SMT Register Deallocation Strategies

(1) Compiler/architecture support to free registers after their last use

- small SMT register files achieve large register file performance

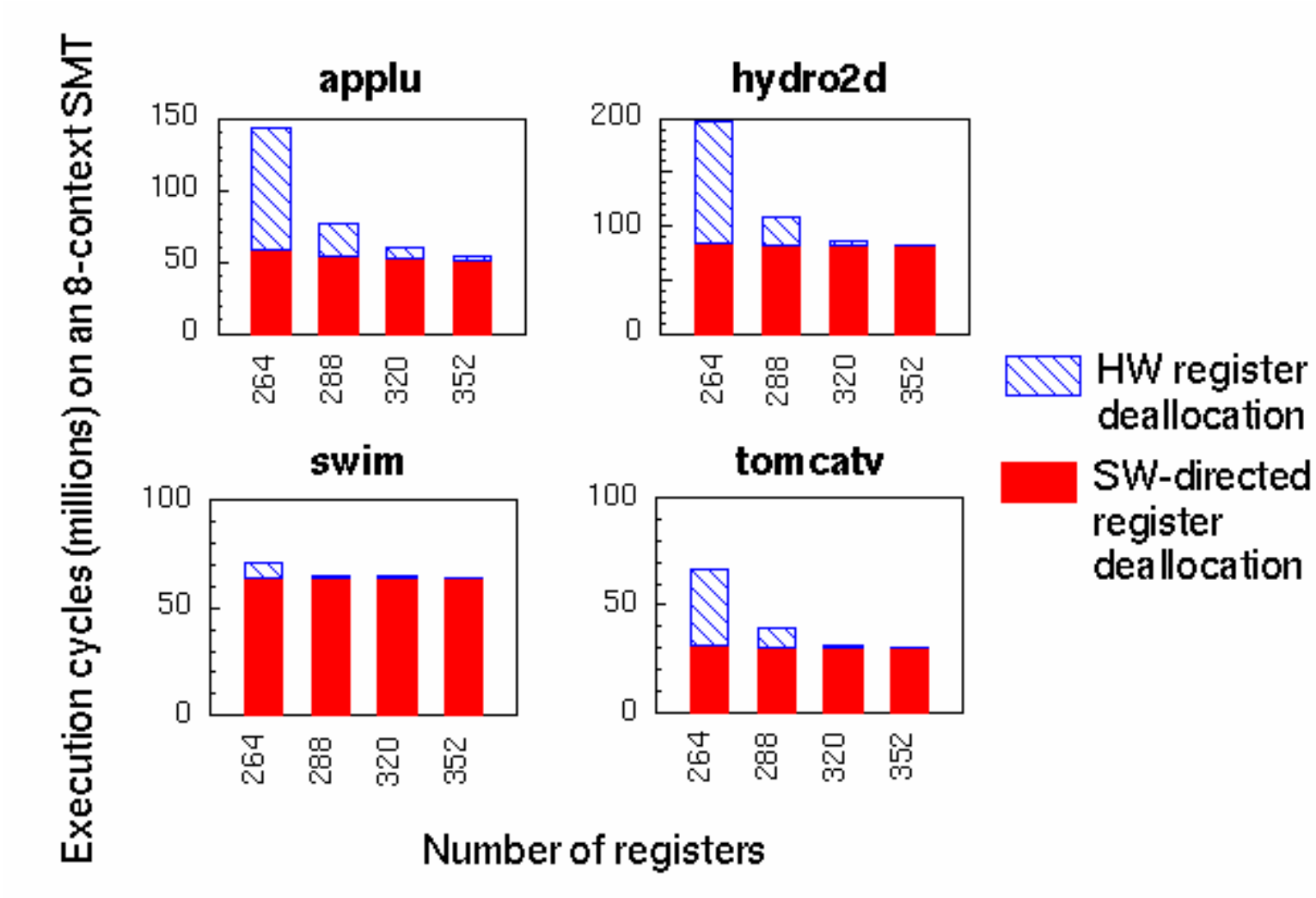
$\Rightarrow$ Use a small register file if the register access sets the cycle time.

(2) OS support to free registers in idle contexts:

- all HW contexts share SMT registers
  (traditional multithreaded processors have separate registers per thread)
- increases performance with a small register file by:
  - up to 50% with 8 threads
  - 3 times with fewer threads.

$\Rightarrow$ Can design a good throughput SMT for 8 threads & not penalize the performance of a single or few threads

Deallocating Registers After Last Use



Putting This in Context

Processor Architecture	Additional registers for register renaming
Conventional register file design for single-threaded CPU	100% (SMT=256)
SMT with OS support to free idle register contexts	27% (down to 70)
SMT with compiler/architecture support to free registers after their last use	3% (down to 8)