# Performance Optimization for Modern z Processors

*C. Kevin Shum*
*IBM Distinguished Engineer*
*z Systems Processor Design*
*Member of IBM Academy of Technology*

*Charles F. Webb*
*IBM Fellow*
*z Systems Development*

# Trademarks

**The following are trademarks of the International Business Machines Corporation in the United States and/or other countries.**

| | | | | | | |
|---|---|---|---|---|---|---|
| BigInsights | DFSMSdss | FICON* | IMS | RACF* | System z10* | zEnterprise* |
| BlueMix | DFSMShsm | GDPS* | Language Environment* | Rational* | Tivoli* | z/OS* |
| CICS* | DFSORT | HyperSwap | MQSeries* | Redbooks* | UrbanCode | zSecure |
| COGNOS* | DS6000* | IBM* | Parallel Sysplex* | REXX | WebSphere* | z Systems |
| DB2* | DS8000* | IBM (logo)* | PartnerWorld* | SmartCloud* | z13 | z/VM* |
| DFSMSdfp | | | | | | |

* Registered trademarks of IBM Corporation

**The following are trademarks or registered trademarks of other companies.**

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.
Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.
Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.
IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.
ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.
Java and all Java based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.
Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and
Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
OpenStack is a trademark of OpenStack LLC. The OpenStack trademark policy is available on the OpenStack website.
TEALEAF is a registered trademark of Tealeaf, an IBM Company.
Windows Server and the Windows logo are trademarks of the Microsoft group of countries.
Worklight is a trademark or registered trademark of Worklight, an IBM Company.
UNIX is a registered trademark of The Open Group in the United States and other countries.
* Other product and service names might be trademarks of IBM or other companies.

**Notes**:

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM 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 ratios stated here.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

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

Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

This information provides only general descriptions of the types and portions of workloads that are eligible for execution on Specialty Engines (e.g, zIIPs, zAAPs, and IFLs) ("SEs"). IBM authorizes customers to use IBM SE only to execute the processing of Eligible Workloads of specific Programs expressly authorized by IBM as specified in the “Authorized Use Table for IBM Machines” provided at www.ibm.com/systems/support/machine_warranties/machine_code/aut.html (“AUT”). No other workload processing is authorized for execution on an SE. IBM offers SE at a lower price than General Processors/Central Processors because customers are authorized to use SEs only to process certain types and/or amounts of workloads as specified by IBM in the AUT.

# Introduction / Motivation

- **Hardware/Software co-optimization is increasingly important to performance**
  - Performance gains from technology scaling have ended
  - Hardware performance gains are coming from design
    - Micro-architectural innovation (and complexity)
    - New instructions and architected features
  - Coding practices and software exploitation needed to get the full value of the hardware
- **More efficient code helps everybody**
  - Increases value of software
    - Extract the maximum useful work from the hardware
  - Increases value of z Systems platform
    - Solutions delivered more cost-effectively
  - Decreases effective cost for end user
- **Goal of this session: Motivate you to make performance a priority**
  - Can only scratch the surface in 45 minutes
  - Highlight a few high-leverage areas
  - Point you to resources available to assist with optimization

# Compilers

- **Biggest single performance lever for many applications**
  - Aggressive use of the latest compiler technology
- **Close Linkage between compiler and hardware development teams**
  - Define new instructions and architectural features
  - Tune code generation for processor micro-architecture
- **ARCH and TUNE options optimize for current hardware designs**
  - ARCH needs to match oldest hardware level supported
    - May be worth experimenting to test value of higher ARCH level on new hardware
  - TUNE should match hardware level for which you care most about performance
    - Usually the latest available hardware level
    - Code will work correctly on all hardware levels
- **Use higher levels of OPT to get the best performance:**
  - At least on performance-sensitive components

# Compilers on z systems

- **IBM continues to invest in the compiler portfolio on z:**
  - Increased focus on application program performance in recent years
  - Continued advancements in languages and operating systems
    - Java / JIT, C/C++, COBOL, PL/I, Linux, z/OS

## Enterprise COBOL for z/OS V5.2

- Leverage SIMD instructions to improve processing of certain COBOL statements.
- Increased use of DFP instructions for Packed Decimal data
- Support COBOL 2002 language features: SORT and table SORT statements
- Allows applications to access new z/OS JSON services

***Up to 14% reduction in CPU time***

## Enterprise PL/I for z/OS V4.5

- Critical Business Language – Committed to invest in leading-edge technology
- Shipped a new release every year since 1999
- Fully Supports z/Architecture, including z13 & z13s processors
- Provide full support for JSON (Parse, Generate, and Validate)

***Up to 17% reduction in CPU time***

## z/OS V2.2 XL C/C++

- Optional feature of z/OS 2.2
- Provides system programming capabilities with Metal C option
- Fully Supports z/Architecture, including z13 & z13s processors
- Ships with High performance Math Libraries tuned for z13

***Up to 24% increase in throughput***

## XL C/C++ for Linux on z Systems V1.2

- New compiler based on Clang and IBM optimization technology
- Fully Supports z/Architecture, including z13 & z13s processors
- Provide easy migration of C/C++ applications to System z

***Up to 14% increase in performance over GCC***

*** The performance improvements are based on internal IBM lab measurements. Performance results for specific applications will vary, depending on the source code, the compiler options specified, and other factors**

# Evolution of IBM COBOL on z Systems

**Day 1 z Processer Support**
**Rel-Rel Performance Improvement**
**New COBOL Language Features**
**App. Modernization Features**

**Application Modernization**
**Middleware Interoperability**
**Internationalization**

**LE, Debug, USS…**

**COBOL/370, COBOL for MVS & VM; COBOL for OS/390 & VM (Ann: 1990's)**
- Language Environment
- Intrinsic functions
- Debug Tool
- Dynamic Libraries, USS, DB2 coprocessor…

**Enterprise COBOL V3 (Ann: 2001-2005)**
- Unicode
- Native Java & XML
- CICS & DB2 co-processors; IMS Java regions
- Debugging of production code with Debug Tool
- Data item limits raised to 128MB (from 16 MB)

**Enterprise COBOL V4 (Ann: 2007-2009)**
- XML System Services parser
- DB2 9 SQL support with coprocessor
- Java 5 & 6 support;
- UNICODE performance improvement
- Improve debug support for optimized code

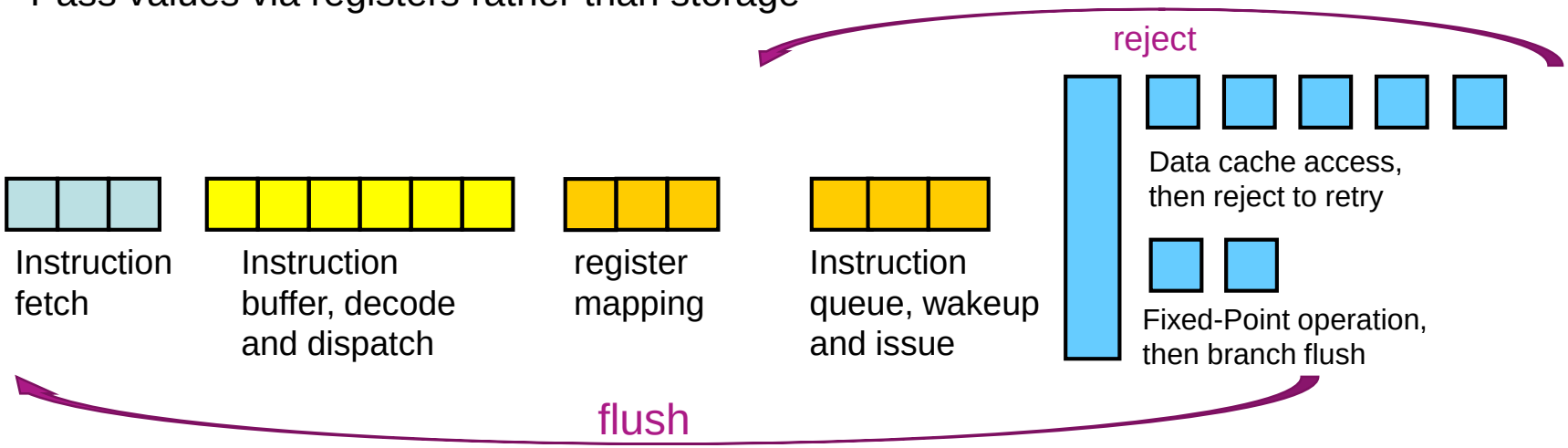
**Enterprise COBOL V5 (Ann: 2013)**
- ***New advanced Optimization Framework***
- New COBOL Runtime
- DWARF Debugging format
- Exploits Program Object
- New COBOL 2002 Language Features
- Generates SMF 89

**Enterprise COBOL V6 (Ann: 2016)**
- Enhanced Scalability – Compile and Optimize very large COBOL programs
- Native JSON "Generate"
- New COBOL 2002 Language Features
- Enhanced Migration Help

# Why SW Optimization Matters

## Processor design

- **Deep instruction pipeline**
  - Driven by high-frequency design
  - Z13 pipeline: 20+ cycles from instruction fetch to instruction finish
- **Pipeline hazards can be expensive**
  - Branch flush – 20+ cycles
  - Cache reject – 12+ cycles
- **Code optimization can help**
  - Arrange frequent code in “fall through” paths
  - Pass values via registers rather than storage

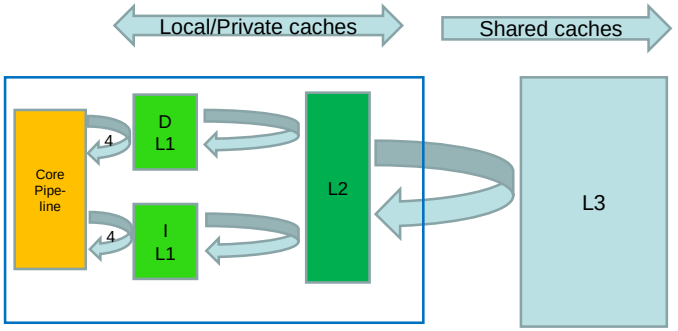


# Why SW Optimization Matters

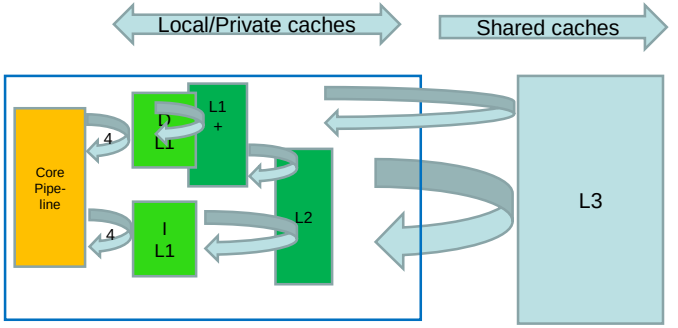
## Cache design

- **Private (per-core) cache evolution**
  - – Allows improvements in size and latency
  - – Unified vs. split L2 for instructions and operands
    - Split L2 keeps data closer to L1
    - Unified (z196) to hybrid (zEC12) to split (z13)
  - – Integrated vs. serial directory lookup
    - Integrated reduces access latency for L2, L3
    - Added for operands (zEC12), Instructions (z13)
- **Allows large, fast L2 caches**
  - – L2 sizes comparable to others' L3s (MBs)
    - Leverages eDRAM technology
  - – Around 10 cycles to access data from L2
- **On-chip shared L3**
  - – Shared by all cores on the CP chip
  - – Now also the sharing point for I-L2 and D-L2
- **Cache line size is 256B throughout hierarchy**
  - – Safe value to use for separation / alignment

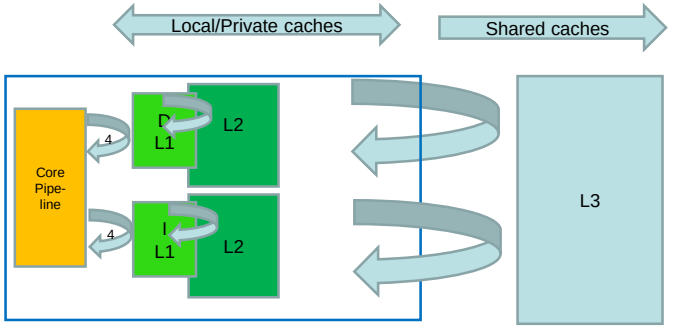
z196 On-chip Cache Hierarchy




zEC12 On-chip Cache Hierarchy




z13 On-chip Cache Hierarchy

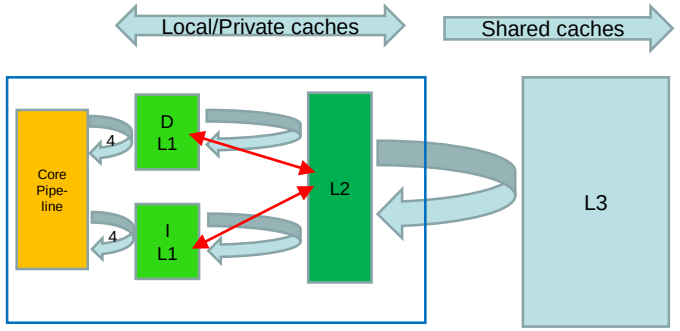


# Optimizations on local data

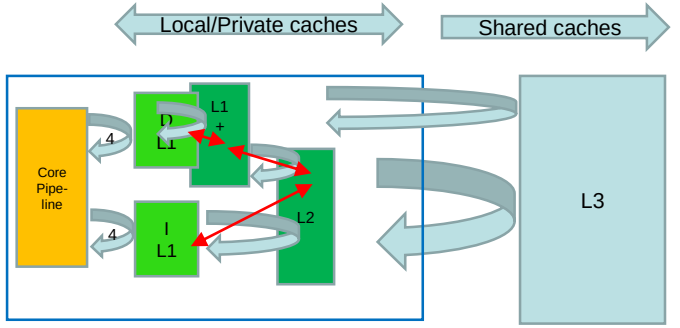
## Instruction / data proximity

- **Instructions & Operands in same cache line**
  - OK (maybe inefficient) if operands read-only
  - Problem if stores to those operand locations
    - Extra cache misses, long delays
- **Split L1 caches (re-)introduced in z900 (2000)**
  - Designs optimized for well-behaved code
    - Increasing cost of I/D cache contention
  - With split of L2 cache, resolution moved to L3
- **Not a problem for**
  - Re-entrant code
  - Any LE-based compiler generated code
  - Dynamic run-time code
- **Problematic Examples**
  - True self-modifying code
  - Classic save area
  - Local save of return address
  - In-line macro parameters
  - Local working area right after code

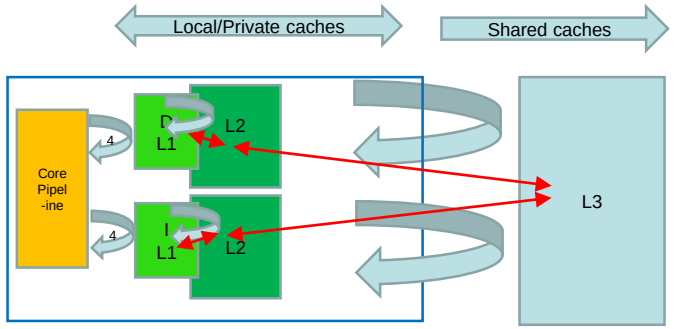
z196 On-chip Cache Hierarchy




zEC12 On-chip Cache Hierarchy




z13 On-chip Cache Hierarchy

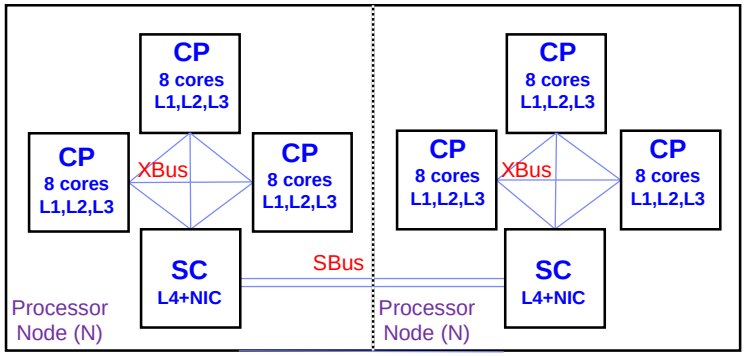


# Optimizations on shared data

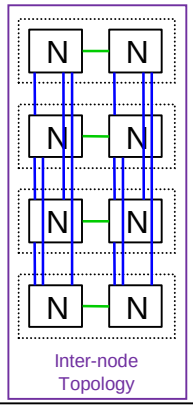
## Shared data structures among SW threads / processes

- **Sharing is not necessarily bad**
  - Can be very useful to leverage strongly consistent architecture
- **…But updates from multiple cores => lines bounce around among caches**
  - Depending on locations of cores, added access latency can be troublesome
  - Need to manage well to get good performance
- **True sharing – real-time sharing among multiple SW threads / processes**
  - Atomic updates, Software locks
  - Higher nWay (concurrent SW threads), more frequent access => more care needed
  - If contested in real-time, can lead to “hot-cache-line” situations
- **False sharing – structures / elements in same cache line**
  - Can be avoided by separating structures into different cache lines

| Cache hit locations | Latencies (no queuing) | Intervention Overhead (if a core owes exclusive) |
|---|---|---|
| L1 | 4 | NA |
| L2 | ~10 | NA |
| L3 (on-chip) | 35+ | 40+ |
| L3 (on-node) | 180+ | 20+ |
| L3 (off-drawer, Far column) | 700+ | 20+ |




Cache topology and latencies for z13



# Moving / Clearing Large Blocks

## Usages of MOVE LONG (MVCL) vs MOVE (MVC) instructions

- **Several ways to move or clear a large block of storage**
  - One MVCL instruction
  - Loops of MVCs to move data
  - Loops of MVC <Len>,<Addr>+1,<Addr> or XC <Len>,<Addr>,<Addr> to pad/clear an area
- **MVCL is implemented through millicode routines**
  - Millicode is a firmware layer in the form of vertical microcode
    - Incurs some overhead in startup, boundary/exception checking, and ending
  - MVCL function implemented using loops of MVCs or XCs
- **Millicode has access to special hardware**
  - Near-memory engines that can do page-aligned move and page-aligned padding
    - Can be faster than dragging cache lines through the cache hierarchy
    - However, the destination will NOT be in the local cache
- **Many factors to consider**
  - Will the target be needed in local cache soon?
    - Then moving “locally” will be better
  - Is the source in local cache?
    - Then moving “locally” may be better
  - How much data is being processed?
    - If many pages, then the near-memory engine usage might be beneficial

# Software Aids to Hardware

## Hardware cannot read programmers' minds: Give it some hints

- **Instructions designed to help hardware optimize performance**
  - Modify details of heuristic / history-based hardware mechanisms
  - Please use responsibly: Over- or mis-use can be counter-productive
    - Increased code image, pathlength
    - One wrong hint can outweigh several correct ones
  - Some experimentation may be needed to fine-tune usage
  - Exact hardware effects will vary by implementation
    - Hardware reserves the right to ignore hints

- **Branch Prediction Preload [Relative] (BPP, BPRP) Instruction**
  - Introduced on zEC12
  - Specifies future branch instruction and its target
    - Target address in GR or relative to current instruction address
  - Performs instruction cache touch of the provided branch target address
  - Architectural no-op

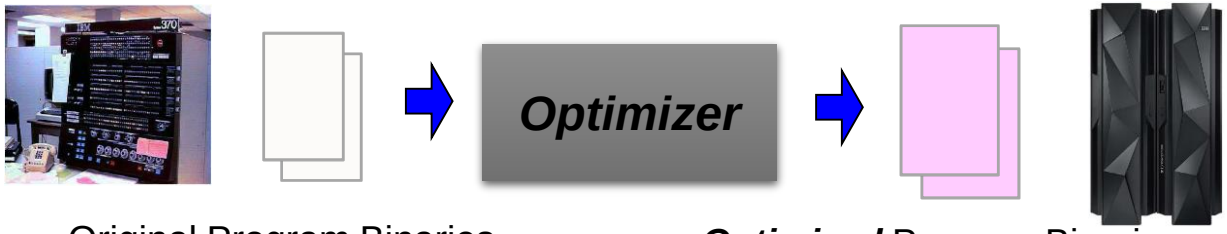
# Software Aids to Hardware (continued)

- **Next Instruction Access Intent (NIAI) instruction**
  - Introduced on zEC12
  - Affects hardware handling of the storage operand of the next instruction
    - Like a “prefix” instruction but architecturally a separate (no-op) instruction
    - Especially useful when referencing shared storage areas / data structures
    - May be used by MVCL millicode to optimize use of near-memory engines
  - “Read”: This program will only read – not write/change – that location / cache line
  - “Write”: This program will be updating the location / cache line later
    - Even though this access is a read/ load
  - “Use once”: This program will not be using this location again
    - Can indicate that the current access is a streaming type access
- **Prefetch Data [Relative] (PFD, PFDRL) instruction**
  - Introduced on z10
  - Helps hardware have the right stuff in the caches when needed
  - Pre-stage cache lines into the local caches (all the way into L1)
    - Specify whether intended usage is read-only or read/write
  - “Untouch” cache lines to remove from local caches
    - Can be helpful when done using a shared data structure
  - Demoting cache line from an exclusive state to a read-only state
    - Can be helpful when done updating a shared data structure
  - Architectural no-op

# IBM Automatic Binary Optimizer (ABO) for z/OS

## Improve Performance of Compiled COBOL Programs

| ABO Features | |
|---|---|
| Internal & Customer Performance Improvements Measuring ~**15%** | ✓ |
| **No Source Code, Migration or Performance Options Tuning Required** | ✓ |
| Targets Latest IBM z Systems : **zEC12, zBC12, z13,z13s** running **z/OS 2.1** or **z/OS 2.2** | ✓ |
| All IBM Enterprise COBOL **v3 & v4** Compiled Programs Are Eligible For Optimization | ✓ |
| Optimized Programs Guaranteed To Be **Functionally Equivalent** | ✓ |
| IBM Problem Determination **Tooling Support**<br>+ Working With Several Key 3rd Party Tooling Vendors In Our Beta Program | ✓ |
| Leverages new z/OS 2.2 Infrastructure To **Target Multiple Hardware Levels Automatically** | ✓ |




Original Program Binaries
(Base ESA390)

***Optimized*** Program Binaries
(Latest z Systems)

# Other Resources

## Like this stuff? There's lots more available:

- Microprocessor Optimization Primer
  - Available under IBM Developerworks' LinuxOne community
    - https://www.ibm.com/developerworks/community/groups/community/lozopensource

- **CPU Measurement Facilities**
  - User-accessible hardware instrumentation data to understand performance characteristics
  - Documentation and education materials can be found on online, some references:
    - For z/OS http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/TC000066 (supported under Hardware Instrumentation Services - HIS)
    - For z/VM http://www.vm.ibm.com/perf/tips/cpumf.html

- **Other related references**
  - "z/Architecture: Principles of operation," Int. Bus. Mach. (IBM) Corp., Armonk, NY, USA, Order No. SA22-7832-10, Feb. 2015. [Online]
  - Dan Greiner's presentations of z/Architecture features with SHARE
  - John R. Ehrman's book: Assembler Language Programming for IBM z System Servers
  - "The IBM z13 multithreaded microprocessor," in IBM J. Res. & Dev., pp. 1:1–1:13, 2015

# Thank you!

**(Chung-Lung) Kevin Shum**
cshum@us.ibm.com
**Linkedin: https://www.linkedin.com/in/ckevinshum**

**Charles Webb**
**cfw@us.ibm.com**