Module-2
CSEN 3104
Lecture 11

Dr. Debranjan Sarkar

Vector Processing

What is Vector Processing?

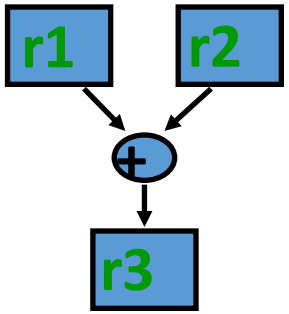
- In matrix algebra, a *vector* $a = [a_1, a_2, a_3, \ldots, a_n]^T$ is a column of numbers
- The scalars a_i are the elements of vector a
- In computer technology, a vector is an ordered set of scalar data items, all of the same type, stored in memory
- Usually, the vector elements are ordered to have a fixed addressing increment between successive elements, called the stride
- A vector processor is a central processing unit that can work on an entire vector in one instruction, whereas the instructions of a scalar processor operate on single data items
- In vector processing, arithmetic and logical operations are applied to vectors
- The conversion from scalar code to vector code is called vectorization

What is Vector Processing?

- The instruction to the processor is in the form of one complete vector instead of its element
- The operand to the instructions are complete vectors instead of one element
- Vector processors reduce the fetch and decode bandwidth as the number of instructions fetched are less
- Reduces software overhead for maintenance of looping control
- Enhanced performance but increased hardware and compiler cost
- A compiler capable of vectorization is called vectorizing compiler
- Vector processors can greatly improve performance on certain applications e.g. numerical simulation and similar tasks e.g. long-range weather forecasting, aerodynamics and space flight simulation etc.

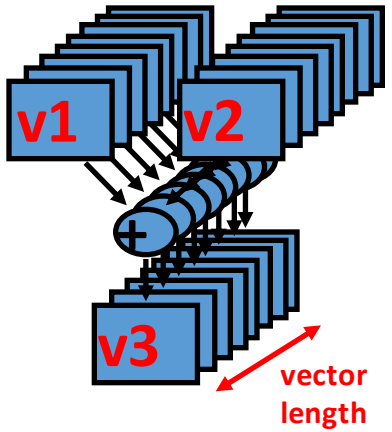
Scalar vs. Vector CPU

SCALAR
(1 operation)




```
add r3, r1, r2
```

VECTOR
(N operations)




```
addv v3, v1, v2
```

Types of Vector Architectures

- *Memory-memory vector processors*
  - all vector operations are memory to memory
  - There is no limitation of size
  - Speed is comparatively slow
  - The hardware cost is low
  - Example: *TI-ASC, CDC STAR-100, and Cyber-205*
- *Vector-register processors*
  - all vector operations are between vector registers (except load and store)
  - Vector equivalent of load-store architectures
  - Register to register architecture has limited size
  - Speed is very high as compared to the memory to memory architecture.
  - The hardware cost is high
  - Example: Cray-1, Fujitsu VP-200, Hitachi HITAC S-810, NEC SX-9

Components of a Vector Processor

- *Vector Register*: A set of registers for all vector operations (except load and store)
  - typically 8-32 vector registers, each holding 64-128 64-bit elements
  - In vector-register architecture, operands are read into vector registers from which they are fed to the functional units and results of operations are written to vector registers
- *Vector Functional Units (FUs)*
  - fully pipelined
  - start new operation every clock
  - FP add, FP multiply, FP reciprocal, integer add, logical, shift
  - may have multiple of same unit
- *Vector Load-Store Units (LSUs)*
  - fully pipelined unit to load or store a vector
  - there may be multiple LSUs
- *Scalar registers*
  - single element for FP scalar or address
- *Cross-bar*
  - to connect FUs , LSUs, registers
- Show the Vector Register Architecture

Vector Instruction Types

- Six types:
  - Vector-vector instructions
    - $f_1: V_j \rightarrow V_i$ Example: $V_1 = \sin(V_2)$
    - $f_2: V_j \ X \ V_k \rightarrow V_i$ Example: $V_3 = V_1 + V_2$
  - Vector-scalar instructions
    - $f_3: s \ X \ V_k \rightarrow V_i$ Example: $s \ X \ V_1 = V_2$
  - Vector-memory instructions
    - $f_4: M \rightarrow V$ Vector Load
    - $f_1: V \rightarrow M$ Vector Store
  - Vector reduction instructions
    - $f_6: V \rightarrow s$ Find maximum, minimum, sum, mean of all elements of a vector
    - $f_7: V_i \ X \ V_j \rightarrow s$ Dopt product of two vectors
  - Gather and scatter instructions
  - Masking instructions

What is a sparse vector?

- A sparse vector is a vector, most of the elements of which are zero
- It is inefficient to use a one-dimensional array to store a sparse vector
- It is also inefficient to add elements whose values are zero in forming sums of sparse vectors
- Gather, Scatter and masking instructions are very useful in handling sparse vectors or matrices
- Advanced vector processors implement these instructions directly in hardware

Gather, Scatter, Masking Instructions

- Gather and scatter instructions
  - f_8: $M \rightarrow V_1$ X V_0 Gather
  - f_9: V_1 X $V_0 \rightarrow M$ Scatter
- Gather fetches from memory the non-zero elements of a sparse vector using indices that themselves are indexed
- Scatter stores into memory a vector in a sparse vector whose non-zero entries are indexed
- The vector register V_1 contains the data and the vector register V_0 is used as an index to gather or scatter data from or to random memory locations
- Masking Instructions
  - f_{10}: V_0 X $V_m \rightarrow V_1$
- This instruction uses a mask vector to compress or to expand a vector to a shorter or longer index vector

Thank you