Arquitetura de Computadores — 2013/14

Licenciatura em Engenharia Informática

Trabalho de Casa 1 (v1.0)

Data de entrega: ***21 de Março de 2012 às 17:00***

Este trabalho de casa consiste num exercício de programação. A resolução deste exercício é individual.

Pode discutir ideias, mas não código com colegas: o desenvolvimento e a escrita do código deve ser estritamente individual. As resoluções serão comparadas de forma automática e os casos de plágio serão punidos de acordo com os regulamentos em vigor. Copiar código da Internet também é considerado plágio e o seu programa será alvo de uma verificação automática para esta situação.

Em futuras aulas praticas haverá uma avaliação individual que consistirá numa extensão à sua resolução deste trabalho de casa. Quem demonstrar desconhecimento do código que entregou terá a sua nota no trabalho de casa alterada para zero valores.

As instruções para a entrega do trabalho serão divulgadas no CLIP. Note que os formatos dos outputs dos exercícios devem ser estritamente obedecidos.

Objectivo

Com este primeiro trabalho de casa de AC, pretende-se que implemente em C uma aplicação que simula uma calculadora com 10 memórias, identificadas pelos dígitos 0 a 9, e todas inicializadas com o valor 0. Cada unidade de memória guarda um número com sinal de 32 bits. A calculadora suporta as operações listadas em baixo. Note que todas as operações têm o mesmo formato: — *instrução argumento1 argumento2* —.

Instrução	Funcionamento
store m1 v	Guarda o valor *v* na memória *m1 (m1 = 0, 1, ..., 9).*
print m1	Imprime no ecrã o valor (inteiro de 32 bits com sinal) guardado em *m1*. **Note que esta instrução apenas tem um argumento e não dois como todas os restantes.**
copy m1 m2	Copia o valor da memória *m2* para a memória *m1*.
add m1 m2	Adiciona o conteúdo da memória *m2* à memória *m1* (i.e., realiza a operação *m1 = m1 + m2*) . A memória *m2* não é alterada por esta operação.
sub m1 m2	Subtrai o conteúdo da memória *m2* à memória *m1* (i.e., realiza a operação *m1 = m1 − m2*) . A memória *m2* não é alterada por esta operação.
mul m1 m2	Multiplica o conteúdo da memória *m2* pela memória *m1* (i.e., realiza a operação *m1 = m1 × m2*) . A memória *m2* não é alterada por esta operação.
div m1 m2	Divide o conteúdo da memória *m2* pela memória *m1* (i.e., realiza a operação *m1 = m1 ÷ m2*). Se *m2* tiver o valor zero, *m1* deverá ficar também com o valor 0. A memória *m2* não é alterada por esta operação.

mod m1 m2	Coloca em *m1* o resto da divisão inteira de *m1* por *m2* (i.e., realiza a operação *m1 = m1 % m2*). Se *m2* tiver o valor zero, *m1* deverá ficar também com o valor 0. A memória *m2* não é alterada por esta operação.
fact m1 m2	Calcula o factorial de *m2* utilizando uma função recursiva e coloca o resultado em m1. Se o valor em m2 for negativo, m1 deverá fica com o valor 0. A memória *m2* não é alterada por esta operação.

Em todas as operações descritas acima, "m1" e "m2" deverão ser substituídos por um inteiro entre 0 e 9, que representa a respetiva unidade de memória. Na operação store, "v" é também um inteiro.

Notas importantes e sugestões

1. Os identificadores da memórias são números inteiros entre 0 e 9.
2. Cada memória guarda um número inteiro com sinal de 32 bits.
3. Pode assumir que o input está bem formatado, i.e., assumir que a linha de comandos contém sempre uma instrução válida e os seus argumentos (um ou dois) inteiros.
4. Uma implementação correta da operação factorial mas que seja iterativa (e não recursiva) terá uma cotação parcial.
5. Itere os argumentos da linha de comando entre *argv[1]* e *argv[argc–1]*.
6. Utilize a função *atoi* para converter os argumentos em números inteiros (quando aplicável). Para obter mais informações sobre esta função, introduza o seguinte comando no terminal "*man atoi*".
7. Utilize a função *strcasecmp* para comparar duas strings ignorando as maiúsculas/minúsculas, e.g., *if (strcasecmp (argv[1], "store") == 0) {...}*. Para obter mais informações sobre esta função, introduza o seguinte comando no terminal "*man strcasecmp*".
8. No seu programa provavelmente irá precisar **pelo menos** dos seguintes *includes*:
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```
9. Para saber que includes são necessários para poder usar uma função da biblioteca do C, veja a respetiva página de manual executando no terminal o comando "*man FUNÇÃO*" e verificando os includes necessários na 3ª secção ("SYNOPSIS").
10. Para testar o seu programa (que deverá ter o nome de "calculadora") deverá escrever a sequência de comandos na respetiva linha de comando.

Trabalho a realizar

Desenvolva um programa em (oito) fases, em que cada uma das fases implemente as operações abaixo descritas:

1. store e print
2. copy
3. add
4. sub
5. mul
6. div
7. mod
8. fact

1. Recomenda-se fortemente que implemente as oito fases de forma sequencial e valide (teste bem) cada uma delas antes de passar à fase seguinte, caso contrário corre o risco trabalhar muito e acabar por ter zero valores no trabalho de casa.

O seu programa deverá chamar-se "calculadora" e aceitar os comandos como argumentos da linha de comando.

Exemplos de utilização:

Utilização simples:

```
./calculadora store 0 20 store 1 5 print 0 print 1 add 1 0 print 1
```

e o output deverá ser

```
20
5
25
```

Utilização com base num ficheiro de instruções:

Escreva as instruções num ficheiro de texto, por exemplo com o nome *teste.txt*, com uma instrução por linha

```
store 0 20
store 1 5
print 0 0
print 1 0
add 1 0
print 1 0
```

e no terminal execute o seguinte comando

```
cat teste.txt | xargs ./calculadora
```

O output deverá ser idêntico ao do exemplo anterior.

Em caso de dúvida na interpretação deste enunciado deve preferencialmente falar com o seu docente das aulas práticas.