CENTRO UNIVERSITÁRIO FEEVALE
INSTITUTO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
CURSO DE CIÊNCIA DA COMPUTAÇÃO

Criação de um Kit de Desenvolvimento de Software para Programação Modular de Emuladores

(Título Provisório)

por

ANDRÉ WAGNER
andre.nho@gmail.com

Anteprojeto de Trabalho de Conclusão

Delfim Luiz Torok
delfimlt@feevale.br

Novo Hamburgo, setembro de 2006

SUMÁRIO

DADOS DE IDENTIFICAÇÃO

Área de Estudo: Emulação

Título provisório do trabalho: Criação de um Kit de Desenvolvimento de Software para Programação Modular de Emuladores

Orientador(a): Delfim Luiz Torok

Identificação do aluno:

Nome: André Wagner

Telefones:

Celular: 9691 1079

Residencial: 3593 5527

Comercial: 3594 4544

E-mail: andre.nho@gmail.com

RESUMO

Com o crescimento do mercado de dispositivos portáveis, video-games e ferramentas de virtualização, o uso de emuladores tem crescido e o desenvolvimento contínuo de novos emuladores tem se tornado uma constante no mercado atual. No entanto, não existe um conjunto de ferramentas nem de regras para este desenvolvimento e, assim, a implementação de novos emuladores geralmente é demorada e cara, e os programadores são muitas vezes forçados a reescrever muito código que poderia ser reaproveitado. Desta forma, este trabalho objetiva suprir esta necessidade, através do desenvolvimento de um Kit de Desenvolvimento de Software que auxilie neste desenvolvimento e facilite o trabalho do desenvolvedor, tornando sua programação mais simples, rápida e rentável.

Palavras-chave: Emulação, Kit de Desenvolvimento de Software, Biblioteca

REFERENCIAL CONCEITUAL

Um emulador é um programa de computador que simula o comportamento de uma arquitetura computacional, de modo que um software escrito para uma plataforma possa ser executado em outra. A Sociedade Britânica de Computação dá a seguinte definição: "Emulação é uma forma precisa de simulação que imita exatamente o comportamento ou as circunstâncias que se estão simulando. Um emulador permite que um tipo de computador opere como se fosse um tipo diferente de computador." [tradução nossa] (BURDETT, 1998, p. 30-31).

Neste respeito, um emulador assemelha-se a uma máquina virtual como a Máquina Virtual Java. Neste caso, o código escrito em java é compilado para um *bytecode*[1] independente de sistema operacional, que não será executado diretamente pelo processador, mas através de um interpretador. Um emulador funciona de forma similar, com a diferença de que ele não busca interpretar um código *bytecode* de uma linguagem criada especificamente para um interpretador, e sim um código executável escrito para uma plataforma existente, diferente daquela na qual se está usando o emulador (LINDHOLM, 1999).

Um exemplo de emulador é o Basilisk II. O Basilisk II é um software que emula um computador Macintosh da série 68k (produzido pela Apple Computer de 1984 a 1998) em uma série de plataformas e sistemas operacionais diferentes, como Windows, Linux, BeOS[2] e o MacOS X[3]. Desta forma, é possível utilizar um PC rodando Windows como se fosse um Macintosh executando o sistema operacional MacOS 8, que era um dos sistemas operacionais usados pelo Macintosh da série 68k. Mas para isto, o usuário deve possuir o MacOS 8 – o emulador emula apenas o hardware, não o software (BAUER, 2006).

1 Conjunto de bytes interpretáveis por uma máquina virtual. Do inglês byte + código (code).
2 Sistema operacional produzido pela Be Inc., de 1991 a 2000, tendo sido substituído pelo YellowTAB. Os programas escritos para o BeOS rodam no YellowTAB de forma nativa, sem qualquer alteração.
3 Sistema operacional produzido pela Apple Computer a partir de 2001.

Os emuladores são usados para diferentes fins:

- Um dos principais usos é facilitar o desenvolvimento de programas para plataformas que não possuem um compilador, ou nas quais o desenvolvimento seria pouco produtivo. Exemplo disto são dispositivos móveis como computadores portáteis e celulares, ou então dispositivos limitados como videogames. Neste caso, os emuladores geralmente são distribuídos pelos próprios fabricantes dos aparelhos;
- Os emuladores também são úteis para se executar aplicações de uma plataforma em outra. Assim, um usuário pode utilizar um software escrito para Windows em seu computador Macintosh. Um exemplo deste tipo de emulador é o QEMU, um emulador de processador genérico e de fonte aberto que atinge uma boa velocidade de emulação usando tradução dinâmica (BELLARD, 2006).
- Um terceiro uso é a emulação da própria plataforma onde o emulador está sendo executado. Para este tipo de emulação geralmente usa-se uma técnica chamada virtualização, onde muitas das instruções não precisam ser interpretadas, mas podem ser executadas diretamente, resultando assim numa maior velocidade (VMWARE, 2006). Isto é útil para se executar programas que rodem em sistemas operacionais diferentes (por exemplo, um usuário pode executar programas para Linux estando dentro do Windows). Um exemplo deste tipo de software é o VMWare, que é um programa especializado na virtualização de computadores PC[4].

4 Computador pessoal padrão IBM, que usa um processador Intel da linha x86. Do inglês *Personal Computer*.

MOTIVAÇÃO

O desenvolvimento de um novo emulador é realizado de forma modular. O programador que está desenvolvendo um emulador precisa ter um grande conhecimento, tanto da plataforma de origem (a que será emulada), como da plataforma de destino (onde o emulador será executado). É necessário desenvolver a emulação de cada um dos componentes da plataforma de origem. Isto significa que o programador deve desenvolver um módulo que emule uma CPU[5], um módulo que emule a placa de vídeo, um módulo que emule a memória, e assim por diante. Depois, estes módulos são unificados, resultando assim no emulador da plataforma completa (BORIS, 1999).

É importante notar que o programador não precisa necessariamente ter conhecimento interno dos componentes do hardware que será emulado. A preocupação não é com o funcionamento interno do componente, mas sim com seu comportamento. Se um programador desenvolver, por exemplo, a emulação da operação ADD[6] de um certo microprocessador, ele não precisa conhecer a forma como isto é feito internamente pelo processador, mas sim saber que quando a instrução equivalente à soma (ADD) de dois números for executada pela CPU, o resultado será armazenado em um determinado registrador, consumindo um determinado número de ciclos de *clock*[7], de acordo com a especificação daquela CPU. (DEL BARRIO, 2001).

O desenvolvimento de um emulador pode ser uma tarefa desafiadora, e em geral os desenvolvedores são obrigados a encarar uma série de dificuldades, tais como:

- Muitas vezes dois computadores diferentes usam o mesmo tipo de CPU. Mas o programador geralmente não consegue reaproveitar o código de um emulador escrito por outro programador em outro emulador, pois não existe um conjunto de regras padrão para o desenvolvimento de emuladores, e cada programador escreve o código da forma que lhe parece melhor (misturando código da CPU com código da placa de vídeo, por exemplo);

5 Microprocessador, ou Unidade de Processamento Central. Do inglês *Central Processing Unit*.
6 Instrução *assembly* geralmente usada para somar dois números. Do inglês add (soma).
7 Sinal eletrônico digital usado para sincronizar as ações de dois ou mais circuitos.

- O programador precisa não apenas conhecer a plataforma de origem (que será emulada) mas também a de destino (onde o emulador será executado), sendo assim necessário realizar o dobro de pesquisa;
- A não ser que o programador mantenha a portabilidade em mente desde o início, geralmente o emulador não é portável, pois muitas das atividades que o emulador executa na plataforma de destino não são padronizadas;
- Existem certas operações que são padrão para todos os emuladores. Por exemplo, todos os computadores acessam algum tipo de memória RAM[8], e praticamente todos tem uma saída para vídeo. Assim, o programador é obrigado a "reinventar a roda" a cada novo emulador;
- Muitas tarefas são necessárias para o programador mas serão desnecessárias para o usuário final. Por exemplo, o programador é obrigado a desenvolver algum tipo de *debugger*[9] para microprocessador que ele está querendo emular. O desenvolvimento de um debugger pode ser, às vezes, tão ou mais complexo que o desenvolvimento do próprio microprocessador;
- Muitos algoritmos são padrão para a maioria dos emuladores, mas geralmente são reimplementados para cada novo emulador. Exemplo disso é o uso de filtros gráficos, que permitem que o usuário tenha uma visão melhorada da saída de vídeo da plataforma que está sendo emulada. No caso de emuladores de computadores portáteis, por exemplo, o usuário muitas vezes deseja que a saída de vídeo do emulador seja maior do que a saída de vídeo do próprio computador portátil.

É a esses problemas que este trabalho se dirige, buscando uma forma de identificar as tarefas que são padrão para todos os emuladores, e desenvolvendo e implementando um Kit de Desenvolvimento de Software[10], composto por uma biblioteca[11] e por ferramentas auxiliares que padronizem e facilitem o desenvolvimento de emuladores.

8 Memória de Acesso Aleatório. Do inglês *Random Access Memory*.
9 Programa que exibe um "mapa" de um microprocessador, permitindo que o usuário ou programador saiba exatamente o que se passa em seu interior.
10 Pacote formado por um conjunto de ferramentas que auxiliam desenvolvedores de software na implementação de novos aplicativos. Geralmente é conhecido por SDK (do inglês *Software Development Kit*)
11 Coleção auxiliar de funções que fornecem serviços a um programa independente.

OBJETIVOS

Objetivo geral:

Este trabalho objetiva projetar, desenvolver e implementar um Kit de Desenvolvimento de Software que dê suporte à criação de novos emuladores, permitindo e facilitando a programação de emuladores de forma modular e reaproveitável, centralizando as operações padrão e oferecendo uma interface gráfica de usuário.

Objetivos específicos:

1. Estudar a modelagem tradicional de emuladores, visando compreender sua forma de funcionamento e identificar quais são as atividades comuns à maioria deles;
2. Estudar os problemas e dificuldades encontrados no modelo tradicional de desenvolvimento de emuladores;
3. Elaborar uma nova metodologia de desenvolvimento para criação de emuladores, que possa unificar as atividades padrão através do uso de uma biblioteca e de ferramentas auxiliares;
4. Definir as ferramentas a serem implementadas e uma API[12] com a descrição das funções da biblioteca;
5. Implementar ferramentas auxiliares que dêem suporte à programação modular, e uma biblioteca que centralize as operações padrão e ofereça uma interface gráfica[13];
6. Criar um emulador de arquitetura relativamente simples, de modo a testar a biblioteca e as ferramentas desenvolvidas;
7. Analisar e validar o processo de criação do novo emulador, identificando os pontos fracos do projeto e possíveis melhorias, bem como traçar idéias para o futuro.

12 Interface de Programação de Aplicativos: conjunto de funções de uma biblioteca sobre o qual o programador pode implementar um aplicativo. Do inglês *Application Programming Interface.*

13 Método de interação com o computador através de manipulação direta de imagens gráficas e controles.

METODOLOGIA

No sentido de alcançar os objetivos propostos para este trabalho, se fará, além do projeto, o desenvolvimento e a implementação da biblioteca e das ferramentas propostas, bem como de um software que faça uso destas ferramentas, de modo a permitir a avaliação da viabilidade da solução proposta e dos problemas encontrados. Para tal, divide-se o trabalho nas etapas relacionadas:

I. Pesquisa de embasamento teórico envolvendo a modelagem tradicional de emuladores, visando compreender sua forma de funcionamento e identificar as atividades padrão – isto é, as atividades que se repetem na criação da maior parte dos emuladores;
II. Realização de estudo, buscando alistar as principais dificuldades e problemas existentes no modelo tradicional de desenvolvimento de emuladores. Neste estudo, procura-se também agrupar os componentes por tipo, buscando identificar as atividades padrão de cada tipo, de modo a oferecer ao programador uma biblioteca que permita o desenvolvimento modular;
III. Elaboração, a partir dos estudos realizados, de uma nova metodologia que permita unificar as atividades padrão através do uso de ferramentas auxiliares e de uma biblioteca, fazendo com que o código desenvolvido pelo programador seja modular, de modo que ele possa reaproveitar seu código e o de outros;
IV. Desenvolvimento de uma API de biblioteca que permita que o programador tenha acesso às operações padrão e a uma interface gráfica, bem como definição das as ferramentas auxiliares a serem implementadas e de sua utilização dentro da metodologia desenvolvida;
V. Implementação da biblioteca e das ferramentas auxiliares desenvolvidas na fase anterior, utilizando as linguagem de programação C e *bash*[14] e de forma portável através do uso de ferramentas GNU[15];

14 Linguagem de linha de comando baseada na linguagem *Bourne Shell (sh)*. Foi criada em 1987 por Brian Fox, e é a linguagem de linha de comando padrão de vários sistemas operacionais, como o GNU/Linux e o MacOS X.

15 Ferramentas livres para desenvolvimento (bibliotecas, utilitários e compiladores), originalmente desenvolvidas para o sistema operacional GNU Hurd, mas portadas para uma grande variedade de plataformas e sistemas operacionais. O nome GNU origina-se, em inglês, do acrônimo recursivo *GNU's Not Unix* (GNU Não é Unix).

VI. Desenvolvimento e implementação de um emulador da plataforma Atari 2600[16], fazendo uso da biblioteca e das ferramentas auxiliares desenvolvidas, de modo a testar o software;
VII. Análise e validação do teste, verificando quais os pontos fracos e os pontos fortes do programa, alistando as possíveis correções e melhorias, e traçando planos para o futuro;
VIII. Apresentação do trabalho à banca avaliadora, apresentando o conteúdo através do uso de recursos multimídia, bem como mostrando o emulador desenvolvido para teste, de modo a exemplificar o funcionamento da biblioteca e das ferramentas.

As tarefas serão realizadas de acordo com um cronograma, sendo que a documentação formal será desenvolvida concomitantemente com as fases apresentadas.

16 Plataforma de jogos desenvolvida pela Atari Inc. em 1977, contendo um microprocessador MOS Technology 6507, 128 bytes de memória RAM, e um chip controlador de vídeo e som chamado Television Interface Adapter (TIA).

CRONOGRAMA

As fases deste projeto serão divididas nas etapas apresentadas na metodologia. O cronograma que deverá ser adotado é o expresso na Tabela 1.

Tabela 1 – Cronograma de Trabalho

Etapas	**Trabalho de Conclusão I**					**Trabalho de Conclusão II**				
	2006					**2007**				
	ago	set	out	nov	dez	fev	mar	abr	mai	jun
I										
II										
III										
IV										
V										
VI										
VII										
VIII										

BIBLIOGRAFIA

BAUER, Christian. **The Official Basilisk II Home Page**. Disponível em <http://basilisk.cebix.net/>. Acesso em: 15 ago. 2006.

BELLARD, Fabrice. **QEMU:** Open Source Processor Emulator. Disponível em <http://fabrice.bellard.free.fr/qemu/about.html>. Acesso em: 15 ago. 2006.

BORIS, Daniel. **How Do I Write an Emulator?** - Part 1. 1999. Disponível em <http://personals.ac.upc.edu/vmoya/docs/HowToDanBoris.txt>. Acesso em: 15 ago. 2006.

BURDETT, Arnold et al. **A Glossary of Computing Terms:** by the British Computer Society**.** Cambridge: Cambridge University Press, 1998. 86p.

DEL BARRIO, Victor Moya del. **Study of the Techniques for Emulation Programming**. Barcelona: Facultat d'Informàtica de Barcelona, 2001. Dissertação de graduação. Disponível em <http://personals.ac.upc.edu/vmoya/docs/emuprog.pdf>. Acesso em: 15 ago. 2006.

LINDHOLM, Tim; YELLIN, Frank. **The Java™ Virtual Machine Specification.** 2ª. ed. Boston: Addison-Wesley Professional, 1999, 473 p.

VMWARE Inc. **Virtualization Overview**: VMWare Whitepaper. Disponível em <http:// www.vmware.com/pdf/virtualization.pdf>. Acesso em: 15 ago. 2006.