Uma Arquitetura Adaptativa

S.S. Barretto and J. J. Neto

***Abstract*— We try to modify an existing CPU architecture to include adaptivity. The chosen architecture, the "Patinho Feio", was the first computer developed in Brazil. The authors list the reasons for this choice, the changes made to implement adaptivity and future plans.**

***Keywords*— Adaptivity, Adaptive Devices. Architecture**

I. INTRODUÇÃO

O computador Patinho Feio foi desenvolvido na Escola Politécnica da USP no início da década de 70 por alunos do curso de pós graduação e foi utilizado pelos autores para aprendizado no desenvolvimento de software básico.

Em um reencontro recente dos autores foi apresentada uma nova "versão" do Patinho Feio, agora no formato de uma APP desenvolvida para dispositivos móveis da Apple.

Surgiu então a idéia de utilizar essa APP como base de estudo para o desenvolvimento de uma arquitetura adaptativa.

Entre os fatores que levaram a essa escolha podemos citar:

- Pelo fato de ser totalmente feita por "software", permite fácil modificação, com o único custo sendo o tempo gasto no seu desenvolvimento.

- Por se tratar de uma arquitetura razoavelmente simples, qualquer modificação pode ser feita em pequeno espaço de tempo.

- Apesar de se tratar de um projeto acadêmico, existe farta documentação sobre o Patinho Feio, tanto no formato de teses como em livro [1][2].

- Do mesmo modo existe uma razoavel quantidade de "sofware" básico desenvolvido para ele, como montador, compilador, ligador e carregador.

II. DESENVOLVIMENTO

Quando se fala em arquitetura adaptativa estamos nos referindo a uma arquitetura que possa ser alterada por meio de programas.

Embora a existência de hardware com instruções que possam ser modificadas por programa não seja exatamente um assunto novo[3], não existe muita literatura a respeito de arquiteturas adaptativas.

Existe pelo menos uma proposta sobre uma linguagem de alto nível para facilitar o desenvolvimento de programas automodificáveis[4].

Para o desenvolvimento, fizemos algumas premissas básicas:

- A arquitetura adaptativa deveria ser compatível com os softwares existentes, permitindo que eles pudessem ser executados sem adaptação.

- A indicação de que se trata de uma instrução adaptativa seria realizada por meio de um prefixo na instrução.

- O código que trata as instruções adaptativas seria formado por instruções similares àquelas que compõem o conjunto de instruções do Patinho Feio.

- A área de memória reservada ao código adaptativo deveria ficar separada da área de memória reservada aos outros programas.

Baseado nessas premissas, escolhemos o código de operação 9F, que na primeira versão do Patinho Feio era uma instrução reservada, sem operação associada, como indicador de adaptatividade.

Posteriormente foi lembrado que, embora a versão 1.0 do Patinho Feio tivesse apenas 4k bytes de memória, foi feita uma segunda versão com 32k bytes, onde o acesso às posições de memória além dos 4k da página atual era realizado por meio de endereçamento indireto.

Para indicar que uma instrução de acesso à memória tinha endereçamento indireto era usado o prefixo 9F, e o endereço de memória (de 12 bits) da instrução indicava a área de 16 bits onde estava o endereço real de memória, sendo o bit mais significativo reservado para indicar níveis adicionais de indireto.

Assim, para manter compatibilidade com a versão 1.0 e 2.0, resolvemos criar uma versão 3.0, onde podem existir vários modos de operação, que são selecionados por uma combinação do código 9F com instruções que são raramente usadas, como operações lógicas do acumulador com dados lidos do registrador de chaves do painel.

As combinações escolhidas foram:

- 9F8A : seleciona modo de operação 1.0
- 9F8B : seleciona modo de operação 2.0
- 9F8C : seleciona modo de operação 3.0
- 9F8D: se estiver no modo 3.0, esta instrução funciona como o prefixo de indireto da versão 2.0, com a diferença que o bit mais significativo é um bit de endereço, permitindo o acesso a 64k bytes de memória.
- 9F9F : este é o prefixo que indica que a próxima instrução e uma instrução adaptativa.
- se a instrução 9F for seguida por outro valor, ela opera como no modo 2.0, permitindo que os programas tenham acesso a qualquer posição dos primeiros 32k bytes de memória.

Na arquitetura proposta os programas de usuário e sistema operacional ficariam limitados aos primeiros 32k de memória e a área acima disso seria reservada ao código adaptativo.

A execução de uma instrução adaptativa (desencadeada pela sequência 9F9F) provocaria o seguinte sequência de eventos:

- a interrupção é inibida (independente da interrupção já

estar inibida ou não, e sem alterar essa condição).

- a instrução a ser executada é armazenada no endereço 8002 (hexa), 8003 se longa e 8004 se indireta.
- o endereço de retorno (próxima instrução) é armazenado nas posições 8006 e 8007 e é executado o código adaptativo armazenado a partir da posição 8008.
- o código adaptativo pode decidir por executar operações pré-adaptativas, ou seja, operações executadas antes da execução da instrução atual, decidir se executa ou não a instrução atual e se executa ou não operações pós-adaptativas.
- o código adaptativo pode inclusive optar por se automodificar, sendo que a único ponto a ser considerado nesse caso, é que todo esse processamento é efetuado com interrupção inibida, podendo causar problemas em alguma operação de entrada ou saida que esteja em andamento, caso esse processamento seja muito longo.
- durante a execução do código adaptativo algumas instruções não podem ser executadas e portanto serão ignoradas se forem encontradas, como as instruções que habilitam e inibem interrupção, as instruções que seecionam o modo de operação e a que indica uma instrução adaptativa.
- o retorno do código adaptativo acontece quando é feito um desvio indireto ao endereço 8006. Após o desvio, e antes de executar a próxima instrução, é desligado o estado de adaptatividade, e restaurado o estado de habilitação da interrupção.

No estado atual estamos implementando o código nas APPs para iPhone e iPad e procurando uma aplicação para testar o conceito. Uma provável aplicação seria a implementação do problema do coletor de nomes[5], uma aplicação adaptativa já bem estudada.

Para o futuro, há a perspectiva de gerar um modelo mais real, com a construção de um painel com leds e chaves controlados por uma placa raspberry pi, ou equivalente, ao qual poderiam ser conectados periféricos reais, como impressoras ou terminais de vídeo.

III. CONCLUSÃO

A emulação de arquiteturas por meio de APPs fornece um meio rápido e barato de realizar o teste de arquiteturas adaptativas ou experimentais.

A utilização de uma arquitetura conhecida como ponto de partida permite que se utilize ferramentas já existentes no auxílio ao desenvolvimento de software.

No desenvolvimento procuramos manter compatibilidade com o software já existente, e estender a arquitetura de modo a ter uma ampla gama de possibilidades de criação de novas instruções adaptativas.

Para testar o conceito, estão sendo modificadas APPs de iPhone e iPad, para incluir a adaptatividade.

Para o futuro pretende-se também implementar o conceito num modelo mais real, com um painel de leds e uma placa processadora onde poderão ser conectados periféricos reais.

REFERÊNCIAS

[1] JUNIOR, G.L.; FREGNI, E. “Projeto de Computadores Digitais” Editora Edgard Blucher, Editora da Universidade de São Paulo , cap 5, 1974

[2] NETO J.J “Aspectos do Projeto de Software de um Minicomputador” Dissertação de mestrado na Escola Politécnica da USP, 1975
.

[3] GRASSELLI, A. “The Design of Program-Modifiable Microprogrammed Control Units” IRE Transactions on Electronic Computers, p 336-339, junho de 1962

[4] NETO, J.J. “Proposal of a High-Level Language for Writing Self Modifying Programs” IEEE Latin America Transactions, p 192-198 abril 2011

[5] SIQUEIRA F.L. “AdapLib: Uma Biblioteca para Execução de Dispositivos Adaptativos” 3º Workshop de Tecnologia Adaptativa, 2009

Sebastião Santiago Barretto Graduado em Engenharia Eletrônica, modalidade Sistemas Digitais pela Escola Politécnica da Universidade de São Paulo (EPUSP) em 1976, onde também estagiou no Laboratório de Sistemas Digitais (LSD) de Dezembro 1972 a Dezembro de 1976. Trabalhou Na Bireme, Scopus Tecnologia e Diebold Procomp. Atualmente desenvolve APPs para iPhone e iPad na Ciência e Arte Informática. Tem também desenvolvido projetos utilizando módulos Arduino.

João José Neto Graduado em Engenharia de Eletricidade (1971), mestre em Engenharia Elétrica (1975), doutor em Engenharia Elétrica (1980) e livre-docente (1993) pela Escola Politécnica da Universidade de São Paulo, SP, Brasil. Atualmente é professor associado da Escola Politécnica da Universidade de São Paulo e coordena o LTA – Laboratório de Linguagens e Tecnologia Adaptativa do PCS – Departameto de Engenharia de Computaçao e Sistemas Digitais da EPUSP. Tem experiência na área de Ciência da Computação, com ênfase nos Fundamentos da Engenharia da Computação, atuando principalmente nos seguinte temas: dispositivos adaptativos, tecnologia adaptativa, autômatos adaptativos, e em suas aplicações à Engenharia de Computação, particularmente em sistemas de tomada de decisão adaptativa, análise e processamento de linguagens naturais, construção de compiladores, robótica, ensino assistido por computador, modelagem de sistemas inteligentes, processos de aprendizagem automática e inferências baseadas em tecnologia adaptativa.