Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Disciplina Software Básico (DCC008) - 2015/02

Trabalho Prático 3 - Expansor de Macros

1 Descrição Geral

Este trabalho envolve a implementação de um expansor de macros para o montador implementado no Trabalho Prático 2.

2 Informações Importantes

- O trabalho deve ser feito **individualmente**, podendo ser discutido entre os colegas, mas o código fonte não poderá ser compartilhado.
- A data de entrega será especificada através de uma tarefa no Moodle;
- **Política de Atrasos:** A entrega de cada trabalho prático deve ser realizada até a data estipulada na tarefa correspondente do Moodle. As tarefas foram programadas para aceitar submissões atrasadas, porém entregas em atrasos serão penalizadas. A penalização pelo atraso será geométrica com o mesmo conforme a fórmula seguir:

$$Desconto(\%) = \frac{2^{d-1}}{0,32}$$

  A fórmula indica o percentual de desconto para d dias de atraso.

  ATENÇÃO: Note que depois de 6 dias o trabalho é avaliado em 0 pontos. Com base na política acima, é altamente recomendável que se esforcem para entregar o TP dentro do prazo para que não tenhamos problemas futuros.
- O trabalho deverá ser implementado **obrigatoriamente em linguagem C**;
- Deverá ser entregue exclusivamente o código fonte com os arquivos de dados necessários para a execução e um arquivo Makefile que permita a compilação do programa nas máquinas UNIX do departamento;
- Além disso, deverá ser entregue uma pequena documentação contendo todas as decisões de projeto que foram tomadas durante a implementação, sobre aspectos não contemplados na especificação, assim como uma justificativa para tais decisões. O documento não precisa ser necessariamente extenso (entre 3 e 5 páginas);

- A ênfase do trabalho está no funcionamento do sistema e não em aspectos de programação ou interface com o usuário. Assim, não é necessário haver tratamento de erros no programa de entrada;
- Todas as dúvidas referentes ao Trabalho Prático serão esclarecidas por meio do fórum, devidamente nomeado, criado no ambiente **Moodle** da disciplina;
- A entrega do trabalho deverá ser feita via **Moodle**, na tarefa criada especificamente para tal. As instruções de submissão, alguns arquivos de teste e o esqueleto da organizaçao dos arquivos estão presentes no arquivo `tp3_seulogin.tar.gz` disponível para download no Moodle.
- **ATENÇÃO:** trabalhos entregues fora do padrão serão penalizados.

3 Especificação da Máquina

Como os trabalhos práticos da disciplina são dependentes das etapas anteriores, convém relembrar algumas informações sobre a máquina virtual:

- A menor unidade endereçável nessa máquina é um inteiro;
- Os tipos de dados tratados pela máquina também são somente inteiros;
- A máquina possui uma memória de não menos que 1000 posições, 3 registradores de propósito específico e 16 registradores de propósito geral;
- Os registradores de propósito específico são:
  - PC (contador de programas): contém o endereço da próxima instrução a ser executada;
  - SP (apontador da pilha): aponta para o elemento no topo da pilha;
  - PSW (palavra de status do processador): consiste em 2 bits que armazenam o estado da última operação lógico/aritmética realizada na máquina, sendo um dos bits para indicar que a última operação resultou em zero, e outro bit para indicar que a última operação resultou num resultado negativo;
- Os registradores de propósito geral são indexados por um valor que varia de 0 a 15;
- A única forma de endereçamento existente na máquina é direta, relativa ao PC;
- As instruções `READ` e `WRITE` leem e escrevem um inteiro na saída padrão do emulador;
- As instruções são codificadas em um inteiro, podendo ter dois, um ou nenhum operando, que é o caso das instruções `RET` e `HALT`.
- Os operandos podem ser uma posição de memória (M, codificado como inteiro) ou um registrador (R, codificado como um inteiro entre 0 e 15).
- O conjunto de instruções é o mesmo da máquina anterior, conforme a Tabela 1. As operações marcadas com * atualizam o valor do PSW.

Cód	Símbolo	Operandos	Significado	Ação
01	LOAD	R M	Carrega Registrador	$Reg[R] \leftarrow Mem[M + PC]$
02	STORE	R M	Armazena Registrador	$Mem[M + PC] \leftarrow Reg[R]$
03	READ	R	Lê valor para registrador	$Reg[R] \leftarrow$ "valor lido"
04	WRITE	R	Escreve conteúdo do registrador	"Imprime" $Reg[R]$
05	COPY	R1 R2	Copia registrador	$Reg[R1] \leftarrow Reg[R2]$ *
06	NEG	R1	Inverte o sinal do registrador	$Reg[R1] \leftarrow -Reg[R1]$ *
07	SUB	R1 R2	Subtrai dois registradores	$Reg[R1] \leftarrow Reg[R1] - Reg[R2]$ *
08	ADD	R1 R2	Soma dois registradores	$Reg[R1] \leftarrow Reg[R1] + Reg[R2]$ *
09	AND	R1 R2	AND (bit a bit) de dois registradores	$Reg[R1] \leftarrow Reg[R1]$ AND $Reg[R2]$ *
10	OR	R1 R2	OR (bit a bit) de dois registradores	$Reg[R1] \leftarrow Reg[R1]$ OR $Reg[R2]$ *
11	XOR	R1 R2	XOR (bit a bit) de dois registradores	$Reg[R1] \leftarrow Reg[R1]$ XOR $Reg[R2]$ *
12	NOT	R1	NOT (bit a bit) de um registrador	$Reg[R1] \leftarrow$ NOT $Reg[R1]$ *
13	JMP	M	Desvio incondicional	$PC \leftarrow PC + M$
14	JZ	M	Desvia se zero	Se $PSW[zero], PC \leftarrow PC + M$
15	JNZ	M	Desvia se não zero	Se $!PSW[zero], PC \leftarrow PC + M$
16	JN	M	Desvia se negativo	Se $PSW[negativo], PC \leftarrow PC + M$
17	JNN	M	Desvia se não negativo	Se $!PSW[negativo], PC \leftarrow PC + M$
18	PUSH	R	Empilha valor do registrador	$SP \leftarrow SP - 1$ $Mem[SP] \leftarrow Reg[R]$
19	POP	R	Desempilha valor no registrador	$Reg[R] \leftarrow Mem[SP]$ $SP \leftarrow SP + 1$
20	CALL	M	Chamada de subrotina	$SP \leftarrow SP - 1$ $Mem[SP] \leftarrow PC$ $PC \leftarrow PC + M$
21	RET		Retorno de subrotina	$PC \leftarrow Mem[SP]$ $SP \leftarrow SP + 1$
22	HALT		Parada	

Tabela 1: Instruções da Máquina Virtual

Além das instruções acima, existem também as pseudo-instruções `WORD` e `END`:

- `WORD A` → Usada para alocar uma posição de memória cujo valor será `A`, ou seja, quando houver tal instrução, a posição de memória que está sendo referenciada pelo `PC` deverá receber o valor `A`
- `END` → Indica o final do programa para o montador.

4 Descrição do Expansor de Macros

O Expansor de Macros deve ser, basicamente, um programa que receberá um arquivo de entrada, identificará as macros que foram definidas e as substituirá nos locais em que elas foram utilizadas, gerando um arquivo que pode ser utilizado como entrada para o montador implementado no Trabalho Prático 2.

A seguir serão descritos alguns detalhes para a implementação do expansor de macros.

4.1 Pseudo-instruções

Para a implementação do expansor de macros são necessárias duas pseudo-instruções:

- `BEGINMACRO` → Indica o início da definição de uma macro.
- `ENDMACRO` → Indica o fim da definição de uma macro.

4.2 Características das Macros

- As macros podem ser definidas em qualquer ponto do programa. O seu expansor será responsável pela interpretação correta do que é definição de macro, o que é código direto e o que é chamada de macro;
- As macros podem ter **um parâmetro** ou não ter parâmetro;
- **Não ocorre** definição de uma macro dentro de outra macro;
- O nome da macro será definido antes da pseudo-instrução BEGINMACRO seguido por dois pontos (e.g. $< nome\_macro >$: BEGINMACRO).

4.2.1 Macro sem Parâmetro

Exemplo:

```
MACRO1: BEGINMACRO
LOAD A
ADD B
STORE A
ENDMACRO
MACRO1
MACRO1
```

Ao ser processado pelo expansor de macros, deverá ser gerado o código:

```
LOAD A
ADD B
STORE A
LOAD A
ADD B
STORE A
```

4.2.2 Macro com Parâmetro

Exemplo:

```
MACRO2: BEGINMACRO Y
LOAD Y
ADD B
STORE Y
ENDMACRO
MACRO2 A
MACRO2 J
```

Ao ser processado pelo expansor de macros, deverá ser gerado o código:

```
LOAD A
ADD B
STORE A
LOAD J
ADD B
STORE J
```

5 Descrição da Tarefa

Os alunos deverão implementar o expansor de macros definido acima. Além disso, deverão ser criados dois programas de testes, **obrigatoriamente** contendo macros (definidas pelo aluno), a saber:

- **Fatorial:** Programa que lê um número e imprime seu fatorial. Exemplo:

$$4! = 24$$

- **Máximo divisor comum (MDC):** Programa que lê dois números e imprime o MDC destes:

$$mdc(24, 16) = 8$$

As implementações dos programas de teste serão avaliadas, portanto tais programas não devem ser compartilhados entre os colegas. Por outro lado, outros programas de teste adicionais podem ser compartilhados. Lembre-se de colocar todos os testes (inclusive os obrigatórios) no diretório "`tst/`" e de citá-los em sua documentação.

6 Formato da Entrada de Dados

A entrada do expansor de macros deve seguir o mesmo formato especificado para o montador implementado anteriormente. A única diferença é a presença da definição e utilização de macros. Para um teste inicial do seu expansor de macros, utilize o programa abaixo (o mesmo salvo em "`tst/teste1m.amv`").

```
READ R0
READ R1
STORE R0 A
SUB R0 R1
LOAD R0 A
JN MB
COPY R2 R0
JMP
MB: COPY R2 R1
L: PRINT
HALT
A: WORD 0
PRINT: BEGINMACRO
WRITE R0
WRITE R1
WRITE R2
ENDMACRO
END
```

Atenção: Tal programa não testa todas as instruções e não deve ser utilizado como único teste do seu expansor.

7 Formato da Saída de Dados

Corresponde ao formato do arquivo de entrada do Trabalho Prático 2, já que o montador será utilizado para passar o código de saída do expansor de macros para o executável que a máquina virtual reconhece.

8 Formato de Chamada do Expansor

O expansor de macros receberá apenas 2 argumentos quando da sua chamada:

- Nome do arquivo de entrada: contendo o programa a ter macros expandidas – informado como **primeiro argumento** na chamada da aplicação.
- Nome do arquivo de saída: informado como **segundo argumento** na chamada da aplicação.

Exemplo:

```
./expansor teste1m.amv teste1.amv  - expande as macros contidas no programa
                                     do arquivo teste1m.amv para o formato
                                     aceito pelo montador e grava no arquivo
                                     teste1.amv
```

A saída do expansor, depois de montada pelo montador, deve ser executada no emulador da mv para garantir que o programa foi expandido e montado corretamente.

9 Sobre a Documentação

- Deve conter as decisões de projeto.
- Deve conter as informações de como executar o programa. Obs.: é necessário cumprir os formatos definidos acima para a execução, mas tais informações devem estar presentes também na documentação.
- Não incluir o código fonte no arquivo de documentação.
- Deve conter elementos que comprovem que o programa foi testado (e.g. imagens da telas de execução). Os arquivos relativos a testes devem ser enviados no pacote do trabalho, conforme descrito na Seção 2. A documentação deve conter referências a esses arquivos, explicação do que eles fazem e dos resultados obtidos.

10 Considerações Finais

É obrigatório o cumprimento fiel de todas as especificações de interface descritas neste documento. As decisões de projeto devem fazer parte apenas da estrutura interna do montador, não podendo afetar a interface de entrada e saída.