TP1
Software Básico

Pedro Araujo Pires 01/05/2011

Introdução

Neste trabalho foi desenvolvida um simulador de uma máquina básica. Essa máquina possui três registradores (PC, SP e AC) e uma memória. A máquina trabalha somente com dados inteiros, e a menor unidade endereçável é um inteiro.

Implementação

Para simular os componentes da máquina foi utilizada uma *struct*. Os registradores `PC`, `SP` e `AC` são simulados através de uma variável inteira para cada um, e a memória é simulada através de um *array* de inteiros. Uma *struct* é uma estrutura de dados boa para a máquina virtual, pois guarda todos os componentes da máquina juntos na memória, permitindo uma maior eficiência no uso da *cache* pelo processador.

Para simular o carregador de programas, foi feito uma função que lê um arquivo "executável", e o coloca na memória da máquina virtual. A posição da memória a partir da qual o programa é carregado é passado como parâmetro na execução do programa. Inicialmente foi utilizada a função `fscanf` para ler os números presentes no arquivo, mas essa abordagem trouxe uma dificuldade: para que o arquivo fosse lido corretamente, era necessário que cada linha do arquivo lido tivesse somente um número inteiro, fazendo com que o desenvolvimento de programas na linguagem da máquina fosse bem complicado para um humano.

Foi feita então uma mudança na forma de ler o arquivo: ao invés da função `fscanf`, foi utilizada a função `fgets`, que lê uma linha inteira do arquivo, e em seguida a liha é convertida para um número inteiro. Como a conversão de *string* para inteiro considera somente os `n` primeiros caracteres que são números, qualquer caractere diferente de números que o programa tiver depois da instrução é ignorado, permitindo que o programa possua comentários explicativos e facilitando o desenvolvimento de programas de teste.

Para simular o funcionamento da máquina, foi feita uma função que lê a próxima instrução e o operando (caso a instrução possua um), e executa essa instrução. Essa função consiste em um grande *switch*, que verifica qual é a instrução, e executa as ações necessárias. A única dúvida existente nesta parte do projeto foi em relação às instruções lógicas (`AND`, `OR` e `XOR`). O resultado deve ser a operação lógica *bit* a *bit* dos números em binário (no computador real), ou deve ser a operação entre os dois inteiros da máquina virtual? Como na linguagem C a função `XOR` só é implementada *bit* a *bit*, ficou definido que todas as funções lógicas seriam implementadas como sendo a operação *bit* a *bit* da representação dos números em binário.

Execução

Para compilar o programa, deve-se usar o comando `make`, dentro da pasta com o código fonte. Ele irá gerar um executável, que deve ser executado da seguinte forma:

```
./mv 0 999 0 s multiply.exec
```

O primeiro argumento é o valor inicial do `PC`, o segundo argumento é o valor inicial do `SP`, o terceiro argumento é a posição de memória a partir da qual o programa será carregado, o quarto é o modo de execução (simples ou *verbose*), e o último é o nome do arquivo que contém o programa a ser executdo pela máquina.

No modo *verbose*, a cada instrução executada é impresso na tela qual instrução está sendo executada, e os valores dos registradores `PC`, `SP` e `AC`.

Testes

Para testar a máquina foi escrito um programa que calcula o fatorial de um número recursivamente. Como para calcular o fatorial de um número é necessário multiplicar dois números, primeiramente foi feito um procedimento que multiplica dois números.

O fatorial recursivo é um bom exemplo para testes pois quase todo o conjunto de instruções é usado. Durante os testes o programa, o fatorial só foi corretamente calculado até o número 12. A partir do número 13, os resultados estavam errados, devido a problemas de *overflow*. A partir do número 62, o programa começou a acessar posições fora da memória da máquina virtual, e o programa não executava até o final.

Os dois procedimentos (multiplicação e fatorial recursivo) estão em arquivos "executáveis" (`multiply.exec` e `recursive_fat.exec`). Para executá-los, basta executar a máquia da forma descrita acima, passando o nome de um deles como último argumento na linha de comando.