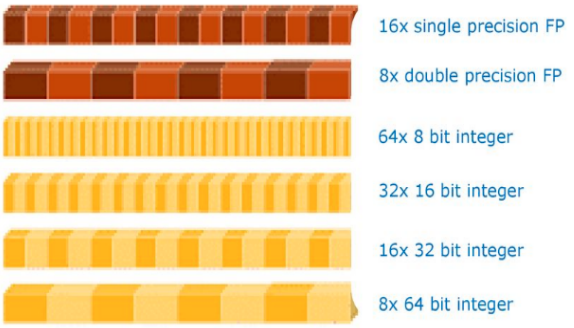
UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
ESCOLA POLITÉCNICA
DEPARTAMENTO DE ENGENHARIA ELETRÔNICA E DE COMPUTAÇÃO

DISCIPLINA ARQUITETURA DE COMPUTADORES

Atividade Prática AVX

Contextualização:

Diferentes arquiteturas de processadores atuais, suportam algum tipo de instruções vetoriais. Estas instruções permitem ao programador extrair mais desempenho do *hardware* disponível, nos processadores x86-64 (Intel e AMD) a versão mais recente desta tecnologia é conhecida como *Advanced Vector Extension* (AVX). Atualmente em sua versão AVX-512, ou seja, a arquitetura suporta operações usando operandos de 512 bits, onde estes bits podem codificar diferentes tipos de vetores numéricos como mostra a figura ao lado.



A listagem abaixo extraída do manual da Intel algumas das instruções de adição oferecidas pela extensão AVX-512:

- **vaddpd ou _mm512_add_pd (__m512d a, __m512d b)**: Add packed double-precision (64-bit) floating-point elements in a and b;
- **vaddps ou _mm512_add_ps (__m512 a, __m512 b)**: Add packed single-precision (32-bit) floating-point elements in a and b;
- **vaddph ou _mm512_add_ph (__m512h a, __m512h b)**: Add packed half-precision (16-bit) floating-point elements in a and b;
- **vpaddb ou _mm512_add_epi8 (__m512i a, __m512i b)**: Add packed 8-bit integers in a and b;
- **vpaddw ou _mm512_add_epi16 (__m512i a, __m512i b)**: Add packed 16-bit integers in a and b;
- **vpaddd ou _mm512_add_epi32 (__m512i a, __m512i b)**: Add packed 32-bit integers in a and b;
- **vpaddq ou _mm512_add_epi64 (__m512i a, __m512i b)**: Add packed 64-bit integers in a and b;
- **vpaddusb ou _mm512_adds_epu8 (__m512i a, __m512i b)**: Add packed unsigned 8-bit integers in a and b using saturation;
- **vpaddusw ou _mm512_adds_epu16 (__m512i a, __m512i b)**: Add packed unsigned 16-bit integers in a and b using saturation.

Assim, a figura e listagem mostram que temos todos os dados de entrada e a saída possuem 512 bits, sendo codificados sobre esses números com 8, 16, 32 e 64 bits, ou seja,

essas instruções recebem múltiplos números (vetores) como entrada. Especificamente 64 números de 8 bits por operando, 32 números de 16 bits por operando, 16 números de 32 bits por operando, 8 números de 64 bits por operando. O desempenho oferecido por essas instruções decorre do fato das operações são realizadas paralelamente em todos os números destes vetores.

Enquanto que mais eficaz em extrair desempenho que as técnicas convencionais de arquitetura de computadores, esse tipo de solução tem dois pontos negativos. O primeiro é que atualmente ainda é responsabilidade do programador explicitamente chamar essas instruções e o segundo decorre do fato de que os circuitos aritméticos vetoriais ocupam uma área de silício maior que circuitos aritméticos escalares, não sendo factível reusar as unidades lógicas escalares para processamento vetorial. Felizmente os projetistas destes circuitos vetoriais conseguem usar o mesmo circuito para processar números de múltiplos tamanhos fixos.

Descrição:

A tarefa a ser realizada é o projeto usando LogiSIM-Evolution, um módulo VHDL que implementa um circuito combinacional de somador/subtrator vetorial de 32 bits. Esse **único circuito** deve operar sobre números inteiros de 4, 8, 16 e 32 bits, onde o tamanho do vetor é informado através de um sinal de controle. **O circuito deve ser projetado visando um sistema orientado a desempenho (baixa latência no cálculo das somas)**. Além do módulo VHDL, o discente deve montar um circuitos que permita o teste do módulo usando testvectora Esse circuito deve possuir apenas as seguintes entradas e saídas:

- Entradas
    - Operandos de 32 bits: **std_logic_vector(31 DOWNTO 0);**
        - A_i, B_i
    - Modo Somador ou Subtrator 1 bit: **std_logic;**
        - mode_i
    - Tamanho do vetor 2 bits: **std_logic_vector(1 DOWNTO 0)**
        - vecSize_i
            - 00 = 4, 01 = 8. 10 = 16 e 11 = 32
- Saída
    - S_o **std_logic_vector(31 DOWNTO 0).**

Critérios de Avaliação:

Serão avaliadas a corretude, aderência aos requisitos solicitados e quantidade de recursos gastos além do estritamente necessário. Cada trio deve entregar um relatório descritivo do seu projeto, destacando as premissas adotadas e o projeto do circuito.

Dicas:

Observar o projeto de circuitos com multiplos bits usand CIs *bit-sliceds* como o 74S182 (Carry-Lookahead) e 74S181 (ALU). Refletir no que significa o carry-in nestes e utilize o simulador para averiguar esse comportamento.