System z Hardware Peeled: z/OS Hardware From the Inside Out

SHARE in Boston

Wednesday, August 4, 2010, 3 PM

Room 311 Hynes Convention Center

By Tom Harper

Product Architect
Neon Enterprise Software
Sugar Land, TX
tom.harper@neon.com

By Dave Kreiss

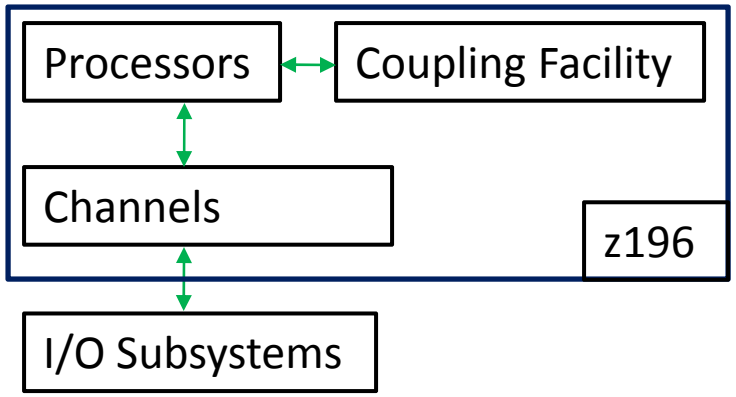
Product Architect
Neon Enterprise Software
Sugar Land, TX
david.kreiss@neon.com

References

- zArchitecture Principles of Operation - SA22-7832-07
- z990 Millicode in zSeries Processors - http://findarticles.com/p/articles/mi_qa3751/is_200405/ai_n9388162/
- IBM zEnterprise System Technical Guide - SG24-7833-00
- Coding Assembler for Performance (SHARE 107 Session 8192) David Bond - www.tachyonsoft.com/s8192db.pdf

What we are discussing

- Processors
- Channels
- I/O Subsystems
- Coupling Facilities



Going to be interesting, but not excessively technical…

Processors

- CPUs
- Main Storage
- Register Sets
- Interruptions
- DAT
- Clocks
- PER
- Packaging
- LPARs
- Documentation

Processors: CPUs

- One or more processors which can be “characterized” to become CPs, zIIPs, zAAPs, IFLs, CFs, or SAPs.
- All processors execute instructions.
- Instructions may be directly implemented by the hardware.
- On the z196, 246 instructions are implemented via millicode.
- Millicode written in a form of assembler language.
- Can only use directly implemented.
- Consists primarily of z/Architecture instructions and some specialized millicode instructions.
- Millicode provides flexibility.
- Did you know millicode is assembled using the Tachyon Assembler?

Processors: CPUs

Execution Unit Processing

PSW

Level 1 Instruction Cache Line

```
LAY   R0,Buffer
LHI   R1,L'Buffer
SR    R15,R15
MVCL  R0,R14
LAY   R1,Buffer
BRAS  R14,Build
```

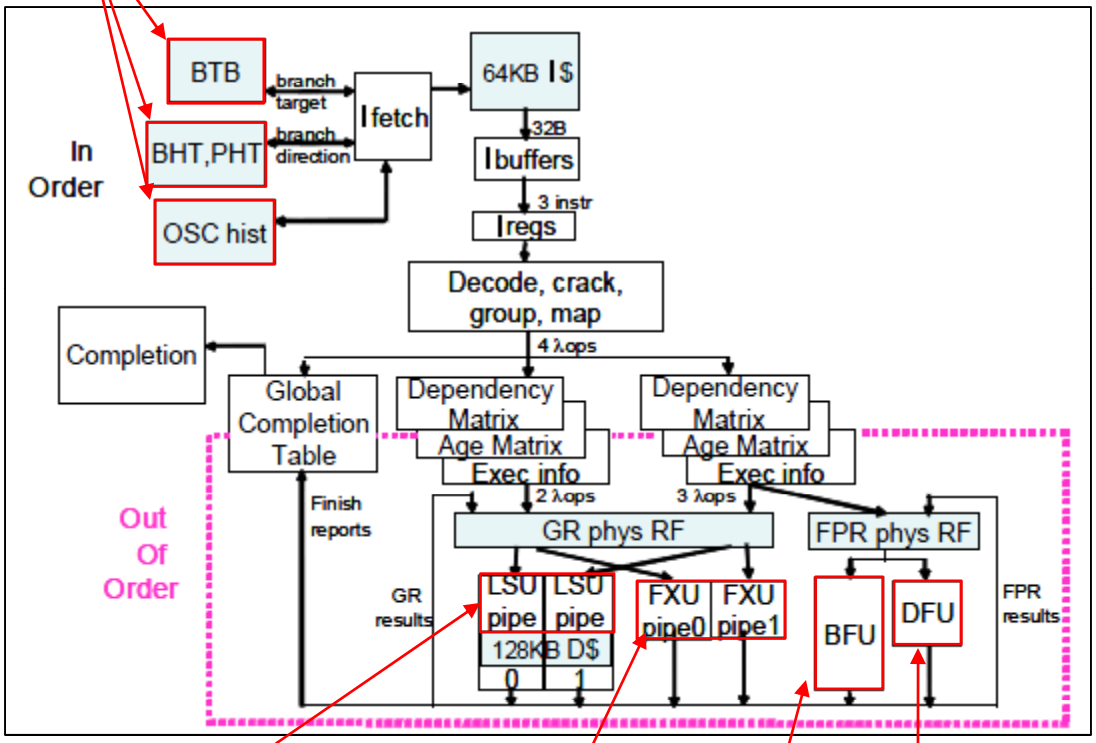
Processors: CPUs

- Each processor unit, or core, is a superscalar, out of order processor, having six execution units.
- Up to three instructions can be decoded per machine cycle.
- Up to five instructions can be executed per machine cycle.
- Branch prediction logic is used.
- Utilizes the Branch History Table (BHT) along with a Pattern History Table (PHT) and Branch Target Buffer (BTB).

Processors: CPUs

Execution Unit

Three Branch Tables



Load and Store Units

Fixed Point Units

Binary Floating Point Unit

Decimal Floating Point Unit

Processors: CPUs

- Millicode resides in protected storage: Hardware Aystem Area (HSA).
- Millicode brought into processor into instruction cache.
- Instruction unit fetches from cache, decodes them, calculates operand addresses, fetches operands, and sends them to execution unit.
- Execution unit executes and stores results.
- Result is relative simplicity.
- Seventy or so additional instructions known as milli-ops.
- **Appendix of all publicly documented millicode instructions is at end of presentation.**

Processors: CPUs

- Opcode of instruction is used as an index into the millicode section.
- Each routine has 128 bytes to perform its work.
- If 128 bytes is not enough, a branch is taken to another area.
- Prior to millicode execution, some setup occurs.
- Instruction text saved in special register for millicode.
- Flags set.
- Some addresses are calculated.

Processors: Main Storage

- Main Storage is accessed by CPUs and by channels.
- Storage divided into 4K frames.
- Key-controlled for fetch and store.
- Storage can also be page protected.
- Low-Address Protection.
- Read and update references are recorded.
- Prefixing allows each CPU to separate 8K of low-addresses.

Processors: Main Storage

Main storage Protection

Key:0 Store	Key:0 Fetch	Key:0 Store	Key:8 Store	Key:8 Store	Key:8 Store
Key:8 Fetch	Key:8 Fetch		Key:0 Store	Key:2 Fetch	Key:0 Fetch
Key:8 Fetch	Key:8 Fetch				
Key:8 Fetch		Key:8 Fetch	Key:8 Fetch	Key:8 Fetch	Key:8 Fetch
	Key:8 Fetch				
	Key:0 Store	Key:0 Fetch	Key:0 Fetch	Key:4 Fetch	Key:7 Fetch
	Key:0 Fetch	Key:0 Store	Key:0 Store	Key:0 Fetch	Key:0 Store

Page protected

Processors: Main Storage

Prefixing

CPU 0	CPU 1
Prefix Register:00000000	Prefix Register:00008000

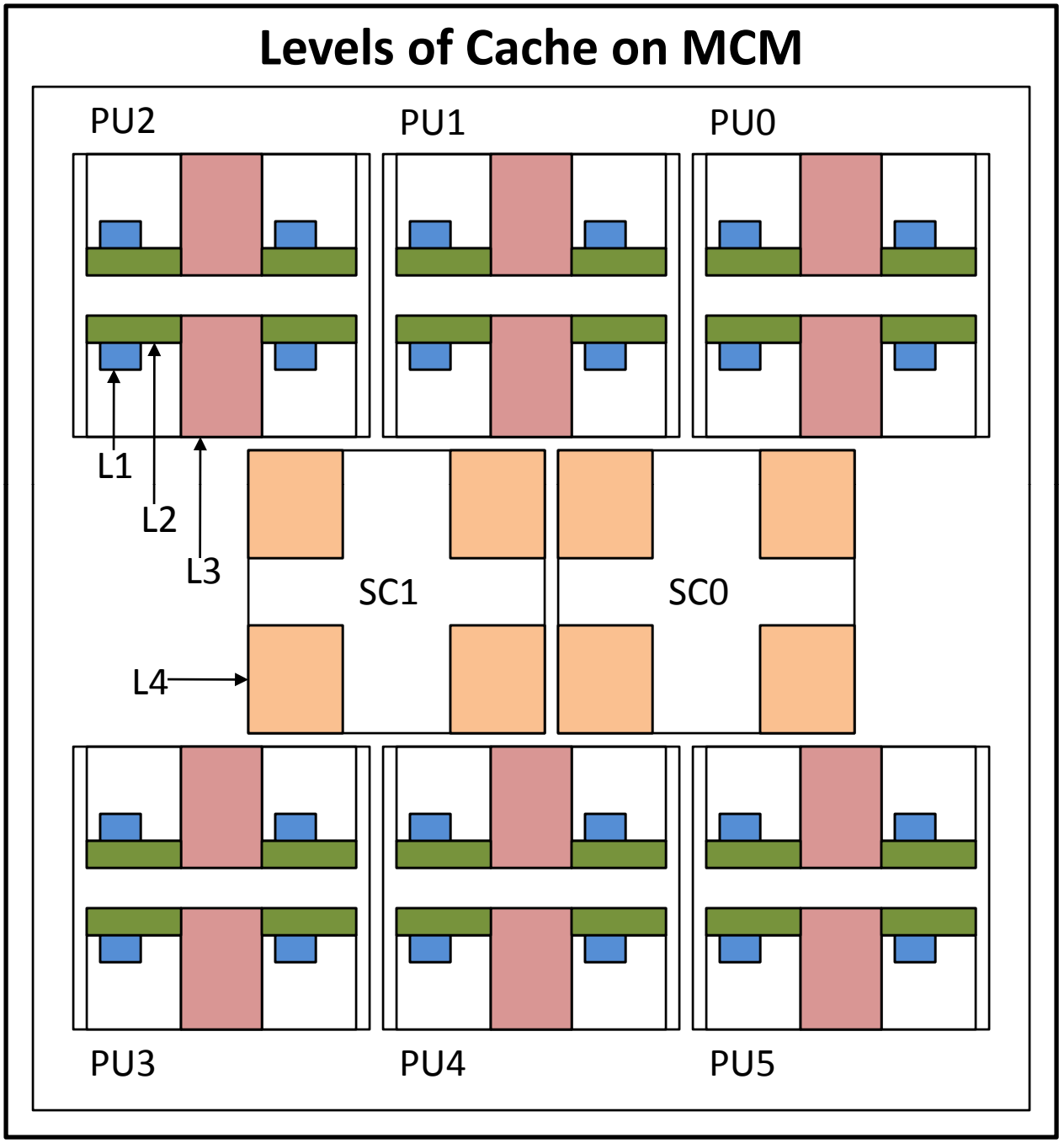
Real address:00000000
Virtual address:00000000
Absolute address:00000000

Real address:00000000
Virtual address:00000000
Absolute address:00008000

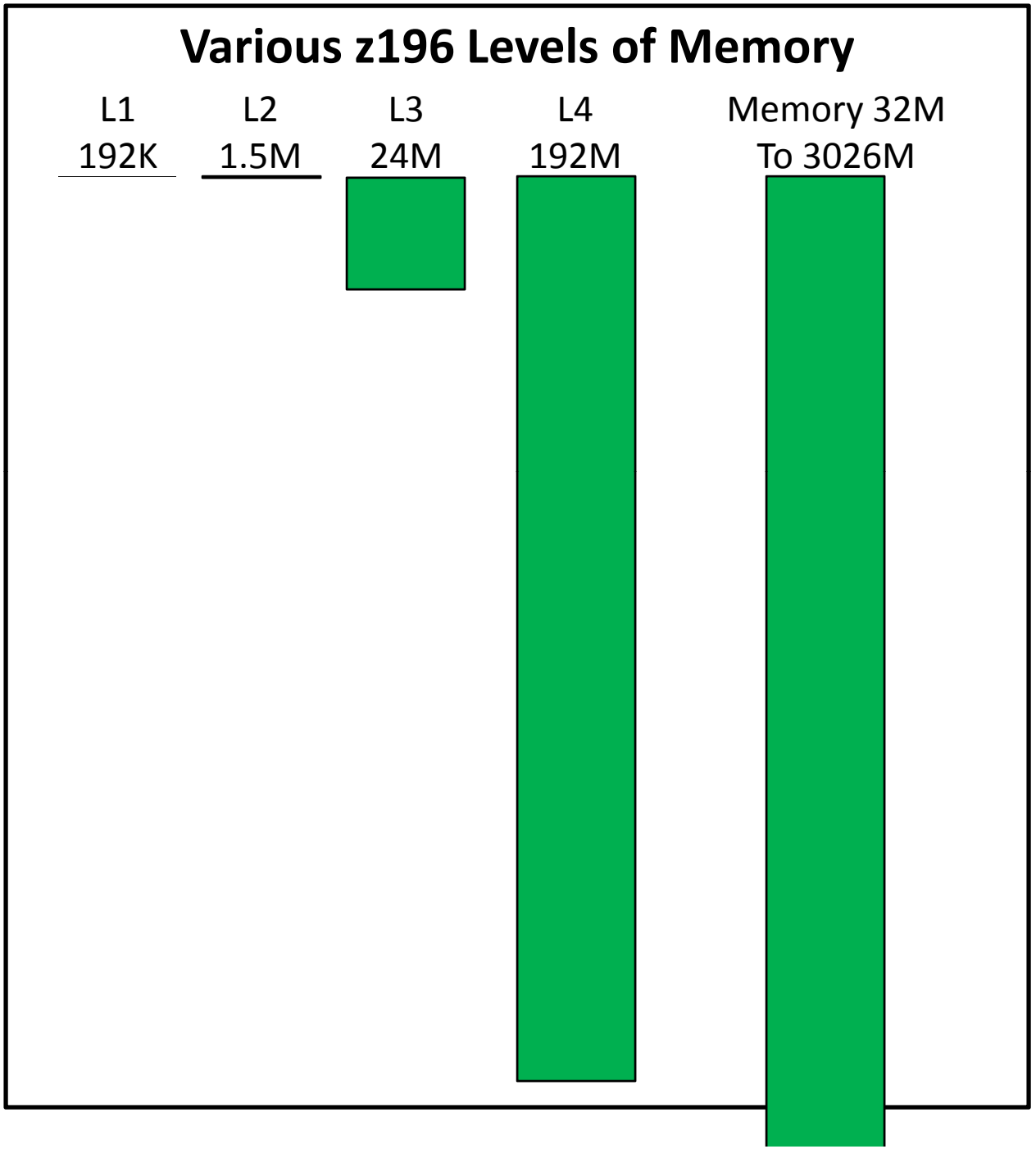
Processors: Main Storage

- CPUs do not access main storage directly as do channels.
- Between CPUs and main storage are multiple levels of cache (four on z196, three on z10).
- Cache is high-speed storage.
- Each level of cache is more expensive and faster than the level below it.
- Data is moved between main storage and cache levels in 256-byte chunks, called cache lines.
- CPU can normally only access data in highest level of cache.
- MVCL can bypass cache.

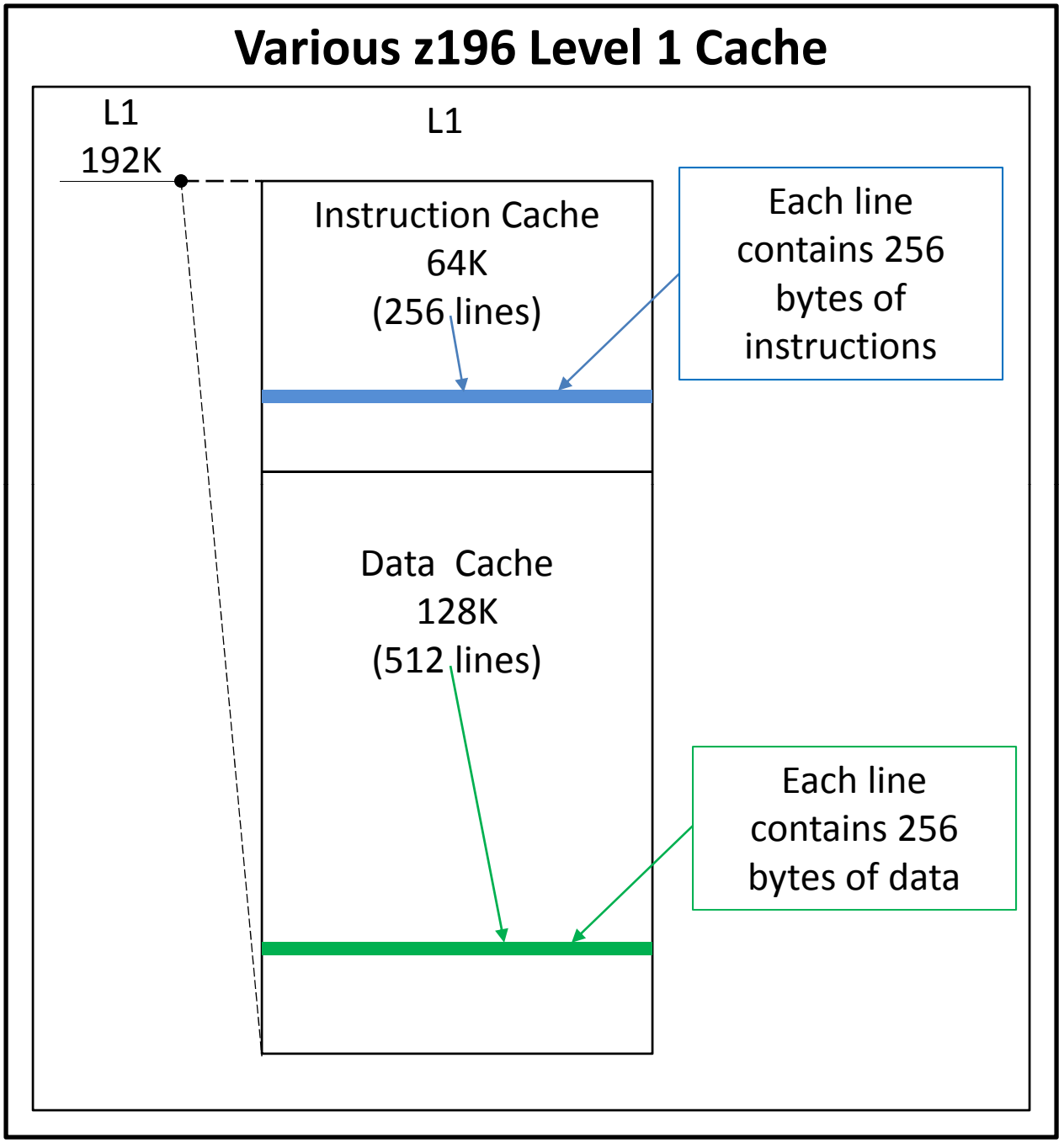
Processors: Main Storage



Processors:
Main Storage



Processors: Main Storage



Processors: Main Storage

- Two flavors of cache: data cache and instruction cache.
- Data Cache is used for fetching as well as storing of data.
- Cache use is controlled on a LRU basis by the processor, along with advice given by programs which pre-fetch cache lines.
- Cache “misses” are expensive: up to 600 cycles and more.
- Instruction count is becoming less important than cache misses in today’s processors.

Processors: Main Storage

Non-synchronized storage update

Task 1 on CP0

```
    OI   Flag,SETON
     .
     .
Interrupt
     .
     .
Dispatched
     .
    TM   Flag,SETON
```

Task 2 on CP1

```
     .
     .
     .
 Dispatched
     NI   Flag,255-SETON
     .
     .
     .
     .
```

Is Flag seen as containing the SETON value by task1 at time of TM?

Processors: Main Storage

- Programming Hints.
- Sequence of storage references:
  - Conceptual sequence
  - Actual sequence
- Synchronizing events.
- With very large caches, we have seen code that apparently used to work on older machines but failed on newer machines.
- Programmers had depended on storage consistency that really was not valid, but exposed by longer memory latency times.

Processors: Register Sets

- General Registers: 16 x 64 bits wide.
- Access Registers: 16 x 32 bits wide.
- Floating-Point Registers: 16 x 64 bits.
- Control Registers: 16 x 64 bits wide.
- Millicode General Registers (MGRs).
- Additional millicode registers
  - MARs (Millicode Access Registers)
  - G1CR and G2CR (Guest Control Registers)
  - SYSR (System Registers)
- MGRs used to hold intermediate results and address customer storage or data in hardware system area.
- When millicode is running, only MGRs are updated.
- Separate millicode PSW is used
- Millicode can branch.

Processors: Register Sets

Processor Registers

General Purpose Registers

R0	R1	R2	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14	R15

Access Registers

AR0	AR1	AR2	AR4	AR5	AR6	AR7	AR8	AR9	AR10	AR11	AR12	AR13	AR14	AR15

Floating Point Registers

FPR0	FPR1	FPR2	FPR4	FPR5	FPR6	FPR7	FPR8	FPR9	FPR10	FPR11	FPR12	FPR13	FPR14	FPR15

Control Registers

CR0	CR1	CR2	CR4	CR5	CR6	CR7	CR8	CR9	CR10	CR11	CR12	CR13	CR14	CR15

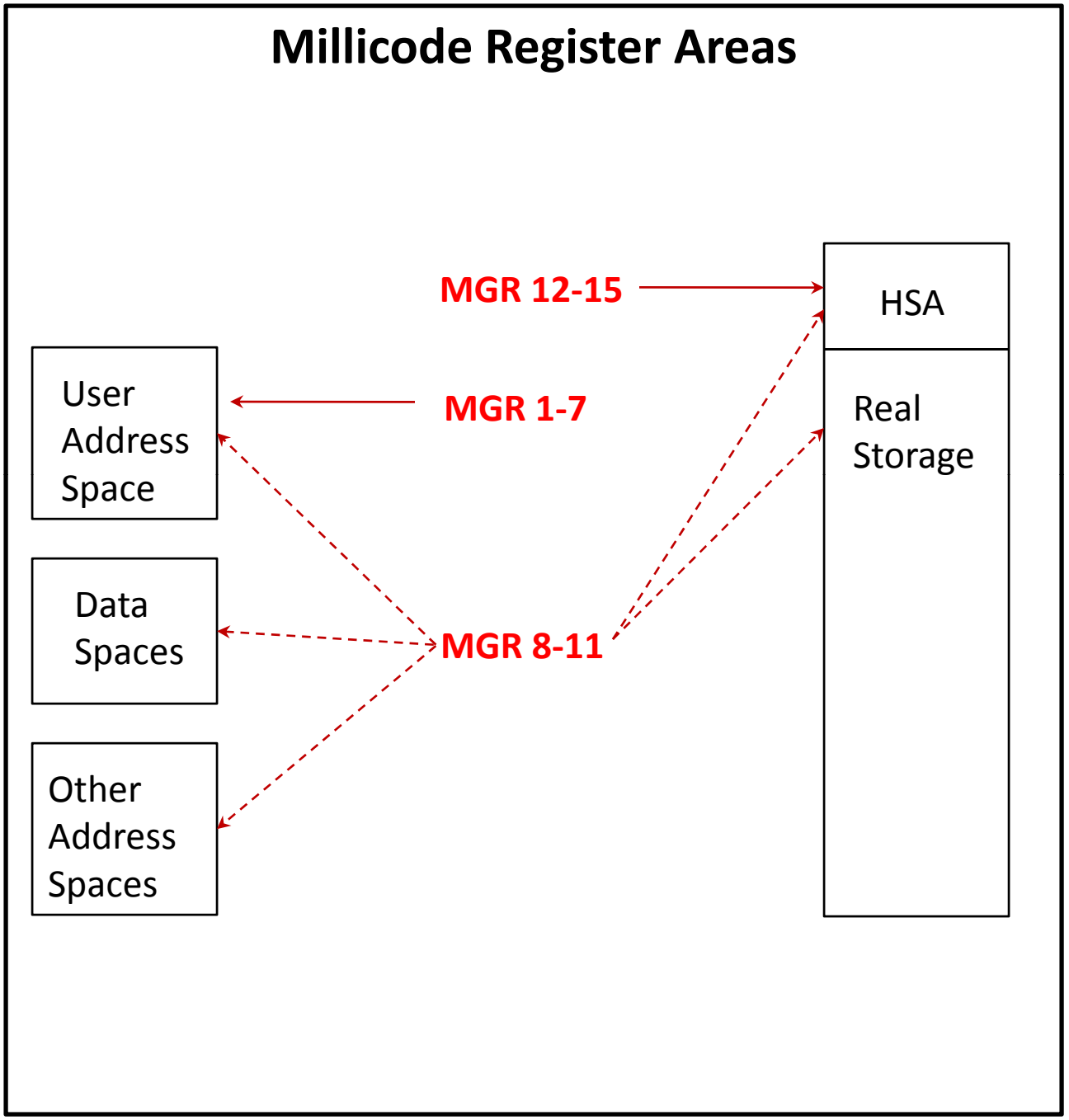
Millicode General Registers

MGR0	MGR1	MGR2	MGR4	MGR5	MGR6	MGR7	MGR8	MGR9	MGR10	MGR11	MGR12	MGR13	MGR14	MGR15

Processors: Register Sets

- When MGRs 1-7 are used, program storage is accessed.
- When MGRs 12-15 are used, hardware system area is accessed.
- When MGRs 8-11 are used, they are paired with four special OACRs.
- Operand Access Control Registers indicate key, ASC (primary, secondary, home, or access), AMODE, ATYPE (real, virtual, host real, absolute, hardware system area).
- OACRs can block PER.

Processors: Register Sets



Processors: Interruptions

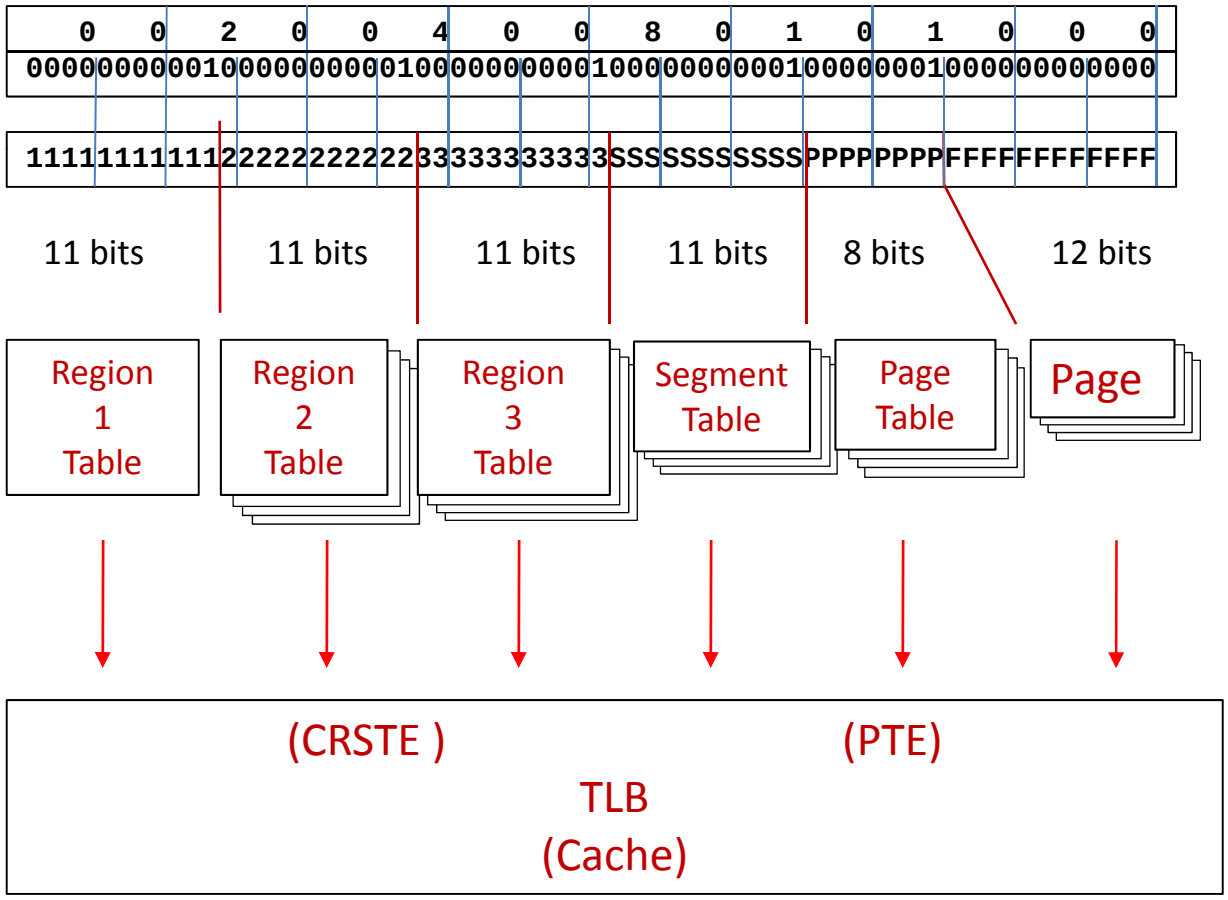
- When an interruption is taken, control is routed to millicode for processing.
- No setup is done as for an instruction.
- Type of interruption is decoded.
- Program Status Words (PSWs) are swapped by millicode.
- Interruption is reset.
- Processing continues with next instruction specified by PSW.

Processors: DAT

- Processor hardware performs DAT and stores result in translation look-aside table (TLB).
- Converts virtual address to real
- Virtual may be primary, secondary, AR, or home.
- Results are stored in either instruction cache or data cache.
- Not available for channels (which is why CCWs and IDAWs must be page-fixed).
- CR1 points to Primary Region-Table.
- CR7 points to Secondary Region-Table.
- CR13 points to Home Region-Table.

Processors: DAT

Mapping of a 64 bit address



Processors: DAT

- PSW bits control which table is used
- Appropriate segment and page-tables are also referenced.
- PURGE TLB can clear the TLB.
- Selected entries can also be cleared.
- Large page support (1 Mb frames) reduces entries in TLB and simplifies translation (no page tables are needed).
- In addition to TLB, there is an ALB for references to data in data spaces.

Processors: Clocks

- Time-of-day (TOD) clock.
- All CPUs share the same clock.
- Runs continuously.
- Controlled by SETCLOCK.
- Queried by Store Clock (Extended).
- TOD programmable register appended to Store Clock value.
- Can be controlled by External Time Reference (ETR).
- Clock Comparator can be used to cause an interruption.

Processors: Clocks

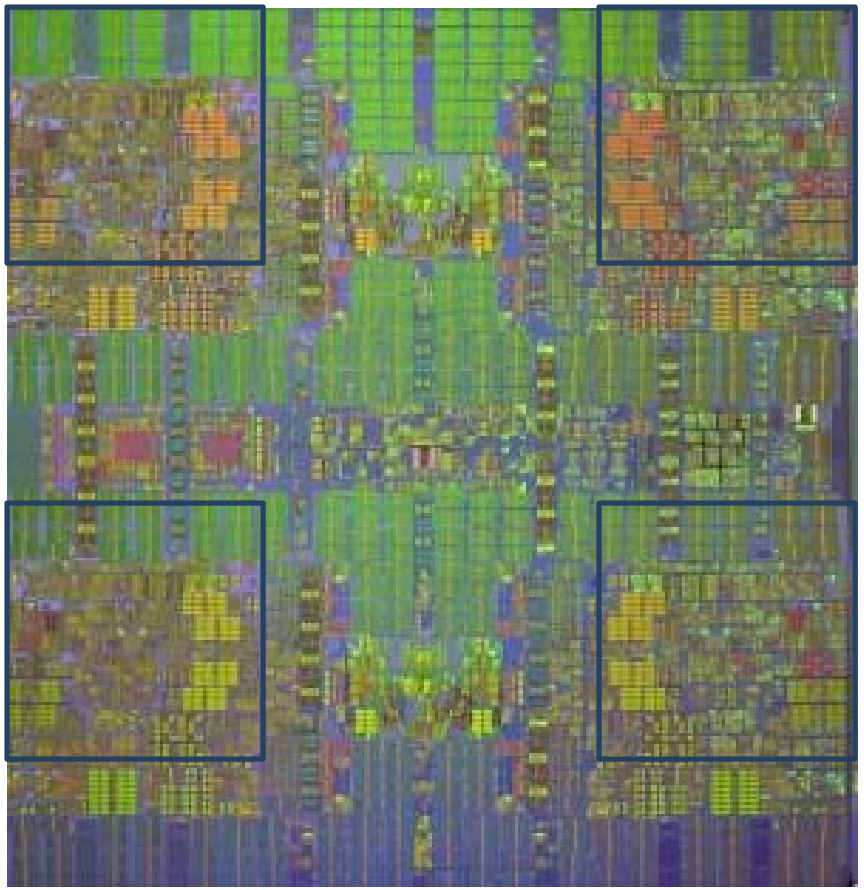
- Each CPU has a separate CPU timer.
- Can be used for measuring elapsed CPU time and for causing an interruption.

Processors: PER

- Program Event Recording (PER).
- Purpose to help debug programs.
- Can use hardware to detect the following:
  - Execution of successful branch
  - Fetching of an instruction
  - Altering of a storage area
- All of the above can work with a range of storage addresses.
- Controlled by CRs 9, 10, and 11 and a bit in the PSW.
- Causes a PER interruption.
- Implemented via SLIP in z/OS.

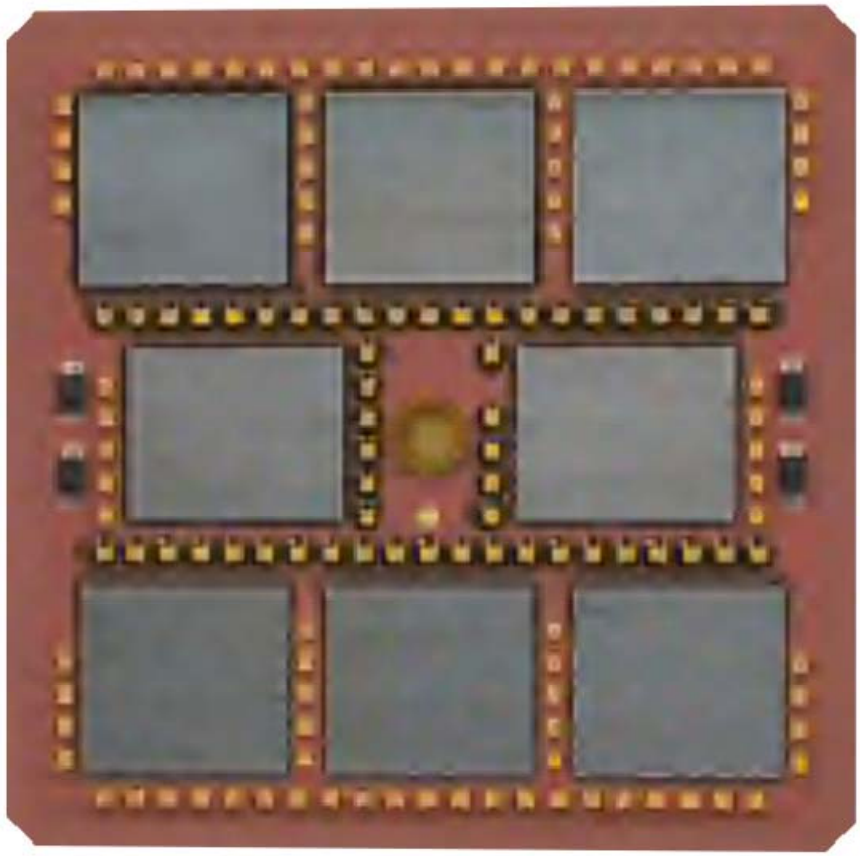
Processors: Packaging

- Processors are built into PUs. The number of processors on a PU varies by the model. Here is what one with four processors looks like:



Processors: Packaging

- PUs are combined onto another piece of hardware called a Multi-Chip Module, or MCM. Here is a photograph of a z196 MCM with six PUs and two storage chips in place:



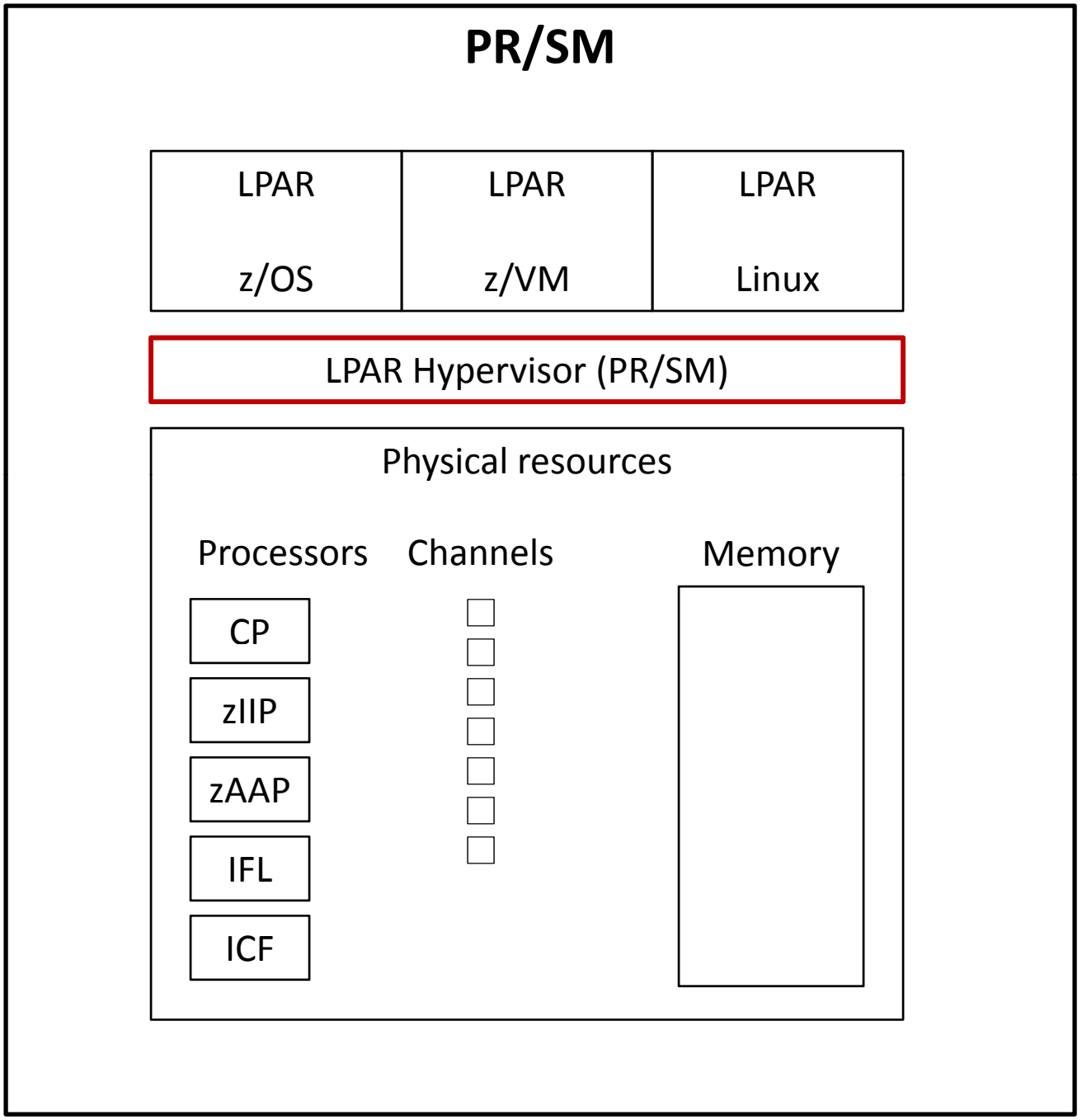
Processors: Packaging

- MCMs for the z10 contained 107 layers and almost one billion transisters; for the z196 103 layers and 11 billion transistors.
- Two levels of instruction and data cache are also present on the MCM.
- Compression and crypto processors are also on the MCM.
- Only a single MCM is needed, but up to four MCMs can be present. Each MCM is called a book. Combined with memory, channels, power supply, etc., they form the zEnterprise processor on the left…

Processors: LPARs

- For some time now, machines can only run in LPAR mode.
- LPARs are controlled by Processor Resource / Systems Manager, PR/SM (also known as a hipervisor).
- PR/SM is just code that controls dispatching of processors between LPARs.
- Processors can be dedicated or shared across LPARs.
- Memory is dedicated to LPARs.
- Channels may be shared across LPARs .

Processors: LPARs

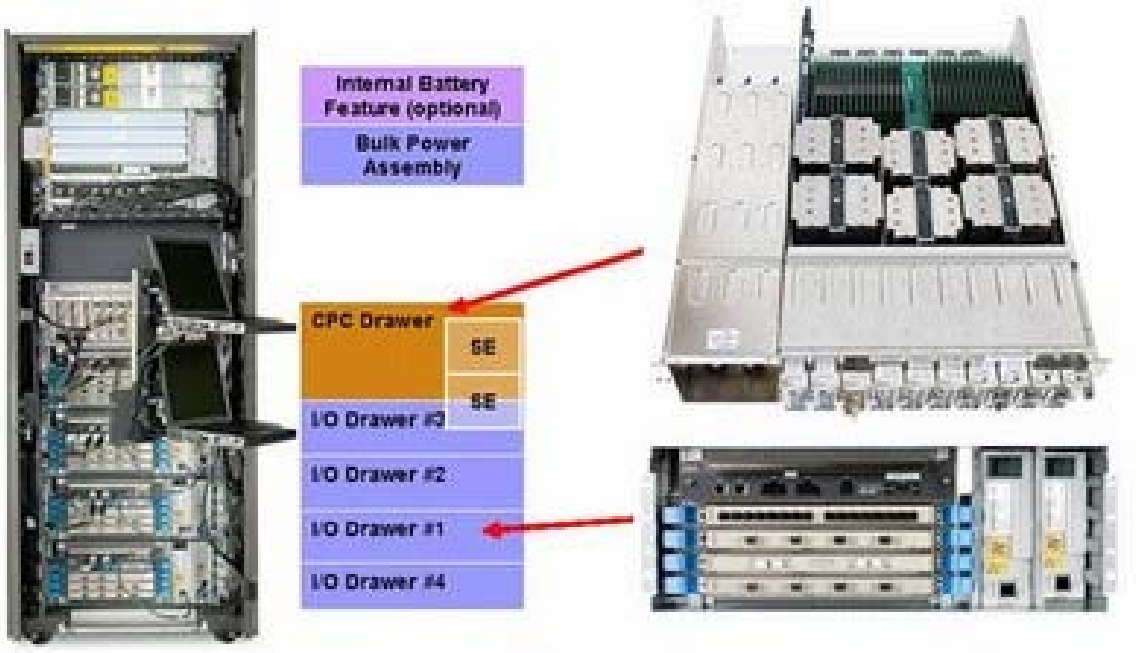


Processors: Documentation

- zArchitecture Principles of Operation.
- Author is Dan Greiner, who is here at SHARE this week. Introduce yourself!
- Did you know: Dan Greiner used to work for Amdahl (arch-rival of IBM in the twentieth century).
- Did you know: The public version of the zArchitecture Principles of Operation is only a subset of the complete document?
- Did you know: The two most complex publicly documented instructions are the CFC and UPT? Eight and seven pages, respectively.

Channels

- How Channels work
- How Channels are connected
- How Channels initiate and terminate I/O operations



Channels

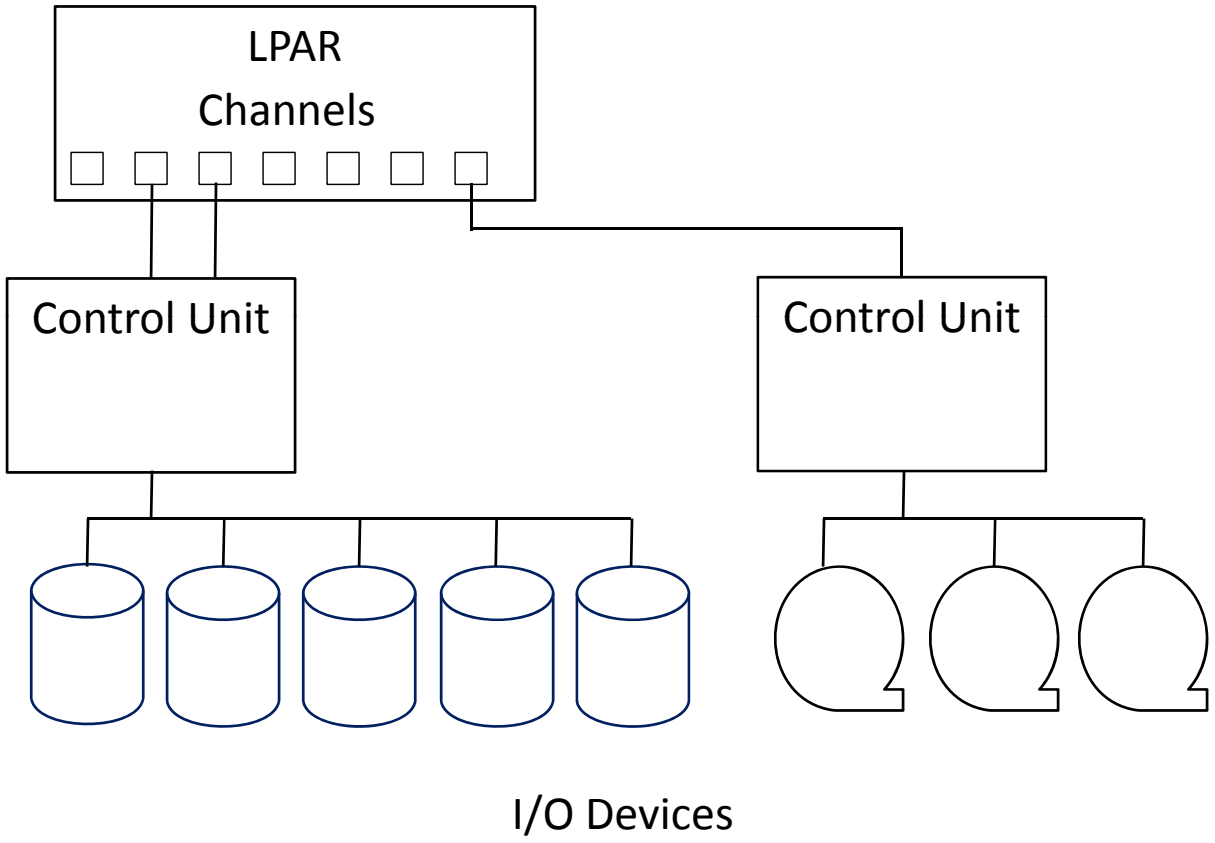
- Channels used to be stand-alone boxes.
- They are now packaged within the processor complex.
- Channels direct flow of information between I/O devices and main storage.
- Channels work independently of CPUs in order to achieve higher throughputs.
- Channels select a channel path to manage flow of information.
- Subchannels are dedicated to each device.

Channels

- I/O devices are connected to subchannels via control units.
- Control units may have multiple channel paths.
- I/O devices may be connected to multiple control units.
- I/O operations are initiated by the processor via the Start Subchannel instruction.
- When an I/O operation completes, an I/O interruption occurs on a processor which is not masked off for that interruption.

Channels

I/O Component Relationship

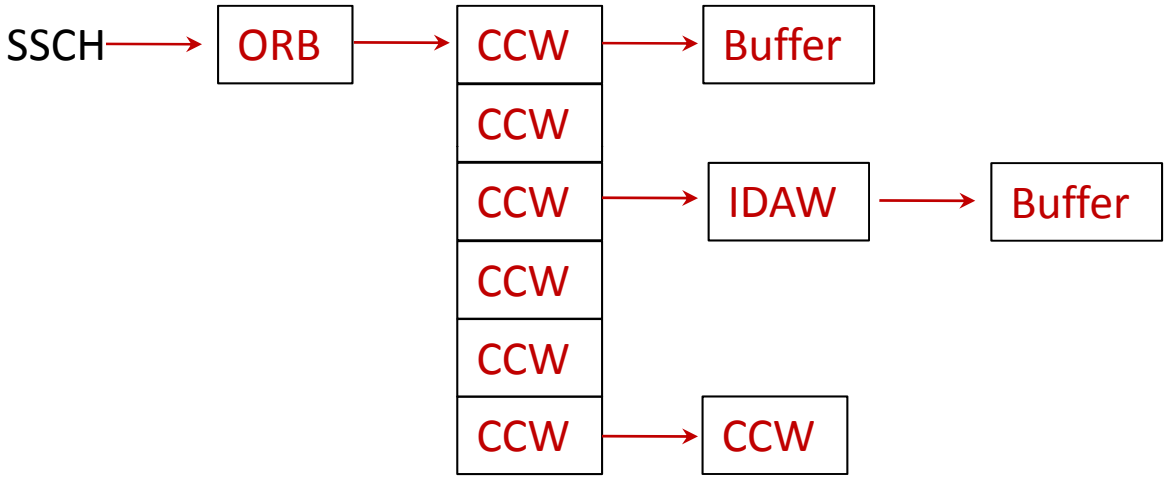


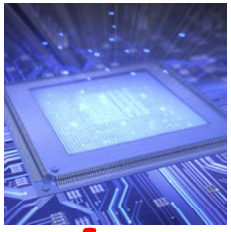
Channels

- The SSCH instruction passes the address of an Operation Request Block (ORB) to the subchannel.
- The ORB contains the address of the first CCW to be executed and their format.
- The subchannel fetches the CCW and commences execution.
- The subchannel selects a path. If none is available, it waits.
- The CCWs are executed.
- When complete, an interruption is made pending.
- Transport Command Words (TCWs) are available recently.

Channels

I/O Request Components





I/O Subsystems

- Disk-based systems
- Tape-based systems

TS7700 Virtual Tape

DS8700 Disk Storage

I/O Disk

- Implemented via Redundant Array of Independent Disks (RAID).
- Various forms of RAID, RAID 0, RAID 1, etc.
- RAID consists of a number of physical disks and solid state memory, along with cache and software.
- Three key RAID concepts:
  - Mirroring
  - Striping
  - Error Correction
- Data may be read/written to/from multiple physical disks at same time.
- Hot swapping of failed drives.

I/O Disk

- RAID arrays permit virtualization of DASD.
- Caching allows bypassing of reading and writing data.
- Caching algorithms based on LRU and detection of sequential processing.
- Multiple CPUs can manage a RAID array.
- RAIDs can be partitioned to function with different RAID levels.
- Can provide transparent compression.
- Battery backup to allow flushing of cache.

I/O Disk

- RAID architecture has allowed new concepts to exist such as Concurrent Copy.
- Did you know? On solid state drives, certain data is read more often than other data, so RAID software moves that around to prevent “wearing out” the solid state memory.

I/O Tape

- Individual tape drives have been replaced by tape library systems.
- These systems automatically fetch and store tape volumes to/from tape drives.
- Data is written in parallel on one side of tape, and then written on the other side so that when volume is full, no re-wind is required.
- Data is re-blocked internally to an optimum size from what you blocked it as, and compressed.

I/O Tape

- Tape storage systems themselves are being replaced by disk-based systems which emulate tape.
- Typically consist of a tiered combination of both disk drives and tape drives which appear as tape to end used.
- Speed is improved by disk drive emulation of tape.

Coupling Facilities

- Coupling Facilities are just one or more characterized PUs that are dedicated to being a coupling facility.
- A Coupling Facility is an architected method of connecting together multiple LPARs.
- Special Licensed Internal Code (LIC) is loaded into the CF in order to perform the CF functions. It is just ordinary assembler code (created using PL/S) like any other operating system code. It is also known as Coupling Facility Control Code (CFCC).
- Coupling Facilities have memory but no channels are defined.

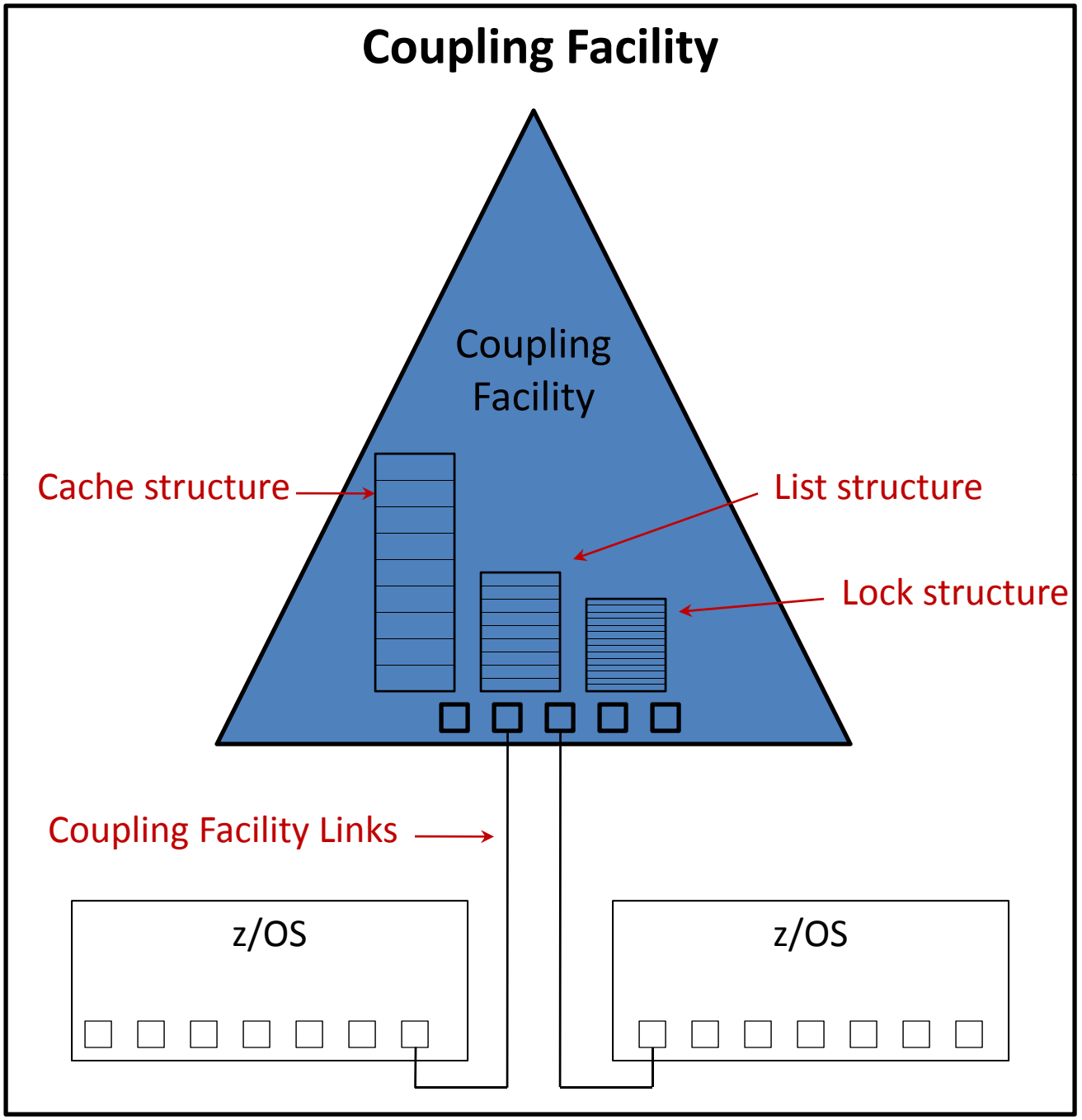
Coupling Facilities

- Instead of channels, Coupling Facilities are connected to other LPARs using Coupling Facility Links.
- Multiple Coupling Facilities are permitted.
- There are eighteen special instructions for supporting it (part of the undocumented zArchitecture).
- There is no virtual memory; only real memory (since no paging).
- Coupling Facilities contain structures, which may be of three types:
  - Lock
  - Cache
  - List

Coupling Facilities

- Structures may be duplexed across multiple Coupling Facilities.
- There is an API to the Coupling Facility, but it is complex and is for authorized code only.
- Coupling Facility is used by many z/OS components, such as GRS, IMS, JES, etc.
- Coupling Facility is significant because non-mainframe architectures do not have a comparable architecture to allow locking and sharing of data which scales effectively.

Coupling Facilities



Appendix A

All publicly-available millicode instructions:

- AMCLHI – And MCR LEFT
- ASR – And Special Register
- BRFLG – Branch on Flags
- BRS – Branch Relative Special
- CCSB – Compare String Bytes
- CLRNG – Compare Logical for Range Check
- DIP – Drain Instruction Pipeline
- DPFET – Divide Decimal Fetch
- DPQUO – Divide Decimal Quotient
- DPSTO – Divide Decimal Store
- EDBYT -
- EXAR – Extract Access Register
- EXARI – Extract Access Register Indirect
- EXGRI - Extract Program GR Indirect
- EXINT - Extract Interrupt
- EXTV - Extract Via ROR
- FBE – Find Byte Equal
- FBED - Find Byte Equal Double
- FBN - Find Byte Not Equal
- FBND - Find Byte Not Equal Double
- LFLG – Load Flags
- LTA - Load and Test Access
- MCEND – Millicode End
- MSET – Set Via ROR
- OSR – Or Special Register
- PXLO - Perform Translator Operation
- RBD – Replicate Byte Double
- RCR – Read Control Register
- RDFLG – Reset Flags
- RIRPT - Reset Interruption
- RSR – Read Special Register
- SFLG – Set Flags
- SPGRI - -Set Program GR Indirect
- TMBP – Test Millicode Branch Points
- TPSW – Test PSW
- TRFET – Translate Fetch
- TRTX – Translate and Test Ending
- WSR – Write Special Register
- WSRS – Write Special Register Short

Appendix A Page 1

- **AMCLHI – And MCR LEFT**

  (5802359 - Mapping processor state into a millicode addressable processor state register array)
- **ASR – And Special Register**

  (5802359 - Mapping processor state into a millicode addressable processor state register array)
- **BRFLG – Branch on Flags**

  (6055623 - Specialized millicode instruction for editing functions)
- **BRS – Branch Relative Special**

  (z990 Millicode in zSeries Processors – heller)
- **CCSB – Compare String Bytes**

  (5611062 - Specialized millicode instruction for string operations)

Appendix A Page 2

- **CLRNG – Compare Logical for Range Check**

  (5621909 - Specialized millicode instruction for range checking)
- **DIP – Drain Instruction Pipeline**
- **DPFET – Divide Decimal Fetch**

  (6055623 - Specialized millicode instruction for editing functions)
- **DPQUO – Divide Decimal Quotient**

  (6055623 - Specialized millicode instruction for editing functions)
- **DPSTO – Divide Decimal Store**

  (6055623 - Specialized millicode instruction for editing functions)
- **EDBYT -**

  (6055623 - Specialized millicode instruction for editing functions)

Appendix A Page 3

- **EXAR – Extract Access Register**

  (5713035 - Linking program access register number with millicode operand access)
- **EXARI – Extract Access Register Indirect**

  (5713035 - Linking program access register number with millicode operand access)
- **EXGRI - Extract Program GR Indirect**

  (z990 Millicode in zSeries Processors – heller)
- **EXINT - Extract Interrupt**

  (z990 Millicode in zSeries Processors – heller)
- **EXTV - Extract Via ROR**

  (5226164 - Millicode register management and pipeline reset)

Appendix A Page 4

- **FBE – Find Byte Equal**

  (5611062 - Specialized millicode instruction for string operations)
- **FBED - Find Byte Equal Double**

  (5611062 - Specialized millicode instruction for string operations)
- **FBN - Find Byte Not Equal**

  (5611062 - Specialized millicode instruction for string operations)
- **FBND - Find Byte Not Equal Double**

  (5611062 - Specialized millicode instruction for string operations)
- **LFLG – Load Flags**

  (6055623 - Specialized millicode instruction for editing functions)

Appendix A Page 5

- **LTA - Load and Test Access**

  (5694587 - Specialized millicode instructions for test PSW validity, load with access test, and character translation assist)
- **MCEND – Millicode End**

  (z990 Millicode in zSeries Processors – heller)
- **MSET – Set Via ROR**

  (5226164 – Millicode register management and pipeline reset)
- **OSR – Or Special Register**

  (z990 Millicode in zSeries Processors – heller)
- **PXLO - Perform Translator Operation**

  (z990 Millicode in zSeries Processors – heller)

Appendix A Page 6

- **RBD – Replicate Byte Double**

  (5611062 - Specialized millicode instruction for string operations)
- **RCR – Read Control Register**

  (5802359 - Mapping processor state into a millicode addressable processor state register array)
- **RDFLG – Reset Flags**

  (6055623 - Specialized millicode instruction for editing functions)
- **RIRPT - Reset Interruption**

  (z990 Millicode in zSeries Processors – heller)
- **RSR – Read Special Register**

  (5802359 - Mapping processor state into a millicode addressable processor state register array)

Appendix A Page 7

- **SFLG – Set Flags**

  (6055623 - Specialized millicode instruction for editing functions)
- **SPGRI - -Set Program GR Indirect**

  (z990 Millicode in zSeries Processors – heller)
- **TMBP – Test Millicode Branch Points**

  (6662296 - Method and system for testing millicode branch points)
- **TPSW – Test PSW**

  (5694587 - Specialized millicode instructions for test PSW validity, load with access test, and character translation assist)
- **TRFET – Translate Fetch**

  (5694587 - Specialized millicode instructions for test PSW validity, load with access test, and character translation assist)

Appendix A
Page 8

- **TRTX – Translate and Test Ending**

  (6055623 - Specialized millicode instruction for editing functions)
- **WSR – Write Special Register**

  (5802359 - Mapping processor state into a millicode addressable processor state register array)
- **WSRS – Write Special Register Short**

  (5802359 - Mapping processor state into a millicode addressable processor state register array)

That's all!

Whew!