UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO

(Bacharelado)

TUTORIAL DA LINGUAGEM ASSEMBLY UTILIZANDO O VXT

TRABALHO DE CONCLUSÃO DE CURSO SUBMETIDO À UNIVERSIDADE REGIONAL DE BLUMENAU PARA A OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA COM NOME EQUIVALENTE NO CURSO DE CIÊNCIAS DA COMPUTAÇÃO — BACHARELADO

MARILENE LINZMEIER

BLUMENAU, JUNHO/1999

1999/1-45

TUTORIAL DA LINGUAGEM ASSEMBLY UTILIZANDO O VXT

MARILENE LINZMEIER

ESTE TRABALHO DE CONCLUSÃO DE CURSO, FOI JULGADO ADEQUADO PARA OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA DE TRABALHO DE CONCLUSÃO DE CURSO OBRIGATÓRIA PARA OBTENÇÃO DO TÍTULO DE:

BACHAREL EM CIÊNCIAS DA COMPUTAÇÃO

_____

Prof. Antônio Carlos Tavares — Orientador na FURB

_____

Prof. José Roque Voltolini da Silva — Coordenador do TCC

BANCA EXAMINADORA

_____

Prof. Antônio Carlos Tavares

_____

Prof. Roberto Heinzle

_____

Prof. Dalton Solano dos Reis

A meus pais, Renato e Sofia, que não pouparam esforços para a conclusão deste curso e de todos os projetos da minha vida.

AGRADECIMENTOS

Ao meu orientador, professor Antônio Carlos Tavares, pelo esforço e dedicação para a conclusão de trabalho.

Ao amigo Cleison Vander Ambrosi, pelas sugestões e idéias dadas no decorrer do desenvolvimento do protótipo, bem como pelas aulas particulares do ambiente Delphi 3.0.

E a todas as pessoas que me apoiaram e incentivaram durante a minha vida para que eu pudesse chegar até aqui.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Este trabalho consiste na implementação de um protótipo de uma ferramenta para auxílio na consulta de especificações técnicas referentes a linguagem de programação *Assembly*. Baseia-se na sistemática do funcionamento do *Hint* do Windows, apresentando uma janela contendo informações mais detalhadas sobre a linguagem e permitindo o acesso a outras funções do tutorial.

ABSTRACT

This work consists of the implementation of an archetype of a tool for aid in the consultation of specifications referring techniques the *Assembly* programming language . It is based on the systematics of the functioning of the *Hint* of Windows, presenting a window that contains more detailed information on the language and allowing the access to other functions of the tutorial one.

1 INTRODUÇÃO

Atualmente, muito se fala a respeito da técnica de Treinamento Baseado em Computador (CBT), que segundo [MDS96], é uma alternativa de ensino mundialmente reconhecida. Esta técnica nos últimos anos, com a necessidade de aperfeiçoamento profissional e educacional constante, ganhou força como a solução ideal para treinamento de grandes quantidades de pessoas.

CBT tem obtido um grande resultado pelos seguinte motivos:

a) possibilita usuário estudar sozinho, no seu próprio ritmo e no momento que melhor lhe convier; o instrutor funciona, portanto, como apoio ao aprendizado;
b) permite ao usuário rever uma lição quantas vezes for necessário, reforçando a assimilação;
c) a filosofia de conceito/tarefa leva o usuário a uma constante interação com a máquina;
d) o usuário só avança para novos estágios quando ele próprio sente-se seguro para isso.

Esta técnica será aplicada na elaboração do presente trabalho, que terá como finalidade o desenvolvimento de um tutorial de *Assembly* (que traduzido do inglês significa montagem).

Assembly é uma linguagem de baixo nível, que pode ser utilizada em programas para acelerar tarefas lentas. Basicamente ela consiste de sentenças que representam instruções em linguagem de máquina (mnemônicos) e, como ela está próxima ao código de máquina, ela se torna rápida [BAS96].

Seguem abaixo as razões para programar em *Assembly*, segundo [BAS96]:

a) trabalhar com o *Assembly* é a melhor maneira de conhecer o funcionamento do seu computador, o que permite o desenvolvimento de programas de forma mais consistente;
b) fazendo uso do *Assembly* pode-se ter um controle maior sobre o computador;
c) *assembler* (montador) permite uma otimização ideal nos programas, seja no seu tamanho ou execução.

1.1 JUSTIFICATIVA

A idéia para a construção deste Trabalho de Conclusão de Curso surgiu na cadeira de Sistemas Operacionais I, do curso de Bacharel de Ciências da Computação desta Universidade, quando foi solicitado aos acadêmicos a elaboração de um trabalho didático baseado no simulador virtual do processador da Intel 8086 (VXt), que emula o funcionamento interno da máquina, mostrando na tela as instruções em *Assembly* de um código executável fornecido pelo usuário.

Durante o contato com este software surgiu a hipótese de que o usuário necessitasse obter informações técnicas sobre as instruções apresentadas da Linguagem *Assembly*, e para que os mesmos não precisassem consultar livros e manuais técnicos perdendo um certo tempo, surgiu a idéia de desenvolver um Tutorial Inteligente de *Assembly* para o VXt, facilitando assim a pesquisa das instruções.

1.2 OBJETIVOS

O presente trabalho tem como objetivo principal, o desenvolvimento de um software tutorial da Linguagem *Assembly* .

Além deste, existem os seguintes objetivos secundários:

a) estudo da técnica CBT;
b) analisar as características da Linguagem *Assembly*;
c) pesquisar alguns softwares tutoriais e fazer comparações;
d) implementação de um protótipo de tutorial de *Assembly* para o VXt.

1.3 ORGANIZAÇÃO DO TEXTO

O presente trabalho está dividido em cinco capítulos. Sendo que o primeiro capítulo trata dos objetivos do trabalho, justificativas, limitações e organização.

O segundo capítulo, aborda o assunto educação. Dentro deste contexto, o capítulo aborda mais especificamente o Computador na Educação e Treinamento. Comenta-se a introdução dos computadores em sala de aula, a atual situação e suas tendências.

O terceiro capítulo trata sobre softwares tutorias, traz algumas características e vantagens.

O quarto capítulo mostra uma apanhado geral da arquitetura do microprocessador da família iAPX - 86/88, a Linguagem *Assembly* e o simulador do processador 8086 - VXt.

O quinto capítulo, detalha a construção do protótipo, apresenta as características e exibe algumas telas.

No sexto capítulo, são feitas conclusões do trabalho, algumas considerações e sugestões para futuros trabalhos.

2 INFORMÁTICA NA EDUCAÇÃO

Em plena virada para o século XXI, praticamente nada se modificou no ensino de forma geral. O aluno ainda é considerado um adulto em miniatura, que raciocina como adulto, mas que é desprovido de conhecimentos básicos e experiências. Tendo em vista este fato, a escola tradicional utiliza o ensino verbalista, que por sua vez estimula a passividade intelectual, por isso é livresco, memorístico e enfatiza apenas a aquisição de conhecimentos, em detrimento do desenvolvimento do espírito lógico e crítico dos alunos [BAC94].

A seguir descreve-se um breve histórico da informática na educação, sua situação atual e as tendências.

2.1 HISTÓRICO

A História da Informática na Educação no Brasil data de mais de 20 anos, nasceu a partir de algumas experiências nas Universidades do Rio de Janeiro (UFRJ), Rio Grande do Sul (UFRGS) e na UNICAMP. Desde o início do programa, a decisão da comunidade de pesquisadores foi a de que as políticas a serem implantadas deveriam ser sempre fundamentadas em pesquisas pautadas em experiências concretas, utilizando a escola pública, prioritariamente, o ensino de 2° grau [VAL97a].

Desde a década de 70, a Educação no Brasil tenta alcançar o *status* de tratamento científico com o uso de tecnologia instrucional. Muito se fez para que os alunos tivessem um ensino individualizado, onde cada aluno seguiria de acordo com seu próprio ritmo e tivesse à sua disposição um *feedback* sobre as respostas desejadas [BAC94].

Quando notou-se que o computador poderia vir a contribuir no processo de ensino-aprendizagem, começou a ser desenvolvida a Política de Informática Educativa. Iniciada na década de 1980, esta política buscou desenvolver mecanismos para inserir o computador no processo de ensino-aprendizagem, na expectativa de que, com sua utilização, pudesse ser garantido um ensino de melhor qualidade [OLI97].

Foi também durante esta década que empresas e órgãos públicos tentaram constituir uma outra informática na comunidade. Até 1988, estas experiências sofreram descontinuidades, sendo que somente em 1989 é que o Ministério da Educação e Cultura

(MEC) aprovou as políticas de informática na educação. Porém, apenas a partir de 1991 os orçamentos foram aprovados, para que as verbas, destinadas à informática na educação, chegassem em 1992 [BER96].

Como resultados das décadas de 70 e 80, pode-se citar, conforme ressalta [ANG95]:

a) formação de equipes interdisciplinares de especialistas em educação, isto é, equipes onde existam profissionais da educação, profissionais da área de computação, além de especialistas da disciplina a qual será tema do software;
b) integração de pesquisadores de universidades com professores de educação básica;
c) investigação sobre as limitações e possibilidades dos softwares educacionais, embora tenha sido pequeno o número deste tipo de software desenvolvido;
d) criação de laboratórios de informática em escolas públicas para o ensino de 1°, 2° e 3° graus;
e) comprovação de resultados de pesquisas (ambientes alfabetizadores, educação especial, robótica, automação, matemática etc.) aplicadas em escolas;
f) contatos com experiências internacionais;
g) conscientização dos técnicos para desenvolvimento e pesquisa de sistemas para educação;
h) desenvolvimento de protótipos para multiplicação e disseminação em Educação Básica.

A partir de 1990, criaram-se Centros de Informática Educativa para atender às escolas garantindo tecnologia da informática para a população. Além disto, estes centros visam prover a formação dos professores para que estes saibam utilizar a tecnologia disponível [BER96].

2.2 SITUAÇÃO ATUAL

A informática por ser uma Ciência muito nova, ainda não despertou todas as suas reais possibilidades para auxílio na área da Educação. Porém o software educacional adentrou o mercado nacional nestes últimos três anos, com uma força muito grande, devido a grande necessidade de se utilizar ferramentas de apoio para desenvolvimento, bem como o crescente uso da *Internet* [BER96].

Os softwares educacionais, definidos por [CAM95] como material educacional para microcomputadores, já estão sendo desenvolvidos a nível de pesquisa e procuram facilitar a vida dos educadores.

Segundo [ANG95], a implantação da Informática na Educação consiste basicamente de quatro componentes: o computador, o software educativo, o professor capacitado para usar o computador como ferramenta educacional e o aluno. Para que o computador seja utilizado como meio de ensino é preciso que os quatro componentes acima citados sejam integrados, não podendo faltar nenhum pois todos tem seu grau de importância.

2.3 TENDÊNCIAS

Não basta introduzir os computadores em uma escola. É preciso, além disso, direcionar a escola, alterar sua organização. É necessário que a introdução da tecnologia ofereça ao aluno uma aprendizagem mais natural, que a avaliação de desempenho torne-se uma auto-promoção de desenvolvimento pessoal, um desafio [BAC94].

Almeja-se que grupos de alunos, de idades diferentes, de diferentes níveis de escolaridade, possam desenvolver seus projetos cooperativamente. As informações coletadas por eles podem ser armazenadas em banco de dados para serem internamente confrontadas em experiências simuladas por eles mesmos.

O conhecimento não deve ser passado através, apenas, de livros didáticos. Os alunos devem se apropriar do conhecimento na medida de suas condições pessoais [BAC94]. Além disso, as redes de comunicação permitirão o relacionamento direto entre os alunos de diferentes escolas e de diferentes países, uma vez que atualmente, pelo menos nas escolas públicas, isto ainda não é uma realidade completa.

Adicionalmente, o ensino à distância, tendo como apoio os software educacionais, permitirá o acesso de um grande número de alunos a grandes especialistas nos mais variados ramos da atividade [BER96].

2.4 O COMPUTADOR NA EDUCAÇÃO

Segundo [ANG95], o computador surgiu inicialmente nas escolas, como uma forma de apoio ao trabalho administrativo, e não na formação do aluno. A utilização de software como ferramenta auxiliar no processo ensino aprendizagem é, portanto, recente. Em função disso, as escolas tem encontrado dificuldades em utilizar adequadamente os softwares educacionais em sala de aula, pois têm tratado a Informática como um fim em si mesmo.

Assim, em geral, as crianças são submetidas a cansativas aulas sobre hardware e sistemas operacionais, o que não está de acordo com seus interesses imediatos. Ao invés disso, o computador deve ser utilizado para auxiliar o aprendizado do aluno nas disciplinas curriculares, como Geografia, História e, hoje, para pesquisa em qualquer área via *Internet*. Dentro, deste contexto, a grande dificuldade da utilização da Informática no processo de ensino aprendizagem é a própria resistência dos professores e não do aluno [BER96].

O uso do computador como meio de ensino nas escolas vem crescendo gradativamente de ano para ano. O êxito só não é maior por falta de equipamento nas escolas e, portanto, a falta de um maior empenho na introdução da informática na educação. A formação de professores para implantar as transformações pedagógicas almejadas exige uma nova abordagem que supere as dificuldades em relação ao domínio do computador e ao conteúdo que o professor ministra. Os avanços tecnológicos têm desequilibrado e atropelado o processo de formação fazendo com que o professor sinta-se eternamente no estado de "principiante" em relação ao uso do computador na educação [VAL97b].

Aproveitando as qualidades do computador como meio para criar ambientes educativos que acrescentem valor aos meios tradicionais de aprendizagem, convém destacar as seguintes qualidades que diferenciam o ensino com o computador de outros meios de aprendizagem:

a) computador tem capacidade para armazenar, processar e apresentar “informação multimídia” de forma “interativa”; assim, é possível criar contextos para aprendizagem nos quais pode-se dar uma relação de diálogo com o nível concreto ou abstrato requeridos, sob controle do usuário ou do programador, segundo a conveniência;

b) computador pode agir com diversos níveis de "inteligência adquirida". A inteligência não é um atributo dicotômico (existe ou inexiste), mas uma qualidade que pode apresentar diversos níveis de desenvolvimento:
   - no nível mais básico apenas pode "dizer" ao aprendiz se a resposta dele é ou não correta;
   - gradualmente, porém, pode realizar outras tarefas, tais como adaptar os exercícios, dependendo das características e da performance do aprendiz, dar explicações ou sugestões derivadas do processo, ou mesmo resolver exercícios propostos pelo estudante;

c) computador viabiliza diferentes níveis de interação. Graças a sua capacidade para processar informação, aos avanços na inteligência artificial e às interfaces de diversos tipos, das quais pode lançar mão, o computador pode fazer viável uma interação de grau zero, *ENTERatividade* (o usuário limita-se a apertar ENTER para continuar, mas o controle da ação está com o programador) até o grau máximo *INTERatividade* (na qual há interação de diálogo entre a máquina e o usuário, em virtude da qual o aprendiz está em controle do que acontece, dentro dos condicionamentos do micromundo em que se desenvolve a ação).

Possibilita a conexão e a articulação com outros meios e recursos para a aprendizagem, permitindo assim a criação de ambientes cooperativos de aprendizagem, o aproveitamento das qualidades únicas de outros meios (transmissivos, experienciais, interativos) e a criação de ambientes educativos multimídia [VAL97b].

2.5 TREINAMENTO BASEADO EM COMPUTADOR

A metodologia de treinamento baseada em computador como meio instrucional já se tornou uma idéia fixa, pois muito se discute a respeito do computador como meio de ensino.

Treinamento baseado em computador (CBT) é uma técnica que quando aplicada visa descentralizar o treinamento, permitindo maleabilidade na montagem de turmas e possibilidade de horários de atendimento personalizados ao treinando, além de que o mesmo poderá aprender sozinho. Está técnica também possibilita que o treinando aprenda mais rápido, pois terá um microcomputador para tirar suas dúvidas [MDS96].

CBT modernos são interativos e utilizam recursos de multimídia (som, animação). Por ser interativo, o CBT possibilita que a informação seja absorvida na metade do tempo que num treinamento feito com métodos tradicionais, isso significa que as pessoas assimilam mais rapidamente as informações. CBT estão disponíveis 24 horas por dia, 7 dias por semana, à noite, em feriados e fins de semana, sem necessidade de planejamento de sala de aula ou de instrutor. CBT muitas vezes é aplicado individualmente, porque o treinando não fica preocupado com a pressão dos colegas, medo ou vergonha de fazer uma pergunta boba [MOC98].

Segundo [MDS96] atualmente, existem no mercado produtos de software educativo que fornecem algumas facilidades, tais como:

a) educação;
b) treinamento;
c) promocionais para feiras e eventos;
d) manuais eletrônicos com recursos hipermídia;
e) divulgação institucional de empresas;
f) quiosques de consultas;
g) apresentações interativas;
h) relatórios anuais.

3 SOFTWARE EDUCACIONAL

Os softwares educacionais existentes atualmente no mercado nacional podem ser classificados nos seguintes tipos, segundo [DET96]:

a) software para exercícios, onde a principal finalidade é fazer revisões do conteúdo disciplinar já visto em sala de aula. Os software para exercícios são desenvolvidos mais especificamente para que o aluno, já fora da sala de aula, possa rever o seu material de estudos de uma forma diferente daquela apresentada em sala de aula. O aluno pode, até mesmo em casa, aprimorar o seu conhecimento;
b) softwares para avaliar as reais capacidades de aprendizado dos alunos, analisando os erros e avaliando os estilos de cada um. Neste caso, os softwares julgam os alunos através de questões referentes a determinado assunto ou disciplina, quantificando o assunto aprendido entre uma nota que pode variar conforme for o seu rendimento defronte as questões apresentadas e suas respectivas resposta;
c) softwares educacionais, basicamente jogos, levando como principal fundamentação o interesse do aluno para o aprendizado. As crianças precisam aprender sozinhas ao invés de serem ensinadas.

Outro tipo de software educacional bastante interessante é o que tem por objetivo treinar o professor. Ele, além de demonstrar a teoria, ainda propõe exercícios que o professor pode realizar para os alunos em sala de aula, oferece sugestões de exercícios, dicas e demonstra como os conceitos são percebidos pelos alunos de forma geral e o que o professor deve fazer para orientar os alunos na construção dos exercícios [DET96].

3.1 ASPECTOS ERGONÔMICOS

Para que a interface de um sistema educacional seja bem sucedida, o mesmo deve ser coerente na determinação da aparência e do comportamento. Grande parte das ferramentas utilizadas para desenvolvimento de software educacionais, que utilizam recursos de multimídia, oferecem ferramentas para desenhar e desenvolver uma interface gráfica bastante amigável ao usuário [BER96].

Especialistas em multimídia afirmam que deve-se introduzir metáforas do mundo real para induzir as pessoas a executarem determinada tarefa no micro. Exemplos do mundo real são mais facilmente entendidos pelas pessoas [BER96].

Usuários gostam de ter o controle da situação, portanto, deve-se evitar comandos ocultos e combinações de clique do mouse com teclas incomuns. Quando desenvolve-se uma interface, deve-se fazê-la de forma que o usuário não necessite de manuais para aprender a utilizá-la [BER96].

3.2 RECOMENDAÇÕES ERGONÔMICAS

Segundo [VAU94], os softwares, deveriam seguir as seguintes recomendações:

a) Seqüência das operações: a seqüência das operações é a relação direta entre a seqüência de operações fixada pelo programa de computador e a ordem necessária ao operador para realizar a sua tarefa em qualquer ambiente. O problema que se apresenta é a preferência que se dá ao desenvolvimento da interface segundo o desenrolar da tarefa prevista, sem levar em consideração o nível de conhecimento do usuário (usuários novatos, intermediários ou experientes) ou os problemas advindos do ambiente, situação esta a mais comum:
   - as informações devem estar disponíveis de forma tão ou mais fácil que o método manual;
   - deve ser possível ao usuário abandonar ou anular a operação corrente a qualquer momento;
   - o usuário deve ser capaz de interromper a operação em curso para realizar alguma atividade paralela de curta duração (por exemplo, consultas) ou então para retomá-la mais tarde;
   - informações devem ser dadas ao usuário quando ele acaba de realizar alguma operação de forma que o mesmo possa localizar-se facilmente após uma interrupção;
   - o usuário deve ter recursos capazes de transferir informações de uma operação para outra;

b) A linguagem de interação: a linguagem de interação define a linguagem através do qual haverá o diálogo entre a aplicação e o usuário. Existem diferentes aspectos que deverão ser considerados:
   - vocabulário: deve ser compatível com o empregado pelos especialistas na área de utilização e não o vocabulário dos especialistas em Informática (caso mais comuns);
   - sintaxe: as regras que definem a forma de expressar os comandos ou os dados, devem apresentar duas características básicas: devem ser simples para facilitar a memorização e homogêneas para diminuírem as chances de erros;

c) Os dispositivos de entrada: a utilização dos diferentes dispositivos de entrada (mouse, teclado, canetas óticas, joysticks, microfones, monitores sensíveis ao toque) é separada conforme as atividades básicas a serem desempenhadas. Para a entrada de dados pelo teclado, os comandos são indicados por palavras-chaves de sintaxe definidas (ENTER, DELETE, ESC) e de funções. O uso das teclas de funções deve ser bastante coerente e seguir os mais adotados nas ferramentas existentes;

d) Os dispositivos de apresentação: a forma mais eficiente de busca é a seletiva. No projeto de interface o projetista deve manter o máximo de homogeneidade possível dentro e entre as aplicações de forma que o usuário consiga criar uma imagem mental clara da localização das informações. O usuário busca as informações de forma sistemática, quando ele não conhece a localização da informação procurada. Existem alguns tipos mais comuns de busca: a busca de alto para baixo e da esquerda para a direita, busca por colunas e a busca seletiva que é a busca baseada no modelo mental, onde o usuário já possui algum conhecimento da aplicação;

e) O tempo de resposta: o maior problema ligado aos tempos de resposta refere-se ao desconforto causado no usuário por tempo de resposta às operações solicitadas muito longos ou muito curtos. Para operações semelhantes, o usuário normalmente espera um tempo de resposta compatível. A falta de um retorno por parte do software causa no usuário ansiedade em saber se a demora é causada por restrições de ordem técnica ou por um erro de operação. Para tempos de resposta que excedam a 2 segundos é necessário que o sistema proporcione um retorno ao usuário informando-lhe que a operação levará algum tempo. Uma boa prática é

enviar mensagem do tipo “Aguarde...”. Deve-se ainda informar em que ponto encontra-se a execução da tarefa (costuma-se muito usar recursos visuais estilo “termômetro”);

f) Tratamento de erros: um erro na execução de uma tarefa é definido como a ocorrência de algo que não era previsto. Em aplicações informatizadas a carga de responsabilidade sobre o erro acaba caindo sobre a ponta do processo, o usuário. Entretanto, a carga deve ser repartida entre todos os participantes do desenvolvimento e utilização da aplicação. Se o usuário realiza um erro, na maioria das vezes é porque ele não foi adequadamente treinado ou porque as mensagens e auxílio disponíveis no software não foram suficientemente claros. O sistema deve prover informações capazes de fazer com que o usuário possa compreender e reparar a situação de erro. Quando um erro ocorre devem ser dadas o mais rápido possível informações sobre a natureza do erro e o que pode ser feito de imediato para repará-lo. Um outro ponto importante que o sistema deve possuir é ser capaz de desfazer uma alteração, um comando do estilo “Undo” ou então “Desfazer”. Quando uma situação não pode ser desfeita, o usuário deve ser alertado antes de executá-la;

g) Condução: a condução refere-se ao nível de auxílio fornecido ao usuário durante a utilização do software. O fornecimento de informações claras sobre o que está ocorrendo é um fator que pode reduzir em muito os erros. Além disso, são importantes as informações que dizem as opções que estão disponíveis para o usuário e qual foi o encadeamento de ações que conduziu ao ponto onde o mesmo se encontra.

Abaixo, são citadas algumas recomendações para a elaboração ergonômica de software, conforme cita [CAM95]:

a) fontes para títulos devem ser uniformes e em negrito para serem facilmente lidas;
b) quando o texto estiver sobre um fundo preto, este deve estar com cor clara, como branco ou uma cor luminosa;
c) sombreamento deve ser utilizado para separar o texto da imagem do fundo;
d) evitar usar texto preto ou colorido sobre um fundo branco;
e) preferencialmente, não utilizar negrito para ressaltar o texto;

f) quando o texto for sublinhado, as linhas devem ter pelo menos dois pixels de largura;
g) procurar utilizar linhas paralelas, caixas e círculos fortes com limitação. Quando utilizá-los, desenhar com linhas grossas e em tamanho grande;
h) evitar usar cores muito quentes (vermelho, amarelo);
i) cores vizinhas devem ser diferentes em intensidade, isto é, não utilizar vermelho e azul em mesma intensidade;
j) quando existir movimento de títulos, trazer os textos lentamente, deixá-los na tela por um espaço de tempo suficiente e só depois apagar;
k) as telas de títulos não devem ser muito cheias, é preferível usar mais páginas.

3.3 QUALIDADE DE SOFTWARE

O tema qualidade de software é motivo de preocupação para todos os desenvolvedores, demonstrando a importância que o tema vêm obtendo nas empresas. Neste sentido, foi elaborada uma norma internacional, ISO/IE 9126 (*International Organization for Standardization/International Electrical*), que trata justamente das características que os softwares devem possuir.

Qualidade é um conceito multidimensional que envolve tanto as pessoas que desenvolvem software, quanto aquelas que farão uso do mesmo. Pode-se definir qualidade de software como o conjunto de propriedades a serem satisfeitas em um determinado grau, de modo que o software satisfaça as necessidades de seus usuários. Entretanto qualidade não pode ser definida universalmente, ou seja, deve ser definida para um item/elemento específico. Para cada produto de software existe um conjunto de características de qualidade mais adequado. Esse conjunto dependerá da natureza e do uso que se espera fazer do produto [VAL97b].

3.4 QUALIDADE DE SOFTWARE EDUCACIONAL

Segundo [ROC93], o desenvolvimento de software educacional possui características próprias e seus objetivos da qualidade devem ser definidos na fase de análise dos requisitos. O software deve refletir a teoria de aprendizagem especificada pelo professor, o que torna-se um requisito educacional.

Um software deve ser desenvolvido para atender as necessidades do usuário, deve ter vida longa, útil e produtiva. Para isso ele deve atender alguns objetivos, conforme os descritos abaixo por [ROC93]:

a) confiabilidade conceitual: o produto necessita satisfazer às necessidades e requisitos que motivam sua construção;
b) confiabilidade de representação: refere-se às características de representação do produto que afetam sua compreensão e manipulação;
c) utilizabilidade: determina a conveniência e a viabilidade de utilização do produto ao longo de sua vida útil.

O objetivo confiabilidade de representação refere-se às características relacionadas à necessidade do software estar representado de modo a facilitar o seu entendimento e manipulação pelos diferentes tipos de usuários. A confiabilidade de representação é atingido através de dois fatores descritos abaixo por [ROC93], [VAL97b]:

a) legabilidade: avalia a possibilidade de diferentes pessoas entenderem o programa com relativa facilidade para que possam utilizá-lo. Relacionados a este fator tem-se os subfatores: clareza e concisão;
b) manipulabilidade: avalia a possibilidade de diferentes pessoas manipularem o programa com facilidade. É atingido através de três subfatores: disponibilidade, estrutura e raestrabilidade.

O objetivo Confiabilidade Conceitual segundo [VAL97b], refere-se às características que garantem que um produto de software atende a seus requisitos, no que se refere ao seu conteúdo. Portanto, avaliar e garantir a confiabilidade conceitual está relacionado a avaliar e garantir a confiabilidade do conteúdo das informações do sistema.

Para que a informação seja transmitida de forma adequada, os textos, imagens ou qualquer outra mídia utilizada devem refletir a realidade a que o sistema se refere. Obviamente algumas adaptações necessitam ser realizadas para que as informações muito técnicas ou complexas possam ser transmitidas de forma didática e compreensível, com o nível adequado de detalhes para o seu entendimento do problema. Além disso, os software educacionais, devem possuir uma documentação que apoie o seu desenvolvimento e manutenção. Desta forma, considerando-se o ponto de vista do desenvolvedor/mantenedor de

software, esse objetivo se refere, também, à qualidade do conteúdo da documentação gerada durante o desenvolvimento e manutenção do sistema [VAL97b].

Utilizabilidade, segundo [VAL97b] e [ROC93], é um objetivo fundamental para a qualidade de um produto de software e considera as diferentes formas de sua utilização, durante as fases de desenvolvimento e uso. Este objetivo exige a existência dos outros dois: confiabilidade da representação e confiabilidade conceitual. Diversos fatores estão relacionados a esse objetivo:

a) manutenibilidade: avalia a facilidade com que o programa pode ser adaptado a fim de atender as necessidades de modificação que surgem depois do seu desenvolvimento;
b) operacionalidade: avalia a facilidade de comunicação com o usuário. Este fator possui dois subfatores: oportunidade e amenidade ao uso;
c) portabilidade: característica de um programa poder ser operado de maneira fácil e adequada em diferentes configurações de equipamento além do original;
d) reutilizabilidade: característica de avalia a possibilidade do reaproveitamento, total ou parcial, de funções desenvolvidas em um programa em outras aplicações;
e) eficiência: característica de o programa realizar suas funções sem desperdício de recursos (memória, periféricos, outros);
f) rentabilidade: característica de o programa ter uma relação custo-benefício aceitável;
g) avaliabilidade: característica que avalia a facilidade com que o programa pode ser avaliado. Este possui dois subfatores: verificabilidade e validabilidade.

3.5 EXEMPLOS DE SOFTWARE EDUCACIONAL

Foram analisados três softwares tutorias: um Tutorial sobre o nível físico de uma rede de computadores, um Tutorial Básico de SQL, e um Tutorial de *Assembly*. Estes softwares foram localizados através da Internet e seus respectivos endereços estão indicados em cada tópico seguinte.

3.5.1 TUTORIAL DE ASSEMBLY

Este tutorial em HTML também está disponível com a extensão .txt no endereço: http//www.inf.ufsm.br/~amaral.

Neste tutorial tem-se todos os passos para o desenvolvimento de um programa escrito em *Assembly*, como estruturar, como transformar o fonte num programa objeto e, como gerar um executável. Traz também uma explicação de todas as instruções com exemplos, e alguns programas prontos.

A estrutura desta página, figura 1, segue uma seqüência lógica das operações, tendo facilidade de navegar sobre ela, ir para algum tópico e poder voltar, sua linguagem é de fácil entendimento ao usuário. Este tutorial é apenas para fazer consultas não havendo interação com o usuário, por isso não há tratamento de erros. Através do índice é possível avançar para o tópico a que ele está relacionado, não há marcadores de tempo para esta resposta, ou seja, se este avanço for muito demorado o usuário terá que esperar sem saber quanto tempo vai levar. Pelo índice ou por palavras chaves que ligam a algum tópico pode-se navegar sobre o tutorial, mas não tem um controle de onde foi que partiu.

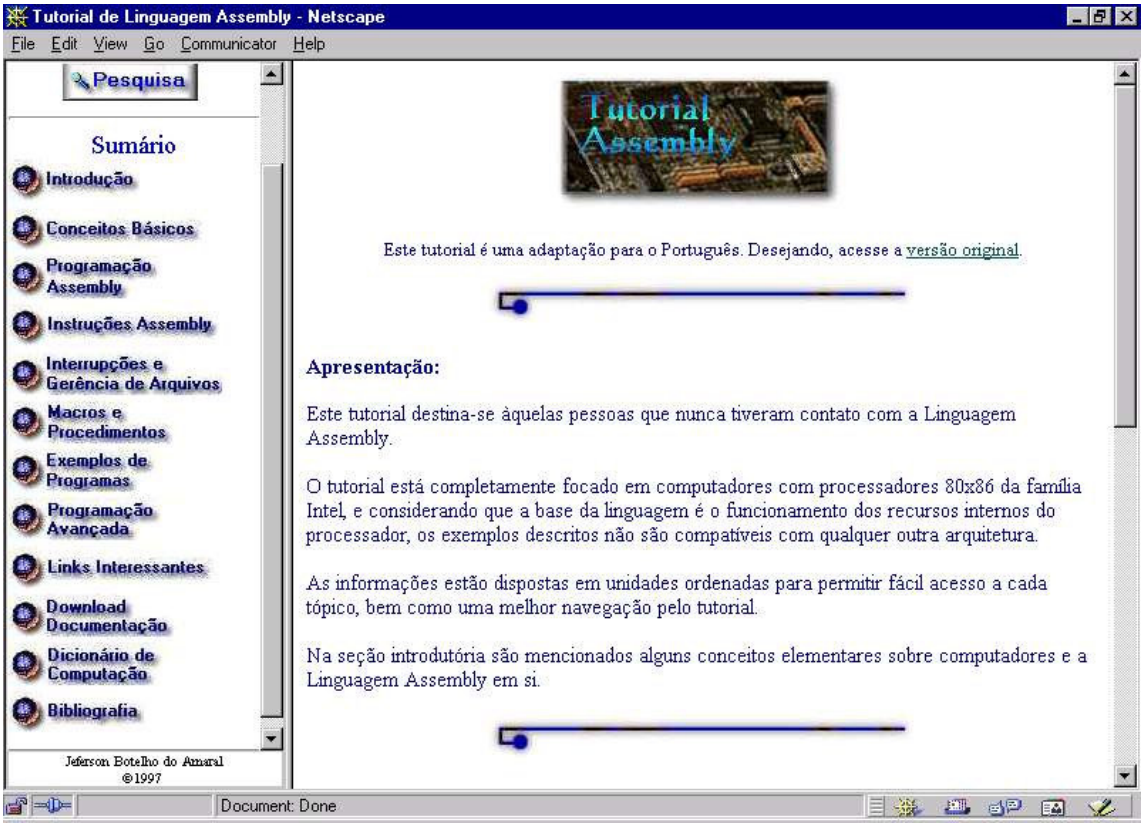

Figura 1: Página inicial do Tutorial de *Assembly*

3.5.2 TUTORIAL SOBRE O NÍVEL FÍSICO DE UMA REDE DE COMPUTADORES

Este tutorial está disponível no seguinte endereço: http://www.uel.br/adm/proenca/curso-redes-graduacao/1998/trab-01/equipe-06/ponte.htm.

Neste tutorial temos a explicação de todas as características de redes de computadores, de maneira clara e concisa, de fácil entendimento, tanto para navegação como compreensão do texto. Na figura 2 a seguir tem-se a tela inicial do Tutorial sobre o nível físico de uma Rede de Computadores.

Figura 2 : Pagina inicial do Tutorial sobre o nível físico de uma rede de computadores

Este tutorial aborda os conceitos de rede de computadores organizando as informações de forma que, acessada pelo índice ou *hiperlink*, possa ser seguida uma seqüência nas consultas, a linguagem apresentada é de fácil entendimento ao usuário. Não há interação do usuário com a máquina, nem tratamento de erros pois este tutorial é só para consultas. Através do índice é possível avançar para o tópico à que ele está relacionado, não há marcadores de tempo para esta resposta.

3.5.3 TUTORIAL BÁSICO DE SQL

Este tutorial encontra-se na Internet no seguinte endereço eletrônico: http://www.geocities.com/SiliconValley/Foothills/7052/.

Este tutorial consiste basicamente em mostrar de maneira clara todas os comandos do *Structure Query Language* (SQL). Trazendo seu conceito e alguns exemplos de como usá-los. Na figura 3 temos a janela inicial deste tutorial. Podemos observar que há palavra com

link, estes ligam à descrição do comando e mostra com um exemplo como usá-lo, é um tutorial apenas para consulta, não havendo interação com o usuário, nem tempo de resposta a consulta solicitada.

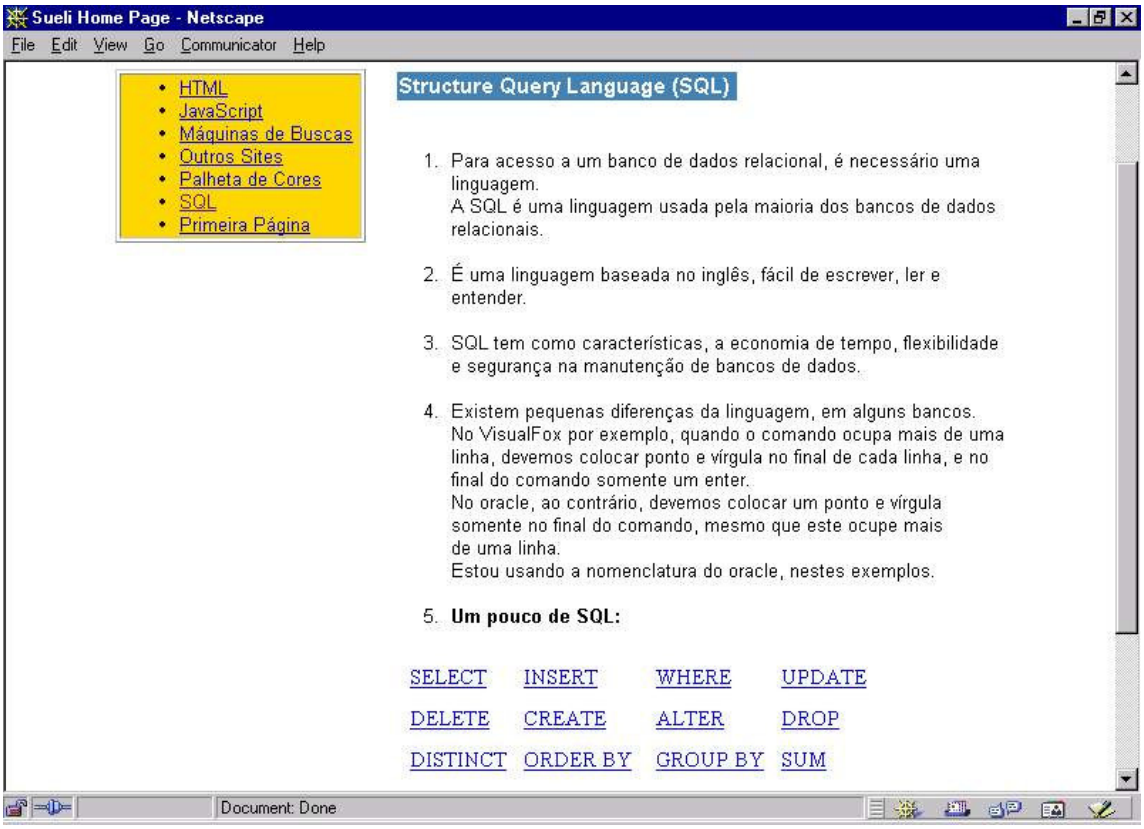

Figura 3 : Página inicial do Tutorial de SQL

3.6 TABELA COMPARATIVA ENTRE OS SOFTWARES

Tabela comparativa entre os tutoriais apresentados anteriormente é mostrada a seguir, contendo características dos mesmos.

	Tutorial de *Assembly*	Tutorial de Redes de Computadores	Tutorial Básico de SQL
Seqüência das operações	sim	sim	sim
Linguagem de interação	sim	sim	sim
Dispositivos de entrada	não	não	não
Dispositivos de apresentação	sim	sim	sim
Tempo de resposta	não	não	não
Tratamento de erros	não	não	não
Condução	não	não	Não

Tabela 1 : Comparação entre os softwares

3.7 SOFTWARE DE AUTORIA

O software de autoria permite ao usuário construir aplicativos completos de multimídia, desde uma simples apresentação, até uma apresentação completa.

Segundo [ANG95] e [SES95], "softwares de autoria são aqueles que já trazem definidas as etapas para criação de um software multimídia. Os softwares de autoria têm a vantagem de serem voltado especificamente para aplicação multimídia e, portanto, mais fáceis de usar. Além disso, agregam um requisito fundamental para a multimídia, que é a possibilidade de criar vínculos entre as informações e estabelecer um bom nível de interatividade do usuário com o programa. Os softwares de autoria funcionam como programa de editoração eletrônica, ou seja, necessitam de vários itens para obter uma composição".

Abaixo estão citados alguns princípios básicos para determinação de um software de autoria, segundo [SES95]:

a) habilidade para integrar texto a todos os elementos de multimídia, utilizando o som, vídeo e animação nos diversos formatos existentes;
b) habilidade para criar programas executáveis, possibilitando a criação de versões do sistema para serem rodados independentemente do software de autoria utilizado;

c) habilidade para criar aplicações como *quiosques* de vendas ou informações CD-ROMs, treinamento. Alguns softwares são bons para apresentações simples, demonstrativos ou protótipos, pecando pela pouca interatividade, que é uma das mais marcantes características deste tipo de software. Os softwares de autoria permitem um alto nível de interatividade, além de normalmente integrarem um banco de dados para satisfazer as necessidades mais sofisticadas.

3.7.1 FERRAMENTAS BASEADAS EM LINHA DE TEMPO

Nestas ferramentas de autoria, que são as mais utilizadas, os elementos de multimídia são dispostos pelo programador ao longo de uma linha de tempo de acordo com a seqüência em que eles serão exibidos, essa linha de tempo tem resoluções com variações entre ½ e 1 minuto em média, para todas as aplicações (este tempo é definido conforme a aplicação e a necessidade). Essa ferramenta é útil para aplicações que precisam sincronizar a exibição de muitos arquivos de som e imagem, por exemplo [BER96].

Este tipo de ferramenta permite que o desenvolvedor trabalhe com os objetivos em uma linha de tempo, ou seja, alinha-se os quadros, figuras e sons para serem executados em um determinado instante e por certo período. Dentro desta filosofia destacam-se as seguintes ferramentas: Tempra Gold, Animation Works Interactive e o Action [BER96].

3.7.2 FERRAMENTAS BASEADAS EM PILHAS DE CARTÕES

Este tipo de ferramenta segue a filosofia de que se tem uma pilha de cartões e que cada cartão indique qual é o próximo cartão a ser apresentado. Os software baseados em páginas ou cartões, distribuem os diversos objetos de aplicação – botões, textos e imagens. - por página, onde são relacionados entre si pelas associações entre os objetos.

Estas ferramentas são muito mais utilizadas quando a carga do seu conteúdo consiste em elementos que podem ser visualizados individualmente, como as páginas de um livro ou fichas em um arquivo. O sistema permite vincular estas páginas ou fichas em seqüências organizadas. O usuário pode pular, através de comandos, para qualquer página que quiser. Os sistemas baseados em páginas ou fichas permitem que o usuário execute elementos de som animação e vídeo digital [BER96].

Os principais representantes desta categoria são os softwares HyperCard, SuperCard, ToolBoolk e Hyper Studio [DET96].

3.7.3 FERRAMENTAS BASEADAS EM ÍCONES

Segundo [SES95], nos softwares de autoria baseados em ícones, o usuário constrói sua aplicação usando um fluxograma de eventos, tarefas e decisões associados a ícones que são fornecidos pelo fluxograma. Desenvolve-se a aplicação através de ícones que indicam qual é a ação a ser executada e qual é o próximo ícone a ser executado.

As ferramentas baseadas em ícones são as que dirigem eventos e simplificam a organização do projeto e, geralmente, apresentam diagramas de fluxo de atividades ao longo dos caminhos de ramificação. Esses ícones podem representar opções de menus, sons e imagens por exemplo. Os programas Authoware (Macromedia), IconAuthor (Aim Tech), Tempra Media Author (Mathematica) e HSC Interactive (HSC Software) pertencem a esta categoria.

4 ARQUITETURA DO MICROPROCESSADOR 8086/88

O 8086/88 são os microprocessador de 16 bits de utilização geral da Intel. O 8086 é um processador de 16 bits completo com respeito tanto à sua estrutura interna como a suas ligações externas, ao passo que o 8088 é um processador de 8 bits que internamente é igual ao 8086 de 16 bits [MOR88]. A figura 4 mostra a arquitetura interna do 8086 e do 8088, onde a única diferença está na fila seqüencial de bytes de instrução. O 8086 utiliza 6 bytes e o 8088 utiliza 4 bytes.

4.1 A CPU

Executa os processamentos numéricos (adição, subtração etc.), as operações lógicas e as funções de temporização. As operações da CPU são controladas por um conjunto de instruções que, quando organizadas em uma seqüência lógica, formam o chamado 'programa', esses por sua vez são dados a serem manipulados na memória [DIA90].

A CPU, segundo [DIA90], é alimentada com dados e sinais de controle, executa uma instrução por vez e produz como saída dados e sinais de controle. Internamente a CPU é formada por três unidades fundamentais:

a) registro ou registradores, veja figura 5, armazenam informações temporárias tais como: endereços de memória, códigos de estado e outras informações úteis durante a execução do programa;
b) unidade lógica e aritmética (ULA), contém um somador que executa operações aritméticas binárias sobre os dados obtidos da memória, dos registros ou de outras entradas;
c) circuito de controle, coordena todas as atividades do microprocessador. Através de pulsos externos de temporização chamados *clocks*, o circuito de controle gera as seqüências apropriadas de eventos necessários à execução das tarefas de processamento.

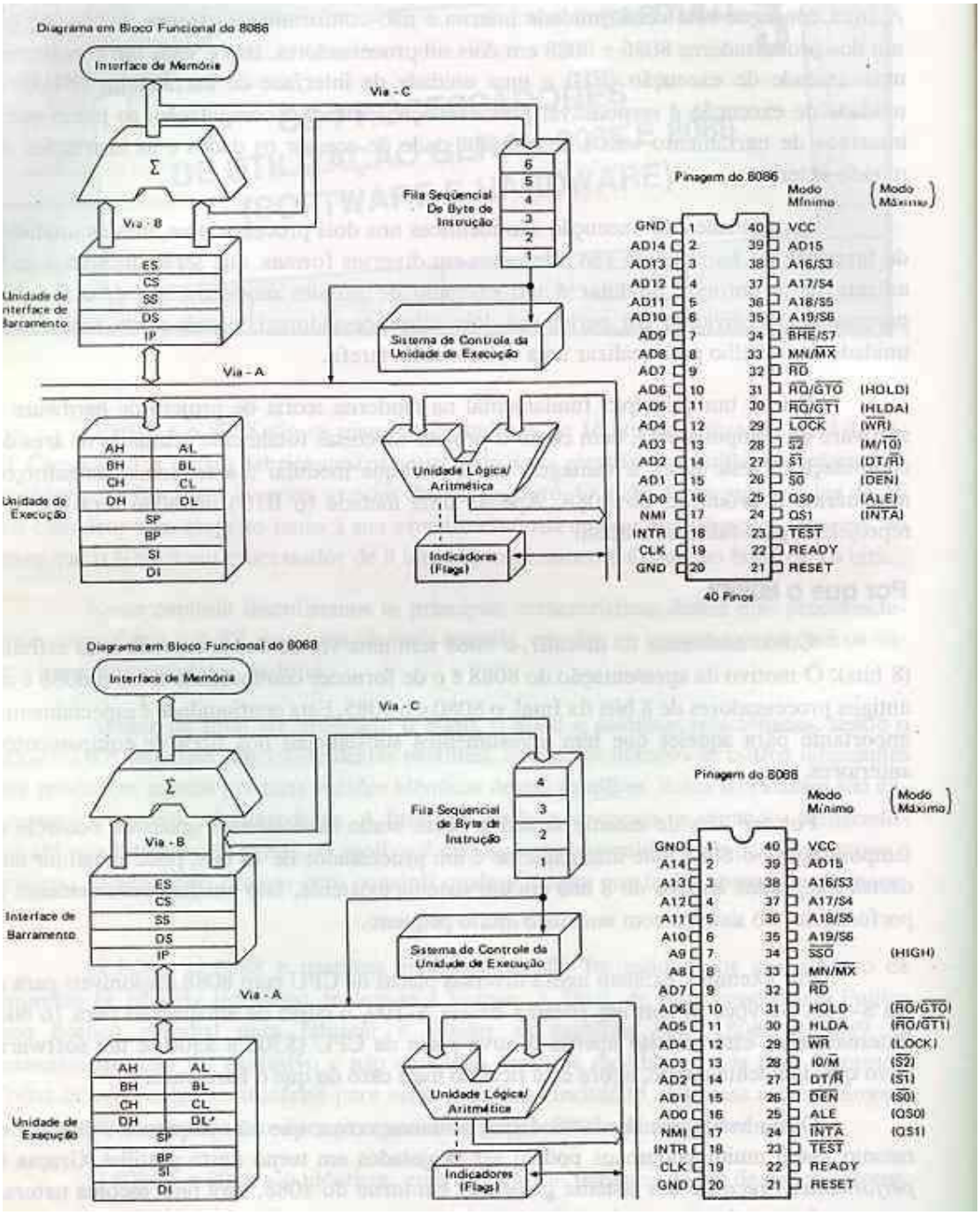

Figura 4 : Arquiteturas dos processadores 8086 e 8088

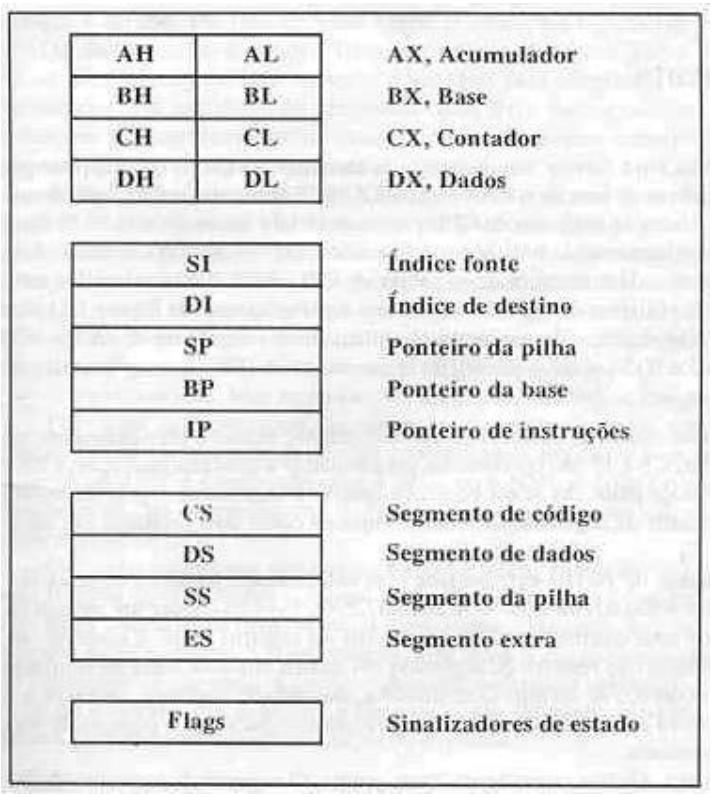

Figura 5 : Os registradores

4.2 MEMÓRIA

Sua principal tarefa é armazenar dados e programas, normalmente de forma temporária. A memória pode ser imaginada como uma coleção de células individuais, cada uma recebendo a atribuição de um número denominado endereço [MOR88].

Os microprocessadores usam dispositivos semicondutores para armazenar programas e dados. Os mais utilizados, segundo [DIA90], [MOR88], são:

a) RAM (*Randon Acess Memory*), memória de acesso aleatório, utilizada para escrita e leitura;
b) ROM (*Read Only Memory*), memória somente de leitura, usada somente para leitura.

Para expandir o espaço de armazenamento de dados, os sistemas microcomputadorizados usam algum dispositivo de armazenamento de massa, tais como:

a) discos flexíveis;
b) discos rígidos;

c) fitas magnéticas.

4.3 ENTRADA E SAÍDA (E/S)

Os dispositivos de entrada e saída, também conhecidos por periféricos, são os meios pelos quais a CPU se comunica com o mundo exterior.

Em um sistema típico, as portas ou canais de entrada são conectados ao teclado, e as portas ou canais de saída são interligados ao vídeo, unidade de disco, impressora, etc.

4.4 MODOS DE ENDEREÇAMENTO

Estes microprocessadores apresentam os seguintes modos de endereçamento, segundo [MON92] e [DIA90]:

a) imediato;
b) direto;
c) indireto;
d) por registrador;
e) indexado;
f) base mais deslocamento.

4.4.1 MODO IMEDIATO

É o modo mais simples e rápido de obter um dado, indicando seu próprio valor no campo operando da instrução, em ver de buscá-lo na memória. A vantagem deste método reside no curto tempo de execução da instrução, pois não gasta ciclo de memória para sua execução, exceto o único requerido para a busca da instrução. A desvantagem é que reside na limitação do tamanho do campo operando das instruções, o que reduz o valor máximo do dado a ser manipulado. Outra desvantagem é o fato de que, em programas repetidamente executados, com valores de variáveis diferentes a cada execução, esse método acarretaria o trabalho de alteração do valor do campo operando a cada execução.

Sintaxe:

MOV AX, 2

4.4.2 MODO DIRETO

É o método mais antigo e um dos mais utilizados em arquitetura de CPU. O valor binário contido no campo operando da instrução indica o endereço de memória onde se localiza o dado. Como o modo imediato, também é um modo simples de acesso, pois requer apenas uma referência à memória principal para buscar o dado (é mais lento que o modo imediato). Uma desvantagem desse processo está na limitação de memória a ser usada, conforme o tamanho do campo operando.

Sintaxe:

ADD AL, BETA

4.4.3 MODO INDIRETO

O valor binário do campo operando representa o endereço de uma célula; mas o conteúdo da referida célula não é o valor de um dado (como no modo direto); é um outro endereço de memória, cujo conteúdo é o valor do dado, ou seja, há um duplo endereçamento para o acesso ao dado.

O endereço intermediário (conteúdo da célula endereçada pelo valor do campo operando) é conhecido como *pointer*, pois indica a localização do dado. Com esse processo, elimina-se o problema do modo direto, de limitação do valor do endereço do dado, pois estando o endereço armazenado na memória (pode ocupar uma ou mais células)

Sintaxe:

MOV AX, [SI]

4.4.4 ENDEREÇAMENTO POR REGISTRO

Esse método tem características semelhantes aos modos direto e indireto, exceto que a célula (ou palavra) de memória referida na instrução é substituída por um dos registradores da CPU. Uma das vantagens é que consiste no menor número de bits necessários para endereçar os registradores. Outra vantagem está no próprio emprego do dado, que passa a ser armazenado em um meio (registrador) cujo acesso é muito mais rápido que o acesso à memória.

Sintaxe:

MOV AX,BX

4.4.5 MODO INDEXADO

Neste tipo de instrução, o endereço do dado é a soma do valor do campo operando e de um valor armazenado em um dos registradores da CPU (normalmente denominado registrador índice).

Sintaxe:

MOV AL, VETOR[SI]

4.4.6 MODO BASE MAIS DESLOCAMENTO

É um modo de endereçamento que tem como propósito reduzir o tamanho das instruções, bem como facilitar o processo de relocação dinâmica de programas. A escolha deste modo decorre de dois fatores:

a) durante a execução de uma grande quantidade de programas, as referências a células de memória, onde se localizam os operandos, são normalmente seqüenciais, ocorrendo poucos acessos a outras instruções fora de ordem (exceto desvios);
b) a maioria dos programas ocupa um pequeno espaço da memória principal disponível.

Este modo consiste na utilização de dois campos na instrução (que substituem o campo operando): um com o endereço de um registrador (chamado de base) e outro, com um valor denominado deslocamento (porque contém um valor relativo à primeira instrução).

Sintaxe:

MOV AH, [BX].BETA[SI]

4.5 LINGUAGEM DE PROGRAMAÇÃO ASSEMBLY

Assembly é uma linguagem de baixo nível, que pode ser utilizada para acelerar tarefas lentas. Consiste basicamente de sentenças que representam mnemônicos, por isso é mais rápida [BAS96].

Para a construção de um programa, segundo [AMA95] e [BAS96], são necessárias algumas ferramentas:

a) um editor para criar o programa fonte com extensão (.asm);
b) um montador, uma ferramenta que irá transformar o programa fonte num programa objeto (.obj);
c) um ligador que irá gerar o programa executável (.com ou .exe) a partir do programa objeto.

Para escrever um programa em *Assembly* segue a seqüência mostrada na figura 6.

```
;use " ; " para fazer comentários em programas Assembly
DOSSEG          ;Diz a CPU como organizar o segmento
.MODEL          ;Define o modelo de memória a usar no programa
.STACK          ;Reserva espaço de memória para as instruções de programa que utilizam a pilha
.DATA           ;O que vai no segmento de dados
.CODE           ;Define as instruções do programa, relacionado ao segmento de código
START           ;Início do código
END             ;Finaliza um programa Assembly
```

Figura 6 : Seqüência de um programa *Assembly*

Na área do "START" são relacionadas as instruções que darão vida ao programa. As instruções mais conhecidas estão na tabela 4.2, com suas respectivas funções, consulte o anexo para conhecer mais instruções.

Instrução	Função
MOV	Move
INT	Interrupção
ADD	Adição
SUB	Subtração
DEC	Decremento
INC	Incremento
JMP	Pula
CALL	Chamada de procedimento
LOOP	Repetição
LODS	Carrega
STOS	Armazena
REP	Repete

Tabela 2 : Algumas instruções dos microprocessadores 8086/88

Sub-rotina ou procedimento é um conjunto de instruções para as quais é possível direcionar o curso do programa, e uma vez que a execução destas instruções do procedimento tenha acabado, o controle retorna para linha que segue a que chamou o procedimento. A instrução *CALL* é utilizada para chamar a sub-rotina e *RET* é a instrução para retornar a rotina que chamou a sub-rotina.

Procedimentos nos ajudam a criar programas legíveis e fáceis de modificar. Quando se invoca um procedimento, o endereço da próxima instrução do programa é mantido na pilha, de onde é recuperado quando do retorno do procedimento [AMA95].

Há dois tipos de procedimentos, os intrasegmento, que se localizam no mesmo segmento da instrução que o chama e o valor de IP é armazenado na pilha, e os inter segmento, que podem se localizar em segmento de memória diferente daquele onde se encontra a instrução que o chama. Neste caso o valor de CS:IP é armazenado. Lembre que o registrador CS indica qual o segmento de código.

A instrução que chama um procedimento é como segue:

CALL NomedoProcedimento

As partes que compõem um procedimento, conforme o exemplo que soma dois bytes e armazena o resultado da soma em BX, são as seguintes:

```
Soma Proc Near    ; Declaração do Procedimento
MOV BX, 0         ; Conteúdo do Procedimento...
MOV BL, AH
MOV AH, 00
ADD BX, AX
RET               ; instrução de retorno
Soma endP         ; Fim do Procedimento
```

Na declaração, a primeira palavra, Soma, corresponde ao nome do procedimento. *Proc* declara-o, e a palavra *Near* indica que o procedimento é do tipo intrasegmento, ou seja, no mesmo segmento. A instrução *Ret* carrega IP com o endereço armazenado na pilha para retornar ao ponto do programa que chamou. Finalmente, Soma *endP* indica o fim do procedimento.

Para declarar um procedimento intersegmento, basta substituir a palavra *Near* para *Far*. A chamada deste procedimento e feito de modo idêntico: *CALL* Soma.

Esses são alguns conceitos básicos sobre arquitetura de computadores e linguagem de programação *Assembly*, a seguir tem-se algumas características do VXt.

4.6 CONHECENDO O VXT

O Virtual XT (VXt), figura 7, é um simulador do processador Intel 8086, o qual constitui-se num protótipo de uma ferramenta de apoio ao ensino de disciplinas de Arquitetura de Computadores, que apresenta conceitos básicos sobre os componentes de hardware de um computador e Sistemas Operacionais, onde são vistos conceitos de como o sistema operacional gerência os recursos de hardware. O VXt está sendo desenvolvido no Departamento de Sistemas da Computação da FURB - Universidade Regional de Blumenau [MAT97].

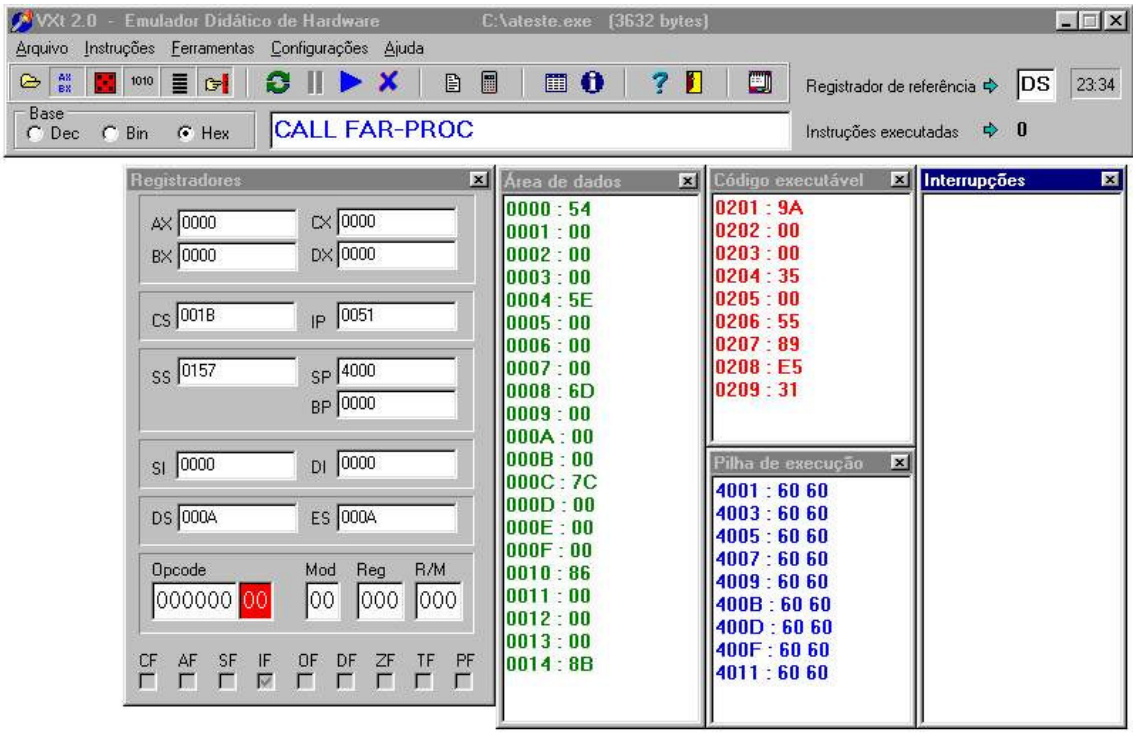

Figura 7 : VXt

Foi adotada a metodologia de aprendizado ativo, na qual os conceitos básicos seriam implementados gradualmente, e a medida em que o projeto avançasse, maiores níveis de abstração seriam obtidos, sem a perda do que já foi assimilado. Outro aspecto importante é que a cada semestre, novos alunos eram incorporados ao projeto, sendo que os mesmos tiveram acesso a toda documentação (especificação + código-fonte da aplicação) disponível de tal forma a revisar o que já havia sido construído e continuar o desenvolvimento do projeto [MAT99].

Na primeira versão foi possível a carga e execução de código binário nativo (gerado a partir de um programa auxiliar) em qualquer posição do espaço de endereçamento de 64Kbytes. A carga do programa era feita a mão visto que não foi implementado um *loader* de arquivos executáveis. Após o arquivo carregado, é possível a execução passo-a-passo de cada instrução, visualização do conteúdo dos registradores, da pilha de execução e da área de dados [MAT99].

4.6.1 CARACTERÍSTICAS TÉCNICAS DO VXT

A seguir são destacadas características técnicas do VXT, segundo [MAT99]:

a) permite a análise por parte do aluno, do *opcode* da instrução sendo executada;
b) apresenta informações sobre o passo de relocação de arquivos .EXE tanto a nível de header quanto ao nível de tabelas de relocação;
c) permite manipulação de arquivos binários puros, o que permite a construção de estruturas muito simples para teste de algum conceito/instrução sem a necessidade de elaboração de um programa completo;
d) permite a alteração da base numérica para representação das informações na tela, sendo possível optar-se por: decimal, hexadecimal e binário;
e) apresenta um registro (*log*) da seqüência de execução realizada, de tal forma que o aluno pode analisar o fluxo de execução de um determinado programa ou rotina, mesmo após o término da execução;
f) a medida que instruções do tipo *INT* são detectadas, as mesmas são filtradas e informações sobre as mesmas apresentadas em uma janela específica, sendo possível ao aluno analisar efetivamente o que ocorre quando um programa está em execução;
g) apresenta dicas explicativas (*hint*) sobre os conceitos teóricos que sustentam todo o funcionamento da CPU;
h) identifica quais são os registradores de referência quando o aluno clica na área de dados, na área de código e na área de pilha, induzindo-o ao entendimento da função dos registradores de segmento e deslocamento.

Nesta versão, o VXt implementa praticamente todas as instruções de um processador 8088/86, exceto aquelas utilizadas para acesso ao espaço de endereçamento de entrada/saída, tais como: *IN, OUT*. Entretanto, visto que as demais instruções foram implementadas, torna-se possível a execução de uma variedade de programas.

5 O PROTÓTIPO

A seguir será relatado o projeto e implementação de um protótipo de tutorial para o aprendizado da Linguagem de programação *Assembly*. Este sistema foi integrado ao simulador de hardware VXt versão 2.0, visando facilitar ao usuário o entendimento da execução das instruções no microprocessador 8086/88.

5.1 ESPECIFICAÇÃO DO PROTÓTIPO

Na figura 8, é apresentado em diagrama o contexto do protótipo do sistema tutorial.

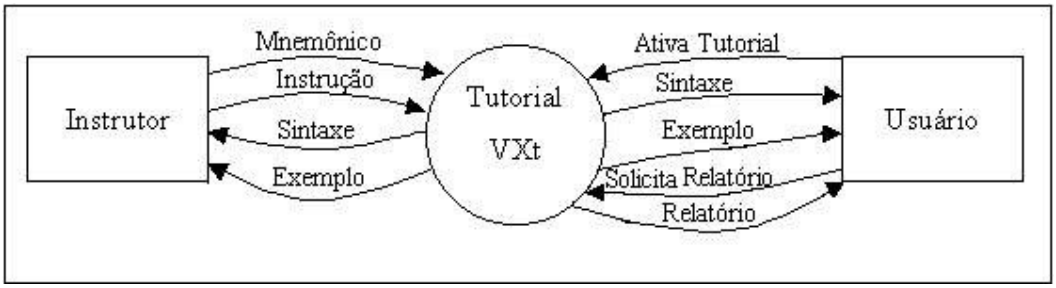

Figura 8 : Diagrama de fluxo de dados

Explodindo o diagrama de contexto, apresentado na figura 8, obtem-se o diagrama de fluxo de dados da figura 9.

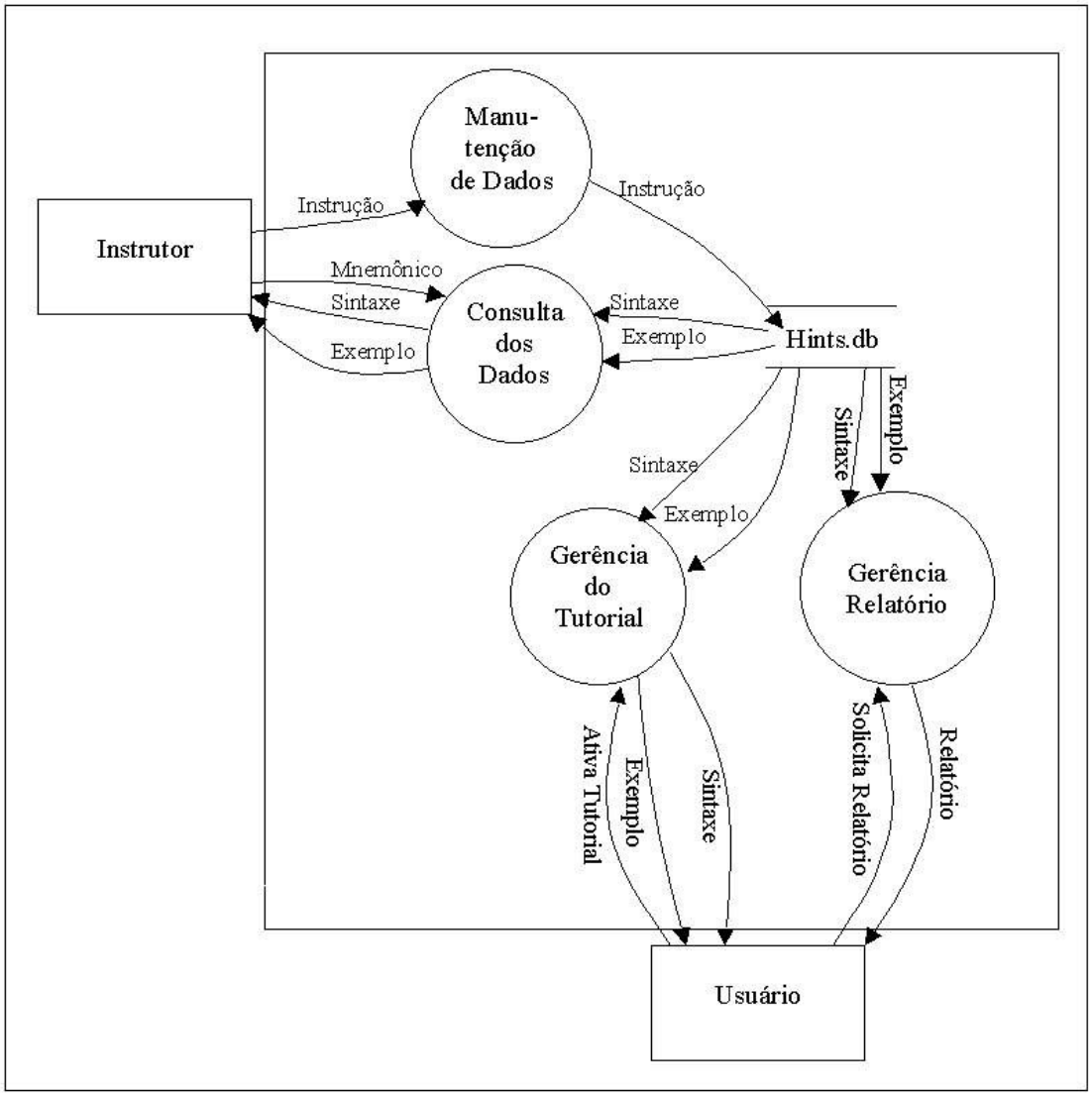

Figura 9 : Processo de manutenção de dados

Onde, Instrutor, é o usuário responsável pela criação e manutenção da tabela *Hints.db,* contendo a instrução com suas respectivas sintaxe e exemplo

Para o atendimento dos objetivos fez-se necessário a inclusão de algumas sub-rotinas no programa original do VXt.

O processo de “manutenção de dados”, figura 9, é responsável por criar a estrutura da tabela *Hints.db*. Esta tabela contém:

a) instrução: campo que contém o nome da instrução;

b) sintaxe: campo que contém a descrição da instrução;

c) exemplo: contém um exemplo relativo a instrução;

d) tópico: número do tópico do *help* para ser usado no campo *help context*.

O procedimento para capturar a instrução que está sendo executada pelo VXt e ativar o *Hint* contendo a sintaxe e o exemplo, é apresentado na figura 10.

```
Procedimento que ativa a Hint;
Variáveis locais;
Inicio
    Guarda a instrução mostrada no VXt;
    Guarda caracter até encontrar uma espaço ou barra;
    Se for barra então
       Copia para variável até a barra
    Senão
       se for  espaço então
          Copia para variável até o espaço
       Senão
          Copia 5 primeiros caracteres;
   Fim;
   Pesquisa o conteúdo da variável na tabela Hints.db;
   Mostra conteúdo pesquisado na tabela;
Fim;
```

Figura 10 : Procedimento para capturar a instrução do VXt

5.2 DESENVOLVIMENTO DO PROTÓTIPO

Optou-se por desenvolver o tutorial no ambiente de programação Delphi 3.0, devido o fato de o projeto VXt ter sido desenvolvido neste ambiente e para computadores que possuam sistema operacional *Windows 95*.

.Para criação da tabela *Hints.db* foi utilizado o Banco de Dados *Paradox*, por este ser parte integrante do ambiente de desenvolvimento Borland Delphi. Para fazer o gerenciamento

da tabela foi desenvolvido um programa, chamado de “Manutenção de Dados” – figura 11, que permite pesquisar uma instrução a fim de consultá-la, excluí-la ou mesmo alterá-la, caso a instrução desejada não seja encontrada será mostrada a instrução mais próxima àquela fornecida como argumento, facilitando a pesquisa no conjunto de instruções.

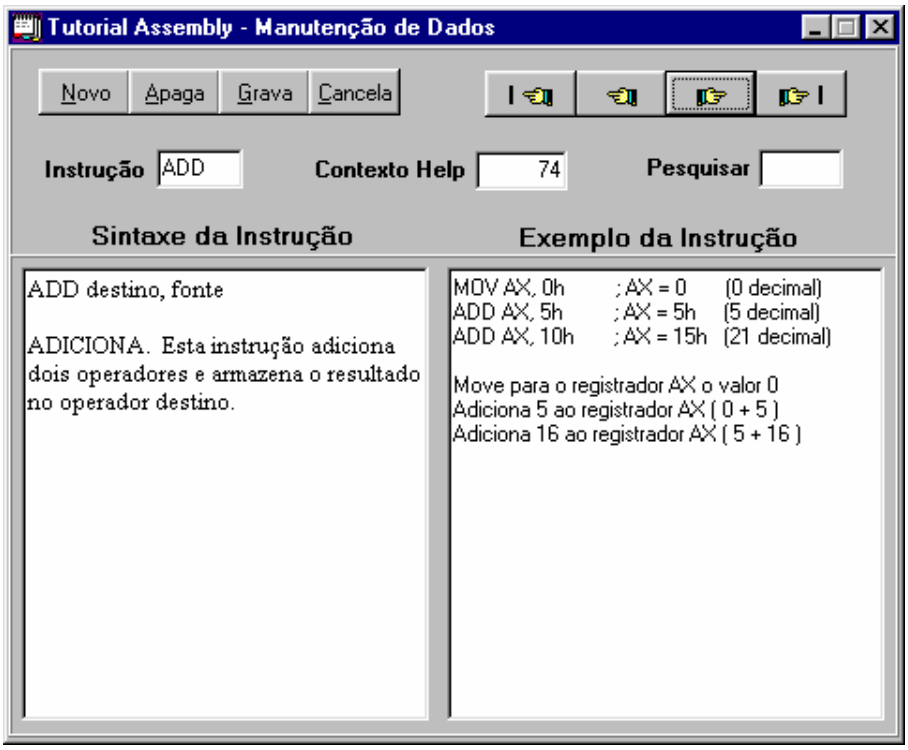

Figura 11 : Manutenção de dados

Para cadastrar uma nova instrução deve-se pressionar o botão ‘novo’, todas as modificações são confirmadas no botão ‘grava’ ou canceladas no botão ‘cancela’. O botão ‘apaga’ exclui a instrução da tabela.

Para construção do módulo principal do Tutorial, utilizou-se como base a propriedade *hint* do Delphi. Esta propriedade permite apresentar uma breve descrição a um determinado objeto. O mecanismo do Tutorial apresentará o texto quando se posicionar o mouse por sobre o campo, circulado na figura 12, que contém a instrução *Assembly*, abrindo a janela “sintaxe da instrução” do Tutorial, figura 13, o conteúdo desta janela e obtido na tabela *Hints*.d*b*, a partir do nome da instrução capturada do VXt, conforme apresentada anteriormente na figura 10.

Figura 12 : VXt

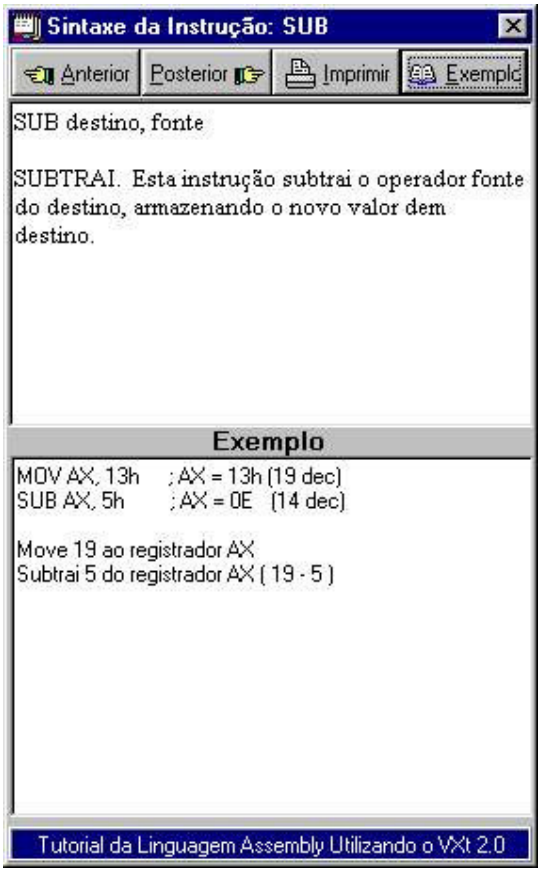

Figura 13 : Janela do tutorial

O VXt faz a carga de um programa executável (.exe ou .com), indicado pelo usuário. Para carregar o programa é solicitado a entrada do valor inicial dos registradores CS e IP, endereço de início do programa. Fornecendo esses valores o programa é colocado na memória e o simulador aguarda um comando do usuário para inicializar a execução do programa, este comando pode ser o botão "*Play*" ou o botão "Seqüencial". Após o Tutorial ativado, será visualizado a sintaxe da instrução, temos a opção do botão "Exemplo" da janela da sintaxe, para ver um exemplo relativo à instrução atual. Há também a opção para visualizar a impressão da instrução atualmente apresentada.

Utilizando os botões ‘anterior’ e ‘posterior’, como o próprio nome diz, será visualizada a instrução anterior e posterior, respectivamente, a partir da instrução atual da janela da sintaxe, através desses botões pode-se navegar por todas as instruções já executadas pelo VXt. Para ver uma instrução em específico, basta pressionar com o botão direito do mouse sobre o botão ‘anterior’ ou ‘posterior’ e abrir a lista de instruções já executadas, escolher uma pressionando com o mouse sobre ela e a mesma será visualizada na janela da sintaxe.

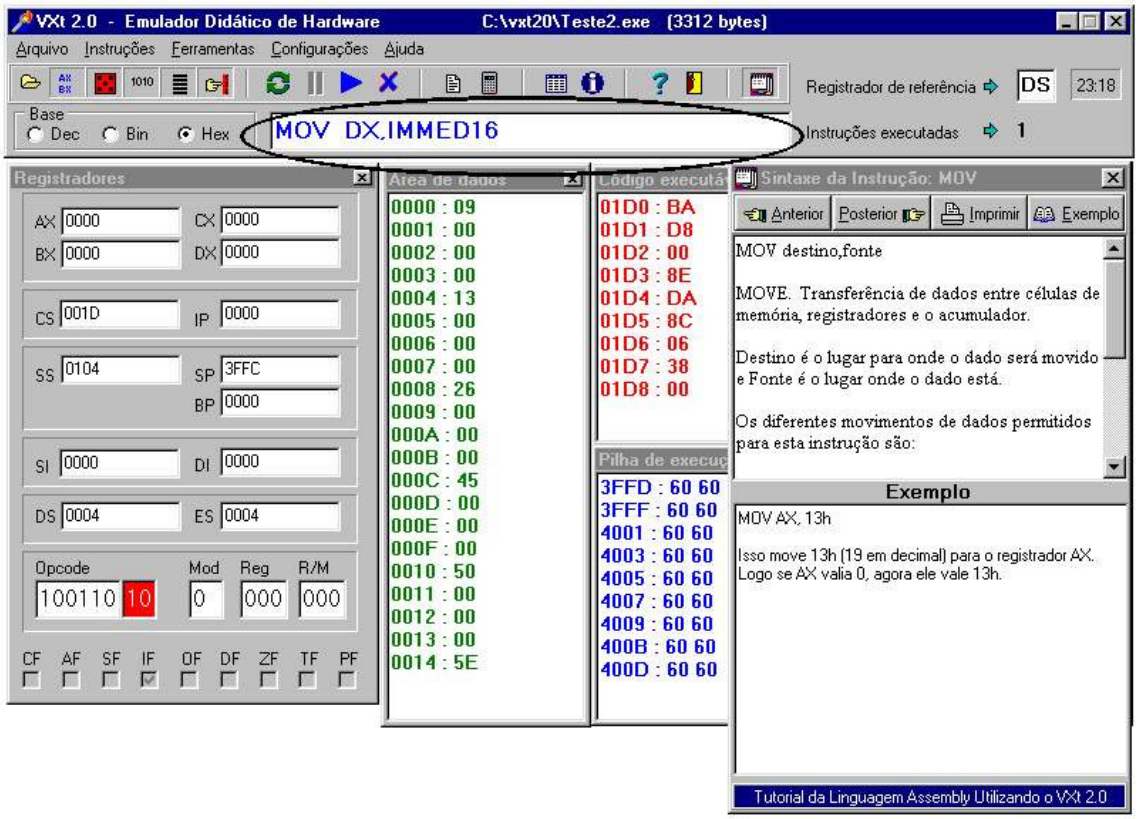

Figura 14 : VXt com o tutorial ativo

Caso não se queira fazer o acompanhamento da execução das instruções pelo Tutorial basta desativá-lo clicando no último botão da barra de ferramentas do VXt 2.0, figura 12.

Figura 15 : Ícone para ativar/desativar o Tutorial

5.3 RECOMENDAÇÕES ERGONÔMICAS ADOTADAS PELO PROTÓTIPO

Abaixo segue recomendações ergonômicas, apresentadas no capítulo 3.2, adotas pelo protótipo:

a) as seqüência das operações são determinadas pelo usuário, de modo que, existem opções de escolha para tanto. As informações estão dispostas de forma fácil de serem visualizadas através dos botões 'Anterior' e 'Posterior';
b) a linguagem de interação foi determinada baseada no conteúdo de livros existentes sobre o assunto, o vocabulário é bastante simples e facilita a memorização do usuário;
c) os dispositivos de entrada são o teclado e o mouse, não existindo qualquer outra forma de interação do usuário com o software;
d) os dispositivos de apresentação, a seqüência de execução do software é de fácil memorização, visto que, o menu está disposto de forma homogênea e auto explicativa;
e) para o tempo de resposta ser imediato, é aconselhado usar computador Pentium 166 com 16 Mega Bytes ou superior;
f) o sistema apresenta mensagem de erro durante a fase de execução, quando aparece uma instrução não cadastrada;
g) quanto a condução durante a execução do programa, o software se comporta de forma a apresentar ao usuário as informações a respeito de *Assembly* não tendo a necessidade de mostrar ao usuário o que deve ser feito.

6 CONCLUSÃO

Neste capítulo, procura-se dar uma visão geral da implementação da ferramenta restrições e sugestões de aperfeiçoamento da mesma para possíveis futuros trabalhos.

6.1 CONSIDERAÇÕES FINAIS

O presente trabalho teve como objetivo principal o desenvolvimento de um protótipo de uma ferramenta para consulta de especificações técnicas da linguagem de programação *Assembly*. Com o desenvolvimento deste protótipo observou-se a grande contribuição que um guia *on-line* proporciona ao usuário nas suas consultas.

O protótipo desenvolvido apresenta, as seguintes vantagens:

a) fornecer um acesso rápido à especificação técnica da instrução mostrando também um exemplo relativo;
b) de acordo com a instrução visualizada, permite ao usuário imprimi-la;
c) no VXt há uma lista das instruções executadas, o protótipo permite deslocar-se nesta lista e mostrar a especificação do item relacionado.

Com a conclusão do trabalho observa-se que foi possível uma análise das principais recomendações ergonômicas, apresentadas no capítulo 3.2, para construção de software educacionais, comparando software já existentes.

Torna-se importante ressaltar que o protótipo desenvolvido, apesar de demonstrar a possibilidade de sua utilização como ferramenta de consulta a especificação técnica, possui algumas limitações, a saber:

a) nem todas as instruções do processador 8086 foram cadastradas;
b) não permite atualizar a base de dados durante a consulta.

6.2 SUGESTÕES

Tomando-se como base o presente trabalho, pode sugerir o que segue:

a) incluir ao protótipo desenvolvido a possibilidade de manutenção da base de dados;
b) expandir o tutorial a ponto que abranja as demais telas do VXt, tais como:
   - registradores;

- área de dados;
- pilha de execução;
- código executável.

c) adicionar ao exemplo, ilustrações de como são manipulados as instruções.

ANEXO

Instruções para o 8086

Aritmética

ADD	ADD destino, fonte soma		Flag	O D I T S Z A P C X X X X X X	
Operandos		Clock	Transferencia	Bytes	Exemplo
registro, registro		3	#	2	ADD CX, DX
registro, memória		9+EA	1	2 4	ADD DL,[BX].ALPHA
memória, registro		16+EA	2	2 4	ADD TEMP, CL
registro, imediato		4	#	3 4	ADD CL, 2
memória, imediato		17+EA	2	3 6	ADD ALPHA, 2
acumulador, imediato		4	#	2 3	ADD AX, 200

SUB	SUB destino, fonte Subtração		Flag	O D I T S Z A P C X X X X X X	
Operandos		Clock	Transferencia	Bytes	Exemplo
registro, registro		3	#	2	SUB CX, BX
registro, memória		9+EA	1	2 4	SUB DX, MATH_TOTAL[SI]
memória, registro		16+EA	2	2 4	SUB [BP+2], CL
acumulador, imediato		4	#	2 3	SUB AL, 10
registro, imediato		4	#	3 4	SUB SI, 5280
memória, imediato		17+EA	2	3 6	SUB [BP] BALANCE, 1000

SBB	SBB destino, fonte Subtração		Flag	O D I T S Z A P C X X X X X X	
Operandos		Clock	Transferencia	Bytes	Exemplo
registro, registro		3	#	2	SBB BX, CX
registro, memória		9+EA	1	2 4	SBB DI, [BX]PAYMENT
memória, registro		16+EA	2	2 4	SBB BALANCE, AX
acumulador, imediato		4	#	2 3	SBB AX, 2
registro, imediato		4	#	3 4	SBB CL, 1
memória, imediato		17+EA	2	3 6	SBB COUNT [SI], 10

INC	INC destino Incremento em 1		Flag	O D I T S Z A P C X X X X X	
Operandos		Clock	Transferencia	Bytes	Exemplo
reg 16		2	#	1	INC CX
reg 8		3	#	2	INC BL
memória		15+EA	1	2 4	INC ALPHA[DI][BX]

DEC	DEC destino Decremento em 1		Flag	O D I T S Z A P C X X X X X	
Operandos		Clock	Transferencia	Bytes	Exemplo
reg 16		2	#	1	DEC AX
reg 8		3	#	2	DEC AL
Memória		15+EA	2	2 4	DEC ARRAY[SI]

NEG	NEG destino Nega		Flag	O D I T S Z A P C X X X X X 1	
Operandos		Clock	Transferencia	Bytes	Exemplo
Registro		3	#	2	NEG AL
Memória		16+EA	2	2 4	NEG MULTIPLIER

IMUL	IMUL fonte Multiplicação inteiro		Flag	O D I T S Z A P C X U U U U U	
Operandos		Clock	Transferencia	Bytes	Exemplo
reg 8		80-98	#	2	IMUL CL
reg 16		128-154	#	2	IMUL BX
mem 8		(86-104)+EA	1	2 4	IMUL RATE BYTE
mem 16		(134-160)+EA	1	2 4	IMUL RATE WORD[BP][DI]

MUL	MUL fonte Multiplicação, ñ sinal		Flag	O D I T S Z A P C X U U U U U	
Operandos		Clock	Transferencia	Bytes	Exemplo
reg8		70-77	#	2	MUL BL
reg16		118-133	#	2	MUL CX
mem8		(76-83)+EA	1	2 4	MUL MONTH[SI]
mem16		(124-139)+EA	1	2 4	MUL BAUD_RATE

DIV	DIV fonte Divisão, ñ atrib		Flag	O D I T S Z A P C U U U U U U	
Operandos		Clock	Transferencia	Bytes	Exemplo
reg 8		80-90	#	2	DIV CL
reg 16		144-162	#	2	DIV BX
mem 8		(86-96)+EA	1	2 4	DIV ALPHA
mem 16		(150-168)+EA	1	2 4	DIV TABLE[SI]

IDIV	IDIV fonte Divisão inteiro		Flag	O D I T S Z A P C U U U U U U	
Operandos		Clock	Transferencia	Bytes	Exemplo
reg 8		102-112	#	2	IDIV BL
reg 16		165-184	#	2	IDIV CX
mem 8		(107-118)+EA	1	2 4	IDIV DIVISOR_BYTE[SI]
mem 16		(171-190)+EA	1	2 4	IDIV [BX]DIVISOR_WORD

CMP	CMP destino fonte Compara destino com fonte		Flag	O D I T S Z A P C X X X X X	
Operandos		Clock	Transferencia	Bytes	Exemplo
registro, registro		3	#	2	CMP BX,CX
registro, memória		9+EA	1	2 4	CMP DH, ALPHA
memória, registro		9+EA	1	2 4	CMP [BP+2],SI
registro, imediato		4	#	3 4	CMP BL, 02H
memória, imediato		10+EA	1	3 6	CMP [BX]RADAR[DI],3420H
acumulador, imediato		4	#	2 3	CMP AL, 00010000B

CMPS	CMPS strings fonte e destino Compara strings		Flag	O D I T S Z A P C X X X X X X	
Operandos		Clock	Transferencia	Bytes	Exemplo
strings fonte e destino		22	2	1	CMPS BUFF1, BUFF2
(repete)string font e dest		9+22/rep	2/rep	1	REPE CMPS ID KEY

CBW	CBW(não há operandos) Converte byte em palavra		Flag	O D I T S Z A P C	
Operandos		Clock	Transferencia	Bytes	Exemplo
(não há operandos		2	#	1	CBW

CWD	CWD(não há operandos) Converter palavra em pal dupla		Flag	O D I T S Z A P C	
Operandos		Clock	Transferencia	Bytes	Exemplo
(não há operandos)		5	#	1	CWD

Lógica

AND	CALL destino, fonte And lógico		Flag	O D I T S Z A P C 0 X X U X 0	
Operandos		Clock	Transferencia	Bytes	Exemplo
registro, registro		3	#	2	AND AL, BL
registro, memória		9+EA	1	2 4	AND CX, FLAG_WORD
memória, registro		16+EA	2	2 4	AND ASCII[DI], AL
registro, imediato		4	#	3 4	AND CX, 0F0H
memória, imediato		17+EA	2	3 6	AND BETA, 01H
acumulador, imediato		4	#	2 3	AND AX, 01010000B

OR	OR destino, fonte Or lógico inclusive		Flag	O D I T S Z A P C 0 X X U X 0	
Operandos		Clock	Transferencia	Bytes	Exemplo
registro, registro		3	#	2	OR AL, BL
registro, memória		9+EA	1	2 4	OR DX, PORT ID[DI]
memória, registro		16+EA	2	2 4	OR FLAG BYTE, CL
acumulador, imediato		4	#	2 3	OR AL, 01101100B
registro, imediato		4	#	3 4	OR CX, 01H
memória, imediato		17+EA	2	3 6	OR [BX] CMD WORD,0CFH

XOR	XOR destino, fonte Or exclusivo lógico		Flag	O D I T S Z A P C 0 X X U X 0	
Operandos		Clock	Transferencia	Bytes	Exemplo
registro, registro		3	#	2	XOR CX, BX
registro, memória		9+EA	1	2 4	XOR CL, MASK_BYTE
memória, registro		16+EA	2	2 4	XOR ALPHA[SI], DX
acumulador, imediato		4	#	2 3	XOR AL, 01000010B
registro, imediato		4	#	3 4	XOR SI, 00C2H
memória, imediato		17+EA	2	3 6	XOR RETURN_CODE, 0D2H

NOT	NOT destino Not lógico		Flag	O D I T S Z A P C
Operandos	Clock	Transferencia	Bytes	Exemplo
Registro	3	#	2	NOT AX
Memória	16+EA	2	2 4	NOT CHARACTER

SHR	SHR destino, cont Shift lógico direita		Flag	O D I T S Z A P C X X
Operandos	Clock	Transferencia	Bytes	Exemplo
registro, 1	2	#	2	SHR SI, 1
registro, CL	8+4/bit	#	2	SHR SI, CL
memória, 1	15+EA	2	2 4	SHR ID BYTE [SI][BX],1
memória, CL	20+EA+4/bit	2	2 4	SHR INPUT WORD, CL

SAHF	SAHF (não há operandos) Armazena AH nos flags		Flag	O D I T S Z A P C R R R R R
Operandos	Clock	Transferencia	Bytes	Exemplo
(não há operandos)	4	#	1	SAHF

SAL/SHL	SAL/SHL destino, cont Shift aritmético esq/Shift lógico esq		Flag	O D I T S Z A P C
Operandos	Clock	Transferencia	Bytes	Exemplo
registro, 1	2	#	2	SAR DX, 1
registro, CL	8+4/bit	#	2	SAR DI, CL
memória, 1	15+EA	2	2 4	SAR N BLOCKS, 1
memória, CL	20+EA+4/bit	2	2 4	SAR N BLOCKS, CL

SAR	SAR destino, fonte Shift arit direita		Flag	O D I T S Z A P C X X X U X X
Operandos	Clock	Transferencia	Bytes	Exemplo
registro, 1	2	#	2	SAR DX, 1
registro, CL	8+4/bit	#	2	SAR DI, CL
memória, 1	15+EA	2	2 4	SAR N BLOCKS, 1
memória, CL	20+EA+4/bit	2	2 4	SAR N BLOCKS, CL

ROR	ROR destino, fonte Para a direita		Flag	O D I T S Z A P C X X
Operandos	Clock	Transferencia	Bytes	Exemplo
registro, 1	2	#	2	ROR AL, 1
registro, CL	8+4/bit	#	2	ROR BX, CL
memória, 1	15+EA	2	2 4	ROR PORT STATUS, 1
memória, CL	20+EA+4/bit	2	2 4	ROR CMD WORD, CL

ROL	ROL destino, fonte Para a esquerda		Flag	O D I T S Z A P C X X
Operandos	Clock	Transferencia	Bytes	Exemplo
registro, 1	2	#	2	ROL BX, 1
registro, CL	8+4/bit	#	2	ROL DI, CL
memória, 1	15+EA	2	2 4	ROL FLAG BYTE[DI], 1
memória, CL	20+EA+4/bit	2	2 4	ROL ALPHA, CL

RCR	RCR atribuição, cont Para a direita por transporte	Flag	O D I T S Z A P C X X	
Operandos	Clock	Transferencia	Bytes	Exemplo
registro, 1	2	#	2	RCR BX, 1
registro, CL	8+4/bit	#	2	RCR BL, CL
memória, 1	15+EA	2	2 4	RCR [BX]STATUS,1
memória, CL	20+EA+4/bit	2	2 4	RCR ARRAY[DI],CL

RCL	RCL destino, fonte Para a esquerda por transporte	Flag	O D I T S Z A P C X X	
Operandos	Clock	Transferencia	Bytes	Exemplo
registro, 1	2	#	2	RCL CX, 1
registro, CL	8+4/bit	#	2	RCL AL, CL
memória, 1	15+EA	2	2 4	RCL ALPHA, 1
memória, CL	20+EA+4/bit	2	2 4	RCL [BP]PARM, CL

TEST	TEST destino, fonte Test ou and lógico indestr	Flag	O D I T S Z A P C 0 X X U X 0	
Operandos	Clock	Transferencia	Bytes	Exemplo
registro, registro	3	#	2	TEST SI, DI
registro, memória	9+EA	1	2 4	TEST SI, END_COUNT
acumulador, imediato	4	#	2 3	TEST AL, 00100000B
registro, imediato	5	#	3 4	TEST BX, 0CC4H
memória, imediato	11+EA	#	3 6	TEST RETURN CODE, 01H

Transferência de Dados

LEA	LEA destino, fonte Carrega end efetivo	Flag	O D I T S Z A P C	
Operandos	Clock	Transferencia	Bytes	Exemplo
reg16, mem16	2+EA	#	2 4	LEA BX, [BP][DI]

LES	LES destino, fonte Carrega ponteiro com ES	Flag	O D I T S Z A P C	
Operandos	Clock	Transferencia	Bytes	Exemplo
reg16, mem32	16+EA	2	2 4	LES DI, [BX].TEXT_BUFF

LODS	LODS string, fonte Carrega string	Flag	O D I T S Z A P C	
Operandos	Clock	Transferencia	Bytes	Exemplo
string fonte	12	1	1	LODS CUSTOMER_NAME
(repete) string fonte	9+13/rep	1/rep	1	REP LODS NAME

LDS	LDS destino, fonte Carrega ponteiro com DS	Flag	O D I T S Z A P C	
Operandos	Clock	Transferencia	Bytes	Exemplo
reg16, mem32	16+EA	2	2 4	LDS SI, DATA.SEG[DI]

STOS	STOS strint-dest Armazena byte ou pal		Flag	O D I T S Z A P C
Operandos	Clock	Transferencia	Bytes	Exemplo
string-dest	11	1	1	STOS PRINT LINE
(repete) string-dest	9+10/rep	1/rep	1	REP STOS DISPLAY

MOV	MOV destino, fonte Move		Flag	O D I T S Z A P C
Operandos	Clock	Transferencia	Bytes	Exemplo
memória, acumulador	10	1	3	MOV ARRAY[SI], AL
acumulador, memória	10	1	3	MOV AX, TEMP RESULT
registro, registro	2	#	2	MOV AX, CX
registro, memória	8+EA	1	2 4	MOV BP, STACK TOP
memória, registro	9+EA	1	2 4	MOV COUNT [DI], CX
registro, imediato	4	#	2 3	MOV CL, 2
memória, imediato	10+EA	1	3 6	MOV MASK[BX][SI], 2CH
seg-reg, reg 16	2	#	2	MOV ES, CX
seg-reg, mem 16	8+EA	1	2 4	MOV DS, SEGMENT BASE
reg 16, seg-reg	2	#	2	MOV BP, SS
memória, seg-reg	9+EA	1	2 4	MOV [BX].SEG SAVE, CS

LAHF	LAHF (não há operandos) Carrega AH dos flags		Flag	O D I T S Z A P C
Operandos	Clock	Transferencia	Bytes	Exemplo
(não há operandos)	4	#	1	LAHF

MOVS	MOVS string dest, string fonte Move string		Flag	O D I T S Z A P C
Operandos	Clock	Transferencia	Bytes	Exemplo
string dest, string fonte	18	2	1	MOVS LINE EDIT DATA
(repete) string dest, string fonte	9+17/rep	2/rep	1	REP MOVS SCREEN, BUF

MOVSB/MOVSW	MOVSB/MOVSW (não há operandos) Move string (byte/pal)		Flag	O D I T S Z A P C
Operandos	Clock	Transferencia	Bytes	Exemplo
(não há operandos)	18	2	1	MOVSB
(repete)(não há operandos)	9+17/rep	2/rep	1	REP MOVSB

XCHG			Flag	O D I T S Z A P C
Operandos	Clock	Transferencia	Bytes	Exemplo
acumulador, reg16	3	#	1	XCHG AX, BX
memória, registro	17+EA	2	2 4	XCHG SEMAPHORE, AX
registro, registro	4	#	2	XCHG AL, BL

IN	IN porta acumulador Insere byte ou palavra		Flag	O D I T S Z A P C
Operandos	Clock	Transferencia	Bytes	Exemplo
acumulador, imed8	10	1	2	IN AL, 0FFEAH
acumulador, DX	8	1	1	IN AX, DX

OUT	OUT porta, acumulador Saída - byte ou pal		Flag	O D I T S Z A P C	
Operandos		Clock	Transferencia	Bytes	Exemplo
imed8, acumulador		10	1	2	OUT 44,AX
DX, acumulador		8	1	1	OUT DX, AL

CLC	CLC(não há operandos) Zera flag de transporte		Flag	O D I T S Z A P C . 0	
Operandos		Clock	Transferencia	Bytes	Exemplo
(não há operandos)		2	#	1	CLC
CMC	CMC(não há operandos) Complemento do flag de transporte		Flag	O D I T S Z A P C . X	
Operandos		Clock	Transferencia	Bytes	Exemplo
(não há operandos)		2	#	1	CMC

STC	STC (não há operandos) Flag de transporte		Flag	O D I T S Z A P C . 1	
Operandos		Clock	Transferencia	Bytes	Exemplo
(não há operandos)		2	#	1	STC
CLD	CLD(não há operandos) Zera flag de direção		Flag	O D I T S Z A P C . 0	
Operandos		Clock	Transferencia	Bytes	Exemplo
(não há operandos)		2	#	1	CLD

STD	STD (não há operandos) Flag de direção		Flag	O D I T S Z A P C . 1	
Operandos		Clock	Transferencia	Bytes	Exemplo
(não há operandos)		2	#	1	STD

INT	INT interruptor-tipo Inrerruptor		Flag	O D I T S Z A P C . 0 0	
Operandos		Clock	Transferencia	Bytes	Exemplo
imed8 (tipo = 3)		52	5	1	INT 3
imed8 (tipo <>3)		51	5	2	INT 67

INTO	INTO(não há operandos) Interrompe se estouro		Flag	O D I T S Z A P C . 0 0	
Operandos		Clock	Transferencia	Bytes	Exemplo
(não há operandos)		53 ou 4	5	1	INTO

ESC	ESC cód op externo, fonte Escapae		Flag	O D I T S Z A P C	
Operandos		Clock	Transferencia	Bytes	Exemplo
imediato, memória		8+EA	1	2 4	ESC 6, ARRAY[SI]
imediato, registro		2	#	2	ESC 20, AL

LOCK	LOCK (não há operandos) Fecha o bus		Flag	O D I T S Z A P C	
Operandos		Clock	Transferencia	Bytes	Exemplo
(não há operandos)		2	#	1	LOCK XCHG FLAG, AL

Salto

JÁ/JNBE	JÁ/JNBE short-label Salto se acima salto se ñ baixo ou igual		Flag	O D I T S Z A P C	
Operandos		Clock	Transferencia	Bytes	Exemplo
short-label		16 ou 4	#	2	JÁ ABOVE

JAE/JNB	JAE/JNB short-label Salto se acima ou =/Salto se ñ abaixo		Flag	O D I T S Z A P C	
Operandos		Clock	Transferencia	Bytes	Exemplo
short-label		16 ou 4	#	2	JAE ABOVE EQUAL
JB/JNAE	JB/JNAE short-label Salto se abaixo/Salto se ñ acima nem =		Flag	O D I T S Z A P C	
Operandos		Clock	Transferencia	Bytes	Exemplo
short-label		16 ou 4	#	2	JB BELOW

JBE/JNA	JBE/JNA short-label Salto se abaixo ou =/Salto se ñ acima		Flag	O D I T S Z A P C	
Operandos		Clock	Transferencia	Bytes	Exemplo
short-label		16 ou 4	#	2	JNA NOT ABOVE

JC	JC short-label Salto se transporte		Flag	O D I T S Z A P C	
Operandos		Clock	Transferencia	Bytes	Exemplo
short-label		16 ou 4	#	2	JC CARRY SET

JCXZ	JCXZ short-label Salto se CX for zero		Flag	O D I T S Z A P C	
Operandos		Clock	Transferencia	Bytes	Exemplo
short-label		18 ou 6	#	2	JCXZ COUNT DONE

JE/JZ	JE/JZ short-label Salto se =/Salto se zero		Flag	O D I T S Z A P C	
Operandos		Clock	Transferencia	Bytes	Exemplo
short-label		16 ou 4	#	2	JZ ZERO

JG/JNLE	JG JNLE short-label Salto se maior/Salto se ñ menor ou =		Flag	O D I T S Z A P C	
Operandos		Clock	Transferencia	Bytes	Exemplo
short-label		16 ou 4	#	2	JZ ZERO

JGE/JNL	JGE/JNL short-label Salto se maior ou =/Salto se ñ menor		Flag	O D I T S Z A P C	
Operandos		Clock	Transferencia	Bytes	Exemplo
short-label		16 ou 4	#	2	JGE GREATER_EQUAL

JL/JNGE	JL/JNGE short-label Salto se menor/Salto se ñ maior ou =		Flag	O D I T S Z A P C	
Operandos		Clock	Transferencia	Bytes	Exemplo
short-label		16 ou 4	#	2	JL LESS

JLE/JNG	JLE/JNG short-label Salto se menor ou =/Salto se ñ maior		Flag	O D I T S Z A P C	
Operandos		Clock	Transferencia	Bytes	Exemplo
short-label		16 ou 4	#	2	JNG NOT_GREATER

JMP	JMP alvo Salto		Flag	O D I T S Z A P C	
Operandos		Clock	Transferencia	Bytes	Exemplo
short-label		15	#	2	JMP SHORT
near-label		15	#	3	JMP WITHIN_SEGMENT
far-label		15	#	5	JMP FAR_LABEL
memptr16		18+EA	1	2 4	JMP [BX].TARGET
regptr16		11	#	2	JMP CX
memptr32		24+EA	2	2 4	JMP OTHER.SEG[SI]

JNC	JNC short-label Salto se ñ transporte		Flag	O D I T S Z A P C	
Operandos		Clock	Transferencia	Bytes	Exemplo
short-label		16 ou 4	#	2	JNC NOT_CARRY

JNE/JNZ	JNE/JNZ short-label Salto se ñ =/Salto se ñ zero		Flag	O D I T S Z A P C	
Operandos		Clock	Transferencia	Bytes	Exemplo
short-label		16 ou 4	#	2	JNE NOT_EQUAL

JNO	JNO short-label Salto se ñ estouro		Flag	O D I T S Z A P C	
Operandos		Clock	Transferencia	Bytes	Exemplo
short-label		16 ou 4	#	2	JNO NO OVERFLOW

JNP/JPO	JNP/JPO short-label Salto se ñ paridade/Salto se limpar		Flag	O D I T S Z A P C	
Operandos		Clock	Transferencia	Bytes	Exemplo
short-label		16 ou 4	#	2	JPO ODD PARITY

JNS	JNS short-label Salto se ñ sinal		Flag	O D I T S Z A P C	
Operandos		Clock	Transferencia	Bytes	Exemplo
short-label		16 ou 4	#	2	JNS POSITIVE

JO	JO short-label Salto se estouro		Flag	O D I T S Z A P C	
Operandos		Clock	Transferencia	Bytes	Exemplo
short-label		16 ou 4	#	2	JO SIGNED_OVRFLW

JP/JPE	JP/JPE short-label Salto se paridade/Salto se par		Flag	O D I T S Z A P C	
Operandos		Clock	Transferencia	Bytes	Exemplo
short-label		16 ou 4	#	2	JPE EVEN_PARITY

JS	JS short-label Salto se sinal		Flag	O D I T S Z A P C	
Operandos		Clock	Transferencia	Bytes	Exemplo
short-label		16 ou 4	#	2	JS NEGATIVE

LOOP	LOOP short-label Loop		Flag	O D I T S Z A P C	
Operandos		Clock	Transferencia	Bytes	Exemplo
short-label		17/,5	#	2	LOOP AGAIN

LOOPE/LOOPZ	LOOPE/LOOPZ short-label Loop se =/Loop se zero		Flag	O D I T S Z A P C	
Operandos		Clock	Transferencia	Bytes	Exemplo
short-label		18 ou 6	#	2	LOOPE AGAIN

LOOPNE/LOOPNZ	LOOPNE/LOOPNZ short-label Loop se ñ =/Loop se ñ zero		Flag	O D I T S Z A P C	
Operandos		Clock	Transferencia	Bytes	Exemplo
short-label		19 ou 5	#	2	LOOPNE AGAIN

Chamada de sub-rotina (procedimento)

CALL	CALL alvo Chama um procedimento		Flag	O D I T S Z A P C	
Operandos		Clock	Transferencia	Bytes	Exemplo
near-proc		19	1	3	CALL NEAR_PROC
far-proc		28	2	5	CALL FAR_PROC
memptr 16		21+EA	2	2 4	CALL PROC_TABLE[SI]
regptr 16		16	1	2	CALL AX
memptr 32		37+EA	4	2 4	CALL [BX], TASK[SI]

Retorno

RET	RET valor-pop-opcional Retorno do procedimento		Flag	O D I T S Z A P C	
Operandos		Clock	Transferencia	Bytes	Exemplo
(intra-segment, não pop)		8	1	1	RET
(intra-segment, pop)		12	1	3	RET 4
(inter-segment, não pop)		18	2	1	RET
(inter-segment, pop)		17	2	3	RET 2

IRET	IRET (não há operando) Retorno da interrupção		Flag	O D I T S Z A P C R R R R R R R R	
Operandos		Clock	Transferencia	Bytes	Exemplo
(não há operandos)		24	3	1	RET

Diversas

HLT	HLT(não há operandos) Para		Flag	O D I T S Z A P C	
Operandos		Clock	Transferencia	Bytes	Exemplo
(não há operandos)		2	#	1	HLT

WAIT	WAIT (não há operandos) Aguarda pino Teste		Flag	O D I T S Z A P C	
Operandos		Clock	Transferencia	Bytes	Exemplo
(não há operandos)		3+5n	#	1	WAIT

STI	STI (não há operandos) Flag de ati de interr		Flag	O D I T S Z A P C . 1	
Operandos		Clock	Transferencia	Bytes	Exemplo
(não há operandos)		2	#	1	STI

CLI	CLI(não há operandos) Zera flag de Interrupção		Flag	O D I T S Z A P C . 0	
Operandos		Clock	Transferencia	Bytes	Exemplo
(não há operandos)		2	#	1	CLI

DAA	DAA(não há operandos) Ajusta decimal para soma		Flag	O D I T S Z A P C X X X X X X	
Operandos		Clock	Transferencia	Bytes	Exemplo
(não há operandos)		4	#	1	DAA

DAS	DAS(não há operandos) Ajuste decimal para subtração		Flag	O D I T S Z A P C U X X X X X	
Operandos		Clock	Transferencia	Bytes	Exemplo
(não há operandos)		4	#	1	DAS

NOP	NOP (não há operandos)		Flag	O D I T S Z A P C	
Operandos		Clock	Transferencia	Bytes	Exemplo
(não há operandos)		3	#	1	NOP

POP	POP destino Tira pal da pilha		Flag	O D I T S Z A P C	
Operandos		Clock	Transferencia	Bytes	Exemplo
registro		8	1	1	POP DX
seg-reg(CS ilegal)		8	1	1	POP DS
memória		17+EA	2	2 4	POP PARAMETER

POPF	POPF (não há operandos) Tira flags da pilha		Flag	O D I T S Z A P C R R R RR R R R R	
Operandos		Clock	Transferencia	Bytes	Exemplo
(não há operandos)		8	1	1	POPF

PUSH	PUSH fonte Coloca a pal na pilha		Flag	O D I T S Z A P C	
Operandos		Clock	Transferencia	Bytes	Exemplo
registro		11	1	1	PUSH SI
seg-reg (CS legal)		10	1	1	PUSH ES
memória		16+EA	2	2 4	PUSH RETURN CODE [SI]

PUSHF	PUSHF (não há operandos) Coloca flags na pilha		Flag	O D I T S Z A P C	
Operandos		Clock	Transferencia	Bytes	Exemplo
(não há operandos)		10	1	1	PUSHF

REP	REP (não há operandos) Repete a operação do string		Flag	O D I T S Z A P C	
Operandos		Clock	Transferencia	Bytes	Exemplo
(não há operandos)		2	#	1	REP MOVS DEST, SRCE

REPE/REPZ	REPE/REPZ (não há operandos) Repete a operação do string		Flag	O D I T S Z A P C	
Operandos		Clock	Transferencia	Bytes	Exemplo
(não há operandos)		2	#	1	REPE CMPS DATA, KEY

REPNE/REPNZ	REPNE/REPNZ (não há operandos) Repete a op de string enquanto ñ=ñ 0		Flag	O D I T S Z A P C	
Operandos		Clock	Transferencia	Bytes	Exemplo
(não há operandos)		2	#	1	REPNE SCAS INPUT LINE

XLAT	XLAT tabela-fonte Traduz		Flag	O D I T S Z A P C	
Operandos		Clock	Transferencia	Bytes	Exemplo
table-fonte		11	1	1	XLAT ASCII TAB

REFERÊNCIA BIBLIOGRÁFICA

[AMA95] AMARAL Jeferson. **Tutorial de Linguagem *Assembly*.** 1995, Endereço Eletrónico: http://www.inf.ufsm.br/~amaral/

[ANG95] ANGIOLETTI, Joacir L. **Protótipo de um Software Educacional para Educação Primária Usando Recursos de Multimídia**. Blumenau, 1995. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Departamento de Sistemas e Computação, Universidade Regional de Blumenau.

[BAC94] BACELLAR, Ricardo. **A Multimídia a Serviço da Educação**. Byte, São Paulo: Voice, Ano 3, n. 3, março/1994, p. 56-57.

[BAS96] BASTOS, Renato Nunes. **Tutorial de Assembler de Adam Hyde 1.0**. 1996, Endereço eletrónico : http://www.geocities.com/SiliconValley/Park/3174.

[BER96] BERRIDO, Alisson Cristian. **Desenvolvimento de um Software para Auxiliar no Ensino de Ciências para a 3ª série do 1° Grau**. Blumenau, 1996. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Departamento de Sistemas e Computação, Universidade Regional de Blumenau.

[CAM95] CAMILLO Junior, Helio da C. **Tutorial de Recomendações Ergonômicas Aplicadas a Implementação de Software**. Blumenau, 1995. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Departamento de Sistemas e Computação, Universidade Regional de Blumenau.

[DET96] DETTMER, Brigida. **Protótipo de Software para Ensino de Informática Básica**. Blumenau, 1996. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Departamento de Sistemas e Computação, Universidade Regional de Blumenau.

[DIA90] DIAS Jr., Wilson Alonso. **Microprocessadores 8086/8088: hardware & software.** São Paulo : McGraw-Hill, 1990.

[MAT97] MATTOS, Mauro M. e TAVARES, Antonio C. **VXT 1.0: Uma ferramenta didática para apoio ao ensino de software e hardware.** Anais XIV SEMINCO, Blumenau, 1997.

[MAT99] MATTOS, Mauro M. e TAVARES, Antonio C. **Desenvolvimento cooperativo de um ambiente de apoio ao ensino de conceitos básicos de hardware e software**. VII Workshop do ensino de informática, Anais SBC, Rio de Janeiro, 1999 (Aceito para Apresentar).

[MOC98] MocKingbird®. MocKingbird® CBT. 1998, Endereço eletrônico: http://www.mockingbird.com.

[MON92] MONTEIRO, Mario Antonio. Introdução à organização dos computadores. Rio de Janeiro : LTC, 1992.

[MOR88] MORGAN, Christopher L. **8086/8088: Manual do microprocessador de 16 bits.** Tradução Lars Gustav Erik Unonius. São Paulo : McGraw-Hill, 1988.

[MDS96] MDS Escola Aberta. Apresentação: MDS Escola Aberta. 1996, Endereço eletrônico : http://www.mds.com.br.

[OLI97] OLIVEIRA, Ramon de. **Informática Educativa.** Campinas : Papirus, 1997.

[ROC93] ROCHA, Ana Regina & CAMPOS, Gilda H. Bernardino. Avaliação da qualidade de software educacional. Em aberto, Brasília : ano 12, n.57, p. 32-44, jan/mar 1993.

[SES95] SESTREN, Giovani A. Protótipo de uma ferramenta de apresentação de recursos multimídia. Blumenau, 1995. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Departamento de Sistemas e Computação, Universidade Regional de Blumenau.

[VAL97a] VALENTE, José Armando. **Visão analítica da informática na educação no Brasil: a questão da formação do professor**. 1997, Endereço eletrônico : http://www.inf.ufsc.br/sbc-ie/revista.

[VAL97b] VALLE, Carla. **Educação de Pacientes através de Sistemas de Acesso Público**. 1997, Endereço eletrônico : http://www.inf.ufsc.br/sbc-ie/revista.

[VAU94] VAUGHAN, Tay. **Multimídia na prática**. São Paulo : Makron Books, 1994.