Coping with a Large Register File

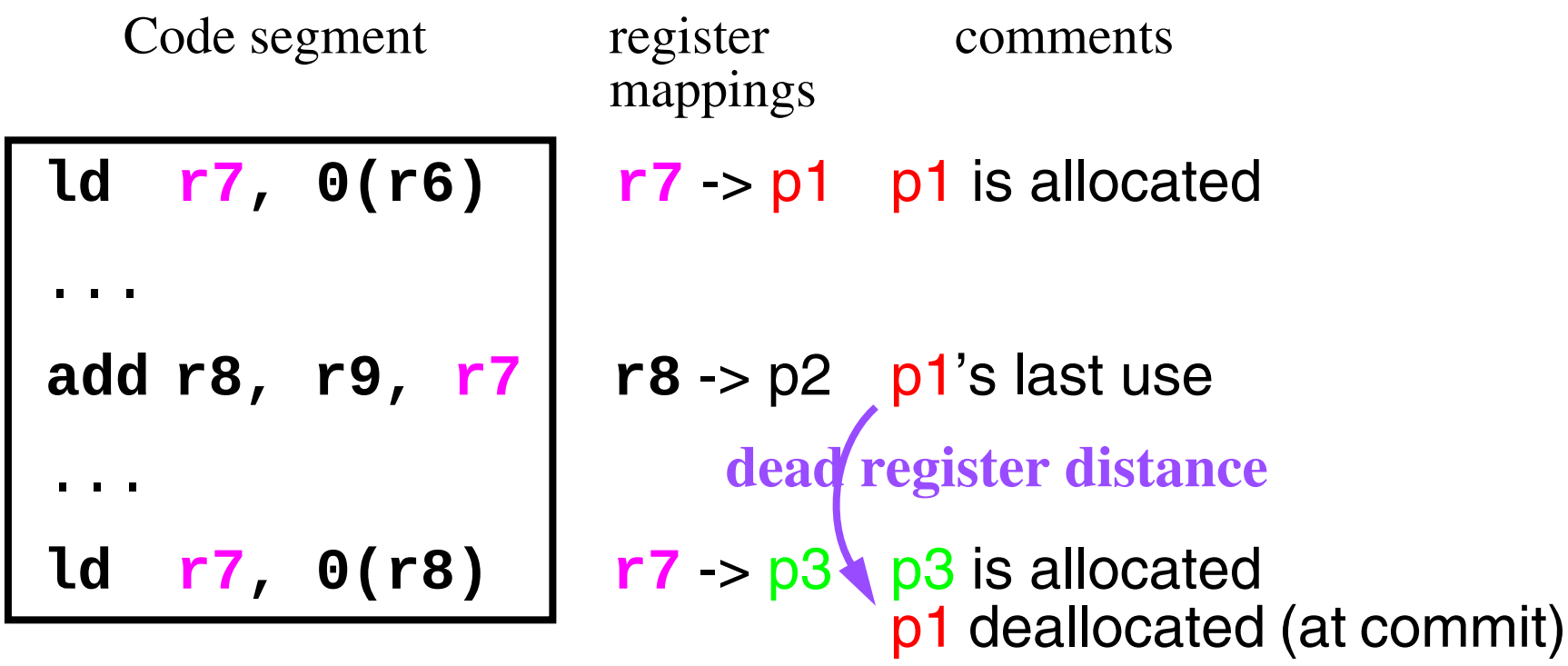
Trade-off:

- SMT register files are large (256+128 registers)
- 12 read ports, 6 wrike ports. (6 FUs)
- Large, multi-ported register files are slow, and right in the middle of the critical path (Maybe).
- Reducing reister file size limits performance because we run out.

Improve utilization of registers to reduce register file size

- Faster access
- Less chip area
- Less power
- More performance fromlarger register files (needs other changes as well)

Register Renaming Inefficiencies



Register deallocation hardware is inefficient

- Only deallocate on next use of arch. register.
- Unaware of live ranges
- So must be conservative

The consequence of hardware register deallocation:

- Many programs ran out of physical registers, causing instruction fetch stalls
- Lots of registers outlive their last use (they are dead).
- big dead register distances

If you could deallocate registers earlier,
might improve register utilization & overall performance

Software-directed Register Deallocation

How can software assist register deallocation hardware?

- The compiler knows, let it identify the last use.
- Provide architectural mechanisms for communicating last use info.

Implementation alternatives:

- additional bits in ISA (hard to come by)
- explicit free-register instructions
- mask instructions
- special forms of existing opcodes

SMT Register Deallocation Strategies

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

+ Small SMT register files achieve large register file performance

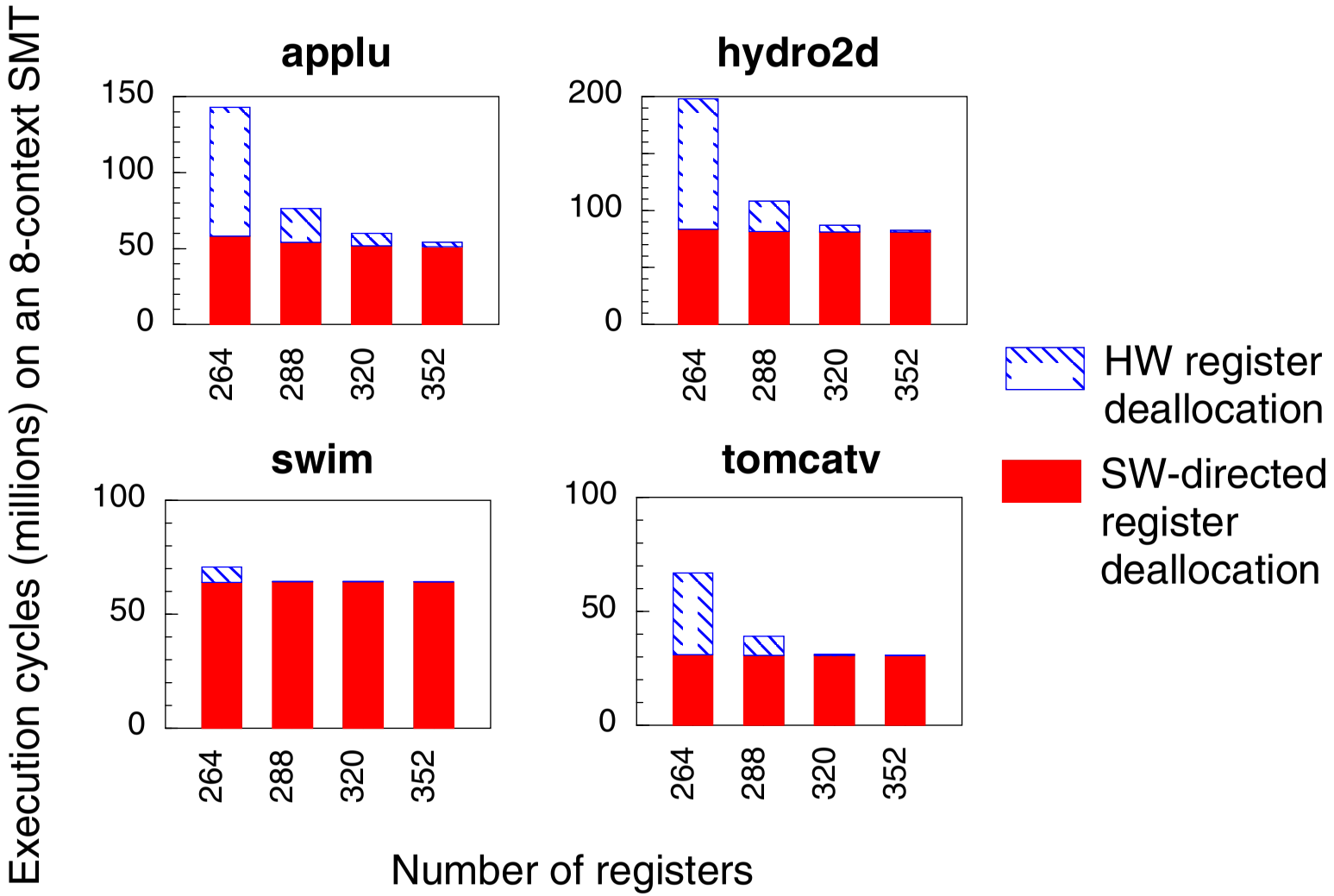
⇒ Use a small register file if the register access sets the cycle time.

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

+ Enables SMT registers to be shared by ***all*** contexts -- traditional multithreaded processors have ***separate*** contexts per thread.
+ Increases small register file performance by:
  - up to 50% with 8 threads
  - 3 times with fewer threads.

⇒ Design a good throughput CPU & not penalize a single-thread

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% (SMT=70)
SMT with compiler/architecture support to free registers after their last use	3% (SMT=8)