LINUX on System z:
Explore Decimal Floating Point support in Linux on System z

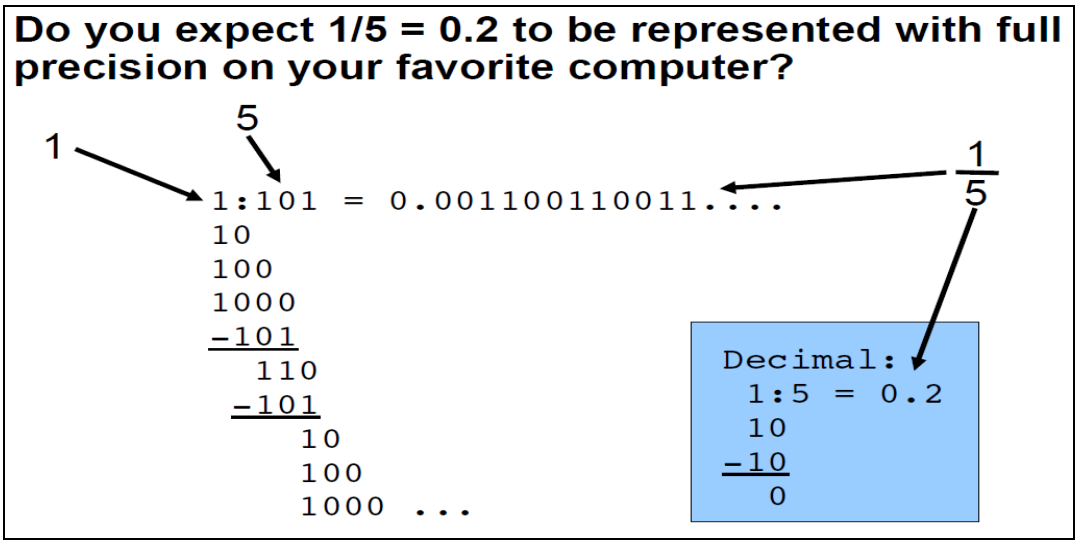
Version 2.1

March 2012

Mario Held, IBM Research & Development, Mario.Held@de.ibm.com
Andreas Krebbel, IBM Research & Development, Andreas.Krebbel@de.ibm.com

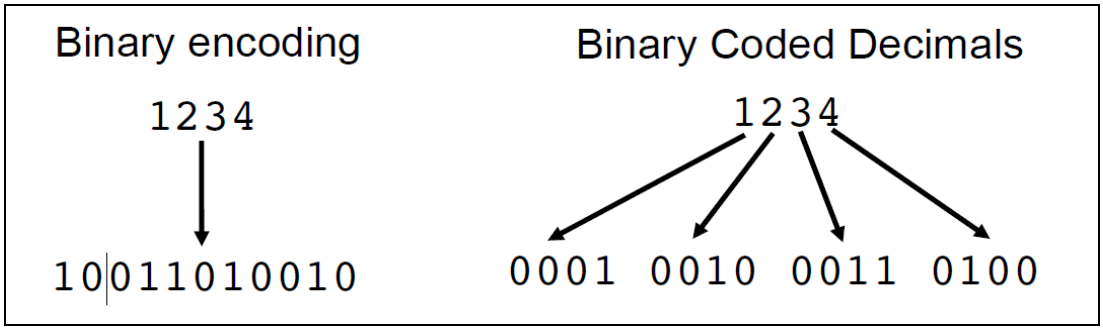
Motivation to use Decimal Floating Point (DFP)

We have had integer and floating point data types in computer languages like C, C++ and FORTRAN to store numerical values. Unfortunately binary floating point has issues to store the values with the accuracy the user expects. Even worse, when making many calculations the inaccuracy builds up.



More examples are shown here http://speleotrove.com/decimal/decifaq1.html#inexact

So for a long time, financial, commercial and user-centric applications and web services have been using binary coded decimal (BCD) format. Beside the big advantage of keeping accuracy in financial calculations, BCD also has some disadvantages. The representation of BCD values is similar to strings and every digit is encoded separately. Storing and operations on data consume slightly more resources. Only fix point arithmetic is supported and the operands have to reside in memory, not in registers.



Decimal floating point (DFP) was introduced to improve the situation and to combine the advantages of binary floating point and binary coded decimal. The term DFP covers a decimal (base 10) encoding of numeric values as well as operations on the numbers.

Advantages and Disadvantages of DFP:

- **Offers the required accuracy for financial, commercial and user-centric applications**
- Uses Densely Packed Decimal encoding to save memory.
- Hardware support for DFP has been available with the System z9-EC and newer machines. DFP operations run much faster when supported by hardware compared to the software solution.
- Exploits a new industry standard for calculations (754-2008 IEEE Standard for Floating-Point Arithmetic) which makes it easier to communicate and interchange data between various programs and software products hosted on various computer systems.
- Calculations using DFP format and operations need slightly more resources than using binary floating data (memory footprint, complexity of operations).

Prerequisites:

Software support requirements (mandatory):

SLES11 SP1 was the first available SUSE Linux Enterprise Server distribution which includes the needed softwares packages at the required level. In-between the more mature SLES11 SP2 dfp support is public available. RHEL 6 was the first Red Hat distribution which includes DFP support on the mainframe. The latest public available updates are RHEL6.1 and RHEL6.2

- **GCC** – system GNU C Compiler with integrated DFP support

  SLES11 SP2 gcc-4.3-62.198

  RHEL6.2 gcc-4.4.6-3.el6.s390x

- **libdfp** - library containing support routines for calculating and printing decimal floating point values. Provides hardware-accelerated overrides of software DFP functions. A hardware and a software version will be installed in order to choose dynamically. libdfp is maintained by IBM as external library. Includes limited software math library support.

  SLES11 SP2 libdfp1-1.0.7-0.7.9

  RHEL6.2 libdfp-1.0.6-2.el6.s390x

- **glibc** - GNU C Library provides the most important standard libraries used by nearly all programs: the standard C library, the standard math library, and the POSIX thread library. A recent glibc is needed to implement printf support externally.

  SLES11 SP2 glibc-2.11.3-17.31.1

  RHEL6.2 glibc-2.12-1.47.el6.s390x

- **binutils** - C compiler utilities: ar, as, gprof, ld, nm, objcopy, objdump, ranlib, size, string and strip. A recent bintutils version is needed to support the DFP instruction set.

  SLES11 SP2 binutils-2.21.1-0.7.25

  RHEL6.2 binutils-2.20.51.0.2-5.28.el6.s390x

- **gdb** – the GNU debugger to debug applications including DFP instructions

  SLES11 SP2 gdb-7.3-0.8.4

  RHEL6.2 gdb-7.2-50.el6.s390x

Hardware support requirements (recommended for best performance):

Hardware support is available with System z9-ec (System z9 GA3), System z10 and IBM zEnterprise 196 (z196).

Quick check for hardware support on System z (example on z196, looks similar on z10 and z9-ec):

```
[root@r35lp53 ~]# cat /proc/cpuinfo
vendor_id       : IBM/S390
# processors    : 1
bogomips per cpu: 9398.00
features        : esan3 zarch stfle msa ldisp eimm dfp etf3eh highgprs
processor 0: version = 00,  identification = 389F25,  machine = 2817
[root@r35lp53 ~]#
```

Usage:

Language C:

- The C standard includes the following new data types (these are native data types just like integer, long and float, double etc): _Decimal32,_Decimal64,_Decimal128
- New constant suffixes: df, dd, dl, DF, DD and DL
- Arithmetic operations (add, subtract, multiply, divide, sqrt, … sinus...)
- String to DFP conversions: strtod32, strtod64, strtod128
- Set and get rounding mode functions (fe_dec_setround and fe_dec_getround)
- New printf format letters: %Hf, %Df, %Ddf
- Function scanf is not yet supported

Build the code:

- **Preferred for compatibility:** Building a DFP application which should run also on machines prior to z9 GA3 (without hardware DFP support): **gcc -O3 -ldfp …** On a machine supporting hardware DFP, the software functions will dynamically be replaced by hardware versions.
- **Fastest:** gcc -O3 **-march=<z9-ec | z10 | z196> -ldfp …** Building a binary exploiting hardware DFP application which is only able to run on z9-ec (GA3) or z10. Usage of new instructions and special scheduling for the target machine is used. It is not necessary to specify **-mhard-dfp** because it is the default.
- **Enforce software dfp:** gcc -O3 **-mno-hard-dfp …** The -mno-hard-dfp switch can be used to to make GCC fall back to software DFP support even with -march=z9-ec or higher. This combination of options seems more relevant in test scenarios.

Usage with Java:

The java.math.BigDecimal class provides software DFP support. New Decimal Floating Point data types are supported. The Java support has been available since Java 1.5. Because of performance improvements and lots of fixes in releases and service packs, the use of the latest package is always recommended.

Test methods for DFP:

There are several sources to find both accuracy and performance tests for DFP in the Internet.

Accuracy Test:

Various test suites are available. For example

General Decimal Arithmetic test-cases Version 2.44 – 7 Apr 2009

and

Floating-Point Test Generator – Fpgen can be downloaded from the Internet.

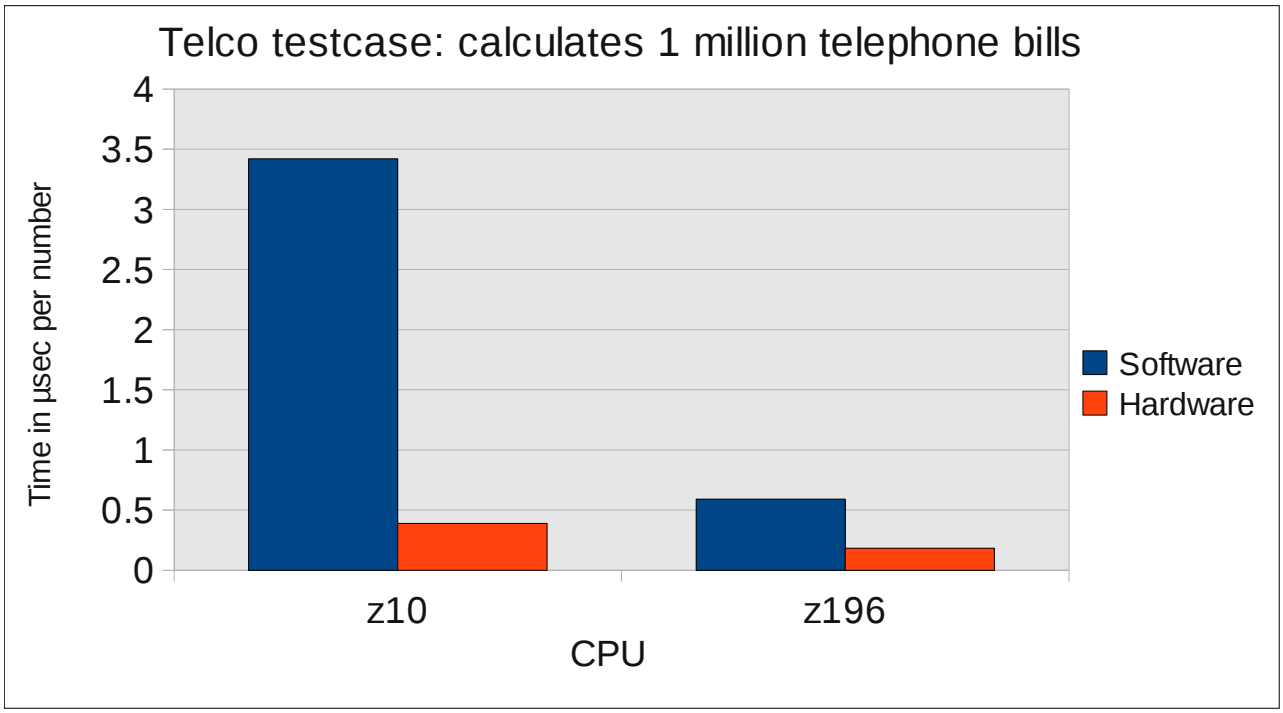
Performance Test:

There is a collection of decimal floating benchmarks downloadable from the Internet at the University of Wisconsin: Decimal Floating-Point (DFP) Benchmarks, Download Version 1.0

The suite includes the following applications: Banking, Tax Solver, Telco, Risk Management and Currency Conversion. These benchmarks do not run out of the box on System z and have to be adapted for use with GCC and System z hardware. Also the input data has to be converted from little endian to big endian format.

First tests were completed using one benchmark out of this collection, the Telco benchmark. This test was carried out on Linux on System z10 and z196. The benchmark is separately available here The ‘telco’ benchmark.

The Telco benchmark and testcase models a telephone company's billing system. The workload comprises billing of one million telephone calls including tax using DFP arithmetics. If DFP hardware support is exploited, the performance improves significantly. The test giving us the following results was carried out using an IBM internal driver on Linux on System z running on a z10 and z196. **The advantage of the hardware implementation can be seen compared with the software solution running on the same system in the following chart.**



The Java variant of the Telco benchmark (jTelco) was tested on SLES11 SP1 using the Java package ibm-java-s390x-sdk-6.0-9.2.s390x.rpm (IBM J9 VM (build 2.4, JRE 1.6.0 IBM J9 2.4 Linux s390x-64 jvmxz6460sr9-20110624_85526 (JIT enabled, AOT enabled))on a System z196. It works as expected.

```
r37lp06:/home/dfp # export PATH=/opt/ibm/java-s390x-60/bin/:$PATH
r37lp06:/home/dfp # export CLASSPATH=.:/opt/ibm/java-s390x-60/

r37lp06:/home/dfp # javac jTelco.java

r37lp06:/home/dfp # java jTelco expon180.1e6b
telco Java benchmark; processing 'expon180.1e6b'
-- telco Java benchmark result --
   1000000 numbers read from 'expon180.1e6b'
   Time per number       1.538us  [1000000]
--
   sumT = 1004737.58
   sumB = 57628.30
   sumD = 25042.17
```

Further reading and DFP resources:

754-2008 IEEE Standard for Floating-Point Arithmetic

General Decimal Arithmetic by Mike Cowlishaw

C language extension TR24732 ratified extension

C++ language extension TR 24733 ratified extension:

IBM Research & Development GmbH
Schoenaicher Str. 220
71032 Boeblingen
Germany
Produced in Germany,
03/2012

All statements regarding IBM’s future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Performance is based on measurements and projections using benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user’s job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance results stated here.

