Introdução

A Atividade Acadêmica de Arquitetura I, tem como objetivo simular instruções vetoriais utilizando as instruções escalares do MIPS. Na arquitetura vetorial, SIMD (Single Instruction, Multiple Data), a mesma instrução é aplicada simultaneamente a diversos dados, e é largamente utilizado para acelerar a resolução de problemas computacionais que utilizam estruturas de dados regulares como vetores e matrizes.

Esse tipo de máquina opera aplicando uma única instrução a um conjunto de elementos de um vetor. Sendo uma máquina que aplique a *n* elementos uma determinada instrução e o vetor *t* contenha os elementos a serem processados, *t* terá todos seus elementos calculados *n* vezes mais rápido que uma máquina SISD (Single Instruction, Single Data) na mesma tarefa. O trabalho realizado por uma instrução vetorial corresponde ao trabalho realizado por uma estrutura do tipo laço (looping) em uma arquitetura escalar.

Implementação

Para simular as instruções vetoriais no MIPS utilizamos o Simulador MARS, onde cada instrução vetorial foi transformada em uma macro. Na proposta do trabalho havia uma tabela com instruções mandatórias para a realização do mesmo. Eis as seguintes instruções: infv, lv, sv, addv, multve, multvv, medv, addvf; as quais serão explicadas ao decorrer do relatório. Há, ainda, as quatros instruções adicionais: subv, cosv, liv, addiv.

- *Infv:* retorna a dimensão do registrador, isto é, quantos elementos vão ser lidos por vez.
- *Lv ($reg, vet):* O primeiro argumento da macro é o registrador onde irá ser salvo o vetor.
- *Sv ($reg1, $reg2):* O primeiro argumento do sv é o registrador que contém os elementos a ser copiado e o segundo registrador é o local destino da informação.
- *Liv ($reg, n):* Função iterativa onde *n* valores são carregados no vetor de *n* posições.
- *Addv ($reg0, $reg1, $reg2):* Soma-se os elementos dos registradores 1 e 2 são somados e guardados no registrador 0. Caso o registrador 1 (ou o 2) seja maior que o 2 (ou 1), os demais valores são copiados para o registrador destino.
- *Multve ($reg1, n):* Nessa função é feita a multiplicação escalar do vetor contido em reg1 por *n*, que é um número inteiro escolhido pelo usuário.
- *Multvv ($reg1, $reg2):* A função multvv faz o produto interno entre os registradores 1 e 2 com uma condição: Que os 2 tenham a mesma dimensão. Retorna o valor no registrador $v0.

- *Medv ($reg):* Faz a média dos valores contidos no vetor carregado em $reg. O valor é retornado em float.

- *Subv ($reg0, $reg1, $reg2):* Segue a mesma ideia da Addv, com apenas a mudança de uma linha de código (add por sub).

- *Cosv ($reg1, $reg2):* Calcula o cosseno do ângulo entre 2 vetores de inteiros, retorna um número (o cosseno) em float.

- *Addiv ($reg1, $reg2):* Soma uma constante, armazenada em $reg2, em todos os elementos do vetor armazenado em $reg1.

- *Addvf ($reg0, $reg1, $reg2):* 2 vetores de floats são somados e armazenados em um novo vetor ($reg0).

O projeto segue as seguintes diretrizes: o primeiro elemento do vetor sempre será o tamanho (dimensão) dele; nas funções matemáticas e de carregando do vetor na memória há dois loops, um para quando a dimensão do vetor for menor ou igual a dimensão do registrador do *processador vetorial.* E ou que enquanto a dimensão do vetor for maior que a dimensão do registrador. Como exemplo podemos usar a seguinte analogia:

Vet[10] ---------- Dimensão Reg[3]

Enquanto a dimensão do vetor for maior que a do registrador irá para o *segundo* loop descrito acima. Na primeira iteração será 10 > 3; na segunda, 7 > 3, uma vez que *decresce* a dimensão do vetor pelo fato desses elementos já terem sido afetados pela função (addv, medv, ...); até que na quarta iteração os valores serão 1 =< 3, logo irá entrar no *primeiro* loop da macro.

Conclusão

Todas as instruções cumpriram seus objetivos, assim como também aprendemos um pouco sobre arquitetura vetorial. Não há como traçar um linear, levando em consideração o tempo computacional gasto, entre um *processador vetorial* e nossa simulação no MARS, porém a simulação de trabalho da máquina vetorial foi alcançada com sucesso. Também houve um aumento plausível em nossa compreensão do simulador de MIPS assim como da linguagem Assembly, onde utilizamos macros e instruções de conversão de números *floats* para números inteiros, e vice-versa, o que conduz a um visual limpo do código, uma vez que uma função em uma macro é similar a uma chamada de função na linguagem *C.*