Ministério da Educação
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
Campus Curitiba
Departamento Acadêmico de Eletrônica
ELEW31 - Sistemas Microcontrolados

Prof.: Guilherme de Santi Peron

LAB 0 - IDE e Assembly

Objetivo:

Utilizando instruções Assembly para Arm Cortex-M4 e o simulador do Keil uVision, implementar um programa capaz de identificar por busca exaustiva uma **série de progressão geométrica** em um vetor de **16 bits** (2 bytes) armazenado na memória RAM.

Ao final, o estudante deverá ser capaz de:

- Ordenar um vetor de inteiros de 16 bits utilizando ***Bubble Sort;***
- Implementar uma busca de **Progressão Geométrica (PG)** por busca exaustiva (força bruta);
- Identificar automaticamente uma sequência de **pelo menos 5 termos** que formam uma PG;
- Armazenar o resultado encontrado em um novo vetor.

Contexto:

Você receberá um vetor de dados de 30 elementos de 16 bits, armazenados em uma região da memória RAM (a partir do endereço 0x20000400);

Os dados estão em ordem aleatória, mas existe uma **única** progressão geométrica com **mais de 5 elementos**, com razão *q* dentro do vetor.

Escreva um programa em Assembly para Arm Cortex-M4 que:

- Ordene o vetor em ordem crescente utilizando *Bubble Sort*;
- Procure no vetor uma sequência de elementos que obedeça as regras de uma PG;
- Confirme que a sequência encontrada possui pelo menos 5 termos;
- Armazene a sequência encontrada em um vetor na memória (a partir do endereço 0x20000600);

Tarefas:

- Carregar um vetor na memória RAM a partir da posição 0x20000400, utilizando um arquivo .ini fornecido pelo professor.
- Implementar o algoritmo de ordenação Bubble Sort para organizar o vetor em ordem crescente;
- Após a ordenação, testar todas as combinações possíveis de pares (*i*, *j*) no vetor para determinar se formam uma razão válida de PG;
- Expandir a sequência multiplicando pelo valor da razão até que não seja mais encontrado o próximo termo no vetor;
- Confirmar se a sequência encontrada contém pelo menos 5 termos em PG;
- Copiar a sequência detectada para um vetor de 16 bits na memória RAM a partir da posição 0x20000600, com os *n* elementos da PG, inclusive os elementos cujo índice for maior que 5.

Mostrar para o professor e depois entregar a pasta do projeto Keil com todos os arquivos zipada, 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 alunos da equipe. Ex.: fulanodetal1_fulanodetal2_fulanodetal3_ap0.zip. Apenas um membro da equipe precisa enviar.

Dicas e Orientações:

1) Carregar o arquivo de inicialização “vetor.ini” da memória RAM fornecido pelo professor, fornecido em anexo neste lab. Para carregar um arquivo .ini na memória, siga os passos a seguir:
    a) Clique com o botão direito, em “Target 1”;
    b) Selecione “Options for Target ‘Target 1’”;
    c) Vá na aba “Debug”;
    d) Na seção do “Use Simulator”, em Initialization File, clique nos 3 pontos (…) e selecione o arquivo .ini fornecido.
    e) Em seguida clique em OK.

2) Para visualizar a memória RAM como vetor de 16 bits, durante a simulação clique com o botão direito em qualquer lugar no Memory 1, vá em “Unsigned” —> “Short”. Se quiser ver em decimal ao invés de hexa, marcar a opção “Decimal”.
3) Declarar antes do label start um “EQU” para definir a posição base da memória RAM para o vetor
   nome1 EQU 0x20000400
   nome2 EQU 0x20000600
4) Para ler os valores da memória RAM, utilizar **LDRH**;
5) Para fazer a escrita na memória RAM utilizar **STRH**;
6) A ordenação dos números pode ser feita lendo posições da memória RAM 2 a 2 e colocar em registradores temporários utilizando LDRH. 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 por meio de STRH;
7) Para fazer a leitura da memória RAM, utilizar um dos métodos de endereçamento indexado que achar mais conveniente: com offset, pré-indexado ou pós-indexado;
8) 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);
9) Durante a varredura da PG conforme for encontrando os múltiplos, pode-se utilizar a pilha para armazenamento temporário da sequência. Lembrar de desempilhar também no caso de a sequência não atingir o número mínimo de termos.