

USP - ICMC - SSC
SSC 0510 - Informática - 2o. Semestre 2009

Disciplina de **Arquitetura de Computadores**

Prof. Fernando Santos Osório

Email: fosorio [at] { icmc. usp. br , gmail. com }

Página Pessoal: <http://www.icmc.usp.br/~fosorio/>

Estagiário PAE **Maurício Dias** - Email: acdias29 [at] yahoo.com.br

Material on-line: COTEIA - <http://coteia.icmc.usp.br>

Aula 08 – Microprocessadores RISC x CISC

Conteúdos Abordados:

- 1. Microprocessador Comerciais:**
6502, 8080, Z80, 68000, 8086, 80x86
- 2. Microprocessadores CISC - Conceito**
- 3. Microprocessadores RISC – Conceito**
- 4. Motivação da criação de Arquiteturas RISC**
- 5. Microprocessadores RISC**

Microprocessador Comerciais “Tradicionais”

Arquiteturas Tradicionais

Microprocessador Comerciais:

6502, 8080, Z80, 68000, 8086, 80x86

MICROPROCESSADOR	Nro. de Instruções (Instruction Set)
Intel 4004	46 instruções
MosTech 6502	151 instruções (total)
Intel 8080	78 instruções diferentes (total ??)
Zilog Z80	158 instruções diferentes (~ 700 total?)
Motorola 68000	56 instruções básica => ?? total
Intel 8086	400 opcodes (?)
Intel Pentium	+ de 400 (?)

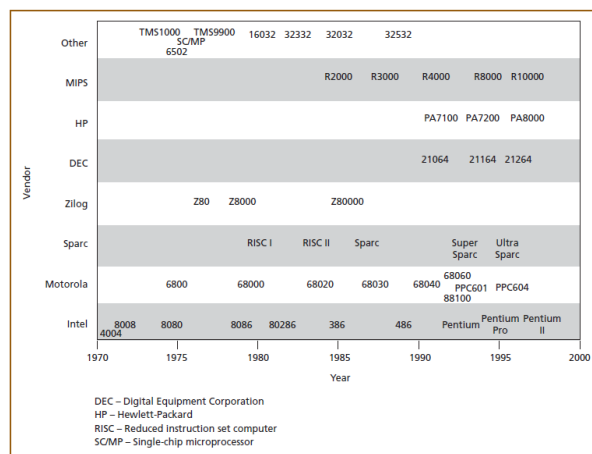
Microprocessador Comerciais

Arquiteturas Tradicionais

Microprocessador Comerciais:

**6502,
8080,
Z80,
68000,
8086,
80x86**

...



Microprocessador Comerciais

Arquiteturas Tradicionais

Microprocessador Comerciais:

6502,

8080,

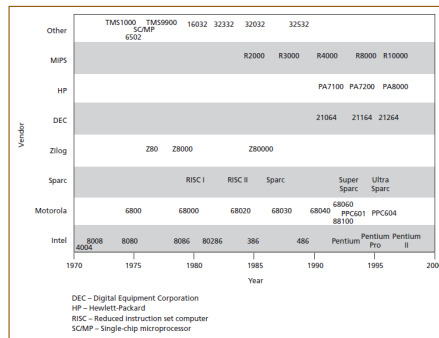
Z80,

68000,

8086,

80x86

...



Arquiteturas RISC: SPARC, MIPS, DEC Alpha

RISC = Reduced Instructions Set Computer

Microprocessador Comerciais

Arquiteturas Tradicionais

Table I. Microprocessor features.

Microprocessor	Date of Introduction	Clock speed (MHz)	Architectural width	Addressable memory	Features
Intel 8086	6/78	4.77/8	16-bit	1 MB, segmented	16-bit successor to 8080/8085
Intel 8088	6/79	5/8	16-bit, 8-bit external	1 MB, segmented	CPU for IBM PC
Intel 80286	2/82	8/10/12	16-bit	16 MB, protected mode	Virtual memory
Intel iAPX432	1980/1983	8	32-bit	1 TB, segmented	Object oriented
Motorola 68000	9/79	4 to 12.5	32-bit, 16-bit external	16 MB, linear	First with 32-bit programmer's view
Motorola 68010	3/82	4 to 12.5	32-bit, 16-bit external	16 MB, linear	Virtual memory
Motorola 68020	3/84	16.67	32-bit	4 GB	3-stage pipeline, instruction cache
Zilog Z8000	1979	4	16-bit	8 MB, segmented	Incompatible successor to Z80
UC Berkeley RISC V/II	1980/1982	8/12	32-bit	4 GB	First RISC microprocessors
Stanford MIPS	1981	8	32-bit	4 GB	Advanced compiler techniques

RISC - Reduced Instruction set computer

Arquiteturas Complexas x Simples

Processor Classifications:

Arbitrarily assigned by me...

Complex/ CISC	Simple/ RISC
4-bit	14500B*
	*Am2901
8-bit	
	*4004
	*4040
	6800, 650x
	*1802
	8051* *8008 * SC/MP
	Z8 *
	*F8
	F100-L* 8080/5 2650
	*NOVA
	*PIC16x
16-bit	
	MCP1600* *Z-80
	*6809
	IMS6100
	80C166* *M17
	*PDP11
	*8086
	*TMS9900
	*Z8000
	*65816
	*56002
32-bit	
	32016* *68000 ACE HOBBIT Clipper R3000
	[3] 96002 *68020 * * * *29000 * *ARM
	*VAX * 80486 68040 *PSC i960 *SPARC *SH
	Z80000* * * TRON48 PA-RISC
	PPro Pent* [1]---*-----
	* * [2]---<860>---*-----
64-bit	
	Rekurs POWER PowerPC * CDC6600 *R4000
	x86-64* *620 U-SPARC * *R8000 *Alpha
	R10000

[1] - About here, from left to right, the Swordfish and 68060.

[2] - In general, Pentium emulator 'clones' such as the 586, AMD K5, and Cyrix M1 fit about here.

Arquiteturas Complexas x Simples

SPARC (from Scalable Processor Architecture) is a RISC microprocessor instruction set architecture originally designed in 1985 by **Sun Microsystems**.

Sun Microsystems: WorkStations Sun Sparc
Sun Java (+ recentemente)



Sun UltraSPARC II
Microprocessor

The SPARC architecture was heavily influenced by the earlier RISC designs including the RISC I & II from the University of California, Berkeley and the IBM 801. These original RISC designs were minimalist, including as few features or op-codes as possible and aiming to execute instructions at a rate of almost one instruction per clock cycle.

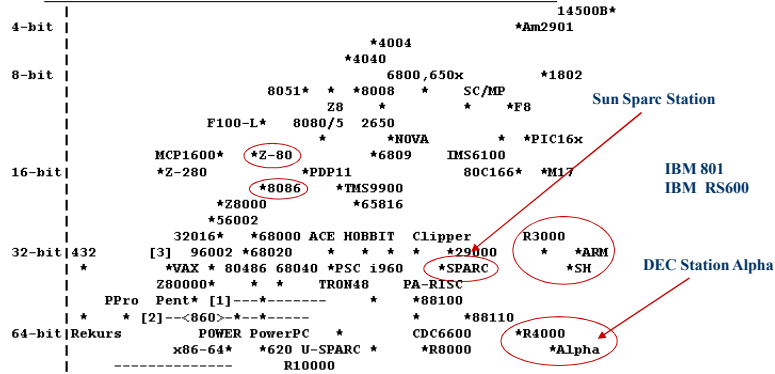
Alpha (AXP), was a 64-bit reduced instruction set computer (RISC) developed by Digital Equipment Corporation (DEC). It was designed to replace the 32-bit VAX ISA and processors. Alpha was implemented in microprocessors originally developed and fabricated by DEC.

MIPS (acronym for Microprocessor without Interlocked Pipeline Stages) is a RISC.

Arquiteturas Complexas x Simples

Processor Classifications:

Arbitrarily assigned by me...



[1] - About here, from left to right, the Swordfish and 68060.

[2] - In general, Pentium emulator 'clones' such as the 586, AMD K5, and Cyrix M1 fit about here.

O que é RISC?

The History of the Microprocessor
Michael R. Betker et al.

- RISC significa **Reduced Instruction Set Computer**, ou seja, um computador com um conjunto de instruções reduzido

“The Beginning of the RISC Argument In the early 1980s, the stage was being set in academia for the next phase of microprocessor evolution. Projects at the University of California at Berkeley and Stanford University in nearby Palo Alto were developing RISC microprocessors. Although the 8086/8088 and 68000 were well established with significant desktop bases, the field of computer architecture was much wider and older than microprocessors alone.

The RISC movement began in reaction to the complexity of a minicomputer architecture, the VAX* from Digital Equipment Corporation (DEC).

The basic tenets of RISC were evident in earlier non-microprocessor architectures such as the IBM 801 by John Cocke and Control Data's 6600 by Seymour Cray.

Unlike contemporary and earlier complex instruction set computer (CISC) processors, the RISC projects endorsed fixed-length, 32-bit instructions, no memory-to-memory instructions (RISC used a load/store architecture), large, general-purpose register files, and pipelining.

Professor David Patterson's project at Berkeley coined the term “RISC” with the RISC I microprocessor.”

O que é RISC?

The History of the Microprocessor
Michael R. Betker et al.

- RISC significa *Reduced Instruction Set Computer*, ou seja, um computador com um conjunto de instruções reduzido

“The Beginning of the RISC... In particular, the RISC projects formalized a fundamental performance metric for computer architectures, namely the amount of CPU time required to execute a given task. This was expressed by the equation:

CPU time =

Instruction count x Clock cycles per instruction (CPI) x Clock cycle time.

A typical CISC had three or four, while RISCs approached the goal of achieving one cycle per instruction. (...)

The claim of RISC's superiority over CISC, outlined by Berkeley RISC and Stanford MIPS, led to the first commercial RISC CPUs in the second half of the 1980s. The workstation manufacturers abandoned Motorola 68K CPUs in favor of their own RISC CPUs. The first commercial RISC CPU, the MIPS* R2000, was based on the Stanford MIPS and was introduced in 1986.”

O que é RISC?

- RISC significa *Reduced Instruction Set Computer*, ou seja, um computador com um conjunto de instruções reduzido

RISC I: 31 instructions

RISC II: 39 instructions

SPARC: 69 instructions

MIPS 4000: 94 Instructions, 4 byte instruction size

Characteristic	Complex Instruction Set (CISC) Computer			Reduced Instruction Set (RISC) Computer		Superscalar		
	IBM 370/168	VAX 11/780	Intel 80486	SPARC	MIPS R4000	PowerPC	Ultra SPARC	MIPS R10000
Year developed	1973	1978	1989	1987	1991	1993	1996	1996
Number of instructions	208	303	235	69	94	225		
Instruction size (bytes)	2-6	2-57	1-11	4	4	4	4	4
Addressing modes	4	22	11	1	1	2	1	1
Number of general-purpose registers	16	16	8	40 - 520	32	32	40 - 520	32
Control memory size (Kbits)	420	480	246	—	—	—	—	—
Cache size (KBytes)	64	64	8	32	128	16-32	32	64

[Stallings]

O que é CISC?

- CISC significa *Complex Instruction Set Computer*, ou seja, um computador com um conjunto de instruções complexo

Os processadores comerciais “tradicionais” (Intel, Motorola, Zilog) vinham crescendo em complexidade e recursos oferecidos:

- Diversos modos de endereçamento;
- Registradores de uso geral com vários modos de acesso;
- Uso de micro-instruções complexas (vários ciclos de clock);

As perguntas que surgiram nesta época eram:

- Tudo isto é realmente necessário?
- Tudo isto é realmente usado?
- Quanto mais complexo um processador (decodificação, execução), ele não fica mais difícil de gerenciar e por consequência mais lento?

RISC x CISC

CISC	RISC
Large instruction set	Compact instruction set
Complex, powerful instructions	Simple hard-wired machine code and control unit
Instruction sub-commands micro-coded in on board ROM	Pipelining of instructions
Compact and versatile register set	Numerous registers
Numerous memory addressing options for operands	Compiler and IC developed simultaneously

Discussão... Programando em Assembly

Em seu trabalho de implementação junto ao 6502:

- Quantas instruções diferentes você usou?
- Quantos modos de endereçamento diferente você usou?

Se você tivesse que programar um 80x86 com centenas de opcodes, você acredita que faria muita diferença?

RISC x CISC

CISC	RISC
Large instruction set	Compact instruction set
Complex, powerful instructions	Simple hard-wired machine code and control unit
Instruction sub-commands micro-coded in on board ROM	Pipelining of instructions
Compact and versatile register set	Numerous registers
Numerous memory addressing options for operands	Compiler and IC developed simultaneously

Discussão... Programando em Linguagens de Alto Nível

Quando compilamos um programa em C ou Pascal (em relação ao compilador):

- Quantas instruções diferentes ele realmente gera no código objeto?
- Quantos modos de endereçamento diferente ele usa?
- Será que todas as instruções são usadas sempre (com a mesma frequência)?
- Será que não é possível implementar as instruções “mais raras” através das “mais comuns”?

RISC x CISC

CISC	RISC
Large instruction set	Compact instruction set
Complex, powerful instructions	Simple hard-wired machine code and control unit
Instruction sub-commands micro-coded in on board ROM	Pipelining of instructions
Compact and versatile register set	Numerous registers
Numerous memory addressing options for operands	Compiler and IC developed simultaneously

Simples: Faça uma estatística das instruções que aparecem no programa!

Discussão... Programando em Linguagens de Alto Nível

Quando compilamos um programa em C ou Pascal (em relação ao compilador):

- Quantas instruções diferentes ele realmente gera no código objeto?
- Quantos modos de endereçamento diferente ele usa?
- Será que todas as instruções são usadas sempre (com a mesma frequência)?
- Será que não é possível implementar as instruções “mais raras” através das “mais comuns”?

RISC x CISC

CISC – *Complex Instruction Set Computer*

- Computadores complexos devido a:
 - Instruções complexas que demandam um número grande de ciclos para serem executadas
 - Dezenas de modos de endereçamento
 - Instruções de tamanhos variados
 - Referência a operandos na memória principal
- Questionamentos quanto à necessidade de certas instruções
 - Levantamentos mostram as instruções mais utilizadas nos programas

RISC x CISC

- Estudos de Knuth, Wortman, Tanenbaum e Patterson em várias linguagens com relação a porcentagem de comandos nos processadores

Comando	Fortran	C	Pascal
atribuicao :=	51%	38%	45%
if	10	43	29
call	5	12	15
loop	9	3	5
goto	9	3	0
outros	16	1	6

RISC x CISC

> Portanto:

- Nas arquiteturas CISC fica mais difícil implementar o pipeline
- A taxa média de execução das instruções por ciclo tende a ser bem menor do que 1 IPC
- A unidade de controle é *microprogramada*
- Códigos compactados podem ser gerados pelos compiladores
- Instruções complexas significa um maior tempo para decodificar e executar, muitas das quais são *raramente usadas*

> Surgem então as arquiteturas RISC

RISC

RISC – *Reduced Instruction Set Computer*

Características:

- Instruções mais simples, demandando um número fixo de ciclos de máquinas para sua execução
- Uso de poucos e simples modos de endereçamento
- Poucos formatos das instruções
- Apenas instruções de load/store referenciam operandos na memória principal
- Cada fase de processamento da instrução tem a duração fixa igual a um ciclo de máquina

RISC

> Portanto:

- Implementadas com o uso do pipeline
 - Formato fixo das instruções facilita o pipeline
- As instruções são executadas na sua maioria em apenas um ciclo de máquina
- A unidade de controle é em geral *hardwired*
 - Não há microprograma para interpretar as instruções
- Arquitetura orientada a registrador
 - Todas as operações aritméticas são realizadas entre registradores
 - Define-se um grande conjunto de registradores
- Processo de compilação é complexo e requer cuidados especiais para otimização do desempenho do código gerado
 - Somente o essencial é implementado na pastilha

RISC

- Primeiros computadores RISC:
 - IBM 801 (1980)
 - É o antecessor do IBM PC/RT (RISC Technology)
 - Seguido depois pelo IBM RS/6000 e do Power Processor
 - Berkeley RISC I e RISC II (1980 e 1981)
 - Projetado por Patterson e Séquin
 - Inspirou o projeto do processador SPARC, da SUN Microsystem
 - Stanford MIPS (1981)
 - Projetado por Hennessy
 - Originou a MIPS Computer Systems

Arquitetura RISC

Uso de um grande número de registradores

- > Deve-se adotar uma estratégia de modo a garantir que os operandos usados mais freqüentemente permaneçam em registradores, pois existe:
 - Grande proporção de comandos de atribuição (movimentação de dados)
 - Número significativo de acesso a operandos
 - Grande número de acesso a variáveis escalares locais

Arquitetura RISC

Uso de um grande número de registradores

- > Duas soluções:
 - Software
 - Delega-se ao compilador a responsabilidade de otimizar o uso dos registradores
 - O compilador tenta alocar nos registradores as variáveis que serão mais usadas durante um determinado período do programa
 - Hardware
 - Utilização de um número maior de registradores, normalmente no espaço deixado pela não utilização da ROM da unidade de controle

Arquitetura RISC

Uso de um grande número de registradores

- A utilização de um grande número de registradores tem por objetivo diminuir o número de acessos à memória
- Separar registradores para variáveis locais e globais
- Problema:
 - O conceito de local varia com a chamada e retorno de procedimentos
 - Parâmetros devem ser passados
 - Resultados devem ser retornados
 - No retorno do procedimento, os valores salvos anteriormente devem ser restaurados

25

Agosto 2008

Arquitetura RISC

Uso de um grande número de registradores

- Dado que:
 - Os procedimentos são chamados com poucos parâmetros
 - A profundidade de ativações de procedimentos varia dentro de uma faixa estreita
- Definição de vários pequenos conjuntos de registradores, cada um alocado a um procedimento diferente
- Cada chamada utiliza um conjunto diferente de registradores de tamanho fixo

26

Agosto 2008

Arquitetura RISC

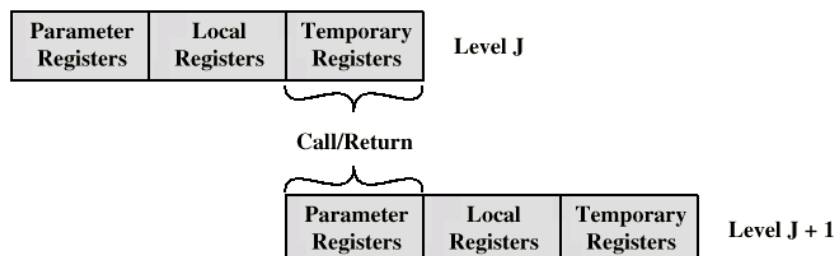
Uso de um grande número de registradores

- > Cada conjunto de registradores possui três áreas de tamanho fixo:
 - Registradores de parâmetros: armazenam valores dos parâmetros passados do procedimento que originou a chamada bem como os resultados a serem retornados]
 - Registradores locais: utilizados para variáveis locais
 - Registradores temporários: utilizados para trocar parâmetros e resultados com o procedimento de nível inferior seguinte

Arquitetura RISC

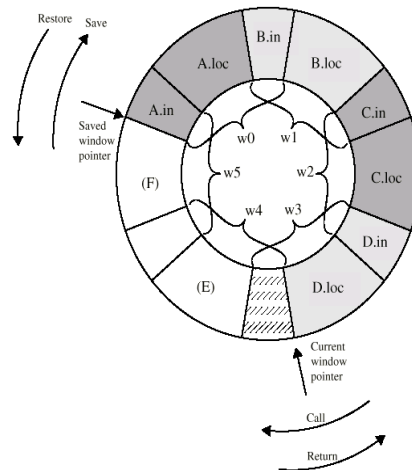
Uso de um grande número de registradores

- Os registradores temporários de um nível são, fisicamente, os mesmos que os registradores de parâmetro do nível inferior



Arquitetura RISC

Uso de um grande número de registradores



29

Agosto 2008

Arquitetura RISC

Uso de um grande número de registradores

> Operação do buffer circular

- Quando uma chamada é feita, o CWP (*Current Window Pointer*) é incrementado
- Se todas as janelas estiverem sendo utilizadas, é gerada uma interrupção que salva a janela mais velha na memória
- Um ponteiro denominado SWP (*Saved Window Pointer*) indica onde as janelas estão salvas na memória

30

Agosto 2008

Arquitetura RISC

Uso de um grande número de registradores

> Variáveis globais

O que fazer com o armazenamento das variáveis globais?

- Deixar as variáveis alocadas na memória e as instruções de máquina que utilizam essas variáveis devem acessar os operandos na memória
 - Solução simples tanto do ponto de vista de hardware quanto de software mas não eficiente caso existam muitas variáveis globais
- Incorporação de um conjunto de registradores globais
 - Divide-se os registradores entre locais e globais
 - Aumenta a complexidade do hardware pois precisa-se lidar com a divisão de endereçamento dos registradores

Arquitetura RISC

Uso de um grande número de registradores

- Utilização de um grande número de registradores ou acesso à memória cache?

Grande número de registradores	Cache
Todas as variáveis escalares locais	Variáveis escalares locais usadas recentemente
Variáveis individuais	Blocos de memória
Variáveis globais designadas pelo compilador	Variáveis globais usadas recentemente
Operações de salvamento/restauração baseadas na profundidade de aninhamento de procedimentos	Operações de salvamento/restauração baseadas no algoritmo de substituição de cache
Endereçamento de registrador	Endereçamento de memória

Arquitetura RISC

Otimização de registradores baseada em compiladores

- Otimização necessária quando a arquitetura RISC possui poucos registradores
- Programa em linguagem de alto nível não contém referência explícita a registradores
- Objetivo do compilador:
 - Manter em registradores, ao invés de na memória, os operandos requeridos no maior número possível de computações, minimizando a transferência de dados entre memória e registradores

Arquitetura RISC

Otimização de registradores baseada em compiladores

- Cada item de dado de um programa é alocado a um registrador simbólico ou virtual
- Associação de um número (ilimitado) de registradores virtuais ou simbólicos a um número (limitado) de registradores reais
- Registradores simbólicos que não se sobrepõem podem compartilhar o mesmo registrador real
- Se uma parte do programa precisar de mais registradores reais do que os disponíveis, alguns dados são armazenados na memória

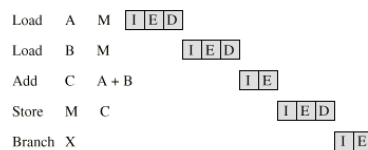
Arquitetura RISC Pipeline de instruções do RISC

- A maioria das instruções são registrador para registrador
- Duas fases:
 - I: busca da instrução
 - E: realiza uma operação da ULA, com entrada e saída em registradores
- Para operações de load/store, três fases:
 - I: busca da instrução
 - E: calcula o endereço de memória
 - D: operação de registrador para memória ou de memória para registrador

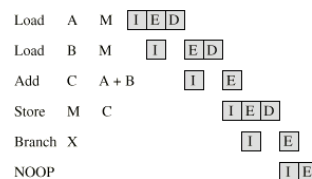
35

Agosto 2008

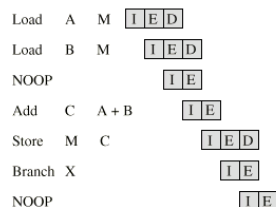
Arquitetura RISC: Pipeline de instruções do RISC



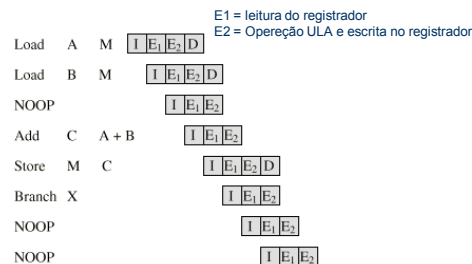
(a) Sequential execution



(b) Two-way pipelined timing



(c) Three-way pipelined timing



(d) Four-way pipelined timing

36

Agosto 2008

Arquiteturas RISC x CISC

Esta batalha ainda não acabou...

E novas frentes de batalha se abriram depois...

RISC

CISC

VLIW

Processadores Super-Escalares

Processadores Vetoriais

Processamento Paralelo: MIMD, SIMD, Multi-Core

Processadores Dedicados: GPU, FPU

Processadores Reconfiguráveis

Arquiteturas RISC x CISC

Esta batalha ainda não acabou...

E novas frentes de batalha se abriram depois...

RISC

CISC

VLIW

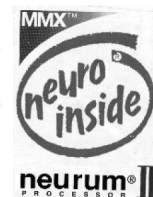
Processadores Super-Escalares

Processadores Vetoriais

Processamento Paralelo: MIMD, SIMD, Mu

Processadores Dedicados: GPU, FPU

Processadores Reconfiguráveis



Microprocessadores

The history of the microprocessor

Michael R. Betker, John S. Fernando, Shaun P. Whalen

Processor Architecture Department, Lucent's Microelectronics Group, Allentown, Pennsylvania
<http://www3.interscience.wiley.com/journal/97518358/abstract>

8080 – Z80 Instruction Set

<http://nemesis.lonestar.org/computers/tandy/software/apps/m4/qd/opcodes.html>

Intel - History of Intel Microprocessors

<http://www.intel.com/pressroom/kits/quickref.htm>

Microprocessor instruction set cards

<http://vmoc.museophile.org/cards/>

William Stallings - Computer Organization and Architecture 6th Edition

Chapter 13 - Reduced Instruction Set Computers

Slides 17-36 / Material de aula da Profa. Sarita / USP-ICMC



INFORMAÇÕES SOBRE A DISCIPLINA

USP - Universidade de São Paulo - São Carlos, SP

ICMC - Instituto de Ciências Matemáticas e de Computação

SSC - Departamento de Sistemas de Computação

Prof. Fernando Santos OSÓRIO

Web institucional: <http://www.icmc.usp.br/ssc/>

Página pessoal: <http://www.icmc.usp.br/~fosorio/>

E-mail: [fosorio \[at\] icmc. usp. br](mailto:fosorio@icmc.usp.br) ou [fosorio \[at\] gmail. com](mailto:fosorio@gmail.com)

Disciplina de Arquitetura de Computadores / Informática

Web disciplina: COTEIA - <Http://coteia.icmc.usp.br>

> Programa, Material de Aulas, Critérios de Avaliação,

> Material Complementar, Trabalhos Práticos, Datas das Provas

Visite regularmente o site da CoTeia - Material em permanente atualização