UVV – Universidade Vila Velha
Ciência da Computação

Cristian Costa Mello
Gustavo Freitas de Amorim

MachineM

Vila Velha
2015

MachineM

Projeto de simulação da Máquina de Registradores - uma Máquina Turing-compatível (Universal).

1. Objetivo do Projeto

Realizar a simulação da Máquina de Registradores.

1.1. Como ocorre a simulação?

Por meio de duas tarefas principais:

1. A partir de um arquivo de texto de entrada, escrito em Linguagem M, monta-se, por meio do AssemblerM, o binário compatível com a CPUM, uma componente executora;

2. Após a execução do binário montado para a CPUM, há a geração da função computada em um arquivo de texto.

Obs.: A linguagem M é detalhada no tópico "Sintaxe da linguagem M".

2. Requisitos de compilação

- Sistema Operacional Unix-like ou GNU/Linux mais recente;
- GCC 4.9.1 ou superior;
- GNU Make 4.0 ou superior.

Requisito opcional

- Qt Creator 5.3.0 ou superior.

3. Estrutura do projeto

A seguir, está disposto a organização básica do projeto MachineM.

```
/MachineM               : pasta-pai do projeto
    /AssemblerM         : pasta do montador AssemblerM
    /CPUM               : pasta da CPU conceitual, a CPUM
    /doc                : pasta da documentação do projeto
    /src                : pasta de códigos-fonte
    COPYING3            : arquivo de licença
    MachineM.pro        : árvore do projeto (Qt Creator)
    Makefile            : arquivo Makefile
    README.md           : arquivo LEIA-ME
```

Obs.: como cortesia da casa, é oferecido um layout simples do projeto para o Qt Creator.

4. Procedimento de compilação do MachineM (via Terminal)

Estando-se na pasta pai do projeto ("/MachineM"), realize o comando abaixo:

```
$ make
```

5. Procedimentos de utilização (via Terminal)

Através de um terminal de comando do Linux, observa-se o formato de entrada abaixo:

```
$ ./MachineM <NOME_DO_ARQUIVO_DE_ENTRADA> <NOME_DO_ARQUIVO_DE_SAIDA> <arg1> <arg2> ...
<argn>
```

Descrição dos argumentos:

```
<NOME_DO_ARQUIVO_DE_ENTRADA> : é um arquivo de texto codificado no padrão ANSI ASCII;

<NOME_DO_ARQUIVO_DE_SAIDA>   : é um arquivo de texto contendo a função computada;

<arg1> <arg2> ... <argn>     : os valores numéricos naturais para configurar os valores
                               iniciais dos registradores de entrada declarados no
                               cabeçalho do arquivo de texto de entrada.
```

Se a execução for bem-sucedida, então será gerado o arquivo de texto contendo a função computada, a rigor de análise.

6. Sintaxe da linguagem M

A sintaxe da linguagem é semelhante às linguagens conceituais monolíticas.

6.1. Sintaxe do Cabeçalho

Todo código-fonte deve ter, em seu cabeçalho (primeira linha), a declaração da máquina a ser utilizada.

```
<NOME_DA_MÁQUINA>:<OUTREG_1>,<OUTREG_2>,...,<OUTREG_N><-
<INREG_1>,<INREG_2>,...,<INREG_N>
```

Significado:

```
<NOME_DA_MÁQUINA>                    : é o nome da máquina;
<OUTREG_1>,<OUTREG_2>,...,<OUTREG_N> : é a declaração dos registradores de saída;
<-                                   : seta que determina o significado dos
                                       registradores (será tratada a seguir);
<INREG_1>,<INREG_2>,...,<INREG_N>    : é a declaração dos registradores de entrada. É
                                       esta lista de registradores que receberá os
                                       valores numéricos inteiros de entrada
                                       descritos anteriormente;
```

A seta descrita acima (<-), pode ser invertida, ou seja, escrita como "->". Todavia, a ordem dos registradores também deverá ser alterada, isto é:

```
<NOME_DA_MÁQUINA>:<INREG_1>,<INREG_2>,...,<INREG_N>-
><OUTREG_1>,<OUTREG_2>,...,<OUTREG_N>
```

Em resumo, a seta indica o sentido do fluxo do processamento descrito nas linhas subsequentes. Tal fluxo pode ser dar como:

```
1. Os registradores de entrada recebem os dados de entrada;
2. Os dados armazenados são processados seguindo a lógica descrita nas linhas
   subsequentes ao cabeçalho;
3. Os registradores de saída armazenam os resultados do processamento.
```

É importante ressaltar que o resultado do processamento só será armazenado nos registradores de saída, caso a lógica descrita no código viabilize este transporte. A utilização de todos os registradores declarados fica a cargo do usuário, mas a tentativa de uso de registradores não declarados gera erro de montagem do programa.

6.2. Sintaxe das Instruções

Toda instrução deve possuir um rótulo, um identificador de operação/teste, um registrador a ser utilizado e o(s) endereço(s) da próxima instrução.

Existem duas operações e um teste. São eles:

- **INC** : incrementa um determinado registrador em uma unidade;
- **DEC** : decrementa um determinado registrador em uma unidade;
- **ZERO** : testa se um registrador é igual a 0.

As instruções de operação seguem as seguintes sintaxes:

```
1. <RÓTULO>: faca <OPERACAO> <REGISTRADOR> va_para <RÓTULO_DESTINO>
2. <RÓTULO>: <OPERACAO> <REGISTRADOR> <RÓTULO_DESTINO>
```

Para ambos os casos, seguem os seguintes significados:

```
<RÓTULO>         : Rótulo da instrução;
<OPERACAO>       : Operação a ser realizada (INC ou DEC);
<REGISTRADOR>    : Registrador a ser utilizado na operação;
<RÓTULO_DESTINO> : Rótulo da próxima instrução.
```

A instrução de teste pode seguir uma das possibilidades:

```
1. <RÓTULO>: se zero <REGISTRADOR> entao <RÓTULO_DESTINO_VERDADE> senao
<RÓTULO_DESTINO_FALSO>
2. <RÓTULO>: faca zero <REGISTRADOR> va_para <RÓTULO_DESTINO_VERDADE>
<RÓTULO_DESTINO_FALSO>
3. <RÓTULO>: zero <REGISTRADOR> <RÓTULO_DESTINO_VERDADE> <RÓTULO_DESTINO_FALSO>
```

Para todos os casos, há os seguintes significados:

```
<RÓTULO>                 : Rótulo da instrução;
<REGISTRADOR>            : Registrador a ter seu valor verificado;
<RÓTULO_DESTINO_VERDADE> : Rótulo da próxima instrução caso o teste retornar verdade.
<RÓTULO_DESTINO_FALSO>   : Rótulo da próxima instrução caso o teste retornar falso.
```

A linguagem é case sensitive ("sensível ao caso").

6.2.1. Determinando o fim do programa

O fim do programa pode ser determinado referenciando um rótulo inexistente no rótulo de destino da instrução.

6.3. Exemplo de Código

O código a seguir transfere o valor contido no registrador b para o registrador a, para valores maiores do que 0.

```
M:a<-b
r1: faca inc a va_para r2
r2: faca dec b va_para r3
r3: se zero b entao r5 senao r1
```

Atenta-se ao fato de que existe mais de uma possibilidade de escrita das instruções. O código a seguir é equivalente ao anterior:

```
M:a<-b
r1: inc a r2
r2: dec b r3
r3: zero b r5 r1
```

Uma observação importante, é que o rótulo "r5", referenciado na instrução rotulada "r3", não existe. Logo, se o teste "zero b" resultar em verdade, o programa será finalizado. Como boa prática de programação, recomenda-se seguir um único padrão de escrita, visando uma maior facilidade de entendimento humano do código.

6.4. Comentários

É possível fazer comentários no código, desde que seja atendida a seguinte condição:

- O comentário deve **OBRIGATORIAMENTE** vir depois de uma instrução;

Para fazer um comentário, deve-se iniciar a escrita com ";" ou "#".

Exemplo de código comentado:

```
M:a<-b
r1: faca inc a va_para r2   ; Incrementa o registrador a e vai para r2
r2: faca dec b va_para r3   ; Decrementa o registrador b e vai para r3
r3: se zero b entao r5 senao r1 ; Se b for zero, então finaliza. Caso contrário, volta
para r1
```

7. Arquivo de Saída (Função Computada)

O arquivo de saída é função computada do programa. Cada linha do arquivo conterá a configuração:

```
(<NÚMERO_DA_PRÓXIMA_INSTRUÇÃO>, (<VALOR_REG1>, <VALOR_REG2>, ..., <VALOR_REG_N>))
```

A ordem dos registradores é a mesma ordem da declaração da máquina realizada no código.

8. Exemplo de Execução

Considere o código abaixo:

```
M:a<-b
r1: faca inc a va_para r2
r2: faca dec b va_para r3
r3: se zero b entao r5 senao r1
```

Se executarmos o código acima com a seguinte linha comando:

```
./MachineM codigo.txt funcaoComputada.txt 3
```

O arquivo funcaoComputada.txt conterá:

```
(1, (0, 3))
(2, (1, 3))
(3, (1, 2))
(1, (1, 2))
(2, (2, 2))
(3, (2, 1))
(1, (2, 1))
(2, (3, 1))
(3, (3, 0))
(4, (3, 0))
```

9. Diagramas do Projeto

Nas duas próximas páginas, são mostrados os diagramas do **AssemblerM** e **CPUM**, respectivamente.

10. Licença

O MachineM é amparado pela licença GNU General Public License V3.0.