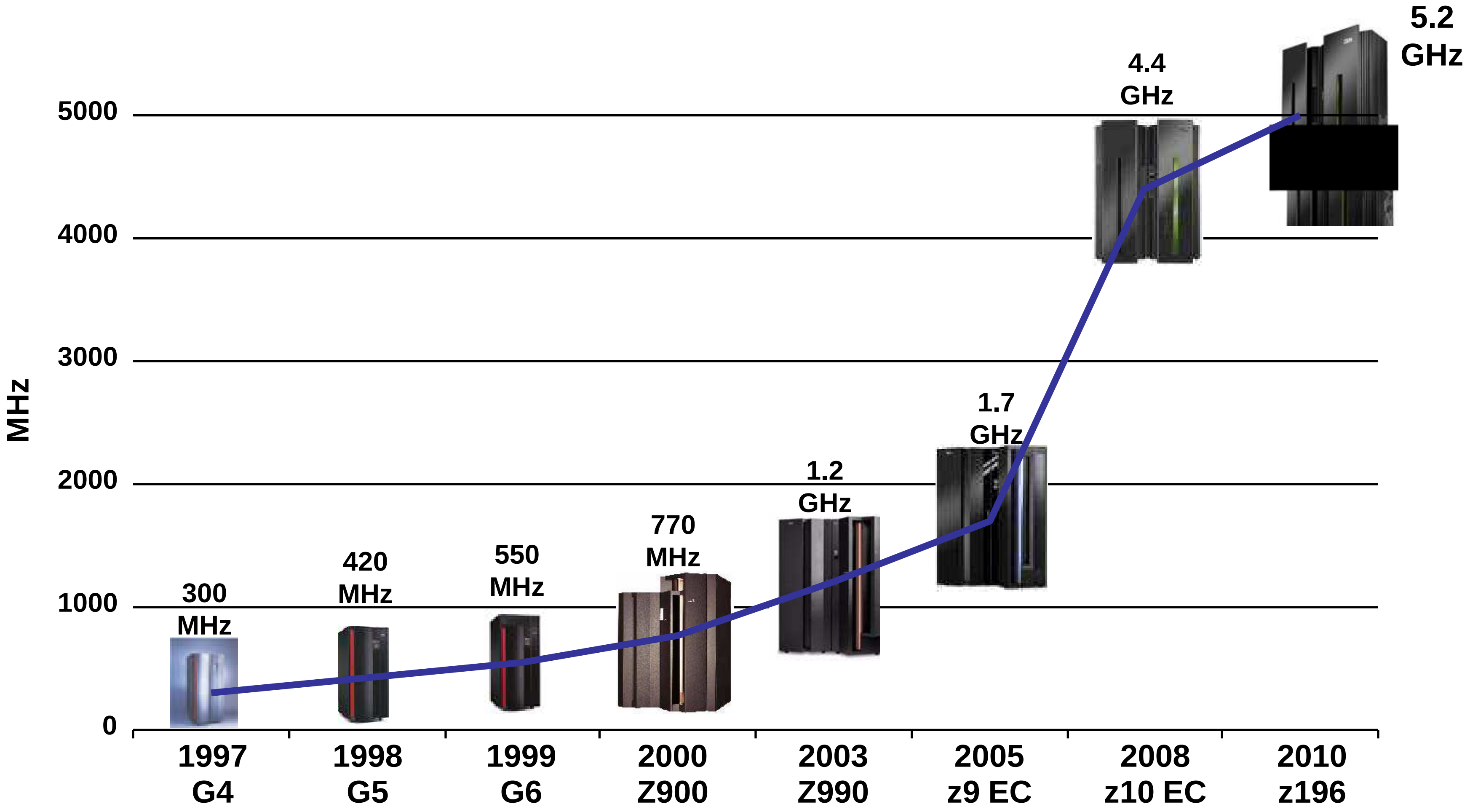
Smarter Systems for a
Smarter Planet

IBM zEnterprise 196 Processor

Brian Curran
Distinguished Engineer
System z Processor Development

IBM zEnterprise Continues the CMOS Mainframe Heritage



Scalar In-order →

1st 64b arch

super-scalar In-order →

deep pipeline

wide super-scalar, out of order

z196 Performance

[redacted]

- Unique blend of
  - Large, robust caches
  - High frequency, out-of-order execution core
- Ideal for large scale data and transaction serving and mission critical applications

  Up to 40% improvement for traditional z/OS workloads [1]

  Up to 60% higher system (50 Billion instructions / sec) capacity [1]
- Ideal for large scale Linux consolidation

  Supports thousands of Linux images
- Ideal for CPU intensive (including JAVA) applications

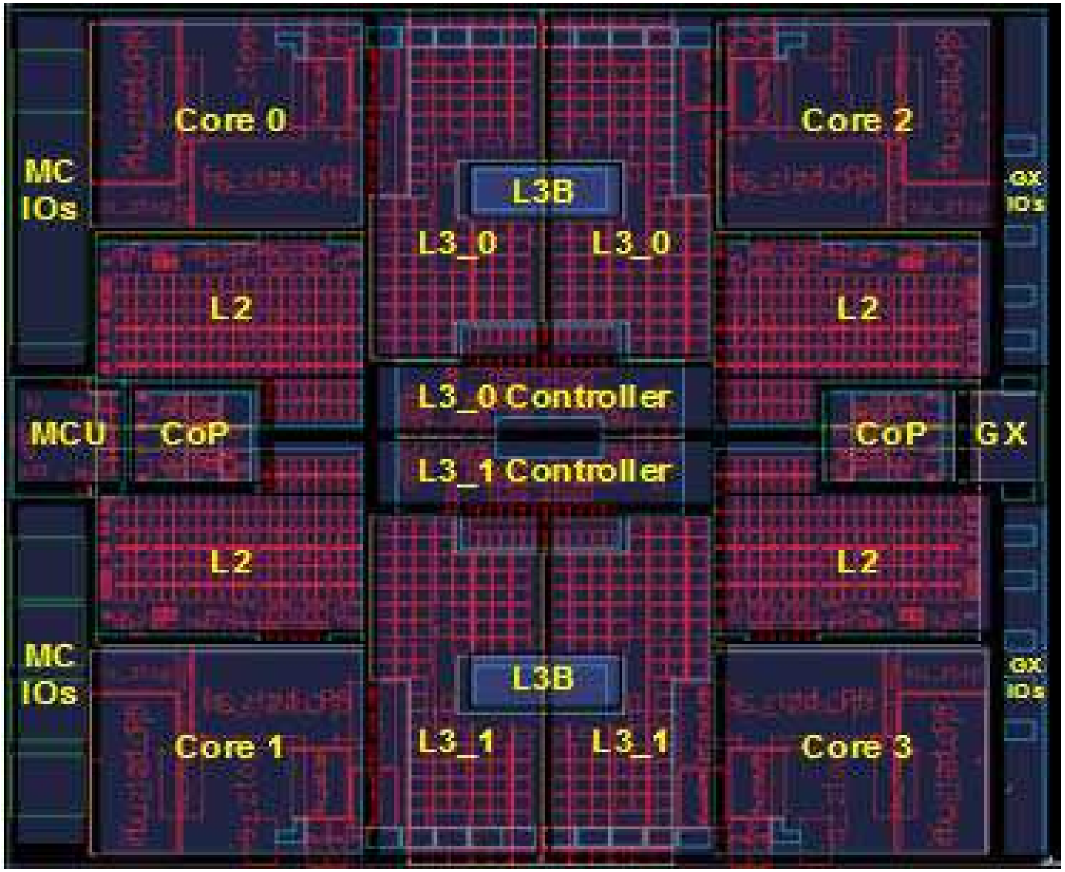
  Typical 40% thread improvement (hardware only)

  Up to additional 30% thread improvement with re-compilation

  Sustained system throughput up to 400 Billion instructions / sec
- No increase in energy consumption [2]

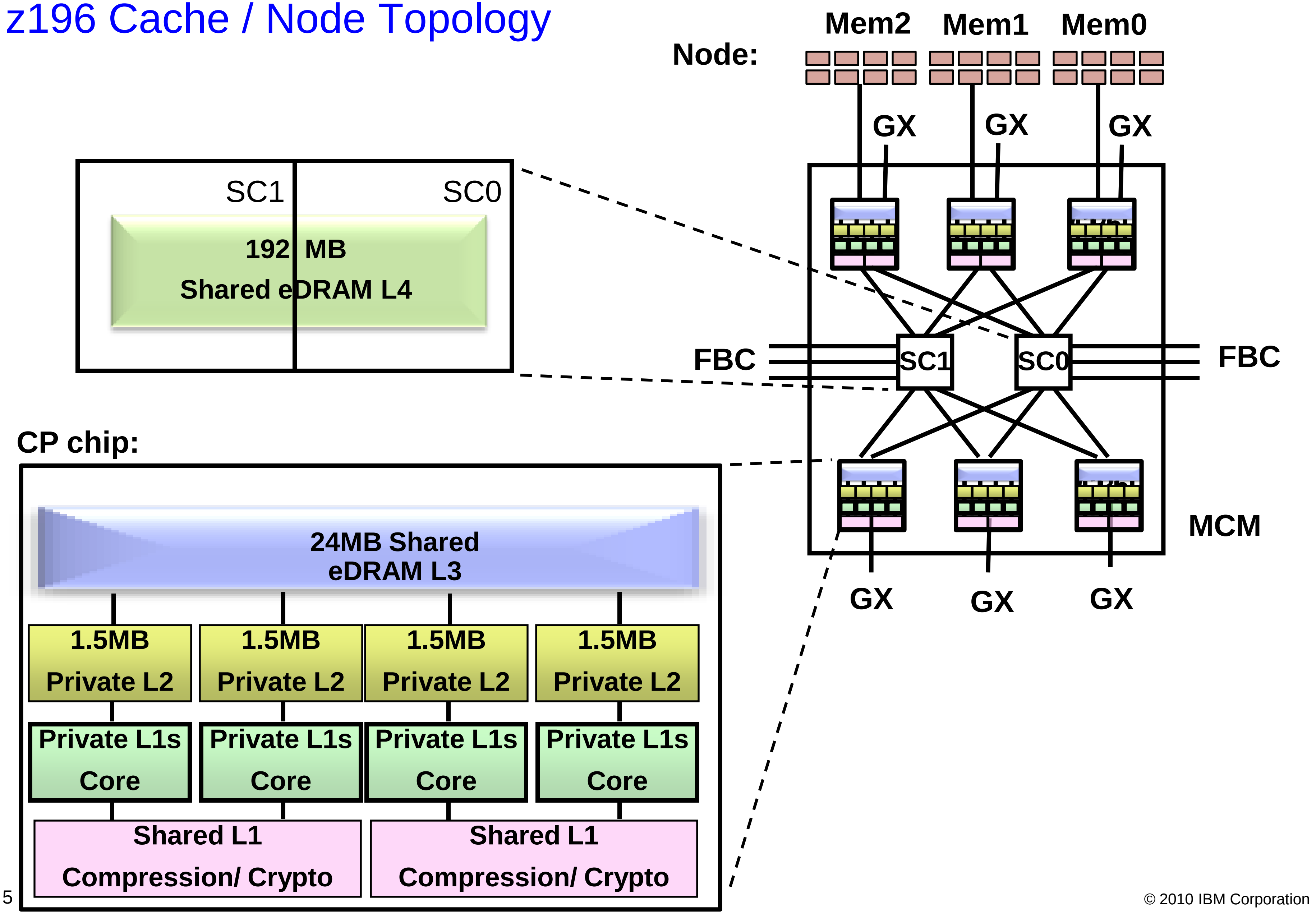
[1] vs. IBM System z10 for average LSPR workloads running z/OS® 1.11
[2] vs. IBM System z10 for comparable configurations

zEnterprise Quad Core z196 Processor Chip



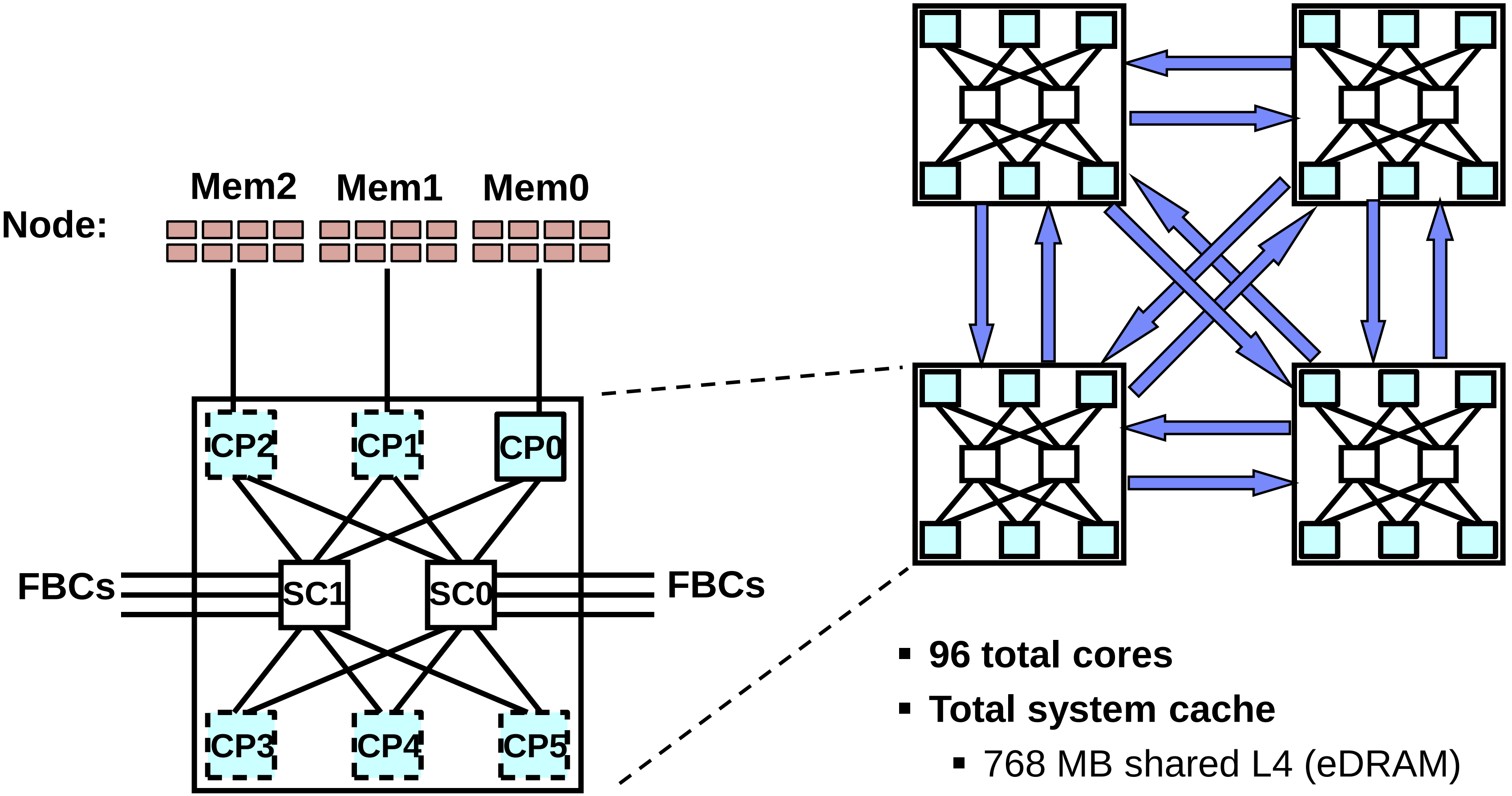
- **45nm PD SOI technology**
  - 13 layers metal
  - 3.5 km wire
  - 1.4 Billion transistors
  - 512 mm^2 chip
- **Four cores per chip**
  - Industry leadership 5.2 GHz operation
  - 64 KB L1 private I-cache
  - 128 KB L1 private D-cache
  - 1.5 MB private L2 cache/ core
- **Two Co-processors (COP)**
  - Crypto & compression accelerators
  - Each shared by two cores

z196 Cache / Node Topology



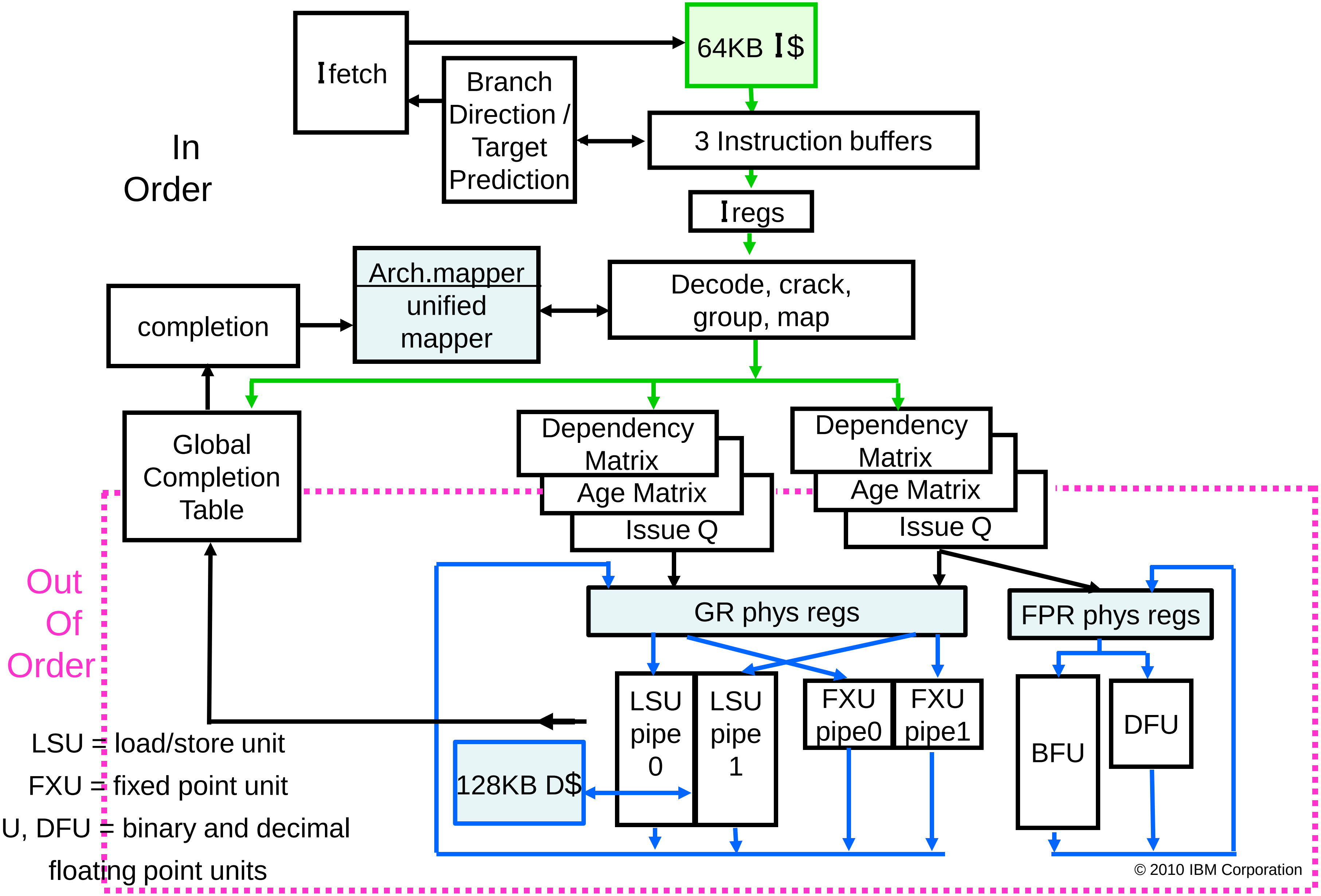
z196 Cache / Node Topology

Fully connected 4 node system:



- **96 total cores**
- **Total system cache**
  - 768 MB shared L4 (eDRAM)
  - 576 MB L3 (eDRAM)
  - 144 MB L2 private (SRAM)
  - 19.5 MB L1 private (SRAM)

z196 Microprocessor Core

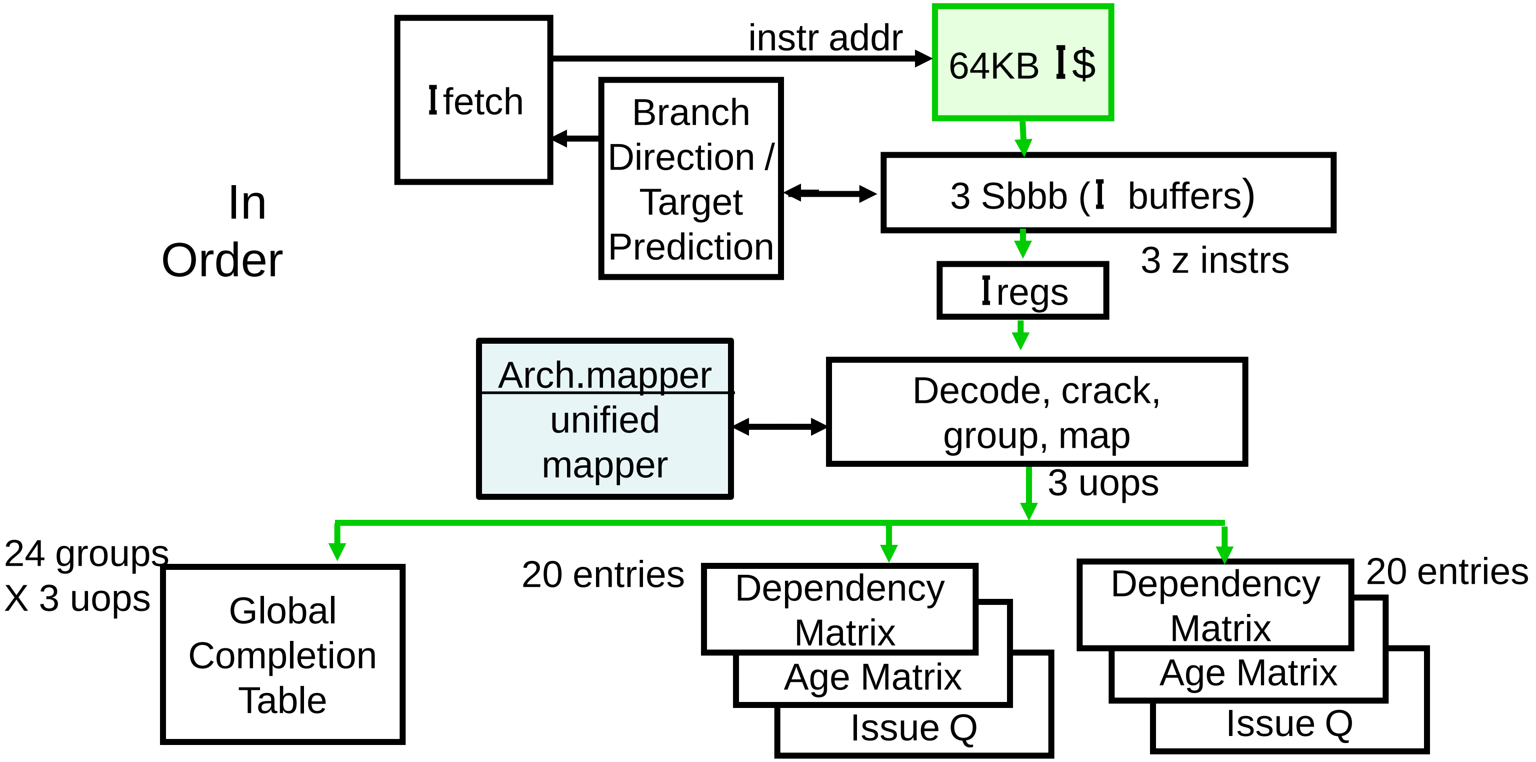


LSU = load/store unit

FXU = fixed point unit

BFU, DFU = binary and decimal floating point units

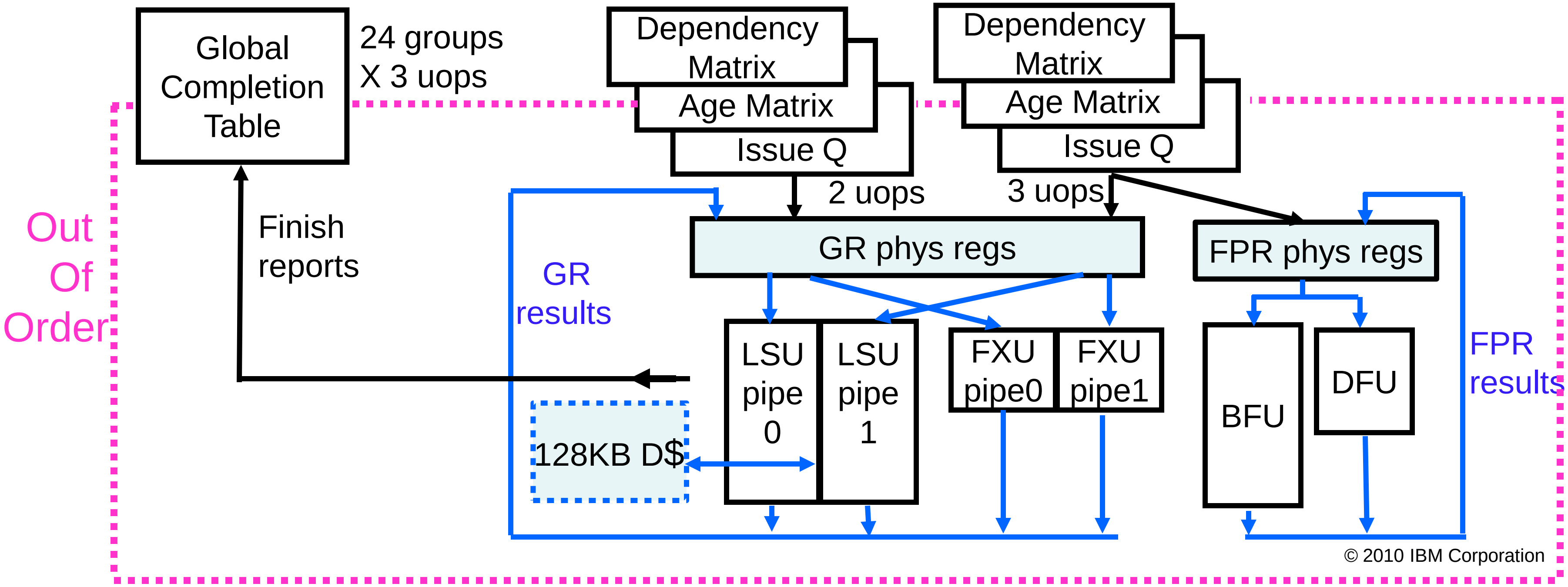
z196 Microprocessor Core (Instruction Flow)



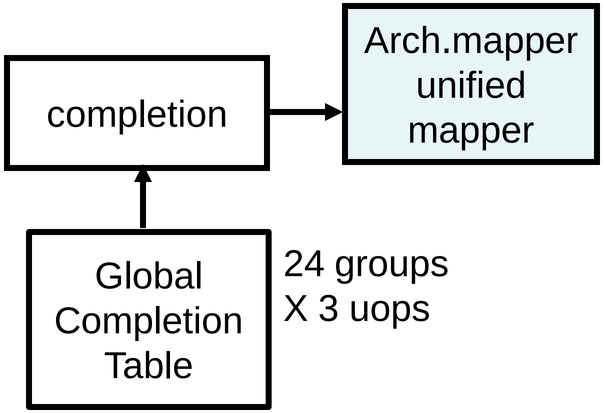
- Aggressive asynchronous branch prediction (direction and target)
- 3 z CISC instructions per cycle decode
- 211 complex instructions cracked into 2 or more RISC-like uops
- Mapper renames logical registers to physical registers
- Global completion table tracks/ completes groups of up to 3 uops

z196 Microprocessor Core (Execution Flow)

- Dependency matrix wakeup
- 40 instr OOO Issue Que with up to 72 instructions in-flight
  - Ooo store and load address generation/ execution
- Issue and execute up to 5 instrs per cycle
  - Resolve up to 2 branches (direction and target) per cycle
- Six RISC-like execution units
  - 2 FXU (integer), 2 Load/store, 1 binary FPU, 1 decimal FPU
- Execution result (including non-completed store data) forwarding

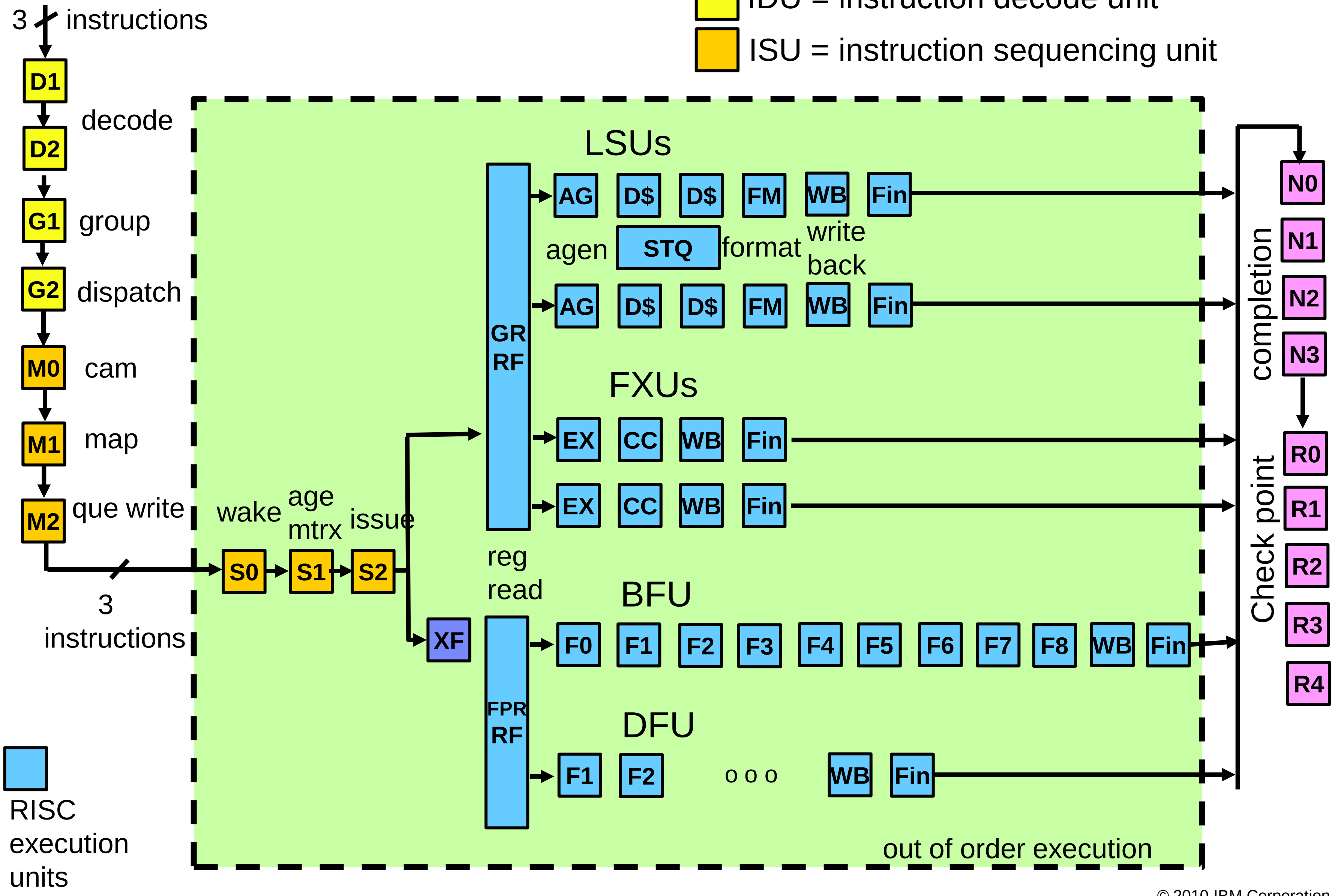


z196 Microprocessor Core (Completion)



- In-order completion
- One group (containing up to 3 uops) per cycle
- All state associated with group committed
  - Architected register mapper state
  - Store data
  - Program status word, etc.
- Data hardened through ECC or duplication with parity

z196 Microprocessor Pipeline

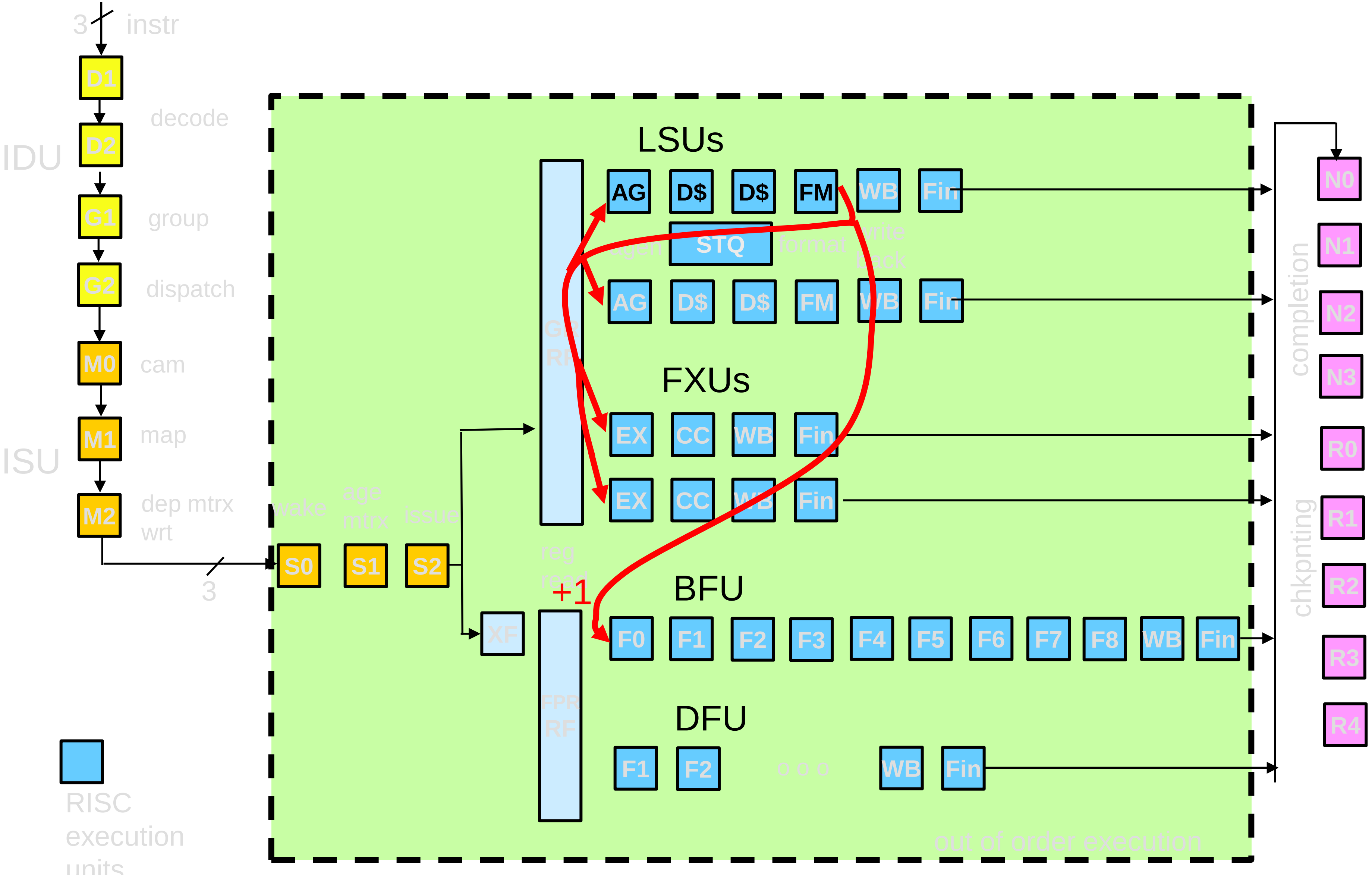


🟨 IDU = instruction decode unit

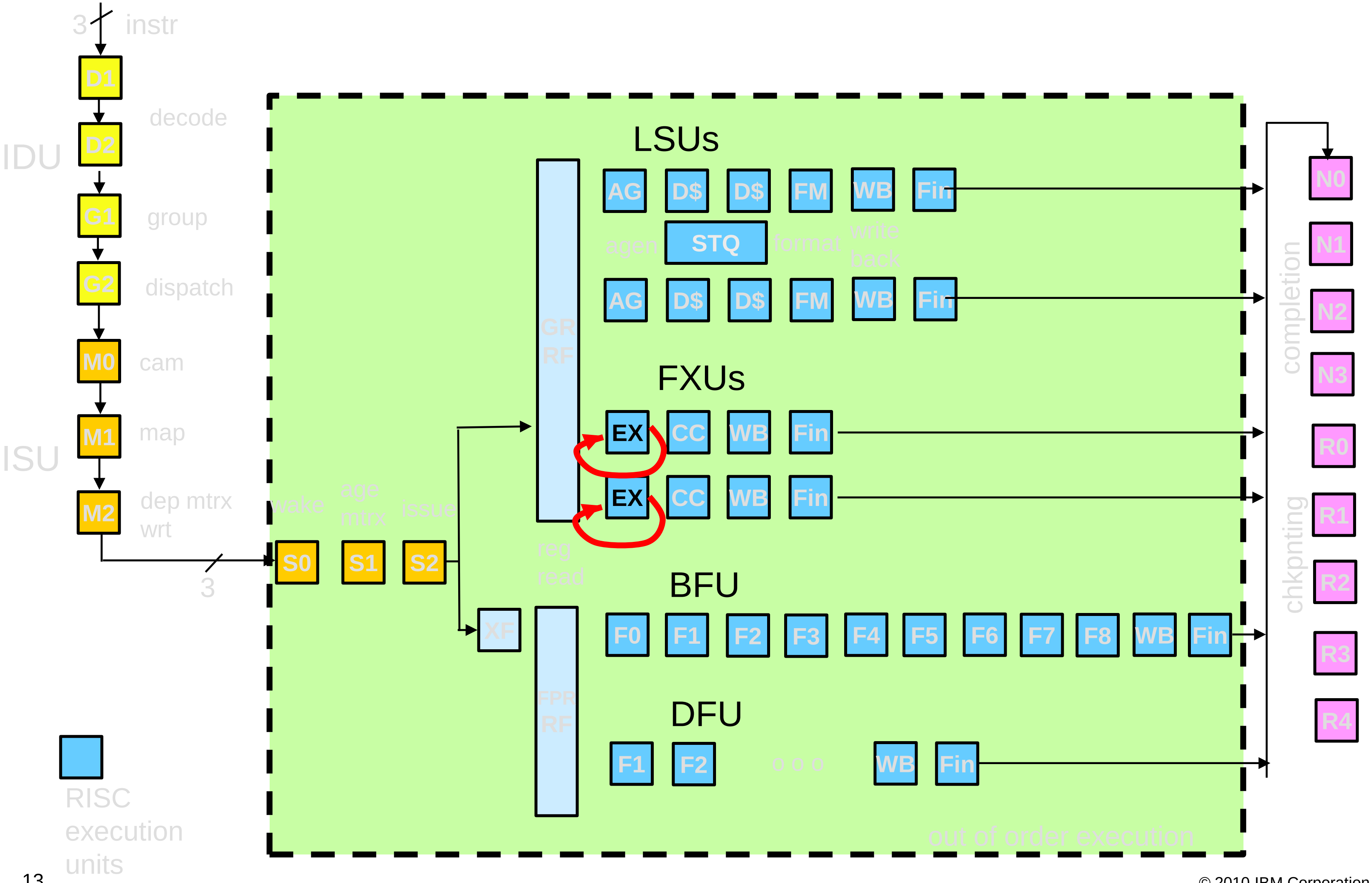
🟧 ISU = instruction sequencing unit

🟦 RISC execution units

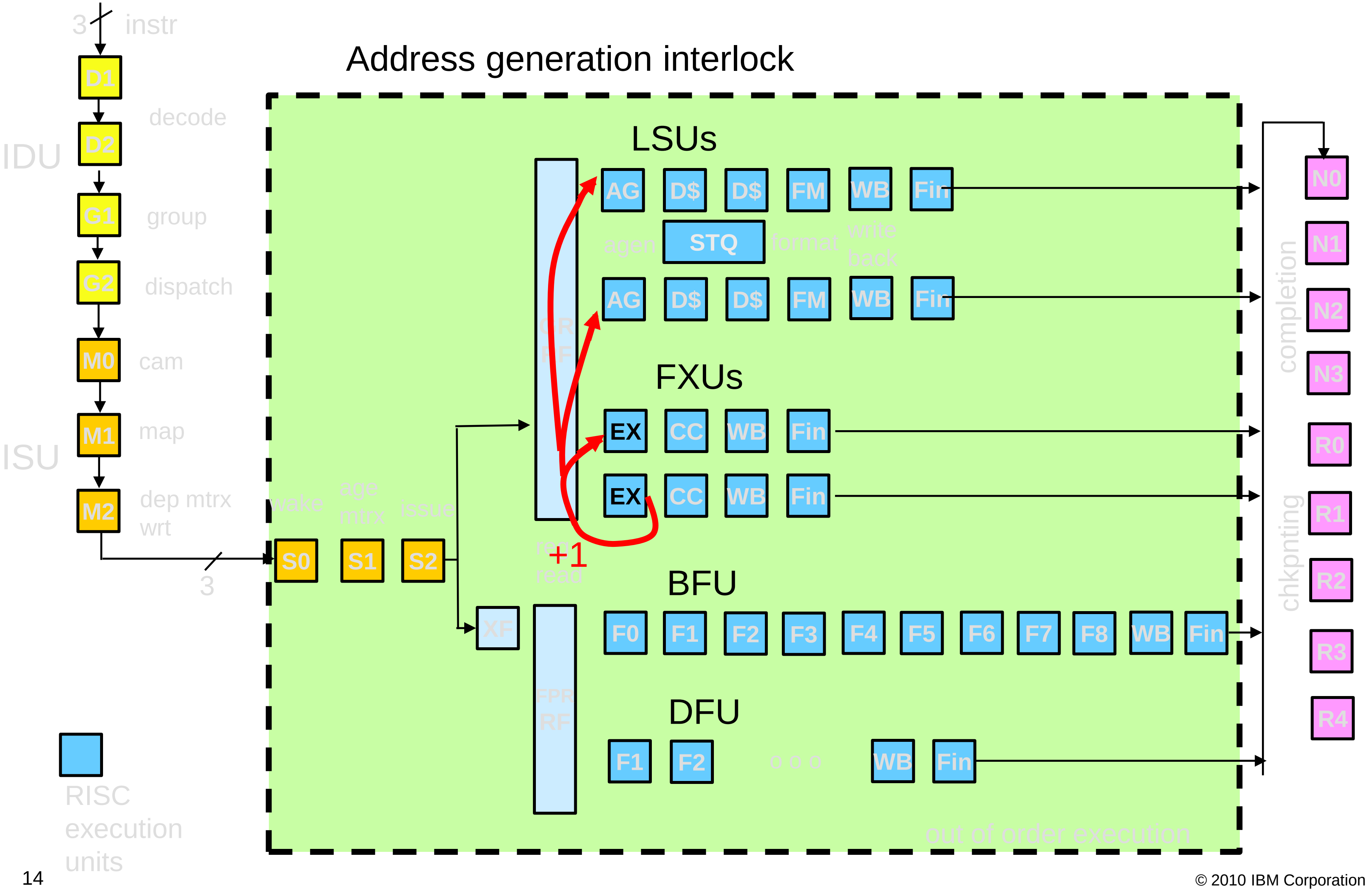
Load Data Forwarding



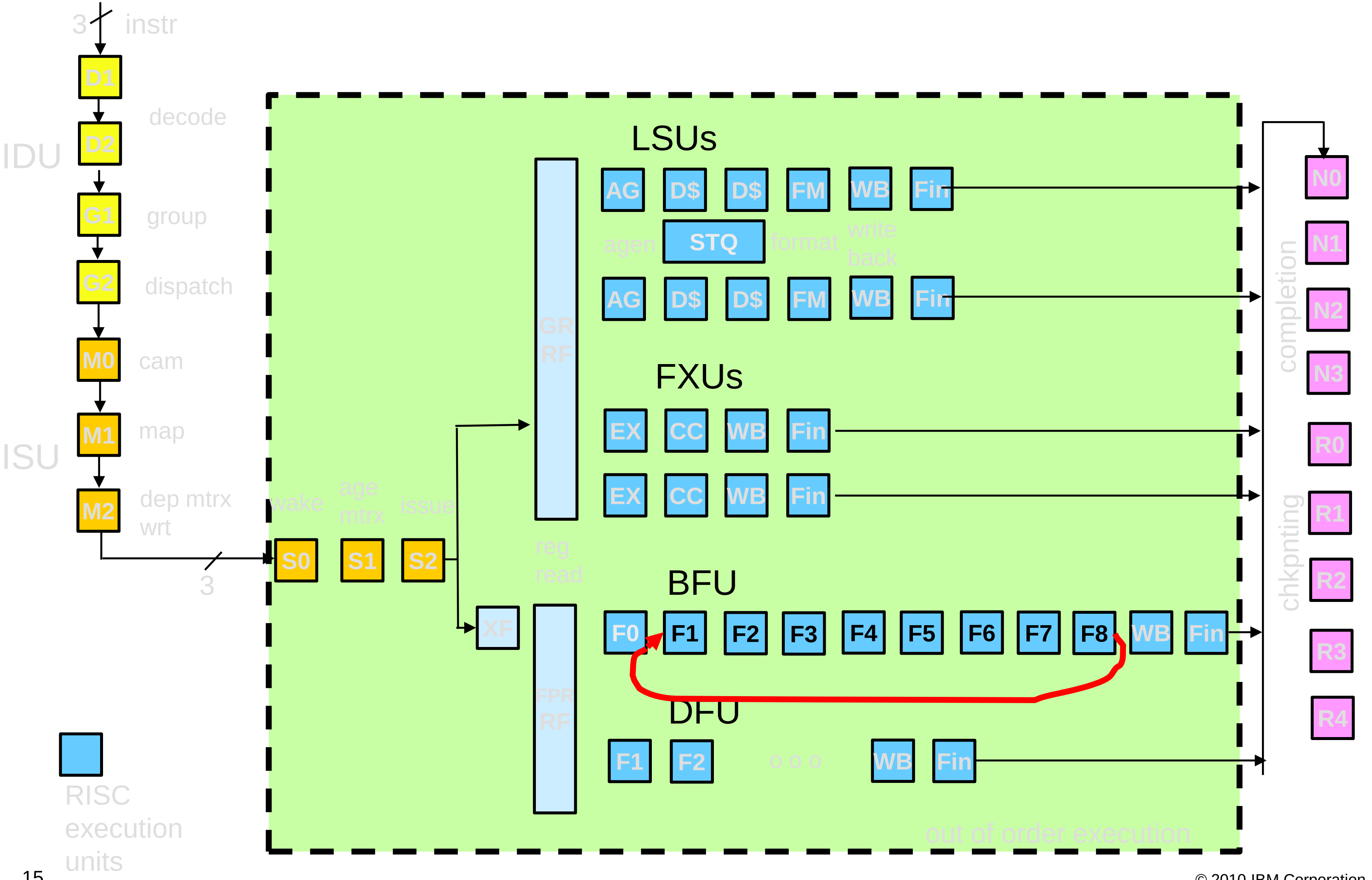
Back-to-back Fixed Point Execution



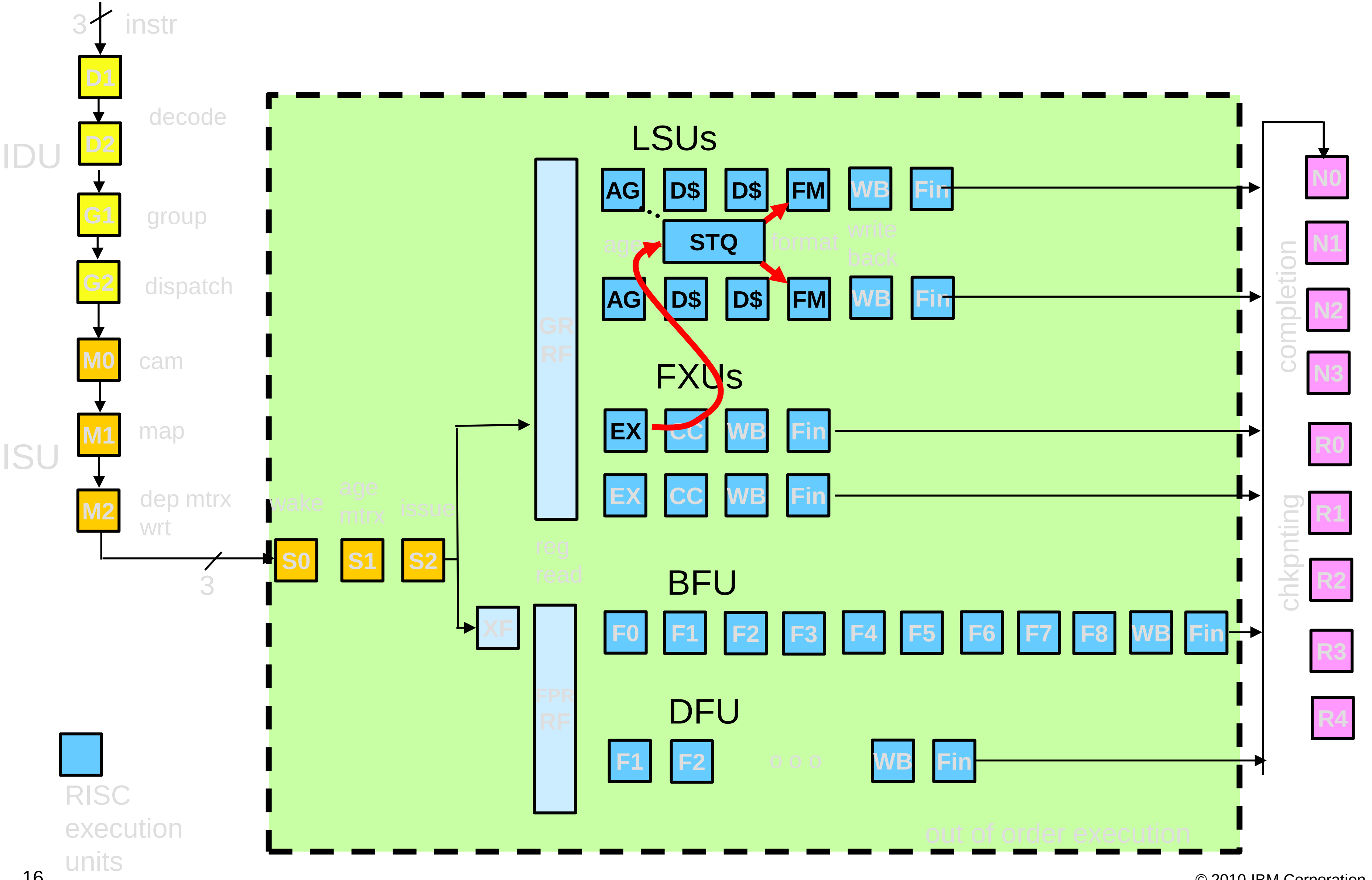
Fixed Point Result Forwarding



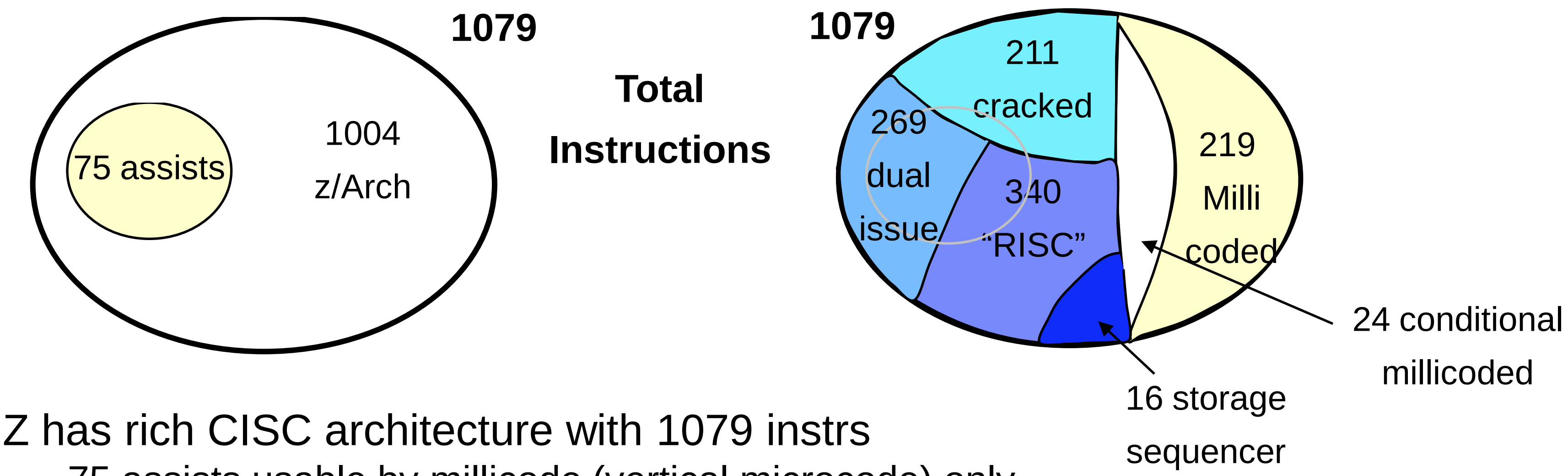
Floating Point Result Forwarding



Non-committed Store Result Forwarding



Instruction Set Architecture (ISA)



- Z has rich CISC architecture with 1079 instrs
  – 75 assists usable by millicode (vertical microcode) only

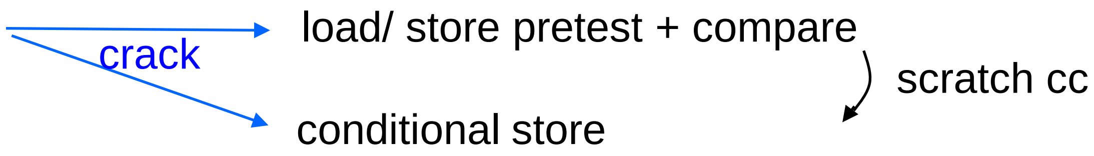
- Most complex 219 instructions are executed by millicode
  – Another 24 instructions are conditionally executed by millicode

- 211 medium complexity instructions cracked at decode into 2 or more uops
- 269 RX instructions cracked at issue → dual issued
  – RX have one storage operand and one register operand

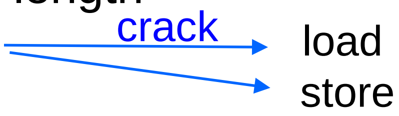
- 16 storage-storage ops executed by LSU sequencer
- Remaining z instructions are RISC-like and map to single uop

Instruction Cracking Flavors

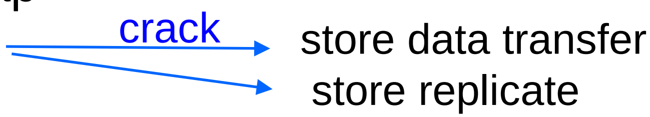
- Unconditional at decode
  - Scratch register or condition code (cc) used to pass intermediate results from one uop to another
  - E.g. compare and swap



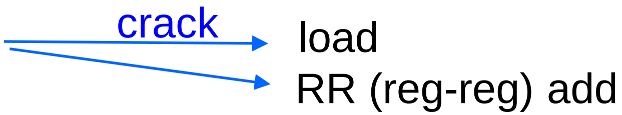
- Conditionally at decode based on operand length
  - E.g. short (8 bytes or less) move character



- Conditionally at decode based on operand overlap
  - E.g. exclusive OR with identical source operands



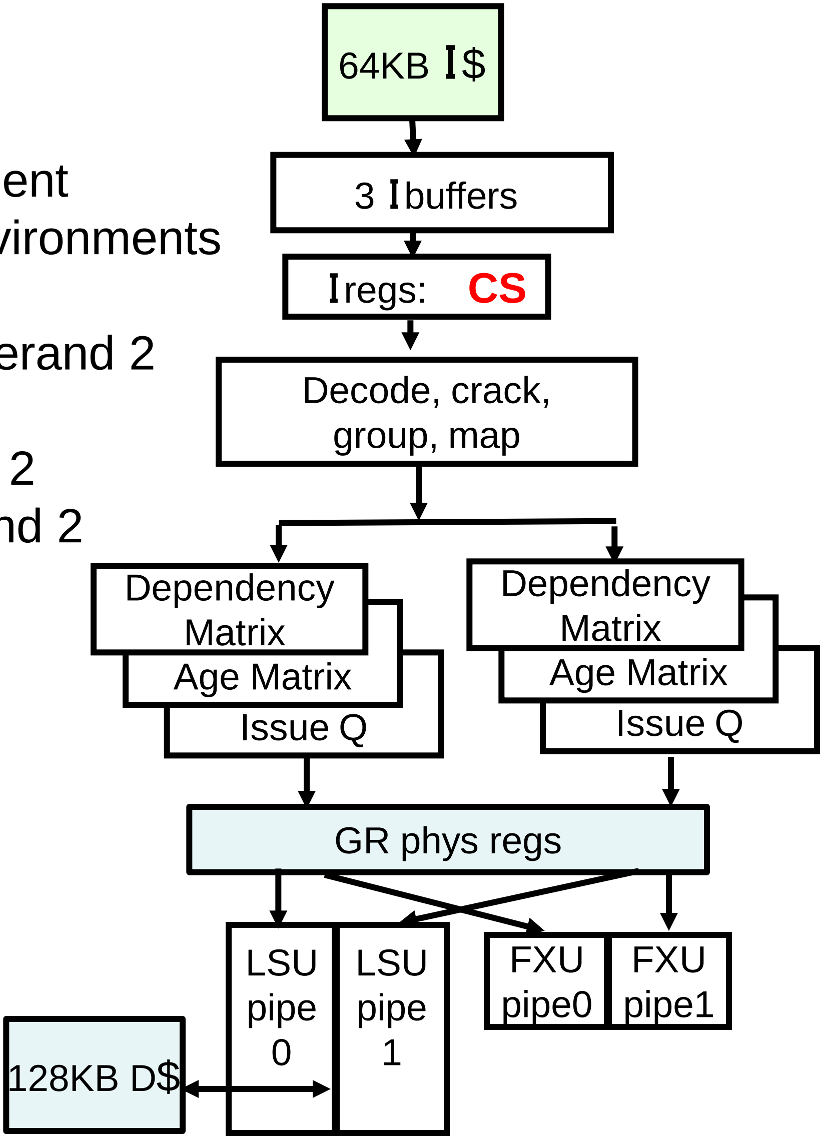
- At issue
  - E.g. RX add



Compare and Swap (CS) Cracking Example

- CISC instruction
- Executes atomically
- Used by software to implement locks in multi-threaded environments
- Function:

```
IF  register 1 == storage operand 2
 THEN  store register 3 to
       location of operand 2
 ELSE  load storage operand 2
       into register 1
```



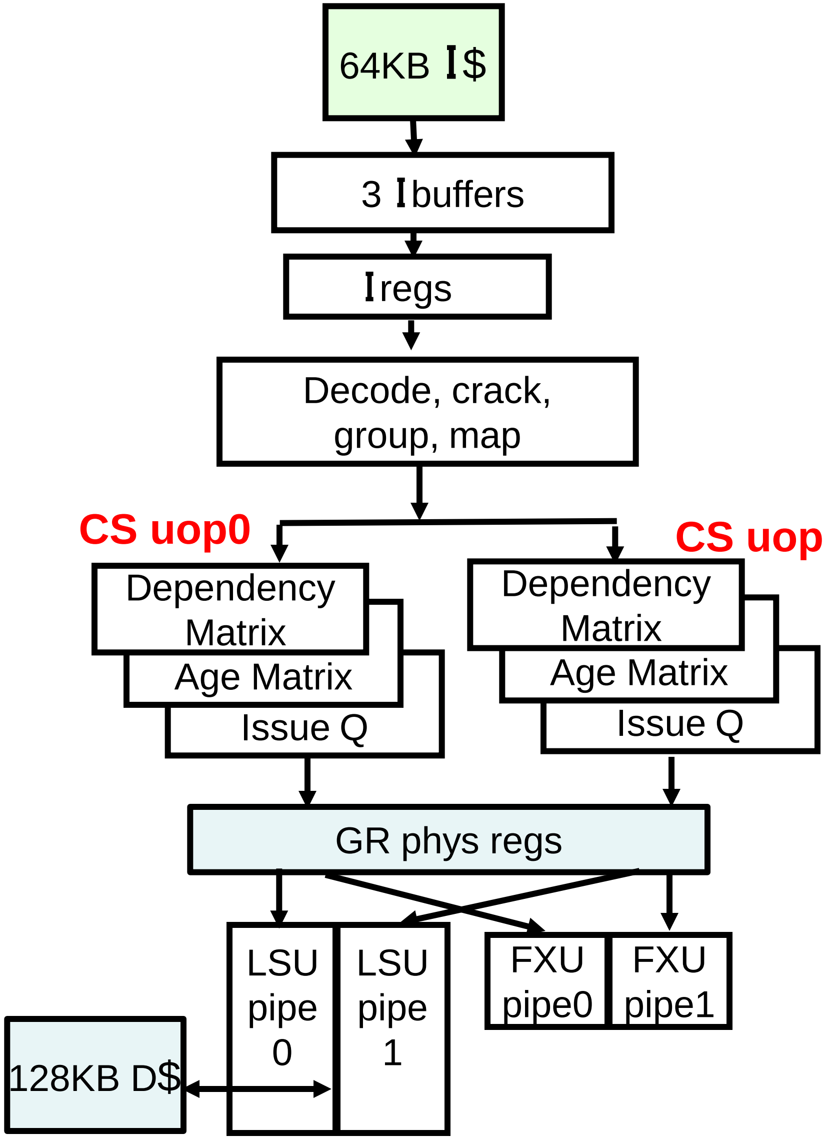
Compare and Swap (CS) Cracking Example

“dual issue”

load/store

pretest

+ compare



conditional

store

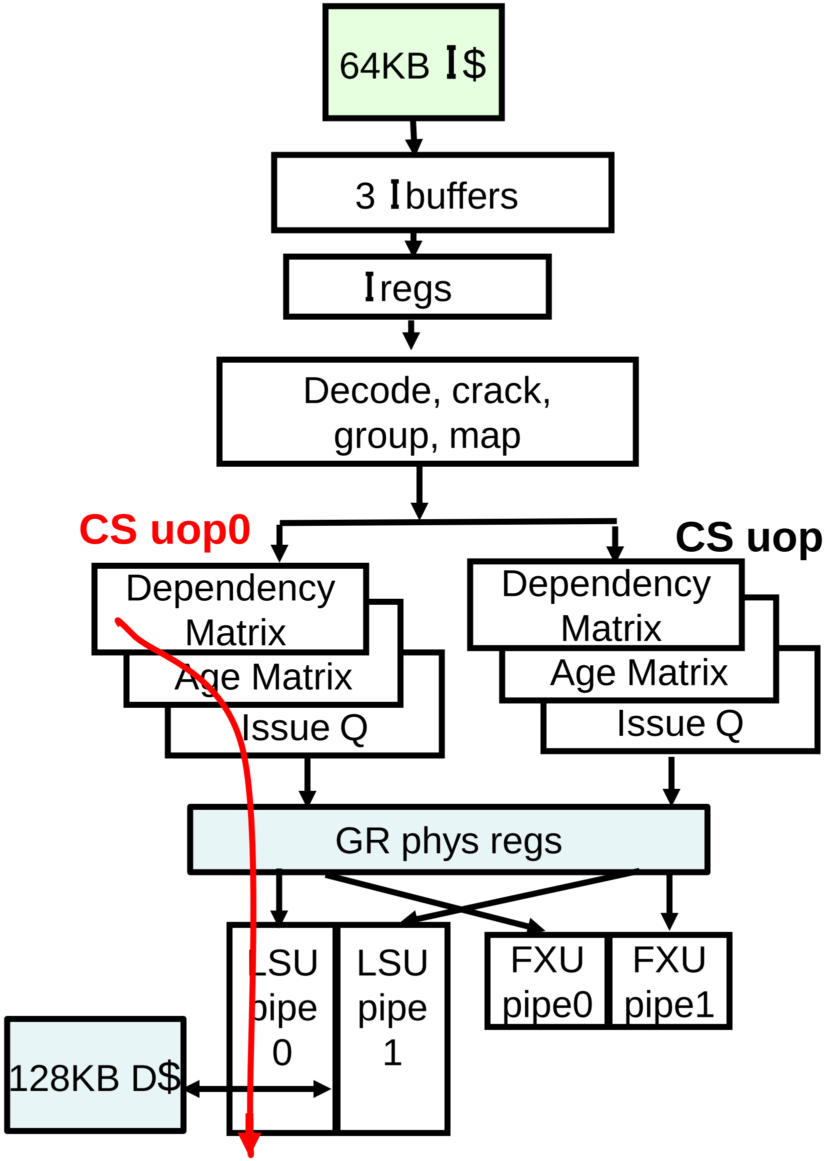
Compare and Swap (CS) Cracking Example

"dual issue"

load/store

pretest

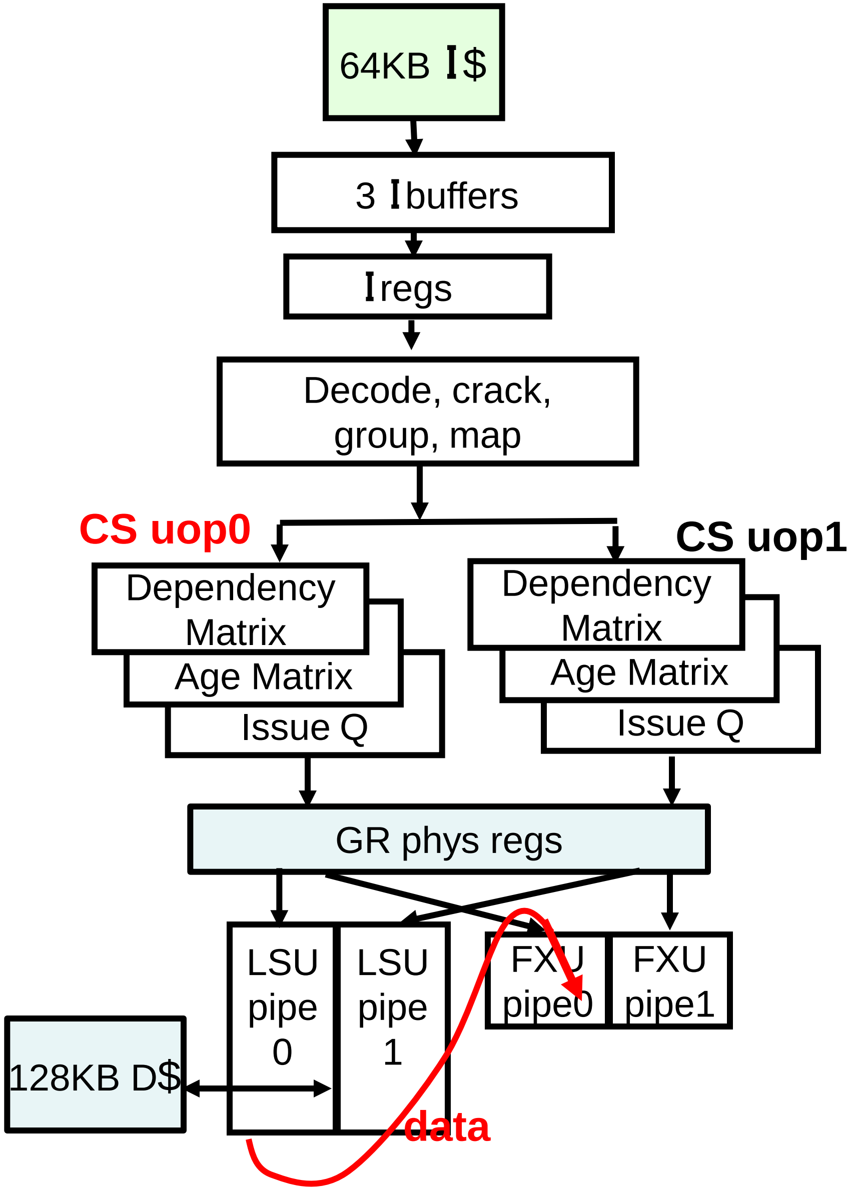
+ compare



conditional

store

Compare and Swap (CS) Cracking Example



“dual issue”

load/store

pretest

+ compare

conditional

store

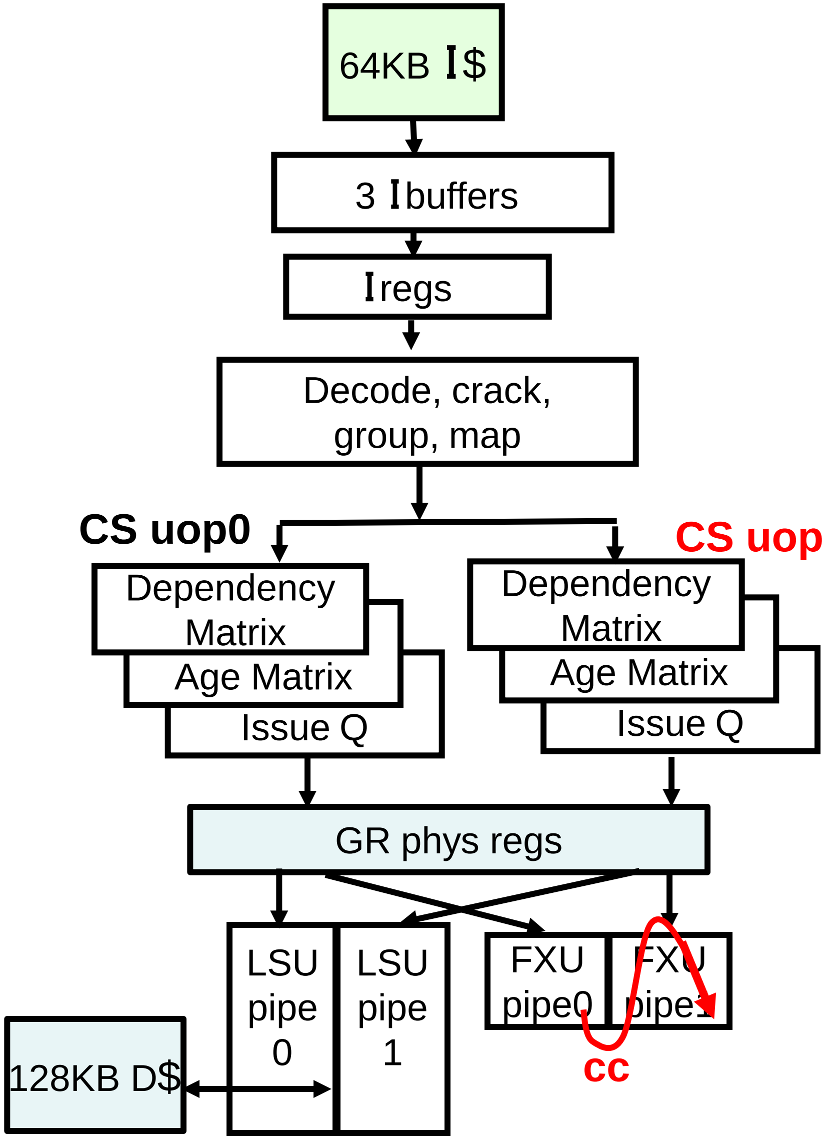
Compare and Swap (CS) Cracking Example

"dual issue"

load/store

pretest

+ compare

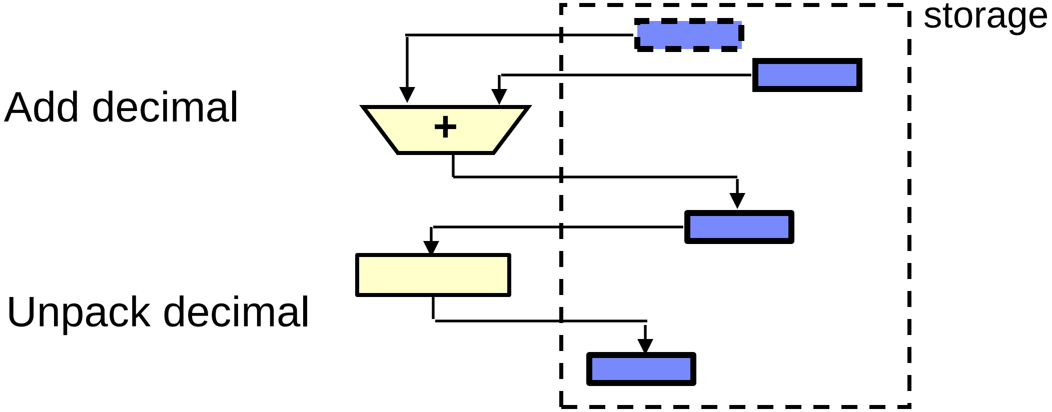


conditional

store

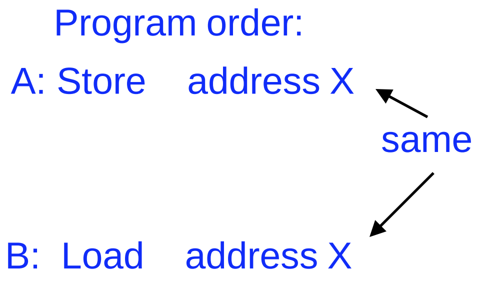
Store / Load Hazards

- Loads and stores can execute out of program order
- Storage hazards are more common than in other platforms
- Large base of z legacy code not recently re-optimized
  - Code exploits rich CISC, storage based ISA
  - E.g. decimal SS ops with storage source operands and result written to storage



- Three issues with out-of-order loads and stores
  - Functional correctness
  - Store-hit-load performance
  - Load-hit-store performance

Store / Load Hazards



Program order:

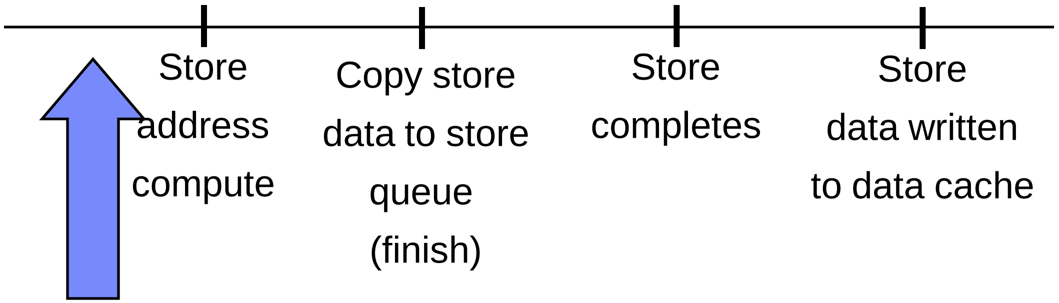
A: Store address X

same

B: Load address X

- Store load dependency
  - – Addresses and thus dependency not known at dispatch

Store A timeline:



Store address compute

Copy store data to store queue (finish)

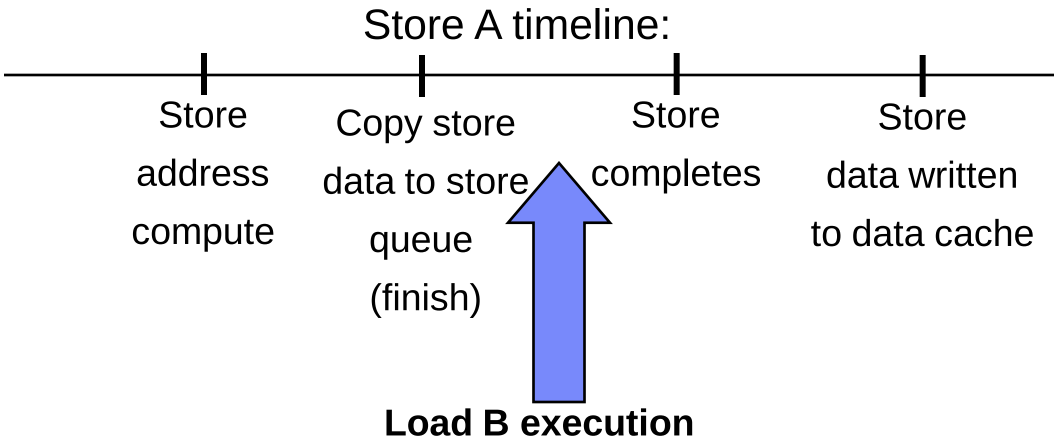
Store completes

Store data written to data cache

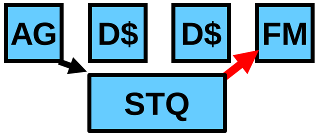
Load B execution

- CASE 1: Store-hit-load (functional correctness case)
  - – Load B executes and writes its address into load queue (LDQ)
  - – Store A executes, its address is compared to all load addresses in LDQ → hits load B
    This means load B got wrong data!
  - – Load B and younger instructions are flushed from pipeline and re-executed
  - – After this learning phase,
    Subsequent dispatches of load B are made dependent on store A

Store / Load Hazards

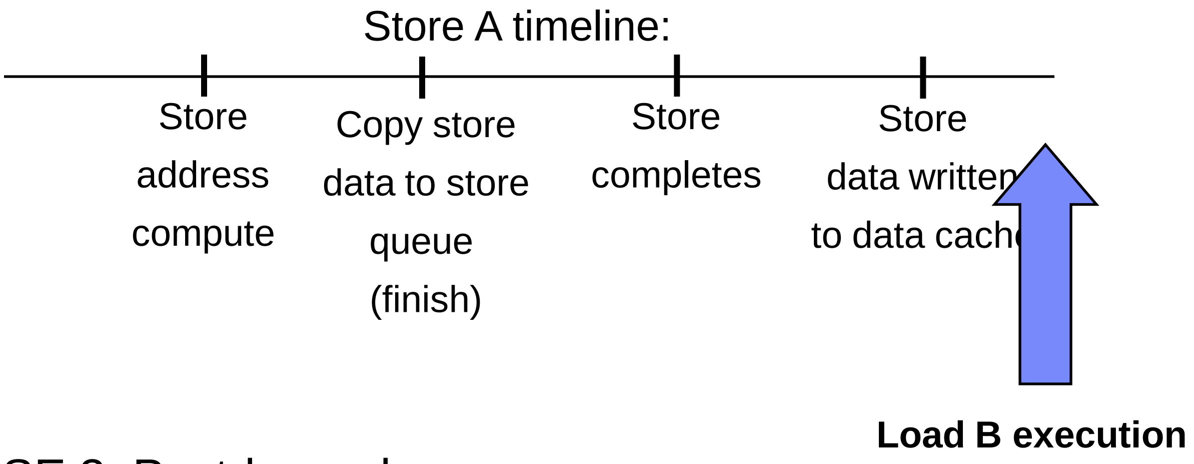

Store A timeline:
Store address compute
Copy store data to store queue (finish)
Store completes
Store data written to data cache
Load B execution

- CASE 2: Load-hit-store (performance case)
  - – Store A executes and writes its address to store queue (STQ)
  - – Load B executes, its address is compared to all store addresses in STQ → hits store A
  - – If store data available in STQ then data is directly forwarded to load B



  - – If store data not in STQ then load B is rejected and re-issued until store data is either in STQ or L1 data cache

Store / Load Hazards



Store A timeline:

Store address compute

Copy store data to store queue (finish)

Store completes

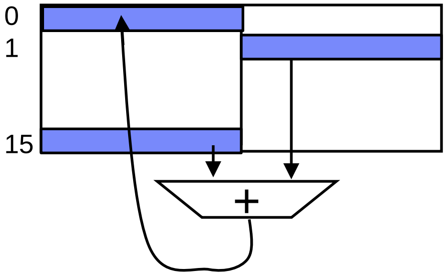
Store data written to data cache

Load B execution

- CASE 3: Post-hazard
  - – Load B serviced normally from data cache

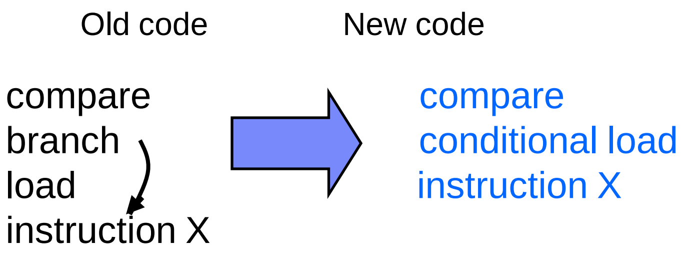
New Instruction Set Architecture

- High word extension
  - General register high word independently addressable
  - Gives software 32 word-sized registers
  - Add/subtracts, compares, rotates, loads/stores
  -



- New atomic ops
  - Load and “arithmetic” (ADD, AND, XOR, OR)
    - •(Old) storage location value loaded into GR
    - •Arithmetic result overwrites value at storage location
  - Load Pair Disjoint
    - •Load from two different storage locations into GR N, N+1
    - •Condition code indicates whether fetches interlocked

- Conditional load, store, register copy
  - Based on condition code
  - Used to eliminate unpredictable branches


Old code

compare
branch
load
instruction X

New code

compare
conditional load
instruction X

zEnterprise Microprocessor Summary

- Major advance in System z processor design
  - – Deep, high-frequency (5.2 GHz) pipeline
  - – Aggressive out of order execution core
  - – 4-level cache hierarchy with eDRAM L3 and L4
- Synergy between hardware and software design
  - – z/Architecture (ISA) extensions
  - – Compiler and micro-architecture co-optimization
  - – Robust performance gain on existing binaries (code)
- Major step up in processor performance
  - – Up to 40% performance gain on existing compute-intensive code
  - – Additional gains achievable with recompilation
- Base technology for zEnterprise system
  - – Announced: 7/22/2010

Thank You!

zEnterprise.

A New Dimension in Computing.

Trademarks

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

AIX*
BladeCenter*
CICS*
Cognos*
DataPower*
DB2*
FICON*
GDPS*
IMS
IBM*
IBM (logo)*
Parallel Sysplex*
POWER*
PR/SM
System z*
System z9*
System z10
WebSphere*
z/OS*
z/VM*
z/VSE
zEnterprise

* 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 there from.
Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. 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.
InfiniBand is a trademark and service mark of the InfiniBand Trade Association.
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.
UNIX is a registered trademark of The Open Group in the United States and other countries.
Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
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.
IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency, which is now part of the Office of Government Commerce.

* All other products may be trademarks or registered trademarks of their respective 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.