Atividade de Laboratório 8

Objetivos

O objetivo desta atividade é exercitar o conceito de funções e ABI na arquitetura ARM.

Descrição

Robinson Late é um aluno da Unicamp que sempre chega atrasado na aula de MC404. Preocupado com a perda de conteúdo importante exposto em sala da aula, ele decidiu construir um aplicativo que calculasse um caminho entre sua residência, a república Broken Clock, e o Ciclo Básico. Após horas de planejamento, ele considerou que a melhor alternativa seria construir a interface do aplicativo em C e o algoritmo de busca de caminho utilizando a linguagem de montagem do ARM. Entretanto, como chegou atrasado a todas as aulas de ARM, ele conseguiu escrever apenas a parte em C do aplicativo e pediu a você, seu dedicado e pontual colega, que implementasse o restante.

Você deve implementar o **algoritmo de DFS recursivo** (https://courses.cs.washington.edu/courses/cse326/03su/homework/hw3/dfs.html (https://courses.cs.washington.edu/courses/cse326/03su/homework/hw3/dfs.html)) para encontrar um caminho entre a república e o Ciclo Básico. Para isso, você receberá, já implementado em C, a função `_start()` que carrega o mapa da cidade; uma função `int daParaPassar(int x, int y)` que retorna '0' caso não seja possível passar na posição (X, Y) e '1' caso seja possível; duas funções que retornam a posição atual do Robinson (`int posicaoXRobinson()` e `int posicaoYRobinson()`) e duas funções que retornam a posição onde ele quer chegar (`int posicaoXLocal()` e `int posicaoYLocal()`). No final, você deve imprimir o resultado como as posições (X, Y) do caminho em cada linha. A primeira tupla deve ser a posição do local e a última a posição do Robinson. É possível que não exista caminho, nesse caso deve ser impresso: "Não existe um caminho!". O tamanho máximo de um mapa será de 10 posições, ou seja, de 0 a 9.

Você deverá implementar a sua solução em linguagem de montagem do ARM e ligar com o código C fornecido. A sua solução deverá conter uma função chamada: `void ajudaORobinson()` que será invocada pela função `main`.

Obs.:

1. as movimentações do caminho podem ser na horizontal, vertical ou na diagonal com no máximo uma casa de distância.
2. você pode pintar o mapa para saber o que já foi visitado, veja as funções `int foiVisitado(int, int)` e `void visitaCelula(int, int)` (reveja o funcionamento do DFS).
3. reveja no lab 06 como imprimir textos e como converter números em texto em ARM (você pode copiar e adaptar o código).

Exemplo

Entrada:

```
0 1 2 3 4 5 6 7 8 9
X X X X X X X X X 0
X _ _ _ X X _ X _ X 1
X _ X _ _ X _ _ _ X 2
X R X X _ _ _ X _ X 3
X _ X _ _ X _ _ _ X 4
X _ _ _ X X _ X _ X 5
X _ X _ _ X _ X _ X 6
X _ _ X _ _ _ X _ X 7
X _ _ L _ X _ _ _ X 8
X X X X X X X X X X
```

Uma das possíveis saídas:

```
3 8
2 7
1 6
1 5
1 4
1 3
```

Compilação

Serão disponibilizados dois códigos: `dfs.h` e `mapa.c`. O primeiro é uma API (*Application Programming Interface*) que possui as definições de rotinas de suporte que serão implementadas em linguagem de montagem. Neste caso há apenas a rotina `ajudaORobinson()` que será chamada pela função `_start()` do `mapa.c`. Para implementar esta rotina, é preciso criar um arquivo `dfs.s` e implementar a rotina descrita no arquivo `dfs.h`.

Como visto em sala de aula, cada rotina deve ser identificada por um rótulo igual ao nome da rotina em "C". Além disso, para permitir que uma rotina implementada em um arquivo seja chamada de outro arquivo você deve sinalizar para o ligador (*linker*) que o rótulo que representa a rotina é um símbolo global. Para fazer isso, basta informar ao montador que o rótulo da rotina é um símbolo global através da diretiva `.global`. O arquivo `example.s` (./example.s) apresenta um exemplo com duas rotinas onde a rotina `dummy_routine1` é definida como global através do uso da diretiva `.global`. Observe que a outra rotina (`dummy_routine2`) não é declarada como global.

O segundo arquivo disponibilizado é o código C onde o programa começa a ser executado.

Arquivos disponibilizados

- dfs.h (./dfs.h)
- mapa.c (./mapa.c)

Configuração de variáveis de ambiente:

```
source /home/specg12-1/mc404/simulador/set_path.sh
```

Geração do arquivo .s referente ao mapa.c:

```
arm-eabi-gcc mapa.c -S -o mapa.s
```

Geração dos arquivos-objeto:

```
arm-eabi-as mapa.s -o mapa.o
arm-eabi-as dfs.s -o dfs.o
```

Ligação:

```
arm-eabi-ld mapa.o dfs.o -o program -Ttext=0x77802000 -Tdata=0x77803000
```

Geração da imagem do cartão SD:

```
mksd.sh --so /home/specg12-1/mc404/simulador/dummyos.elf --user program
```

Importante:

As ferramentas usadas na compilação atual supõem que a função `_start` inicia no endereço `0x77802000` e salta para este endereço após iniciar o sistema. Para posicionar a função `_start` neste endereço, nós utilizamos a *flag*

`-Ttext=0x77802000` para sinalizar ao ligador que todas as seções `.text` sejam posicionadas a partir do endereço `0x77802000`. Para que a função `_start` seja posicionada nesse endereço, ela deve ser a primeira função no arquivo `mapa.c` e o arquivo `mapa.o` deve ser o primeiro na lista de arquivos passados para o ligador. Você pode utilizar o desmontador (`objdump`) para inspecionar o programa programa e verificar se a função `_start` foi de fato associada ao endereço `0x77802000`.

Simulação:

```
arm-sim --rom=/home/specg12-1/mc404/simulador/dumboot.bin --sd=disk.img
```

Entrega e avaliação

Você deve submeter APENAS um arquivo no SuSy, chamado `raXXXXXX.s`, em que XXXXXX é seu ra com 6 dígitos.

Endereço da atividade no sistema SuSy:

O arquivo referente à atividade deve ser submetido para avaliação utilizando-se o sistema Susy em: https://susy.ic.unicamp.br:9999/mc404abef/08ab (https://susy.ic.unicamp.br:9999/mc404abef/08ab) ou https://susy.ic.unicamp.br:9999/mc404abef/08ef (https://susy.ic.unicamp.br:9999/mc404abef/08ef).