Simultaneous MultiThreading (SMT)

•The z13, aligned with multithreading industry directions, can process up two simultaneous threads in a single core.
- This capability is known in z Systems as simultaneous multithreading (SMT) and offers full z/Architecture capability for each thread.
  NOTE: SMT is supported *only* by zIIP and IFL speciality engines on z13.

• An operating system with SMT support can be configured to dispatch work to a thread on a zIIP (for eligible workloads in z/OS) or an IFL (for z/VM on behalf of Linux on z System guest or native Linux on z Systems) core in single thread or SMT mode.

• The use of SMT provides a better and more efficient use of the processor resources and helps address memory latency, resulting in overall throughput gains.
  NOTE: SMT is designed to deliver better overall throughput for many workloads, but not all workloads, where cases may be superior *using single threading*.

• Each software Operating System / Hypervisor has the ability to intelligently drive SMT in a way that is best for its unique requirements.
- z/OS SMT management consistently drives the cores to high thread density, in an effort to reduce SMT variability and deliver repeatable performance across varying CPU utilization thus providing more predictable SMT capacity.
  > z/VM SMT management optimizes throughput by spreading a workload over the available cores until it demands the additional SMT capacity.

How SMT benefits performance - HiperDispatch is required if SMT is enabled.

• When a program accesses a memory location that is not in the cache, it is called a *cache miss*.
- Because the processor then must wait for the data to be fetched from the next cache level, or from main memory before it can continue to execute, cache misses directly influence the performance and capacity of the core to execute instructions.

• With simultaneous multithreading exploitation, when one thread in the core is waiting, for example, for data to be fetched from the next cache levels or from main memory, the second thread in core can utilize the shared resources rather than remain idle.

z/OS SMT management consistently drives the cores to high thread density, in an effort to reduce SMT variability and deliver repeatable performance across varying CPU utilization - thus providing more predictable SMT capacity.

z/OS V2.2 is planned to support the use of SMT for zIIP processors.

• To support this new function, z/OS is planned to provide:
  – IPL-time controls to enable SMT and set the SMT mode
  – Post-IPL controls to dynamically switch the SMT mode
  – SMF Type 30 record fields with a normalized value for CPU time spent on a processor running in SMT mode
  – SMF Type 70 records with new SMT-related fields
  – Parallel Sysplex services (XES) use of SMT mode for workloads using zIIPs to help improve physical processor utilization for synchronous requests
  – Hardware Instrumentation Services (HIS) updates to provide measurement data in SMT mode
  – New RMF metrics to help you with capacity planning and performance analysis.

To underpin SMT, the z13 has a double symmetric instruction pipeline width and full architectural state per thread.

• Beyond this, the CPU address changes and the 16-bit CPU ID consist of 15-bit core ID and a 1-bit thread ID. For example, the CPU ID 6 (b'0000000000000110') means core 3 thread 0 and the CPU ID 7(b'0000000000000111') means core 3 thread 1. For CPs, only thread 0 is used in each core.

Any given pipleline stage Simultaneous multithreading technology allows instructions from more than one thread to run in any given pipeline stage at a time.

• Each thread has its own unique state information, such as PSW and registers.

• The simultaneous threads cannot necessarily run instructions instantly and must at times compete to use certain core resources that are shared between the threads.

• In some cases, threads can use shared resources that are not experiencing competition.

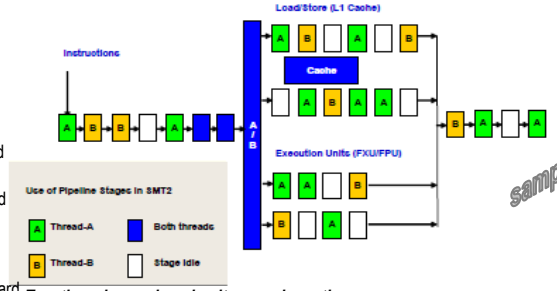
• The use of SMT provides more efficient use of the processors' resources and helps address memory latency, resulting in overall throughput gains.
- The active thread shares core resources in space, such as data and instruction caches, TLBs, branch history tables, and in time, such as pipeline slots, execution units and address translators.

• Although SMT increases the processing capacity, the performance in some cases might be superior if you use a single thread.

• Enhanced hardware monitoring support measures through CPUMF, thread usage, and capacity.
- For workloads that need maximum thread speed, the partition's SMT mode can be turned **off**.
- For workloads that need more throughput to decrease the dispatch queue size, the partition's SMT mode can be turned **on**.

• The SMT exploitation is functionally transparent to middleware and applications, and no changes are required to run them in an SMT-enabled partition.

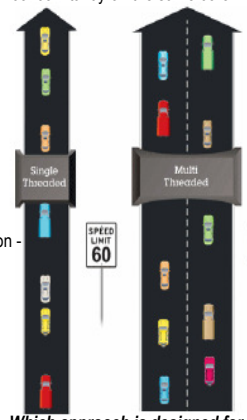
• The Monitors Dashboard on the HMC and SE are enhanced with an adapter table.
  NOTE: You can now display the activity for an Logical Partition (LPAR) by processor type and the Monitors Dashboard is enhanced with showing simultaneous multithreading (SMT) usage (ex: see bottom- far right diagram).



Two threads running simultaneously on the same processor core

To activate SMT function on z/OS, define the **PROCVIEW CORE** option in **LOADxx**. If you do not use the SMT-2 function, define **PROCVIEW CPU**. The default parameter of **PROCVIEW** is **CPU**.
Define MT_ZIIP_MODE=2 in IEAOPTxx to execute multiple threads on zIIP SMT mode.
Note: When you activate SMT2 function, the CPU address changes. A 16-bit CPU ID consists of a 15-bit Core ID and 1-bit Thread ID.

Simultaneous multithreading (SMT): Two simultaneous threads running concomitantly on the same core.



Which approach is designed for the highest volume of traffic?
Which road is faster?

Up to 141 configurable processors or up to 128 processors per LPAR on z13 processors when running in SMT mode

The SMT is functionally transparent to middleware and applications where no changes are required to run in a SMT partition.

Application serving with SSL could see up to 2x improvement in throughput per core with IBM 64-bit SDK for z/OS, Java Technology Edition, Version 8 on IBM z13 with SMT vs. Java 7 on zEC12.

When the SMT facility is enabled, the hardware threads within a core share certain hardware resources such as execution units and caches. When the SMT is not enabled, a core executes a single hardware thread.


```
HWNAME    SCZP501
LPARNAME  A01
SYSPLEX   PLEX76   Y
IODF      ** SYS6     L06RMVS1 01 Y
SYSCAT    BH6CAT123CMCAT.BH6CAT
PARMLIB   SYS1.PARMLIB
PARMLIB   SYS1.IBM.PARMLIB
PROCVIEW CORE,CPU_OK
```

LOADxx definition sample with CORE,CPU_OK

Single-instruction, multiple-data (SIMD): Vector processing unit (one instruction is applied to a vector of data)

• z13 introduces architectural extensions with instructions that are designed to allow reduced processor quiesce effects, reduced cache misses, reduced pipeline disruption, and increased parallelism with instructions that process several operands in a single instruction, thus Single Instruction Multiple Data (SIMD).

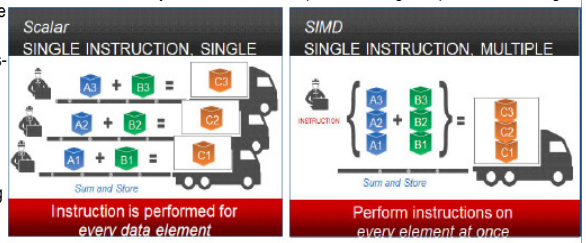
• The z13 superscalar processor has 32 vector registers and an instruction set architecture that includes a subset of 139 new instructions, known as SIMD, added to improve the efficiency of complex mathematical models and vector processing.
- These new instructions allow a larger number of operands to be processed with a single instruction.
- The SIMD instructions use the superscalar core to process operands in parallel.

• z/OS V2.2 is planned to provide support for the new vector extension facility (SIMD) instructions available on IBM z13 servers
-This new support, also planned to be available for z/OS V2.1 with PTFs intended to help enable high-performance analytics processing, and is planned to be exploited by **z/OS XML System Services**; **IBM 31-bit SDK for z/OS**, **Java Technology Edition, Version 8 (5655-DGG); IBM 64-bit SDK for z/OS, Java Technology Edition, Version 8 (5655-DGH); Enterprise PL/I for z/OS V4.5 (5655-W67); and Enterprise COBOL for z/OS 5.2 (5655-W32).**
- IBM intends to exploit the **64-bit SDK for z/OS, Java Technology Edition, Version 8 in IBM WebSphere® Liberty Profile for z/OS,** and in the *full profile* of **WebSphere Application Server for z/OS**, which is also expected to benefit from SIMD exploitation.
  NOTE: This feature will be fully used by z/OS V2R1 where several compilers have built-in functions for SMID.

• SIMD uses the enhanced superscalar z13 core to process a large number of operands (vector) through a single instruction, allowing the development of smaller and optimized codes to improve efficiency of complex mathematical models and vector processing.

• SIMD - MASS and ATLAS libraries for z/OS (and Linux on z Systems):
  – Allow construction of richer, complex analytics models using ISV/SWG Analytics products that exploit SIMD to provide better accuracy of insight
  – Allow Analytics workloads to be ported from Power and x86 with ease that can accelerate analytics to provide speedy business insight
  – Increase programmer productivity of ISV and customer analytics workload development leading to rapid business insight generation for competitive advantage

• SIMD instructions allow a larger number of operands to be processed with a single instruction.
- The SIMD instructions use the superscalar core to process operands in parallel, enabling more interactions.
- The set of SIMD instructions are a type of data parallel computing and vector processing that can decrease amount of code and accelerate code with integer, string, character and floating point data types.



Schematic comparison between a scalar instruction and a SIMD instructions

• SIMD provides the next phase of enhancements of z Systems analytics capability.
- The set of SIMD instructions are a type of data parallel computing and vector processing that can decrease the amount of code as well as accelerate code that handles integer, string, character, and floating point data types.
- The SIMD instructions improve performance of complex mathematical models and permit integration of business transactions and analytic workloads on z Systems.

Enhancements to vector processing can be used for analytics acceleration using new SIMD facilities.

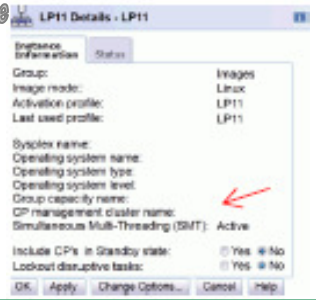
• The 32 new vector registers have 128 bits.

• The 139 new instructions include string operations, vector integer, and vector floating point operations.
- Each register contains multiple data elements of a fixed size. The instructions code specifies which data format to use and whether the size of the elements is byte (sixteen 8-bit operands), halfword (eight 16-bit operands), word(four 32-bit operands), doubleword (two 64- bit operands), or quadword (one 128-bit operand).
- The collection of elements in a register is called a *vector*.
  NOTE: A single instruction operates on all of the elements in the register. Instructions have a non-destructive operand encoding that permits the addition of the register vector A and register vector B and stores the result in the register vector A (A = A + B).

SIMD Registers

D M=CPU command with PROCVIEW CPU

```
D M=CORE
IEE174I 14.54.21 DISPLAY M 902
CORE STATUS: HD=Y   MT=2  MT_MODE: CP=1  zIIP=2
ID    ST   ID RANGE   VP   ISCM  CPU THREAD STATUS
0000   +   0000-0001  H   0000  +N
0001   +   0002-0003  H   0000  +N
0002   +   0004-0005  H   0000  +N
0003   +   0006-0007  H   FC00  +N
0004  +I   0008-0009  H   0200  ++
0005   -   000A-000B
0006   -   000C-000D
0007   -   000E-000F
0008   -   0010-0011
0009  -I   0012-0013
```

March/2015