Single-node Optimization

Customer Documentation and Training

PGI Compiler Options

`-help` option displays command line options

- Request information about either a single option or groups of options

  ```
  % pgf90 -help=groups
  ```

- Display the available optimization switches

  ```
  % pgf90 -switch –help
  ```

- `% pgf90 -fast -help`
- `Reading rcfile /opt/pgi/6.2.2/linux8664/6.2/bin/.pgf90rc  -fast`
- `Common optimizations; includes -O2 -Munroll=c:1 -Mnoframe -Mlre`

PGI Compiler Options

`-fast`	Chooses generally optimal flags for the target platform and includes: `-O2 -Munroll=c:1 -Mnoframe -Mlre -Mautoinline -Mvect=sse -Mscalarsse -Mcache_align -Mflushz`
`-fastsse`	`== -fast`
`-O2`	This level performs all level-one local optimization as well as level two global optimization
`-Munroll=c:1`	Invoke the loop unroller. This also sets the optimization level to a minimum of -O2. `c:1` instructs the compiler to completely unroll loops with a constant loop count less than or equal to 1
`-Mnoframe`	Don't set up a true stack frame pointer for functions; this allows a slightly more efficient operation when a stack frame is not needed, but some options override this.
`-Mlre`	Enable loop-carried redundancy elimination
`-Mautoinline`	Inline functions with the inline attribute up to n (default 5) levels deep
`-Mvect=sse`	Pass options to the internal vectorizer. `sse` : use SSE, SSE2, 3Dnow, and prefetch instructions in loops where possible.
`-Mscalarsse`	Utilize SSE and SSE2 instructions to perform the operations coded
`-Mcache_align`	Align unconstrained data objects of size greater than or equal to 16 bytes on cache-line boundaries
`-Mflushz`	Set SSE to flush-to-zero mode

PGI Compiler Options

`-Mnovintr`	Do not generate vector intrinsic calls
`-M[no]prefetch`	[Do not] Add prefetch instructions
`-M[no]stride0`	Alternate code for a loop that contains an induction variable with a possible increment of zero
`-M[no]zerotrip`	[Do not] Include a zero-trip test for loops
`-M[no]depchk`	[Do not] Assume that potential data dependencies exist
`-Mnontemporal`	Enable nontemporal move prefetching
`-M[no]reentrant`	[Do not] Disable optimization that may produce non reentrant code
`-Mextract=option`	Subprogram extraction phase to prepare for inlining
`-Mprof=option`	Profile options
`-Mlist`	Creates a listing file
`–Mipa=option`	Enable and specify options for InterProcedural Analysis (IPA) `Mipa=fast,inline` is recommended
`-Minline=levels:#`	Levels of inlining, levels:10 recommended for C++
`-Msafeptr=all`	C/C++ safe pointers
`-Minfo=option`	Informational messages
`-Mneginfo=option`	Why optimizations were not done
`-mp=nonuma`	OpenMP; Supported on CNL multi-core systems only

PGI Fortran Compiler Flags

`-default64`	Fortran driver option for -i8 and -r8
`-i8, -r8`	Treats INTEGER and REAL variables in Fortran as eight bytes (Caution: Use the `ftn -default64` option to link the right libraries, NOT `-i8` or `-r8`)
`-byteswapio`	Reads big-endian files in Fortran. (Cray XT systems are little endian)
`-Mnomain`	Uses the `ftn` driver to link programs with the main program (written in C or C++) and one or more subroutines (written in Fortran)
Note: The PGI Fortran `stop` statement writes a `FORTRAN STOP` message to standard out. In a parallel application, this may not scale well. To turn off this message, use the environment variable `NO_STOP_MESSAGE`.	

PathScale Compiler Options

`-Ofast`	Use optimizations selected to maximize performance. Equivalent to: `-O3 -ipa -OPT:Ofast -fno-math-errno -ffast-math`
`-O3`	Turns on aggressive optimization
`-ipa`	Invokes inter-procedural analysis
`-OPT:Ofast`	Use optimizations selected to maximize performance
`-ffast-math`	improves FP speed by relaxing ANSI & IEEE rules
`-fno-math-errno`	Do not set ERRNO after calling math functions that are executed with a single instruction, e.g. sqrt. A program that relies on IEEE exceptions for math error handling may want to use this flag for speed while maintaining IEEE arithmetic compatibility.
`-OPT:alias=restrict`	`alias` specifies the pointer aliasing model to be used. `restrict` specifies that distinct pointers are assumed to point to distinct, non-overlapping objects

GCC Compiler Options

`-ffast-math`	Sets `-fno-math-errno, -ffinite-math-only, -funsafe-math-optimizations, -fno-trapping-math, -fno-rounding-math, -fno-signaling-nans and fcx-limited-range`. This option causes the preprocessor macro `"__FAST_MATH__"` to be defined. This option should never be turned on by any `-O` option since it can result in incorrect output for programs which depend on an exact implementation of IEEE or ISO rules/specifications for math functions.
`-O3`	Turns on all optimizations specified by `-O2` and also turns on the `-finline-functions`, `-funswitch-loops and -fgcse-after-reload` options.
`-funroll-loops`	Unroll loops whose number of iterations can be determined at compile time or upon entry to the loop. `-funroll-loops` implies `-frerun-cse-after-loop`. This option makes code larger, and may or may not make it run faster.
`-fprefetch-loop-arrays`	If supported by the target machine, generate instructions to prefetch memory to improve the performance of loops that access large arrays. This option may generate better or worse code; results are highly dependent on the structure of loops within the source code. Disabled at level `-Os`

Other Compiler Information

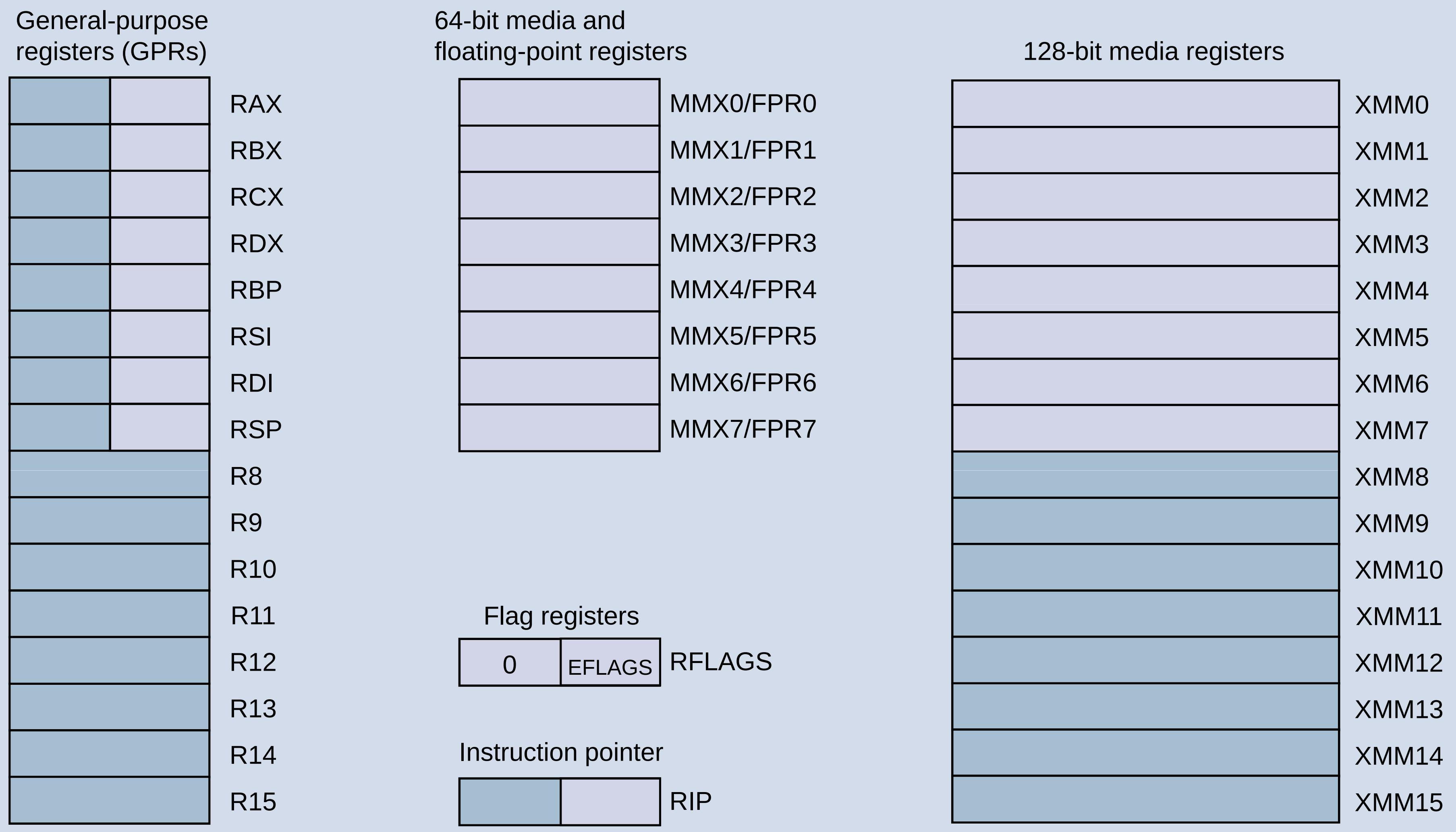
- The PGI option `-Mconcur`, `-mprof=mpi`, `-Mmpi`, and `-Mscalapack` are not supported
- Fortran interfaces can be called from a C program by adding and underscore to the respective name
  - Pass arguments by reference rather than by value
  - For example to call `dgetrf()`

```
dgetrf_ (&uplo, &m, &n, a, &lda, ipiv, work, &lwork, &info);
```

SSE on the AMD Opteron

- SSE means "Streaming SIMD Extensions" that enhance the x86 instruction set
- SSE2 – for single/dual-core processors
- SSE3 – for Revision E processors (dual-core)
- SSE4a – for quad-core processors
- SSE2 enhancements include:
  - 8 new 128-bit SIMD floating-point registers that are directly addressable (XMM8-15)
    - For 32-bit variables: vector instructions with vector length 4
    - For 64-bit variables: vector instructions with vector length 2
  - 50 new instructions for packed floating-point data
  - MMX instructions for multimedia, video, mpeg

AMD Registers



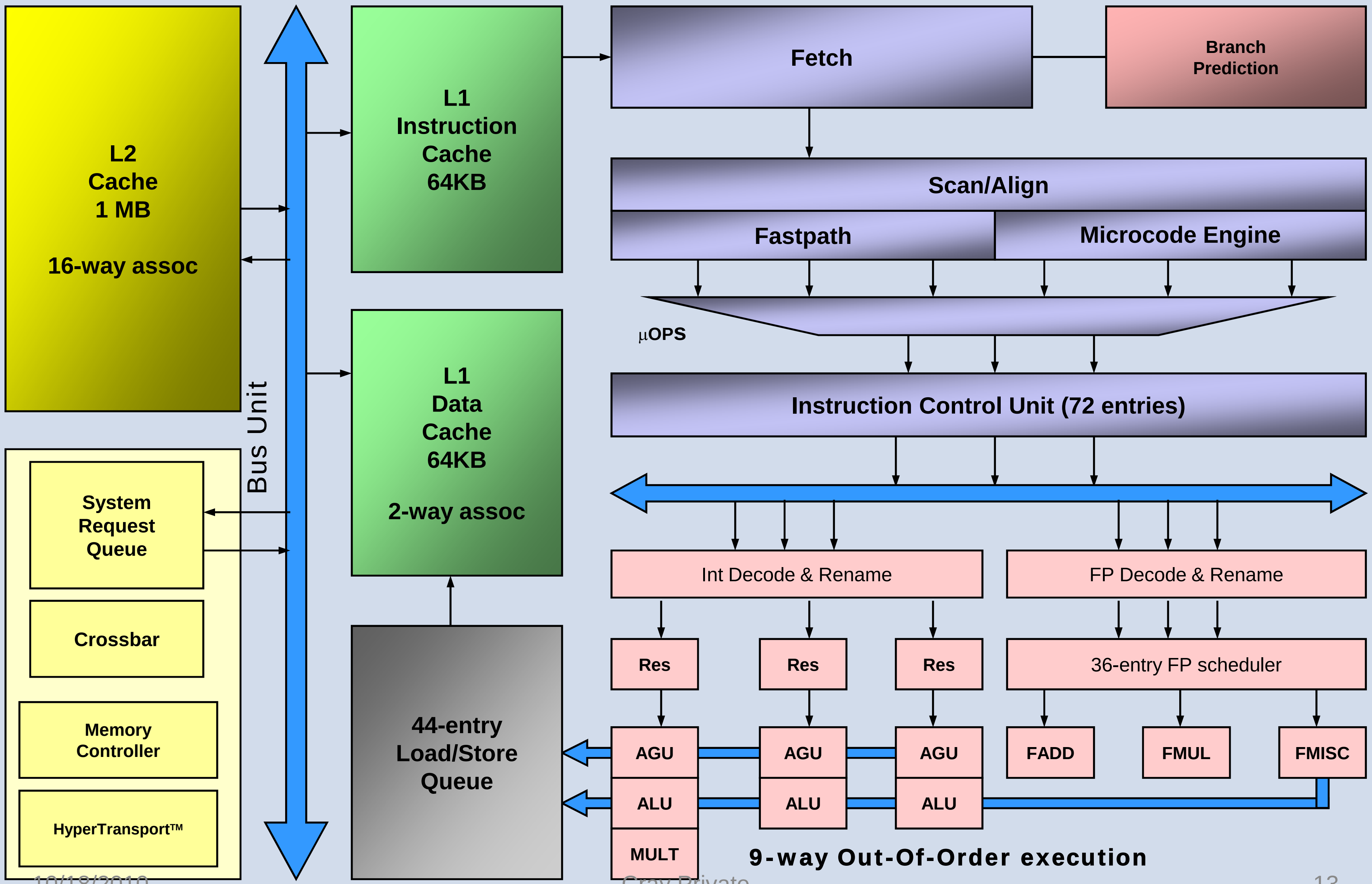
Single CPU Performance: Vectorization

- If you want your application to perform well, vectorize the code
  - Strongly recommended for single precision arithmetic
- What vectorizes
  - Inner stride-1 loops
    - Problems may occur with large body loops
      `-Mvect=nosizelimit`
  - Use the PGI `-Minfo` option to recognize loops that vectorize
- What does not vectorize
  - Loops that have `if` statements or any indirect or non-unit stride references
  - Outer loops
  - Use the PGI `–Mneginfo` option to recognize loops that do not vectorize

Vectorization

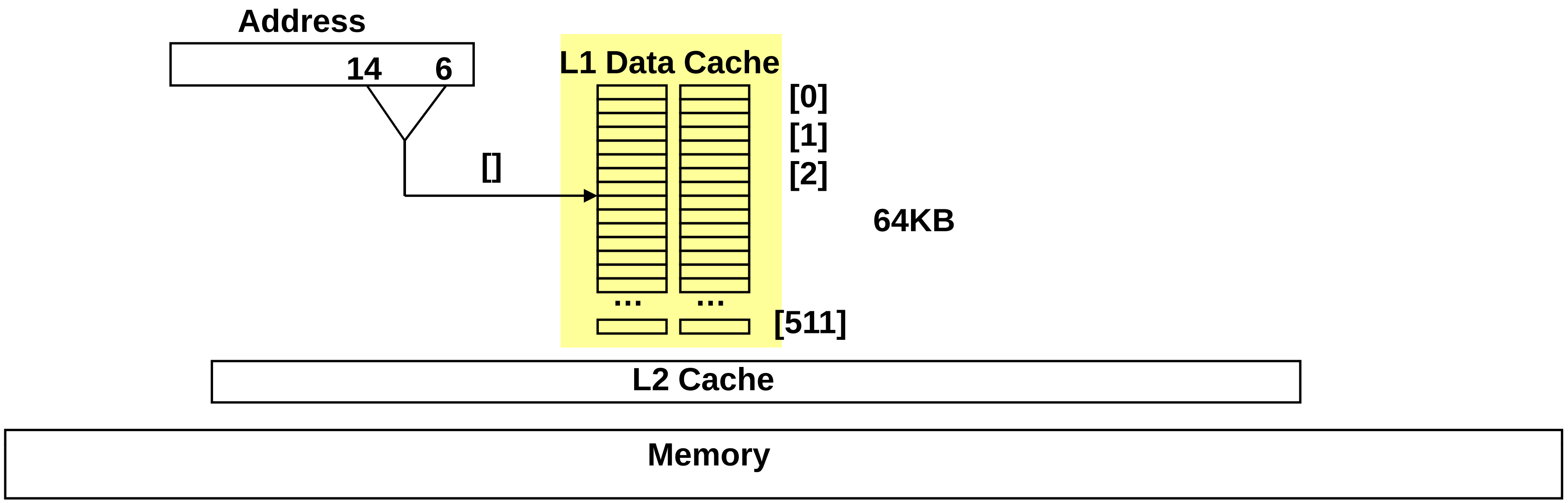
- The `–Mvect` option is included as part of `-fast`
- The vectorizer scans code searching for loops that are candidates for high-level transformations, such as
  - loop distribution
  - loop interchange
  - cache tiling
  - idiom recognition (replacement of a recognizable code sequence, such as a reduction loop, with optimized code sequences or function calls)
- In addition, the vectorizer produces extensive data dependence information for use by other phases of compilation and detects opportunities to use vector or packed Streaming SIMD Extensions (SSE/SSE2)
- In 32-bit mode, the vector length is 4 with SSE2
- In 64-bit mode, the vector length is 2 with SSE2

AMD Opteron Single-core Processor



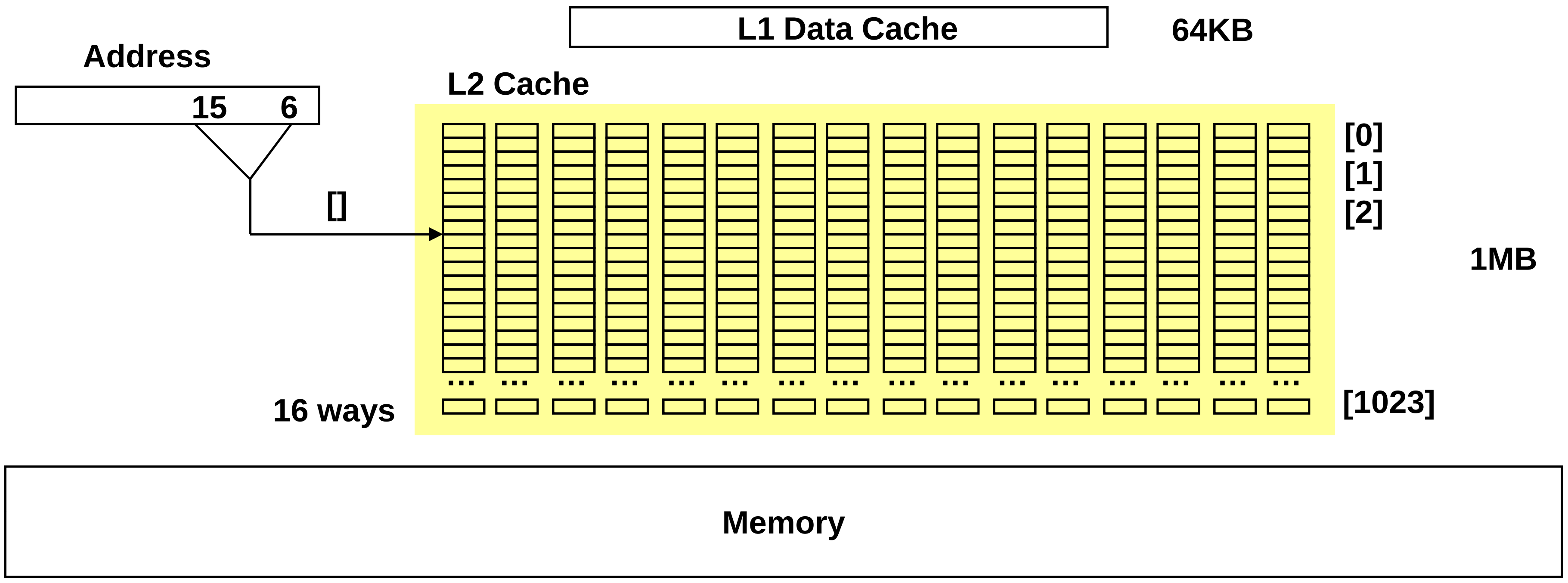
Memory Access Patterns (L1)

- The Opteron L1 data cache line is 64 bytes (512 bits)
- Latency from the L1 cache is 3 clocks (2/cycle)
  - Any stride that is a multiple of 64 causes L1 bank conflicts
- Addresses 512*64 (or 32768 apart) come back to the same row; a refill from L2 has a latency of 11 or more clocks; a useful principle is to avoid large powers of 2



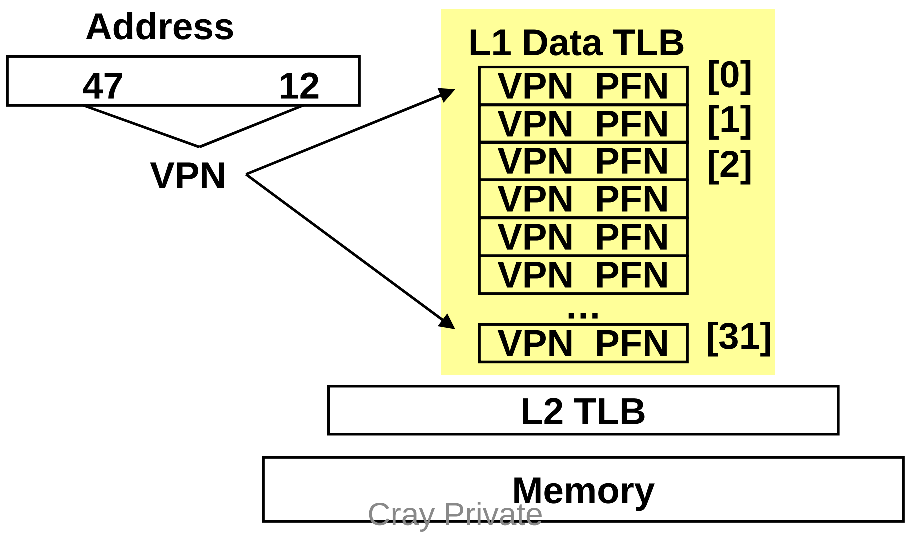
Memory Access Patterns (L2)

- The Opteron L2 cache line is 64 bytes
- Latency from the L2 cache is 11 or more clocks
- Addresses that are 1024*64 (or 65536) apart come back to the same row; latency for a refill from memory is ~150cp; a useful principle is to avoid large powers of 2



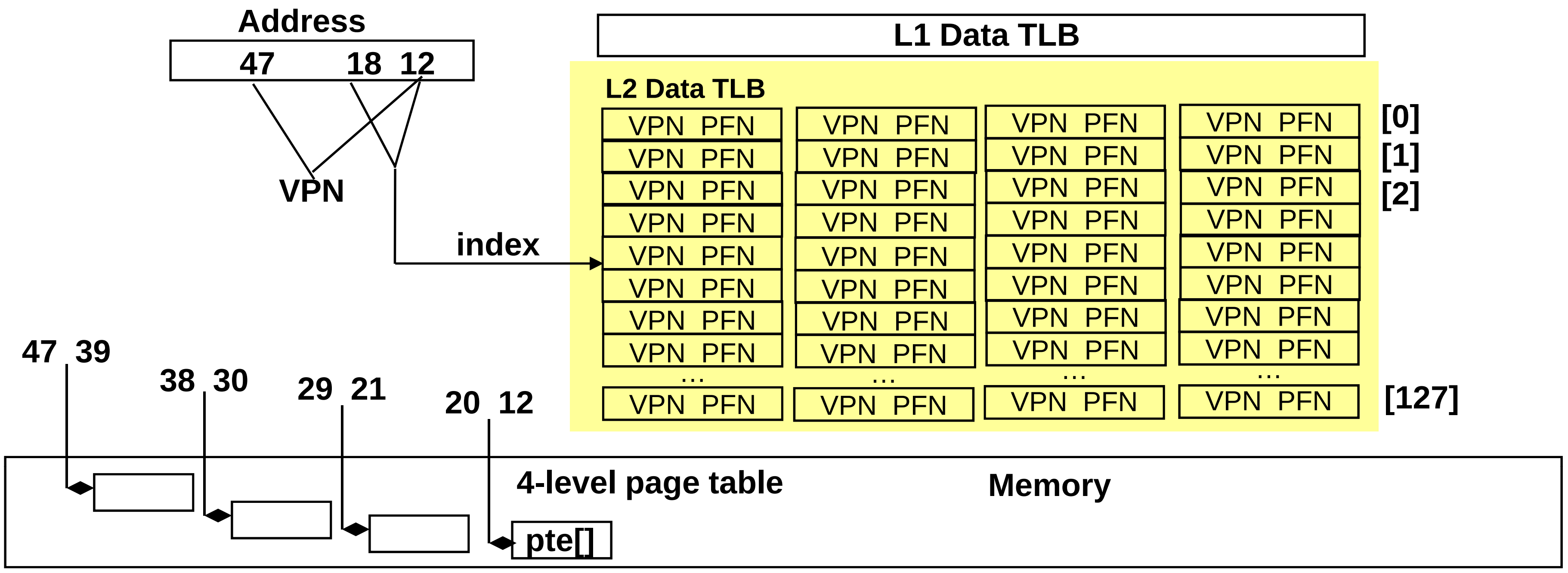
Memory Access Patterns (L1 TLB)

- Opteron L1 data TLB caches
  - 8 VPNs for 2-MB pages or larger
    - 2-MB pages are the default for Catamount
    - Not available for CNL (until release CLE 2.1)
  - 32 virtual page numbers (VPNs) for 4-KB pages
    - 4-KB pages are the only option for CNL, until CLE 2.1
    - For Catamount use: yod -small_pages
- When using 4-KB pages the L1 TLB is refilled from the L2 TLB

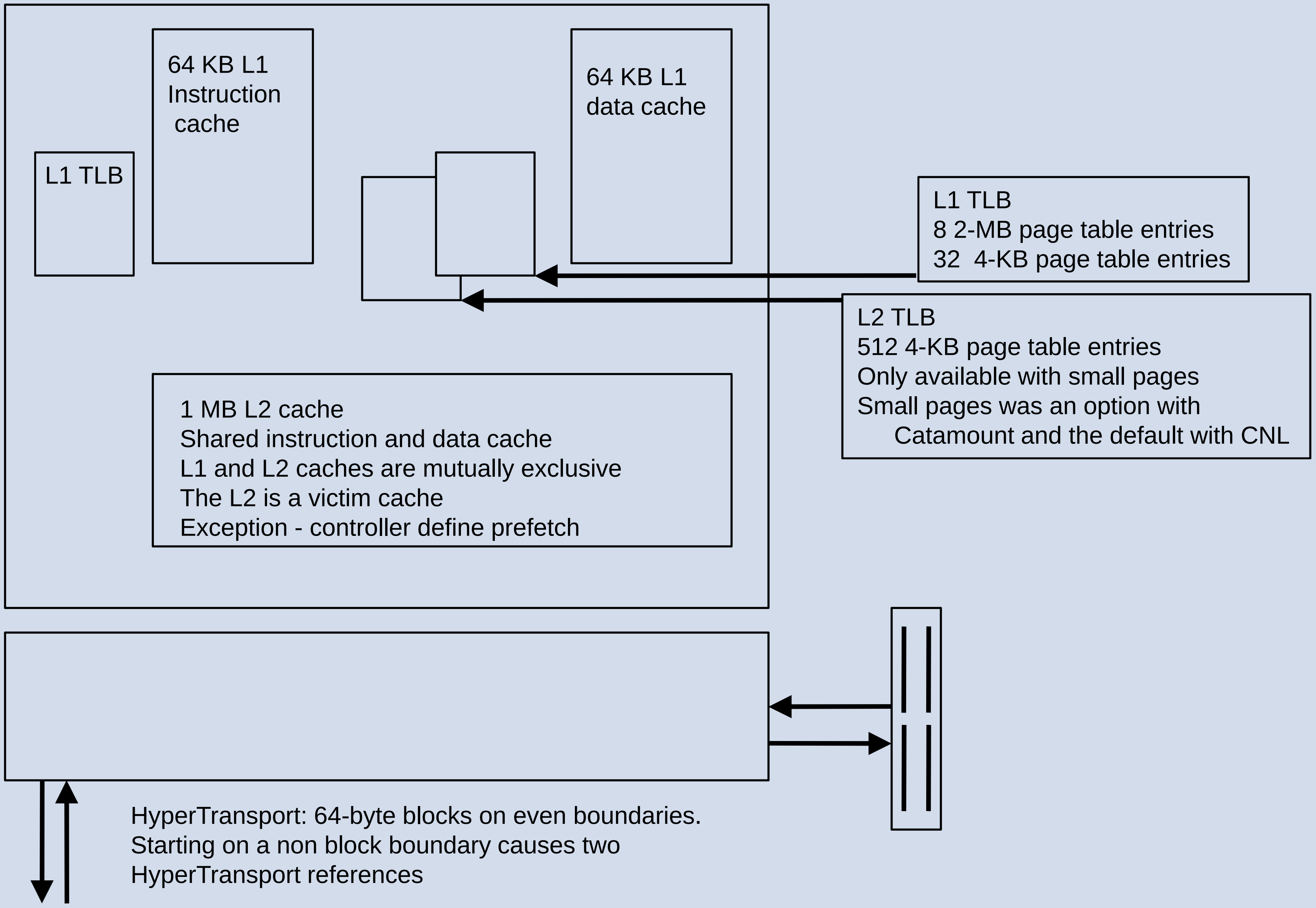


Memory Access Patterns (L2 TLB)

- The Opteron L2 data TLB caches 512 VPNs in 128 rows, 4 ways: Only for 4 KB pages!
- Address bits 18-12 index to a row in which the VPN must occur
- Addresses that are 512 KB apart wrap to the same row

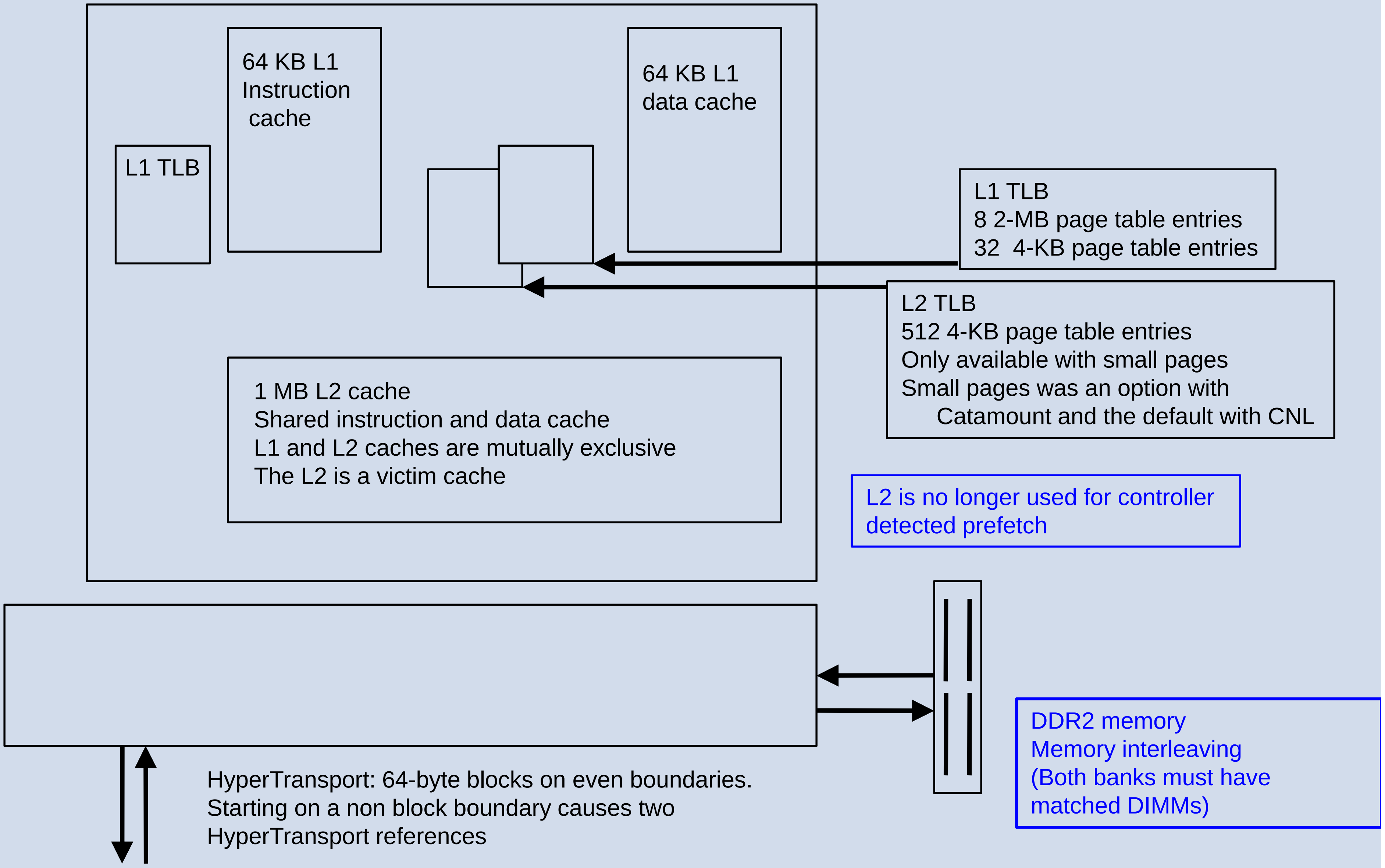


Single-core Opteron Caches


64 KB L1
Instruction
cache
L1 TLB
64 KB L1
data cache
L1 TLB
8 2-MB page table entries
32 4-KB page table entries
L2 TLB
512 4-KB page table entries
Only available with small pages
Small pages was an option with
Catamount and the default with CNL
1 MB L2 cache
Shared instruction and data cache
L1 and L2 caches are mutually exclusive
The L2 is a victim cache
Exception - controller define prefetch

HyperTransport: 64-byte blocks on even boundaries.
Starting on a non block boundary causes two
HyperTransport references

One Core of a Dual-core Opteron



L1 TLB
8 2-MB page table entries
32 4-KB page table entries

L2 TLB
512 4-KB page table entries
Only available with small pages
Small pages was an option with Catamount and the default with CNL

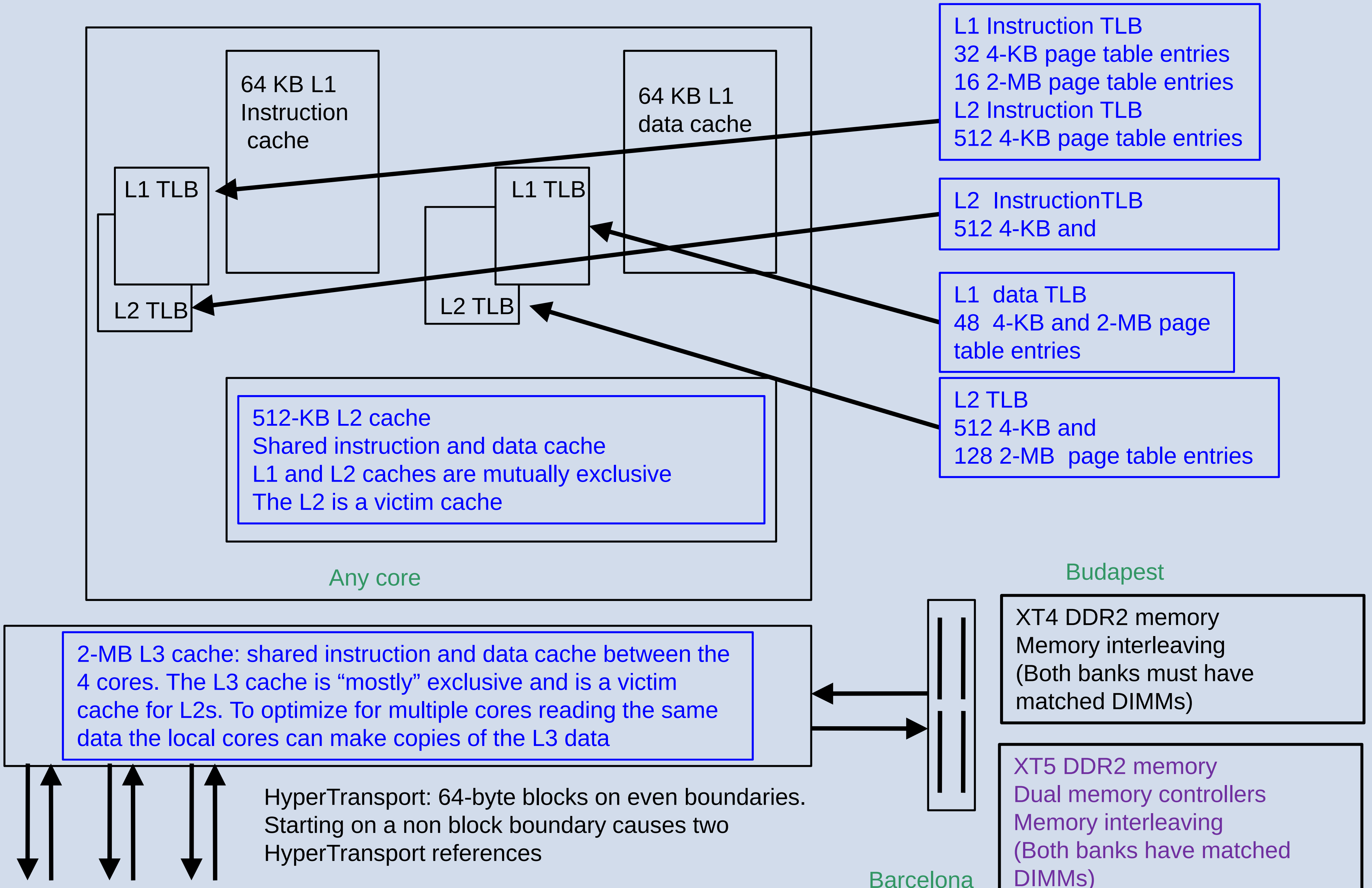
1 MB L2 cache
Shared instruction and data cache
L1 and L2 caches are mutually exclusive
The L2 is a victim cache

L2 is no longer used for controller detected prefetch

DDR2 memory
Memory interleaving
(Both banks must have matched DIMMs)

HyperTransport: 64-byte blocks on even boundaries.
Starting on a non block boundary causes two HyperTransport references

One Core of a Quad-core Opteron



L1 Instruction TLB
32 4-KB page table entries
16 2-MB page table entries
L2 Instruction TLB
512 4-KB page table entries

L2 InstructionTLB
512 4-KB and

L1 data TLB
48 4-KB and 2-MB page table entries

L2 TLB
512 4-KB and
128 2-MB page table entries

512-KB L2 cache
Shared instruction and data cache
L1 and L2 caches are mutually exclusive
The L2 is a victim cache

2-MB L3 cache: shared instruction and data cache between the 4 cores. The L3 cache is “mostly” exclusive and is a victim cache for L2s. To optimize for multiple cores reading the same data the local cores can make copies of the L3 data

HyperTransport: 64-byte blocks on even boundaries. Starting on a non block boundary causes two HyperTransport references

XT4 DDR2 memory
Memory interleaving
(Both banks must have matched DIMMs)

XT5 DDR2 memory
Dual memory controllers
Memory interleaving
(Both banks have matched DIMMs)

Additional Quad-core Enhancements

- Floating-point is now 128 bits wide
  - 1 cycle per vector
    - Previous processors consumed two cycles per vector
- Out-of-order loads, ordered retirement
- Can do 2 128-bit loads or 2 64-bit stores
- Other improvements
  - Improved performance of shuffle instructions
  - Improved transfers between floating-point and general purpose registers
  - Improved transfers between floating-point registers
  - Optimization of repeated move instructions
  - More efficient PUSH/POP stack operations
  - Adaptive prefetch, automatically advances the fetch ahead distance if the demand stream catches up to the prefetch stream
  - Write combining buffers improve performance

Quad-core Cautions

- The XMM registers are 128 bits wide; partial loads can cause merge dependencies
  - Initializations that zero the unused bit do not have these merge dependencies
- Arrays that are aligned 32K bytes apart can cause cache corruption, particularly if PREFETCHNTA is used

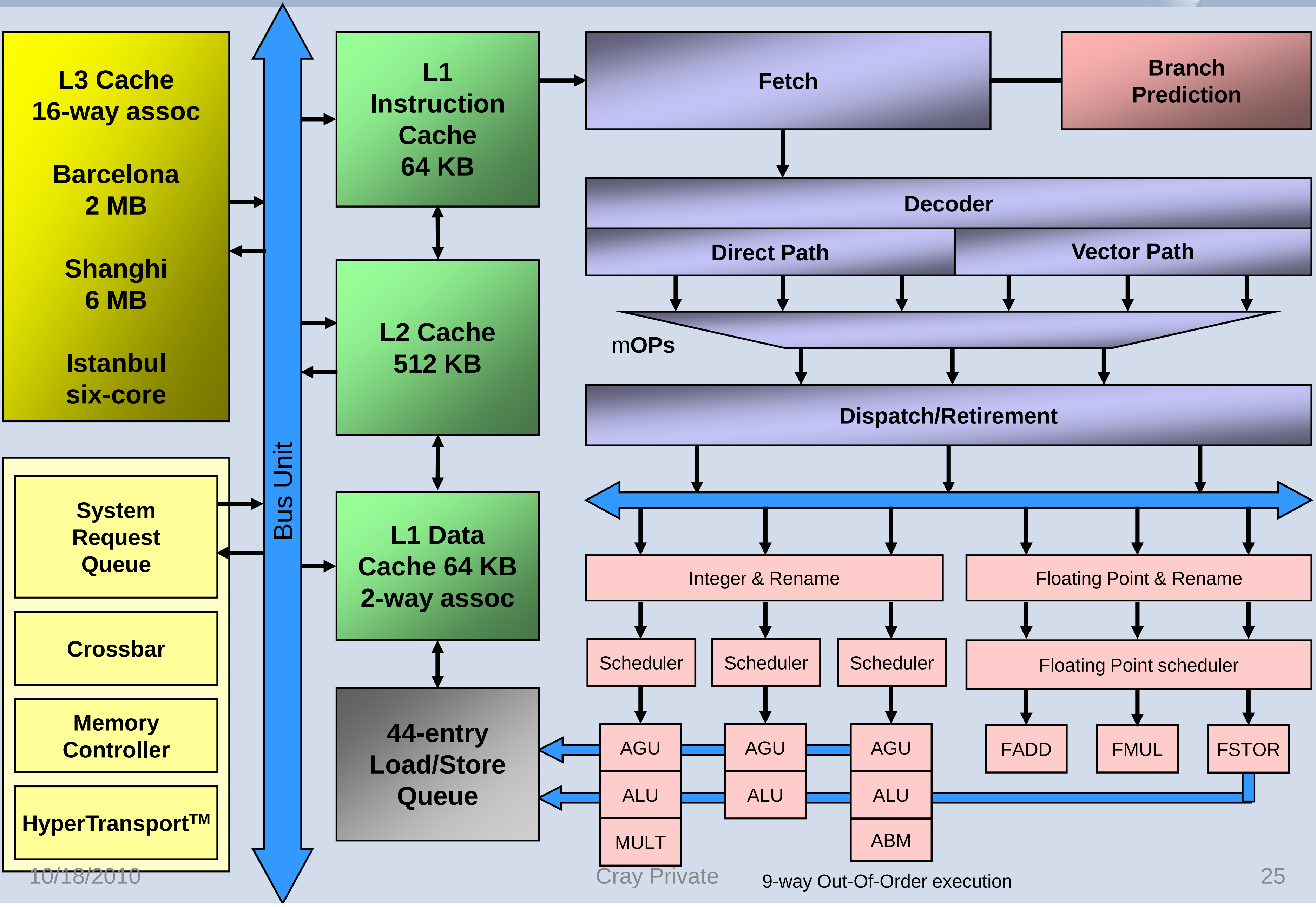
Quad-core Information

- Stride-1 memory access patterns are important
  - Stride-1 uses all of the cache line before it can be evicted
  - The compiler often performs prefetch for stride-1 loops, not as often for non stride-1 loops
- It is advisable to align hot loops on 32-byte blocks
- PREFETCH versus PREFETCHW
  - PREFETCHW is recommended if you expect to modify the data
  - PREFETCHW sets hints of intent to write
- Use streaming instructions instead of PREFETCHW
  - If the code will overwrite one or more complete lines with new data
  - If the new data will not be used again
- Maximum of eight outstanding cache-misses
  - References to the same cache line are merged
- Align floating point data on 16-byte boundaries

Additional XT Quad-core Information

- **General module additions for quad-core**
  - `module load xtpe-quadcore (Depricated)`
  - `module load xtpe-barcelona`
  - `module load xtpe-shanghai`
  - `module load xtpe-istanbul`
  - `module load xtpe-mc8 (Magny-cours)`
  - `module load xtpe-mc12 (Magny-cours)`

Quad-core Diagram



9-way Out-Of-Order execution