Optimizations Related to the Virtual Memory System

Background:
- The processor works with virtual addresses
- All caches work with physical addresses
- Both address spaces are organized in pages
- Typical page size: 4KB
- So the address translation translates virtual page numbers into physical page numbers

Diagram:
- Processor
- Virtual address
- VPN
- PPN
- Physical address
- L1 cache
- 12 bits
- 2^12 = page size
- 64B + 2^6B
- Associativity = 8 = 2^3

Notes:
- VPN = virtual page number
- VPO = " " offset
- PPN = physical page number
- PPO = " " offset
- SI = set index
- BO = block offset

Address translation:
- VPN \rightarrow PPN
- VPO = PPO = SI \cup BO => cache lookup can start before VPN \rightarrow PPN translation is finished

Address translation:
- Uses a cache called translation look-aside buffer (TLB)
- Core 2: four levels of caches for loads
  - DL1BO: 16 entries
  - DL1B: 256 entries
- Core 1: DL1BO hit: no penalty
- DL1B hit: 2cycles penalty
- Miss: possibly very expensive

Consequence:
- Repeatedly accessing a working set that
  is spread over >256 pages leads to TLB misses — possible severe slowdown
Solution 1: use large pages may require different kernel (OS) and C std library

Solution 2 (if possible): copy cوةج set into contiguous memory

How does this affect MMM

\[ \begin{bmatrix} a \end{bmatrix} \begin{bmatrix} b \end{bmatrix} \begin{bmatrix} c \end{bmatrix} = \begin{bmatrix} M \end{bmatrix} \begin{bmatrix} M \end{bmatrix} \begin{bmatrix} M \end{bmatrix} \]

which memory regions are repeatedly accessed?

- block row of a: is contiguous
- all of b: is contiguous
- tile of c: can be spread over \( N_S \) pages

if \( M > 512 \), \( 512 \) double = \( 4 \times 1 \times 3 = 12 \) page faults

but: typically \( N_S < 100 < 512 \) (DTCB1)

so at most 2 cycle penalty per row

\( \Rightarrow \) not worth to copy (on Core)

But: the BLAS 3 function dgemm has this interface:

\[ \text{dgemm}(a, b, c, N, M, M, Lda, Lds, ldc) \]

The leading dimensions enable dgemm to be called on submatrices of larger matrices:

\[ \begin{bmatrix} a \end{bmatrix} \begin{bmatrix} b \end{bmatrix} \begin{bmatrix} c \end{bmatrix} = \begin{bmatrix} M \end{bmatrix} \begin{bmatrix} M \end{bmatrix} \begin{bmatrix} M \end{bmatrix} \]

which memory regions are repeatedly accessed?

- block row of a: spread over \( \leq N_S \) pages
- all of b: spread over \( \leq K \) pages
- tile of c: spread over \( \leq N_S \) pages

Here copy may pay for large enough K
Code:

```
// all of B reused; possibly copy
for i = 0 : Nb : N-1  
  // block row of A reused; possibly copy
    for j = 0 : Nq : M-1 
      // tile of C reused; possibly copy
        for k = 0 : Nc : K-1 

```