Arquitectura de Computadores II

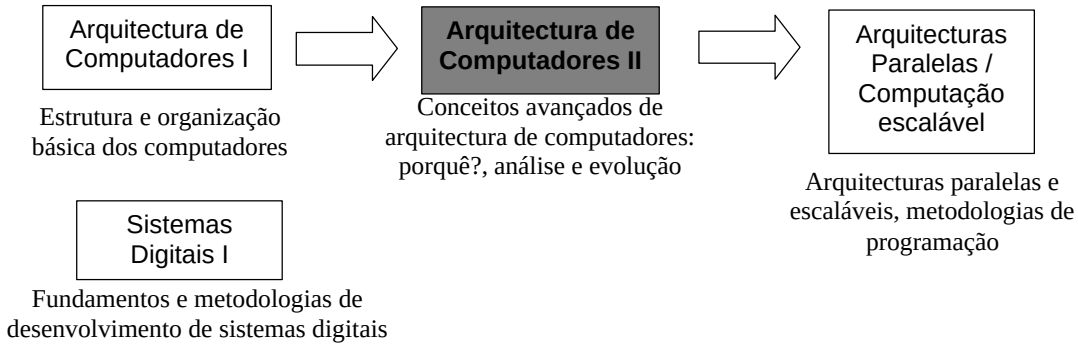
LESI - 3º Ano

Introdução

João Luís Ferreira Sobral
Departamento do Informática
Universidade do Minho

Janeiro 2002

Objectivos da disciplina e relação com outras disciplinas



- Conceitos já adquiridos em disciplinas anteriores:
  - Organização e estrutura de um computador
  - Interface do processador com programas escritos em HLL (imperativas)
  - Interface do processador com os restantes componentes do computador
  - Metodologia de programação em linguagem máquina
  - Sistemas digitais com elementos constituintes de qualquer computador

- Com ACII pretende-se analisar arquitecturas de suporte a computação escalável e estabelecer pontes entre os conceitos adquiridos e as tecnologias existentes.

- Conceitos a adquirir na disciplina:
  - Estrutura e organização dos processadores actuais, incluindo o suporte ao paralelismo ao nível da instrução (*pipeline* e super-escalaridade).
  - Análise comparativa do desempenho de programas reais.
  - Metodologia de desenvolvimento de programação com periféricos, de baixo nível, com periféricos reais.
  - Paralelismo ao nível da arquitectura de sistemas de computação, incluindo a análise de sistemas de memória partilhada e de memória distribuída.
  - A utilização de máquinas virtuais como solução para o problema da portabilidade do código.

Programa

Arquitectura interna de um processador

- Revisão do *datapath* de um processador e desenho da unidade de controlo. Implementação dos componentes do *datapath*
- Extensão da arquitectura para suporte à execução de instruções em vários ciclos de relógio. Análise da unidade de controlo, da respectiva máquina de estados finita e da sua implementação com ROM ou PLA.
- Técnicas alternativas para a implementação da unidade de controlo: microprogramação.
- Integração de interrupções e excepções numa unidade de controlo.
- Paralelismo ao nível da instrução através de mecanismos de *pipelining*: identificação de problemas com *pipeline* e sua resolução. *Pipelining* dinâmico.
- Técnicas alternativas de paralelismo ao nível da instrução: processadores super-escalares e *very long instruction word* (VLIW)
- Análise de desempenho de arquitecturas com *pipeline* e super-escalaridade.

Arquitecturas paralelas/distribuídas

- Classificação das arquitecturas paralelas segundo Flynm. Análise de exemplos (SIMD, MIMD e vectoriais).
- Sistemas de memória partilhada (UMA e NUMA), coerência de caches e modelos de programação.
- Sistemas de memória distribuída, topologia de interligação e modelos de programação. *Clusters* de SMD
- Arquitecturas paralelas com elevada disponibilidade, com redundância ao nível do SW e HW.
- Desenvolvimento de aplicações portáveis através da utilização de máquinas virtuais.

Programação de periféricos

- Barramentos, discos, redes, entrada/saída de dados e rendimento de E/S
- Desenvolvimento de aplicações para um periférico genérico (PG01/UM)

Bibliografia

- Essencial

COD2e *Computer Organization and Design: the hardware/software interface*, D. Patterson and J. Henessy, Morgan Kaufmann, 2ªedição, 1998. (capítulos 5, 6, 8 e 9)

- Complementar

CA2e *Computer Architecture: a quantitative approach*, D. Patterson and J. Henessy, Morgan Kaufmann, 2ªedição, 1996.

COA5e *Computer Organization and Architecture – Designing for Performance*, W. Stalling, Prentice Hall, 5ªedição, 2000.

Planeamento

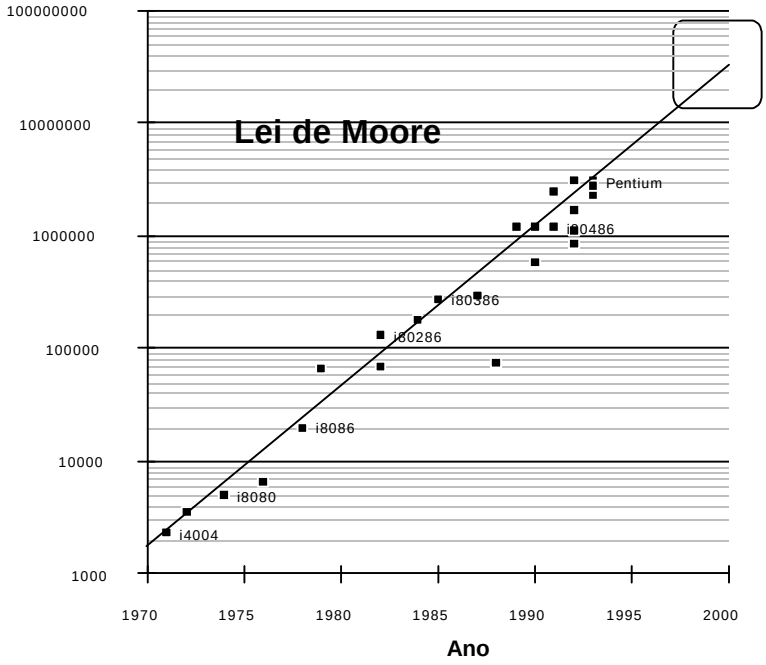
Aulas	#Aulas	Matéria	**Bibliografia**
1	1	- Objectivos e programa da disciplina - Evolução das arquitecturas de computadores	COD2e 1 e 2 CA2e 1 e 2
2 – 3	2	- Revisão do *datapath* dum processador MIPS	COD2e 5
4 – 6	3	- Execução de instruções em vários ciclos máquina	COD2e 5
7 – 9	3	- E/S e periféricos	COD2e 8 COA5e 3, 5 e 6
10 – 13	4	- *Pipelining* e super-escalaridade	COD2e 6
14 – 16	3	- *Pipelining* avançado e ILP	CA2e 3 e 4
17 – 19	3	- Processamento paralelo	COD2 9
20 – 21	2	Máquina virtuais	

Avaliação

- A frequência às aulas teóricas e teórico-práticas não é obrigatória, mas **é recomendada**
- 100% Exame escrito

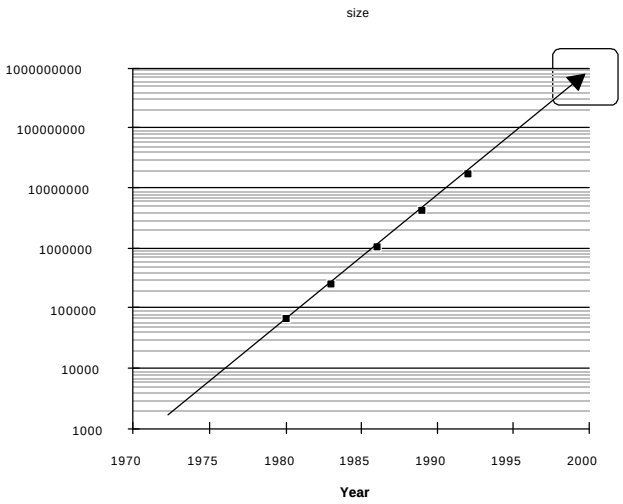
Evolução das arquitecturas de computadores

- Capacidade dos microprocessadores (transistores):



Evolução de CMOS:
Die size: 2X cada 3 anos
Largura de linha: ½ / 7 anos

- Capacidade de um só chip de DRAM:



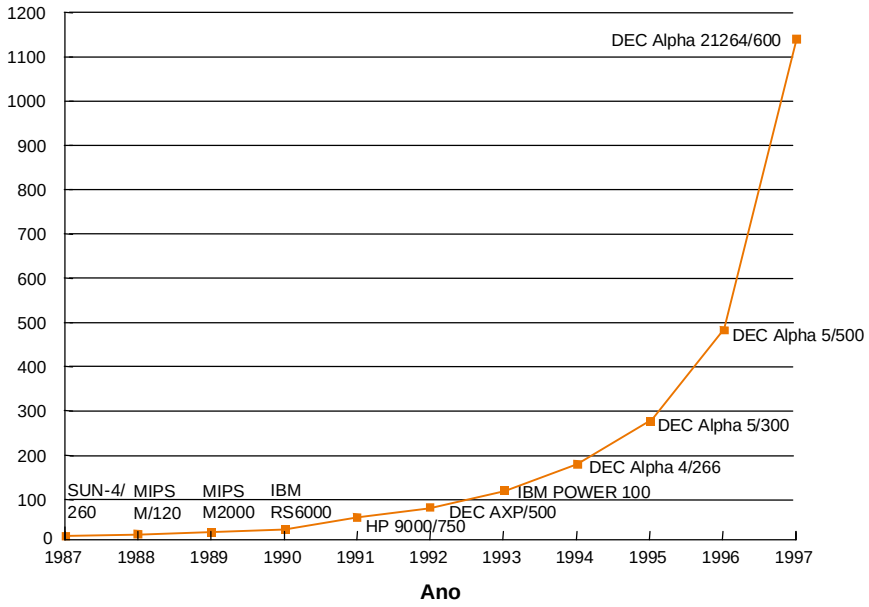
Ano	Capaci-dade(Mb)	Tempo de acesso
1980	0,0625	250 ns
1983	0,25	220 ns
1986	1	190 ns
1989	4	165 ns
1992	16	145 ns
1996	64	120 ns
2000	256	100 ns

- Evolução tecnológica:

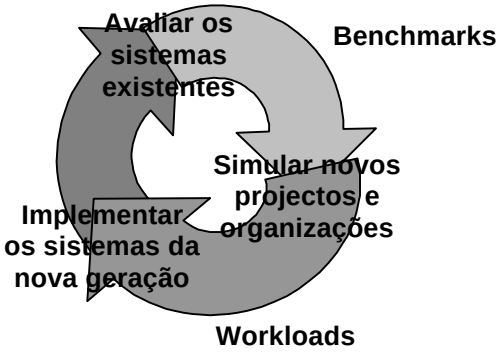
	Capacidade	Velocidade (latência)
Lógica	2x cada 3 anos	2x cada 3 anos
DRAM	4x cada 3 anos	2x cada 10 anos
Disco	4x cada 3 anos	2x cada 10 anos

Evolução das arquitecturas de computadores

- Desempenho do processador (em Spec Mark) duplica cada 3 anos



- Metodologia da Engenharia de Computadores



- Medição de desempenho
  - Hardware: custo, dissipação de calor, consumo, área
  - Tempo de execução (tempo de reposta), débito

Evolução das arquitecturas de computadores

Como determinar o desempenho: limite = tempo de execução

- Uma máquina X é *n vezes* mais rápida que outra Y se:

$$n = \frac{\text{Tempo de execução em Y}}{\text{Tempo de execução em X}}$$

Lei de Amdahl

- Ganho devido a uma melhoria E:

$$\text{ganho} = \frac{\text{Tempo de execução sem E}}{\text{Tempo de execução com E}}$$

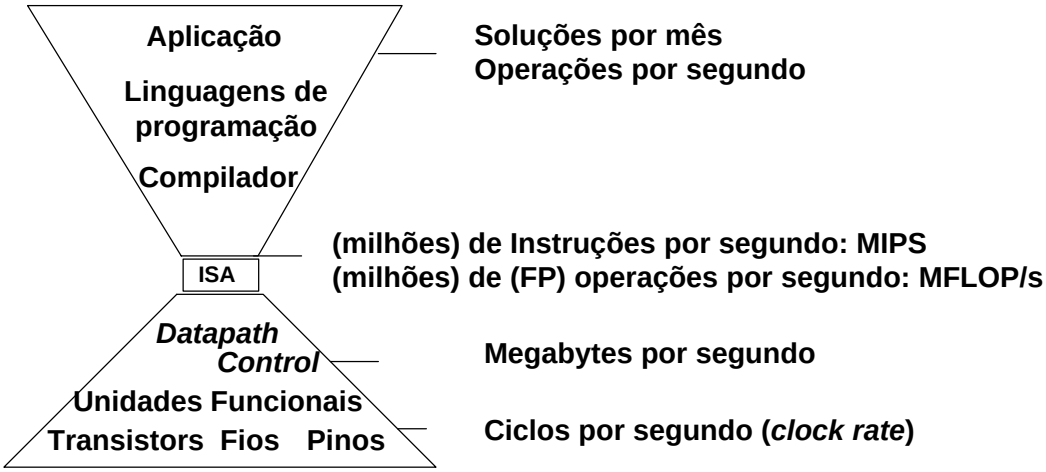
- Se a melhoria E acelerar um factor **A** uma fracção **F** do programa e o resto do programa permanecer inalterado:

$$\text{ganho} = \frac{1}{(1 - \textbf{F}\text{racção}) + \textbf{F}\text{racção}/\textbf{A}\text{celeração}}$$

- Exemplo: qual o ganho se melhorarem as instruções de FP em 2X, sabendo que apenas 10% das instruções são FP:

$$\text{ganho} = \frac{1}{(1 - 0{,}1) + 0{,}1/2} = \frac{1}{0{,}95} = 1{,}053$$

Métricas de desempenho



 © João Luís Sobral 2002

Evolução das arquitecturas de computadores

Desempenho do CPU

- O tempo de execução de um programa é dado por:

  #Instruções x ciclos por instrução (CPI) x período do relógio (Tcc)

  = #I x CPI x Tcc

- Impacto de cada nível de abstracção em cada parcela:

	#I		Tcc
Programa			
Compilador			
Conj. instruções	X	X	
Organização		X	X
Tecnologia			X

- CPI = número médio de ciclos por instrução:

  CPI = número de ciclos / número de instruções

- O CPI pode ser determinado para cada classe de instruções, sendo o CPI médio calculado multiplicando a frequência de cada instrução pelo seu CPI:

$$CPI_{global} = \sum_{i=1}^{n} CPI_i * F_i \qquad onde\ F_i = \frac{\#I_i}{\#I}$$

- Exemplo:

Operação	Freq (F*i*)	Ciclos por Instr. (CPI*i*)	CPI*i* x F*i*	(% tempo)
ALU	50%	1	0,5	(33%)
Load	20%	2	0,4	(27%)
Store	10%	2	0,2	(13%)
Branch	20%	2	0,4	(27%)
		Σ=	1,5	

Combinação típica (mix)

SPECint95 dada p

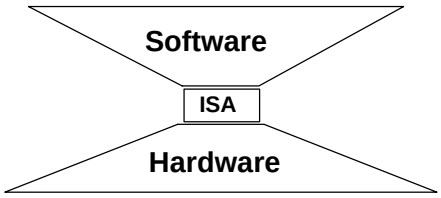
SPECfp95 dada por 10 programas com aritmética de vírgula flutuante

- Um único conjunto de optimizações para todos os programas

Evolução das arquitecturas de computadores

Evolução do conjunto de instruções do processador (ISA)

- O conjunto de instruções de um processador efectua a interface entre o HW e o SW, sendo a face do HW visível ao programador (e compilador):



- Tipos de ISA:

  - *Stack*

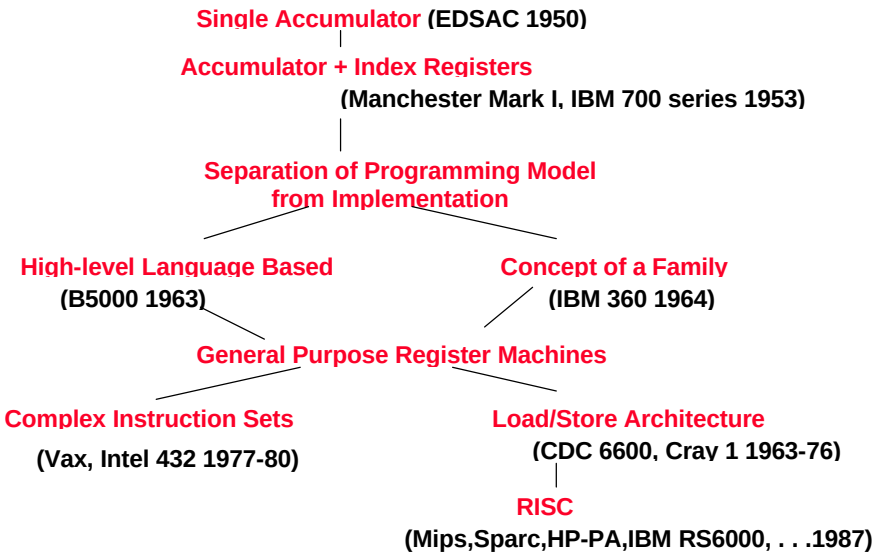
  - Acumulador

  - Registos de uso genérico (GPR) (Mem-reg e Load-store)

  Exemplo: Calcular C = A + B

Stack	Acumulador	Reg-Mem	*Load-store*
Push A	Load A	Load R1, A	Load R1, A
Push B	Add B	Add R1, B	Load R2, A
Add	Store C	Store R1, C	Add R3, R1, R2
Pop C			Store R3, A

- Conjuntos de instruções mais significativos



Evolução do conjunto de instruções do processador (ISA)

- tipicamente associados a alterações no desenho do ISA

  Ex. Stack vs GPR (System 360)

- O espaço de endereçamento é normalmente uma das principais limitações do ISA

  Ex. 8086 continha endereços de 20 bits, limitando a RAM a 1 MB
  A linha Intel Pentium utiliza 32 bits, limitando a RAM a 4 GB !!!

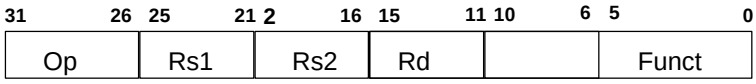
- As opções de projecto devem ter em conta:

  -
  - organização da máquina
    linguagens de programação
  -
  - sistemas operativos

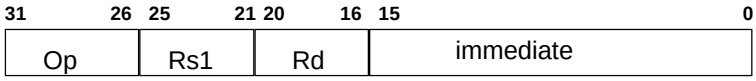
-

  Instruções de tamanho fixo (32 bits)
  - genérico de 32 bits
    3 modos de endereçamento, aritmética registo registo
    único modo de endereçamento para o e *store*
  - condições de salto simples
    saltos retardados

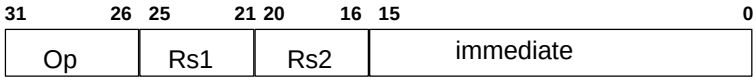
TIPO R : Register-Register



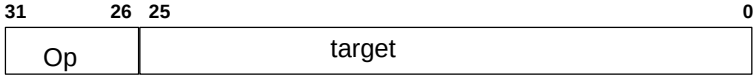
TIPO I : Register-Immediate



TIPO I : Branch



TIPO J : Jump / Call



© João Luís Sobral 2002

Bibliografia

- **Secções 1.1, 1.4, 1.6, 2.1, 2.2, 2.3 e 2.4** de *Computer Organization and Design: the hardware/software interface*, D. Patterson and J. Hennessy, Morgan Kaufmann, 2ª edição, 1998.

- **Adicional:**

  - Secções 1.3, 1.5, 1.6, 2.2 e 2.11 de *Computer Architecture: A Quantitative Approach*, J. Hennessy and D. Patterson, Morgan Kaufmann, 2ªedição, 1996.