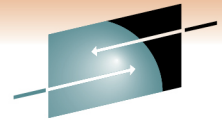
How Do You Do What You Do When You're a z196 CPU?

Bob Rogers
IBM Corporation
rrrogers@us.ibm.com

SHARE in Anaheim

Trademarks



The following are trademarks of the International Business Machines Corporation in the United States and/or other countries. For a complete list of IBM Trademarks, see www.ibm.com/legal/copytrade.shtml IBM, IBM logo, BladeCenter*, Build Forge*, CICS*, ClearCase*, ClearQuest*, DB2*, DB2 Connect, DB2 Universal Database, Domino, Enterprise Storage Server*, eServer, GDPS*, Geographically Dispersed Parallel Sysplex, HiperSockets, Lotus*, NetView*, OMEGAMON*, OS/390*, OS/400*, Parallel Sysplex*, pSeries*, RACF*, Rational*, RequisitePro*, Sametime*, SiteProtector, System i, System p, System Storage, System x, System z, System z9*, System z10, Tivoli*, TotalStorage*, WebSphere*, z9, z10,z/OS*, z/VM*, and zSeries*.

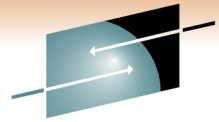
The following are trademarks or registered trademarks of other companies

Lotus, Notes, and Domino are trademarks or registered trademarks of Lotus Development Corporation
Java and all Java-related trademarks and logos are trademarks of Sun Microsystems, Inc., in the United States and other countries
Linux is a registered trademark of Linux Torvalds
UNIX is a registered trademark of The Open Group in the United States and other countries.
Microsoft, Windows and Windows NT are registered trademarks of Microsoft Corporation.
SET and Secure Electronic Transaction are trademarks owned by SET Secure Electronic Transaction LLC.
Intel is a registered trademark of Intel Corporation
* 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.
References in this document to IBM products or services do not imply that IBM intends to make them available in every country.
Any proposed use of claims in this presentation outside of the United States must be reviewed by local IBM country counsel prior to such use.
The information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.
Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

Important Disclaimer



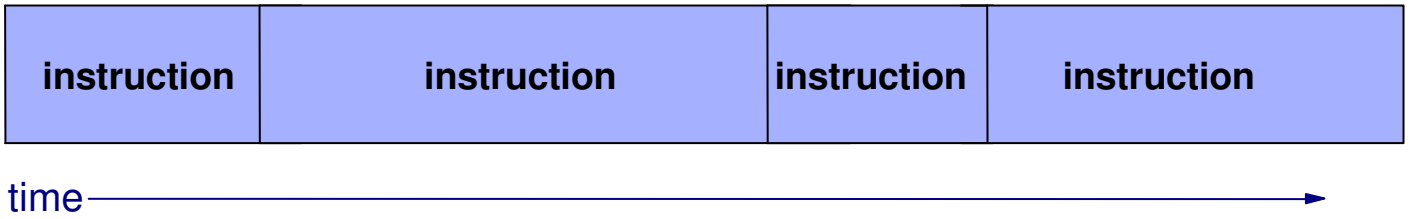
- THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY.
- WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED.
- IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION.
- NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, OR SHALL HAVE THE EFFECT OF:
  - CREATING ANY WARRANTY OR REPRESENTATION FROM IBM (OR ITS AFFILIATES OR ITS OR THEIR SUPPLIERS AND/OR LICENSORS); OR
  - ALTERING THE TERMS AND CONDITIONS OF THE APPLICABLE LICENSE AGREEMENT GOVERNING THE USE OF IBM SOFTWARE.

Topics

- Overview of instruction Processing
- What's different about z10
- Superscalar Grouping
- The Pipeline and its Hazards
- What's different about z196
- Branch Prediction
- Cache Topology
- Coprocessors
- TLB2 and Large Pages

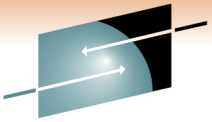
Conceptual View of Execution

Instructions are executed in the order they are seen.
Every instruction completes before the following instruction begins.
Instructions take a varying amount of time.
Instructions have direct and immediate access to main storage.



But, this is an illusion.

Pipeline View of Instructions



Individual instructions are really a sequence of dependent activities, varying by instruction:

Instruction Fetch	Instruction Decode	Operand Address	Operand Fetch	Execute	Putaway Result

for example: `A R1,D2(X2,B2)`

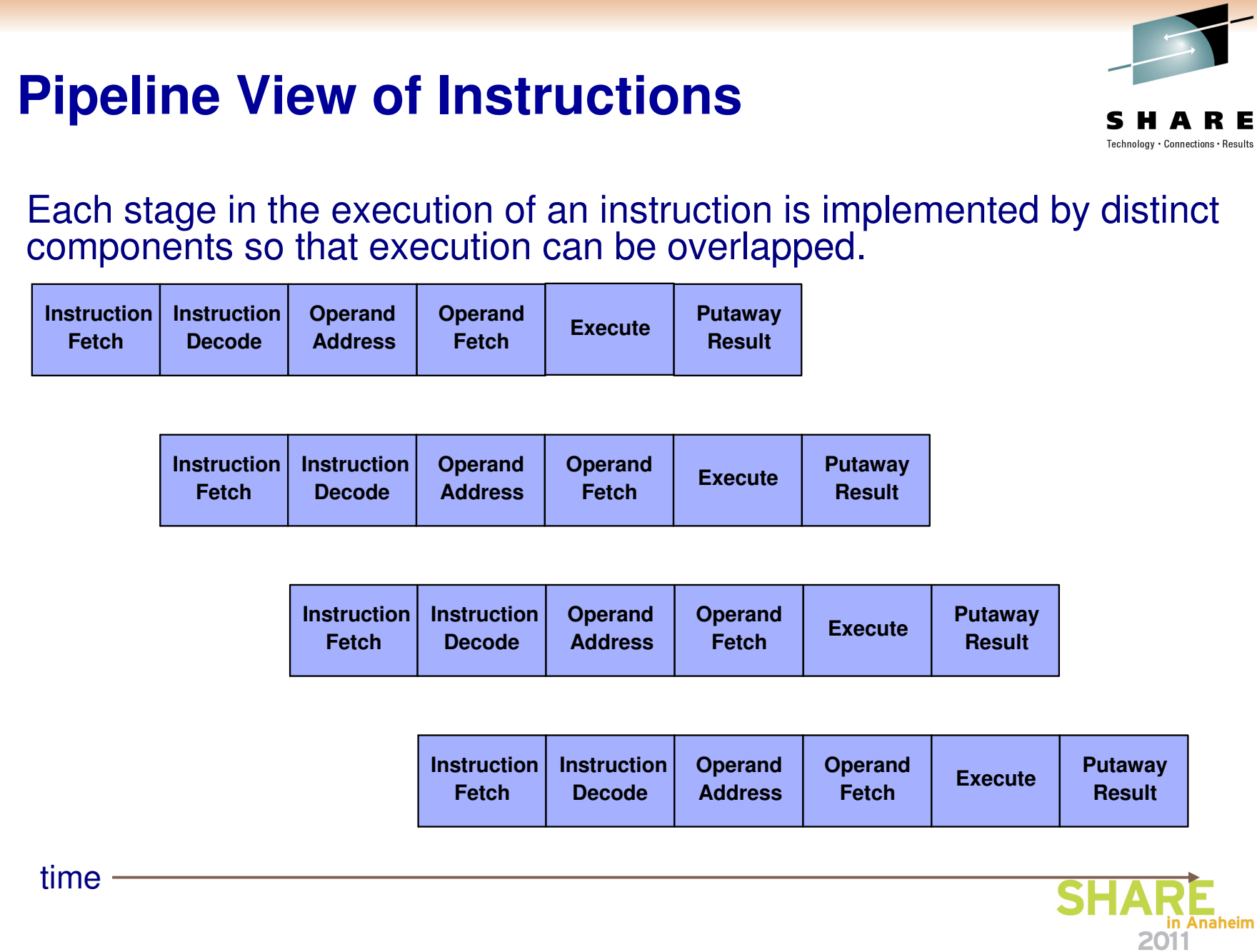
Instruction Fetch	Instruction Decode	Operand1 Address	Operand1 Fetch	Operand2 Address	Operand2 Fetch	Execute	Putaway Result

for example: `CLC D1(L,B1),D2(B2)`

Instruction Fetch	Instruction Decode	Execute Instruction as an "internal subroutine" (millicode)

for example: `UPT` **(Update Tree)**

Pipeline View of Instructions



Each stage in the execution of an instruction is implemented by distinct components so that execution can be overlapped.

Instruction Fetch	Instruction Decode	Operand Address	Operand Fetch	Execute	Putaway Result

Instruction Fetch	Instruction Decode	Operand Address	Operand Fetch	Execute	Putaway Result

Instruction Fetch	Instruction Decode	Operand Address	Operand Fetch	Execute	Putaway Result

Instruction Fetch	Instruction Decode	Operand Address	Operand Fetch	Execute	Putaway Result

time

Pipeline stalls and rejects

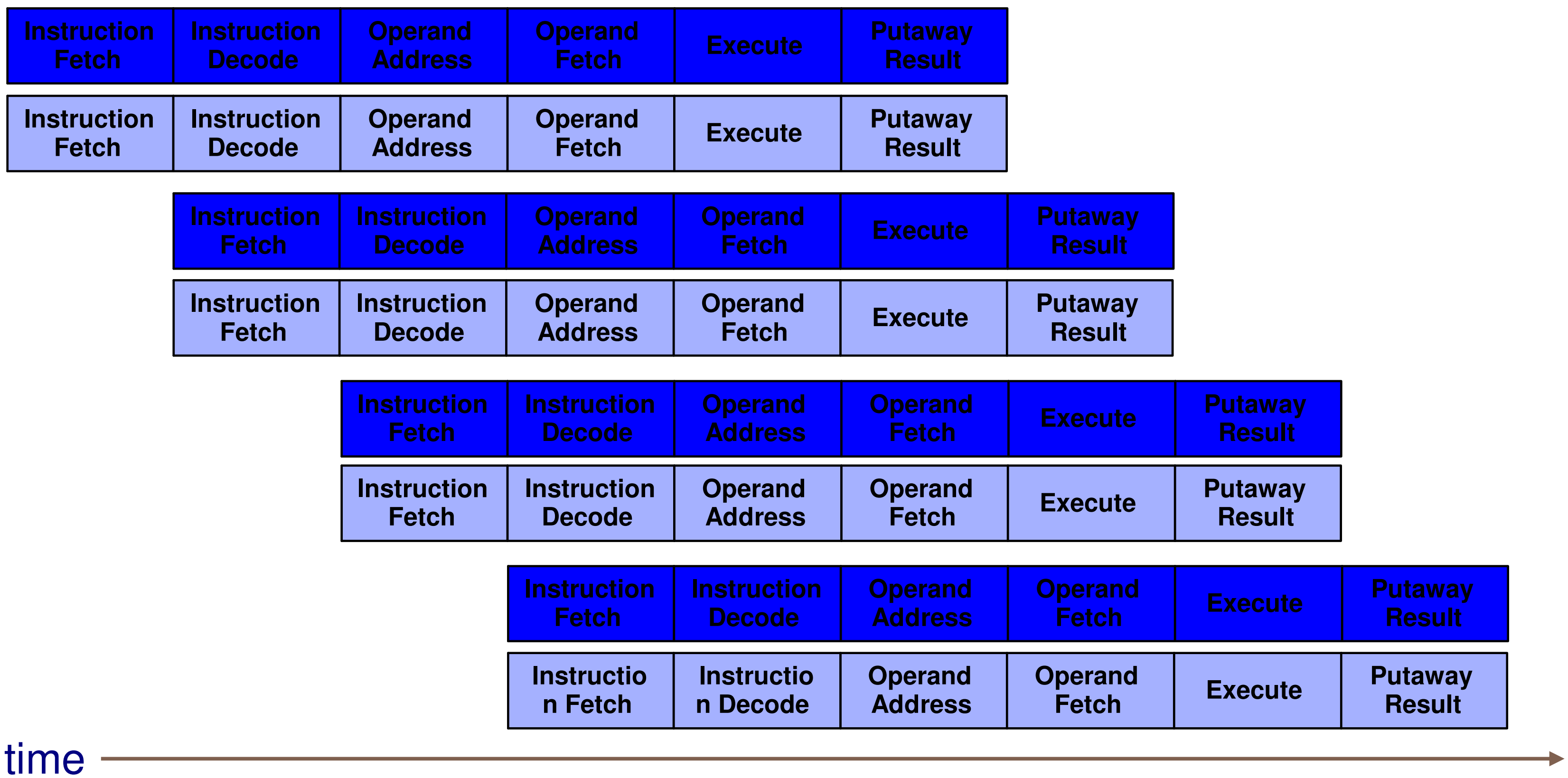
- Address Generation Interlock (AGI)
  - Waiting for the results of a previous instruction to compute an operand address
  - z10 and z196 have AGI bypasses that makes the results of Load Address and some Load instructions available before Putaway
  - A group (on z10) or single instruction (on z196) is stalled in the decode/issue unit until interlock is resolvable to avoid pipeline reject later
- Operand Store Compare (OSC)
  - Waiting to re-fetch a recently modified operand
  - The data is unavailable while in the “store queue” waiting to be updated in L1 cache.

Pipeline stalls and rejects

- Instruction Fetch Interlock (IFI)
  - reloading instructions as a result of stores into the instruction stream (actually anywhere in the same cache line)
  - causes pipeline flush, clearing decoded instructions and refetching of instruction cache line (very costly)
- Branch Misprediction
  - branching (or not branching) in a way other than the processor has guessed.
  - z10 and z196 have complex branch prediction logic
  - relative branches have a lower penalty for incorrect prediction
  - untaken branches don't need to be predicted
  - "code straightening" is a good idea

Superscalar multiple instruction overlap

A Superscalar processor can process multiple instructions simultaneously because it has multiple units for each stage of the pipeline. But, the apparent order of execution is still maintained.



Superscalar Grouping Rules on z10

- Most single-cycle instructions are “superscalar”
- Instruction groups contain 1 or 2 superscalar instructions
- First or Last instruction can be a branch instruction
- Instruction groups are held in decode dispatch unit to avoid pipeline hazards like AGI and OSC
- Some instructions that were superscalar on z9 are not superscalar in z10

Instruction Scheduling for In-Order Execution

Original Code Sequence

7 instruction groups and 10 cycles AGI delay

```
AGI seq     instruction text              | seq   instruction text
    01   LLGT   @04,XFORNP31              |
<4> 02   L      @04,FW(,@04)              | 03   ST    @04,XFORS
    04   LG     @05,TOPPTR                |
<2> 05   LG     @09,RTTOP(,@05)           |
<2> 06   ST     @04,RSISIZE(,@09)         | 07   SLR   @02,@02
    08   ST     @02,RSIPREV(,@09)         | 09   LG    @02,RDIPTR64
<2> 10   LH     @08,RDITYPE(,@02)         |
```

Reordered Code Sequence

5 instruction groups and 6 cycles AGI delay

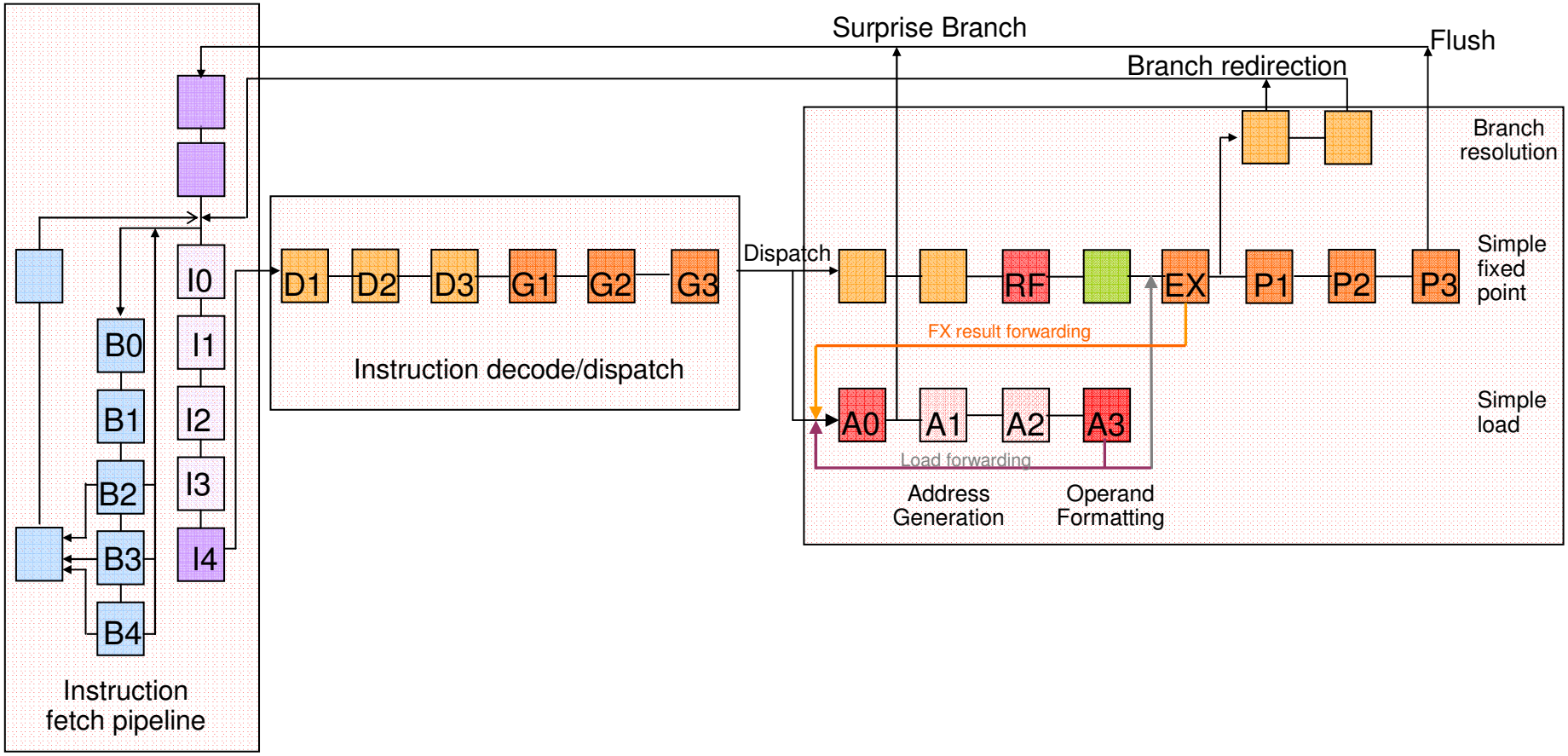
```
AGI seq     instruction text              | seq   instruction text
    01   LLGT   @04,XFORNP31              | 04   LG    @05,TOPPTR
<2> 05   LG     @09,RTTOP(,@05)           | 07   SLR   @02,@02
<2> 02   L      @04,FW(,@04)              | 06   ST    @04,RSISIZE(,@09)
    08   ST     @02,RSIPREV(,@09)         | 09   LG    @02,RDIPTR64

<2> 03   ST     @04,XFORS                 | 10   LH    @08,RDITYPE(,@02)
```

The IBM System z10 compared to z9

- z10 has a radically different instruction processor
  - *high frequency* processor
    - 4.4 GHz vs 1.7GHz (2.5x)
  - much longer instruction pipeline
    - 14 stages vs 6 stages
  - different type of instruction pipeline
    - Rejecting pipeline vs stalling pipeline
    - Reject-recycle cost about 9 cycles
  - still performs in-order execution
  - still favors RX instructions

System z10 Instruction Pipeline (partial)



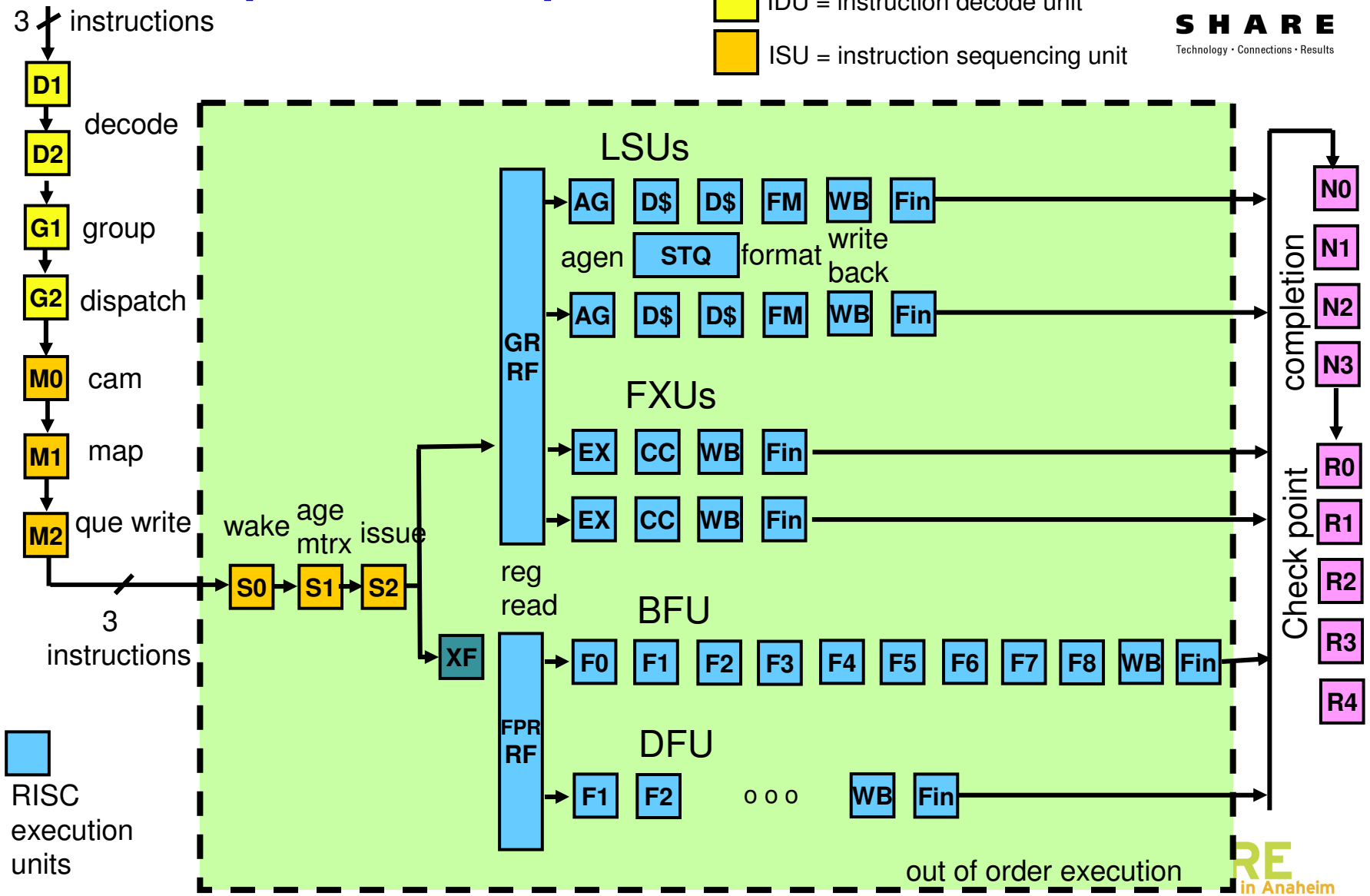
High frequency is great, but….

- There are some negative affects cause by the short cycle time. For example:
  - Some instructions can no longer be done in the shorter cycle time and now take more than one cycle
    - Most instructions that involve sign propagation (e.g. LH) are no longer single cycle
- Keeping the pipeline fed with instructions and data is very challenging
  - Memory access seem to take longer when measured in instruction cycles.
  - i-cache and d-cache size reduced to retain low latency at high frequency.
- Some pipeline hazards are more costly
  - Longer pipeline causes more cycles lost on reject/recycle and branch mispredict
  - More cases cause reject/recycle rather than stall

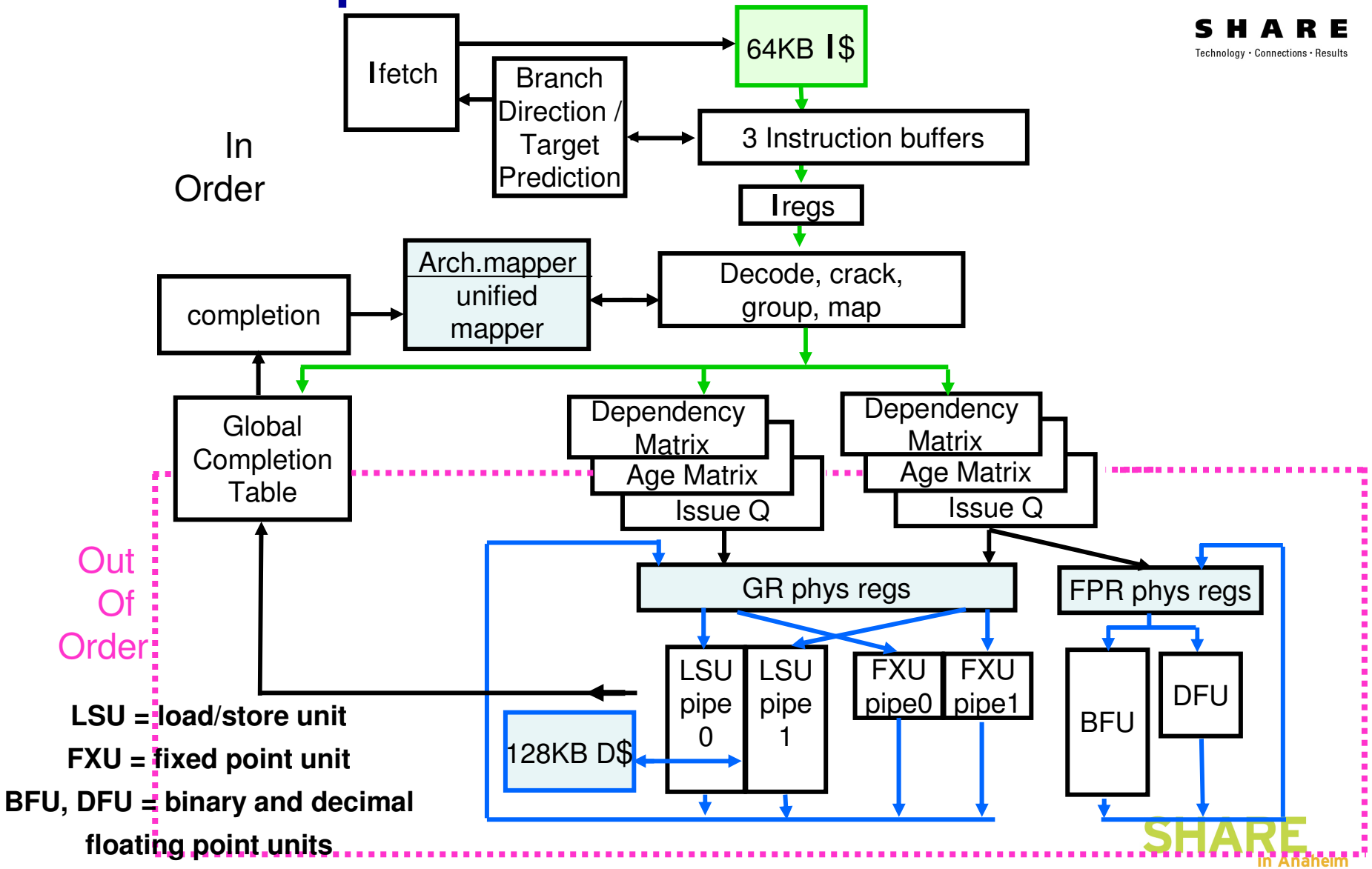
The IBM System z196 compared to z10

- z196 continues evolution high frequency and performance
  - Higher frequency
    - 5.2 GHz vs 4.4 GHz
  - Variable length instruction pipeline
    - 15 to 17 stages vs 14 stages (fixed point)
  - Out-of-Order vs In-Order execution
    - Instruction queue of 40 instructions
    - Up to 72 instructions in flight
  - RX-type instruction no longer being favored more than RISC-like instructions
    - However, simple RX instruction have some benefits in instruction path-length with the dual issue design of issue queue
  - Decode up to 3 instructions/cycle vs only 2
  - Execute up to 5 instructions/cycle vs only 2

z196 Microprocessor Pipeline



z196 Microprocessor Core



New on z196

- Instruction Cracking
  - Breaking more complex instructions into simpler micro-ops
- Register Renaming
  - Using a larger set of physical registers to enable multiple logical copies of the same architected registers
- Out-of-Order Execution (OOO)
  - Executing instructions before their normal execution order once any dependencies have been resolved
  - Micro-ops from cracked instructions can be scheduled independently

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 → crack → load/ store pretest + compare; conditional store
    scratch cc
- Conditionally at decode based on operand length
  - E.g. short (8 bytes or less) move character → crack → load; store
- Conditionally at decode based on operand overlap
  - E.g. exclusive OR with identical source operands → crack → store data transfer; store replicate
- At issue
  - E.g. RX add → crack → load; RR (reg-reg) add

Ex. of Cracking, Renaming and OOO

- Identify dependencies between instructions
- speculatively execute instructions out of order
- uses extra physical registers to enable OOO without getting incorrect results

```
L   R1,A
A   R1,B
ST  R1,C
L   R1,X
A   R1,Y
ST  R1,Z
```

crack / rename →

```
L   P1,A
L   P2,B
AR  P2,P1
ST  P2,C
L   P3,X
L   P4,Y
AR  P4,R4
ST  P4,Z
```

group →

```
L   P1,A / L   P2,B
L   P3,X / L   P4,Y / AR P2,P1
AR  P4,R4/ ST  P2,C
ST  P4,Z
```

Branch Prediction on z196

- The Branch Target Table remembers branches
  - BTB is indexed by part of the instruction address [halfword within 4K page]
  - Multiple states – *taken*, *strongly taken*, *not taken*, *strongly not taken*, *use PHT*
  - There is a Branch Pattern recording the last 12 branch directions (0/1)
  - A Pattern History Table is indexed by the Branch Pattern

Program Memory (halfwords)

Red "B"s are taken; Black "B"s are not taken

		B							
B					B				
		B						B	
	B					B			B
		B			B				
B				B					
				B					
						B			
			B					B	
B						B			

z/Architecture branch instructions and targets can be on any halfword
BTB has a row for each halfword in a page

Branch instruction address	Branch target address	History state
1015C	1016e	T/S
1028C	10310	NT/S
10290	102F2	T/W
.....	.....	
21032	2104E	NT/W
2108C	10028	PHT

Branch Target Table
2048 x 4
(indexed by 48-58 of IA)

Branch pattern

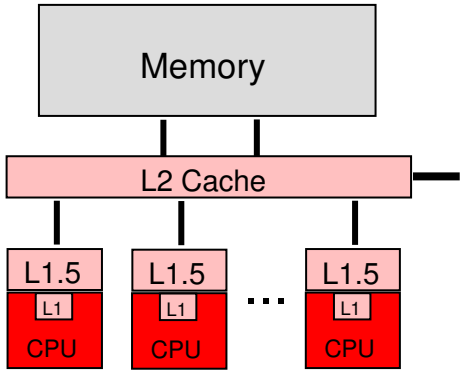
xxxxxxxxx

12-bits

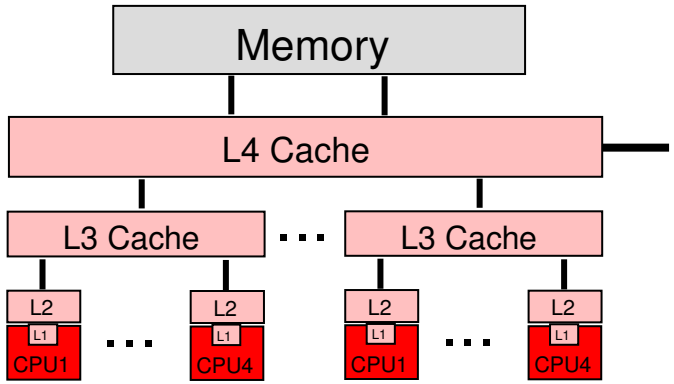
Pattern History Table
4096 enties,
2 bits wide, 4 states

z196 vs z10 hardware comparison

- z10 EC
  - CPU
    - 4.4 GHz
  - Caches
    - L1 private 64k i, 128k d
    - L1.5 private 3 MB
    - L2 shared 48 MB / book
    - book interconnect: star

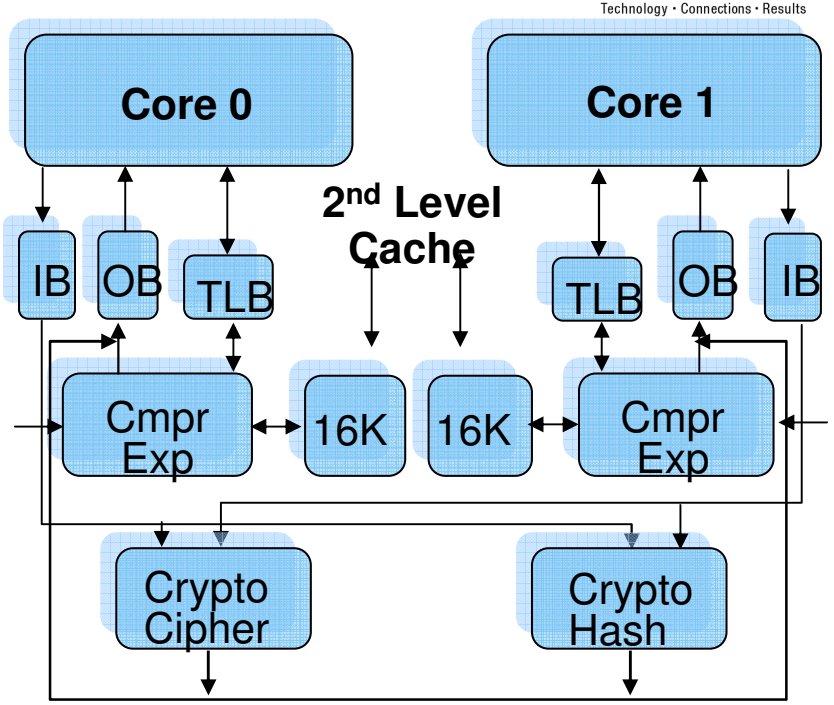


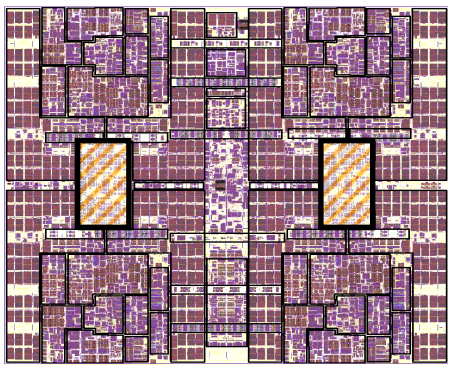
- z196
  - CPU
    - 5.2 GHz
    - Out-Of-Order execution
  - Caches
    - L1 private 64k i, 128k d
    - L2 private 1.5 MB
    - L3 shared 24 MB / chip
    - L4 shared 192 MB / book
    - book interconnect: star

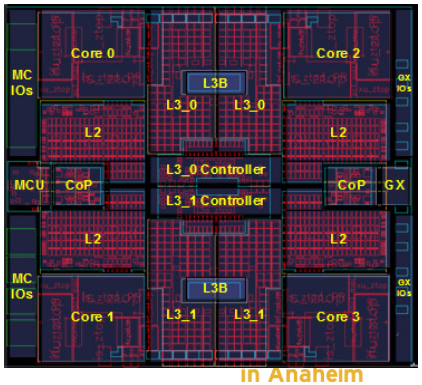


Compression and Cryptography Accelerator

- **Accelerator unit shared by 2 cores**
  - Independent compression engines
  - Shared cryptography engines
  - Co-operates with core millicode
  - Direct path into core store buffers
- **Data compression engine**
  - Static dictionary compression/expansion
  - Dictionary size up to 64KB (8K entries)
    - Local 16KB caches for dictionary data
  - Up to 8.8 GB/sec expansion
  - Up to 240 MB/sec compression
- **Cryptography engine**
  - 290-960 MB/sec bulk encryption rate
    - DES (DEA, TDEA2, TDEA3)
    - SHA-1 (160 bit)
    - SHA-2 (256, 384, 512 bit)
    - AES (128, 192, 256 bit)
- **Enhancements on z196**
  - Enhancements for new NIST standard
  - Complemented prior ECB and CBC symmetric cipher modes with XTS, OFB, CTR, CFB, CMAC and CCM
  - New primitives (128b Galois Field multiply) for GCM







z10 TLB2 and Large Pages

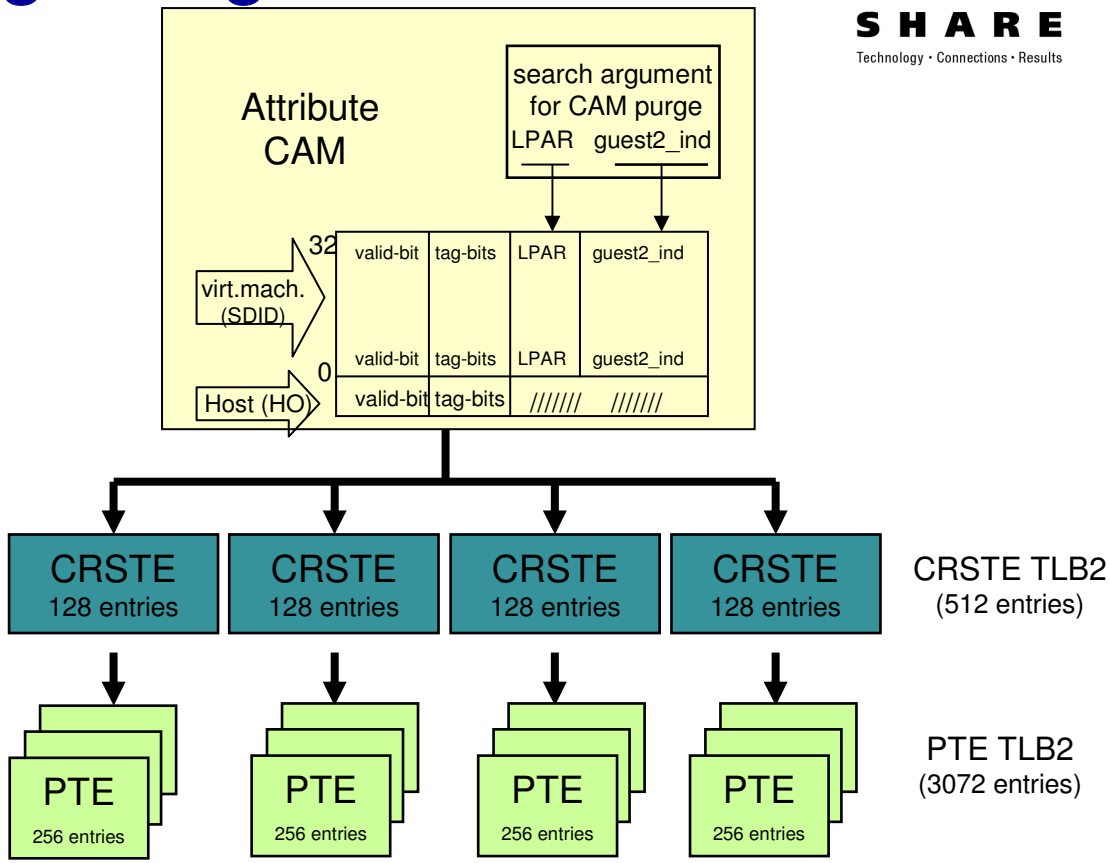
–TLB2 introduced in z990

–TLB2 contains *Combined Region and Segment Table Entries* (CRSTEs) and 4K pagetable entries

–TLB1 still contains only 4K entries

–CRSTEs are used to avoid accessing Region and Segment Tables but Page Table must still be accessed for 4K pages to create a TLB1 entry

–CRSTE can be used directly for 1MB pages to create a TLB1 entry



▪On z10, TLB1 misses on Large Pages that hit in TLB2 can be resolved without accessing a page table entry

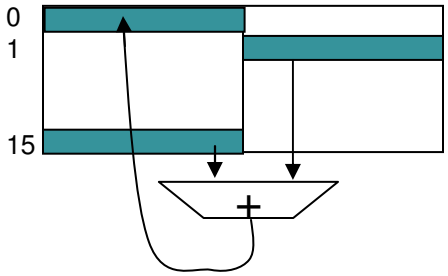
▪On z196, there is a separate TLB1 for 1MB entries so there is no need at all to create 4K entries for large pages

New Instructions on z10

- **Compare and Branch type**
  - To help on condition code limitation
- **Compare and Trap**
  - null pointer checks
- **Some new relative instructions**
  - Load Relative and Store Relative and “execute” relative
- **Immediate Instructions**
  - Move Immediate and compare immediate (16, 32, 64 bits)
  - Add Immediate (arithmetic and logical)
- **Fill necessary holes in latest architecture**
  - Some Multiply Immediate, some Multiply long displacement
- **Powerful bit manipulation instructions**
  - Rotate Then (AND, OR, XOR, INSERT) Bits

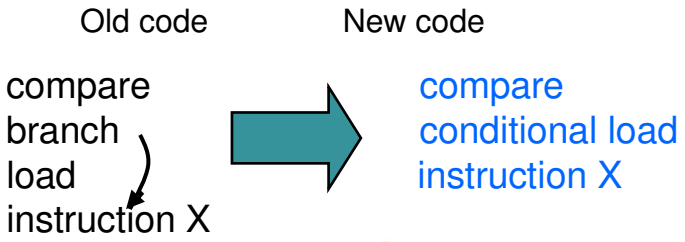
New Instructions on z196

- **High word extension (30 instructions)**
  - General register high word independently addressable
  - Gives software 32 word-sized registers
  - Add/subtracts, compares, rotates, loads/stores

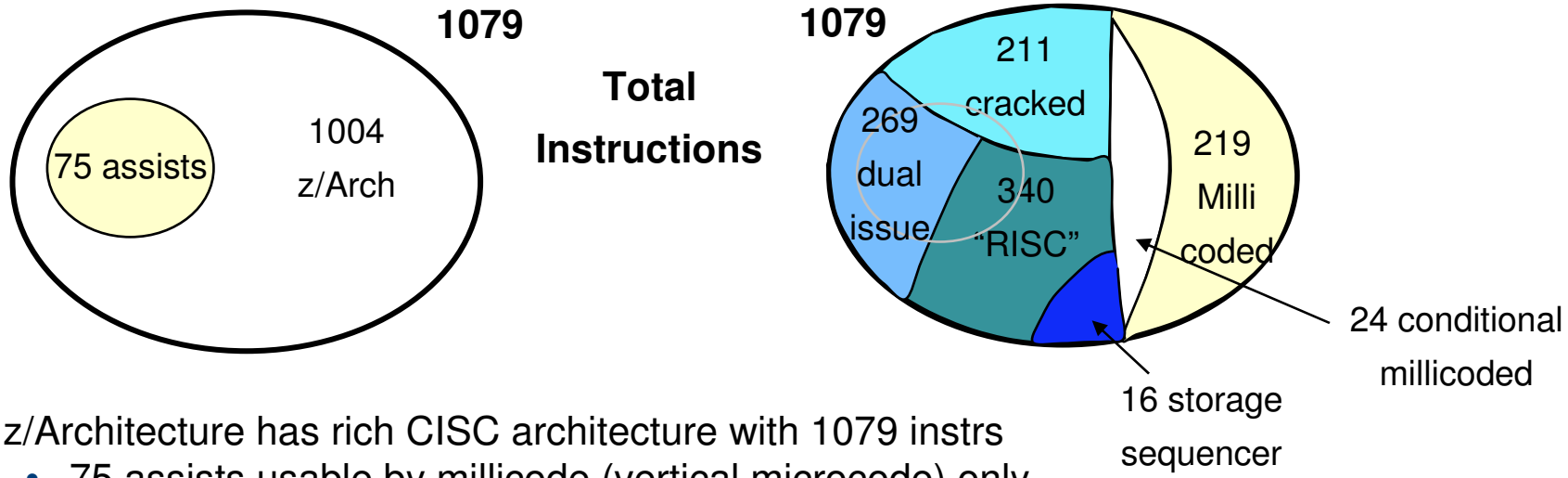


- **New atomic ops**
  - Load and “arithmetic” (ADD, AND, XOR, OR)
    - (Old) storage location value loaded into GR
    - Arithmetic result overwrites value at storage location
  - Load Pair Disjoint
    - Load from two different storage locations into even-odd register pair
    - Condition code indicates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

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



z196 Instruction Set Architecture Summary



- z/Architecture 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/Architecture instructions are RISC-like and map to single uop