Cell Broadband Engine Architecture

School of Electrical Engineering and Computer Science
University of Central Florida

Cell History

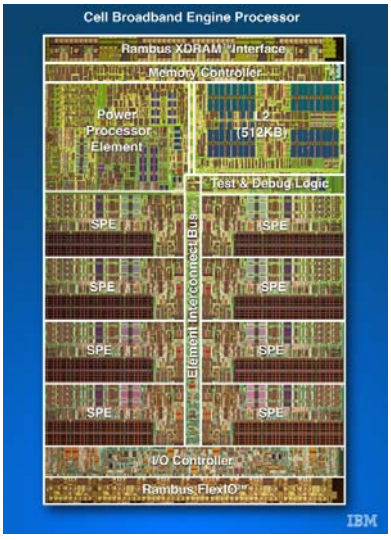
- IBM, Sony Computer Entertainment Incorporated SCEI/Sony, Toshiba Alliance (STI) formed in 2000.
- The objectives for the new processor were the following:
  - Outstanding performance, especially on game/multimedia applications.
  - Real-time responsiveness to the user and the network.
  - Applicability to a wide range of platforms.
  - Support for introduction in 2005.
- First technical disclosures, 2005.
- Platforms: PS3, Cell Blade, Cell Accelerator Board
- Trademarks
  - Cell Broadband Engine and Cell Broadband Engine Architecture are trademarks of Sony Computer Entertainment, Inc.

Overview of the Cell Broadband Engine

- One PowerPC Processor Element (PPE) + Eight Synergistic Processor Element (SPE).
- Increased efficiency and performance:
  - Attacks on the "Power Wall"
    - Non Homogenous Coherent Multiprocessor
    - High design frequency @ a low operating voltage with advanced power management
      - 45nm: 0.8v, 50w, 3.2GHz.
  - Attacks on the "Memory Wall"
    - Streaming DMA architecture
      - 128 simultaneous transfers between the eight SPE local stores and main storage.
    - 3-level Memory Model: Main Storage, Local Storage, Register Files
  - Attacks on the "Frequency Wall"
    - Highly optimized implementation
    - Large shared register files and software controlled branching to allow deeper pipelines

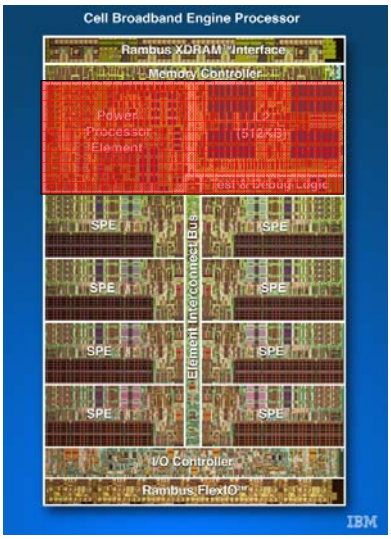
Highlights

- 90nm
- 3.2 GHz
- 241M transistors
- 9 cores, 10 threads
- >200 GFlops (SP)
- > 20 GFlops (DP)
- Up to 25 GB/s memory B/W
- Up to 75 GB/s I/O B/W
- >300 GB/s EIB



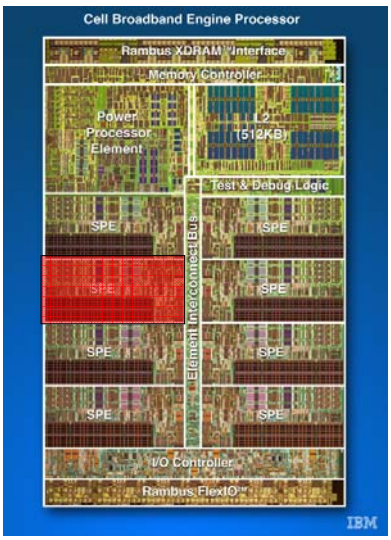
PowerPC Processor Element

- General purpose, 64-bit RISC processor (PowerPC)
- 2-Way SMT
- L1 : 32KB I ; 32KB D
- L2 : 512KB
- Coherent load / store
- VMX-32
- 32 128-bit vector registers
- The L2 cache and the address-translation caches use replacement-management tables that allow software to control use of the caches.



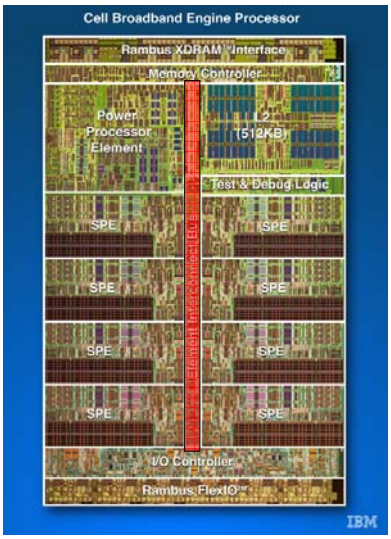
Synergistic Processor Element

- Synergistic Processor Unit (SPU)
- Synergistic Memory Flow Control (MFC)
  - Data movement and synchronization
  - Interface to high performanceElement Interconnect Bus
- Simple RISC User Mode Architecture
  - Dual issue VMX-like
  - Graphics SP-Float
  - IEEE DP-Float
- Dedicated resources: unified 128x128-bit RF, 256KB Local Store
- Dedicated DMA engine: Up to 16 outstanding requests



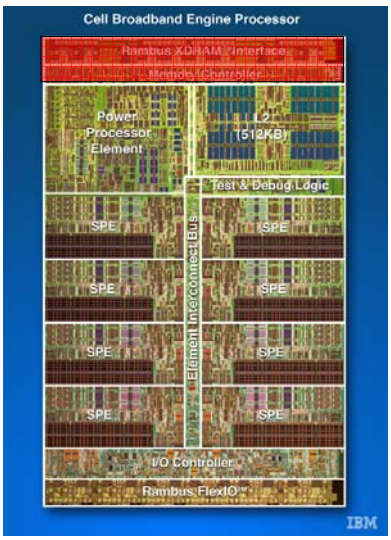
Element Interconnect Bus

- Four 16 byte data rings supporting multiple simultaneous transfers per ring
- 96Bytes/cycle (64B data + 32B tags) peak bandwidth
  - 307.2GB at 3.2GHz
- Over 100 outstanding requests

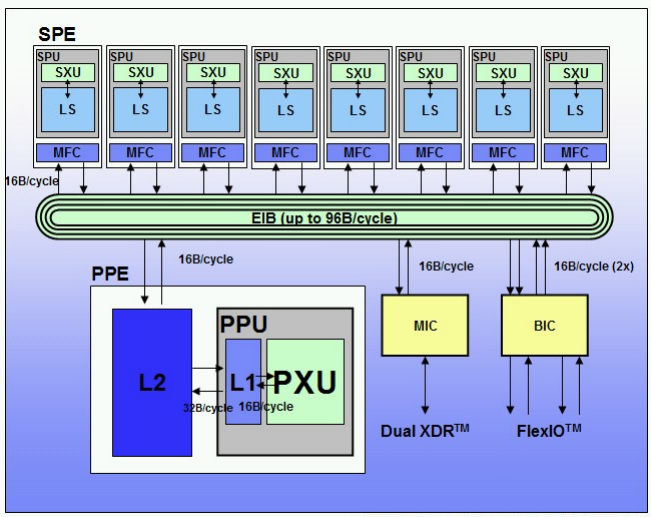


Memory Interface

- 16 B/cycle
- 25.6 GB/s (1.6 GHz)

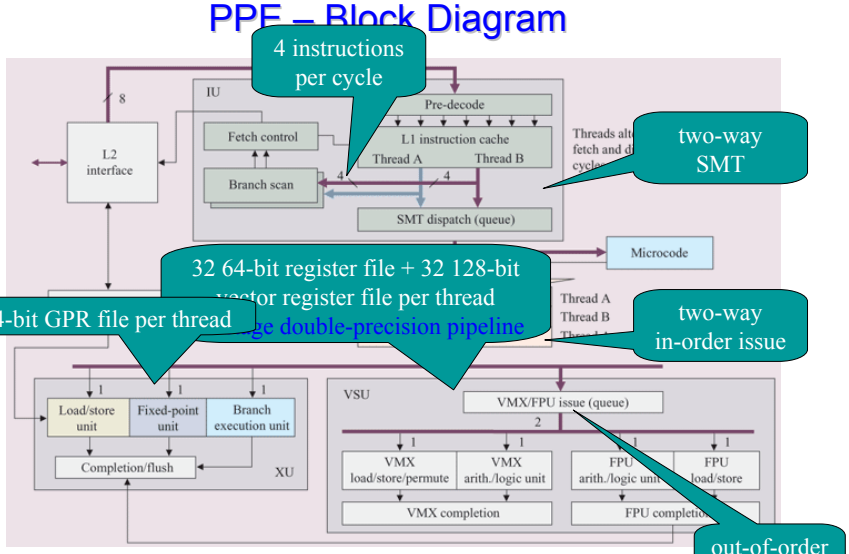


CELL – Block Diagram



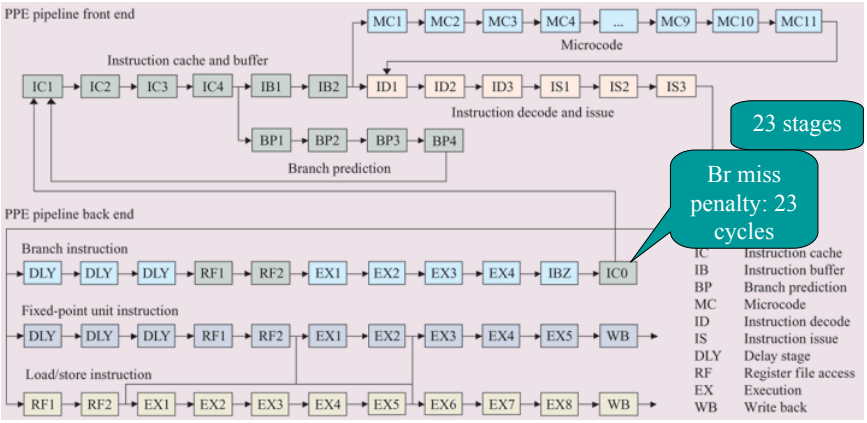
Source: M. Gschwind et al., Hot Chips-17, August 2005

PPE – Block Diagram



Source: J. Kahle, et al., Introduction to the Cell Multiprocessor

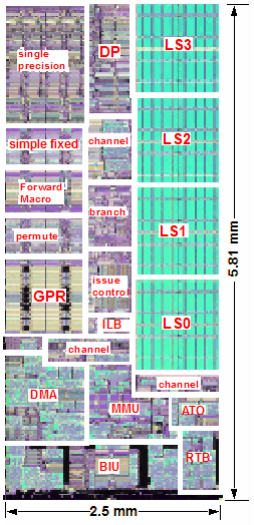
PPE - Pipeline



Source: J. Kahle, et al., Introduction to the Cell Multiprocessor

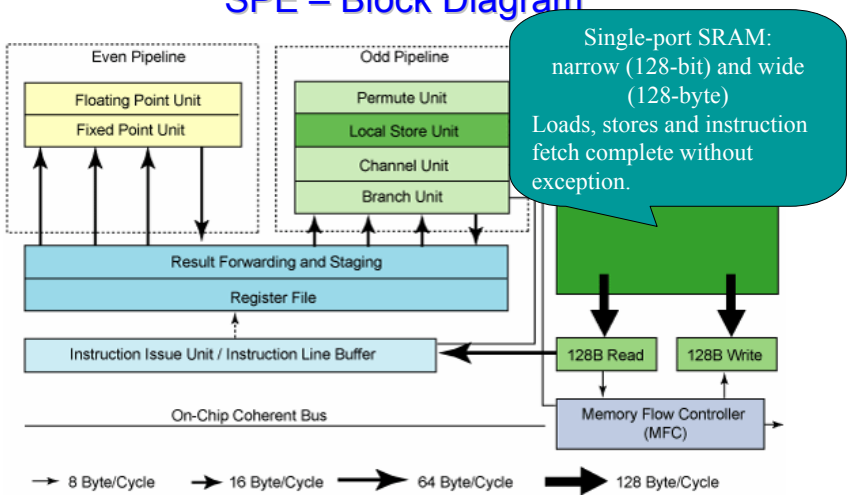
SPE – Die Photo

- BIU: bus interface unit
- MMU: Memory Management Unit
- ATO: Atomic (memory) unit responsible for coherency observation/interaction with dataflow on the EIB.
- RTB: The TB in RTB stands for "Test Block". It contains the ABIST (Array Built In Self Test) engines for the Local Store and other arrays in the SPE, as well as other test related control functions for the SPE.



Source: www.gamespot.com

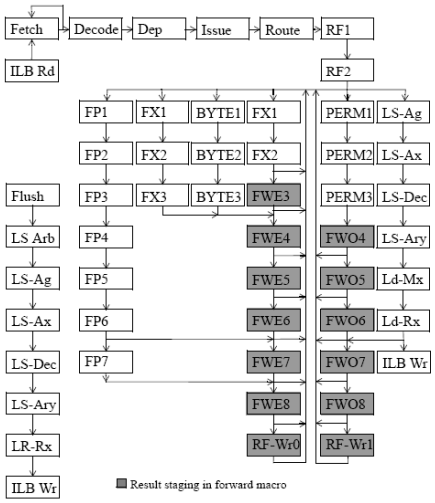
SPE – Block Diagram



Source: T. Chen, R. Raghavan, J. Dale, E. Iwata, Cell Broadband Engine Architecture and its first implementation

SPE - Pipeline

- 23 stages.
- Br miss prediction penalty: 18 cycles.
- Forward macro: 6 read ports.



Source: B. Flachs, et al., A Streaming Processing Unit for a CELL Processor

SPE – Dual-Issues Rules

- SPU has two pipelines:
  - even (pipeline 0) & odd (pipeline 1)

Instruction Class	Description	Latency (clock cycles)	Pipeline
LS	Load and store	6	Odd
HB	Branch hints	15	Odd
BR	Branch resolution	4	Odd
CH	Channel interface, special-purpose registers	6	Odd
SP	Single-precision floating-point	6	Even
DP	Double-precision floating-point	13	Even
FI	Floating-point integer	7	Even
SH	Shuffle	4	Odd
FX	Simple fixed-point	2	Even
WS	Word rotate and shift	4	Even
BO	Byte operations	4	Even
NOP	No operation (execute)	-	Even
LNOP	No operation (load)	-	Odd

SPE – Dual-Issues Rules cont.

- Condition:
  - the first instruction must come from an even word address and use the even pipe.
  - the second instruction must come from an odd word address and use the odd pipe.

SPE - Branch

- Prediction: static (not-taken).
- 3.5 fetched lines are stored in the instruction line buffer (ILB)
  - 32 x 4B / line.
  - 0.5 line holds instructions for the issue logic
  - 1 line holds the single entry software managed branch target buffer (SMBTB)
  - 2 lines are used for inline prefetching.
- Efficient software manages branches in three ways:
  - replaces branches with bit-wise select instructions
  - it arranges for the common case to be inline (not-taken)
  - it inserts branch hint instructions to identify branches and load the probable targets into the SMBTB.

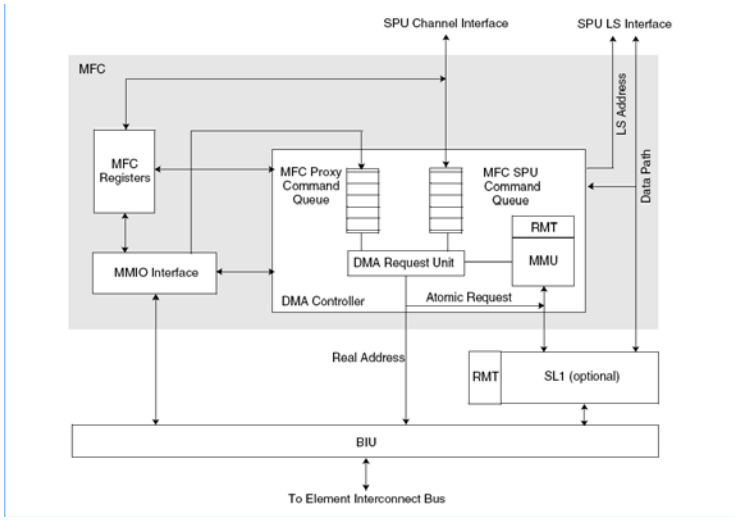
SPE – Branch Hint

- Hint for Branch (HBR) instructions
  - Placed before the corresponding branch.
  - Without a branch hint, fetch the following instruction.
  - With a branch hint, fetch the target.
  - If hint is wrong, flush and refetch the branch.
- A HBR provides: target, address of the branch, when to prefetch target
- How it works:
  - Load a branch-target buffer (BTB) in the SPU: target and branch address.
  - BTB monitors the instruction stream.
  - When meet the branch, go to the target.

SPE – Branch Hint Rules

- Enough separation between the hint and the branch (11 cycles).
- If HBR is too close, stall the branch until HBR finished.
- If HBR is closer than 4 instruction pairs plus one cycle, it is ignored.
- Only one HBR can be active at a time.
- An HBR can be moved outside of a loop and will be effective on each loop iteration as long as another HBR instruction is not executed.
- The HBR instruction only affects performance.

MFC



From: Cell Broadband Engine Architecture, v 1.02, IBM

Compiler Optimizations – Scalar code

- Scalar code on SIMD units
  - 128-bit registers.
  - Primary slot: left-most 32-bit.
  - Problems:
    - Address and Branch condition must be in the primary slot.
    - a = b +c (a is in word0, b is in word1, c is in word2)
      - Scalar: two loads, one add, one store (4 instructions)
      - Vector: two loads, one permute, one add, one load, one permute, one store (7 instructions)
  - Solutions:
    - allocate all local and global scalars to their own private 128-bit local memory lines and align the scalars into their primary slots.
    - Try to put them in the register file (reducing loads/stores).
    - Vectorization: reduce scalar operations.

Compiler Optimizations – Subword

- Integer promotion rule
  - Promote 8-bit/16-bit integers to 32-bit integers before computation.
  - Problems:
    - Extra instructions in SPE: load doesn't promote integers.
    - Different hardware support: SPE supports 16-bit hardware multiplier.
    - Not good for vectorization.
  - Solution: using subword when possible.
- More:
  - Using advanced compiler technology to exploit the performance of the Cell Broadband Engine™ architecture
  - http://www.research.ibm.com/journal/sj/451/eichenberger.html

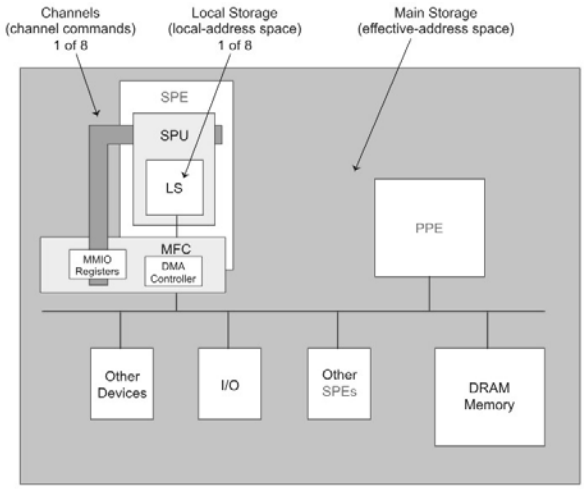
Threads / Tasks

- Main thread (a Linux thread on PPE)
  - Cell Broadband Engine Linux tasks
    - Linux threads (on PPE)
    - Linux threads (on SPE)
- SPE threads follow the M:N thread model
  - M threads distributed over N processor elements
- Sample:

```
#include <libspe2.h>
extern spe_program_handle_t spe_code;
...
spe_context_ptr_t ctx;
unsigned int entry = SPE_DEFAULT_ENTRY;
if ((ctx = spe_context_create(0, NULL)) == NULL) {
 perror("Failed creating SPE context"); exit(1); }
if (spe_program_load(ctx, &spe_code)) { perror("Failed loading
 program"); exit(1); }
if (spe_context_run(ctx, &entry, 0, NULL, NULL, NULL) < 0) {
 perror("Failed running context"); exit(1); }
```


Storage Domains

- Three types of storage domains:
  - one *main-storage domain*
  - eight SPE *local store domains*
  - eight SPE *channel domains.*



Storage Domains - Constraints

- No direct (SPU-program addressable) access to main storage. The SPU accesses main storage only by using the MFC's DMA transfers.
- No direct access to system control, such as page-table entries. PPE privileged software provides the SPU with the address-translation information that its MFC needs.
- With respect to accesses by its SPU, the LS is unprotected and un-translated storage.

SPE Local Store Domain

- Stores all instructions and data used by the SPU.
- Supports one access per cycle from either SPE software or DMA transfers.
- SPU data-access bandwidth is 16 bytes per cycle, quadword aligned.
- DMA-access bandwidth is 128 bytes per cycle.
- An SPU can only fetch instructions from its own LS with load and store instructions, and it performs no address translation for such accesses.
- An SPE program references its own LS using a Local Store Address (LSA)

SPE Addressing and Address Aliasing

- The LS of each SPE is also assigned a Real Address (RA) range within the system's memory map.
  - An SPU's LS can also be accessed by the PPE and other devices through the main-storage space
  - PPE accesses main storage with load and store instructions, without the need for DMA transfers
  - SPEs must use DMA transfers to access the LS in the main storage

  ➔No direct (SPU-program addressable) access to main storage

Communication Between the PPE and SPEs

- **DMAs**
  - To transfer data between main storage and the LS
- **Mailboxes**
  - Queues for exchanging 32-bit messages
- **Signal notification registers**

Communication Between the PPE and SPEs - Sample

- PPE loads data to memory →Use mailbox or signal to tell SPE →SPE computes the data.
- SPE completes the computation → Use DMA to put results into memory → Use mailbox to notify PPE

DMA

- Through MFC.
- DMA transfer requests contain both an LSA and an EA.
- Can check or wait on the completion.
- Max transfer size: 16 KB.
- Peak performance can be achieved:
  - Both EA and LSA are 128-byte aligned.
  - Transfer size is a multiple of 128 bytes.

Mailbox

- Two mailboxes (the SPU Write Outbound Mailbox and the SPU Write Outbound Interrupt Mailbox) are provided for sending messages from the SPE to the PPE
- One mailbox (the SPU Read Inbound Mailbox) is provided for sending messages to the SPE
- Between SPE: DMA to MMIO registers in the main memory.

Signal notification registers

- Each SPE has two 32-bit signal-notification registers, each has a corresponding memory-mapped I/O (MMIO) register into which the signal-notification data is written by the sending processor
- Signal-notification channels, or *signals*, are inbound (to an SPE) registers
- They can be used by other SPEs, the PPE, or other devices to send information, such as a buffer-completion synchronization flag, to an SPE

SPE ISA

- 204 instructions, 11 classes.

Type	Number
Memory Load and Store	16
Constant Formation	6
Integer and Logical Operations	59
Shift and Rotate	31
Compare, Branch, and Halt	40
Hint-for-Branch	3
Floating-Point	28
Control	8
SPU Channel	3
SPU Interrupt Facility	7
Synchronization and Ordering	3

Load and Store Instructions

- Only for Quadword
- Load Quadword
  - lqd rt,symbol(ra)
  - lqx rt,ra,rb
  - lqa rt,symbol
  - lqr rt,symbol (Load Quadword Instruction Relative)
    - Also has a Store Quadword Instruction Relative.

Constant-Formation Instructions

- Immediate Load Halfword
  - ilh rt,symbol (to all 16-bit slots of the register).
- Immediate Load Word
  - il rt,symbol (to all 32-bit slots of the register).

Integer and Logical Instructions

- Add Halfword
  - ah rt,ra,rb
- Add Halfword Immediate
  - ahi rt,ra,value (add value to each 16-bit slot).
- Count Leading Zeros
  - clz rt,ra (for each 32-bit slot).
  - 0 in rt means negative in ra, 32 in rt means 0 in ra.
- Count Ones in Bytes
  - cntb rt,ra (for each 8-bit slot).
  - 0 in rt means 0 in ra, 8 in rt means -1 in ra.
- Average Bytes
  - Avgb rt, ra, rb
  - For each 8-bit in ra and rb, rt = (ra + rb + 1) >> 1.
  - Avg (4, 3) = 4.
- Absolute Differences of Bytes
  - Absdb rt, ra, rb (for each 8-bit slot).

Shift and Rotate Instructions

- Rotate Halfword
  - roth rt, ra, rb
  - For each 16-bit slot, rotate contents in RA to the left.
  - Rotate count is in bits 12 to 15 of the corresponding 16-bit slot in RB.

Compare, Branch and Halt Instructions

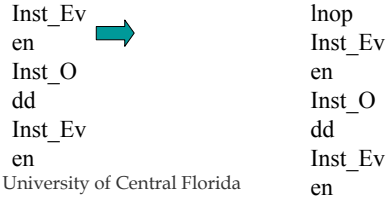
- Halt If Equal
  - heq ra, rb
  - The stop that occurs is not precise; as a result, execution can generally not be restarted.
- Compare Equal Byte
  - ceqb rt, ra, rb
  - For each 8-bit slot
    - if equal, all ones.
    - if not, all zeros.

Compare, Branch and Halt Instructions cont.

- Branch Relative
  - br symbol (PC = PC + symbol)
  - If symbol is zero?
- Branch Relative and Set Link
  - brsl rt, symbol (PC = PC + symbol, rt = PC + 4)

Control Instructions

- Two no operation instructions:
  - nop (even pipeline)
  - lnop (odd pipeline)
  - Dual issue rule:
    - the first instruction must come from an even word address and use the even pipe.
    - the second instruction must come from an odd word address and use the odd pipe.
  - Useful to pad instructions



References

1. Introduction to the Cell multiprocessor: www.research.ibm.com/journal/rd/494/kahle.html
2. One-Day IBM Cell Programming Workshop at Georgia Tech: www-static.cc.gatech.edu/f%7Ebader/CellProgramming.html
3. M. Gschwind, The Cell Broadband Engine: Exploiting Multiple Levels of Parallelism in a Chip Multiprocessor, International Journal of Parallel Programming, Vol. 35, No. 3, June 2007.
4. T. Chen, R. Raghavan, J. Dale, E. Iwata, Cell Broadband Engine Architecture and its first implementation, http://www-128.ibm.com/developerworks/power/library/pa-cellperf/
5. J. Kahle, et al., Introduction to the Cell Multiprocessor, IBM Journal of Research and Development, Vol. 49, No. 4/5, July/Sept. 2005, pp. 589-604.
6. The Cell Explained, www. gamespot.com.
7. B. Flachs, et al., A Streaming Processing Unit for a CELL Processor, ISSCC 2005.
8. A. E. Eichenberger, Using advanced compiler technology to exploit the performance of the Cell Broadband Engine architecture, IBM Systems Jounal, Vol. 45, No.1, 2006.
9. Software Development Kit for Multicore Acceleration Version 3.0 Programming Tutorial, IBM.
10. Synergistic Processor Unit Instruction Set Architecture, V 1.2, IBM.
11. Cell Broadband Engine Architecture, v 1.02, IBM.