Dynalab

Francisco M. C. Oliveira
Universidade do Minho, Portugal

***Resumo*—Este texto descreve brevemente a motivação, arquitectura e principais funcionalidades desenvolvidas no projecto Dynalab. Todos os recursos deste projecto podem ser encontrados em http://www.cs.montana.edu/ dynalab.**

I. Introdução

A ideia inicial na implementação deste projecto pretendia essencialmente desenvolver uma ferramenta para o estudo e ensino de programação de computadores. De entre as funcionalidades necessárias o software deveria:

- incluir uma colecção de exemplos devidamente construídos;
- executar os programas em dois modos: avanço/recuo;
- evidenciar a comando em execução na linguagem de alto nível;
- mostrar claramente os valores de variáveis e parâmetros;
- delinear claramente as variáveis e parâmetros locais de um função.

II. E-machine

O sistema proposto tem como componente fundamental o E-machine (Education Machine). O E-machine funciona como uma máquina virtual com a sua própria linguagem máquina, o e-code, e é responsável pela execução das instruções e-code resultantes da tradução de linguagens de mais alto nível (ex.: Pascal, C, Ada. etc).

Uma das funcionalidades mais importantes é a execução em modo reverso. Numa máquina virtual ou real, o program counter, registos, memória principal e outras informações de estado, podem ser vistas como variáveis que se alteram à medida que a máquina executa as instruções. O valor destas variáveis podem também ser vistas como o "estado"da máquina. Se sabemos o estado actual, podemos determinar o estado seguinte. Não é vulgar contudo, o estado actual conter informação suficiente para obter um estado anterior, ou seja, não é guardada a história da execução. Mas é precisamente com a história da execução que podemos implementar a execução reversa.

Para implementar a execução reversa duma forma eficiente, o e-Machine tem que reter a mínima informação necessária para recuar para o estado imediatamente anterior.

Podemos destacar alguns elementos comuns a estas arquitecturas nomeadamente, um espaço de memória, um program counter, vários apontadores da stack e vários tipos de registos. Um outro elemento, não comum noutras arquitecturas, é o previous program counter onde fica registado o endereço da instrução executada previamente.

Com todas estas considerações, foi escolhida uma arquitectura baseada em stack de entre outras possíveis, devido essencialmente à sua simplicidade.

III. Outros Componentes

A. E-Machine Emulator

Foi desenvolvido um emulador para E-machine em ANSI C. No ambiente Dynalab, o emulador funciona como um escravo para o program animator, e executa pacotes de instruções e-code após cada chamada.

B. Program Animator

Este componente é responsável pela visualização da dinâmica dos programas compilados com um dos compiladores do Dynalab. Existem já desenvolvidos animadores para MS-Windows e OSF/Motif. Os program animators permitem a execução dos programas linha-a-linha ou continua, em modo avanço ou reverso, e permite a visualização dos valores de variáveis e parâmetros.

C. Compiladores

O Instrunction Set do E-machine é relativamente simples e completo e permite um acesso fácil aos dados. Estas características permitiram um rápido desenvolvimento de compiladores para C, Pascal e Ada/CS. Apesar de os compiladores estarem incompletos, permitem a execução de grande parte das instruções da linguagem. Cabe aos compiladores de E-machine a implementação de mecanismos que permitam a execução de uma linha de código de alto nível de cada vez e a execução reversa.

IV. Exemplo

Programa em C:

```
int foo(C)
int *C
{
*C = 1;
}
```

Código para E-machine:

```
label c,5
inst c,V3
pop c,A,V3
push I,C1
pop c,I,(V3)
uninst c,V3
return
```

Referências

[1] Patton, Samuel Dellette *The E-Machine: Supporting the Teaching of Program Execution Dynamics.* Master's thesis - Department of Computer Science, Montana State University - Bozeman, June 1989.

[2] Birch, Michael Leigh *An Emulator for the E-Machine.* Master's thesis - Department of Computer Science, Montana State University - Bozeman, June 1990.

[3] Goosey, Frances Wren *A MiniPascal Compiler for the E-Machine.* Master's thesis - Department of Computer Science, Montana State University - Bozeman, June 1993.
[4] Eneboe, Torlief James, *An ANSI C Compiler for the E-Machine.* Master's thesis - Department of Computer Science, Montana State University - Bozeman, June 1995.