Ministério da Educação
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
Campus Curitiba
Departamento Acadêmico de Eletrônica
ELF52 - Sistemas Microcontrolados

LAB 0 - IDE e Assembly

Objetivo:

Utilizando instruções Assembly para Cortex-M4 e o simulador do Keil uVision, encontrar os números primos em uma lista de números previamente fornecida e ordená-la utilizando o algoritmo de Bubble Sort: https://www.embarcados.com.br/algoritmos-de-ordenacao-bubble-sort/

Tarefas:

- Carregar uma lista de números fornecida começando na posição da memória RAM 0x20000D00;
- Verificar quais destes números são primos e copiar apenas estes números a partir da posição 0x20000E00;
- Ordenar esta nova lista do menor para o maior pelo algoritmo *bubble sort*.

Mostrar para o professor e depois entregar a pasta do projeto Keil com todos os arquivos zipada, junto com a imagem fluxograma (pdf, jpg ou png) da ideia proposta também dentro da pasta (preferencialmente em algum site ou aplicativo, e.g. http://draw.io). Nomear o arquivo com o nome e o último sobrenome dos dois alunos da dupla. Ex.: fulanodetal1_fulanodetal2_ap1.zip. Apenas um membro da dupla precisa enviar.

Roteiro:

Dada a seguinte lista de números aleatórios {193; 10; 74; 127; 43; 14; 211; 3; 203; 5; 21; 7; 206; 233; 157; 237; 241; 105; 252; 19}, encontrar quais números são primos e fornecer uma lista com os números primos ordenados ao final utilizando o algoritmo de *bubble sort*.

1) Declarar antes do label start um “EQU” para definir a posição base da memória RAM para a lista de números aleatórios e uma posição base da memória para a lista a ser ordenada (nome EQU 0x20000D00 e nome EQU 0x20000E00).

2) Carregar a lista de números aleatória para a memória RAM a partir da posição 0x20000D00 utilizando o STRB (escrita byte a byte) e endereçamento indexado (pós-indexado);
3) Fazer uma varredura da lista de números aleatórios para encontrar quais números são primos e quando o número for primo escrevê-lo na memória RAM a partir da posição 0x20000E00, formando ainda uma lista desordenada, guardando em um registrador o tamanho da lista sendo formada. (Veja abaixo dicas de como encontrar um número primo).
4) Depois que a lista for formada, ordená-la utilizando o algoritmo *bubble sort* (https://www.embarcados.com.br/algoritmos-de-ordenacao-bubble-sort/)
5) Ao final do código a lista de números deve estar ordenada a partir da posição 0x20000E00.

Dicas e Orientações:

1) Antes do label start um "EQU" para definir a posição base da memória RAM para a lista de números aleatórios e uma posição base da memória para a lista a ser ordenada:
   nome EQU 0x20000D00
   nome EQU 0x20000E00

2) Utilizar o comando de gravar na memória RAM em bytes (**STRB**) e ler da memória RAM em bytes (**LDRB**);
3) Utilizar um registrador para armazenar a posição da memória RAM atual que está sendo varrido número da lista.
4) Utilizar outro registrador para armazenar a posição atual da memória RAM que irá escrever o número primo. Ao gravar na memória RAM, utilizar o modo pós-fixado, que após gravar incrementa o registrador;
5) Utilizar ainda um outro registrador para guardar o tamanho do vetor para saber o tamanho da lista a ser ordenada;
6) Para saber se um número é primo:

a) Utilizar um registrador para iterar de 2 até o número atual para verificar se o número tem algum divisor intermediário, fazendo-o número não primo;

b) Não existe operação de resto de divisão em Assembly Cortex-M4. Neste caso, deve-se utilizar duas operações UDIV e MLS. Com UDIV calcula-se o divisor de um número. Sabendo-se o divisor, realiza-se a operação MLS (MLS Rd, Rm, Rs, Rn --> Rd = Rn - Rm*Rs)

c) Caso o número atual seja primo, coloque-o imediatamente na memória RAM; Caso contrário não o coloque na memória RAM;

7) Após colocar a lista de números primos na memória RAM a partir do endereço 0x20000E00, ler posições da memória RAM 2 a 2 e colocar em registradores temporários utilizando LDRB. Comparar se um número é menor que o outro, se o primeiro for o menor não fazer nada. Se o segundo for o menor trocar as posições na RAM através do STRB.
8) Repetir o passo anterior para todas as iterações do algoritmo *bubble sort*, até a lista estar ordenada.