



# DESENVOLVIMENTO DE UM MICROPROCESSADOR 8086 NA ARQUITETURA RISC



Dênis Araújo da Silva  
Engenharia da Computação  
Universidade Federal de Itajubá  
silvadenisaraujo@gmail.com

Marcos Aurélio F. de A. Costa  
Engenharia da Computação  
Universidade Federal de Itajubá  
macaufreitas@gmail.com

Maurílio Pereira Coutinho  
Orientador  
Universidade Federal de Itajubá  
maurilio.coutinho@gmail.com

Robson Luiz Moreno  
Co-Orientador  
Universidade Federal de Itajubá  
moreno@unifei.edu.br

## Abstract

The Reduced Instruction Set Computers (RISC) are receiving much attention in the last few years. The main concept of this architecture allows a fast execution in nowadays machines. Once we have fast memories, it isn't necessary to avoid accessing it, so short instructions which do a simple task is the better option. This way, this paper introduce the job of converting a CISC machine (iAPX8086) in a RISC one. The whole process consist in analyse and reduce the instruction set, after this some adjustments need to be done, like adapter all the instructions in order to have all of them in the same size.

## Introdução

A arquitetura CISC possui um conjunto de instruções extenso, onde cada instrução é capaz de executar uma tarefa bastante complexa. Tal formato requer instruções de tamanhos variados e que necessitam de uma diferente quantidade de ciclos de clock para serem executadas. Esta definição é válida quando se trabalha com memórias de baixa velocidade, onde cada acesso é custoso, e se torna necessário evitar acessos à memória de modo a reduzir o tempo de execução. Porém, atualmente as memórias são de alto desempenho, o que torna desnecessário o intuito de evitar acessá-las.

Por sua vez, a arquitetura RISC possui um conjunto de instruções reduzido, onde cada instrução é capaz de realizar um simples tarefa. Há instruções específicas para acessar a memória, de modo a permitir que a seguinte regra possa ser respeitada: A cada ciclo de clock uma nova instrução deve começar a ser executada. Nesta arquitetura todas as instruções possuem o mesmo tamanho e levam a mesma quantidade de ciclos de clock para serem executadas, com excessão das instruções que acessam a memória. Desta forma, este formato é interessante para o hardware disponível atualmente.

## Desenvolvimento

De acordo com (??), uma das mais importantes abstrações é a interface entre o hardware e o software de baixo nível. Por causa de sua importância, é dado uma nomenclatura especial: **arquitetura do conjunto de instruções** (ISA), ou simplesmente arquitetura de uma máquina. O conjunto de instruções, inclui qualquer coisa que programadores necessitam para saber como programar em linguagem de máquina corretamente, incluem instruções, dispositivos E/S, entre outros. Tipicamente o sistema operacional irá encapsular os detalhes da realização da E/S, alocação de memória, e outras funcionalidades de baixo nível do sistema, portanto, programadores não precisam se preocupar com estes detalhes. Dois tipos de conjuntos de instruções existentes serão explicados a diante.

**CISC** é uma arquitetura de processador, que teve como princípio o uso eficiente de memória e a facilidade de programar. Cada instrução desse processador tem várias operações em seu interior ajudando o programador a implementar programas. A maioria dos projetos de microprocessadores comuns - incluindo o Intel (R) 80x86 e séries Motorola 68K - também seguem a filosofia CISC (??).

Na década de 1980, ocorreu as mudanças para a nova arquitetura, o modelo **RISC** (Reduced Instruction Set Computer). As melhorias nas linguagens de programação, tecnologia de compiladores e custo de memória significaram que, menos programação estava sendo feita no nível do assembly, de modo que os conjuntos de instruções poderiam ser medidos pela forma como os compiladores os usavam, ao contrário de como os programadores em assembly os usavam. Praticamente todos os novos conjuntos de instruções desde 1982, seguiram essa filosofia RISC de tamanhos de instrução fixos, conjuntos de instrução load/store, modos de endereçamento limitados e operações limitadas. ARM, Hitachi SH, IBM PowerPC, MIPS e Sun SPARC são todos exemplos de arquiteturas RISCs (??).

Portanto, a **regra de ouro** é que todas as instruções devem ser executadas em apenas um ciclo de máquina e devem possuir um tamanho fixo. O problema quanto à regra de ouro são as instruções LOAD e STORE que requerem mais ciclos de relógio, uma vez que estas fazem acesso à memória. Neste ponto, a regra de ouro tem de ser adaptada. A solução proposta é que a cada ciclo de relógio a execução de uma nova instrução seja iniciada. E nessa característica é que entra o uso da técnica de **pipelining**.

Para o microprocessador desenvolvido neste trabalho as instruções possuem 4 bytes. Este tamanho de instrução foi definido visando uma melhor administração da memória disponível e respeitando a regra de ouro, onde todas as instruções tem o mesmo tamanho. Na figura 1 a seguir tem-se a estrutura das instruções aritméticas e lógicas.

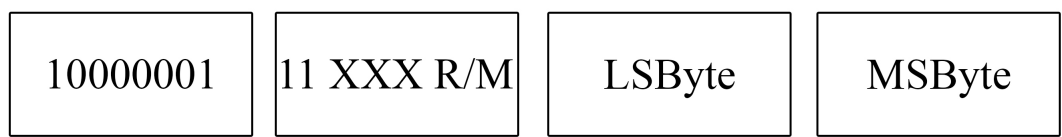


Figura 1: Bytes das Instruções Aritméticas e Lógicas

Na figura 2 a seguir tem-se a estrutura da instrução MOV.



Figura 2: Bytes da Instrução MOV

”O design do conjunto de instruções deve ter vários objetivos, sendo o mais óbvio e útil a performance do microprocessador.”(??).

Neste momento a preocupação é desenvolver um microprocessador que tenha um funcionamento básico e muito explícito, pois o foco deste trabalho não é desenvolver uma tecnologia nova, porém compreender profundamente o desenvolvimento da arquitetura e dos componentes de um microprocessador e lembrando que, as máquinas RISC só se tornaram viáveis devido aos avanços de software no aparecimento de compiladores otimizados.(??).

Para o devido gerenciamento de memória, o microprocessador a ser desenvolvido não irá ter em seu conjunto os modos que fazem uso de segmentação, portanto a memória será enxergada como uma memória linear. Como se tivéssemos somente um segmento sendo utilizado, tal definição será explicitada mais a diante. Semelhante ao funcionamento do microprocessador 386 que possui além de um modo de memória segmentada, um modo protegido o qual a memória é vista linearmente.

Portanto, após todas as considerações tomadas para o desenvolvimento do conjunto de instruções, temos na tabela 1 o conjunto de instruções escolhido para o desenvolvimento deste microprocessador.

Tabela 1: Instruções Escolhidas e seus devidos Opcodes

Instrução	Opcode
<b>ADD Reg16,Imed16</b>	10000001 11000 R/M I16L I16H
<b>OR Reg16,Imed16</b>	10000001 11001 R/M I16L I16H
<b>ADC Reg16,Imed16</b>	10000001 11010 R/M I16L I16H
<b>SBB Reg16,Imed16</b>	10000001 11011 R/M I16L I16H
<b>AND Reg16,Imed16</b>	10000001 11100 R/M I16L I16H
<b>SUB Reg16,Imed16</b>	10000001 11101 R/M I16L I16H
<b>XOR Reg16,Imed16</b>	10000001 11110 R/M I16L I16H
<b>CMP Reg16,Imed16</b>	10000001 11111 R/M I16L I16H
<b>MOV Reg16,Imed16</b>	00000000 10111 R/M I16L I16H

Para realizar a **implementação em VHDL** baseou-se nos componentes internos conhecidos do microprocessador Intel® 8086 e buscou-se a maneira de deixar mais nítido e auto-explicativo o funcionamento de cada componente desenvolvido, a seguir temos todos os componentes desenvolvidos e testados, para que no fim fossem todos unidos e controlados pela Unidade de Controle do microprocessador.

- **Registro de Propósito Geral:** O registrador de propósito geral tem como objetivo encapsular os registradores AX, BX, CX, DX, SP, BP, SI e DI, onde cada registro possui seu devido código R/M a ser substituído nas instruções aritméticas e lógicas.
- **Registro de Segmento:** Com funcionamento semelhante do Registro de Propósito Geral, possui o objetivo de encapsular os registradores que possuem o funcionamento de manipulação de memória que são, CS, ES, DS, SS, além do contador de índice IP. Diferentemente, do Registro de Propósito Geral, o registro de Segmento, possui uma linha de entrada denominada incrementa IP, tal linha foi implementada a fim de facilitar o comando de incremento de IP para se caminhar continuamente na memória, sendo que o registro possui acesso direto ao registrador IP, em vez de se desenvolver uma estrutura própria para este cálculo.
- **Calculadora de Endereço:** Foi desenvolvido uma calculadora de endereço para realizar o cálculo do endereço relativo para o endereço físico.
- **Demultiplexador e Multiplexador:** Utilizados para realizar a comunicação entre alguns componentes.
- **Registro de Flags:** O Registro de Flags tem como objetivo guardar o resultado dos Flags que são calculado pela Unidade Lógica Aritmética.
- **Unidade Aritmética e Lógica:** É a estrutura responsável por executar todas as operações aritméticas e lógicas do microprocessador. É composta por algumas estruturas auxiliares que implementam funcionalidades necessárias na execução bit a bit de cada instrução.

- **Detector do Flag Auxiliar:** É uma estrutura criada para ser utilizada na Unidade Aritmética e Lógica. Esta estrutura detecta a ocorrência de um ”vai um”do bit 3 para o bit 4 ou quando há ”vem um”do bit 4 para o bit 3, durante a execução de alguma instrução aritmética.
- **Detector do Flag de Paridade:** É uma estrutura que detecta a paridade do resultado obtido em uma instrução aritmética ou lógica. Caso haja um número par de bits 1 no resultado da operação, o flag de paridade recebe valor 1, caso contrário, recebe valor 0.
- **Detector do Zero Flag:** Esta estrutura verifica o resultado obtido em uma operação aritmética ou lógica. Caso o valor final seja 0, o zero flag recebe valor 1, caso contrário, recebe valor 0.

- **Registro de Propósito Geral**
- **Registro de Propósito Geral**

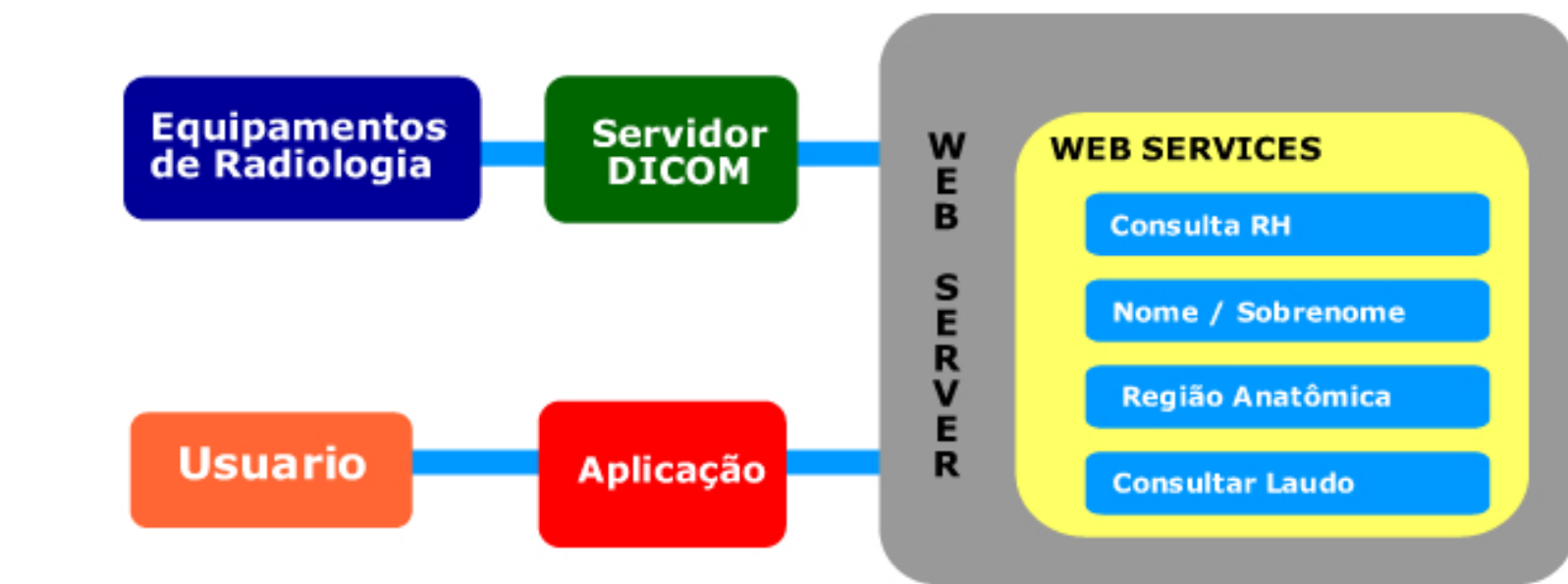


Figura 3: Funcionamento de um Web Service na implementação de um PACS.

Como já foi dito anteriormente, SOA não constitui uma tecnologia mas sim um conceito de como desenvolver sistemas que sejam facilmente interligáveis e como interligar sistemas previamente existentes. Com esse objetivo, foram criadas várias tecnologias para que sistemas desenvolvidos utilizando tecnologias diferentes pudessem comunicar-se entre si. Entre essas tecnologias podemos citar CORBA, RMI e Web Services, porém tais tecnologias usadas de forma isolada ainda apresentavam falhas, algumas por não darem suporte a diversas tecnologias, outras por serem difíceis de implementar ou até mesmo por não possuírem uma especificação que fosse seguida sempre. O maior problema dessas tecnologias era que elas só ligavam um sistema a outro, e sempre que fosse necessária a ligação de vários sistemas elas geravam uma arquitetura muito complexa para se manter, e com o surgimento de novos sistemas a arquitetura tenderia a ficar ainda mais complexa, o que as tornaria inviáveis para o sistema proposto.

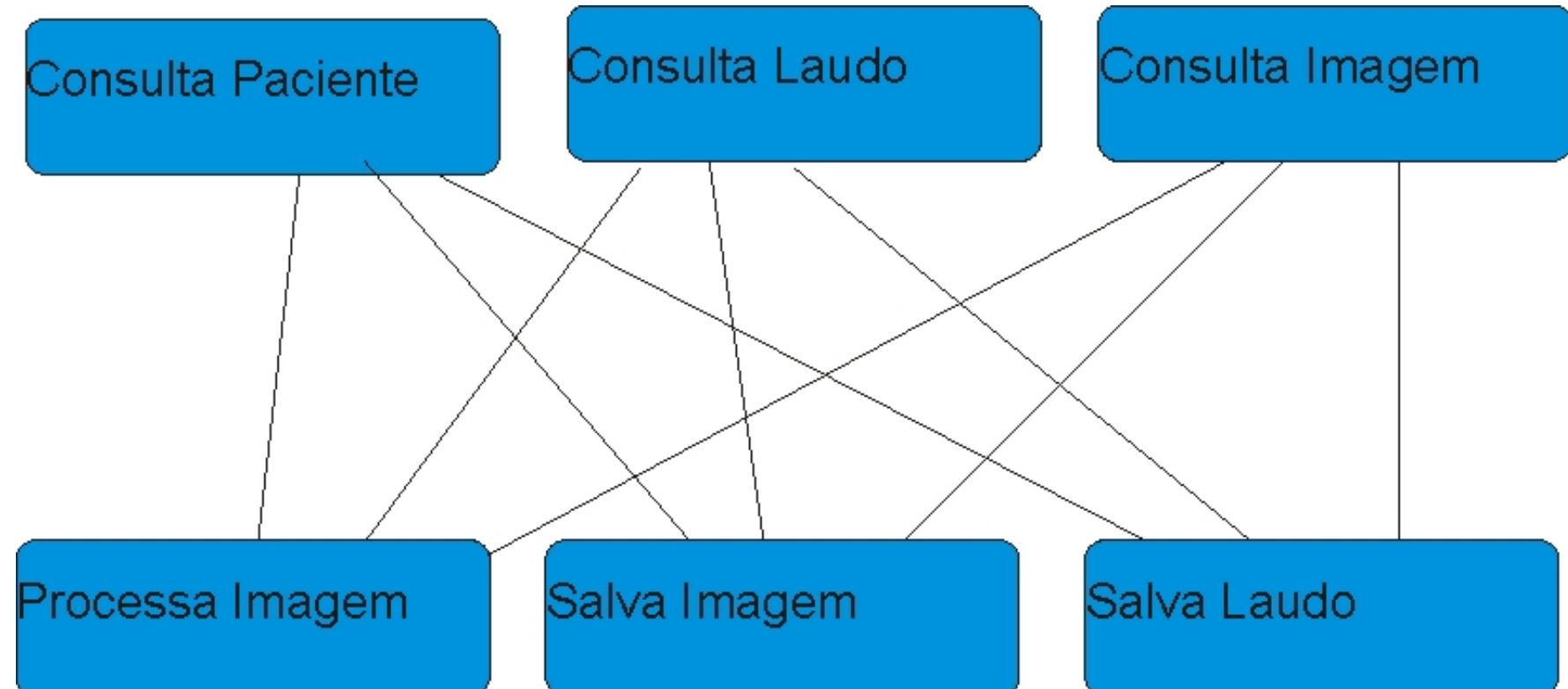


Figura 4: Arquitetura do sistema antes da implantação de um ESB.

Com esse problema, surge a necessidade de uma centralização de requisições para os serviços, e nesse contexto entra o Enterprise Service Bus (ESB), que surge como o cerne de uma Arquitetura Orientada a Serviços, pois ele torna-se o responsável pela parte de segurança do sistema, roteamento de requisições e a publicação dos serviços, tornando uma arquitetura mais “limpa”, onde todos os serviços são publicados no barramento, o que torna o sistema mais expansível. Desta forma, a cada surgimento de um novo sistema, só será necessário publicá-lo como novos serviços no barramento, não necessitando a alteração dos outros serviços pra que possam integrar-se com o novo sistema.

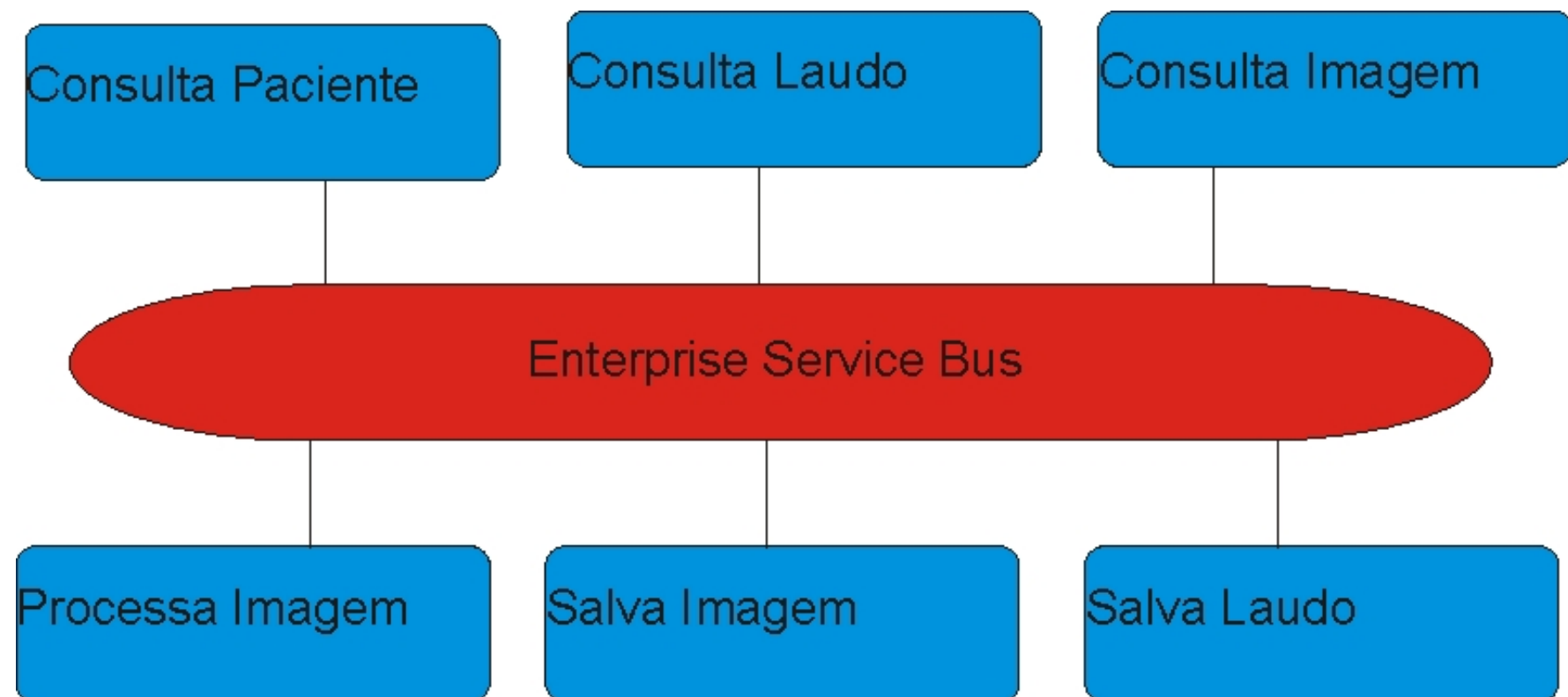


Figura 5: Arquitetura do sistema após a implantação de um ESB.

## Conclusão

Conforme descrito neste documento, a adaptação de uma máquina CISC para RISC é um trabalho válido, onde se tem uma redução da quantidade de instruções disponíveis e a diminuição no tempo de execução de cada uma.

O microprocessador resultante deste trabalho é uma máquina eficaz e de simples programação. Estão disponíveis as principais instruções necessárias, com as quais é possível escrever programas que executam as mais diversas tarefas. O desempenho destes programas é de certa forma incrementado e o resultado obtido é considerado satisfatório.

Uma boa comparação a ser feita é a análise do código do sistema operacional DOS da Microsoft. Onde as instruções mais utilizadas são as que constam na máquina desenvolvida neste documento.