Evaluation of vector libraries on AJIT

Gauri Patrikar

October 19, 2021

Abstract

Vector or SIMD (single instructions multiple data) instructions are a class of instructions that enable parallel processing. It performs the same operation on multiple input data streams at the same time, exploiting data level parallelism, hence enhancing the performance. AJIT processor is an implementation of the SPARC-V8 ISA. We have added 64 bit vector instructions to the existing 32 bit SPARC V8 ISA, implemented on the AJIT 64 bit processor. These instructions use 2 32 bit registers to execute a 64 bit instruction.

We developed String and blas libraries using the SPARC V8 assembly with the newly added extensions to the AJIT processor. DSP functions using these blas libraries were also implemented.

The performance for implemented string and blas functions is characterized and improvement in performance of upto 92% for string and upto 96% for blas was observed.The performance is measured in terms of clock cycles required.

1 Introduction

Vector instructions are widely used in AI, DSP and graphics applications to significantly enhance the performance of these operations. They perform operations on multiple data sets in a single clock cycle. The SPARC V8 is a 32 bit ISA, which has no vector instructions. For the AJIT 64 bit processor, we have extended the ISA to include 64 bit instructions, some of which are vector instructions. Using these instructions, we have developed string and linear algebra libraries. These libraries have shown significant improvement over

original functions which use the 32 bit SPARC ISA. These newly developed libraries an be used for various appplications in AI/DSP etc. A couple of DSP functions are also developed to demonstrate the enhanced performance.

The paper is divided into following sections - section 3 details the vector instructions, section 4 talks about the SPARC ABI, section 5 gives specifications of all implemented libraries, section 6 gives the testing methodology, section 7 outlines the results, and section 8 concludes the paper.

2 AJIT Vector Instructions

The AJIT processor is based on the 32 bit SPARC V8 ISA. New assembly instructions that perform 64 bit operations have been added to enhance the performance. They can be categorized as follows -

- Integer unit extensions - 64 bit arithmetic-logic-shift instructions. These accept 64 bit input operands and the output is also 64 bits.
- SIMD - These instructions are vector instructions which work on two source registers (each a 64 bit register pair), and produce a 64-bit vector result. The vector elements can be 8-bit/16-bit/32-bit.
- SIMD (bytereduce) - These instructions are vector instructions which reduce a 64 bit source register to a destination using an associative operation.
- Floating point vector - These are vector float operations which work on two single precision operand pairs to produce two single precision results.
- Half precision conversion instructions - These instructions allow conversion between IEEE half-precision numbers and IEEE single/double precision numbers and integers.

These instructions are performed in a single clock cycle and help reduce number of operations, hence making a program much faster.

Strings terminate with a null byte. Hence, finding the zeroth byte is important. Using the zbytedpos (a type of byte reduce instruction, gives the positions of zero bytes in a 64 bit input) instruction, we can find the null

byte in a 64 bit number in one instruction itself. In case of other instructions also, operations are performed two at once, for eg., multiplying up to 8 8 byte inputs at once.

3 SPARC ABI

SPARC ABI is a calling convention used when an assembly program needs to be called using a C program. It gives the protocol for argument passing, writing the prologue and epilogue of the assembly functions and so on. This ensures that the process of shifting between the C and the assembly programs is seamless.

The libraries have been written in assembly, simply because generating a compiler was a more complex task. These libraries can be called from a C program, and hence the SPARC ABI protocol is used.

4 Implemented functions

We have implemented 5 string functions and 5 blas functions. The blas functions have been written for 4 data types. 2 DSP functions using the blas libraries were also written. Th list of libraries is given blow, followed with restrictions and usage of these libraries.

4.1 String

The following functions are being implemented here (all functions input two strings) -

1. strcpy - copies source string to destination
2. strncpy - copies n bytes of source string to destination.
3. strcmp - compares two input strings
4. strcasecmp - compare two input strings ignoring case of the character
5. strcat - concatenates source string at the end of destination string.

4.2 Blas

Each of the given functions are implemented for 5 data types - 8, 16 and 32 bit unsigned int and single precision floating point.

1. asum - sum of absolute values of an array
2. scal - scales elements of array by a constant
3. axpy - implements on each element of the array
4. dot - computes the dot product between two arrays
5. Isamax - finds the largest element of the array

Using these Blas libraries following functions were implemented -

1. FFT
2. Convolution

4.3 Restrictions

As we were dealing with 64 bit operations, aligning the inputs to 64 bit boundaries was required. It is the users responsibility to make sure all inputs are aligned to make use of the 64 bit libraries.

4.4 Usage

All of the mentioned functions can be called from a C program.

5 Testing framework

We have used a cross-compiler to generate SPARC-V8 executable code. We have added the 64 bit extensions to the binary utils, hence the 64 bit instructions can be assembled.

The libraries have been tested on the AJIT processor core on an FPGA prototyping board (KC705).

We have compared the string libraries with the existing uclibc C string functions. The blas functions are compared to those in the gnu scientific

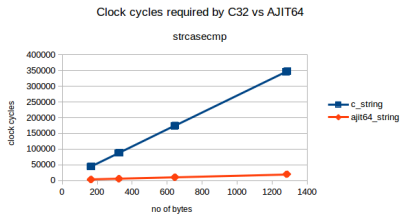
library. The DSP functions were compared by writing both in C and adding inline assembly for the vector case.

6 Results

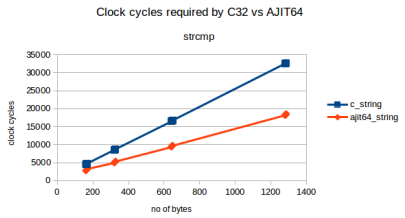
The results are given in terms of clock cycles required by each function.

6.1 String

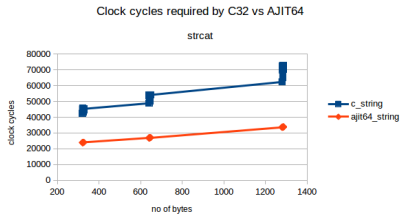
For string libraries, functions are compared at various string lengths. taken in multiples of 160 bytes. We have tested for 160*K+n, where k = 1,2,4,8 and n = 0 to 7. In each case improvement is noted.



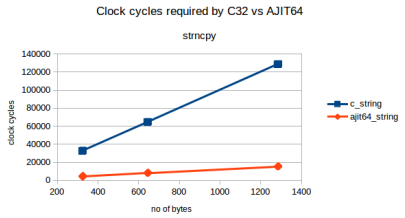
(a) strcasecmp



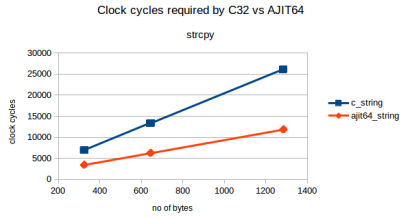
(b) strcmp



(c) strcat



(d) strncpy

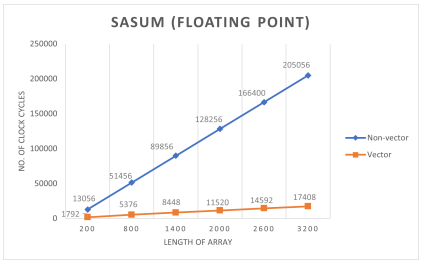


(e) strcpy

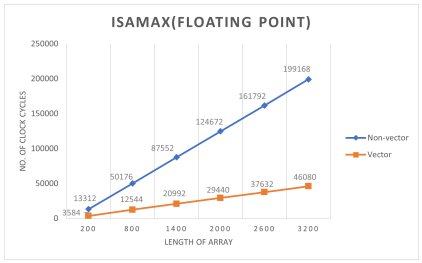
Figure 1: String functions

6.2 Blas

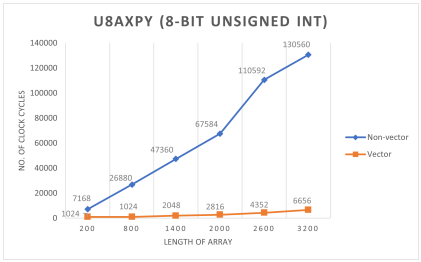
For blas libraries, lengths of have been taken from 200 to 3200 in increments of 600. For each function comparison of one data type out of the four is given.



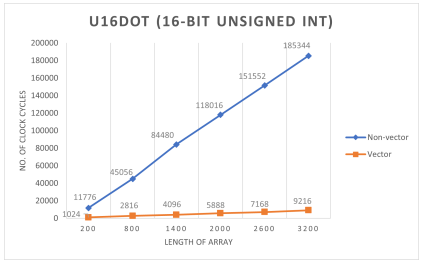
(a) sum



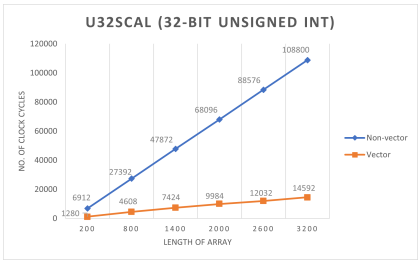
(b) max



(c) u8axpy



(d) dot product



(e) scaling

Figure 2: Blas functions

7 Conclusions

In this paper we have presented newly developed vector string and blas libraries. All of libraries have shown significant improvement of up to 96% compared to their non vectored equivalent. DSP applications making use of

these libraries also showed improvement of . All these functions have multiple iterations as the main function. These iterations are reduced greatly due to the use of 64 bit and vector instructions, which provides the performance enhancement.