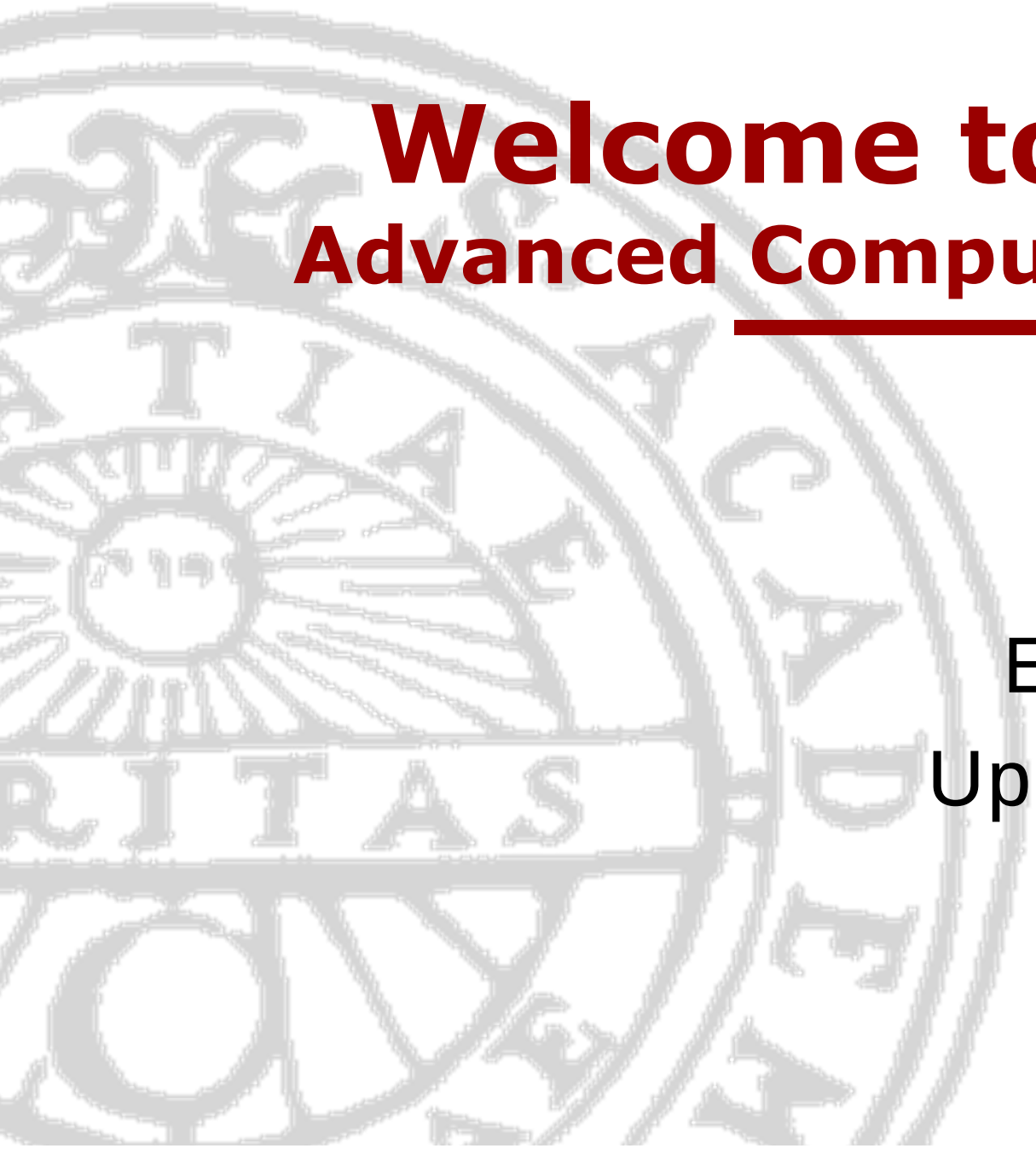


Welcome to "AVDARK"
Advanced Computer Architecture

Erik Hagersten
Uppsala University
Sweden

UPPSALA UNIVERSITET

Ericsson, CPU designer 1982-84: ➔APZ212 ”supercomputer”

MIT 1984-85: ➔Dataflow parallel architectures

Ericsson computer science lab 1985-1988: ➔NetInsight, (Erlang)

SICS, parallel architectures 1988-1993 ➔COMA, (Simics & Virtutech)

Sun Microsystems, chief architect servers 1993 – 1999 ➔
Servers: WildFire, E6000, E6500, E15000, E25000

Startup: Acumem AB 2006 – 2010 ➔ ThreadSpotter

Chief scientist Rogue Wave Software Inc. 2010 – 2012

Professor Uppsala University 1999 – ➔ New modeling + Acumem

AVDARK 2013

UPPSALA UNIVERSITET

AVDARK in a nutshell

Lecturer — Erik Hagersten gives most lectures and is responsible for the course Andreas Sembrant and Mahdad Davari are responsible for the hand-in grading and the labs

Literature — Computer Architecture A Quantitative Approach (5th edition) John Hennesey & David Pattersson [Optional]

Mandatory Assignment — There are four lab assignments that all participants have to complete before a hard deadline. Each lab can earn you bonus points for the exam.

Optional Assignment — There are four (optional) hand-in assignments. Each can earn you a bonus points for the exam.

Examination — Closed-book written exam at the end of the course.

Bonus system — Exam: 64p max, 32p to pass.
(All bonus points = 32p)

UPPSALA UNIVERSITET

Schedule in a nutshell: 5 Batches

1. **Memory systems**
   Caches, optimizing SW, VM, DRAM, microbenchmarks.

2. **Multiprocessors**
   Coherence, memory models, interconnects, scalability, clusters, …

3. **Scalable multiprocessors**
   Scalability, synchronization, clusters, …

4. **CPUs**
   ILP: pipelines, scheduling, superscalars, VLIWs, Vector instructions…

5. **Widening the view** (no lab, hand-in or bonus)
   Technology impact, GPUs, multicores, future trends

AVDARK 2013

UPPSALA UNIVERSITET

Contents of each batch 1-4

- **Recorded on-line lectures teaching the basic facts**
  - Interactive quizzes during lectures
  - ”Question” button to ask me on-line questions during the lectures (answered in forum)
  - Or, press ”Confused” button if you do not understand
  - Answers quizzes/Questions/Confused clicks are recorded anonymously
  - Have to watch all lectures before a deadline to get hand-in bonus
- **Two IRL lectures**
  - Covering Confusions and Question
  - Covering more complex issues
  - In-class examples and exercises (similar to exam questions)
  - Have to attend both IRL lectures to get hand-in bonus
- **One optional hand-in.**
  - Complete before a deadline for hand-in bonus
- **One mandatory lab**
  - Lab preparation lecture (have to attend for lab bonus)
  - Lab preparation slot (computer lab available, no lab teacher)
  - 3 Lab slots of 4h each (group A, B or C). Demo for lab teacher.
  - Complete extra assignment during lab slot for lab bonus

UPPSALA UNIVERSITET

In Summary

- 4 Labs (mandatory)
- 4 Bonus assignments per lab (optional)
- 4 Hand-in (optional)
- Written Exam

Each bonus automatically gives you full score (4p) for a corresponding question at the exam

➔ 32p/64p at the exam = PASS

AVDARK 2013

UPPSALA UNIVERSITET

AVDARK on the Web

www.it.uu.se/edu/course/homepage/avdark/ht13

Welcome!
News
FAQ
Schedule
Slides
New Papers
Assignments
Reading instructions
Exam

AVDARK 2013

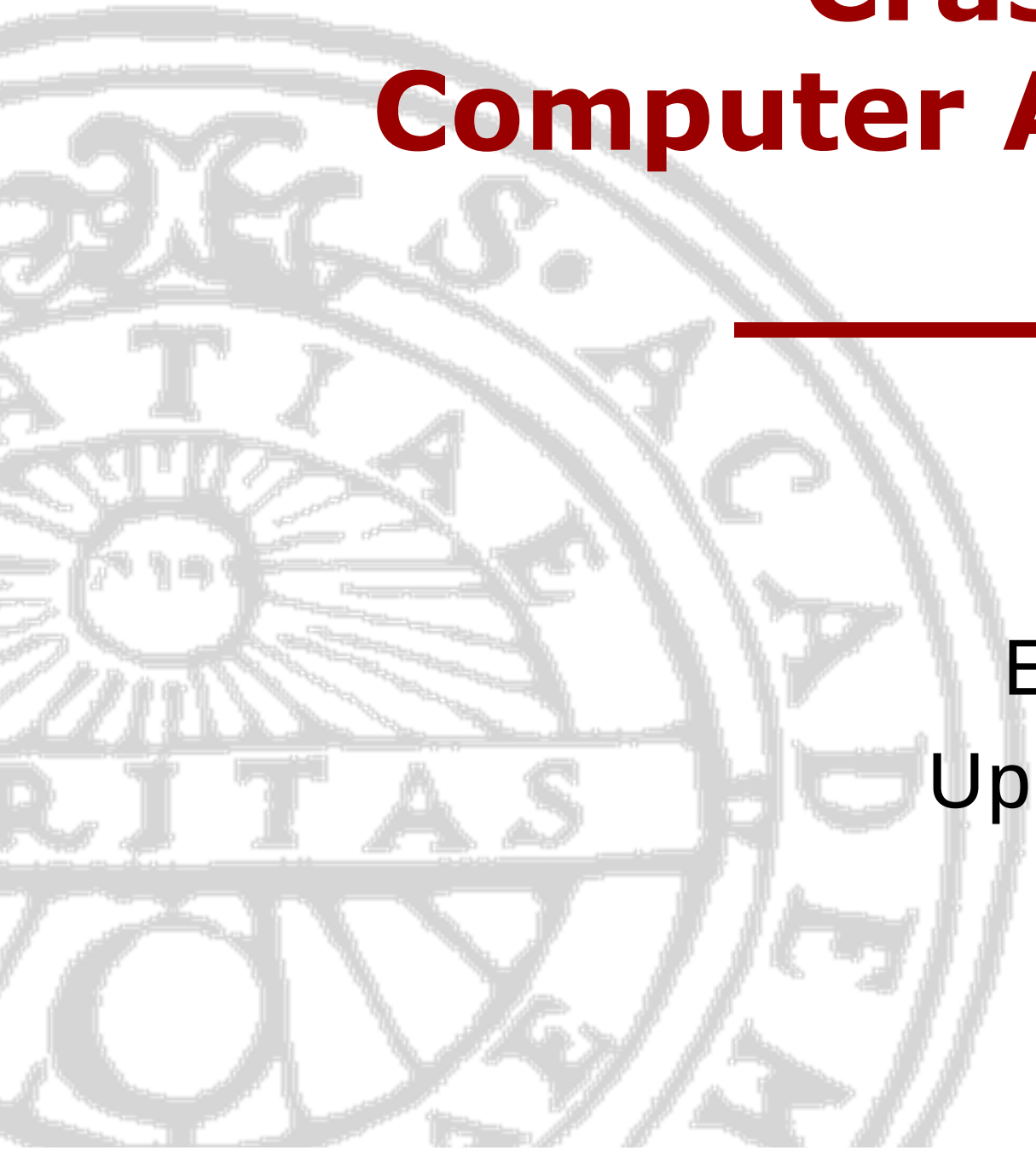
UPPSALA UNIVERSITET

Goal for this course

- Understand **how and why** modern computer systems are designed the way the are:
  - pipelines
  - memory organization
  - virtual/physical memory …
- Understand **how and why** multiprocessors are built
  - Cache coherence
  - Memory models
  - Synchronization…
- Understand **how and why** parallelism is created and leveraged
  - Instruction-level parallelism
  - Memory-level parallelism
  - Thread-level parallelism…
- Understand **how and why** multiprocessors of combined SIMD/MIMD type are built
  - GPU
  - Vector processing…
- Understand **how** computer systems are adopted to different usage areas
  - General-purpose processors
  - Embedded/network processors…
- Understand the physical limitation of modern computers
  - Bandwidth
  - Energy
  - Cooling…

AVDARK 2013

Crash Course in Computer Architecture

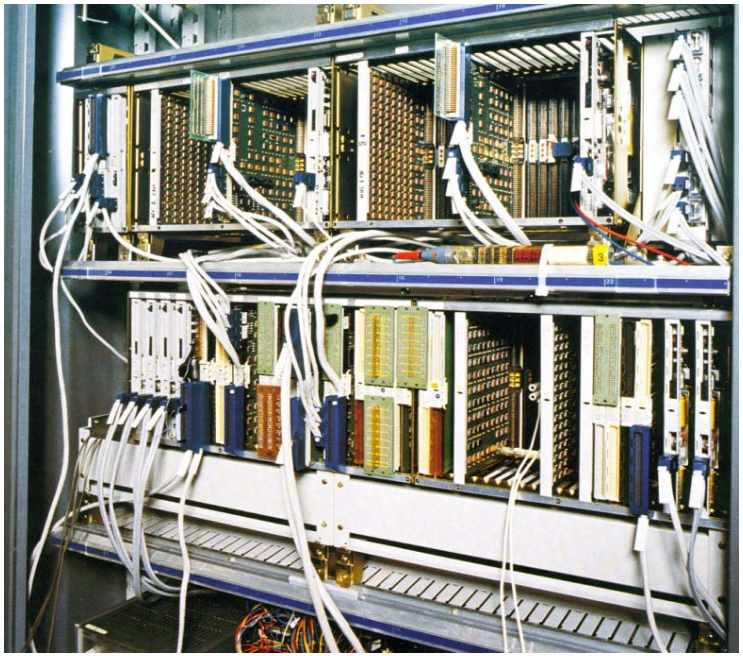


Erik Hagersten
Uppsala University

UPPSALA UNIVERSITET

≈30 years ago: APZ 212 @ 5MHz

”the AXE supercomputer”



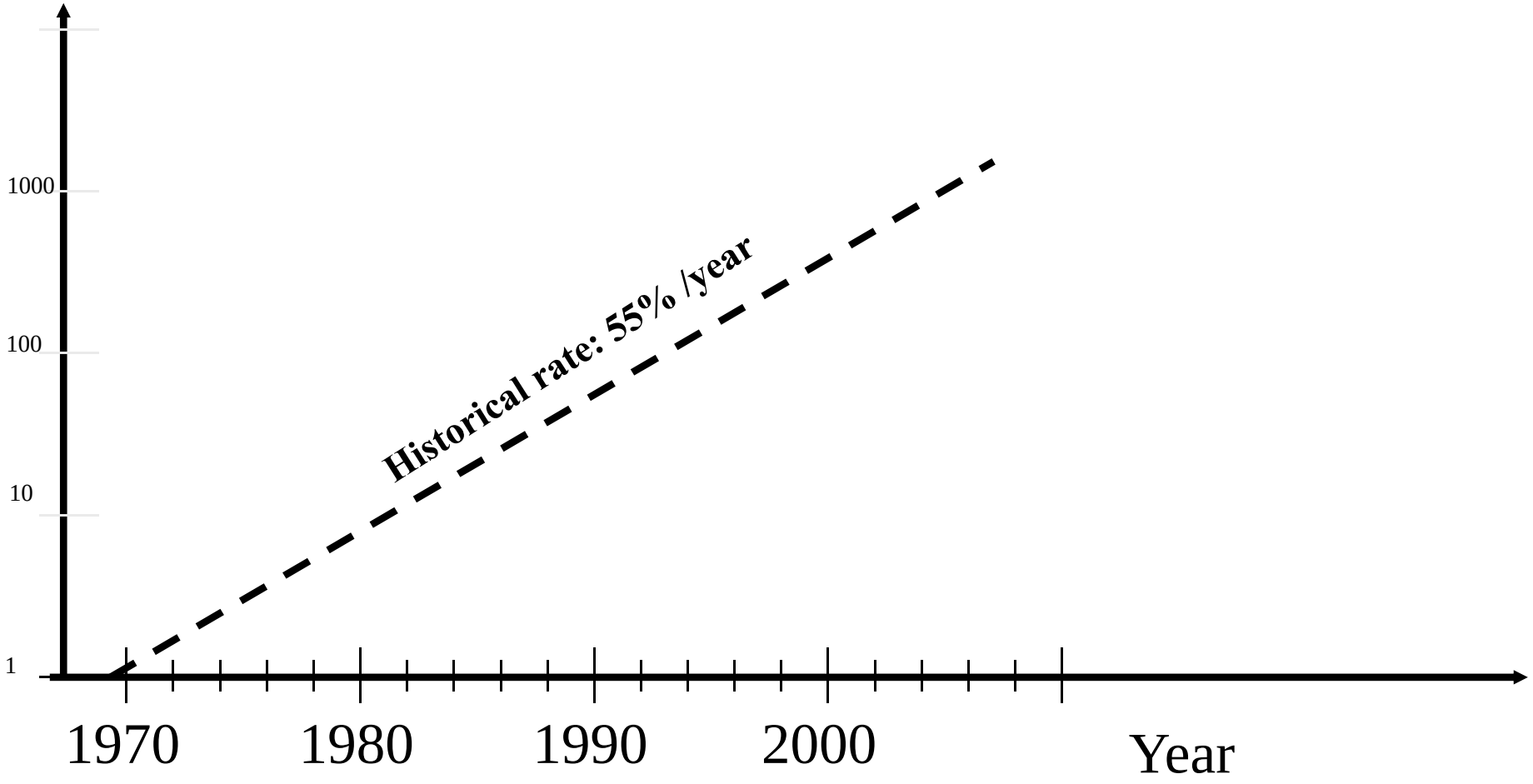
AVDARK 2013

APZ 212 marketing brochure quotes:

- ”Very compact”
  - 6 times the performance
  - 1/6:th the size
  - 1/5 the power consumption
- ”A breakthrough in computer science”
- ”Why more CPU power?”
- ”All the power needed for future development”
- ”…800,000 BHCA, should that ever be needed”
- ”SPC computer science at its most elegance”
- ”Using 64 kbit memory chips”
- ”1500W power consumption

CPU Improvements

Relative Performance
[log scale]



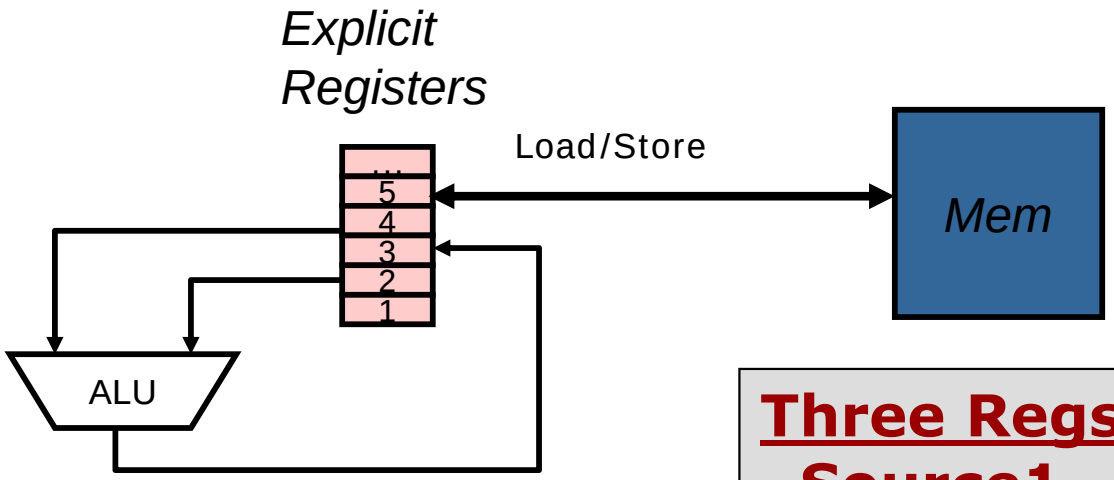
AVDARK
2013

UPPSALA UNIVERSITET

How to get efficient architectures…

- ~~Increase clock frequency~~ **Very hard today**
- Create and explore locality:
  a) Spatial locality
  b) Temporal locality
  c) Geographical locality
- Create and explore parallelism
  a) Instruction level parallelism (ILP)
  b) Thread level parallelism (TLP)
  c) Memory level parallelism (MLP)
- Speculative execution
  a) Out-of-order execution
  b) Branch prediction
  c) Prefetching

Load/Store architecture (e.g., "RISC")
ALU ops: Reg -->Reg
Mem ops: Reg <--> Mem



Example: C = A + B → Compiler

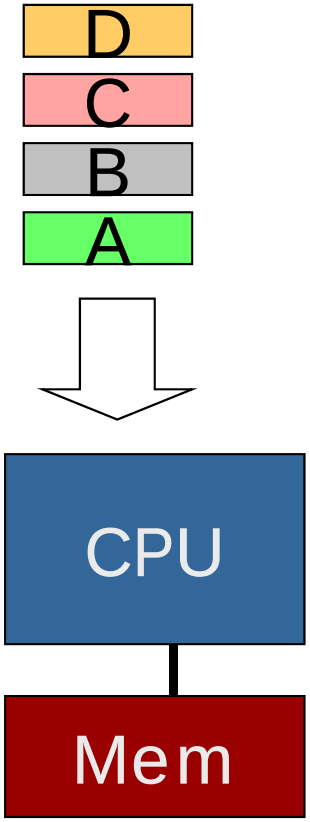
```
LD R1, [A]
LD R3, [B]
ADD R2, R1, R3
ST R2, [C]
```

Three Regs:
- Source1
- Source2
- Destination

Memory Accesses

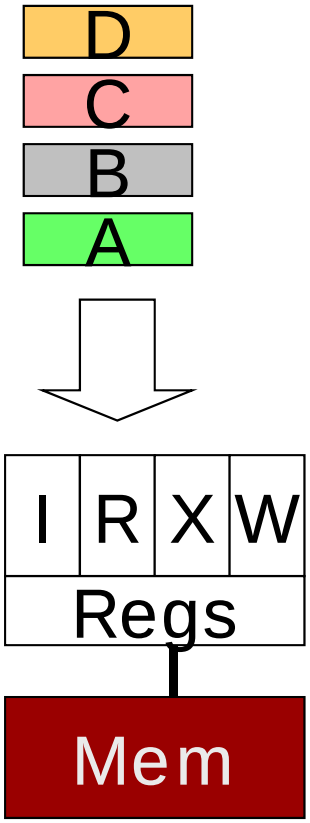
Lifting the CPU hood (simplified…)

Instructions:

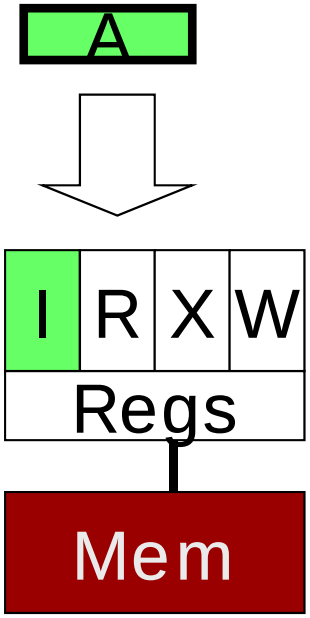


Pipeline

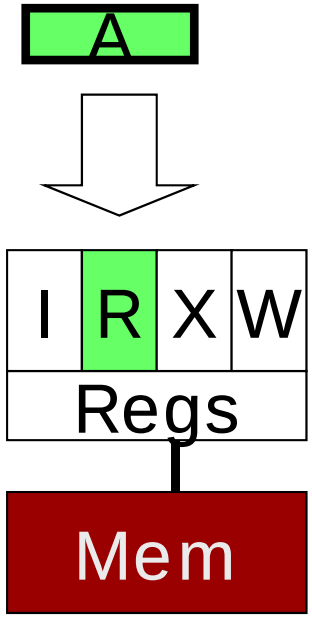
Instructions:



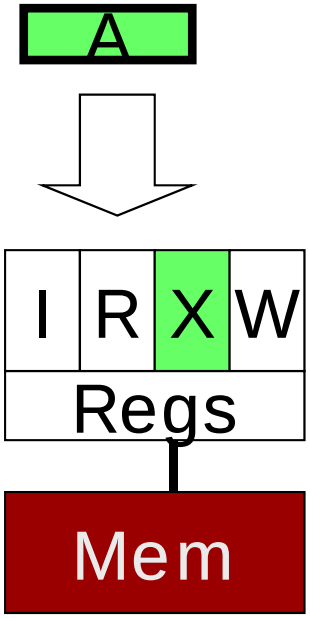
Pipeline



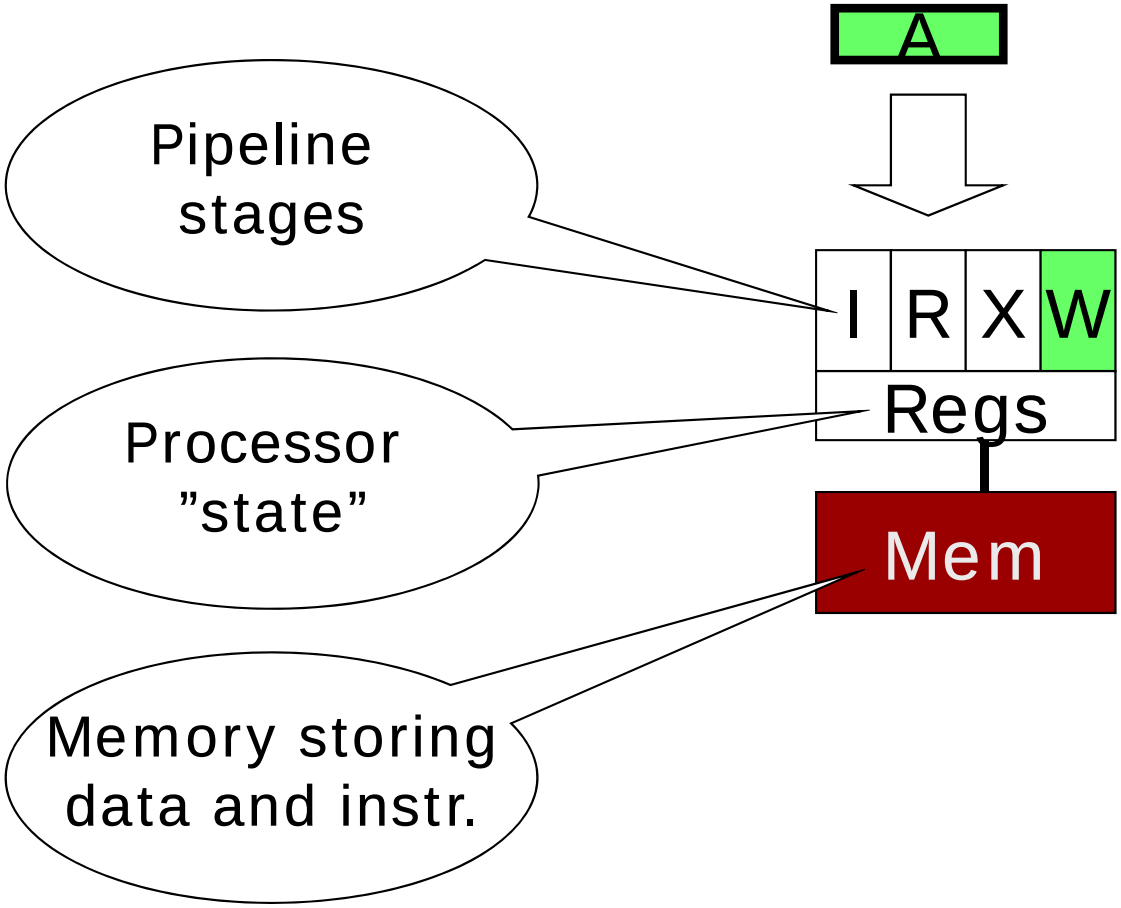
Pipeline



Pipeline



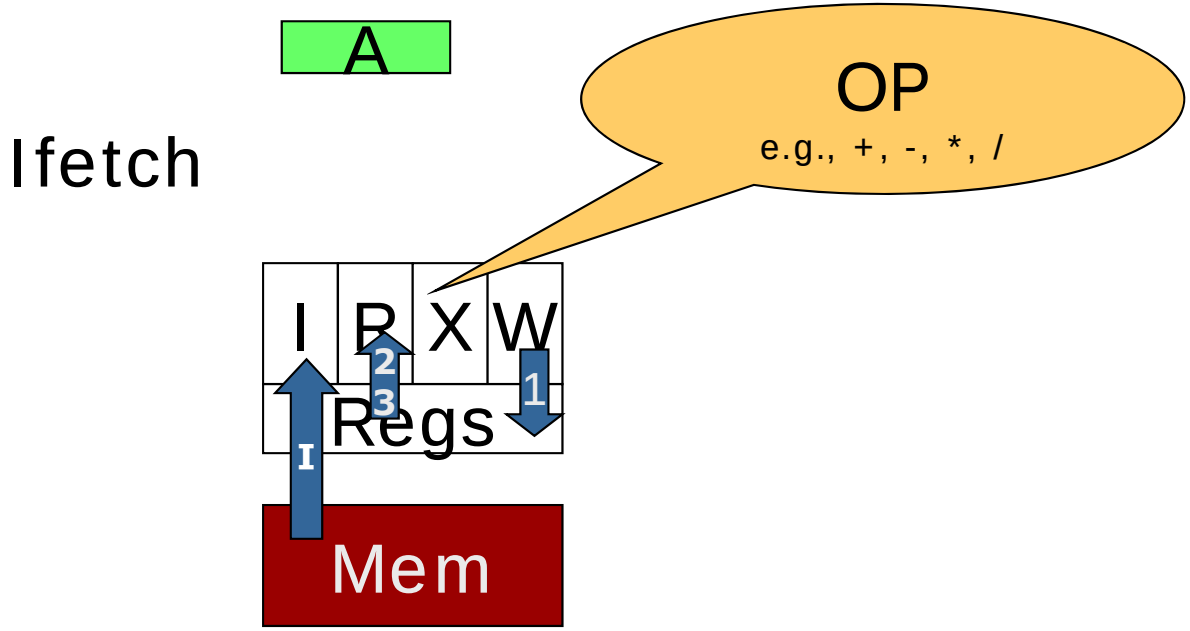
Pipeline:



I = Instruction fetch
R = Read register
X = Execute
W= Write register/memory

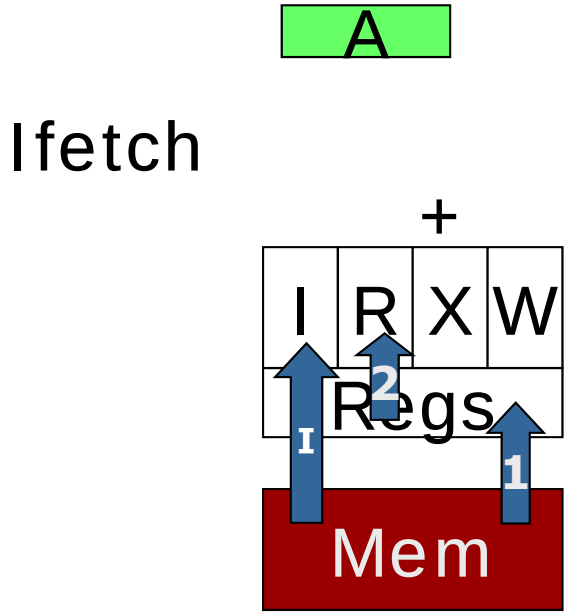
Register Operations [aka ALU operation]

ADD R1, R2, R3
a.k.a. R1 := R2 op R3



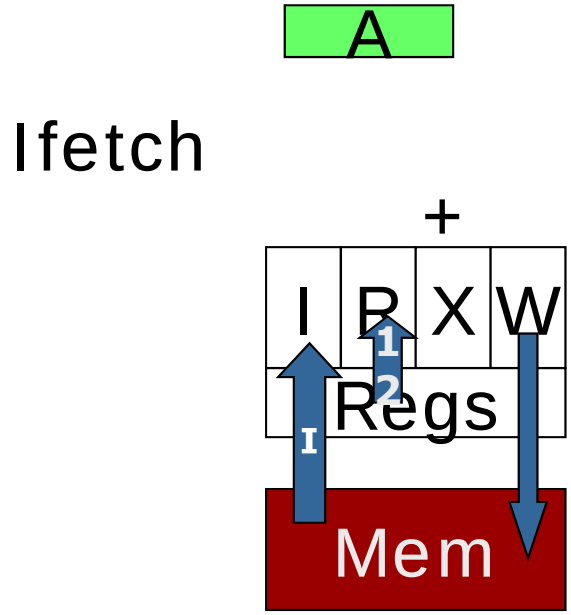
Load Operation:

LD R1, mem[cnst+R2]



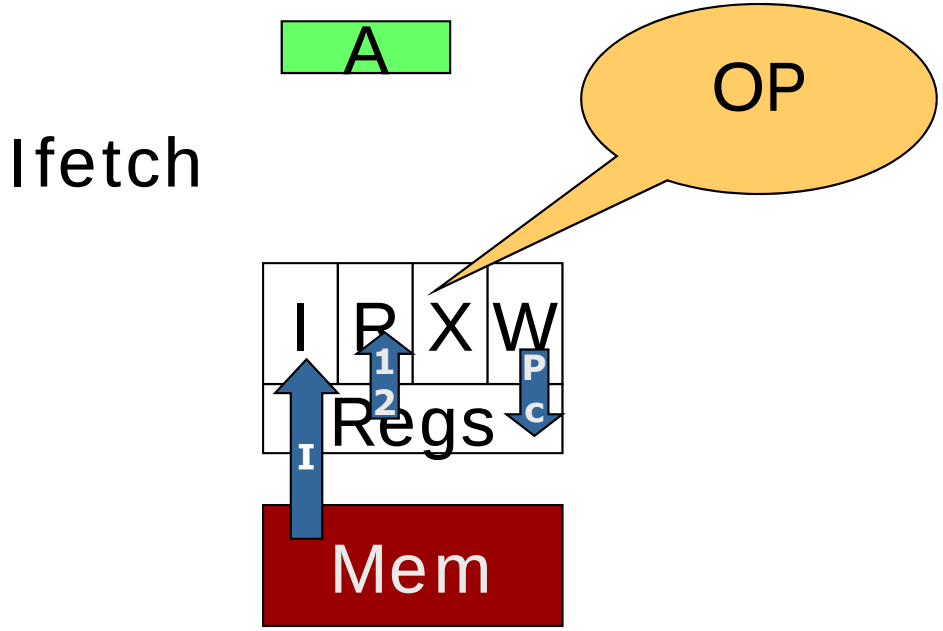
Store Operation:

ST R2, mem[cnst+R1]



Branch Operations:

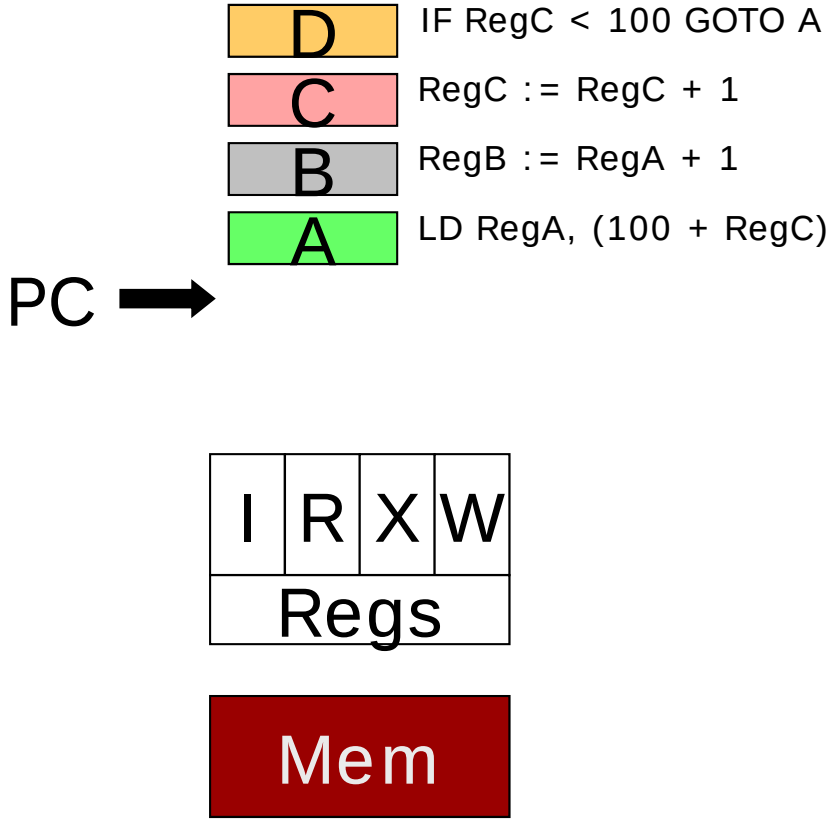
if (R1 Op Const) GOTO mem[R2]



PC = Program Counter.
A special register pointing to the next instruction to execute

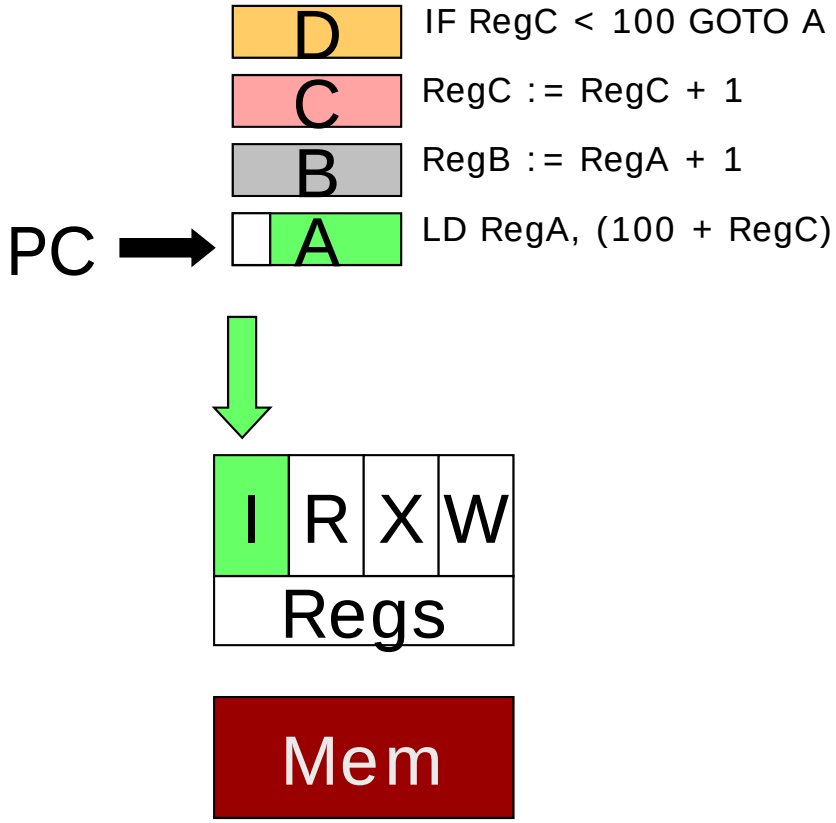
UPPSALA UNIVERSITET

Initially



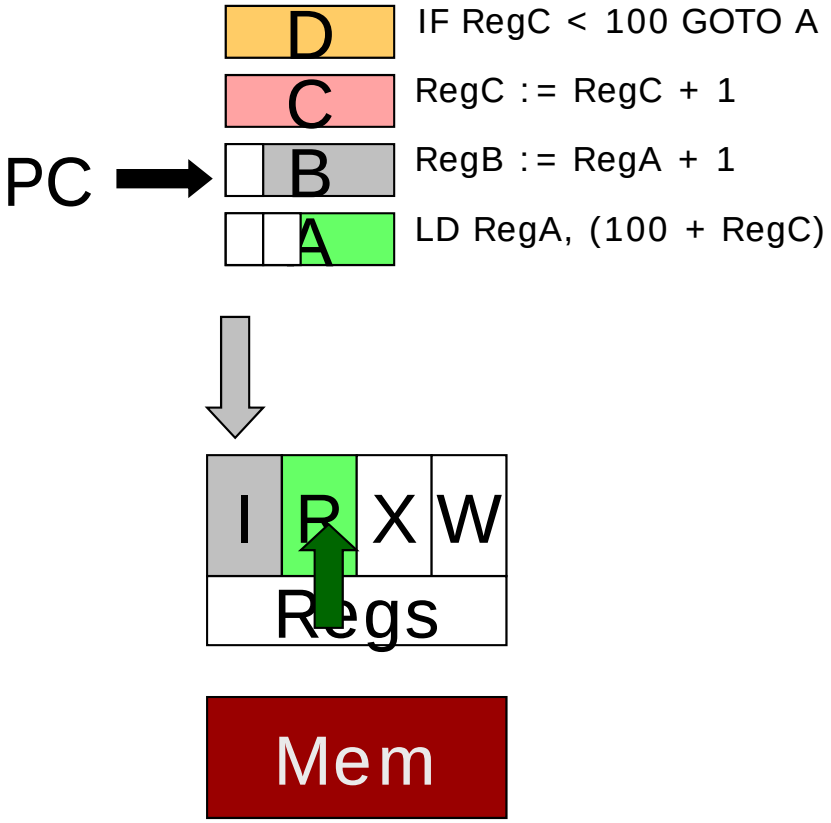
AVDARK 2013

Cycle 1

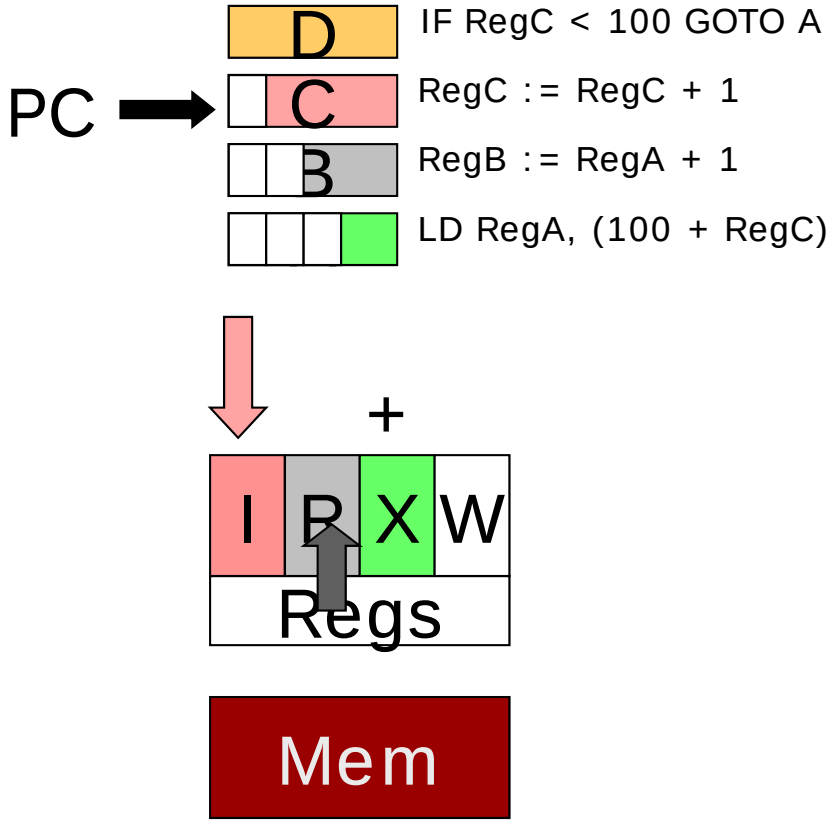


AVDARK 2013

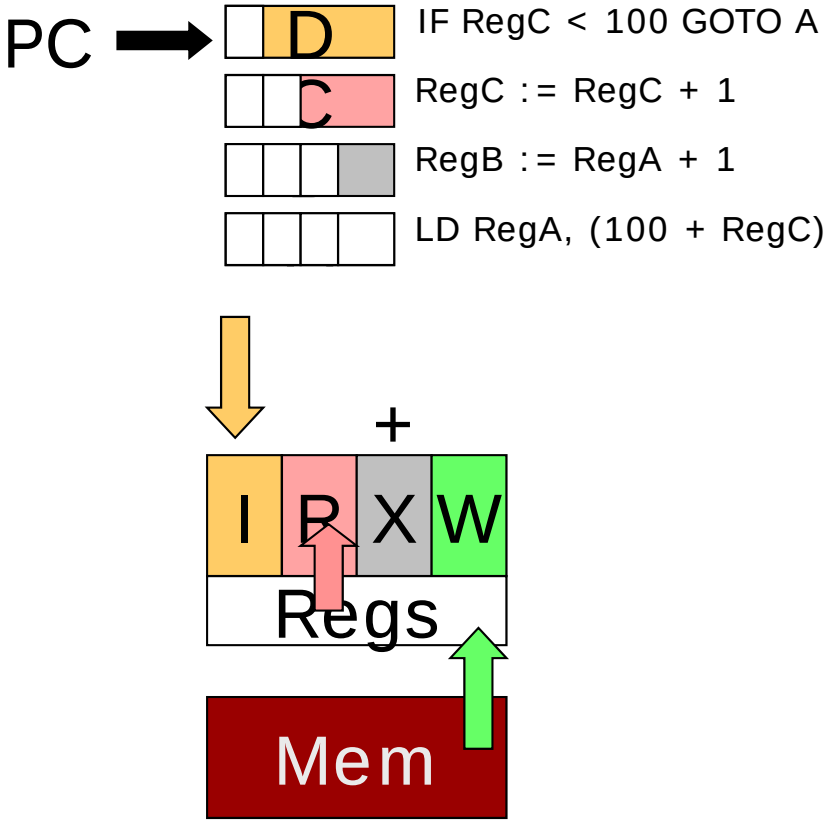
Cycle 2


D
IF RegC < 100 GOTO A
C
RegC := RegC + 1
PC
B
RegB := RegA + 1
A
LD RegA, (100 + RegC)
I
R
X
W
Regs
Mem

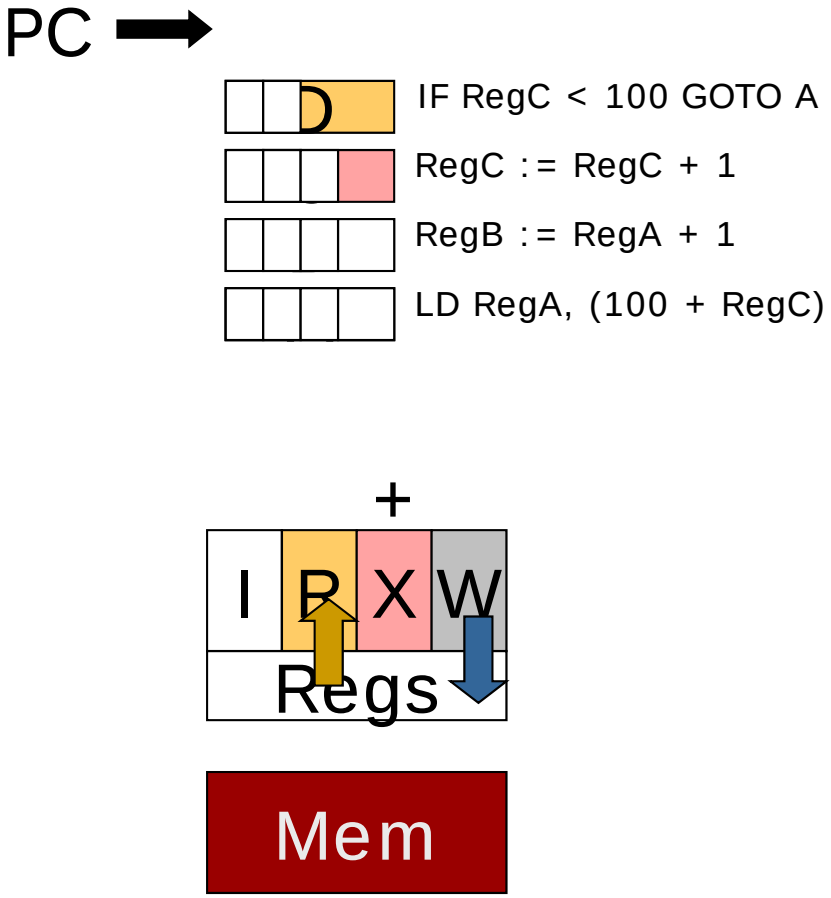
Cycle 3


D
IF RegC < 100 GOTO A
PC
C
RegC := RegC + 1
B
RegB := RegA + 1
LD RegA, (100 + RegC)
+
I
R
X
W
Regs
Mem

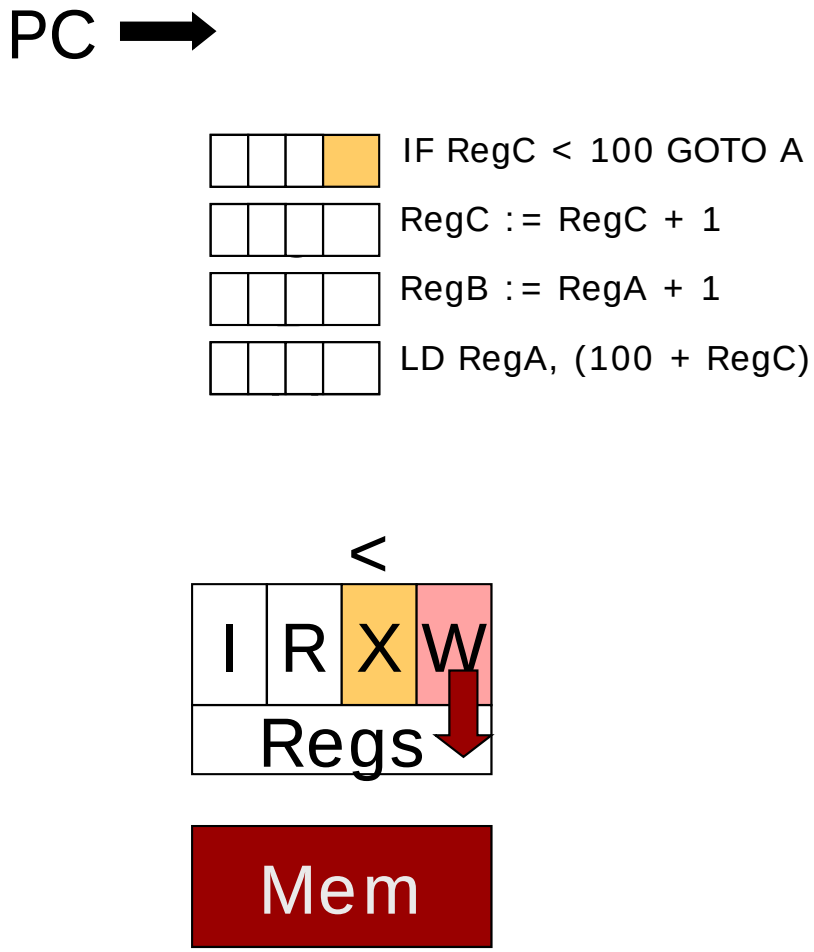
Cycle 4


PC
D
C
IF RegC < 100 GOTO A
RegC := RegC + 1
RegB := RegA + 1
LD RegA, (100 + RegC)
+
I
R
X
W
Regs
Mem

Cycle 5

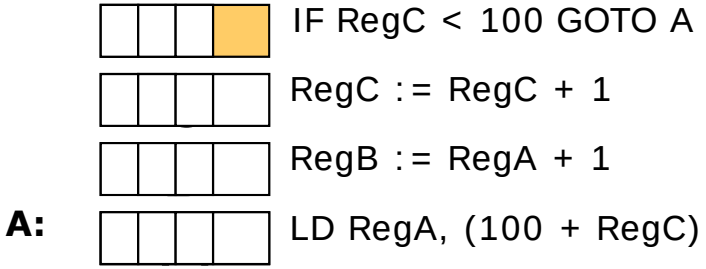

PC
D
IF RegC < 100 GOTO A
RegC := RegC + 1
RegB := RegA + 1
LD RegA, (100 + RegC)
+
I
R
X
W
Regs
Mem

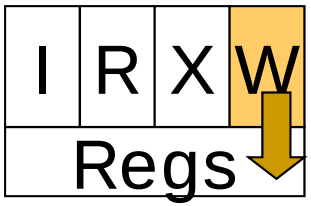
Cycle 6


PC
IF RegC < 100 GOTO A
RegC := RegC + 1
RegB := RegA + 1
LD RegA, (100 + RegC)
<
I
R
X
W
Regs
Mem

Cycle 7

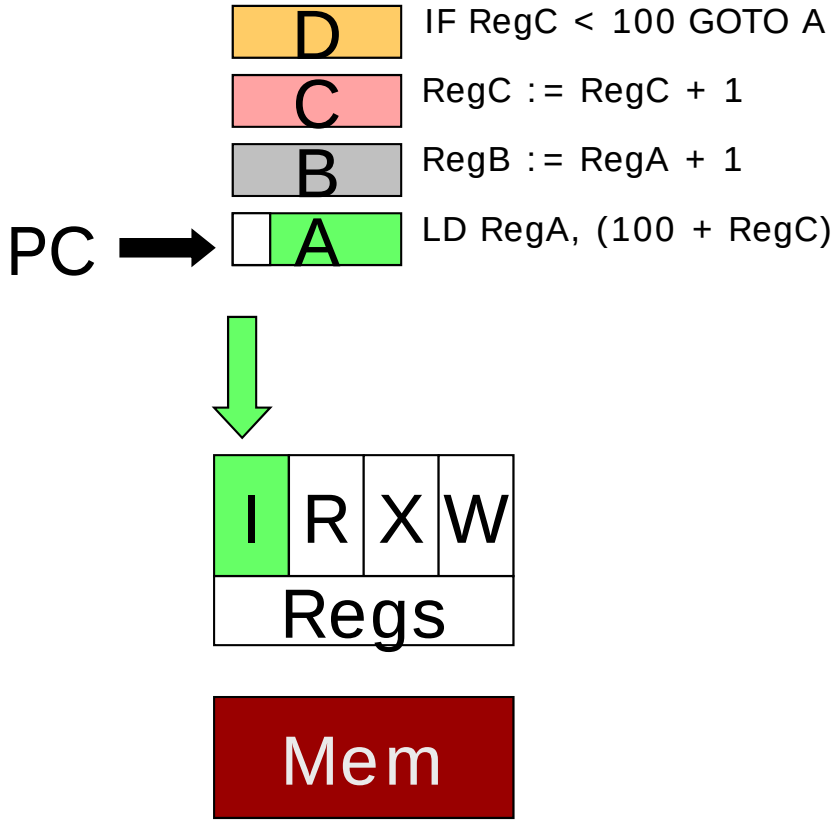
PC ➞





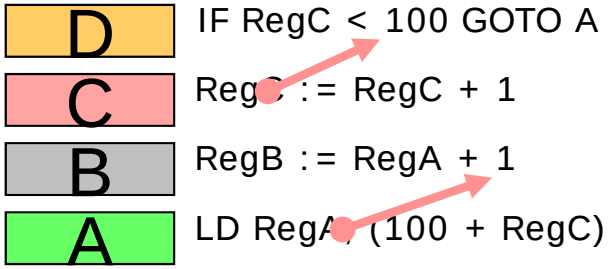
Branch: Addr(A) ➔ PC

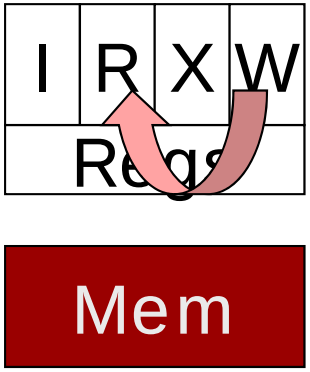
Cycle 8



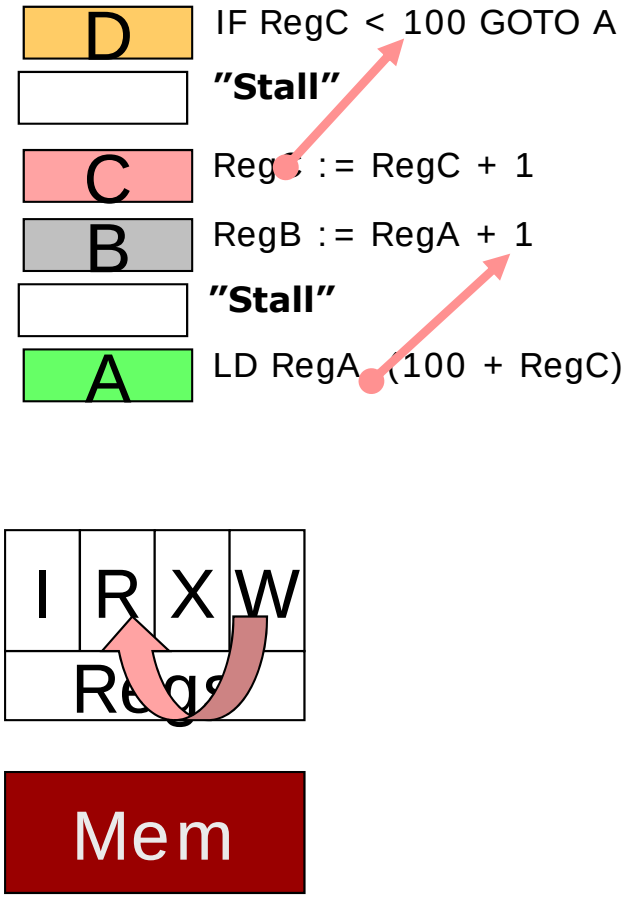
AVDARK 2013

Data dependency ☹
Previous execution example wrong!

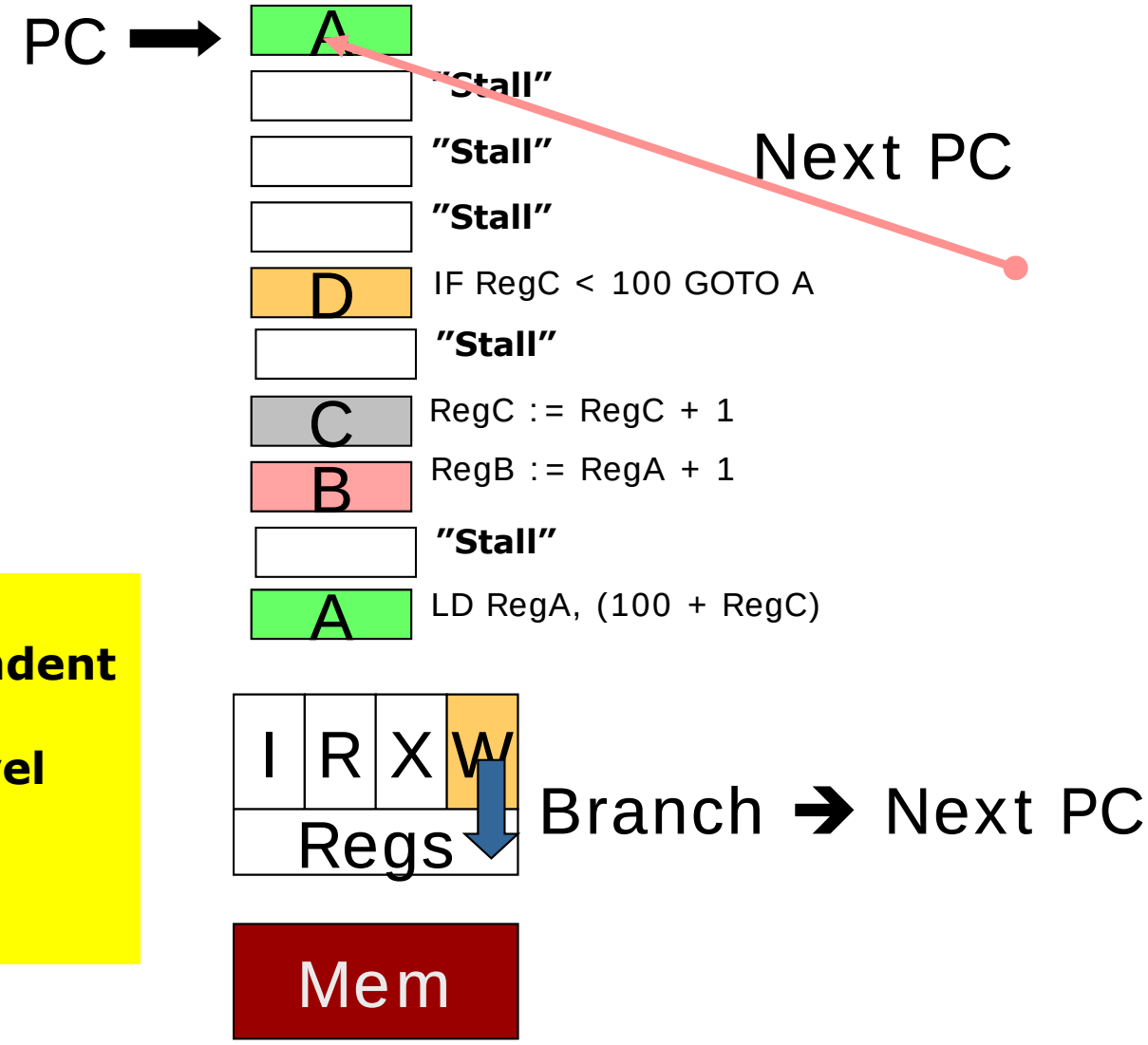




Data dependency fix 1: pipeline delays



Control dependence: Branch delays ☹



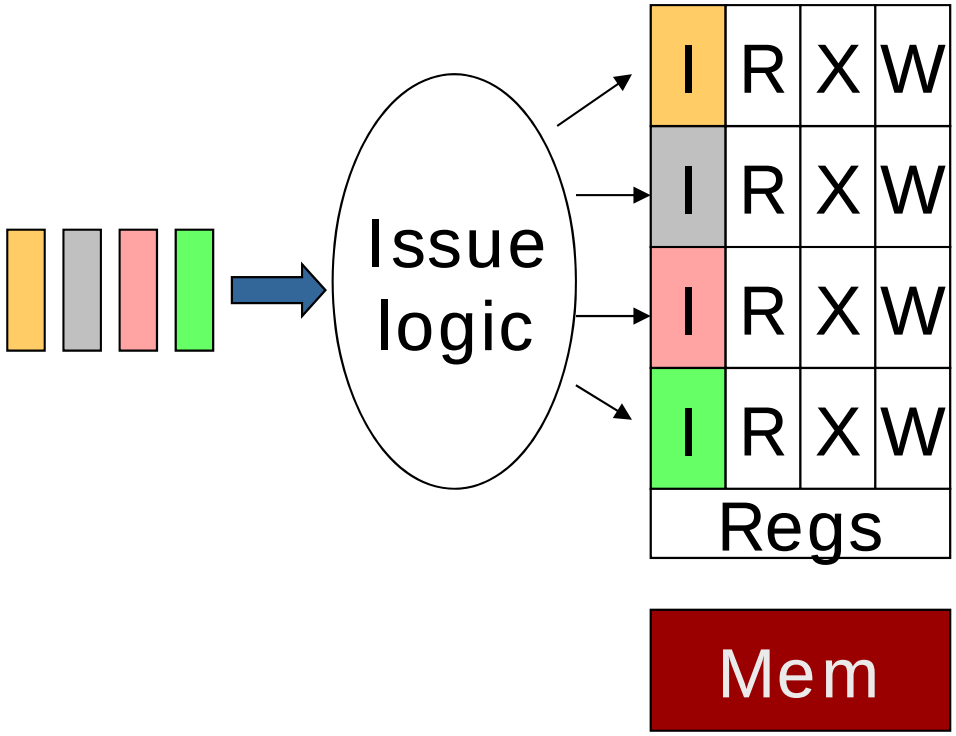
Need to find independent instructions (aka Instruction-Level Parallelism, ILP) to avoid stalls!

9 cycles per iteration of 4 instructions ☹

Need longer basic blocks with independent instr.

It is actually a lot worse!

Modern CPUs:"superscalars" with ~4 parallel pipelines

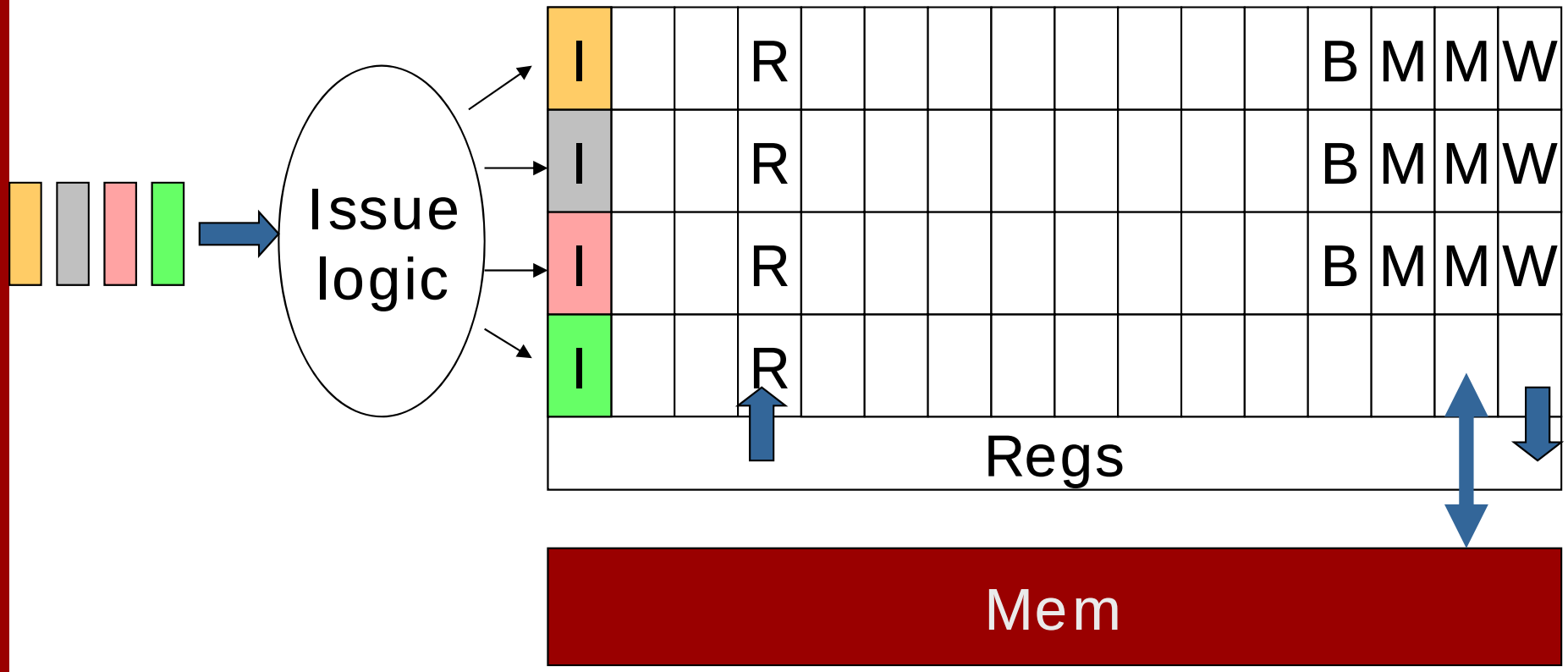


Need to find 4x more ILP

+Higher throughput
- More complicated architecture
- Harder to find "enough" independent instr.

It is actually a lot worse!

Modern CPUs: ~10-20 stages/pipe

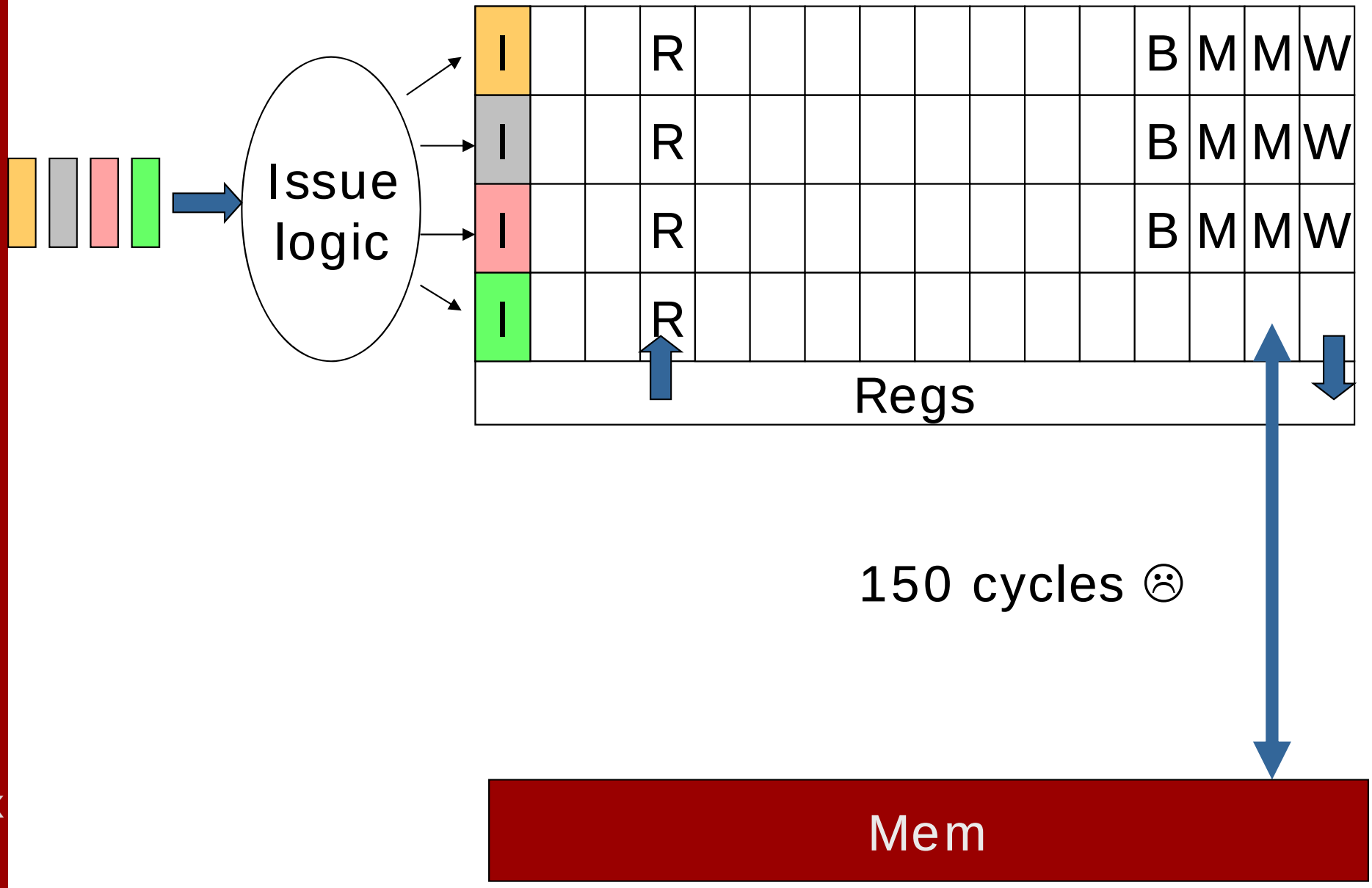


+Shorter cycletime (higher GHz)

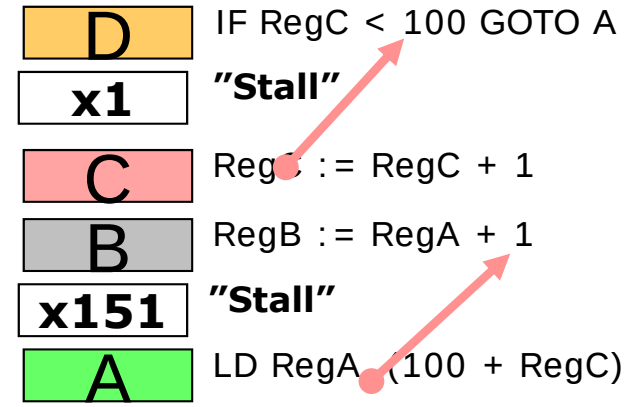
- Even harder to find ”enough” independent instr.

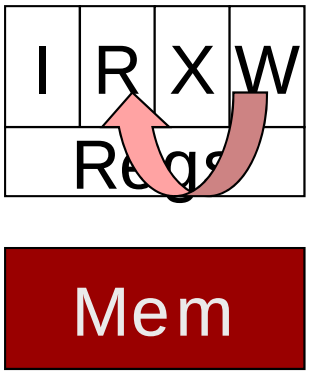
It is actually a lot worse!

DRAM access: ~150 CPU cycles

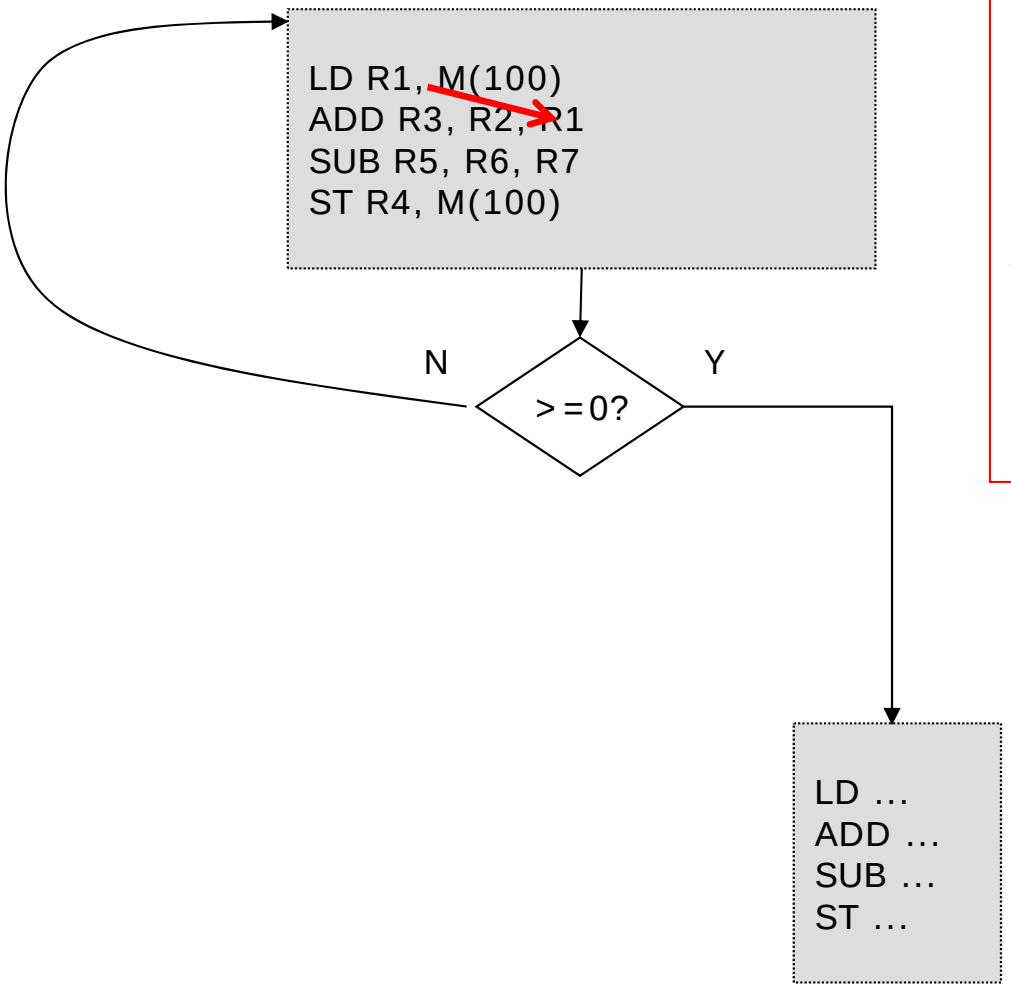


Data dependencies costs more!





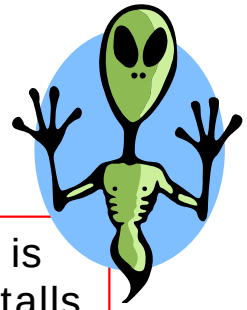
Fix 1: Out-of order execution:
Improving ILP

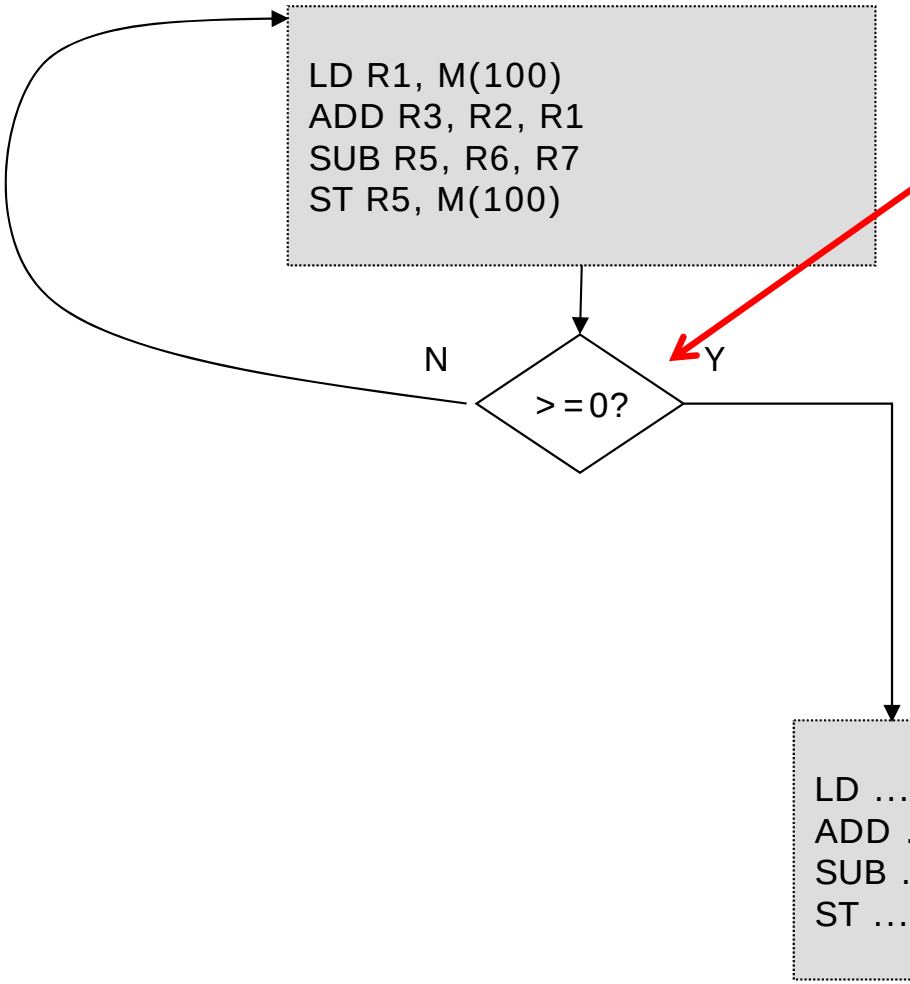


The HW may execute the instructions of a ”basis block” in a different order, but will make the ”side-effects” of the instructions appear in order.

Assume that LD takes a long time. The ADD is dependent on the LD ☹
Start the SUB and ST before the ADD
Update R5 and R4 and M(100) **after** R3

Fix 2: Branch prediction





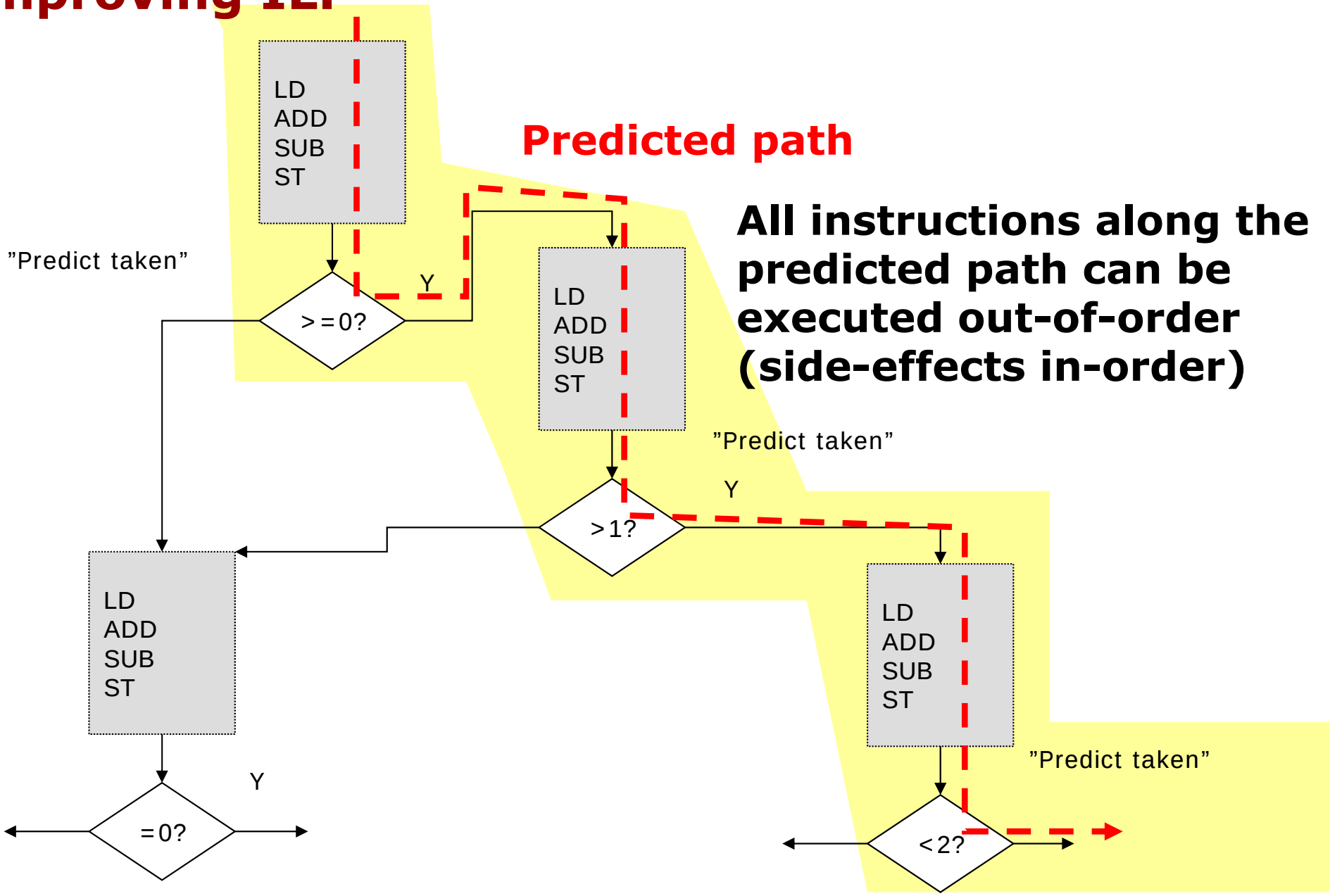
The HW can guess if the branch is taken or not and avoid branch stalls if the guess is correct.

Assume the guess is ”Y”.

The HW can start to execute these instruction **before** the outcome the the branch is known, but cannot allow any ”side-effect” to take place until the outcome is known

Fix 3: Scheduling Past Branches

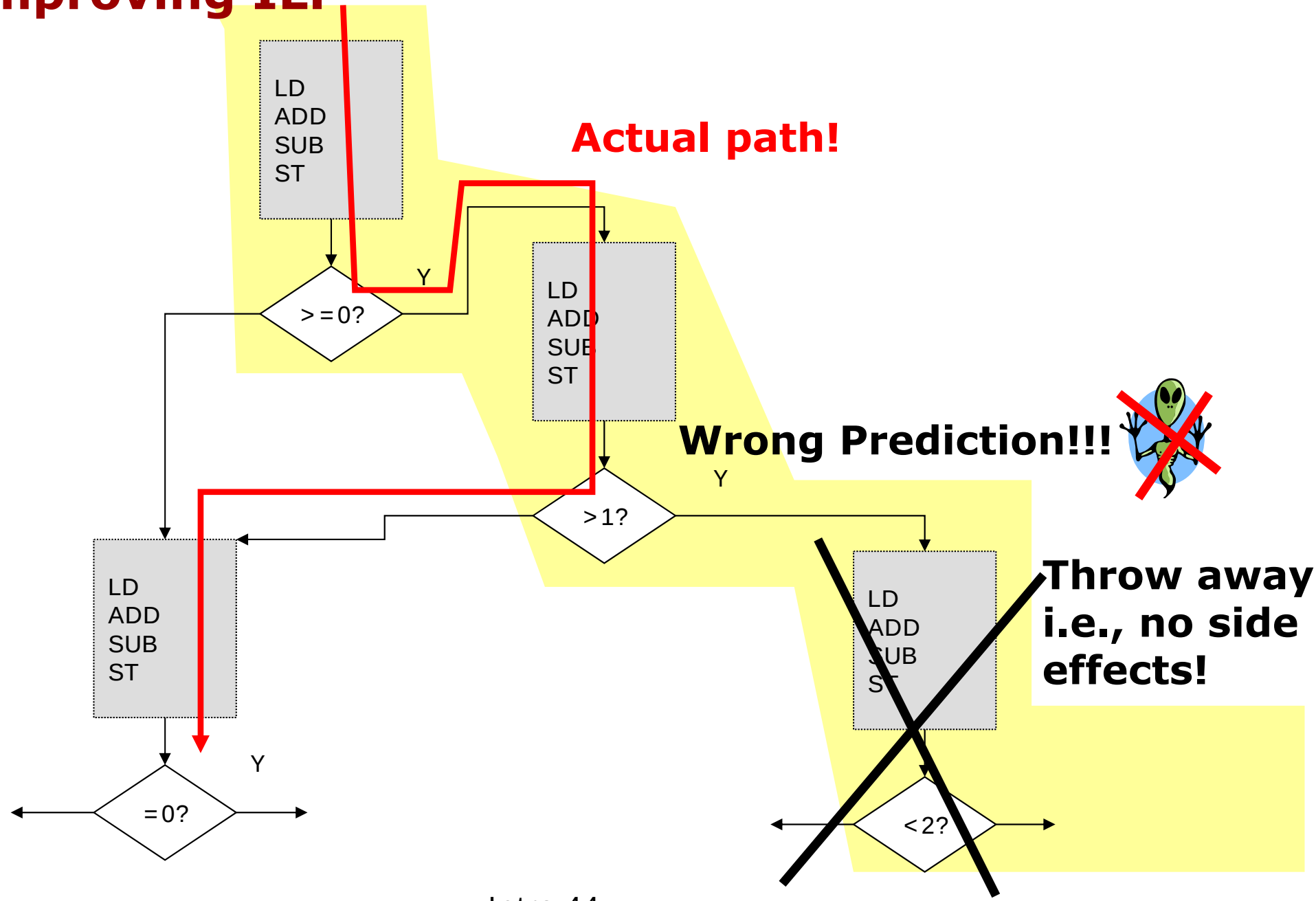
Improving ILP



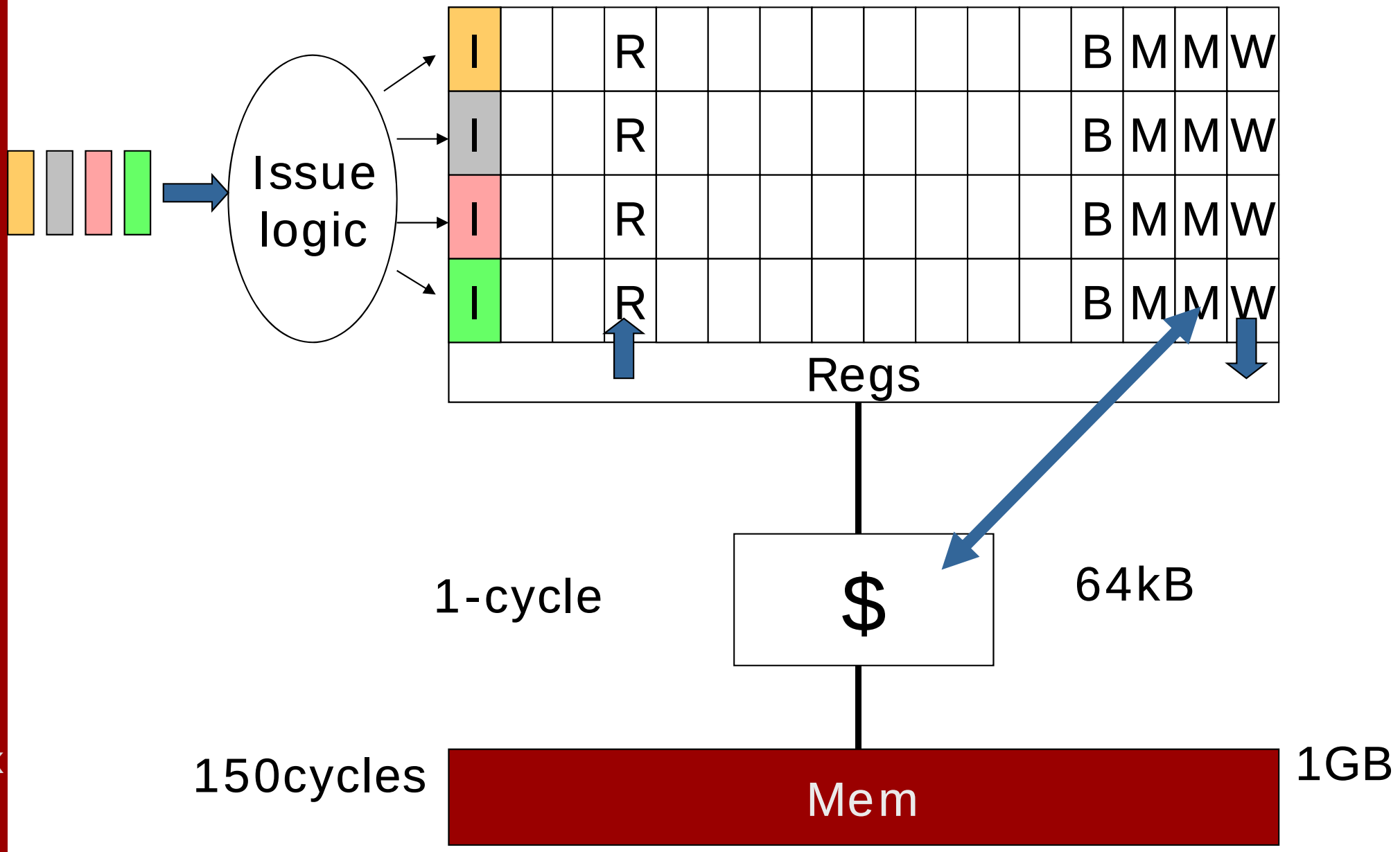
All instructions along the predicted path can be executed out-of-order (side-effects in-order)

Fix 3: Scheduling Past Branches

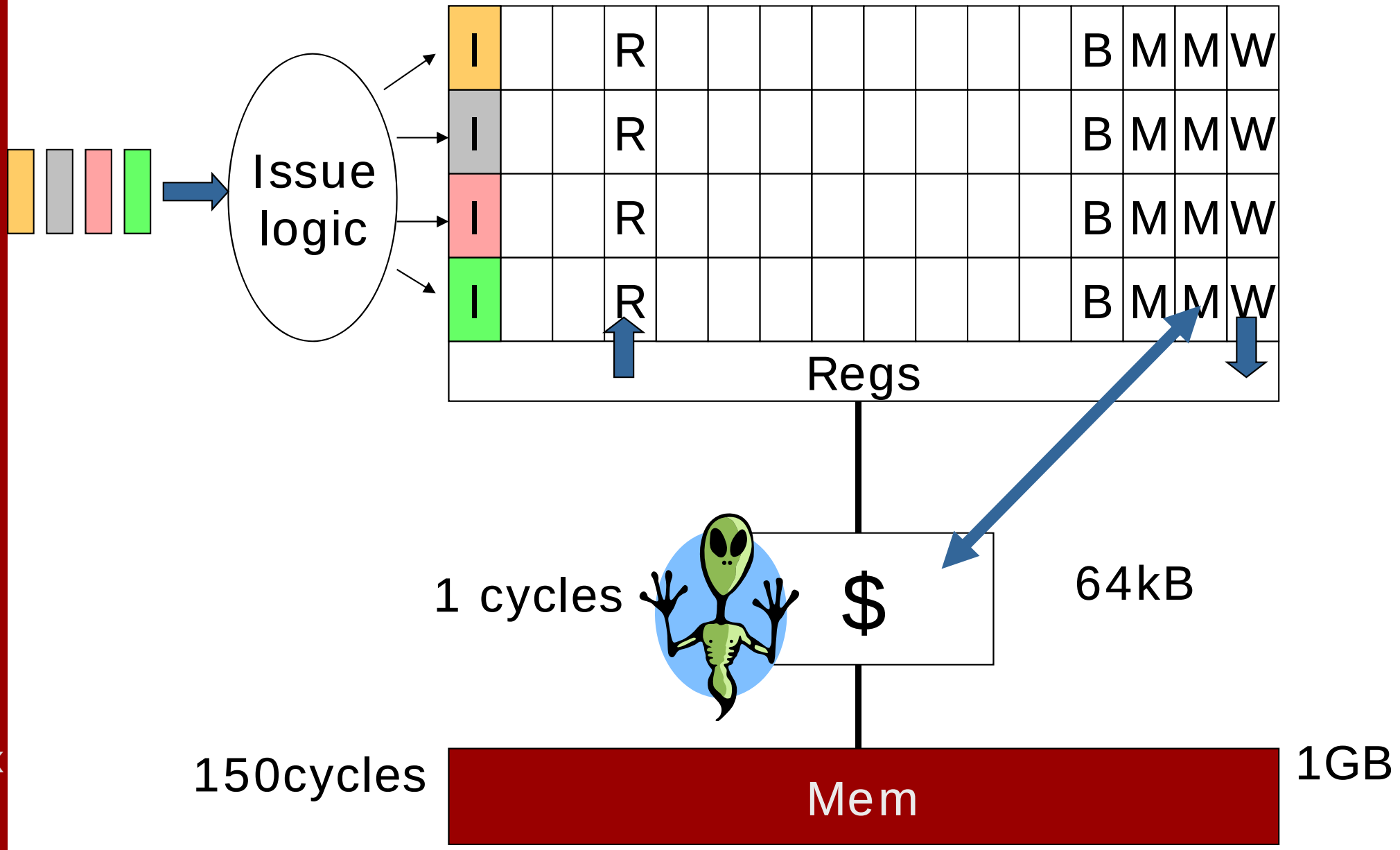
Improving ILP



Fix 4: Use a caches



Fix 5: Hardware prefetching



UPPSALA UNIVERSITET

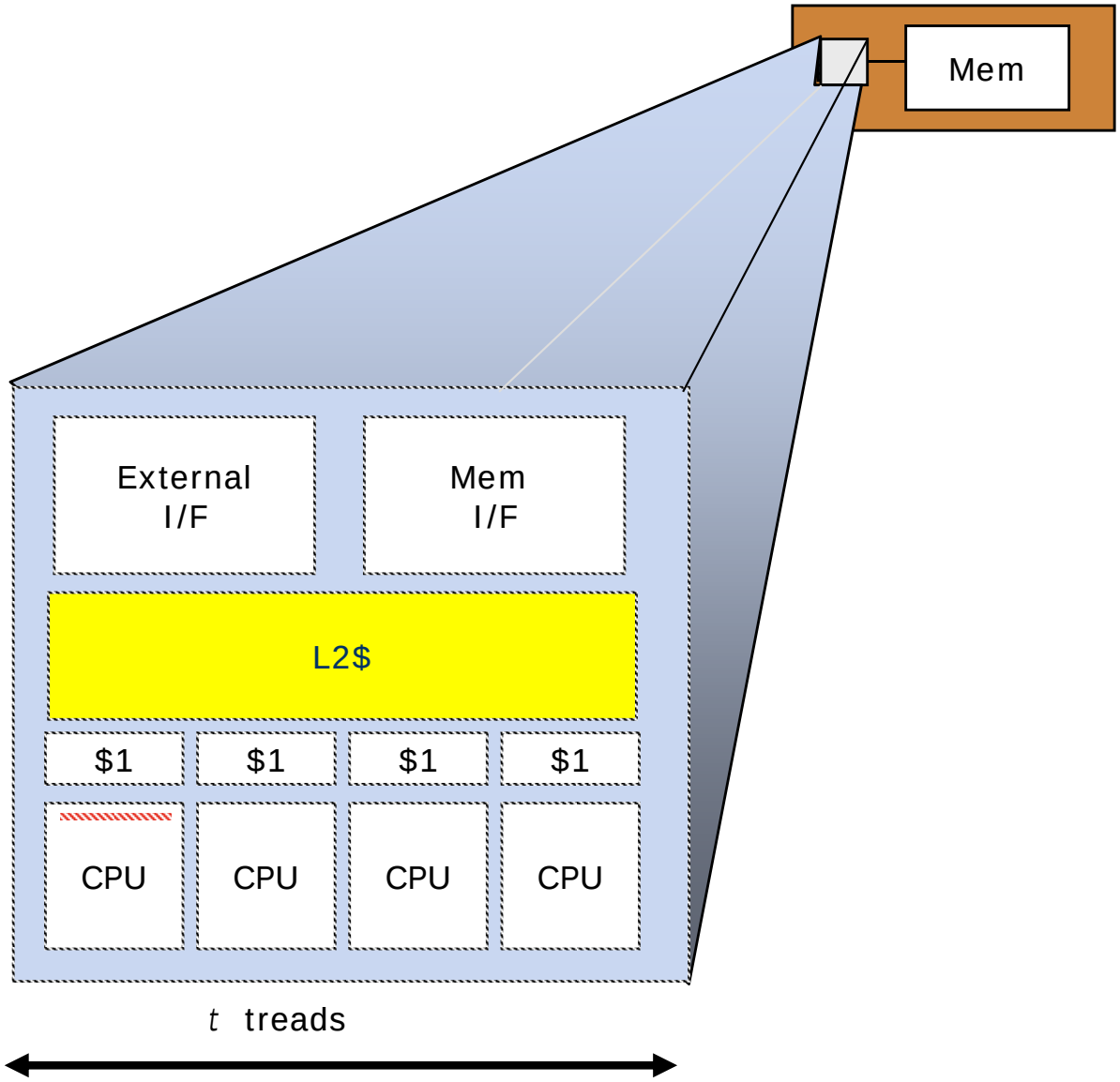
Woops, using too much power

- Running at 2x the frequency ➔ will use much more than 2x the power
- It is also really hard to find enough ILP
- Speculation results in a fair amount of wasted work

AVDARK 2013

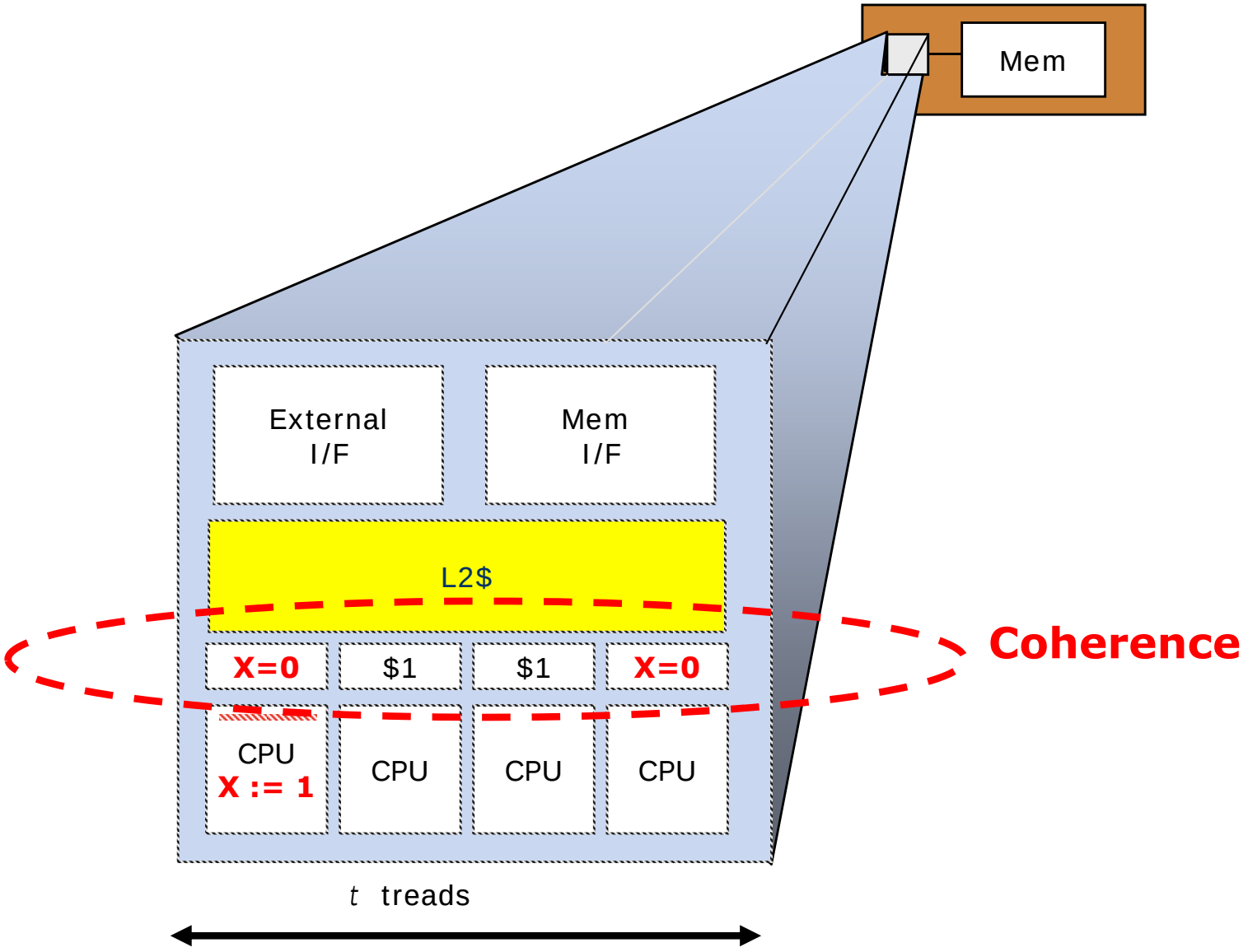
Fix6: Multicores

But now we also need to handle Thread-Level Parallelism (TLP)

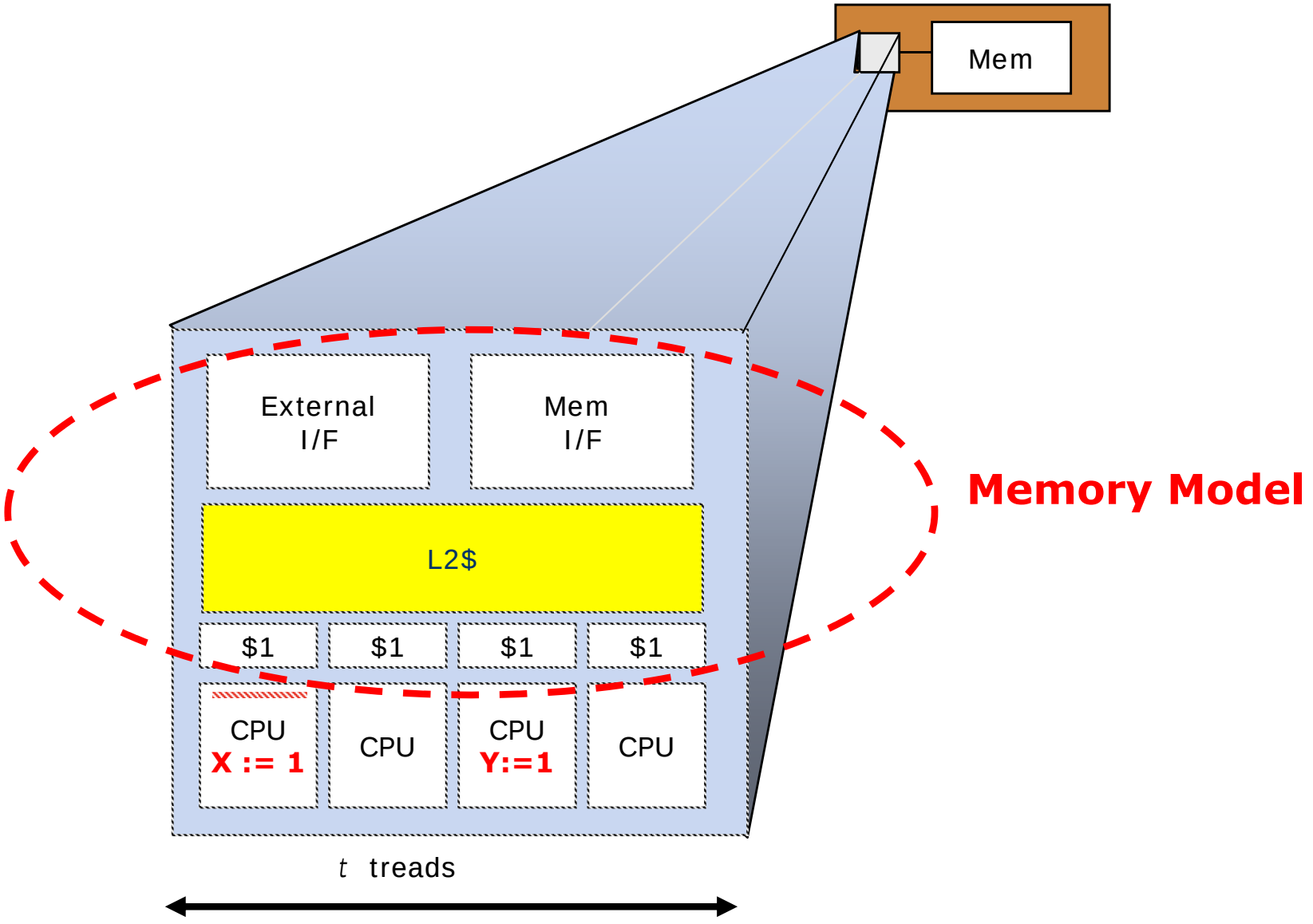


AVDARK 2013

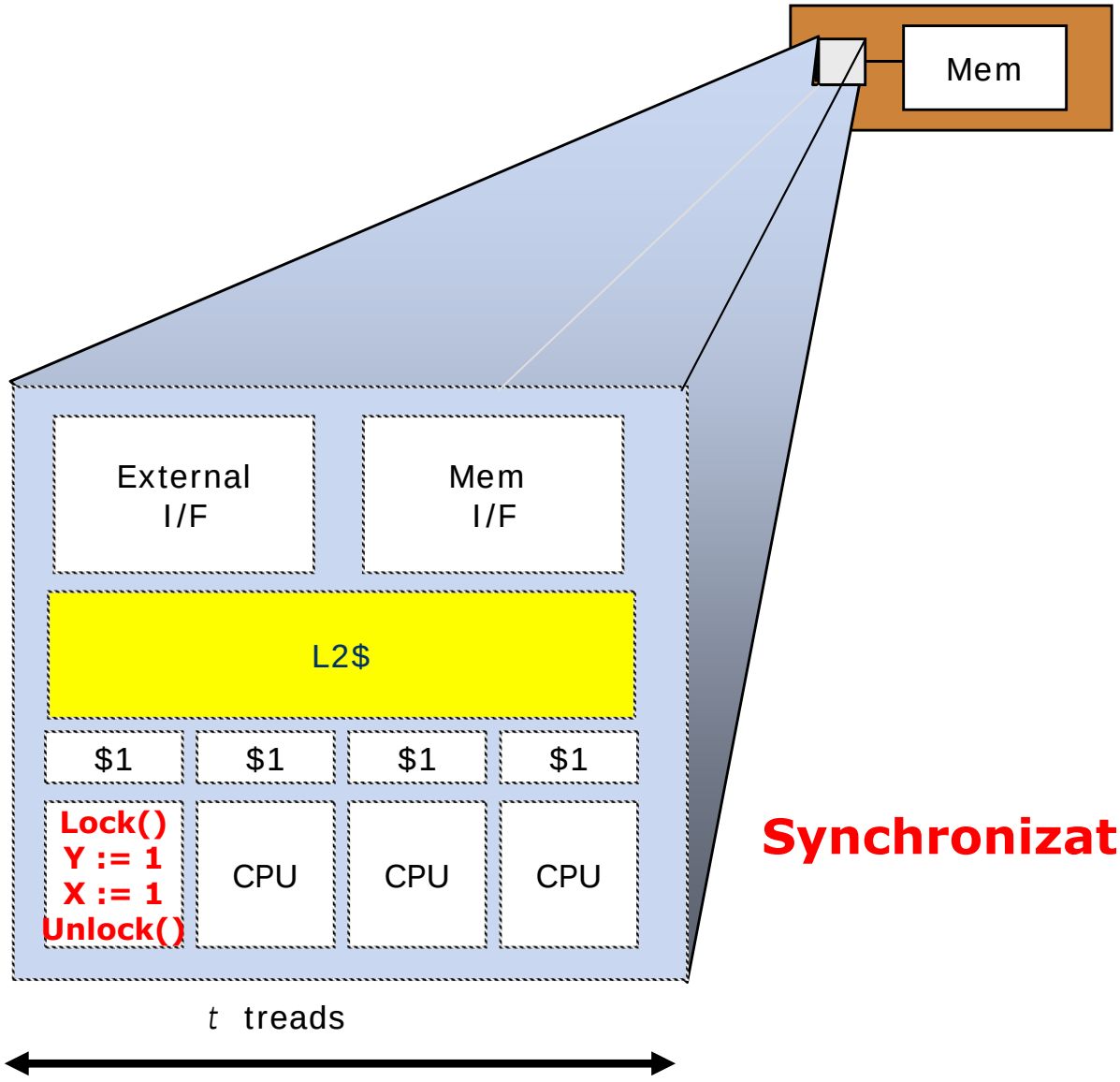
Fix7: Coherence



Fix8: Memory Models

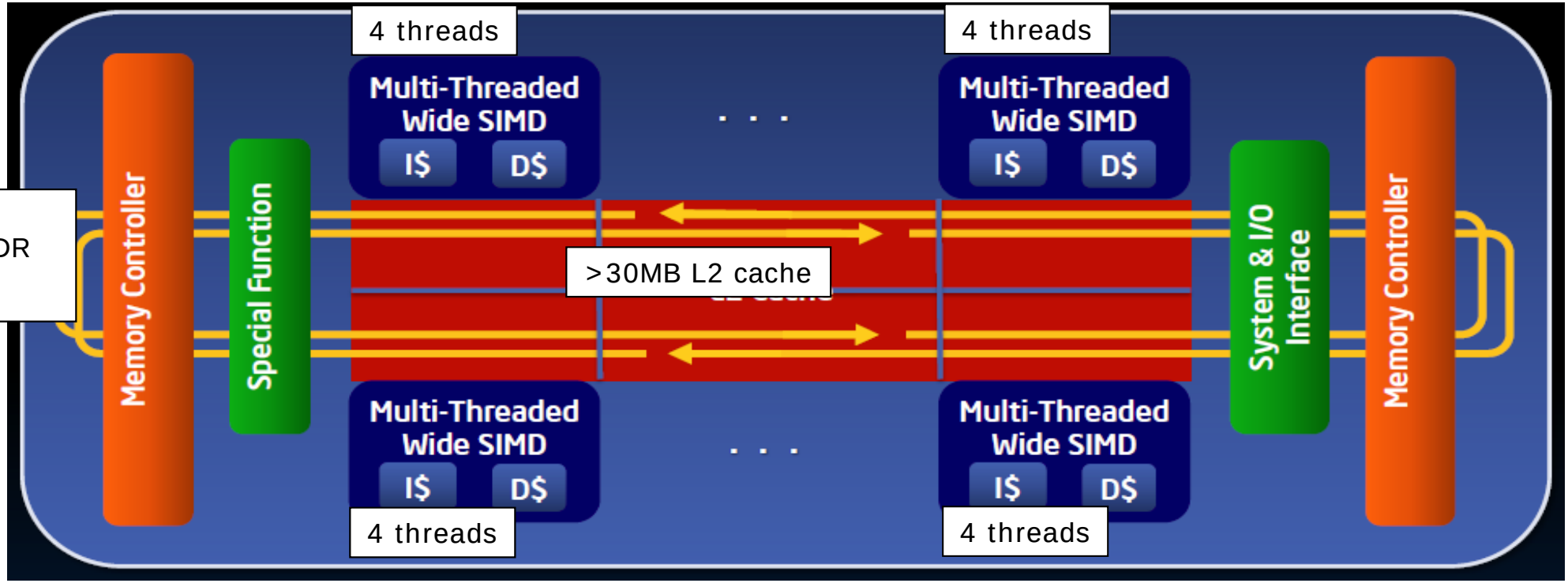


Fix9: Synchronization



Synchronization

Make it fast and scalable!



- Intel Xeon Phi (aka MIC or Knights Corner), 22nm
- >60 cores, X86 instructions (”enhanced”)
- Each core executes 4 threads ”in parallel”
- Each instruction can perform “16 multiplications” in parallel

UPPSALA UNIVERSITET

How to get efficient architectures…

- Increase clock frequency
- Create and explore locality:
  a) Spatial locality
  b) Temporal locality
  c) Geographical locality
- Create and explore parallelism
  a) Instruction level parallelism (ILP)
  b) Thread level parallelism (TLP)
  c) Memory level parallelism (MLP)
- Speculative execution
  a) Out-of-order execution
  b) Branch prediction
  c) Prefetching