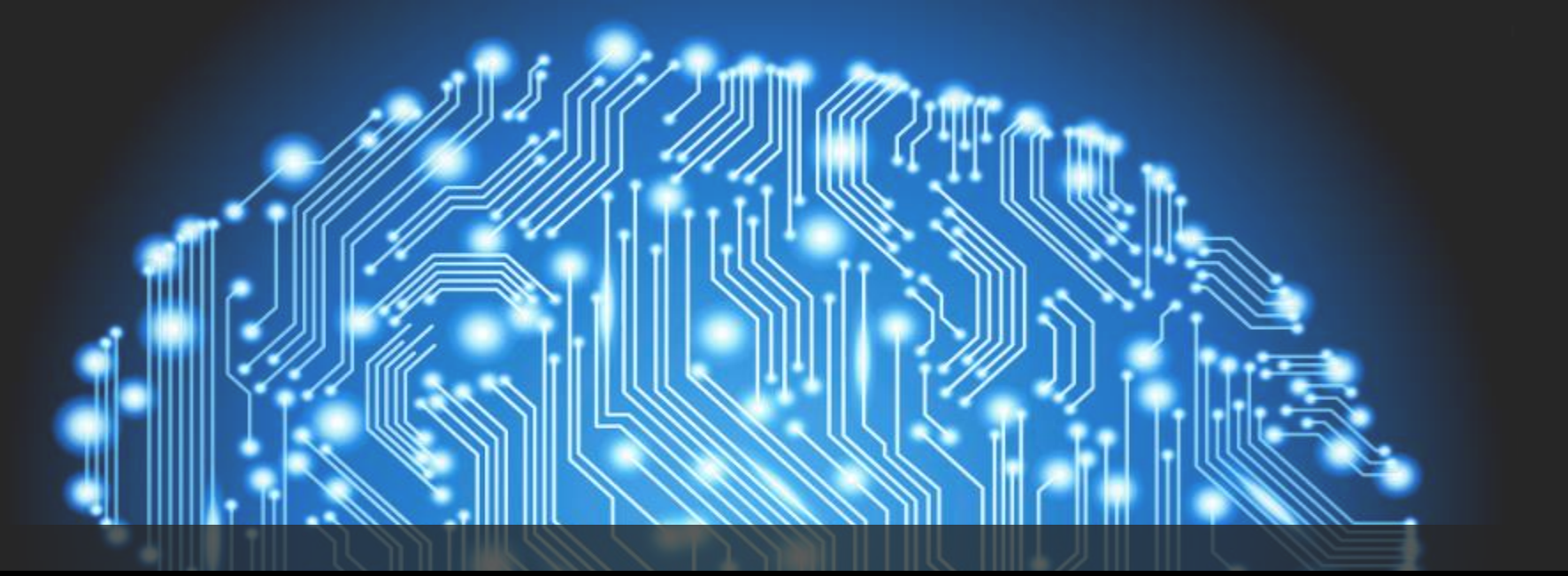


Introdução à Computação (I.C)

Módulo 05

Prof. Daniel Caixeta

Conteúdo programático

10

Instruções de máquina: Como o processador recebe as instruções?

10.1. Breve introdução.
- 10.1.1. Formato das instruções.
- 10.1.2. Executando as instruções.

11

Arquiteturas & Processadores: RISC *versus* CISC

11.1. Conceitos iniciais.
11.2. Um pouco da história.
11.3. Definições RISC e CISC.
- 11.3.1. RISC.
- 11.3.2. CISC.

11.4. RISC *versus* CISC.
- 11.4.1. Análise comparativa.
- 11.4.2. Quadro comparativo: Vantagens e Desvantagens.
- 11.4.3. Atualmente.

Referências

10. INSTRUÇÕES DE MÁQUINA.

Como o processador recebe as instruções.

10.1. BREVE INTRODUÇÃO

- Segundo Monteiro (2007 & 2014), uma máquina pode executar tarefas complicadas e sucessivas se for “instruída” sobre o que fazer e em que sequência isso deve ser feito.
- Os seres humanos, ao receberem uma instrução, *e.g.*, fazer café, precisam realizar uma série de ações intermediárias, até que a tarefa seja realizada por completo.
- Então, considerando o exemplo acima, qual forma “correta” de se fazer café?

Dependem das instruções.

- Da mesma forma, para o computador é necessário que cada instrução seja detalhada em pequenas etapas. Isso ocorre porque eles são projetados para entender e executar pequenas operações, ou seja, as operações mais básicas (*e.g*., soma, subtração). Essas pequenas etapas dependem do conjunto de instruções do computador. (FÁVERO, 2011).
- Assim, uma instrução de máquina pode ser definida pela formalização de uma operação básica que o *hardware* é capaz de realizar diretamente no processador. (MONTEIRO, 2007 & 2014).
- Em outras palavras, consiste em transformar instruções mais complexas em uma sequência de instruções básicas e compreensíveis pelo processador. (FÁVERO, 2011).

- Como exemplo, vamos considerar um processador com uma ULA capaz de executar a soma ou a multiplicação de dois números (operações básicas), mas não as duas coisas ao mesmo tempo.
- Agora imaginem que esse processador precise executar de uma só vez a instrução abaixo:

$$X = A + B * C$$

- Esse processo poderá gerar a necessidade de transformar essa instrução, considerada complexa, em uma sequência de instruções mais básicas, que poderá ser da seguinte forma:

1º passo: Executar $T_R = B * C$ (T_R = registrador ou memória temporária).

2º passo: Realizar a operação $X = A + T_R$.

Ordem de precedência das operações aritméticas.

cont.[...]

- O projeto de um processador é centrado no conjunto de instruções de máquina que se deseja que ele execute, ou que poderá executar. Quanto menor e mais simples for o conjunto de instruções, mais rápido é o ciclo de tempo do processador. (MONTEIRO, 2007 & 2014).
- Portanto, segundo Fávero (2011), um processador precisa dispor de instruções para:

1. Movimentação de dados.	7. Desvio.
2. Aritméticas.	8. Modificação de memória.
3. Lógicas.	9. Formas de ligação à sub-rotina.
4. Edição.	10. Manipulação de pilha.
5. Deslocamento.	11. Entrada e saída.
6. Manipulação de registros de índice.	12. Controle.

10.1.1. FORMATO DAS INSTRUÇÕES

- De acordo com Monteiro (2007 & 2014), uma instrução é formada basicamente por dois campos:

1. Código de operação (*Opcode*):
   - Trata-se de um subgrupo de *bits* que identifica a operação a ser realizada pelo processador.
   - É o campo da instrução cujo valor binário identifica a operação a ser realizada.
   - Esse valor é a entrada no Decodificador de Instruções na Unidade de Controle.
   - Cada instrução deverá ter um código único que a identifique.

2. Operando:
   - Trata-se de um subgrupo de *bits* que identifica o endereço de memória onde está contido o dado que será manipulado, ou pode conter o endereço onde o resultado da operação será armazenado.

- E Farias (2013), complementa Monteiro (2007 & 2014), trazendo o três tipos de instruções inseridas nas arquiteturas de *hardware* (processadores). São eles:

    1. Transferência de dados: Realizam cópia de valores entre registradores da UCP e a memória principal. *E.g.*, STORE e LOAD.
    2. Lógicas e aritméticas: Ativam os circuitos específicos da ULA para a realização das operações. *E.g.*, ADD, SHIFT, OR, AND, etc.
    3. Controle: Responsável por tratar as sequências da execução do programa. *E.g.*, JUMP e CALL. Esses comandos são usados para alterar o fluxo normal de um programa, implementando assim os desvios condicionais, códigos de repetição, chamada de função e retorno. Aqui não há manipulação dos dados.

- Exemplifiquemos uma instrução, com a codificação de operação STORE e seus operandos. (FARIAS, 2013).

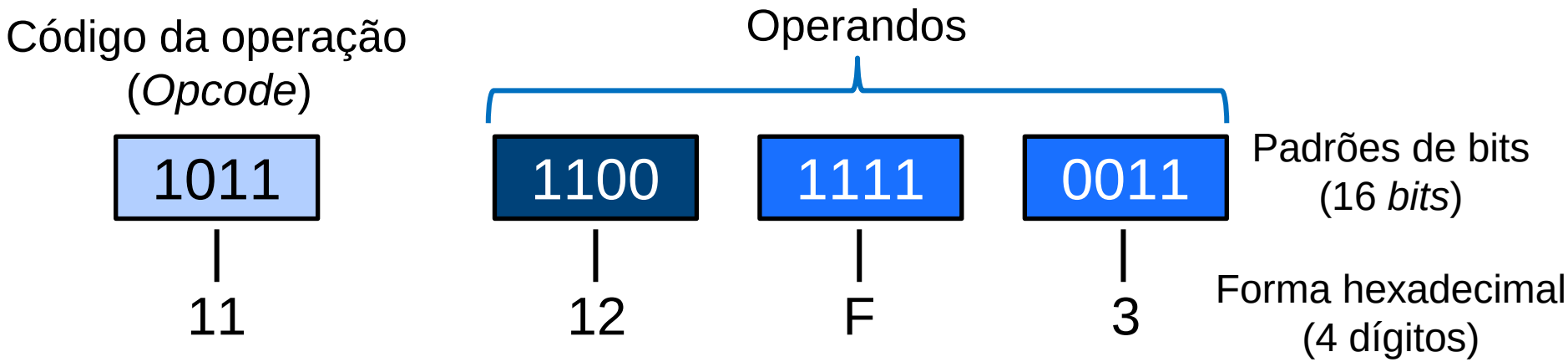


Figura 1. Ilustração do exemplo.

- O primeiro dígito hexadecimal, o 11, representa, a operação *STORE* (armazena o conteúdo de um registrador em memória).
- O dígito hexadecimal seguinte (= 12), representa o identificador do registrador que possui o conteúdo a ser gravado.

- Já os pares de dígitos hexadecimais F e 3 representam o endereço na memória principal onde o conteúdo do registrador 12 será armazenado.
- Podemos traduzir este código da seguinte forma:
  - ✓ Armazena o padrão de *bits* contido no registrador 12 para a célula de memória de endereço F3.

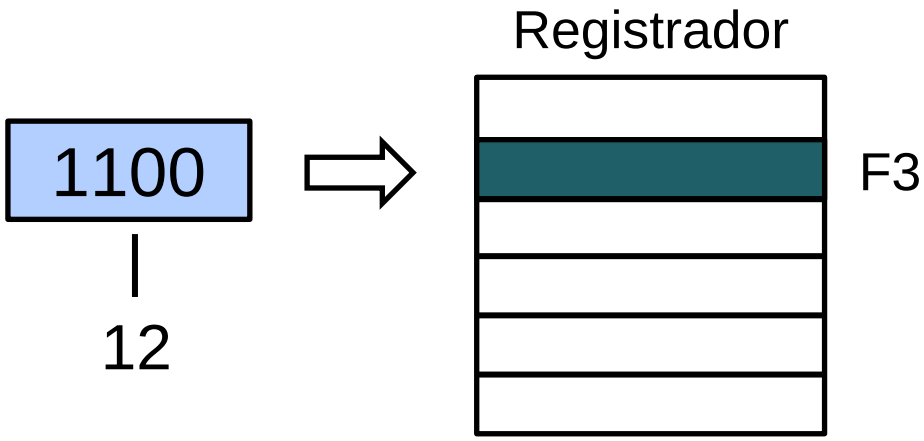


Figura 2. Ilustração do exemplo.

10.1.2. EXECUTANDO AS INSTRUÇÕES

- Um programa é uma sequência de instruções em uma linguagem a ser executada com o objetivo de realizar uma determinada atividade pré-programada. (FARIAS, 2013).
- Para isso, o programa compilado em linguagem de máquina, é posto na memória principal pela UC, que busca cada instrução de máquina, a decodificação e o gerenciamento da execução desta instrução com o auxílio da ULA.
- Mas o que é Linguagem de Máquina?
    - É o conjunto de instruções codificadas em *bits* com os comandos a serem executados durante os processo. É uma escrita puramente binária.

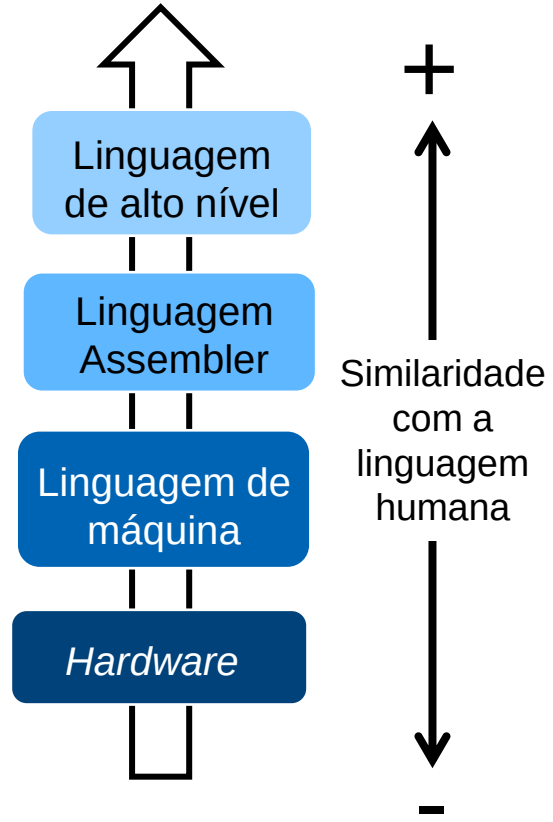


Figura 3. Níveis das língua-gens programação.

cont.[...]

- Para cada ciclo, a UC possui dois registradores específicos que são: o contador de instruções e o registrador de instruções. Vamos às suas definições:
  - O contador de instruções armazena o endereço de memória da próxima instrução a ser executada, assim a UC fica informada sobre a posição do programa em execução.
  - Já o registrador de instruções guarda a instrução de máquina que está em execução.

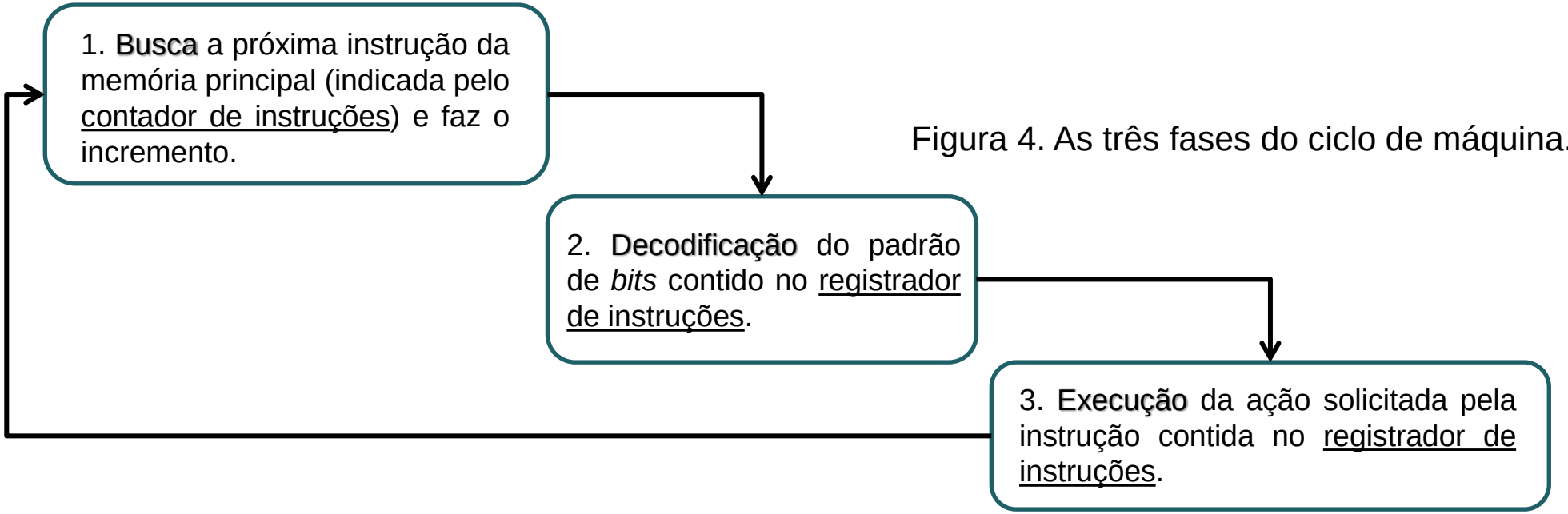
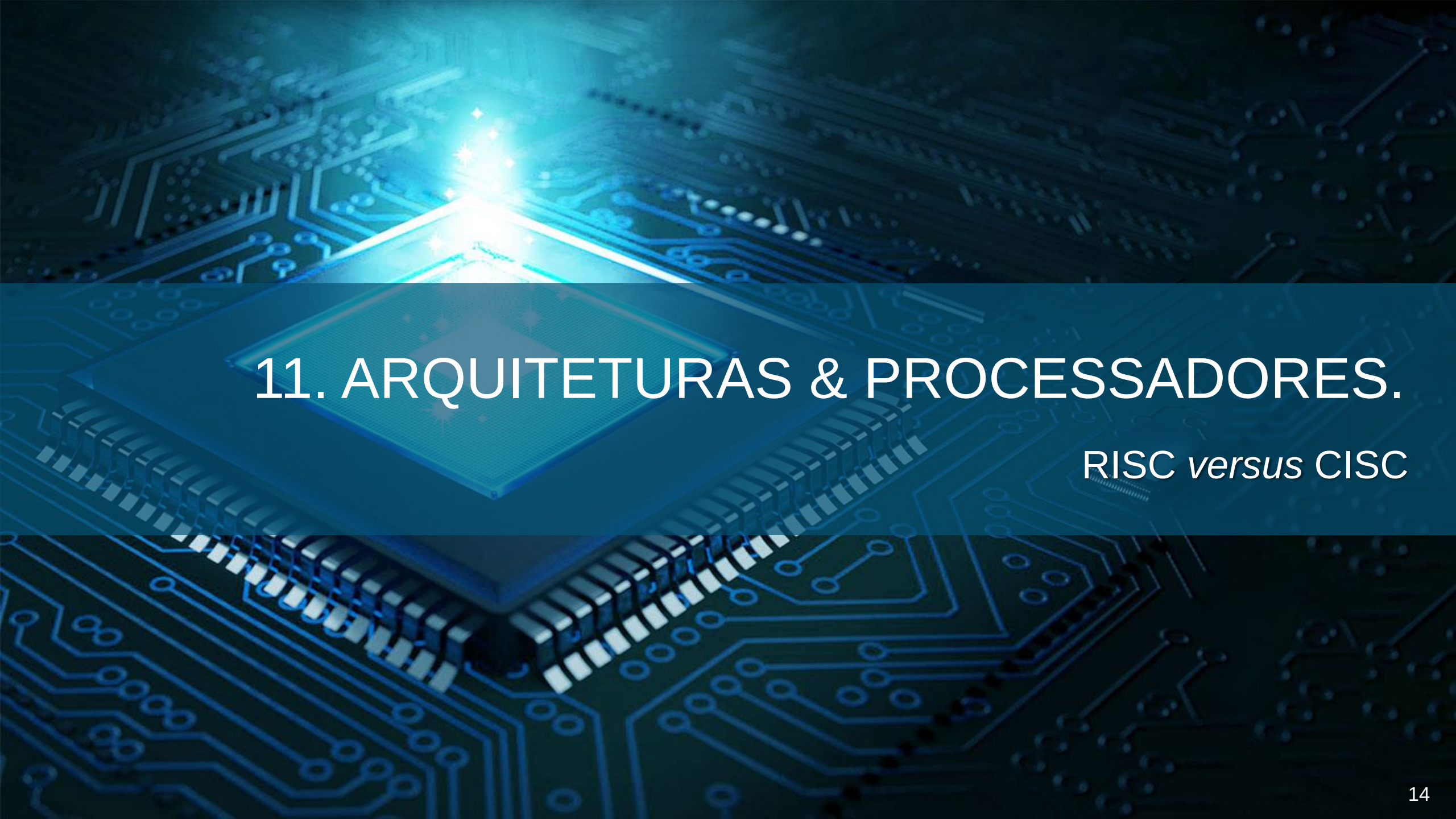


Figura 4. As três fases do ciclo de máquina.



11. ARQUITETURAS & PROCESSADORES.

RISC *versus* CISC

11.1. CONCEITOS INICIAIS

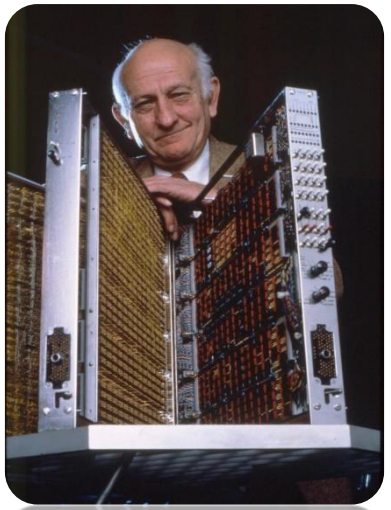
- De acordo com Caixeta, *et.al.* (2009), o estudo da arquitetura de processadores é sem sombra de dúvidas um dos assuntos mais interessantes no ramo da ciência da computação, sendo ainda um tema de grande relevância didática para as diversas áreas da T.I., por isso que a compreensão de um sistema computacional é fundamental para entendermos a evolução dos *hardwares* e *softwares*.
- Na realidade, sempre existiu a discussão em torno de qual seria a melhor ou a mais apropriada [...]. É interessante observar que não sendo possível adotar qualquer uma das duas arquiteturas, o que podemos ter é um sistema híbrido que contenha o que há de melhor entre as duas tecnologias. (*ibidem*).

cont.[...]

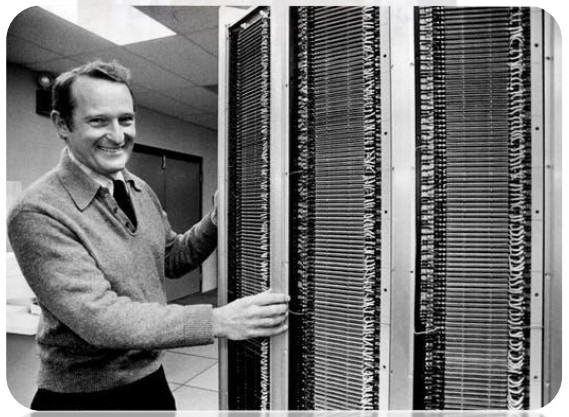
- Portanto, ainda há controvérsias, mas ressalto que avanços tecnológicos permitiram a criação de novas arquiteturas de processadores, *e.g*., modelos *multicore*, mas na essência ainda permanecem as mesmas.
- Sendo assim, surge a necessidade de um breve estudo comparativo para avaliar em quais cenários cada uma destas é melhor empregada, e o mais importante, como são utilizadas. (AZEVEDO & OLIVEIRA, 2014).
- E fato, essa complexidade impulsionou tecnologicamente as formas de se projetar arquiteturas de computadores.

11.2. UM POUCO DA HISTÓRIA

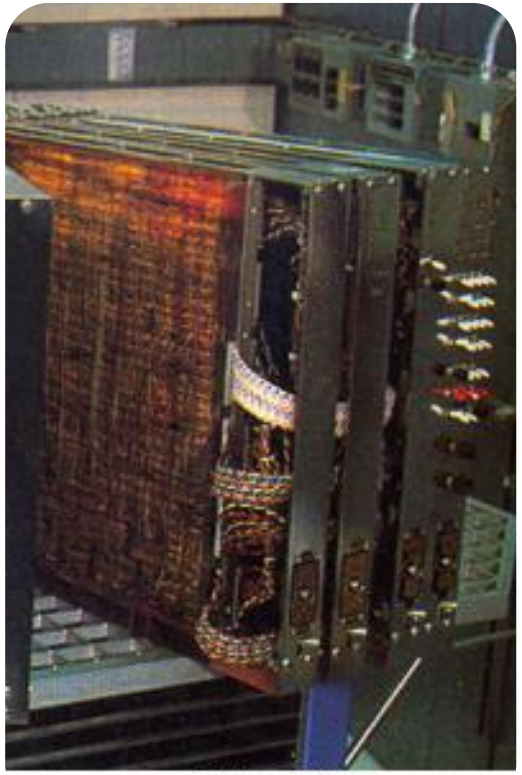
- No início dos anos de 1970, os projetistas de *hardware* não pensavam em projetar máquinas comuns. (CAIXETA, *et.al.,* 2009).
- A criação do modelo CISC, que dominava o mercado tecnológico, fez com que pesquisadores pensassem em desenvolver arquiteturas que competissem com essa tecnologia. (*ibidem*).
- Então, um grupo de pesquisadores da IBM, liderados por John Cocke, pesquisava uma forma de incorporar em um minicomputador de alta performance as ideias de Seymour Cray, possibilitando o desenvolvimento do modelo 801/IBM. (*ibidem*).



John Cocke (1925 - 2002)



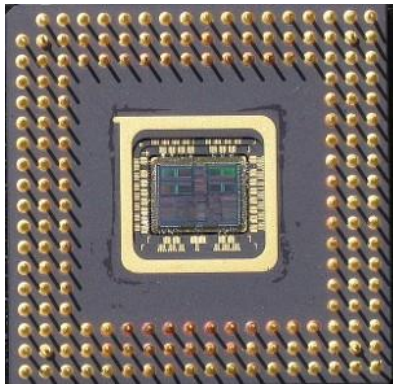
Seymour Cray (1925 - 1996)



- Na verdade, esse minicomputador nunca fora comercializado. Mas, as informações obtidas durante a fase de pesquisa e desenvolvimento vazaram e outros grupos concorrentes começaram a estudar esse modelo de arquitetura. (*ibidem*).
- No ano de 1980, com a criação do modelo VLSI (*Very Large Scale Integrator*), dispositivo esse que não mais utilizava o modo de interpretação da informação, desenvolvido por um grupo de pesquisa da Universidade de Berkeley (USA) e liderado por David Patterson e Carlo Séquin, criou-se o termo RISC (*Reduced Instruction Set Computer*). (*ibidem*).

- No ano seguinte, 1981, a SUN *Microsystems* desenvolveu os *chips* processadores denominados SPARC[1] e o MIPS[2], que foram um grande sucesso de comercialização. (CAIXETA, *et.al*., 2009).
- Esses processadores apresentavam arquiteturas diferentes dos computadores da época, pois os projetistas não tinham a necessidade de vincular a um conjunto de instrução padrão, já que o sistema possuía a liberdade de escolha de instruções que apresentassem melhor performance. (*ibidem*).

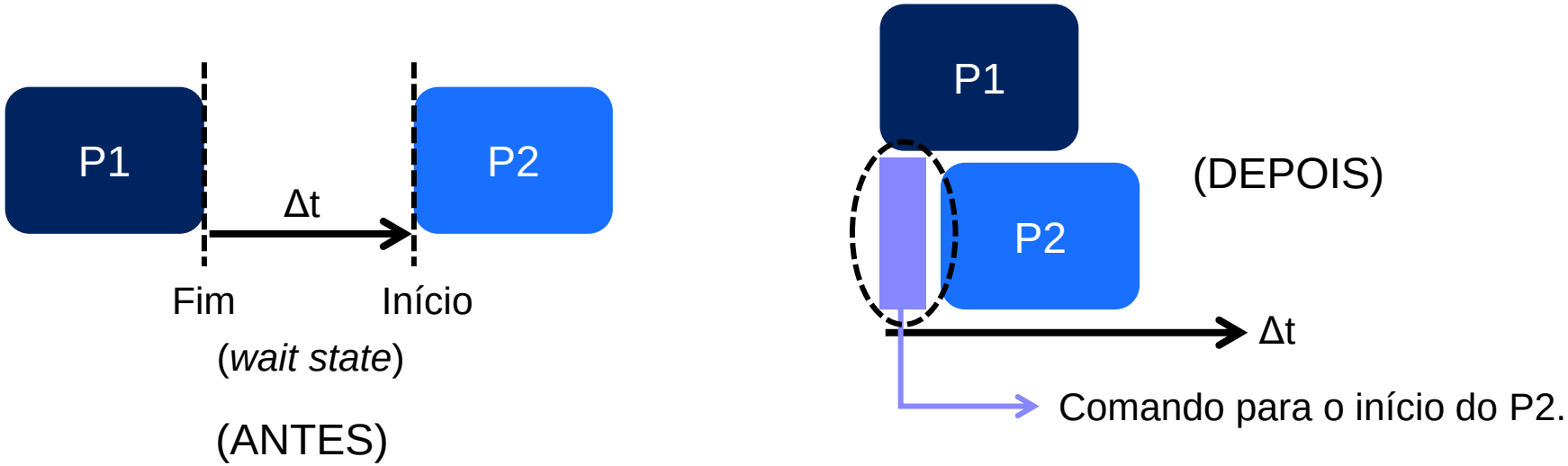
SPARC



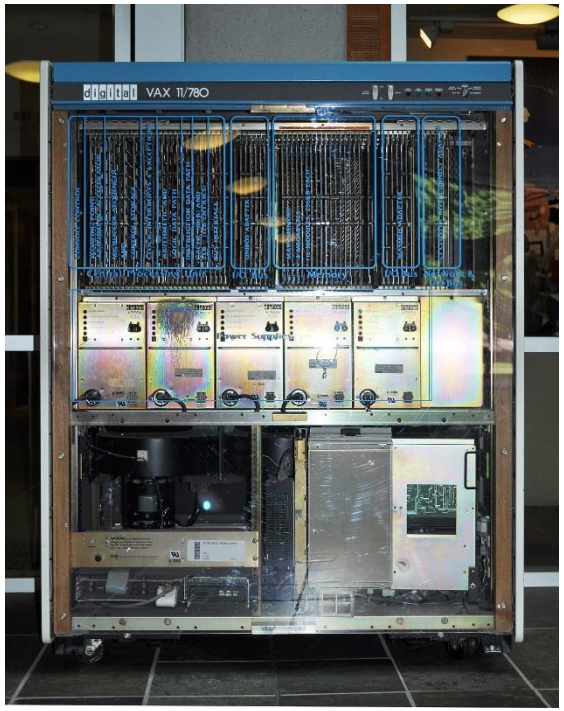
MIPS

1. SPARC - *Scalable Processor ARChitecture*/Arquitetura de Processadores Escaláveis.
2. MIPS - *Microprocessor without Interlocked Pipeline Stages*/Microprocessador sem Estágios Intertravados de *Pipeline*.

- Embora a ênfase dessa nova arquitetura era um *chip* com instruções mais simples, percebeu-se a necessidade de instruções que pudessem ser iniciadas prontamente, ou seja, antes de terminar uma instrução, outro processo se iniciava e com isso, a quantidade de comandos que poderiam ser iniciadas em um segundo, tornou-se mais importante que o tempo que uma instrução levava para ser executada. (CAIXETA, *et.al*., 2009).



- Mas o que chamou mesmo a atenção à época foram as quantidades de instruções que eles poderiam executar. (*ibidem*).



Computador VAX 11/780

- Na verdade essas instruções e procedimentos eram em torno de 50, um número bem menor se comparado aos conjuntos de instruções contidas nos processadores instalados nos computadores no mercado, que possuíam de 200 a 300 instruções, como os computadores da família VAX da DEC e os grandes *mainframes* da IBM. (CAIXETA, *et.al*., 2009).
- Sendo a arquitetura RISC (*Reduced Instruction Set Computer*) o inverso da arquitetura CISC (*Complex Instruction Set Computer*), que como a própria designação da arquitetura, informa “instruções complexas”, ao contrário da arquitetura RISC, que é composta de “instruções simples”.

11.3. DEFINIÇÕES DE RISC E CISC

11.3.1. RISC

- A arquitetura RISC (*Reduced Instruction Set Computer*/Computador com um Número Reduzido de Instruções) é uma arquitetura que usa um pequeno grupo de instruções e levam praticamente o mesmo tempo para serem processadas. Essas instruções são executadas diretamente pelo *hardware* da máquina, por não haver micro programação, ou seja, é um processamento direto. (CAIXETA *et.al*., 2009).
- Processa operações simples, tendo que executar mais instruções e ao mesmo tempo operações mais complexas. Entretanto, é necessário dividir essas instruções mais complexas em vários processos mais simples. (*ibidem*).
- Essa arquitetura pode realizar até cinco instruções ao mesmo tempo em comparação com a arquitetura CISC, que executa apenas uma. (*ibidem*).

cont.[...]

- Algumas características segundo Caixeta (*et.al*., 2009):

CARACTERÍSTICAS COMUNS
✓ Limitado número de instruções.
✓ Codificação de instruções em palavras de tamanho simples.
✓ Execução sem microcódigos.
✓ Altas taxas de execução.
✓ Uso intenso de *pipelines*.
✓ Poucos modos de endereçamento.
✓ Operações envolvendo a memória principal restritas a transferências.
✓ Operações lógicas e aritméticas entre registradores, tipicamente com instruções de três endereços.

11.3.2. CISC

- No início dos anos 1970, os compiladores eram bastante simples e pouco robusto, devido ao fato que a própria memória RAM era lenta e bastante cara, causando sérias limitações tecnológicas na dimensão dos algoritmos elaborados. (CAIXETA *et.al*., 2009).
- Esse “problema”, levou alguns pesquisadores da área de tecnologia a preverem uma crise no ramo do desenvolvimento de programas, o que de fato ocorreu com o barateamento dos *hardwares* e o aumento expressivo no preço dos *softwares*. (*ibidem*).
- A partir daí, um grande número de pesquisadores e arquitetos passaram a defender que a única alternativa seria mudar a complexidade do *software*, que era mais caro, para funcionar em um *hardware* que na época era mais barato. (*ibidem*).

cont.[...]

- A ideia dessa associação foi impulsionado pela criação da arquitetura CISC que à época sugeriram a mudança do código *assembly* para linguagens de alto nível como o C ou Pascal. (CAIXETA *et.al*., 2009).
- As principais razões que promoveram essa arquitetura foram, segundo Patterson & Séquin (1998):
  - ✓ Redução das dificuldades de escrita de compiladores.
  - ✓ Redução do custo global do sistema e dos custos de desenvolvimento de *software.*
  - ✓ Redução da diferença semântica entre as linguagens de programação e linguagem de máquina.
  - ✓ Fazer com que os programas escritos em linguagens de alto nível sejam executados com mais eficiência.
  - ✓ Melhorar compactação do código.
  - ✓ Facilitar a detecção e correção de erros nos códigos.

- A principal característica da arquitetura CISC é processar grandes e complexas instruções, como operações de multiplicação e divisão, assim como a execução e decodificação de grandes quantidades de operações.

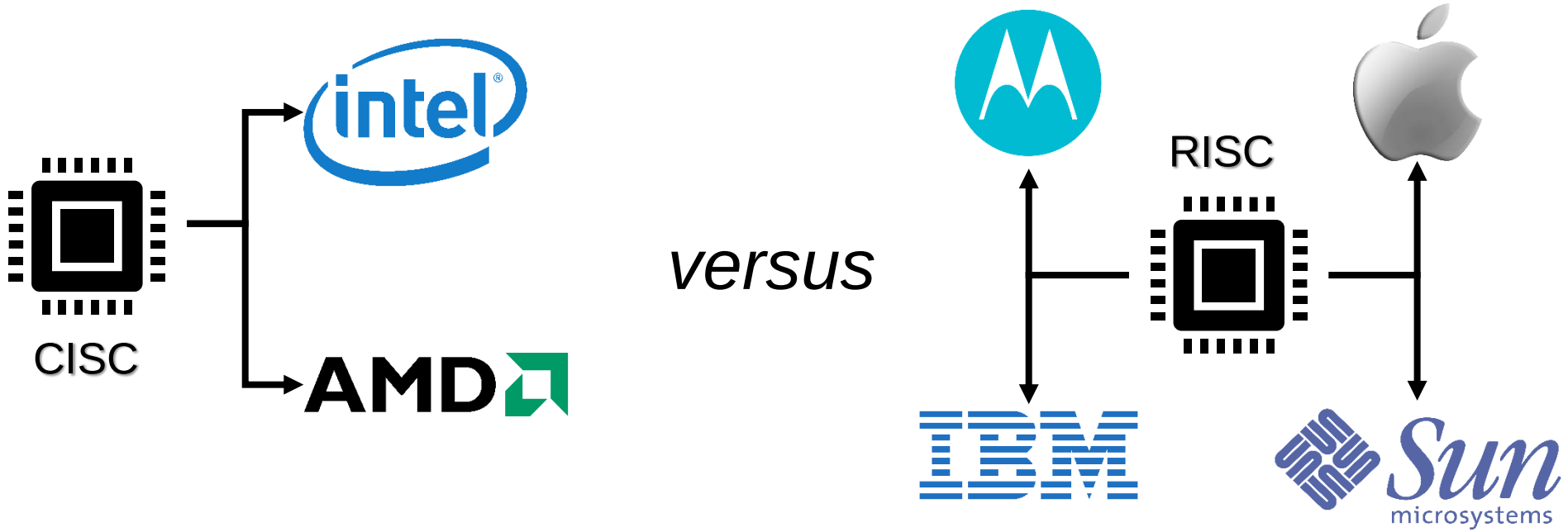


Figura 5. Arquitetura CISC *versus* RISC.

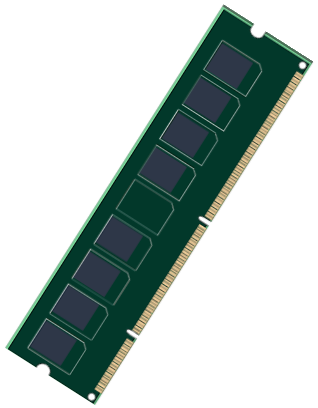
11.4. RISC *versus* CISC

- De acordo com Barbosa (s/d), ao contrário da complexa arquitetura CISC, os processadores RISC são capazes de executar apenas algumas poucas instruções simples, o que justifica o fato de os *chips* serem mais simples e bem mais baratos.
- Outra vantagem dos processadores RISC, é que por terem um menor número de circuitos internos, podem trabalhar com *clocks* mais altos, ou seja, são mais rápidos no processamento das instruções.
- Apesar de parecer estranho [...]. Como um processador que executa instruções simples pode ser considerado mais rápido que o que executa instruções mais complexas? A resposta [...]:

Está no fato que o processador CISC não executa a maioria das instruções frequentemente, o que difere da arquitetura RISC, que executam instruções simples, mas com mais frequência. (BARBOSA, s/d).

11.4.1. ANÁLISE COMPARATIVA

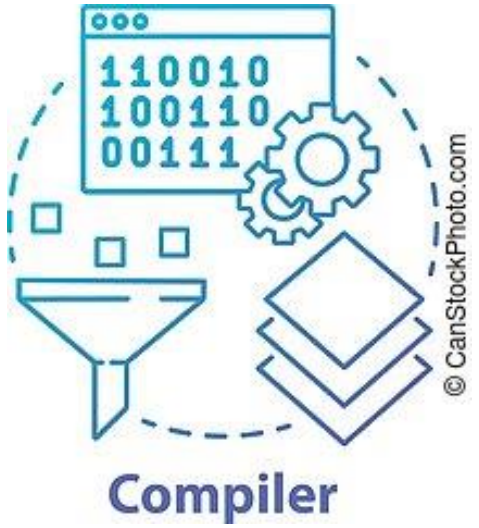
- Faremos agora uma breve análise comparativa levando-se em consideração os recursos de armazenamento, memória, compiladores e conjunto de instruções, segundo Azevedo (2014).



1. <u>Armazenamento e memória</u>
   - De acordo com Azevedo (2014), em dias atuais, os dispositivos de armazenamento e as memórias de computadores são mais baratas e mais rápidas.
   - Portanto, essas preocupações, que em décadas passadas eram bastante discutidas em relação às densidades dos códigos nas instruções CISC, hoje são desnecessárias.

2. Compiladores

- Segundo Azevedo (2014), os compiladores RISC tentam manter os operandos em registradores para poderem usar simples instruções registro-registro. Isto ocorre, para que se possam manter os operandos que serão reutilizados em registradores ao invés de acessar a memória repetidamente. (grifo meu).
- Usam-se as instruções *LOAD's* e *STORE's* para poderem acessar a memória para que os operandos não sejam rejeitados após o término da execução de uma instrução, como ocorre em arquiteturas que utilizam o modelo de execução memória-memória.



3. Conjuntos de instruções

- Como dito anteriormente, o microcódigo usado na arquitetura CISC permite aos programadores executarem códigos menores, com menos linhas e mais denso, pois o conjunto de instruções é muito vasto e complexo, já na arquitetura RISC, isto não ocorre e os programadores precisam escrever instruções complexas a partir do conjunto de instruções simples, nativa da arquitetura. (AZEVEDO, 2014).
- Outro ponto importante é o fato de que, por possuir instruções mais complexas, o CISC pode implementar tarefas com menos instruções, o que leva a um menor uso da memória pelo programa. Já a arquitetura RISC, por possuir um conjunto de instruções mais simples, requer menos transistores e, com isso, pode-se projetar processadores menores. (*ibidem*).

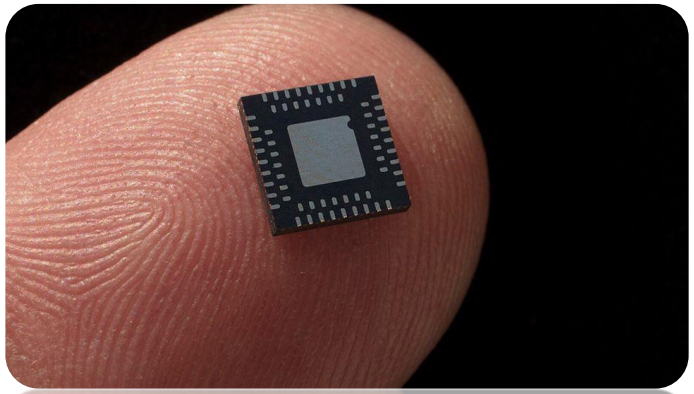
11.4.2. QUADRO COMPARATIVO: VANTAGENS E DESVANTAGENS

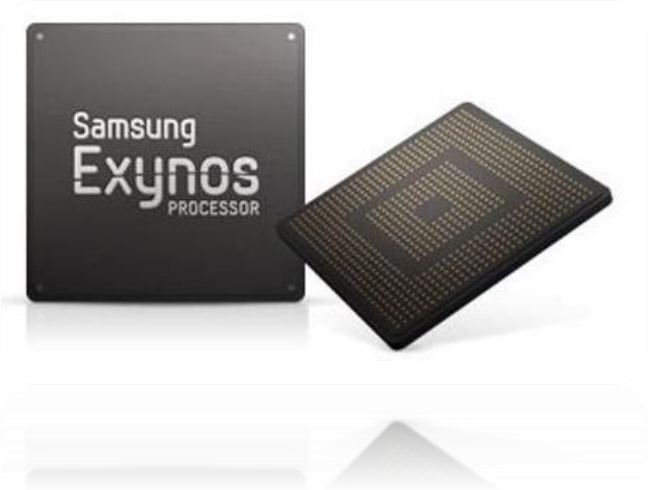
RISC	
VANTAGENS	DESVANTAGENS
• Executam instruções mais rápido porque seu método de codificação usa menos *bits*, reduzindo o tempo.	• Menos instruções requerem que instruções sejam executadas.
• As instruções são executadas diretamente pelo *hardware* e não por um *software*.	• Falta de compatibilidade com versões anteriores.
• Máquinas mais rápidas e mais baratas.	• Requer sistema de memória rápida para aumentar suas instruções.
• Simplicidade do *hardware*.	• A performance depende diretamente do código gerado pelo programador.

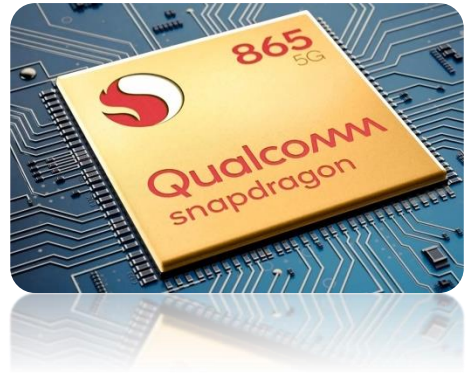
CISC	
VANTAGENS	DESVANTAGENS
• Instruções mais complexas podem redundar em código-objeto menor, menos instruções e reflexos nos custos.	• Nem sempre menos instruções acarretam em menos *bits*.
• São micro programados, trazem mais flexibilidade ao projeto de máquinas.	• Por ser micro programado, acarreta uma sobrecarga adicional de interpretação de cada instrução.
• Muitas das instruções estão guardadas no próprio processador, o que facilita o trabalho dos programadores de linguagem de máquina.	• Instruções diferentes levam quantidade diferentes de período de *clock* para executar, o que pode tornar a máquina excessivamente lenta.

11.4.3. ATUALMENTE

- De acordo com Barbosa (s/d), atualmente não podemos afirmar com exatidão que um processador utiliza apenas a arquitetura CISC ou RISC, pois os modelos atuais de processadores abrigam as características de ambas as arquiteturas.
- Processadores ARM usados em celulares são um exemplo de uso da arquitetura RISC, outro exemplo de uso dessa arquitetura estão nos consoles como o Nintendo 64 e o Playstation.

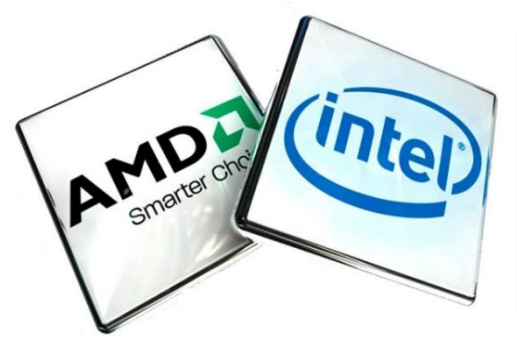


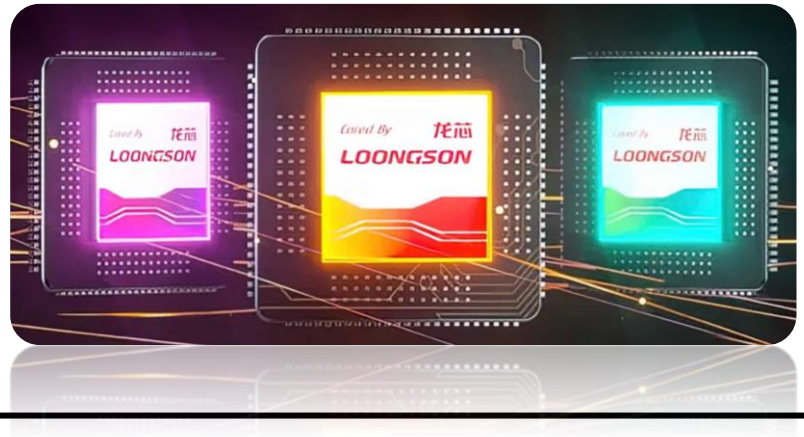




cont.[...]

- Com o passar dos anos, tanto a Intel quanto a AMD perceberam que usar alguns conceitos da arquitetura RISC em seus processadores poderia ajudá-las a criar processadores mais rápidos. Porém, ao mesmo tempo, existia a necessidade de continuar criando processadores compatíveis com os antigos. (BARBOSA, s/d).
- A ideia então passou a ser construir *chips* híbridos, que fossem capazes de executar as instruções x86, sendo compatíveis com todos os programas, mas ao mesmo tempo comportando-se internamente como *chips* RISC, quebrando estas instruções complexas em instruções simples que podem ser processadas por seu núcleo RISC. (*ibidem*).





REFERÊNCIAS

AZEVEDO, D. L. O., OLIVEIRA, N. M. Comparação entre as arquiteturas RISC e CISC. UFRPE, 2014.

BARBOSA, L. S. O. Arquitetura e Organização de Computadores. Universidade do Estado do Amazonas.

CAIXETA, D. F., VILARINHO, A. K., SANTOS, E. A., Estudo de performance das arquiteturas RISC e CISC. Um breve histórico da evolução das principais arquiteturas de hardware. FAJESU. 2009.

FÁVERO, Eliane Maria de Bortoli. Organização e Arquitetura de Computadores – Curso técnico em Informática. UTFPR – Universidade Tecnológica Federal do Paraná. 2011.

FARIAS, Gilberto. Introdução à computação. UFPB, 2013.

FERNANDEZ, M. P., Arquitetura de computadores. 3ª edição. UAB/CE, 2015.

TANENBAUM, Andrew S. AUSTIN, Todd. Organização Estruturada de Computadores. 6ª ed. Pearson, 2013.

MONTEIRO, Mário A. Introdução à Organização dos Computadores. 5ª ed. LTC. 2014.

TANENBAUM, Andrew S. VAN STEEN, Maarten. Sistemas Distribuídos. Princípios e Paradigmas. 2ª ed. Pearson, 2007.

RIBEIRO, Carlos; DELGADO, José. Arquitetura de Computadores. 2ª ed. Rio de Janeiro: LTC, 2009.

Obrigado!

Disciplina: Introdução à Computação (I.C)