RYERSON UNIVERSITY

Faculty of Engineering, Architecture, and Science

Department of Electrical and Computer Engineering

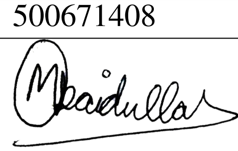
Course Number	EE8207
Course Title	High Performance Computer System Design
Semester/Year	Winter/2016

Instructor	Dr. Nagi N. Mekhiel

Lab No.	**2**

Assignment Title	Evaluating Performance of Computer Systems

Submission Date	12^{th} February 2016
Due Date	12^{th} February 2016

Student Name	Muhammad Obaidullah
Student ID.	500671408
Signature*	

*By signing above you attest that you have contributed to this written lab report and confirm that all work you have contributed to this lab report is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a "0" on the work, an "F" in the course, or possibly more severe penalties, as well as a Disciplinary Notice on your academic record under the Student Code of Academic Conduct, which can be found online at: www.ryerson.ca/senate/current/pol60.pdf.

1 Objectives of the lab

1. Evaluating performance of a computer system with SimpleScalar.

2. Using Benchmarks.

3. Measure performance of real computer and compare its performance to a simulator based system that uses same parameters (cache, speed, bus bandwidth).

2 Selecting Benchmarking Applications

2.1 Application 1: Dhrystone

Dhrystone program is an old benchmark which was written in 1984 by Reinhold Weicker and measured integer performance of processors and compilers. Since then, it has been replaced by more complex benchmarking programs such as SPEC and CoreMark. [1]

Dhrystone evaluates general-purpose integer performance of the DUT (Device Under Test). However it does not resemble any real-life program, is very susceptible to compiler optimizations, and due to the small code size, it may fit in the instruction cache of a modern CPU hence diluting instruction fetch performance.[2]

2.2 Application 2: Linpack

Linpack benchmark was introduced by Jack Dongarra. This benchmarks tries to solve dense system of linear equations. [3] The benchmark is designed to solve system of linear equations in the form of $Ax = b$ with three different sizes: 100×100 problem (inner loop), 1000×1000 problem (three loop complete program), and a scalable parallel problem. Linpack was actually ported from Fortan programming language into c so that it can benchmark more variety of computers.

2.2.1 Versions of Linpack benchmark

There are pre-processor symbols in the code which can be defined in order to compile 4 different versions of the code.

1. **Single Precision with Rolled loop (SP-RL):** The first version which can be used as benchmark involves solving several floating-point single precision linear equations without unrolling the loop. This version is supposed to be less computationally intensive than double precision linear equations. Rolled loop means 1 linear equation per iteration. This means that after each equation is solved, then the iteration is incremented.

2. **Double Precision with Rolled loop (DP-RL):** As the name suggests, double precision (64-bits) variable holds twice the bits of a float variable (32-bits). This version is more computationally intensive than single precision because of the double accuracy in calculating values and handling 64-bit values. Since the loop is rolled, this means that after each equation is solved, then the iteration is incremented. 1 linear equation per iteration.

3. **Single Precision with Un-rolled loop (SP-UL):** The third version which can be used as benchmark involves solving several floating-point single precision linear equations and taking advantage of unrolling the loop. This means that several equations are being solved in a single loop iteration. This reduces the control overhead from the instructions and reduces the total number of instructions to execute.

4. **Double Precision with Un-rolled loop (DP-UL):** As the loop is unrolled, several control hazards and the latency due to control hazards is reduced. So this version of code might take less time to execute compared to rolled loop double precision version. However, this version is more computationally intensive than single precision because of the double accuracy in calculating values and handling 64-bit values. The depth of unrolling the loop can also be controlled in the c program using pre-processor symbols.

2.2.2 Effect of Unrolling the Loop

Following is a rolled loop:

```
for (int i = 0; i < 10; i++)
{
  y[i] = y[i] + alpha*x[i];
}
```

Following is the loop unrolled 4 times. This reduces the control overhead from the instructions and reduces the total number of instructions to execute. This is achieved by setting the unrolling depth to 4 in the *linpack.c* file:

```
for (int i = 0; i < 10; i = i + 4)
{
  y[i] = y[i] + alpha*x[i];
  y[i+1] = y[i+1] + alpha*x[i+1];
  y[i+2] = y[i+2] + alpha*x[i+2];
  y[i+3] = y[i+3] + alpha*x[i+3];
}
```

2.3 Application 3: Whetstone

Whetstone is a statistics based synthetic program which is widely used for benchmarking CPUs and parallel CPU clusters. Results obtained from running Whetstone program were written in terms of Millions of Whetstone Instructions Per Second (MWIPS). The program iterates through many instructions and performs complex trigonometric and root operations to test the full potential of the Device Under Test (DUT). It each main application loop, it goes through several modules which include following:

1. **Simple Identifiers:** Continuously assigns integer values to variables iteratively.
2. **Array Elements:** Calculates array elements by performing operations on other array elements.
3. **Array as parameters:** Passes an array to a function as a parameter several times.
4. **Conditional Jumps:** Does couple of conditional jumps (if statements) based on an integer value.
5. **Integer Arithmetic:** Does complex integer calculations involving addition, subtraction, multiplication, and division. Also involves some array values into calculations.
6. **Trigonometric Functions:** Calls Sin, Cos, and Tan functions repeatedly.
7. **Procedure Calls:** Calls a function several times and passes values by value and reference as well.
8. **Array References:** Shuffles 3 array elements around several times.
9. **Integer Arithmetic:** Contains several simple integer addition and subtractions.
10. **Standard Functions:** Performs several square root, exponential, and logarithmic functions on a dummy variable.

3 Benchmarking Procedure

3.1 Running Code on Simulator

In order to benchmark the PC by running the same program on PC, a profiling tool is needed. Fortunately, GNU tool-chain has a built-in profiling tool called as *gprof*. In order profile a c code *(whetstone.c)* on Linux, following steps are taken.

1. Compile the program using simpleScalar gcc compiler.

```
~/SScalar3.0d >> ./bin/sslitlle-na-sstrix-gcc benchmark-codes/uncompiled/linpack.c
```

This will generate a.out file which is executable using the SimpleScaler simulator.

2. Run the program

```
~/SScalar3.0d >> ./simple-sim-3.0/sim-safe a.out
```

3. Traces are generated by using the following command:

```
sim-outorder -ptrace FOO.trc :1000 test-math
```

4. To view the trace file, following command is run to instantiate pipeview program so that the trace file is parsed and is displayed in a proper manner:

```
pipeview.pl FOO.trc
```

3.2 Running Code on PC

In order to benchmark the PC by running the same program on PC, a profiling tool is needed. Fortunately, GNU tool-chain has a built-in profiling tool called as *gprof*. In order profile a c code *(whetstone.c)* on Linux, following steps are taken.

1. Compile the program using the "-pg" option

```
~/SScalar3.0d >> gcc -Wall -pg -lm benchmark-codes/uncompiled/whetstone.c -o benchmark-
    codes/compiled-pc/whetstone
```

2. Run the program

```
~/SScalar3.0d >> ./benchmark-codes/compiled-pc/whetsone
```

3. Generate the output file and save statistics

```
~/SScalar3.0d >> gprof whetstone whetstone-gmon.out > whetstone-results.txt
```

This will run the executable and store the results in the text file *whetstone-results.txt*

4 Results

4.1 Application 1: Dhrystone

4.1.1 PC Results

```
Each sample counts as 0.01 seconds.
%   cumulative   self              self     total
time   seconds   seconds    calls  ms/call  ms/call  name
25.07      0.02     0.02   500000     0.00     0.00  Proc1
18.80      0.04     0.02  1500000     0.00     0.00  Proc7
12.54      0.05     0.01  1500000     0.00     0.00  Func1
12.54      0.06     0.01   500000     0.00     0.00  Proc3
12.54      0.07     0.01   500000     0.00     0.00  Proc8
12.54      0.08     0.01        1    10.03    80.23  Proc0
6.27      0.08     0.01   500000     0.00     0.00  Proc6
0.00      0.08     0.00   500000     0.00     0.00  Func2
0.00      0.08     0.00   500000     0.00     0.00  Func3
0.00      0.08     0.00   500000     0.00     0.00  Proc2
0.00      0.08     0.00   500000     0.00     0.00  Proc4
0.00      0.08     0.00   500000     0.00     0.00  Proc5
```

4.1.2 Simulator Results

```
sim: ** simulation statistics **
sim_num_insn                533507945 # total number of instructions executed
sim_num_refs                215504362 # total number of loads and stores executed
sim_elapsed_time                   19 # total simulation time in seconds
sim_inst_rate           28079365.5263 # simulation speed (in insts/sec)
ld_text_base               0x00400000 # program text (code) segment base
ld_text_size                    28080 # program text (code) size in bytes
ld_data_base               0x10000000 # program initialized data segment base
ld_data_size                    11876 # program init'ed `.data' and uninit'ed `.bss' size in bytes
ld_stack_base              0x7fffc000 # program stack segment base (highest address in stack)
ld_stack_size                   16384 # program initial stack size
ld_prog_entry              0x00400140 # program entry point (initial PC)
ld_environ_base            0x7fff8000 # program environment base address address
ld_target_big_endian                0 # target executable endian-ness, non-zero if big endian
mem.page_count                     17 # total number of pages allocated
mem.page_mem                      68k # total size of memory pages allocated
mem.ptab_misses                    19 # total first level page table misses
mem.ptab_accesses          2565216032 # total page table accesses
mem.ptab_miss_rate             0.0000 # first level page table miss rate
```

4.2 Application 2A: Linpack - Single Precision with Rolled Loop (SP-RL)

4.2.1 PC Results

```
Each sample counts as 0.01 seconds.
%   cumulative   self              self     total
time   seconds   seconds    calls  ms/call  ms/call  name
100.06      0.03     0.03   133874     0.00     0.00  daxpy
0.00      0.03     0.00     2574     0.00     0.00  dscal
0.00      0.03     0.00     2574     0.00     0.00  idamax
0.00      0.03     0.00       72     0.00     0.00  second
0.00      0.03     0.00       27     0.00     0.00  matgen
0.00      0.03     0.00       26     0.00     1.11  dgefa
0.00      0.03     0.00       26     0.00     0.04  dgesl
0.00      0.03     0.00        8     0.00     0.00  print_timer
0.00      0.03     0.00        1     0.00     0.00  dmxpy
0.00      0.03     0.00        1     0.00     0.00  epslon
```

4.2.2 Simulator Results

```
sim: ** simulation statistics **
sim_num_insn                342182095 # total number of instructions executed
sim_num_refs                143969999 # total number of loads and stores executed
sim_elapsed_time                   13 # total simulation time in seconds
sim_inst_rate          26321699.6154 # simulation speed (in insts/sec)
ld_text_base               0x00400000 # program text (code) segment base
ld_text_size                    97776 # program text (code) size in bytes
ld_data_base               0x10000000 # program initialized data segment base
ld_data_size                   332292 # program init'ed `.data' and uninit'ed `.bss' size in bytes
ld_stack_base              0x7fffc000 # program stack segment base (highest address in stack)
ld_stack_size                   16384 # program initial stack size
ld_prog_entry              0x00400140 # program entry point (initial PC)
ld_environ_base            0x7fff8000 # program environment base address address
ld_target_big_endian                0 # target executable endian-ness, non-zero if big endian
mem.page_count                     78 # total number of pages allocated
mem.page_mem                     312k # total size of memory pages allocated
mem.ptab_misses                  3101 # total first level page table misses
mem.ptab_accesses          1658370601 # total page table accesses
mem.ptab_miss_rate             0.0000 # first level page table miss rate
```

4.3 Application 2B: Linpack - Double Precision with Rolled Loop (DP-RL)

4.3.1 PC Results

```
Each sample counts as 0.01 seconds.
%    cumulative    self              self     total
time    seconds    seconds    calls   us/call  us/call   name
50.03      0.01      0.01   133874     0.07     0.07   daxpy
50.03      0.02      0.01       26   384.86   754.84   dgefa
0.00       0.02      0.00     2574     0.00     0.00   dscal
0.00       0.02      0.00     2574     0.00     0.00   idamax
0.00       0.02      0.00       72     0.00     0.00   second
0.00       0.02      0.00       27     0.00     0.00   matgen
0.00       0.02      0.00       26     0.00    14.87   dgesl
0.00       0.02      0.00        8     0.00     0.00   print_timer
0.00       0.02      0.00        1     0.00     0.00   dmxpy
0.00       0.02      0.00        1     0.00     0.00   epslon
```

4.3.2 Simulator Results

```
sim: ** simulation statistics **
sim_num_insn                329275681 # total number of instructions executed
sim_num_refs                139015684 # total number of loads and stores executed
sim_elapsed_time                   12 # total simulation time in seconds
sim_inst_rate          27439640.0833 # simulation speed (in insts/sec)
ld_text_base               0x00400000 # program text (code) segment base
ld_text_size                    96896 # program text (code) size in bytes
ld_data_base               0x10000000 # program initialized data segment base
ld_data_size                   655012 # program init'ed `.data' and uninit'ed `.bss' size in bytes
ld_stack_base              0x7fffc000 # program stack segment base (highest address in stack)
ld_stack_size                   16384 # program initial stack size
ld_prog_entry              0x00400140 # program entry point (initial PC)
ld_environ_base            0x7fff8000 # program environment base address address
ld_target_big_endian                0 # target executable endian-ness, non-zero if big endian
mem.page_count                    115 # total number of pages allocated
mem.page_mem                     460k # total size of memory pages allocated
```

```
mem.ptab_misses                 10516 # total first level page table misses
mem.ptab_accesses          1672529376 # total page table accesses
mem.ptab_miss_rate             0.0000 # first level page table miss rate
```

4.4 Application 2C: Linpack - Single Precision with Un-rolled Loop (SP-UL)

4.4.1 PC Results

```
Each sample counts as 0.01 seconds.
%    cumulative    self              self     total
time    seconds    seconds    calls  us/call  us/call  name
100.06      0.02      0.02   133874     0.15     0.15  daxpy
0.00        0.02      0.00     2574     0.00     0.00  dscal
0.00        0.02      0.00     2574     0.00     0.00  idamax
0.00        0.02      0.00       72     0.00     0.00  second
0.00        0.02      0.00       27     0.00     0.00  matgen
0.00        0.02      0.00       26     0.00   739.94  dgefa
0.00        0.02      0.00       26     0.00    29.75  dgesl
0.00        0.02      0.00        8     0.00     0.00  print_timer
0.00        0.02      0.00        1     0.00     0.00  dmxpy
0.00        0.02      0.00        1     0.00     0.00  epslon
```

4.4.2 Simulator Results

```
sim: ** simulation statistics **
sim_num_insn                303113503 # total number of instructions executed
sim_num_refs                118762050 # total number of loads and stores executed
sim_elapsed_time                   11 # total simulation time in seconds
sim_inst_rate           27555773.0000 # simulation speed (in insts/sec)
ld_text_base               0x00400000 # program text (code) segment base
ld_text_size                   100816 # program text (code) size in bytes
ld_data_base               0x10000000 # program initialized data segment base
ld_data_size                   332308 # program init'ed `.data' and uninit'ed `.bss' size in bytes
ld_stack_base              0x7fffc000 # program stack segment base (highest address in stack)
ld_stack_size                   16384 # program initial stack size
ld_prog_entry              0x00400140 # program entry point (initial PC)
ld_environ_base            0x7fff8000 # program environment base address address
ld_target_big_endian                0 # target executable endian-ness, non-zero if big endian
mem.page_count                     77 # total number of pages allocated
mem.page_mem                     308k # total size of memory pages allocated
mem.ptab_misses                110080 # total first level page table misses
mem.ptab_accesses          1451588619 # total page table accesses
mem.ptab_miss_rate             0.0001 # first level page table miss rate
```

4.5 Application 2D: Linpack - Double Precision with Un-rolled Loop (DP-UL)

4.5.1 PC Results

```
Each sample counts as 0.01 seconds.
%    cumulative    self              self     total
time    seconds    seconds    calls  us/call  us/call  name
100.06      0.02      0.02   133874     0.15     0.15  daxpy
0.00        0.02      0.00     2574     0.00     0.00  dscal
0.00        0.02      0.00     2574     0.00     0.00  idamax
0.00        0.02      0.00       72     0.00     0.00  second
```

```
0.00      0.02      0.00       27     0.00     0.00  matgen
0.00      0.02      0.00       26     0.00   739.93  dgefa
0.00      0.02      0.00       26     0.00    29.75  dgesl
0.00      0.02      0.00        8     0.00     0.00  print_timer
0.00      0.02      0.00        1     0.00     0.00  dmxpy
0.00      0.02      0.00        1     0.00     0.00  epslon
```

4.5.2 Simulator Results

```
sim: ** simulation statistics **
sim_num_insn              299999180 # total number of instructions executed
sim_num_refs              117945956 # total number of loads and stores executed
sim_elapsed_time                 12 # total simulation time in seconds
sim_inst_rate         24999931.6667 # simulation speed (in insts/sec)
ld_text_base             0x00400000 # program text (code) segment base
ld_text_size                  99920 # program text (code) size in bytes
ld_data_base             0x10000000 # program initialized data segment base
ld_data_size                 655028 # program init'ed `.data' and uninit'ed `.bss' size in bytes
ld_stack_base            0x7fffc000 # program stack segment base (highest address in stack)
ld_stack_size                 16384 # program initial stack size
ld_prog_entry            0x00400140 # program entry point (initial PC)
ld_environ_base          0x7fff8000 # program environment base address address
ld_target_big_endian              0 # target executable endian-ness, non-zero if big endian
mem.page_count                  157 # total number of pages allocated
mem.page_mem                   628k # total size of memory pages allocated
mem.ptab_misses                5061 # total first level page table misses
mem.ptab_accesses        1515660884 # total page table accesses
mem.ptab_miss_rate           0.0000 # first level page table miss rate
```

4.6 Application 3: Whetstone

4.6.1 PC Results

```
Each sample counts as 0.01 seconds.
%   cumulative   self              self     total
time   seconds   seconds    calls  ns/call  ns/call  name
60.11      0.09     0.09  8091000    11.14    11.14  P3
26.72      0.13     0.04                             main
6.68       0.14     0.01  5544000     1.81     1.81  P0
6.68       0.15     0.01   126000    79.52    79.52  PA
```

4.6.2 Simulator Results

```
sim: ** simulation statistics **
sim_num_insn              155673615 # total number of instructions executed
sim_num_refs               45353172 # total number of loads and stores executed
sim_elapsed_time                  5 # total simulation time in seconds
sim_inst_rate         31134723.0000 # simulation speed (in insts/sec)
ld_text_base             0x00400000 # program text (code) segment base
ld_text_size                  91056 # program text (code) size in bytes
ld_data_base             0x10000000 # program initialized data segment base
ld_data_size                  12288 # program init'ed `.data' and uninit'ed `.bss' size in bytes
ld_stack_base            0x7fffc000 # program stack segment base (highest address in stack)
ld_stack_size                 16384 # program initial stack size
ld_prog_entry            0x00400140 # program entry point (initial PC)
```

```
ld_environ_base          0x7fff8000 # program environment base address address
ld_target_big_endian              0 # target executable endian-ness, non-zero if big endian
mem.page_count                   33 # total number of pages allocated
mem.page_mem                   132k # total size of memory pages allocated
mem.ptab_misses                  34 # total first level page table misses
mem.ptab_accesses         739290943 # total page table accesses
mem.ptab_miss_rate           0.0000 # first level page table miss rate
```

5 Discussions

5.1 Memory Allocation

To compare how much memory allocation each application need, following figure is provided:

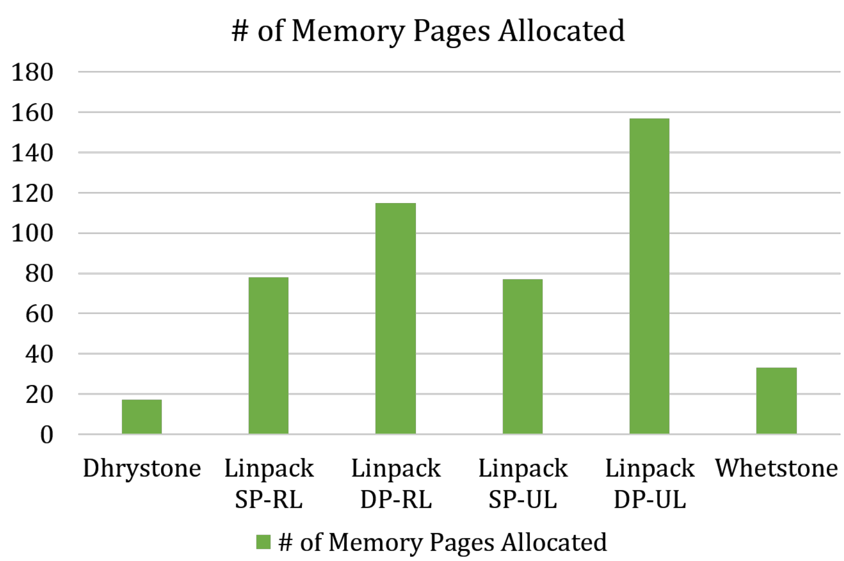


Figure 5.1: Comparison of number of memory pages being allocated in different applications.

Since Linpack DP-UL has to store four times the 64-bit double precision value because of an un-rolled loop, it has the maximum memory being allocated while rolled loop DP-RL does not require much memory for single iteration.

5.2 Number of Instructions

Number of instruction for all these application is given in the figure below: When a loop is unrolled, the number control instructions reduce and therefore there is a decrease in overall number of instruction from DP-RL to DP-UL.

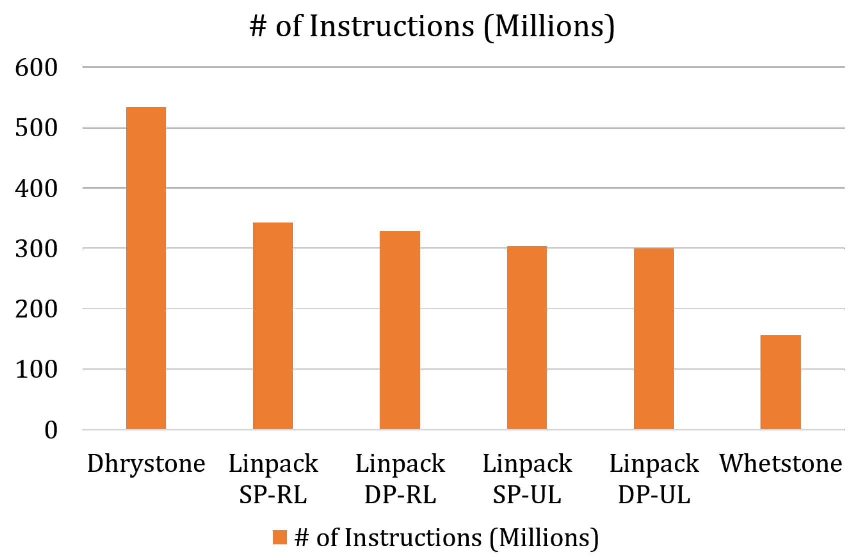


Figure 5.2: Comparison of number of instructions in different applications.

5.3 Memory bandwidth

Percentage of load/stores in different applications according to formula $\% \, Of \, Load/Store = \frac{sim\_num\_refs}{sim\_num\_insn} \times 100\%$ is given in the figure below:

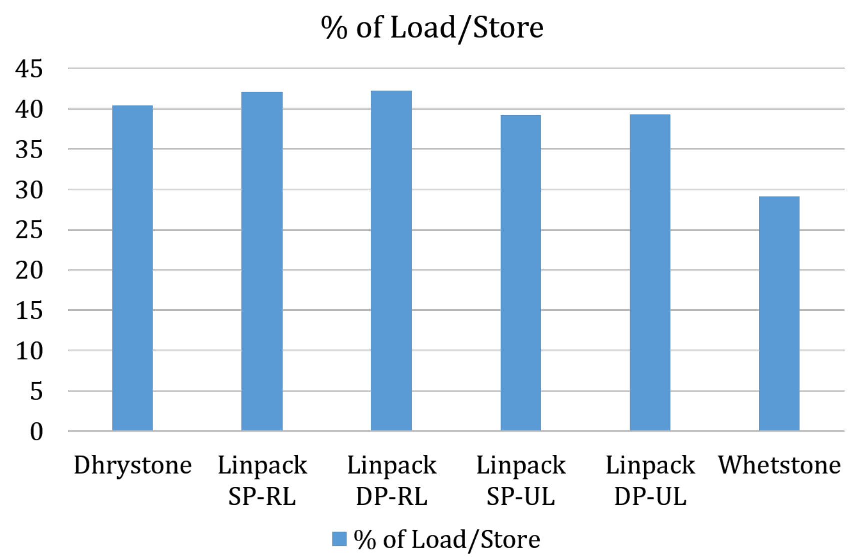


Figure 5.3: Comparison of Load/Store instructions in different applications.

5.4 PC vs Simulator Speed

CPU speedup is calculated as:

$$\%Speed - Up_{CPU} = \frac{t_{simulator}}{t_{CPU}} \tag{5.1}$$

For different applications, the speedup is given in the figure below:

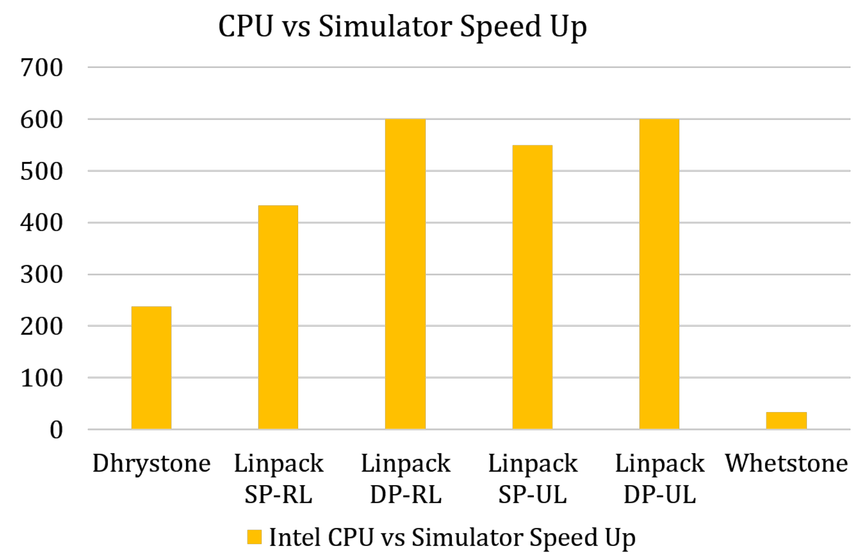


Figure 5.4: Comparison of CPU speedup in different applications.

6 Conclusion

1. Intel CPUs are more efficient in handling floating point and double precision variables as it can be seen from the speedup figure that the most speed-up is while performing double precession operations on an unrolled loop.
2. Since the CPU is multi-core, some advantage comes from intelligently scheduling the instructions for parallel processing on multiple CPU cores.
3. Overall on average, CPU proved to be about 400 times more faster than the simulator.
4. Unrolling the loop helps significantly in reducing the memory bandwidth with the help of burst data request using AXI or AMBA or similar bus protocol.

References

[1] S. LLC. (2001, December) Simplescalar tutorial slides. SimpleScalar LLC. [Online]. Available: http://www.simplescalar.com/docs/simple_tutorial_v4.pdf

[2] D. B. T. M. Austin. (1997) The simplescalar tool set, version 2.0. SimpleScalar LLC. 2395 Timbercrest Court, Ann Arbor, MI 48105. [Online]. Available: http://www.simplescalar.com/docs/users_guide_v2.pdf

[3] J. J. Dongarra, P. Luszczek, and A. Petitet. (2001, December) The linpack benchmark: Past, present, and future. Online. Netlib Repository at UTK and ORNL. [Online]. Available: http://www.netlib.org/utk/people/JackDongarra/PAPERS/hpl.pdf