Ministério da Educação
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
Campus Curitiba
Departamento Acadêmico de Eletrônica (Daeln-CT)
ELF52 – Sistemas Microcontrolados

Prof.: Guilherme de S. Peron
Prof.: Marcos E. P. Monteiro
Prof.: Paulo D. G. da Luz

Atividade Prática 1 - Arquitetura, IDE e Assembly

Contexto:

Números primos são os números naturais que têm **apenas dois divisores diferentes**: o 1 e ele mesmo. Desta forma o 1 não é primo, pois ele só tem 1 divisor.

Objetivo:

Utilizando instruções *Assembly* para Cortex-M4 e o simulador do Keil uVision (alternativamente pode ser utilizado o *Code Composer Studio*), 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 0x20000400;
- Verificar quais destes números são primos e copiar apenas estes números a partir da posição 0x20000500;
- Ordenar esta nova lista do menor para o maior pelo algoritmo *bubble sort*.

Ao finalizar o código, após simular as instruções e finalizar os testes de mesa criar um arquivo **readme.txt** na pasta AP1:

Conteúdo ao arquivo (readme.txt):

```
Disciplina: ELF52
Atividade Prática: 1
Equipe:
Nome do aluno 1, RA
Nome do aluno 2, RA
..., ...
Data: XX/YY/ZZZZ
```

Para entregar o Laboratório enviar um *email* para o professor com o diretório do projeto compactado em um arquivo .zip.

```
Email: garcez@professores.utfpr.edu.br
Título: ELF52 – Atividade Prática 1
Corpo do email:
Equipe:
Aluno1, RA
Aluno2, RA
Anexo:  AP1.zip
```

Roteiro:

Dada a seguinte lista de números aleatórios {193; 63; 176; 127; 43; 13; 211; 3; 203; 5; 21; 7; 206; 245; 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 0x20000400 e nome EQU 0x20000500).
2) Carregar a lista de números aleatória para a memória RAM a partir da posição 0x20000400 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 0x20000500, 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 0x20000500.

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 0x20000400
   nome EQU 0x20000500
2) Utilizar o comando de gravar na memória RAM em bytes (**STRB**) e ler da memória RAM em bytes (**LDRB**)(Método 01); Utilizar um Vetor definido na memória ROM no final do programa e carregar ele para o endereço de RAM(Método 02);
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 / 2 +1) 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 0x20000500, 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.
9) Primeiros números Primos (até 8bits ...):

2	3	5	7	11	13	17	19	23	29
31	37	41	43	47	53	59	61	67	71
73	79	83	89	97	101	103	107	109	113
127	131	137	139	149	151	157	163	167	173
179	181	191	193	197	199	211	223	227	229
233	239	241	251						