# Υποθετικός Πολυνηματισμός

# Τα προβλήματα του παράλληλου προγραμματισμού

- Εντοπισμός παραλληλισμού
  - χειροκίνητα (επισκόπηση)
  - αυτόματα (compiler)
- Έκφραση παραλληλισμού
  - low-level (π.χ. Pthreads, MPI)
  - high-level (π.χ. OpenMP, Cilk, Intel TBB, Galois, UPC κ.λπ.)
- Απεικόνιση
  - scheduling, creation, termination, etc.
  - operating system, runtime system
- Συγχρονισμός
  - εύκολος όπως ο coarse-grain
  - αποδοτικός όπως ο fine-grain
  - deadlock-free
  - composable
- Απαιτήσεις
  - κλιμακωσιμότητα
  - ευκολία προγραμματισμού
  - υψηλή παραγωγικότητα
  - ορθότητα
  - architectural awareness

# Parallelizing Compilers

- Στοχεύουν σε «κανονικές» εφαρμογές
  - for loops, arrays
  - (παραδοασικά, HPC scientific codes…)
- Συνήθως, 2 περάσματα:
  - Είναι ένα loop «ασφαλές» για παραλληλοποίηση;
    - στατική ανάλυση (dependence, alias)
  - «Αξίζει» να παραλληλοποιηθεί;
    - cost model, profiling
  - Παραδείγματα: Intel CC, SUIF-1, Polaris, PGI, Open64

- Τι γίνεται με τις «ακανόνιστες» εφαρμογές;
  - ασαφείς/μη-σταθερές εξαρτήσεις (π.χ. input dependent)
  - «περίπλοκες» δομές

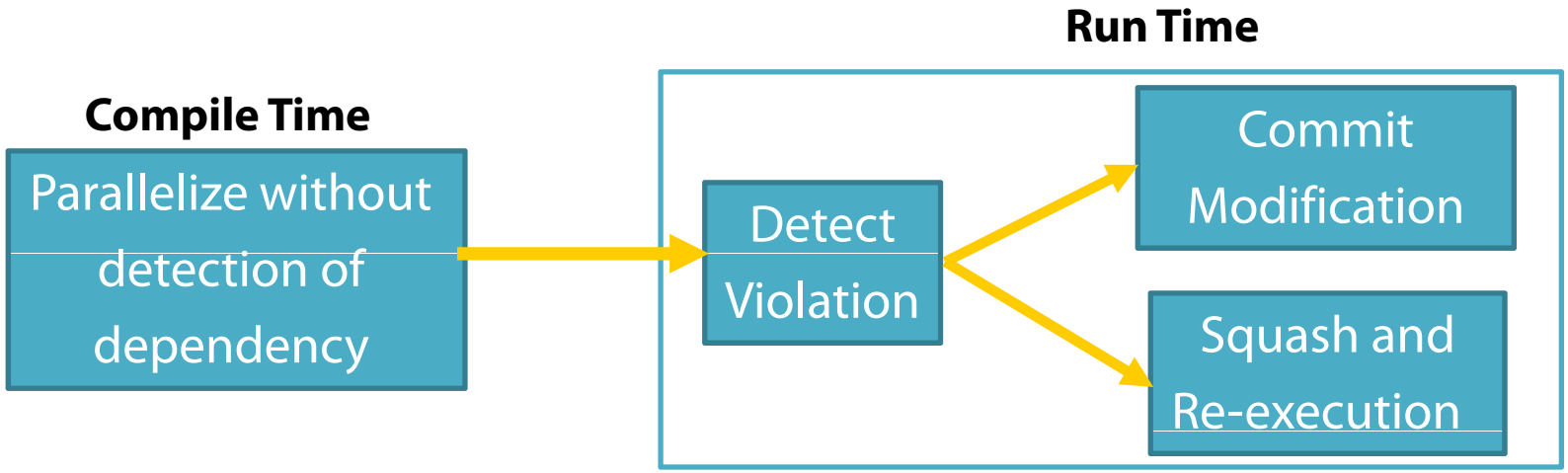
# Ενδεικτική κατηγοριοποίηση

| **Χαρακτηριστικά κώδικα** | **Παράδειγμα** | **Πεδίο εφαρμογής** |
|---|---|---|
| Arrays with direct addressing | for(i = 1; i<n; i++) {<br>A[i] = B[i]*C[i];<br>} | • scientific codes<br>• media applications |
| Arrays with indirect addressing | for(i = 1; i<n; i++) {<br>z = A[K[i]];<br>A[L[i]] = z + C[i];<br>} | • circuit simulations<br>• structural mechanics modeling<br>• molecular dynamics simulation<br>• fluid flows |
| Recursive Data Structures (RDS):<br>• trees<br>• lists<br>• graphs<br>• sets<br>• hash-tables<br>• … | while(ptr=ptr->next) {<br>ProcessElement(ptr->val);<br>} | • graph algorithms (shortest paths, minimum spanning trees, max flows)<br>• simulations (N-body, graphics)<br>• meshes (refinement, triangulation)<br>• dynamic programming<br>• data mining<br>• … |

# Υποθετικός Πολυνηματισμός

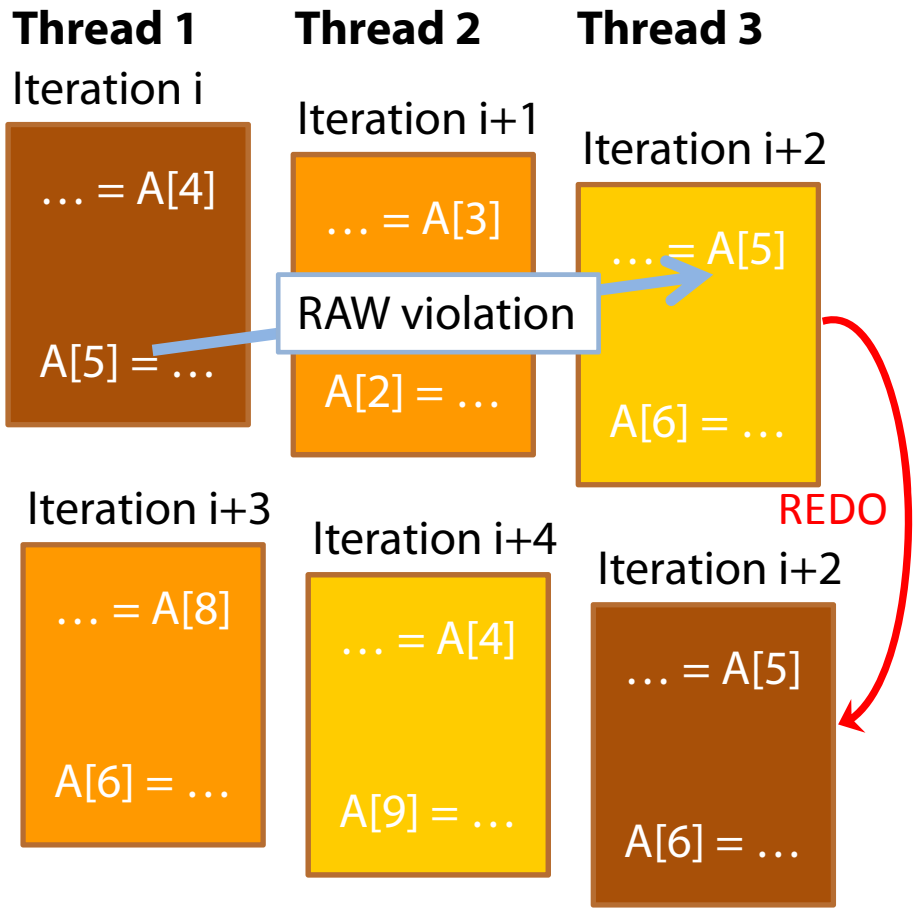
Speculative Multithreading or Thread-Level Speculation

- επιτρέπει τη δημιουργία και εκτέλεση παράλληλων νημάτων παρά τις πιθανές εξαρτήσεις δεδομένων
- ελπίζουμε στην μη-ύπαρξη εξαρτήσεων
  - «αισιόδοξος» παραλληλισμός



# Παράδειγμα

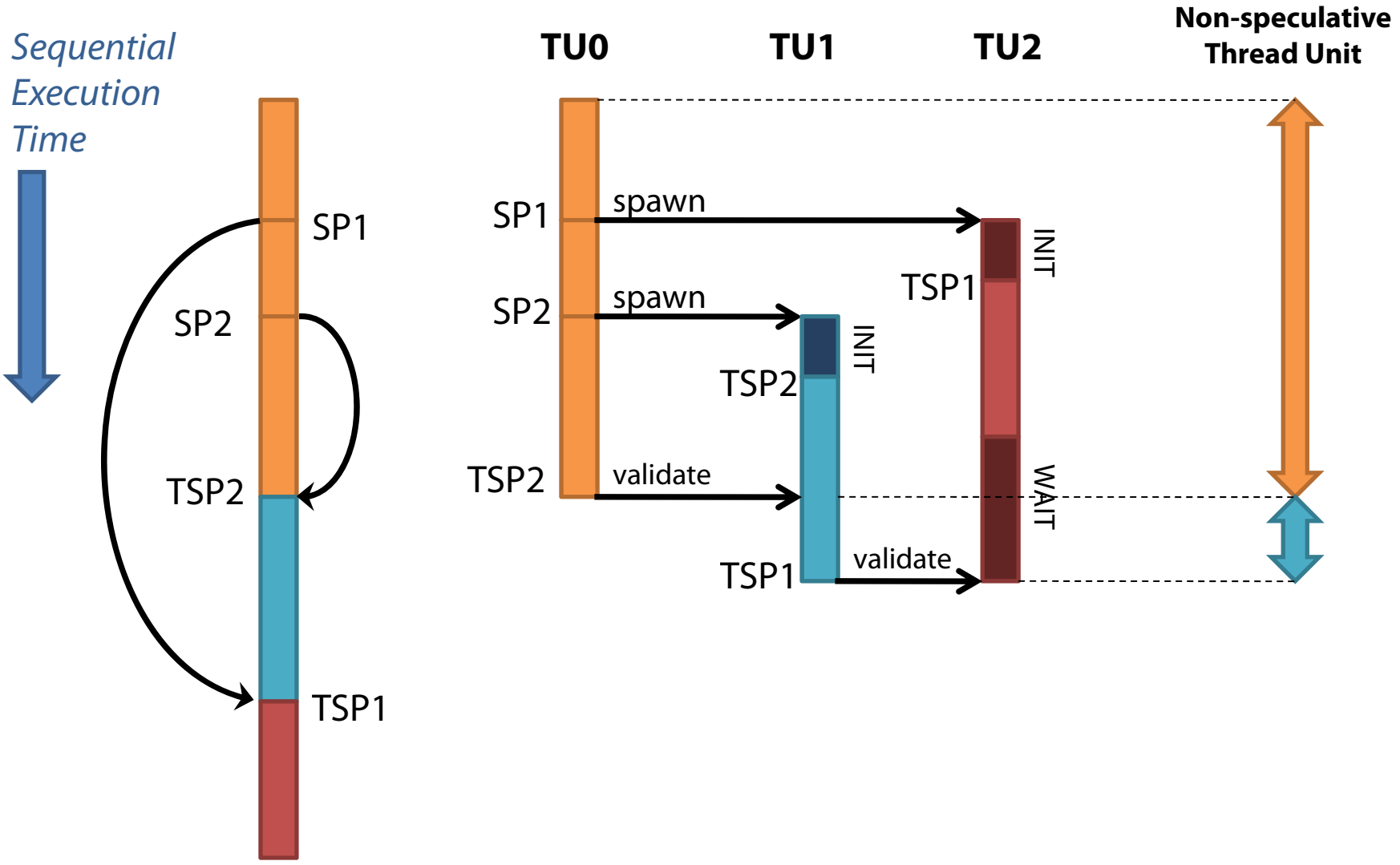
```
for(i=0; i<n; i++) {
    … = A[B[i]];
    …
    A[C[i]] = …
}
```



# Γενικό Μοντέλο

- 1 main (non-speculative) thread
  - γηραιότερο
  - το μόνο που επιτρέπεται να κάνει commit
- N ≥ 0 speculative threads
- σχέσεις predecessor-successor ανάμεσα στα threads
  - sequential semantics
- οποιοδήποτε thread μπορεί να κάνει spawn κάποιο άλλο (speculative) thread

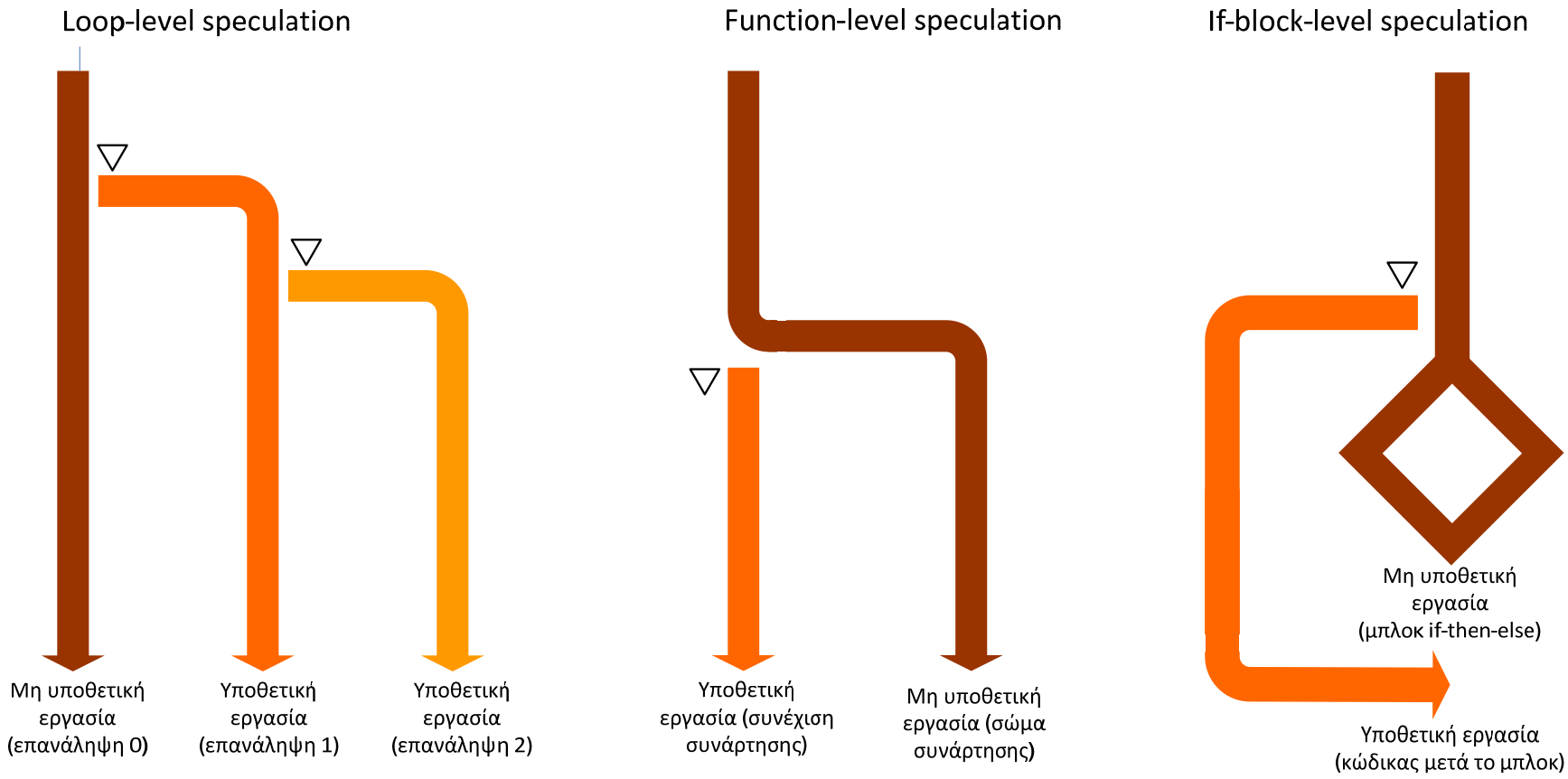
# Γενικό Μοντέλο




SP: spawning point
TSP: thread start point

# Προγραμματιστικές Δομές

- ξεκινάμε από το σειριακό πρόγραμμα και εξάγουμε «ordered» νήματα (ή εργασίες )



# Υποθετικός Πολυνηματισμός

Το HW/SW (runtime system) παρέχει υποστήριξη για:

- checkpointing των καταχωρητών στην αρχή της εκτέλεσης μιας εργασίας
- buffering της υποθετικής κατάστασης που δημιουργείται
  - π.χ. write-buffer, cache
- παρακολούθηση των λειτουργιών μνήμης των εργασιών
  - παραβίαση εξαρτήσεων: αναίρεση των επιδράσεων της (πιο) υποθετικής εργασίας και επανεκκίνηση
    - π.χ. απόρριψη write-buffer, restore registers
  - μη παραβίαση εξαρτήσεων: commit αποτελεσμάτων των εργασιών
    - στη σειρά προγράμματος!

# Παράδειγμα Ανίχνευσης Παραβιάσεων

- Επέκταση cache με
  - timestamp («epoch number»): δείχνει τη σειρά στο σειριακό πρόγραμμα
  - violation flag
- Επέκταση cache line με ειδικά bits
  - SL: κάποιο speculative load έχει προσπελάσει την cache line
  - SM: η γραμμή έχει τροποποιηθεί υποθετικά
- Σε κάθε write-invalidation, μια εργασία μαρκάρεται για ακύρωση αν:
  - η γραμμή είναι παρούσα στην cache της
  - το SL bit της είναι set
  - το «epoch number» του invalidator υποδεικνύει προγενέστερο νήμα στη σειρά προγράμματος

# Παράδειγμα Ανίχνευσης Παραβιάσεων

**Processor 1**
**Epoch 5**

…

②**STORE *q = 2;**

…

**L1 Cache**

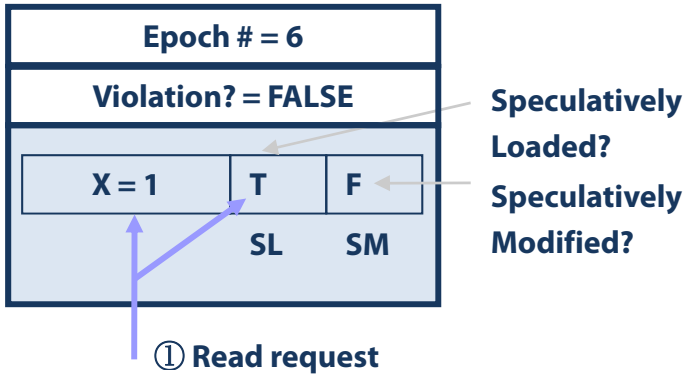
| Epoch # = 5 |
| --- |
| Violation? = FALSE |
| |

**p = q = &x**

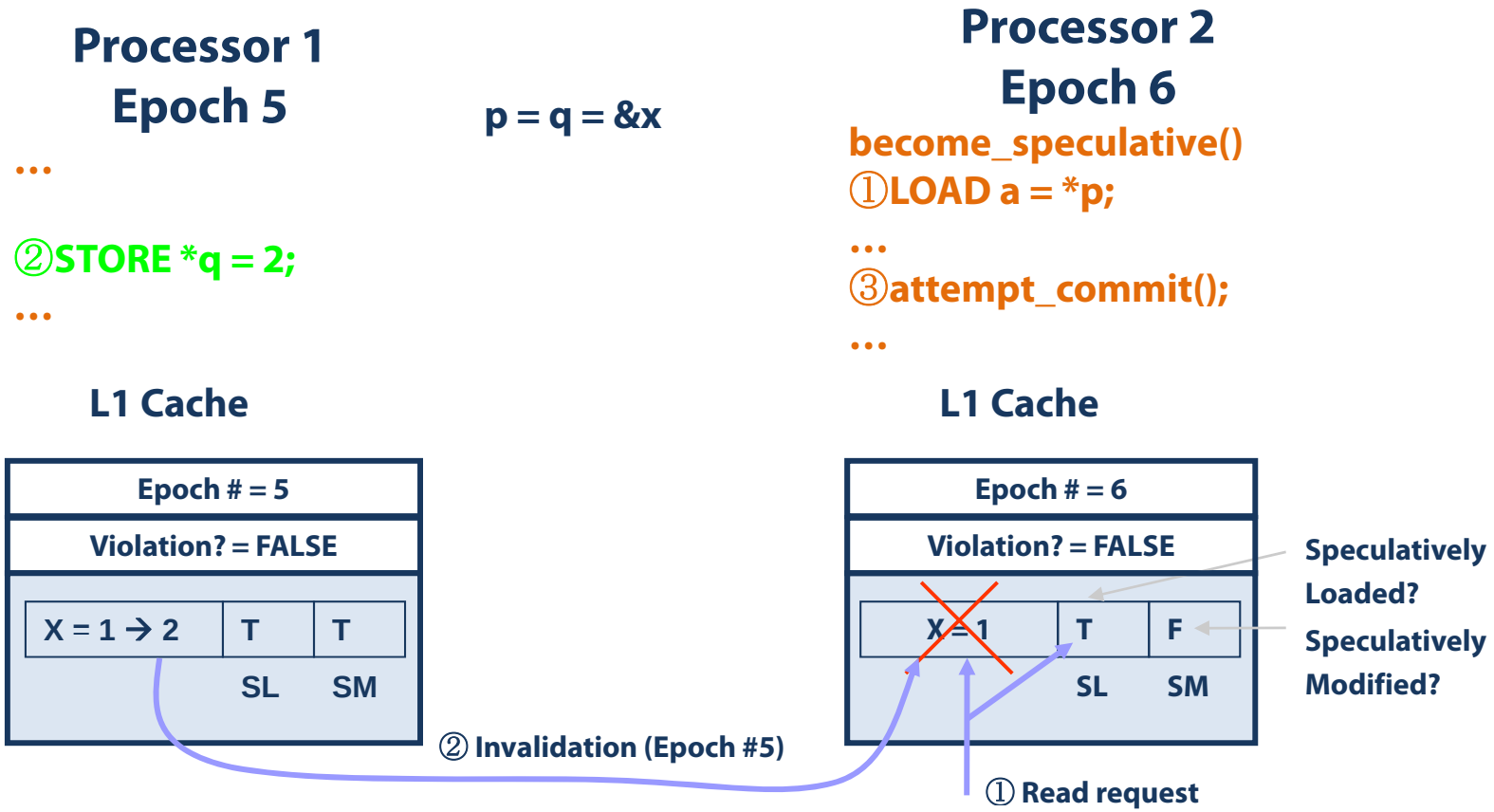
**Processor 2**
**Epoch 6**

**become_speculative()**
①**LOAD a = *p;**
…
③**attempt_commit();**
…

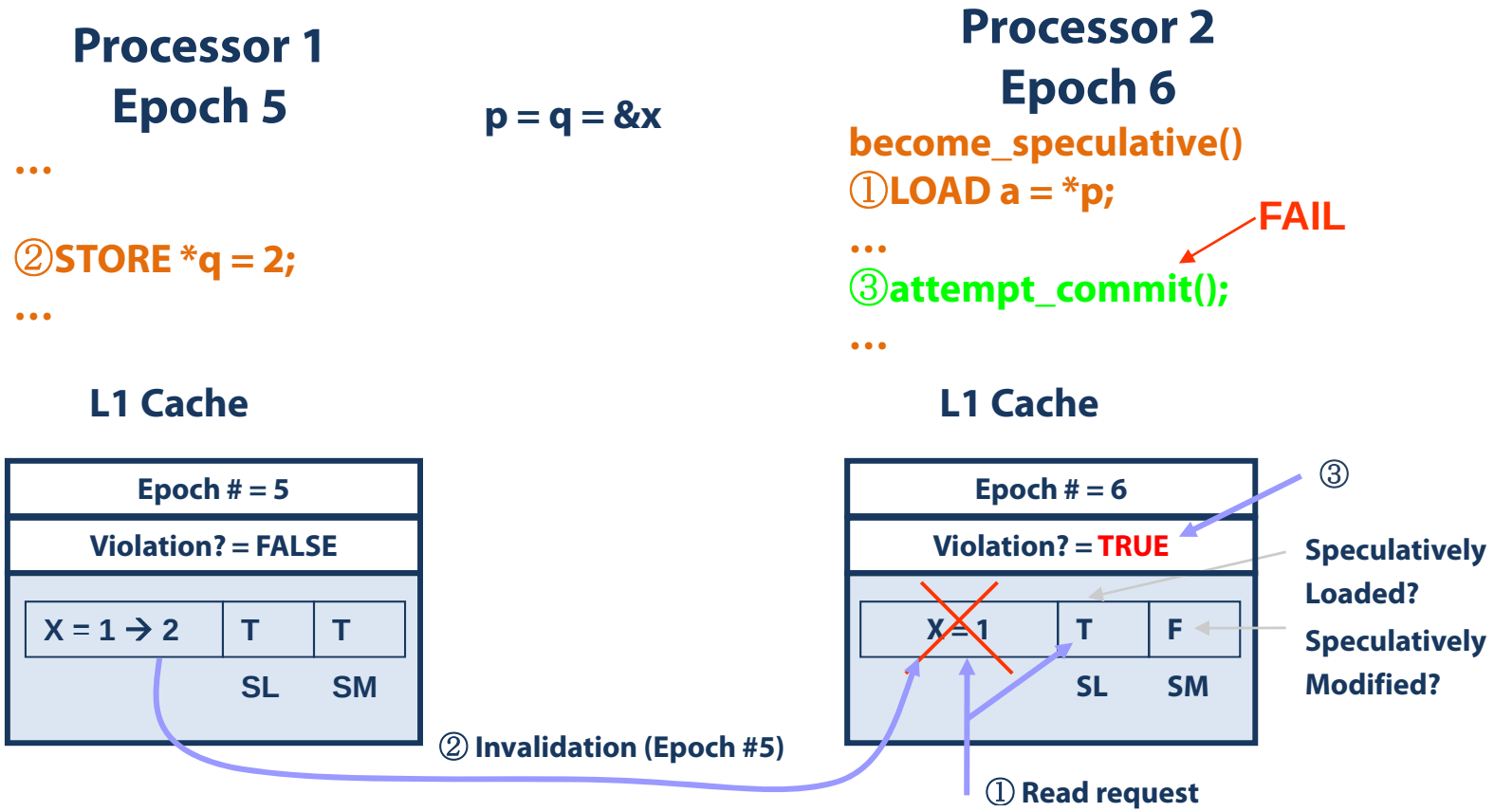
**L1 Cache**



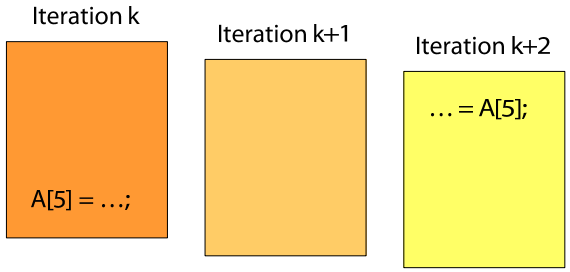
# Παράδειγμα Ανίχνευσης Παραβιάσεων



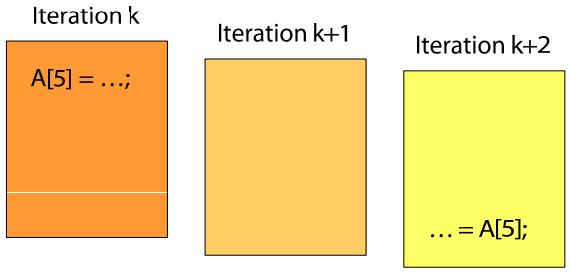
# Παράδειγμα Ανίχνευσης Παραβιάσεων



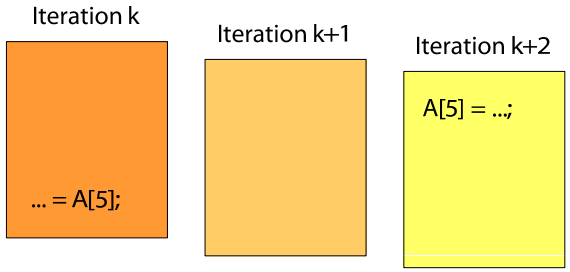
# Τι εξαρτήσεις μας νοιάζουν τελικά;



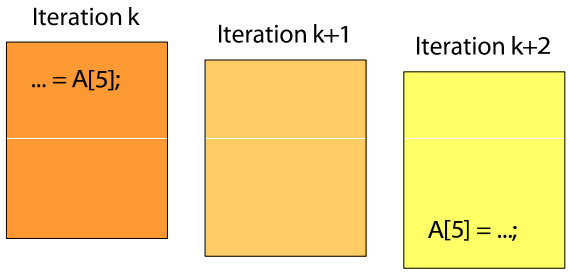

**RAW** dependence violated:
violates sequential semantics (k's store should have preceded k+2's load in program order)




**RAW** dependence respected:
no problem (theoretically), but requires data forwarding mechanism



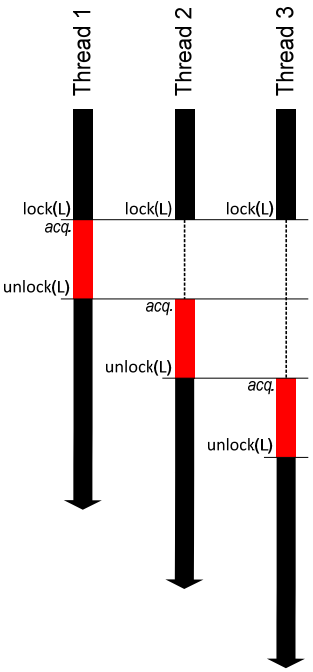

**WAR** dependence violated:
no problem if speculative writes are being buffered

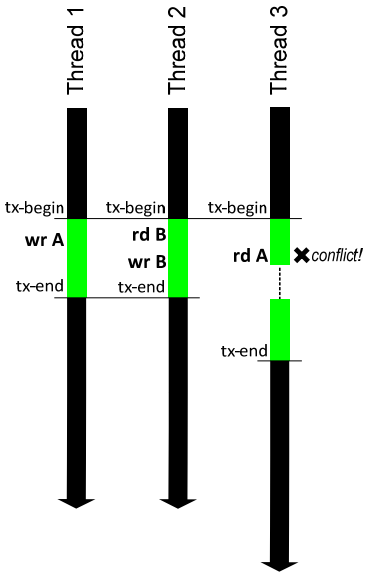



**WAR** dependence respected:
no problem

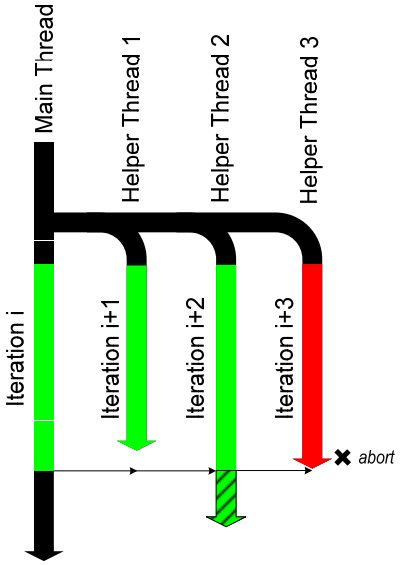
# TM vs TLS

- αισιόδοξος συγχρονισμός

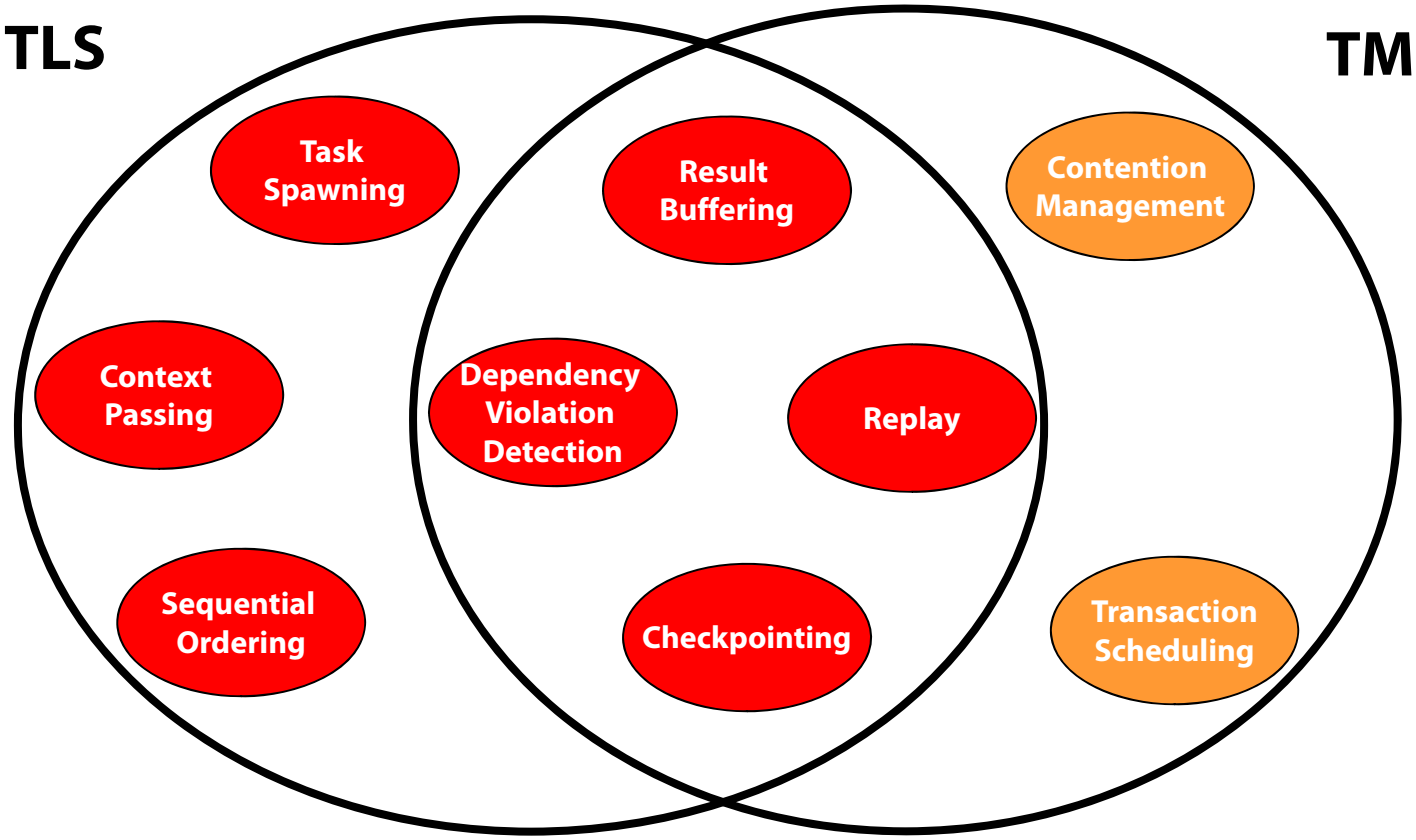



- αισιόδοξος *παραλληλισμός*

# TM vs TLS




**TLS and TM share multiple hardware components**

# Βιβλιογραφία

# Speculative Multithreading στο Hardware


- [Hammond98] L. Hammond, M. Willey and K. Olukotun. “Data Speculation Support for a Chip Multiprocessor”. SIGOPS Oper. Syst. Rev., vol. 32, no. 5, pages 58–69, 1998.
- [Hydra] Hydra project at Stanford University. http://ogun.stanford.edu/
- [Iacoma] I-ACOMA group at University of Illinois at Urbana-Champaign. http://iacoma.cs.uiuc.edu/work/speculation.html
- [Krishnan99] V. Krishnan and J. Torrellas. “A Chip-Multiprocessor Architecture with Speculative Multithreading”. IEEE Trans. Comput., vol. 48, no. 9, pages 866–880, 1999.
- [Multiscalar] Multiscalar project at University of Wisconsin-Madison. http://pages.cs.wisc.edu/~mscalar
- [Renau05] J. Renau, J. Tuck, W. Liu, L. Ceze, K. Strauss and J. Torrellas. “Tasking with Out-of-Order Spawn in TLS Chip Multiprocessors: Microarchitecture and Compilation”. 19th annual International Conference on Supercomputing (ICS ’05).
- [Quinones05] C. Quinones, C. Madriles, J. Sanchez, P. Marcuello, A. Gonzalez, D. Tullsen. "Mitosis Compiler: An Infrastructure for Speculative Threading Based on Pre-Computation Slices" . ACM SIGPLAN conference on Programming language design and implementation (PLDI ’05)
- [Sohi95] G. Sohi, S. Breach and T. Vijaykumar. “Multiscalar Processors”. 22nd Annual International Symposium on Computer architecture (ISCA ’95).
- [Stampede] STAMPede project at Carnegie-Mellon University. http://www.cs.cmu.edu/~stampede/
- [Steffan98] G. Steffan and T. Mowry. “The Potential for Using Thread-Level Data Speculation to Facilitate Automatic Parallelization”. IEEE 4th International Symposium on High Performance Computer Architecture (HPCA ’98).
- [Steffan00] G. Steffan, C. Colohan, A. Zhai and T. Mowry. “A Scalable Approach to Thread-Level Speculation”. 27th Annual International Symposium on Computer Architecture (ISCA ’00).

# Speculative Multithreading στο Software


- [Berger09] E. Berger, T. Yang, T. Liu, G. Novark. "Grace: Safe Multithreaded Programming for C/C++". 24th ACM SIGPLAN conference on Object oriented programming systems languages and applications (OOPSLA'09).
- [Kulkarni07] M. Kulkarni, K. Pingali, B. Walter, G. Ramanarayanan, K. Bala and L. Chew. "Optimistic Parallelism Requires Abstractions". ACM SIGPLAN 2007 conference on Programming language design and implementation (PLDI '07), June 2007
- [Rauchwerger95] L. Rauchwerger and D. Padua. "The LRPD test: speculative run-time parallelization of loops with privatization and reduction parallelization". ACM SIGPLAN 1995 conference on Programming language design and implementation (PLDI '95). p.218-232, June 18-21, 1995, La Jolla, California, United States.
- [Tian08] C. Tian, M. Feng, V. Nagarajan and R. Gupta. "Copy or Discard Execution Model for Speculative Parallelization on Multicores." 41st IEEE/ACM International Symposium on Microarchitecture (MICRO '08), pages 330–341, Washington, DC, USA, 2008. IEEE Computer Society.
- [von Praun07] C. von Praun, L. Ceze, C. Cascaval. "Implicit Parallelism with Ordered Transactions". 2th ACM SIGPLAN symposium on Principles and practice of parallel programming (PPoPP'07).

# Compiler Frameworks for Spec. Multithreading

- [Bhowmik02] A. Bhowmik and M. Franklin. “A general compiler framework for speculative multithreading”. 14th annual ACM symposium on Parallel algorithms and architectures (SPAA ’02).
- [Liu06] W. Liu, J. Tuck, L. Ceze, W. Ahn, K. Strauss, J. Renau and J. Torrellas. “POSH: a TLS compiler that exploits program structure”. 11th ACM SIGPLAN symposium on Principles and practice of parallel programming (PPoPP ’06).

# Parallelizing Compilers


- [Allen02] R. Allen and K. Kennedy. "Optimizing Compilers for Modern Architectures: A Dependence-Based Approach". Morgan Kaufmann, 2002.
- [Brandes97] T. Brandes, S. Chaumette, M. Counilh, J. Roman, A. Darte, F. Desprez and J. Mignot. "HPFIT: a set of integrated tools for the parallelization of applications using high performance Fortran. part I: HPFIT and the Transtool environment". Parallel Computing, 23(1-2), 1997.
- [Bugnion96] E. Bugnion, J. Anderson, T. Mowry, M. Rosenblum and M. Lam. "Compiler-Directed Page Coloring for Multiprocessors". International Symposium on Architectural Support for Programming Languages and Operating Systems (ASPLOS '96).
- [Goff91] G. Goff, K. Kennedy and C. Tseng. "Practical Dependence Testing". ACM SIGPLAN Conference in Programming Language Design and Implementation (PLDI '91).
- [Hall96] M. Hall, J. Anderson, S. Amarasinghe, B. Murphy, S. Liao, E. Bugnion and M. Lam. "Maximizing multiprocessor performance with the SUIF compiler". IEEE Computer, 29(12), 1996.
- [ICC] Intel C/C++ compiler. http://software.intel.com/en-us/intel-compilers/
- [Ishihara06] M. Ishihara, H. Honda and M. Sato. "Development and implementation of an interactive parallelization assistance tool for OpenMP: iPat/OMP". IEICE - Transactions on Information and Systems, E89-D(2), 2006.
- [Kennedy91] K. Kennedy, K. McKinley, and C. Tseng. "Interactive parallel programming using the Parascope editor". IEEE TPDS, 2(3), 1991.
- [Open64] Open64. http://www.open64.net
- [Padua86] D. Padua and M. Wolfe, "Advanced Compiler Optimizations for Supercomputers," Comm. ACM, vol. 29, pp. 1,184–1,201, Dec. 1986.
- [Padua93] D. Padua, R. Eigenmann, J. Hoeflinger, P. Petersen, P. Tu, S. Weatherford and K. Faigin. "Polaris: A new-generation parallelizing compiler for MPPs". Technical report, In CSRD No. 1306. UIUC, 1993.
- [Wolfe89] M. Wolfe. "Optimizing Compilers for Supercomputers". The MIT Press, 1989.

# Transactional Memory & Spec. Multithreading


- [Baek07] W. Baek, C. Cao Minh, M. Trautmann, C. Kozyrakis and K. Olukotun. "The OpenTM Transactional Application Programming Interface". 16th International Conference on Parallel Architecture and Compilation Techniques (PACT '07).
- [Guo08] R. Guo, H. An, R. Dou, M. Cong, Y. Wang and Q. Li. "LogSPoTM: A Scalable Thread Level Speculation Model Based on Transactional Memory". 13th Asia-Pacific Computer Systems Architecture Conference (ACSAC '08).
- [Porter09] L. Porter, B. Choi and D. Tullsen. "Mapping Out a Path from Hardware Transactional Memory to Speculative Multithreading". 18th International Conference on Parallel Architectures and Compilation Techniques (PACT '09).
- [Yoo08] R. Yoo and H. Lee. "Helper Transactions: Enabling Thread-Level Speculation via A Transactional Memory System". Workshop on Parallel Execution of Sequential Programs on Multicore Architectures (PESPMA '08).