de instruções e apenas **uma sequência** de dados.
  - **SIMD** (Single Instruction Multiple Data) - suportam **uma única** sequência de instruções e **múltiplas sequências** de dados.
  - **MISD** (Multiple Instruction Single Data) - suportam **múltiplas** sequências de instruções e **uma única** sequência de dados.
  - **MIMD** (Multiple Instruction Multiple Data) - suportam **múltiplas sequências** de instruções e **múltiplas sequências** de dados.
- A grande maioria dos computadores com um único processador pertencem a arquitetura **SISD**, enquanto os sistemas com múltiplos processadores pertencem a arquitetura **MIMD.**

Arquiteturas paralelas

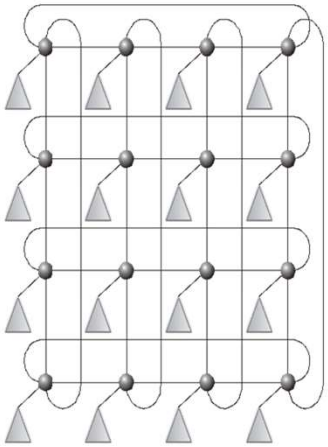
Formas de se interligar processadores:



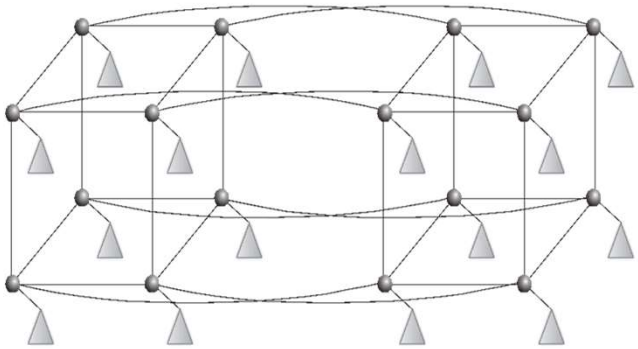
(b) Anel



(c) Grid 2D



(d) Torus 2D



(e) Hipercubo

Obrigado