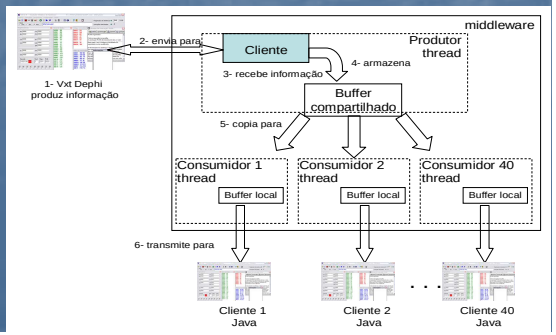
1997

PROPAGAÇÃO DA INTERFACE DO VXT USANDO O MODELO CLIENTE/SERVIDOR

Acadêmico: Elton Fernando Goedert

Orientador: Prof.Mauro Marcelo Mattos

2006



Introdução - VXt

- Projeto iniciado em 1997;
- Implementa um simulador de um processador Intel 8086;
- Motivado pelos problemas de aprendizagem na disciplina de Sistemas Operacionais.
- Versão atual somente executa em ambiente Windows;
- Última versão implementa arquitetura cliente/servidor primitiva que dificulta administração de entrada/saída de alunos.

Objetivos

- Implementar um *middleware* de comunicação entre o simulador da CPU (em Delphi) e as interfaces dos clientes (em Java);
- Permitir ao usuário do simulador do processador que a interface possa ser executada em uma máquina diferente daquela onde o simulador está sendo executado;

Projeto VXt - Publicações

- Ao longo do tempo, vários trabalhos foram publicados descrevendo o andamento da implementação do projeto:
  - MATTOS, Mauro M.; TAVARES, Antonio C. Virtual XT: um ambiente de apoio ao ensino de conceitos básicos de hardware e software. In SEMINARIO DA COMPUTAÇÃO, 6., 1997, Blumenau. Anais... Blumenau: FURB, 1997 p. 70-74.
  - MATTOS, Mauro M.; TAVARES, Antonio C.; OLIVEIRA, Emerson. VXt: descrição da implementação de um simulador de hardware. In: SEMINÁRIO DA COMPUTAÇÃO, 7., 1998, Blumenau. Anais… Blumenau: FURB, 1998. p. 138-149.
  - MATTOS, Mauro M.; OLIVEIRA, Emerson. Desenvolvimento de um ambiente de apoio ao ensino de conceitos básicos de hardware e software. In: SEMINÁRIO INTEGRADO DE INICIAÇÃO CIENTÍFICA, 5., 1999, Joaçaba. Anais... Joaçaba: FURB-UNIVALI, 1999. p. 79.
  - LINZMEIER, Marilene. Tutorial da linguagem Assembly utilizando o VXt. 1999. 58 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.
  - MATTOS, Mauro M.; TAVARES, Antonio C. Desenvolvimento cooperativo de um ambiente de apoio ao ensino de conceitos básicos de hardware e software. In: WORKSHOP DE ENSINO EM INFORMÁTICA, 7., 1999, Rio de Janeiro. Anais... Rio de Janeiro: [S.l], 1999a. Paginação irregular.
  - MATTOS, Mauro. M. ; TAVARES, Antonio C. VXt: experiência de desenvolvimento cooperativo de um ambiente didático. In: CONGRESO IBEROAMERICANO DE EDUCACIÓN SUPERIOR EN COMPUTACION, 7., 1999, Asunción. Anais… Asunción: [S.l.], 1999b. Paginação irregular.
  - MATTOS, Mauro M. et al. VXt: um ambiente didático para ensino de conceitos básicos de sistemas operacionais e arquitetura de computadores. In: WORKSHOP DE COMPUTAÇÃO DA REGIÃO SUL, 1., 2004, Florianópolis. Anais… Florianópolis: Unisul, 2004. Paginação irregular.

Projeto VXt - Bolsistas

- As primeiras versões (em DOS)
  - desenvolvidas pelos acadêmicos do curso de Ciências da Computação entre 1997 e 1998;
- Período: Mar/1998 a Fev/1999
  - Bolsista: **Emerson Oliveira**, Programa PIPe/Artigo 170
  - Projeto: PROJETO VXT, Subproj.: IMPLEMENTAÇÃO DA MÁQUINA VIRTUAL VXT
    - Unificação das versões em Pascal e geração da 1ª versão em Delphi
- Período: Out/2003 a mar/2004
  - Bolsista: **Filipe Renaldi**, Programa PIBIC/Artigo 170,
  - Projeto: PROJETO VXT – VIRTUAL XT, Subproj.: IMPLEMENTAÇÃO DA MÁQUINA VIRTUAL VXT
    - Criação da primeira versão cliente/servidor e Inclusão da biblioteca softx86
- Período: Mar/2006 a fev/2006
  - Bolsista: **Daniel Severo Estrázulas**, Programa PIBIC/Artigo 170
  - Projeto PROJETO VXT – VIRTUAL XT, Subproj.: IMPLEMENTAÇÃO DO VXT SERVER.
    - Implementação do espaço de IO
- Período: Mar/2006 a fev/2007
  - Bolsista: **Daniel Severo Estrázulas**, Programa PIBIC/Artigo 170
  - Projeto: PROJETO VXT – VIRTUAL XT, Subproj.: IMPLEMENTAÇÃO DO VXT CLIENT.
    - Construção de periféricos auxiliares (MDA e teclado) e conversão da interface para Java

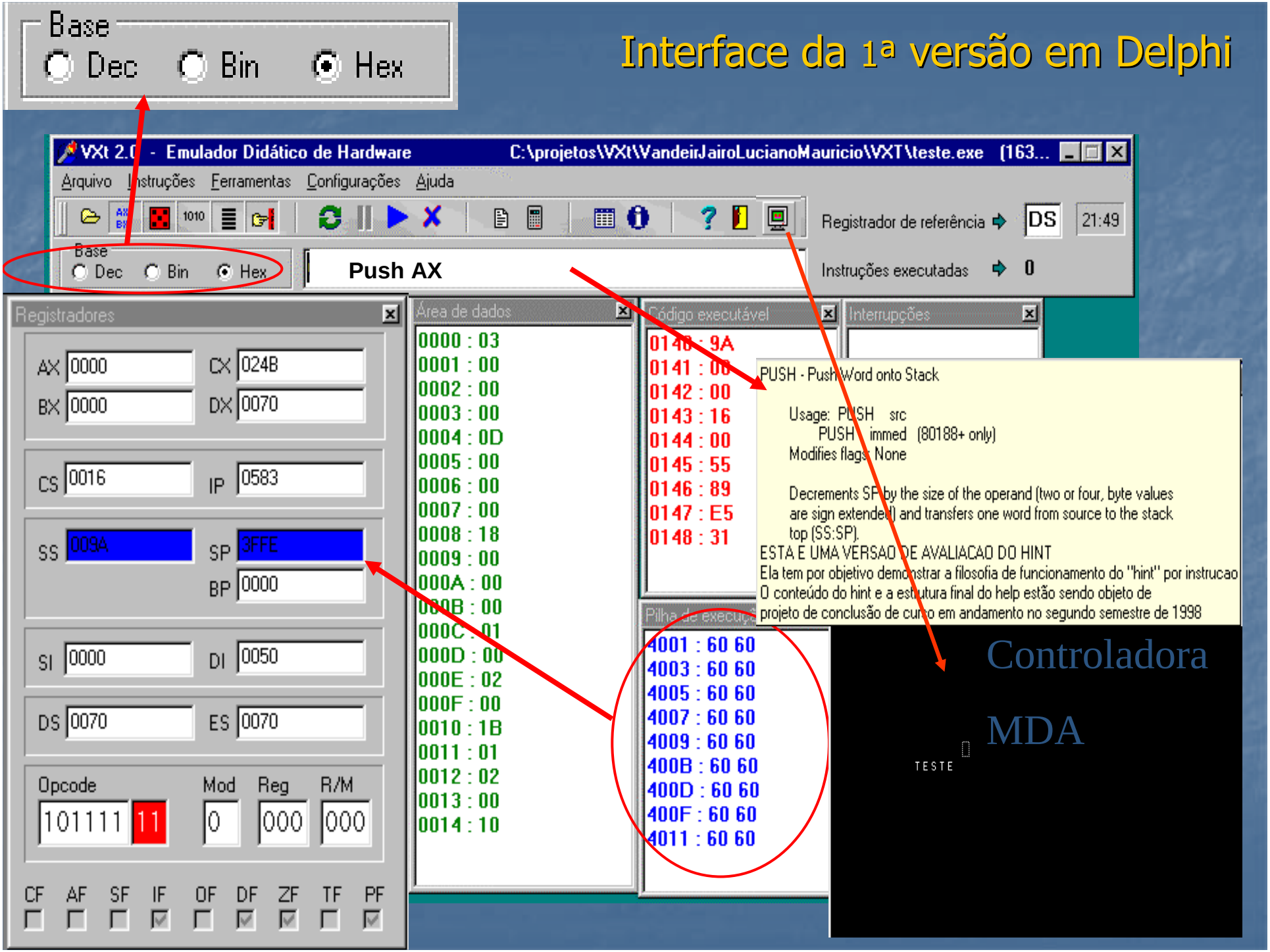
Motivação para a escolha do processador Intel 8086

- O processador a ser implementado deveria ser de conhecimento dos alunos;
- O processador alvo, deveria possuir todo um aparato de ferramentas de desenvolvimento tais como, compiladores, montadores, depuradores e ambientes de desenvolvimento adequados;
- O processador alvo deveria executar um sistema operacional que fosse de conhecimento do público-alvo;
- Deveria haver disponibilidade de literatura tendo em vista permitir a implementação do conjunto de instruções do processador;

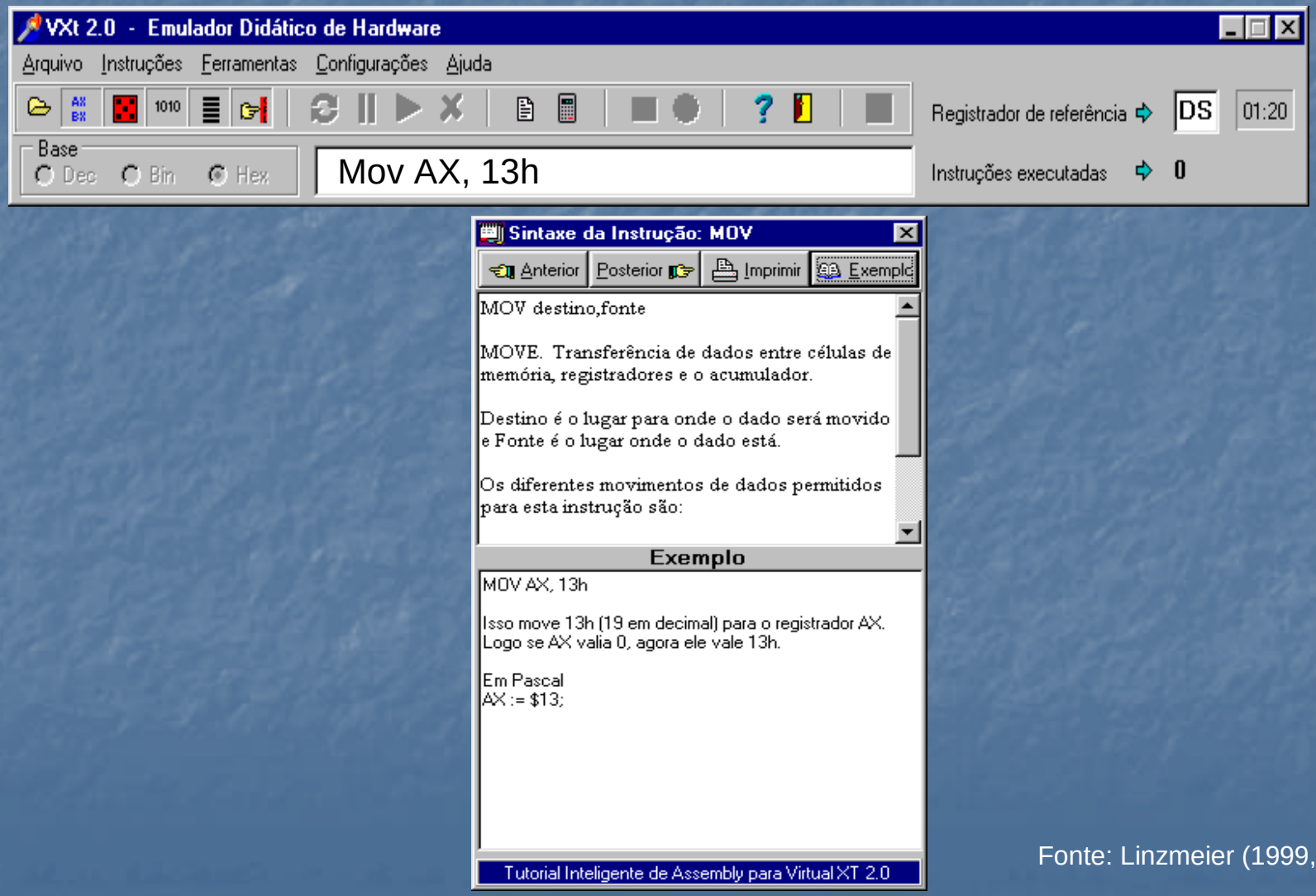
Interface da 1ª versão em Pascal

Fonte: Mattos,M. e Tavares,A. (1999).

Interface da 1ª versão em Delphi



1º TCC sobre VXt

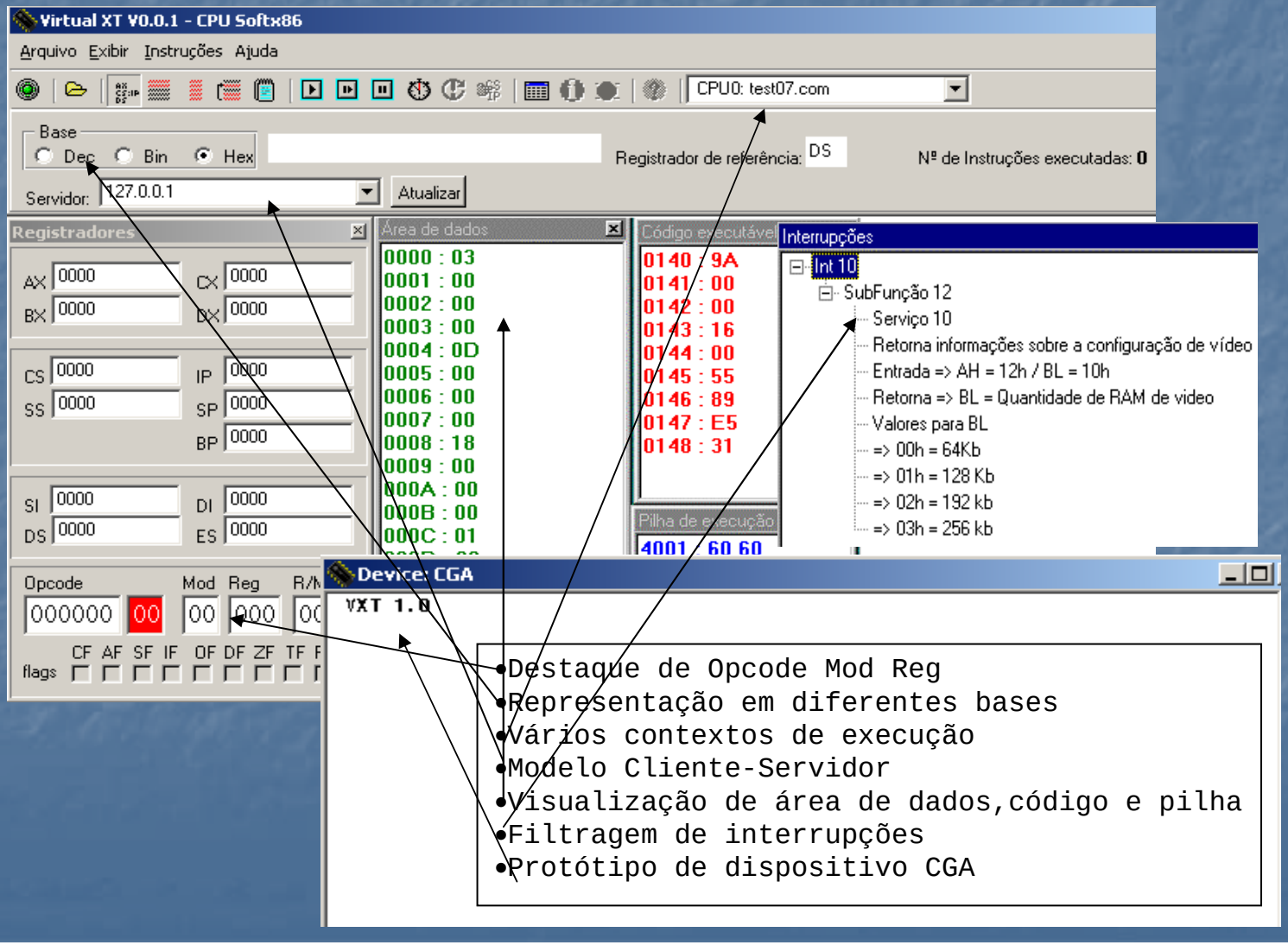


Fonte: Linzmeier (1999, p. 39, 40).

Características VXt - Versão 2004

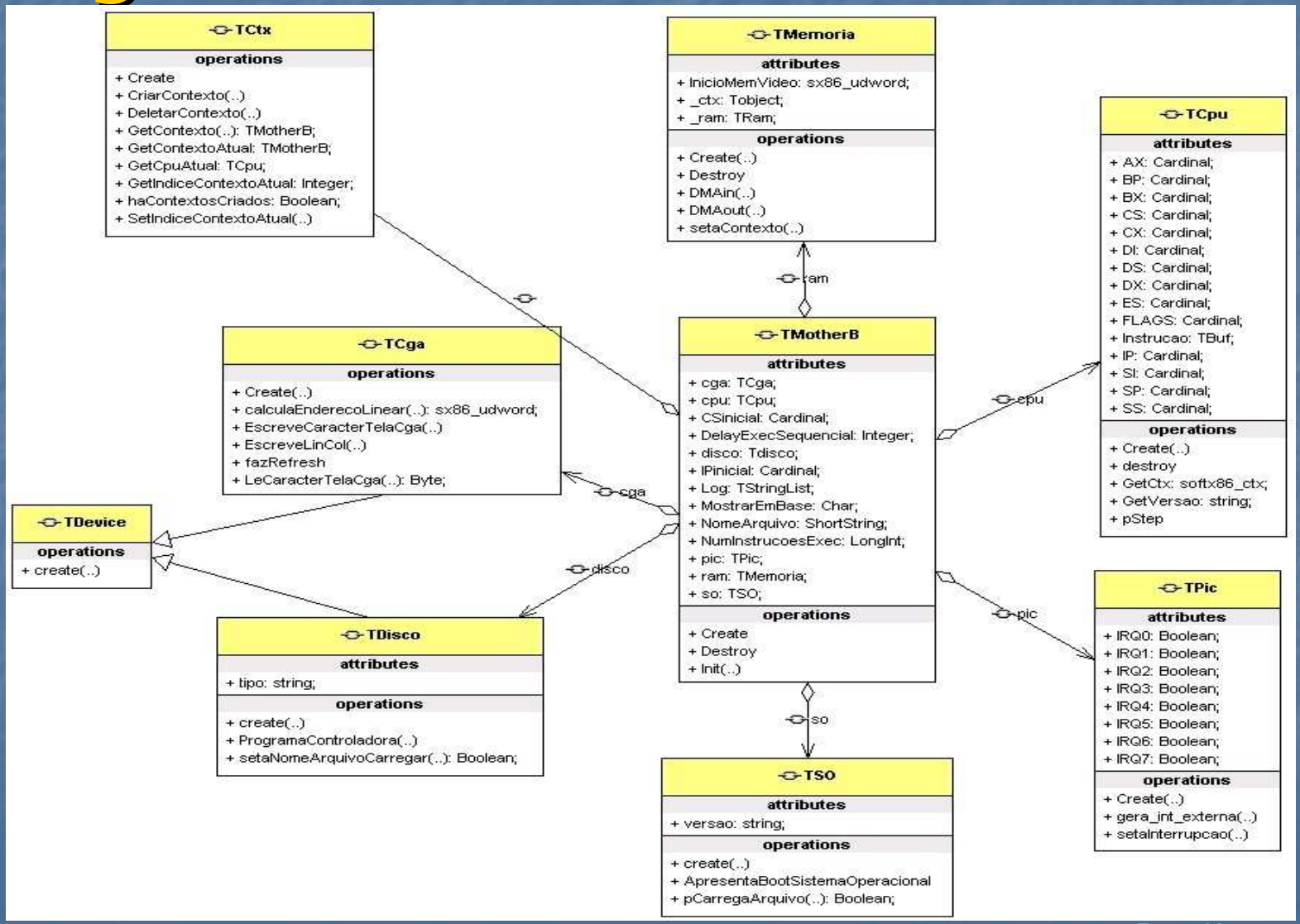
- Modelo cliente-servidor;
- Isolamento entre a interface com o usuário e a implementação do processador
  - Utilização da biblioteca softx86;
- Possibilidade de execução de vários exemplos em paralelo através do congelamento de contextos;
- Possibilidade de armazenar e resgatar o contexto de execução de um programa em outro momento;
- Possibilidade de salvar o "*log*" de execução do programa em arquivo para posterior análise;

Versão VXt 2004



- Destaque de Opcode Mod Reg
- Representação em diferentes bases
- Vários contextos de execução
- Modelo Cliente-Servidor
- Visualização de área de dados,código e pilha
- Filtragem de interrupções
- Protótipo de dispositivo CGA

Diagrama de Classes – VXt 2004

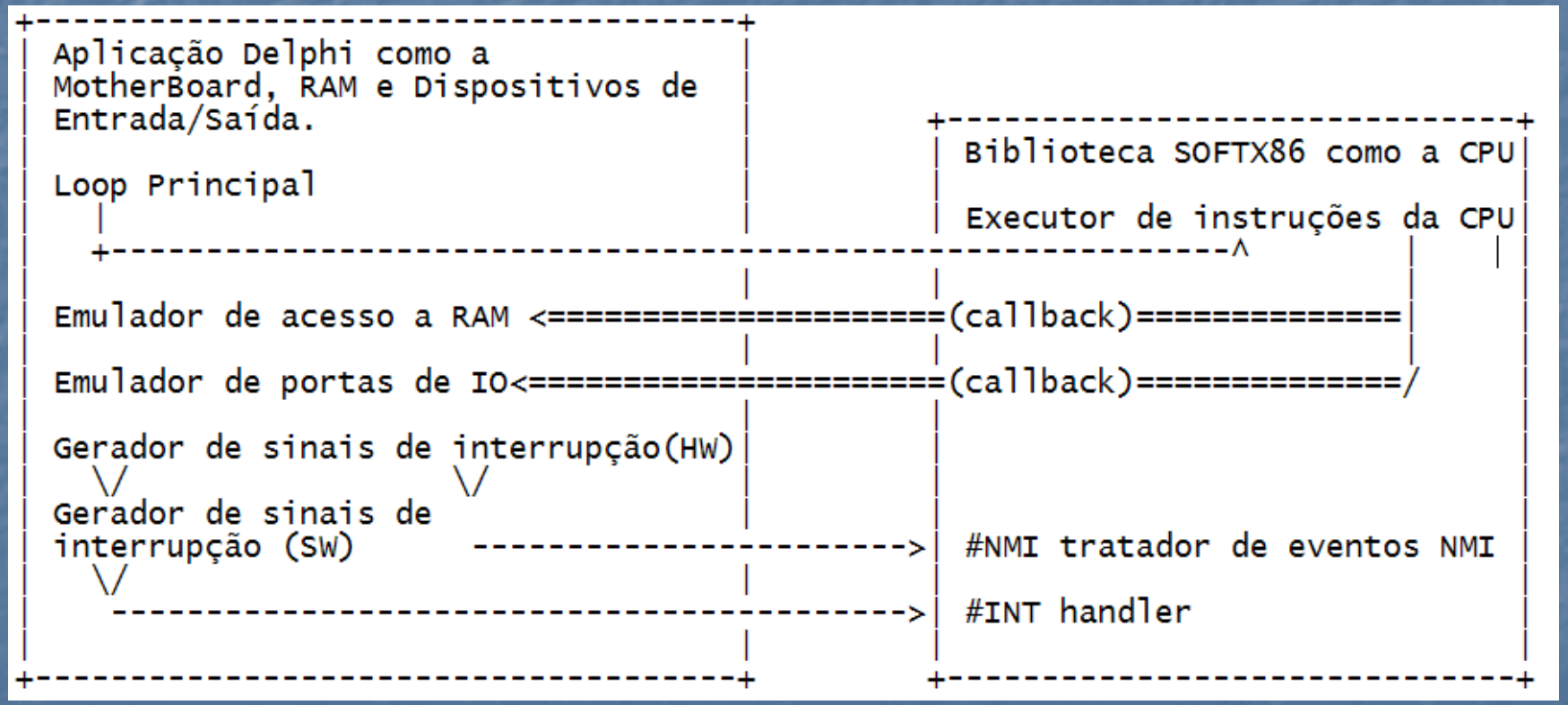


Fonte: Mattos et al. (2004).

Biblioteca softx86 - Características

- Precisão na emulação da CPU;
- Tamanho da memória definida pelo aplicativo Delphi (hoje 640Kb + 360Kb de Espaço de IO);
- Projeto que modela o comportamento dos seguintes sinais de hardware (na forma de *callbacks* no programa hospedeiro) :
  - Interrupção de relógio,
  - acesso ao espaço de I/O,
  - acesso ao espaço de endereçamento da memória,
  - interrupções por software,
  - interrupções de hardware,
  - Handshake com o processador de interrupções (PIC).

Integração entre VXt e softx86



A solução implementada

Características – VXt-C/S

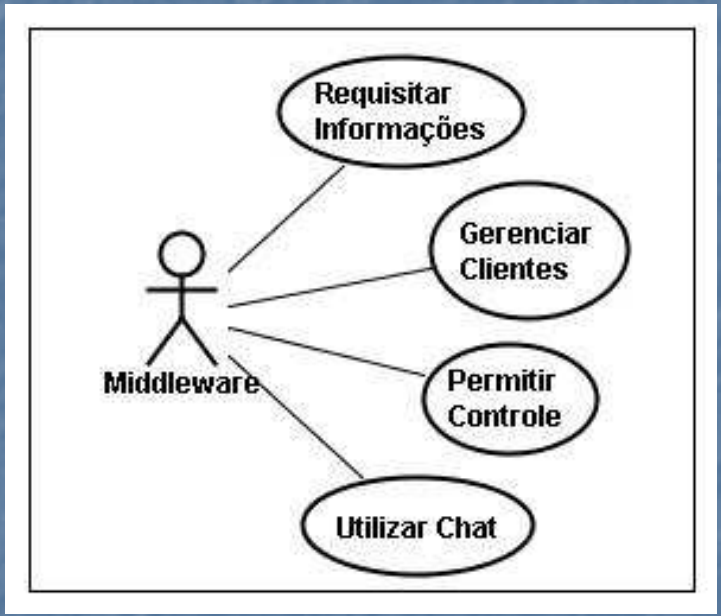
- O *middleware* comunica-se com o simulador do processador e media a propagação das informações de simulação entre os usuários conectados.
- O *middleware* gerencia a entrada e saída de usuários ao longo do processo de simulação.
- Os clientes tem a opção de conectar-se ao servidor, comunicar-se com o servidor através do chat e salvar o log das instruções executadas.

Base para implementação do VXt – C/S

- separação das funções cliente e servidor que estavam implementadas no mesmo fonte, fazendo com que a versão em Delphi seja responsável por operar o software e propagar as alterações de tela para o *middleware*;

- inclusão de um conjunto de meta-forms para viabilizar a separação entre a versão em Delphi e o *middleware*.

Diagrama Casos de Uso - VXt



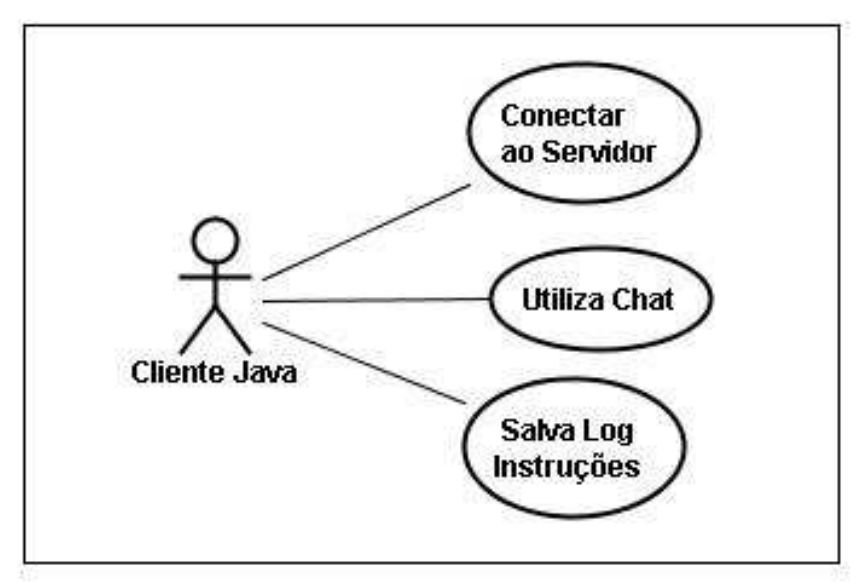
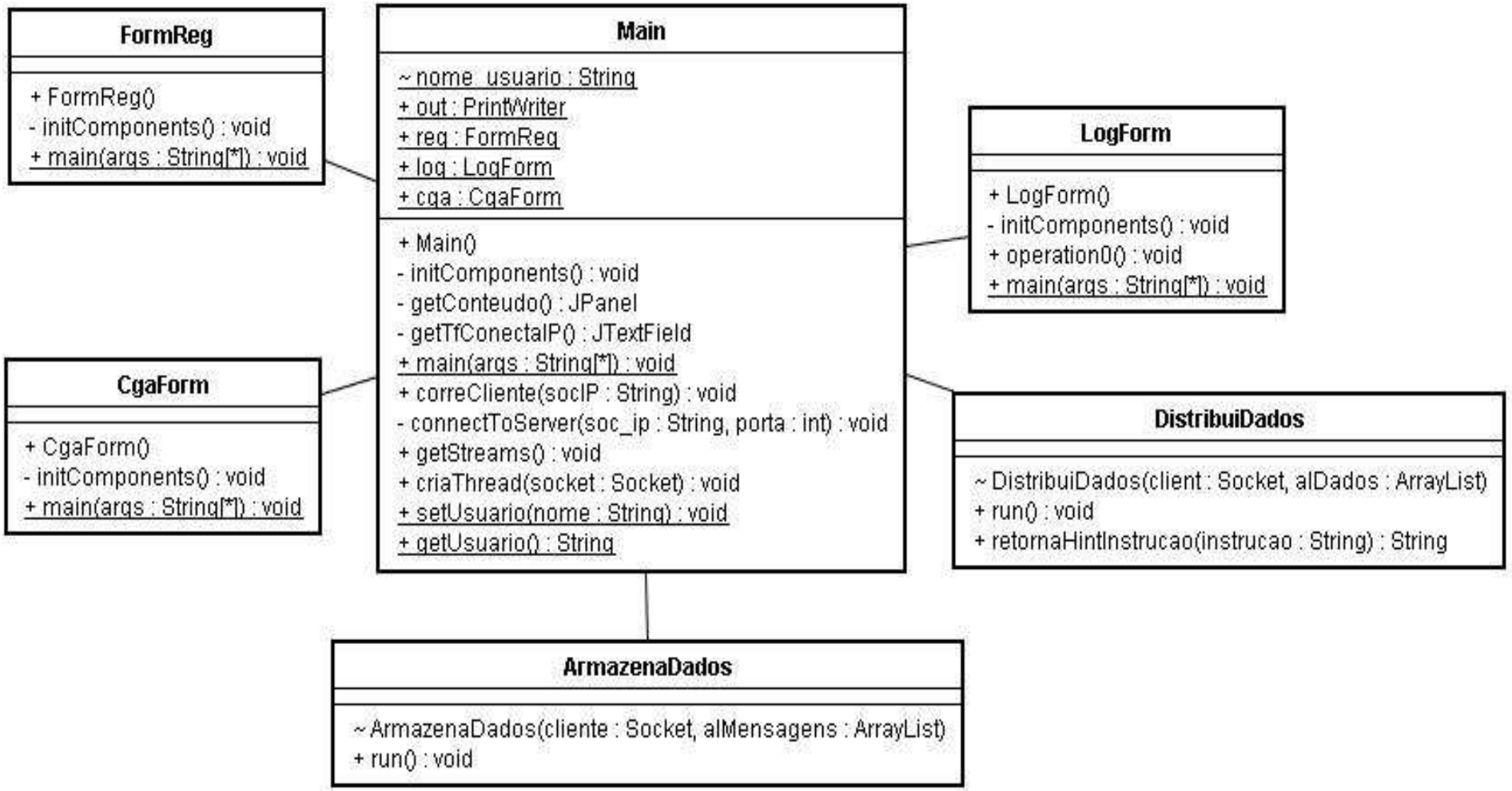
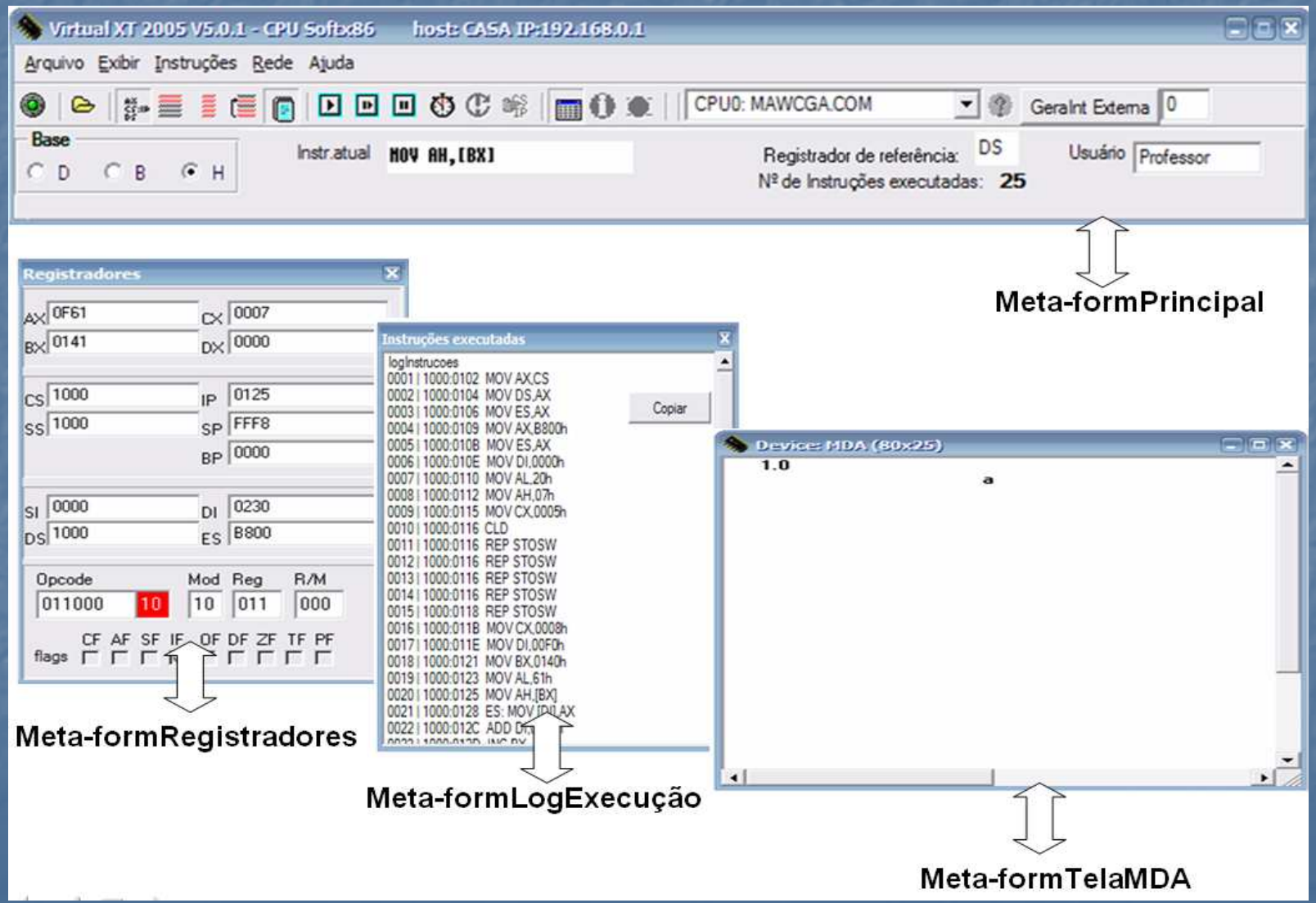


Diagrama de classes - Cliente Java



VXt – C/S


Virtual XT 2005 V5.0.1 - CPU Softx86
host: CASA IP:192.168.0.1
Arquivo Exibir Instruções Rede Ajuda
CPU0: MAWCGA.COM
GeraInt Externa
Base
D B H
Instr.atual
MOV AH,[BX]
Registrador de referência:
DS
Usuário
Professor
Nº de Instruções executadas:
25
Meta-formPrincipal
Registradores
AX 0F61
CX 0007
BX 0141
DX 0000
CS 1000
IP 0125
SS 1000
SP FFF8
BP 0000
SI 0000
DI 0230
DS 1000
ES B800
Opcode
Mod
Reg
R/M
011000
10
10
011
000
flags
CF AF SF IF OF DF ZF TF PF
Meta-formRegistradores
Instruções executadas
logInstrucoes
0001 | 1000:0102 MOV AX,CS
0002 | 1000:0104 MOV DS,AX
0003 | 1000:0106 MOV ES,AX
0004 | 1000:0109 MOV AX,B800h
0005 | 1000:010B MOV ES,AX
0006 | 1000:010E MOV DI,0000h
0007 | 1000:0110 MOV AL,20h
0008 | 1000:0112 MOV AH,07h
0009 | 1000:0115 MOV CX,0005h
0010 | 1000:0116 CLD
0011 | 1000:0116 REP STOSW
0012 | 1000:0116 REP STOSW
0013 | 1000:0116 REP STOSW
0014 | 1000:0116 REP STOSW
0015 | 1000:0118 REP STOSW
0016 | 1000:011B MOV CX,0008h
0017 | 1000:011E MOV DI,00F0h
0018 | 1000:0121 MOV BX,0140h
0019 | 1000:0123 MOV AL,61h
0020 | 1000:0125 MOV AH,[BX]
0021 | 1000:0128 ES: MOV [DI],AX
0022 | 1000:012C ADD DI,
Copiar
Meta-formLogExecução
Device: MDA (80x25)
1.0
a
Meta-formTelaMDA

Padrão de Projeto
Model View Controller (MVC)

- Características gerais
  - Separar dados (*Model*) da interface do usuário (*View*) e do fluxo da aplicação (*Controller*);
  - Permitir que uma mesma lógica de negócio possa ser acessada e visualizada através de várias interfaces;
  - A lógica de negócio é independente da camada de interface com o usuário (*View*).

(SILVEIRA, 2006, p. 29)

O padrão MVC no contexto VXt

- a versão em Delphi implementa parte do controlador PC-XT
  - Nesta versão controle da aplicação ainda permanece na aplicação Delphi, porém, as precondições para a separação foram implementadas;
  - A biblioteca softx86 implementa o emulador propriamente dito;
- o *middleware* implementa parte do controlador da aplicação
  - propaga as informações de interface para os clientes
  - propaga os comandos de abertura/fechamento de janelas e operação do sistema para os clientes;
- o módulo de interface do aluno implementa a camada de visão.

Arquitetura Cliente-Servidor

- Uma arquitetura cliente/servidor consiste em um processo de cliente e um outro processo de servidor, que podem ser distinguidos um do outro, embora possam interagir totalmente;
- A parte cliente e a parte servidor podem operar em diferentes plataformas de computador;
- Tanto a plataforma do cliente como a do servidor podem ser atualizadas independente uma da outra ;
- O servidor pode atender a vários clientes simultaneamente. Em alguns sistemas cliente/servidor, os clientes podem acessar vários servidores;

Bochenski (1995, p. 9),

Threads - Sincronização

- O uso de memória compartilhada entre as *threads* obriga o programador a sincronizar as ações de suas *threads*.
- Para isso, Java provê *monitores* ou *locks*.
  - um *lock é* como uma permissão para que apenas uma *thread* possa utilizar um recurso por vez.
- Cada objeto em Java possui um *lock* e ele deve ser obtido através do comando *synchronized*.

Cordeiro (2005)

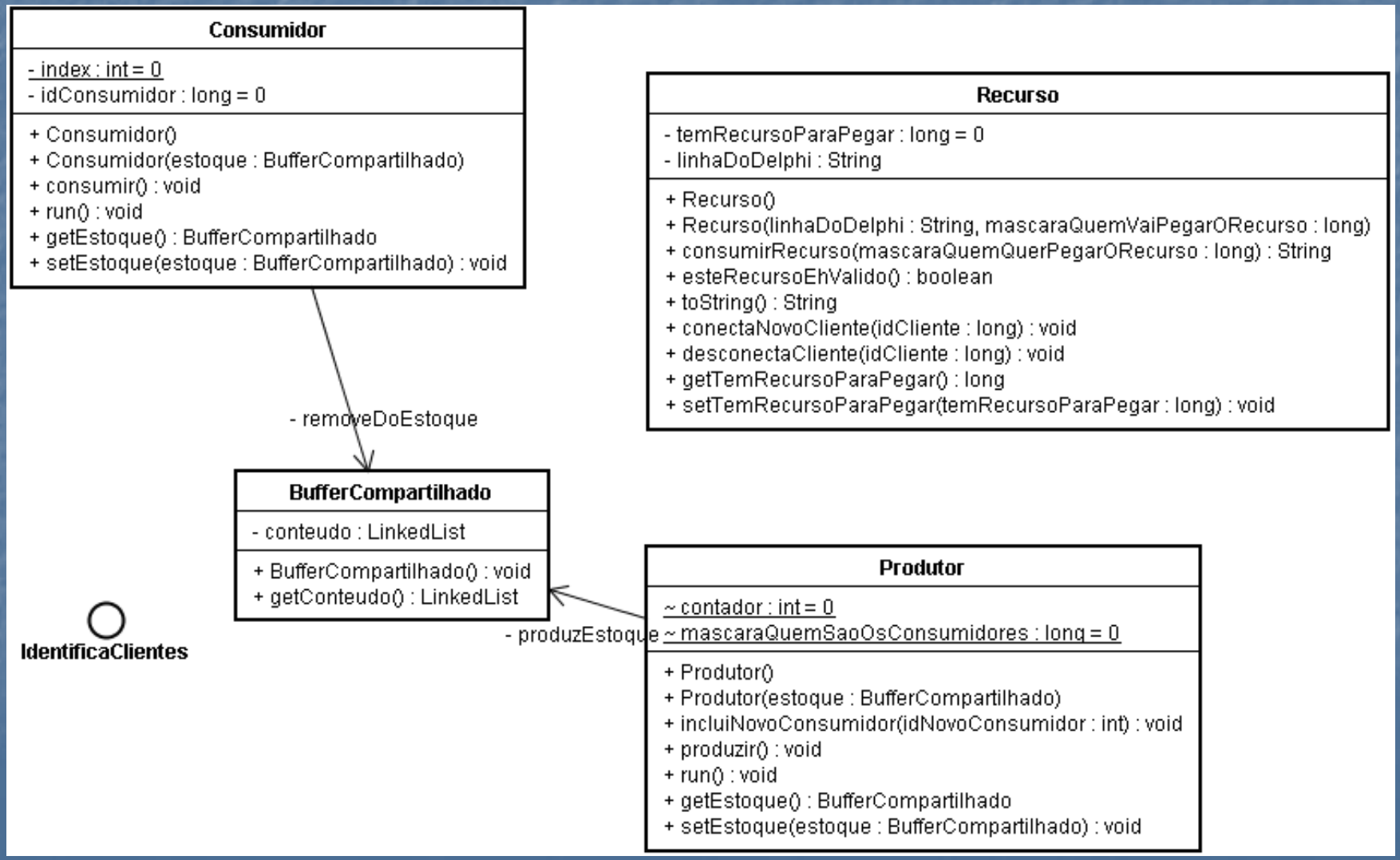
Threads – Elementos de Sincronização

- Wait(): fica a espera de uma notificação por parte do produtor. Quando notificada readquire o lock antes de recomeçar a execução.
- Notify(): libera a Thread para acessar a seção crítica.
- NotifyAll(): acorda todas as threads bloqueadas na fila de espera de wait() do objeto.

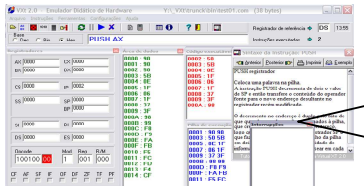
Produtor/Consumidor

- Dois processos compartilham um buffer.
- O produtor insere informação no buffer.
- O consumidor remove informação do buffer.

Diagrama Produtor/Consumidor



Funcionamento Arquitetura Produtor/Consumidor implementada no VXt



1- Vxt Dephi
produz informação

2- envia para

middleware

Produtor
thread

Cliente

3- recebe informação

4- armazena

Buffer
compartilhado

5- copia para

Consumidor 1
thread

Buffer local

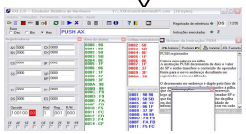
Consumidor 2
thread

Buffer local

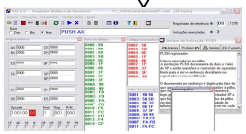
Consumidor 40
thread

Buffer local

6- transmite para



Cliente 1
Java



Cliente 2
Java

Cliente 40
Java

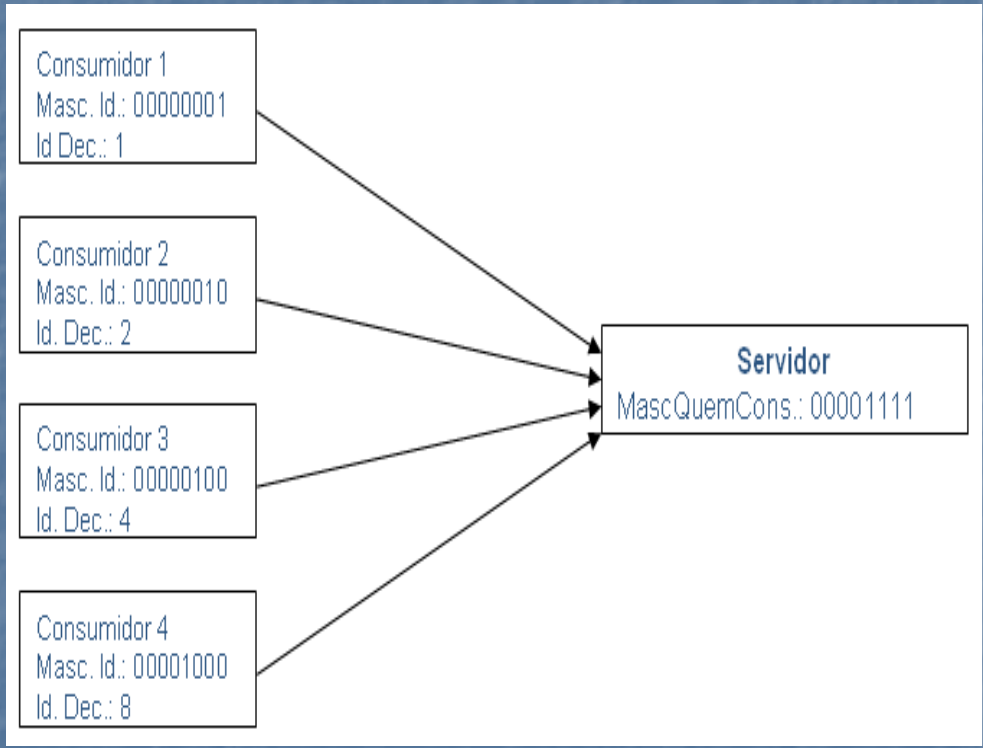
Interface Identificaclientes

- **Implementa um identificador único para, no máximo, 40 clientes.**

```
public interface IdentificaClientes {
    public static long codIdConsumidor[] = {
            1,          2,          4,          8,
            16,         32,         64,         128,
            256,        512,        1024,       2048,
           4096,        8192,       16384,      32768,
            65536,      131072,     262144,     524288,
            1048576,    2097152,    4194304,    8388608,
            16777216,   33554432,   67108864,   134217728,
            268435456,  536870912,  1073741824, 2147483648l,
            4294967296l, 8589934592l, 17179869184l, 34359738368l,
            68719476736l, 137438953472l, 274877906944l, 549755813888l
               };
}
```

Solução implementada

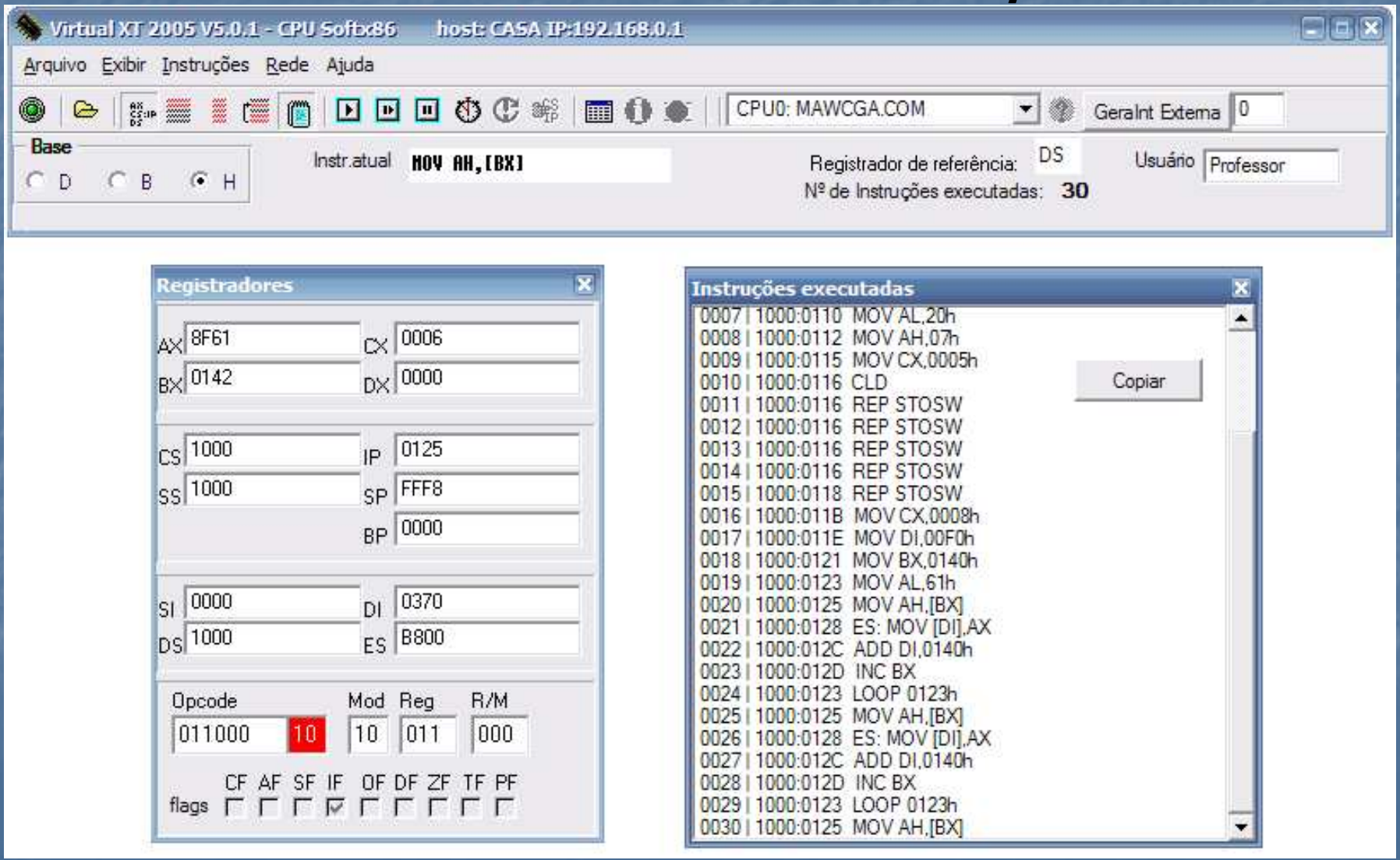
- Produtor possui uma máscara de quem pode consumir
- Cada cliente possui um identificador único
- O cliente acessa a seção crítica e verifica se existe alguma mensagem para ele.
  - Se existir, consome-a
  - Se for o último a consumir
    - remove a mensagem do buffer



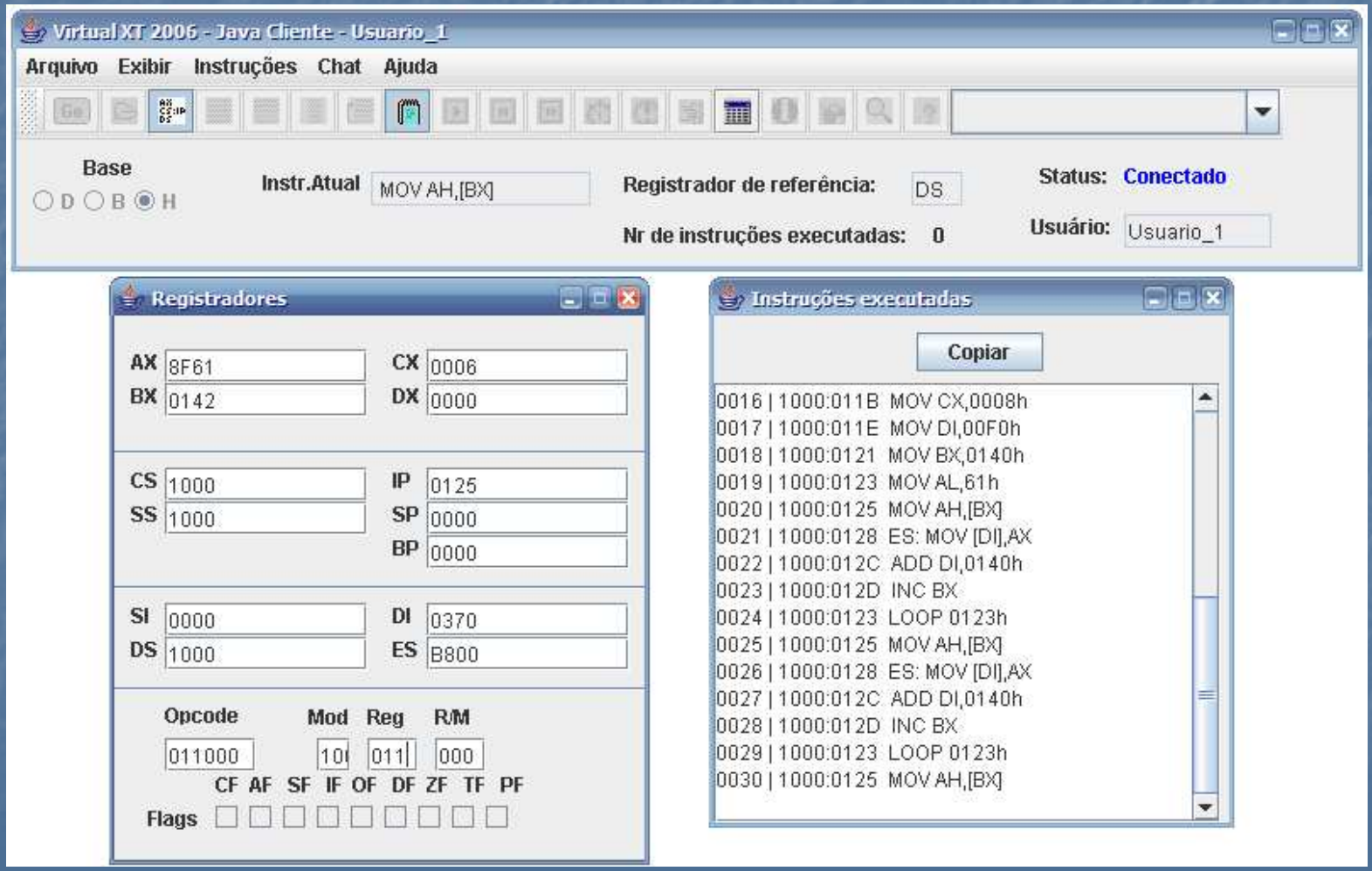
Trabalhos Correlatos

- Tutorial Linguagem Assembly (LINZMEIER, 1999);
- SOsim (MAIA, 2005);
- UserMonitor (USERMONITOR, 2005);
- VirtualClass (VIRTUALCLASS, 2005);

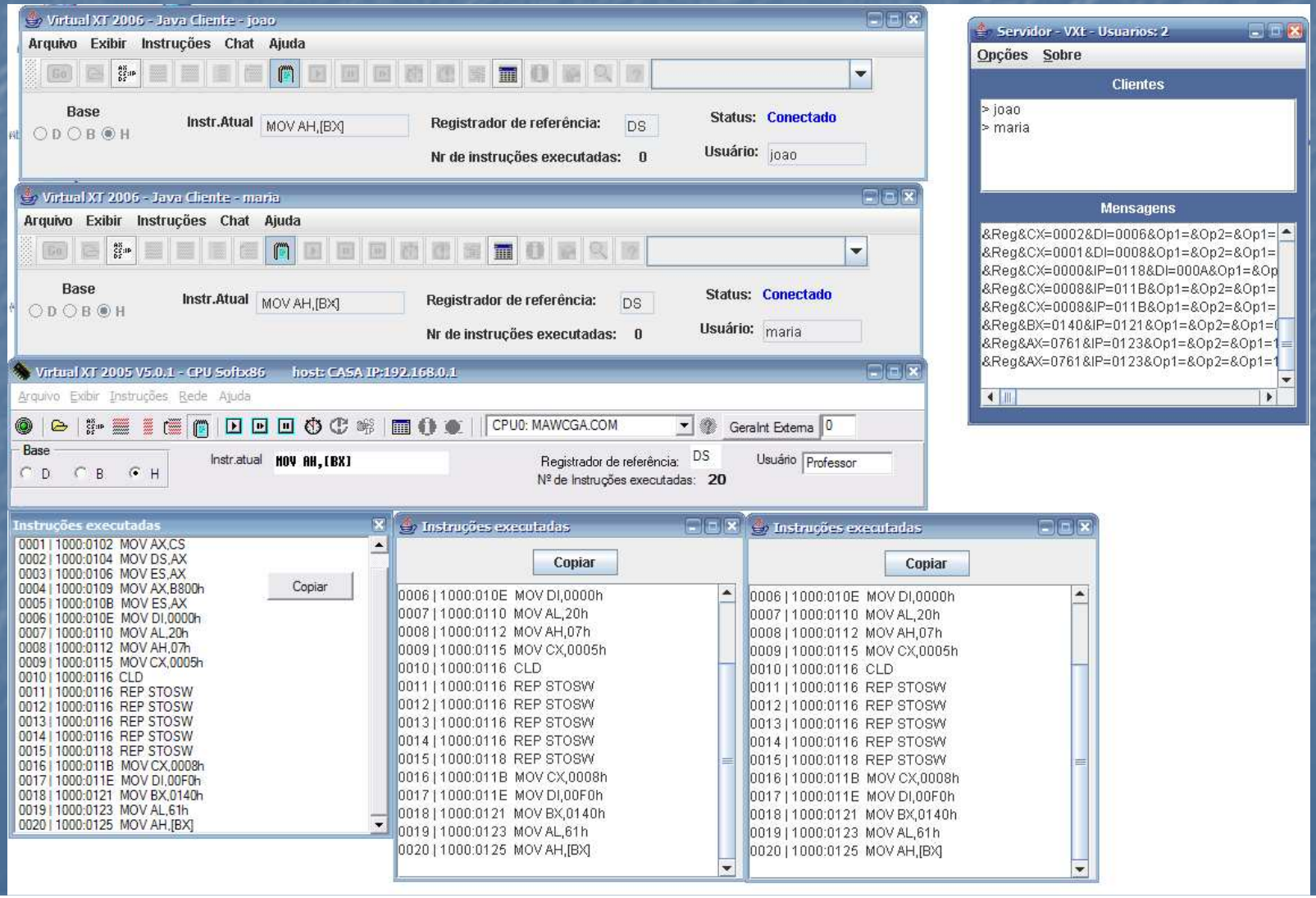
Interface - VXt–C/S



Interface Cliente



Funcionamento: Delphi + 2 clientes Java



Resultados e discussões

- Clientes não perdem informações enviadas pelo middleware;
- Clientes não podem obter o controle da aplicação;
- Clientes podem ser executados em linux, e o VXt em windows;
- Comunicação pode ser feita através da internet;

Dificuldades

- Conhecer o funcionamento do VXt;
- VXt estava sendo modificado por outras pessoas;
- Funcionamento e sincronização das Threads.

Apresentação da implementação