RESUMO

São tocados ali uns aspectos de um sistema de desenvolvimento cruzado, constituído por um simulador interpretador, para apoio de projectos que envolvam micro-computadores.

1 - Apresentam-se, nesta comunicação, alguns aspectos de um projecto em desenvolvimento no Centro de Informática do Laboratório Nacional de Engenharia Civil, relacionado com a construção de um simulador interactivo de sistemas que recorram a microprocessadores.

Este projecto nasceu por um lado do envolvimento de vários grupos de trabalho em projectos com microprocessadores e por outro da possibilidade de acesso a um computador central o DEC-system 10, de grande porte e sistema de operação sofisticado.

Este projecto, agora delineado, está claramente inserido no domínio dos produtos-cruzados ("cross-products"). É este um tipo de solução que cremos se deverá encarar sempre que haja, por parte de um projectista a possibilidade de recurso a uma maquina de dimensões apreciáveis.

2 - Dum certo ponto de vista, o simulador/Interpretador que se pretende construir pode considerar-se como um substituto possível para um sistema de desenvolvimento nativo, i.e., um sistema de desenvolvimento montado em torno da m aquina que surgirá na versão final do produto em projecto

Verifica-se actualmente que tendem a ser os sistemas nativos, cada mais, os únicos oferecidos pelos fabricantes, talvez por razoes de índole comercial. Cremos assim ser do maior interesse a construção de um sistema cruzado, geral e flexível, que permita, mesmo na ausência da maquina real, arrancar um projecto ou testar hipóteses alternativas.

Num estádio extremamente evoluído deveria ser possível declarar a configuração hardware escolhida, escrever os programa para a aplicação do estudo e realizar a sua correcção/afirmação sem nunca recorrer ao microprocessador real.

Para além disso, durante o desenvolvimento> do Projecto estariam disponíveis ferramentas de diagnóstico e controle muito mais poderosas do que se poderá encontrar num sistema nativo, permitindo deste modo uma conclusão rápida e segura do projecto.

Não são ainda todas estas facilidades que se pretendem oferecer no simula dor/interpretador em construção. As linguagens de declaração de configurações necessitem ainda de um importante progresso ate se tornarem ferramentas comuns, apesar de certas contribuições recentes neste domínio, (3, 4).
Um outro aspecto que não tencionamos contemplar, pelo menos numa primeira fase, tem a ver com o problema da simulação de vários processos com tempos independentes, i.e,, situações de verdadeiro tempo real. Será o caso, por exemplo, de aplicações destinadas a recolha de dados de experiências que ocorrem de maneira aleatória e obviamente não sincronizável, Num linguagem mais perto do problema de simulação diremos que mio tentaremos simular, com generalidade, processos conduzidos por interrupções.

3 - Para efeitos de apresentação podemos considerar que o fulcro do sistema de simulação se situara na possibilidade de reproduzir por software, o conjunto processador/memória de uma determinada máquina, e a execução nele realiza da de um certo programa de aplicação.

E interessante fazer notar aqui que ê de facto muito rara a existência de processadores que, do ponto de vista de um utilizador, seja estritamente hardware. Não pensando já em processadores microprogramaveis, um utilizador tem normalmente acesso urna máquina provida já de algumas facilidades software, disponíveis através de um sistema de operação, porventura numa forma embrionária. Significa isto que, mesmo fora do domínio genericamente aceite como de simulação, o utilizador vê normalmente aquilo que já é de facto uma maquina virtual, podendo em muitos casos esquecer corno ela é implementada.

Essa maquina, numa apresentação esquemática, pode dizer-se que é constituí da do seguinte modo:

A memória central é um conjunto de posições endereçáveis directamente onde se armazenarão dados e programas a serem manipulados A execução de um programa é realizada por um interpretador capaz de analisar as instruções máquina que vai buscar a memória e invocar as operações primitivas adequada e os correspondentes operadores. Estas primitivas manipulam dados não só da memória central como também dos registos rápidos. Podem ainda interactuar com o equipamento de entrada/saída. É corrente, em computadores hardware, verificar-se uma correspondência unívoca entra as instruções maquina e as operações primitivas. Trata-se no entanto de um simples pormenor de implementação.
Na simulação de uma maquina podemos assim ver que o problema central será o de implementar por software, o interpretador, porventura, a custa de operações primitivas diferentes, mas de tal modo que o resultado dessas acções seja equivalente. A memória central e os registos serão do ponto de vista actual , simplesmente representados por outros elementos de memória do computador onde decorre a simulação e a que o interpretador irá recorrer.

Vejamos agora, um pouco mais em pormenor qual o esquema de interpretação que se processa na execução de um programa. O ciclo base envolvido nessa execução e sobejamente conhecida, pode apresentar-se do seguinte modo:

O papel do interpretador é pois basicamente o do ter capaz de descodificar urna instrução obtida da memória, reconhecer os "nomes" dos operandos envolvidos, achar o seu valor e invocar a operação que tenha reconhecido como resultado da descodificação. Estas operações, utilizando os valores que o interpretador lhes forneceu, executarão então acções varias que poderão interagir com a memória, registo; etc. Num processo de simulação tratar-se-a normalmente de sub-rotinas escritas em qualquer linguagem suportada pelo "host-computer"

Este aspecto central do interpretador, afinal de avaliação (“evaluation") de funções é extremamente geral e intervém em domínios aparentemente muito longe da simulação por software de processadores. E, nomeadamente, o âmago de certas linguagens interpretadas.

4 - Embora a simulação de um processador/memória de uma maquina se possa considerar um aspecto central de um sistema de desenvolvimento não nativo outros módulos são essenciais para que se atinja alguma funcionalidade. Durante o desenho e afinação de um programa correspondente a dada aplicação, erros e outras dificuldades irão por certo surgir. Surge assim uma necessidade de executar controladamente um programa que se ensaia. As operações mais básicas que esse controle exige são simples. Paragem da execução em determinados pontos escolhidos anteriormente, possibilidade de consulta ou modificação dos valores contidos em posições de memória ou registos. Nos sistemas de desenvolvimento nativos é comum que as referências feitas aos pontos de paragem ou consulta seja feita através dos seus endereços numéricos. Num simulador, com facilidade se permite ao utilizador o recurso a referências simbólicas, exactamente aqueles que ele terá empregue ao escrever o programa fonte.
A observação do conteúdo de posições de memória por outro lado, deverá poder ser apresentada sob várias interpretações: como números em diferentes bases, como caracteres, como Instruções mnemónicas, etc. Será conveniente, além disso o alargamento das facilidades de controle, tais corno, execução instrução a Instrução, paragem cm certos pontos, apenas depois de determinado numero de passagens, etc. Surge afinal um módulo supervisor, normalmente designado por "debugger” i.e, recebendo Indicações do utilizador, controla o módulo processador. Neste supervisor deverão ser incluídas, para além . das primitivas de controle directo, facilidades de diagnóstico. Por exemplo a contabilização do tempo de processador utilizado até certo instante, a detecção das zonas do programa mais vezes invocadas, a tentativa de acesso, por erro, a certas regiões da memória ou registos. Vemos assim que um supervisor, ele próprio um programa nativo do 'host-computer", pode, com certa facilidade. alargar significativamente as possibilidades de observação de certo programa que se desenvolve mantendo informação global sobre a execução desse programa.

5 - Mas seria ainda Interessante poder evoluir noutro sentido. Dado que a detecção de um hipotético erro,

envolvera muitas vezes a rescrita de certas partes do programa fonte seria ideal para alterações pequenas, poder faze-lo reentrando imediatamente a seguir À execução simulada. Partes novas a acrescentar ou que substituem outras anteriores deverão ser dadas, obviamente em forma mnemónica. Surge assim a necessidade de, dentro do simulador/interpretador, dispor dos serviços de um assembler, e de um gestor minimamente sofisticado do espaço de trabalho do "host computer".
Frisemos aqui que se advoga que a rescrita e reconsideração de partes de um programa seja feita sistematicamente durante um trabalho interactivo. Uma facilidade como a descrita pode ser, no entanto, um auxiliar excelente para pequenas modificações e seu teste imediato.

6 - Em muitas aplicações de um mínimo de dimensão existe um periférico que interessa poder reproduzir. Trata se de um terminal com o qual se possa verificar uma interacção lenta Tipicamente poderemos imaginar um terminal operado pelo utilizador da aplicação em desenvolvimento. Será conveniente para simular este periférico, o recurso ao mesmo terminal que e usado para controlar a simulação. Claro que o supervisor devera saber reconhecer quando certa operação de entrada ou saída se refere a aplicação em este ou ao próprio sistema simulador

Poderemos, neste momento, esboçar o esquema do sistema interactivo em desenvolvimento

7 - Interessa agora apresentar em linhas gerais, qual o processo de implementação deste simulador/interpretador. Adoptou-se para base da implementação aquilo que poderíamos designar por "executor de funções". Este sistema, ou máquina, pode simplificadamente ser visto como um sistema interpretativo que cada um nome de uma função e respectivos argumentos a calcula ("evaluate") dando um valor corno resultado A definição da operação a executar esta guardada naquilo que podemos apresentar como um dicionário. Um pequeno sistema tipo LISP ou FORTH corresponderia a esta Imagem que se pretende dar. Para além de aceitar frases de um utilizador dialogante, i e., lista. de funções, argumentos, este sistema pode avaliar funções chamadas dentro de outras. Alargando um pouco mais estes conceitos as funções referidas não serão mais do que um caso particular daquilo que é guardado no dicionário referido. Em geral encontraremos aí objectos a que podem corresponder a várias definições. Uma delas será precisamente do tipo função. Um objecto importante que se pode inserir neste sistema correspondera ao conjunto memória/registos a simular. A este objecto estará então ligada uma definição que a caracteriza como um array onde se registarão as instruções de um programa a ensaiar. Sobre este objecto poderá actuar em função processador, que eventualmente o modifica. A chamada directa desta função pelo utilizador (ie, a "alto nível") levará, pura e simplesmente à execução não controlada do programa em toste. Não será pois comum este modo de utilização Normalmente serão sim invocadas funções correspondentes ao modulo de "debbug" que, por sua vez a nível interno, invocarão a função processador. O sistema arrancara, numa utilização típica, provido já de certas primitivas do "debbug". Nada impede, porem, que com base nessas funções outras mais complexas vão sendo construídas. Também serão providas, à partida, funções base do diagnóstico por exemplo, poderemos imaginar uma função que contabilize um tempo total de execução duma "instrução" ou o número de vezes que ela é executada. Novamente, tais funções serão tipicamente chamadas a nível interno por exemplo pela própria função Processador. As próprias instruções a executar durante o percurso do programa simulado, mais não são do que funções de um vocabulário base que define um determinado microprocessador. De um certo ponto de vista o programa é na sua parte de instruções um sistema de invocação sistemática da própria máquina avaliadora de funções;

Vejamos como nesta visão muito geral se inserem actividades de edição e assemblagem de novas Partes do programa No processo de edição será designado um objecto sobre cujas propriedades se irá fazer alterações. Também as funções que implementarão esta facilidade têm caracter geral. Nomeadamente poder-se-á alterar o conteúdo da propriedade dependente do objecto programa.. Estar-se-á numa situação de edição sobre o programa a ensaiar.

A função de assemblagem recorre também a esta maquina de avaliação de funções. Esta adquire neste caso um caracter de executor de funções semânticas A introdução de novas linhas de programa corresponde, dentro desta visão, à invocação de uma função de assemblagem seguida de uma inserção na propriedade de um outro objecto, o programa propriamente dita.

8 - A procura de uma visão feral como se esboçou, e a implementação de uma máquina que a concretiza permitem construir com facilidade e passo a passo, o simulador/interpretador que se pretende oferecer. Alem disso abre-se ao utilizador a própria facilidade de poder definir funções mais complexas com base noutras anteriormente definidas de modo a segui os diagnosticar as dificuldades que encontre nos desenvolvimentos dos seus Programas, chegar ao sistema de desenvolvimento mais

adoptado no seu caso.