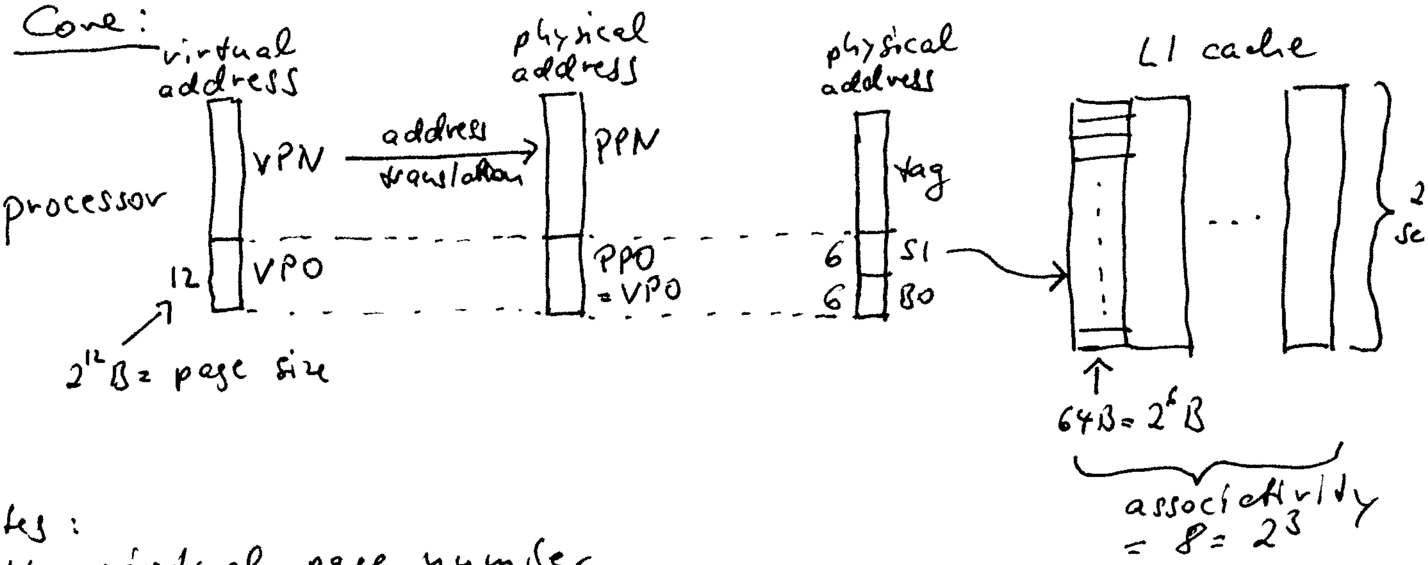
Optimizations Related to the Virtual Memory Syste

Background: (Core)
- 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

Core:



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

address translation: VPN → PPN
VPO = PPO = SI ∪ BO ⇒ cache lookup can start before VPN → PPN translation is finished!

address translation
- uses a cache called translation lookaside buffer (TLB
- Core 2: two levels of caches for loads
  DTLB0: 16 entries
  DTLB1: 256 entries

Pentium 4:
- one TLB
- 64 entries

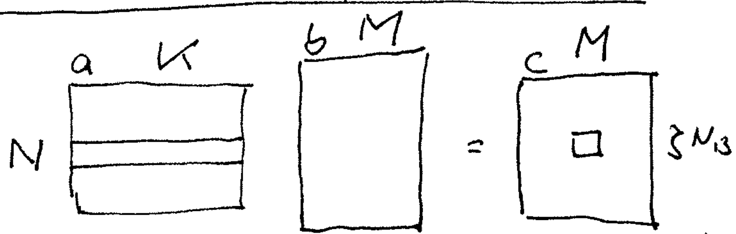
Case 1: DTLB0 hit: no penalty
DTLB1 hit: 2cycle penalty
miss: possibly *very* expensive

Consequence: Repeatedly accessing a working set that is spread over >256 pages leads to TLB misses → possibly severe slowdown

Solution 1: use larger pages
may require different kernel (OS) and C std library

Solution 2 (if possible): copy working set into contiguous memory
⟹ less pages are used

How does this affect MMM?



ijk loop order,
blocked into mini-MMM

which memory regions are repeatedly accessed?

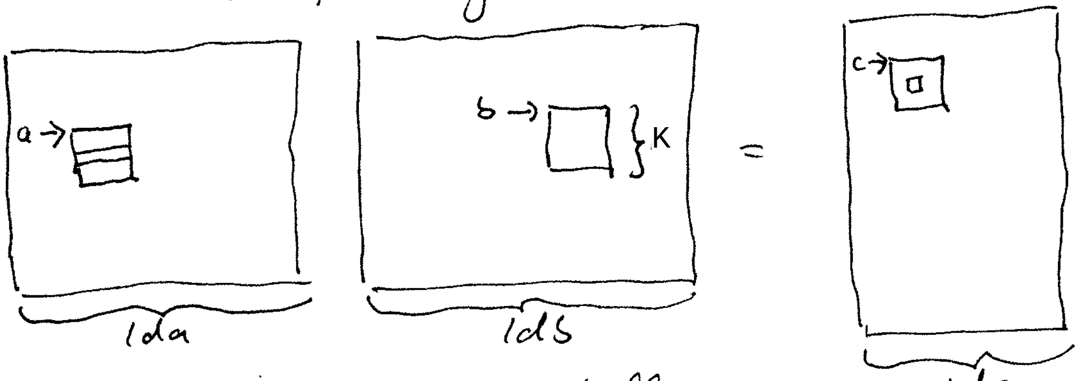
- block row of a: is contiguous
- all of b: is contiguous
- tile of c: can be spread over N_B pages if $M > 512$ (512 doubles = 4KB = page size)

but: typically $N_B < 100 <$ size(DTLB1)
so at most 2 cycles penalty per row
⟹ not worth to copy (on Core)

But: the BLAS 3 function dgemm has this interface:

$$\text{dgemm}(\underbrace{a, b, c}_{\text{matrix pointers}}, \underbrace{N, K, M}_{\text{matrix dimensions}}, \underbrace{lda, ldb, ldc}_{\cancel{\text{leading}}\ \text{"leading" dimensions}})$$

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



which memory regions are repeatedly accessed?

- block row of a: spread over $\le N_B$ pages
- all of b: spread over $\le K$ pages ←
- tile of c: spread over $\le N_B$ pages

Here copy may pay for large enough K

Code :

```
// all of B reused : possibly copy
for $i = 0 : N_B : N-1$
    // block row of A reused : possibly copy
    for $j = 0 : N_B : M-1$
        // tile of C reused : possibly copy
        for $k = 0 : N_B : K-1$
            . . . . . . . . . . .
```