Universidade do Estado de Santa Catarina - UDESC
Bacharelado em Ciência da Computação - BCC
Linguagem de Programação - LPG0001 - 02/2016

Trabalho sobre Estruturas e Alocação Dinâmica de Memória

Monitor: Mateus Boiani

Orientador: Guilherme Koslovski

Entrega da versão final: 10/11/2016 - 23:55 - No moodle.

1 Processador simplificado para manipulação de inteiros

Um processador é composto por um conjunto básico de instruções, utilizadas para acessar e manipular os dados temporariamente armazenados na memória de um computador. Antes de operar os valores, a informação deve ser posicionada em unidades internas de memória (os registradores).

No presente trabalho, simularemos a execução das instruções *load, store, add, sub, div, rem* e *mul* em um processador com 2 registradores, identificados por *reg1* e *reg2*. O processador proposto é limitado, ou seja, somente manipula inteiros.

O conjunto de instruções compreendidas pelo processador é:

- **load**: carrega uma informação da memória para um registrador. Protótipo de utilização: *load <reg_destino> <mem_origem>*
- **store**: armazena na memória uma informação previamente manipulada em um registrador. Protótipo de utilização: *store <var_destino> <reg_origem>*
- **add**: soma o conteúdo de duas variáveis, armazenando o resultado no primeiro registrador. Protótipo de utilização: *add <reg_b> <reg_a>*
- **sub**: subtração do conteúdo de um registrador pelo conteúdo informado em outro. O resultado é armazenado no primeiro registrador. Protótipo de utilização: *sub <reg_b> <reg_a>*
- **div**: divide o conteúdo de um registrador pelo denominador informado em um segundo. O resultado é armazenado no primeiro registrador. Protótipo de utilização: *div <reg_b> <reg_a>*
- **mul**: efetua uma multiplicação entre os valores dos registradores.O resultado é armazenado no primeiro registrador. Protótipo de utilização: *mul <reg_b> <reg_a>*

2 Memória: organização e manipulação

Uma biblioteca para manipulação das variáveis estará disponível para manipulação. A biblioteca abstrai a execução das chamadas *malloc, free* e *realloc*. Ou seja, a alocação deve, obrigatoriamente, ser realizada pela biblioteca fornecida.

De forma complementar as instruções do processador, duas operações são utilizadas para declaração e liberação da memória.

- **int**: informa que uma variável inteira deve ser alocada e inicializada com o valor informado. Protótipo de utilização: *int <var> <valor>*
- **rem**: informa que uma variável não é mais necessária, ou seja, pode ser desalocada. O valor 0 é arbitrariamente explicitado na instrução. Protótipo de utilização: *rem <var> <0>*

O tamanho máximo para identificadores de variáveis é 100.

3 Definições para implementação

Definições necessárias para implementar o trabalho:

```
1  #define TAMANHO_ID 100
2
3  char *CPUInt_instrucoes[] = {
4      "load", "store", "add",
5      "mul", "div", "sub",
6      "rem"
7  };
8
9  typedef struct {
10     int valor;
11     char identificador[TAMANHO_ID];
12 } Variavel;
13
14 typedef struct{
15     int reg1;
16     int reg2;
17     char *instrucao;
18 } CPUInt;
19
20 typedef struct {
21     int elementos;
22     Variavel *variaveis;
23 } RAMInt;
```

Protótipos de funções que devem ser implementadas (outras funções podem ser definidas durante a implementação):

```
1  CPUInt *inicializaCPU(void);
2  RAMInt *inicializaRAM(void);
3  void destruirCPU(CPUInt *);
4  void destruirRAM(RAMInt *);
5
6  void declararRAMInt(
7      RAMInt *ram,
8      int valor,
9      char *id
10 );
11
12 void removerRAMInt(
13     RAMInt *ram,
14     int valor,
15     char *id
16 );
17
18 void executarStoreCPUInt(
19     CPUInt *cpu,
20     RAMInt *ram,
21     char *dest,
22     char *orig
23 );
24
25 void executarLoadCPUInt(
26     CPUInt *cpu,
27     RAMInt *ram,
28     char *dest,
29     char *orig
30 );
31
32 /* ---------------------------------------- *
33  * A definicao do prototipo das demais     *
34  * funcionalidades e possiveis ponteiros   *
35  * para funcao eh parte do trabalho.       *
36  * ---------------------------------------- */
```

4 Implementação do trabalho

O trabalho poderá ser implementado em duplas (2 participantes), no máximo. O prazo para desenvolvimento foi calculado considerando eventuais dúvidas e visitas ao monitor da disciplina. Para o presente trabalho, questões por e-mail não serão resolvidas, somente dúvidas presenciais.

5 Como o trabalho será avaliado?

Para avaliação do trabalho, os seguintes itens serão considerados, com pesos distintos:

- Avisos e compilação correta;
- Independência de arquitetura, sistema operacional, bibliotecas e ambiente de execução;
- Funcionamento correto dos casos de teste;
- Organização e prototipagem de funções;
- Decomposição do código em arquivos fonte e de cabeçalho;
- Alocação dinâmica de memória (e liberação);
- O uso da biblioteca para manipulação de memória fornecida é obrigatório. Não use diretamente *malloc*, *free* e *realloc*. A biblioteca não pode ser alterada.
- Utilização de *struct, union, typedef* e ponteiros para funções;
- Outros itens observados durante a correção.