

Relaxed Consistency Models



NALABOLU SAN...



KOUSIK RAJESH



YOGESH KUMAR



ASWATHY N S



SWATI UPADHYAY



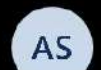
VEDIKA JITENDR...



IMLIJUNGLA LON...



K CHITRA



ADITYA KUMAR S...



TANVISH



SARASWATULA P...



Hemangee Kalpe...

Relaxed memory consistency models

- For directory protocols, misses have long latency and collecting acknowledgements can take even longer
- To preserve SC we need to restrict many performance optimisations done by modern compilers and processors
- High cost of memory access motivates overlapped reads/writes or reordering
- Therefore allow reorder up to some extent still preserving semantics
=> Relaxed consistency models



NALABOLU SAN...



KOUSIK RAJESH



YOGESH KUMAR



ASWATHY N S



SWATI UPADHYAY



VEDIKA JITENDR...



IMLIJUNGLA LON...



K CHITRA



ADITYA KUMAR S...



TANVISH



SARASWATULA P...



KUSHAL SANGW...




KARTIKEYA SAXE...



Hemangee Kalpe...

Characterising different memory-consistency models

- Based on two key characteristics:
- (1) how they relax the program order requirement (i, ii, iii), AND
- (2) how they relax the write atomicity requirement (iv)
- (i) Relax $W \rightarrow R$
- (ii) Relax $W \rightarrow W$ 
- (iii) Relax $R \rightarrow R, R \rightarrow W$
- (iv) Read others write early [i.e. before many others have seen inv or update message]
- (v) Relax program order as well as write atomicity = Read own write early [our own write is read by self before other sharers are inv or updated. e.g. Read from write-buffer, or wr-thru-caches (read from cache before write is complete to next level)
 - In a cache based system this allows the read to return the value of the write before the write is serialised with respect to other writes to the same location and before the inv/updates of the write reach any other processor



Program order:

Applies to
Different locations

NS

NALABOLU SAN...



KOUSIK RAJESH

YK

YOGESH KUMAR

AS

ASWATHY N S

SU

SWATI UPADHYAY

VK

VEDIKA JITENDR...



IMLIJUNGLA LON...



K CHITRA

AS

ADITYA KUMAR S...



TANVISH

SP

SARASWATULA P...

KS

KUSHAL SANGW...



KARTIKEYA SAXE...



Hemangee Kalpe...

Relaxations?

- Programmer can over-ride relaxations
 - By using explicit fence instruction
 - Such mechanisms are called safety-nets for a model
 - Discuss Figures in next 2 slides
- Assumptions during relaxation
 - (1) A write is eventually made visible to all processors and writes to same location are serialised
 - These are met easily if no-cache in system or by cache-coherence protocol in presence of shared data caching
 - (2) All models enforce uni-processor data and control dependencies



Relaxation	$W \rightarrow R$ Order	$W \rightarrow W$ Order	$R \rightarrow RW$ Order	Read Others' Write Early	Read Own Write Early	Safety net
SC [16]					✓	
IBM 370 [14]	✓					serialization instructions
TSO [20]	✓				✓	RMW
PC [13, 12]	✓			✓	✓	RMW
PSO [20]	✓	✓			✓	RMW, STBAR
WO [5]	✓	✓	✓		✓	synchronization
RCsc [13, 12]	✓	✓	✓		✓	release, acquire, nsync, RMW
RCpc [13, 12]	✓	✓	✓	✓	✓	release, acquire, nsync, RMW
Alpha [19]	✓	✓	✓		✓	MB, WMB
RMO [21]	✓	✓	✓		✓	various MEMBAR's
PowerPC [17, 4]	✓	✓	✓	✓	✓	SYNC

Figure 8: Simple categorization of relaxed models. A ✓ indicates that the corresponding relaxation is allowed by straightforward implementations of the corresponding model. It also indicates that the relaxation can be detected by the programmer (by affecting the results of the program) except for the following cases. The “Read Own Write Early” relaxation is not detectable with the SC, WO, Alpha, and PowerPC models. The “Read Others’ Write Early” relaxation is possible and detectable with complex implementations of RCsc.



Relaxation	Example Commercial Systems Providing the Relaxation
$W \rightarrow R$ Order	AlphaServer 8200/8400, Cray T3D, Sequent Balance, SparcCenter1000/2000
$W \rightarrow W$ Order	AlphaServer 8200/8400, Cray T3D
$R \rightarrow RW$ Order	AlphaServer 8200/8400, Cray T3D
Read Others' Write Early	Cray T3D
Read Own Write Early	AlphaServer 8200/8400, Cray T3D, SparcCenter1000/2000

Figure 9: Some commercial systems that relax sequential consistency.

0



Relax W- \rightarrow R

- Relaxing the Write to Read program order
- Motivation: Allow hardware to hide latency of write operation
- While write-miss is still pending, the processor can issue and complete reads that hit in the cache or even a single read that misses in its cache
 - i.e. Read can be reordered wrt previous write from the same processor
- Consequence
 - Ex-1: will violate SC results. Will fail to provide SC in ex-1
 - Result correct for Ex-2



NALABOLU SAN...



KOUSIK RAJESH



YOGESH KUMAR



ASWATHYN S



SWATI UPADHYAY



VEDIKA JITENDR...



K CHITRA



ADITYA KUMAR S...



TANVISH



SARASWATULA P...



KUSHAL SANGW...



KARTIKEYA SAXE...



IMLIUNGLA LON...



ABHAY PRATAP G...



ANSHUL MITTAL



Hemangee Kalpe...

EX-1

P1	P2
flag1 = 1;	flag2 = 1;
if(flag2 == 0)	if(flag1 == 0)
// CS	// CS

EX-2

P1	P2
Data = 2000;	while(Head == 0);
Head = 1;	read Data;

EX-5

P1	P2	P3
A = 1;	while (A == 0);	
	B = 1;	while (B == 0);
		print A;

EX-3

P1	P2
A = 1;	while(flag == 0);
flag = 1;	print A;

EX-6

P1	P2
(i) A = 1 ;	(iii) B = 1;
(ii) print B ;	(iv) print A;

NS

NALABOLU SAN...

KOUSIK RAJESH

YK

YOGESH KUMAR

AS

ASWATHY N S

SU

SWATI UPADHYAY

VK

VEDIKA JITENDR...

K CHITRA

K CHITRA

AS

ADITYA KUMAR S...

TANVISH

TANVISH

SP

SARASWATULA P...

KS

KUSHAL SANGW...

KARTIKEYA SAXE...

IMLIJUNGLA LON...

IMLIJUNGLA LON...

ABHAY PRATAP G...

ABHAY PRATAP G...

AM

ANSHUL MITTAL

Hemangee Kalpe...

Hemangee Kalpe...

Relax W- \rightarrow R

- Relaxing the Write to Read program order
- Motivation: Allow hardware to hide latency of write operation
- While write-miss is still pending, the processor can issue and complete reads that hit in the cache or even a single read that misses in its cache
 - i.e. Read can be reordered wrt previous write from the same processor
- Consequence
 - Ex-1: will violate SC results. Will fail to provide SC in ex-1
 - Result correct for Ex-2

NS

NALABOLU SAN...

KOUSIK RAJESH

KOUSIK RAJESH

YK

YOGESH KUMAR

AS

ASWATHY N S

SU

SWATI UPADHYAY

VK

VEDIKA JITENDR...

K CHITRA

K CHITRA

AS

ADITYA KUMAR S...

TANVISH

TANVISH

SP

SARASWATULA P...

KS

KUSHAL SANGW...

KARTIKEYA SAXE...

KARTIKEYA SAXE...

IMLIJUNGLA LON...

IMLIJUNGLA LON...

ABHAY PRATAP G...

ABHAY PRATAP G...

AM

ANSHUL MITTAL

Hemangee Kalpe...

Hemangee Kalpe...

01-Nov-2021 - Google

jamboard.google.com/d/1Bdp_J-09t0rgXuhYJEtZxtLtJhsVBFsb3mwrB1E1R4/viewer?f=0

01-Nov-2021

1/2

Share

Set background Clear frame

ex-1

P1

Flag1 = 1

if (Flag2 == 0)

// CS

P2

Flag2 = 1

if (Flag1 == 0)

// CS

94% (3:55) Mon, November 1, 15:11

NS
NALABOLU SAN... KOUSIK RAJESH

YK
YOGESH KUMAR AS
ASWATHY N S

SU
SWATI UPADHYAY VK
VEDIKA JITENDR...

AS
K CHITRA ADITYA KUMAR S...

SP
TANVISH SARASWATULA P...

KS
KUSHAL SANGW... KARTIKEYA SAXE...

IMLIJUNGLA LON... ABHAY PRATAP G...

AM
ANSHUL MIITAL NARESH BHARAS...

Hemangee Kalpe...

01-Nov-2021 - Google | 28_Oct - Mentimeter | +

jamboard.google.com/d/1Bdp_J-09t0rgXuhYJEtZxtLtJhsVBFsb3mwrB1E1R4/viewer?F=0

01-Nov-2021

1/2

Set background Clear frame

SC *

P1

W
↓
R

Flag1 = 1

if (Flag2 == 0) ✓

// CS

P2

W → R

Flag2 = 1

if (Flag1 == 0)

// CS

92% (3:28) Mon, November 1, 15:14

NS
NALABOLU SAN... KOUSIK RAJESH

YK
YOGESH KUMAR AS
ASWATHY N S

SU
SWATI UPADHYAY VK
VEDIKA JITENDR...

AS
K CHITRA ADITYA KUMAR S...

SP
TANVISH SARASWATULA P...

KS
KUSHAL SANGW... KARTIKEYA SAXE...

AM
IMLIJUNGLA LON... ANSHUL MITTAL

NARESH BHARAS... K CHITRA

Hemangee Kalpe...

01-Nov-2021 - Google | 28_Oct - Mentimeter | +

jamboard.google.com/d/1Bdp_J-09t0rgXuhYJEtZxtLJhsVBFsb3mwrB1E1R4/viewer?F=1

01-Nov-2021

2/2

Set background Clear frame

ex-2

SC → W → R ✓

P1

Data = 2000

Head = 1

P2

while(Head == 0);

Read Data

ex-3

SC = W → R

P1

A = 1

Flag = 1

P2

print B

print A

NS

NALABOLU SAN...

KOUSIK RAJESH

YK

YOGESH KUMAR

AS

ASWATHY N S

SU

SWATI UPADHYAY

VK

VEDIKA JITENDR...

AS

K CHITRA

ADITYA KUMAR S...

SP

TANVISH

SARASWATULA P...

KS

KUSHAL SANGW...

KARTIKEYA SAXE...

AM

IMLIJUNGLA LON...

ANSHUL MITTAL

NARESH BHARAS...

K CHITRA

VA

VARHADE AMEY...

Hemangee Kalpe...

Relax W- \rightarrow R

- There are 3 models under this:
- **IBM 370: strictest**
 - Prohibits read from returning value of write until write is made visible to all processors
 - Even on same processor if write is pending (i.e. others not seen) then read to same address has to wait
- **SPARC TSO (Total Store Order)**
 - Partially relaxes above one by allowing processor to read its own write before it is seen by all and even before it is serialised wrt other writes to same location
 - BUT read is not allowed to return another processors write until all have seen it
- **PC: Relaxes both constraints (Processor Consistency)**
 - Read can return the value of any write before write is serialised or made visible to all



01-Nov-2021 - Google | 28_Oct - Mentimeter | jamboard.google.com/d/1Bdp_J-09t0rgXuhYJEtTzxtLjhsVBFsb3mwrB1E1R4/viewer?F=1

01-Nov-2021

Set background Clear frame

2/2

Share

ex-2

SC → W → R ✓

P1

W Data = 2000
↓
W Head = 1

P2

while(Head == 0);
Read Data

ex-3

SC = W → R ✓

P1

W A = 1
↓
W Flag = 1

P2

print B
print A

NS
NALABOLU SAN... KOUSIK RAJESH

YK
YOGESH KUMAR AS
ASWATHY N S

SU
SWATI UPADHYAY VK
VEDIKA JITENDR...

K CHITRA
ADITYA KUMAR S...

AS
SP
SARASWATULA P...

KS
KUSHAL SANGW... KARTIKEYA SAXE...

AM
IMLIJUNGLA LON... ANSHUL MITTAL

NARESH BHARAS... K CHITRA

VA
VARHADE AMEY... Hemangee Kalpe...

01-Nov-2021 - Google | 28_Oct - Mentimeter | jamboard.google.com/d/1Bdp_J-09t0rgXuhYJEtTzxtLtJhsVBFsb3mwrB1E1R4/viewer?F=2

01-Nov-2021

3/3

Set background Clear frame

ex-4

SC → W → R ✓

W
↓
W

P1
A = 1
B = 1

P2
print B
print A

ex-5

IBM ✓
TSO ✓
PC ✗

P1
A = 1

P2
while (A == 0);
B = 1

P3
while (B == 0);
print A

① → ② → ③ → ④

NS
NALABOLU SAN... KOUSIK RAJESH

YK
YOGESH KUMAR AS
ASWATHY N S

SU
SWATI UPADHYAY VK
VEDIKA JITENDR...

AS
K CHITRA ADITYA KUMAR S...

SP
TANVISH SARASWATULA P...

KS
KUSHAL SANGW... KARTIKEYA SAXE...

AM
IMLIJUNGLA LON... ANSHUL MITTAL

VA
NARESH BHARAS... K CHITRA

VA
VARHADE AMEY... Hemangee Kalpe...

Relaxations

- For Ex-3 and Ex-4
 - TSO = PC = SC
 - P1 has wr->wr. Therefore not reordered
- Ex-5
 - PC violates SC semantics
 - TSO satisfies SC
 - PC violates because it does not guarantee wr-atomicity
 - i.e. P2 reads wr-A of P1 early: before P3 sees new-A
 - Therefore P2 reads new-A and changes B and P3 sees new-B and reads old-A => violates SC



01-Nov-2021 - Google Jamboard

28_Oct - Mentimeter

jamboard.google.com/d/1Bdp_J-09t0rgXuhYJEtZxtLtJhsVBFsb3mwrB1E1R4/viewer?F=3

01-Nov-2021

4/4

Set background Clear frame

Share

ex-6

P1
1a A=1
1b print B

P2
B=1 2a
print A 2b

$B=0 \Rightarrow 1b \rightarrow 2a \text{ (1)}$
 $P0 \Rightarrow 1a \rightarrow 1b \text{ (2)} ; 2a \rightarrow 2b \text{ (3)}$

$1b \rightarrow 2a \rightarrow 2b \rightarrow 1b \quad \langle A, B \rangle = \langle 0, 0 \rangle$

$1a \rightarrow 1b \rightarrow 2a \rightarrow 2b$
 $B = "0"$ $A = "1"$

SC
not-SC

83% (2:19) Mon, November 1, 15:34

TANYISH
NALABOLU SAN...

YK
YOGESH KUMAR

AS
ASWATHY N S

SWATI UPADHYAY

AS
ADITYA KUMAR S...

SP
SARASWATULA P...

KS
KUSHAL SANGW...

KARTIKEYA SAXE...

IMIJUNGLA LON...

AM
ANSHUL MITTAL

NARESH BHARAS...

VA
VARHADE AMEY...

Hemangee Kalpe...

Relaxations

- EX-6: PC and TSO violate SC
- In EX-6 no interleaving in SC can print both A and B as '0'.
- Proof: program orders (i) \rightarrow (ii) and (iii) \rightarrow (iv)
- If $B = 0 \Rightarrow$ (ii) \rightarrow (iii) \Rightarrow (i) \rightarrow (iv)
- But (i) \rightarrow (iv) $\Rightarrow A = 1$
- If $A = 0 \Rightarrow \dots B=1$
- This is popular mutual-exclusion algorithm called Dekker's Algo.
- TSO and PC violate SC because they allow read to complete before write i.e. both read $A \ \& \ B = 0$ then do write $A \ \& \ B$.



TANYISH



NALABOLU SAN...



KOUSIK RAJESH



YOGESH KUMAR



ASWATHY N S



SWATI UPADHYAY



K CHITRA



ADITYA KUMAR S...



SARASWATULA P...



KUSHAL SANGW...



KARTIKEYA SAXE...



IMIJUNGLA LON...



ANSHUL MITTAL



NARESH BHARAS...



VARHADE AMEY...



Hemangee Kalpe...

01-Nov-2021 - Google Jamboard

28_Oct - Mentimeter

jamboard.google.com/d/1Bdp_J-09t0rgXuhYJEtZxtLJhsVBFsb3mwrB1E1R4/viewer?f=3

01-Nov-2021

3/4

Set background Clear frame

Share

ex-6

P1
1a A=1
1b print B

P2
2a B=1
2b print A

W
↓
R

$B=0 \Rightarrow 1b \rightarrow 2a$ (1)

$P0 \Rightarrow 1a \rightarrow 1b$ (2); $2a \rightarrow 2b$ (3)

$1b \rightarrow 2a \rightarrow 2b \rightarrow 1b$ $\langle A, B \rangle = \langle 0, 0 \rangle$

1a \rightarrow 1b \rightarrow 2a \rightarrow 2b
B="0" A="1"

A, B
1 1 ✓
0 1 ✓
1 0 ✓
0 0 not-SC

SC

IBM, TSO, PC
 \rightarrow SC X

TANYISH

NS

NALABOLU SAN...

KOUSIK RAJESH

YK

YOGESH KUMAR

AS

SU

ASWATHY N S

SWATI UPADHYAY

K CHITRA

AS

ADITYA KUMAR S...

SP

KS

SARASWATULA P...

KUSHAL SANGW...

KARTIKEYA SAXE...

IMIJUNGLA LON...

AM

ANSHUL MITTAL

NARESH BHARAS...

VA

VARHADE AMEY...

Hemangee Kalpe...

Safety-nets in relax W -> R

(1) to preserve program order requirement

* IBM 370 model

- Has special serialisation instructions placed between the two operations. E.g. Compare & Swap

- In Ex – 1

Flag1 = 1

Flag2 = 1

sync ;

sync ; //sync instr.

Read Flag 2 ;

Read Flag 1;



TANYISH



NALABOLU SAN...



KOUSIK RAJESH



YOGESH KUMAR



ASWATHY N S



SWATI UPADHYAY



K CHITRA



ADITYA KUMAR S...



SARASWATULA P...



KUSHAL SANGW...



KARTIKEYA SAXE...



IMIJJUNGLA LON...



ANSHUL MITTAL



NARESH BHARAS...



VARHADE AMEY...



Hemangee Kalpe...

Safety-nets in relax W -> R

*TSO

- No explicit safety nets available
- Programmers use read-modify-write (rmw) operation to provide illusion that program order is maintained between write and the following read
 - i.e. shared variables are updated automatically using synchronisation primitives
- In EX-1
 - To modify flag we write a rmw instruction
 - `rmw(flag1, 1)` // i.e. flag=1 r write flag
 - `rwm(flag2, read-value)` // “read flag” instruction replaced with rmw, i.e. write same value read



TANVISH



NALABOLU SAN...



KOUSIK RAJESH



YOGESH KUMAR



ASWATHY N S



SWATI UPADHYAY



K CHITRA



ADITYA KUMAR S...



SARASWATULA P...



KUSHAL SANGW...



KARTIKEYA SAXE...



IMIJUNGLA LON...



ANSHUL MITTAL



NARESH BHARAS...



VARHADE AMEY...



VADIGE PRANEET...



Hemangee Kalpe...

01-Nov-2021 - Google | 28_Oct - Mentimeter | +

jamboard.google.com/d/1Bdp_J-09t0rgXuhYJEtTzxtLTJhsVBFsb3mwrB1E1R4/viewer?F=0

01-Nov-2021

Set background Clear frame

EX-1

SC +

W → R

SC ✓ → W

P1

W
↓
R

Flag1 = 1

if (Flag2 == 0) ✓

// CS

rmw (flag1, 1) ✓

rmw (flag2, read-value)

P2

Flag2 = 1

if (Flag1 == 0) }

// CS

W → R

TANYISH

NS

NALABOLU SAN...

KOUSIK RAJESH

YK

YOGESH KUMAR

AS

ASWATHY N S

K CHITRA

AS

ADITYA KUMAR S...

SARASWATULA P...

KS

KUSHAL SANGW...

KARTIKEYA SAXE...

IMLIJUNGLA LON...

AM

ANSHUL MITTAL

NARESH BHARAS...

VA

VARHADE AMEY...

VC

VADIGE PRANEET...

Hemangee Kalpe...

Safety-nets in relax W -> R

*PC

- 'Read' replaced with rmw instruction
- 'Write replaced with rmw instruction is not sufficient because ...
- In TSO, rmw prevents write to any location until rmw instruction is executing
- BUT, in PC rmw prevents write to 'same' location during rmw but other locations can be changed
- Therefore, our rmw does not prevent other writes (to other locations) from being completed/reordered



TANYISH



NALABOLU SAN...



KOUSIK RAJESH



YOGESH KUMAR



ASWATHY N S



K CHITRA



ADITYA KUMAR S...



SARASWATULA P...



KUSHAL SANGW...



KARTIKEYA SAXE...



IMLIJUNGLA LON...



ANSHUL MITTAL



NARESH BHARAS...



VARHADE AMEY...



VADIGE PRANEET...



Hemangee Kalpe...

Safety-nets in relax W -> R

(2) To preserve write-atomicity requirement

- ***IBM**: Not required as it does not relax atomicity
- ***TSO**:
 - Need safety net for only those writes which are followed by reads to same location in same processor (as TSO allows “read own write early”)
 - Use rmw if write to location-A is followed by Read-A on same processor
- ***PC**:
 - In PC our own write-read order can be maintained by rmw
 - In PC others writes can be read early and this cannot be controlled by rmw, because we cannot control when the others write becomes visible to everybody
 - Therefore, this (others) write will be atomic, if all those who will read this (others) write will need to replace read by rmw so that no one reads this write early
 - Therefore all such reads in other processes have to be replaced by rmw (to prevent them from early reading the new value of others)



TANYISH



NALABOLU SAN...



KOUSIK RAJESH



YOGESH KUMAR



ASWATHY N S



K CHITRA



ADITYA KUMAR S...



SARASWATULA P...



KUSHAL SANGW...



KARTIKEYA SAXE...



IMLIJUNGLA LON...



ANSHUL MITTAL



NARESH BHARAS...



VARHADE AMEY...



VADIGE PRANEET...



Hemangee Kalpe...

01-Nov-2021 - Google

28_Oct - Mentimeter

jamboard.google.com/d/1Bdp_J-09t0rgXuhYJEtZxtLtJhsVBFsb3mwrB1E1R4/viewer?F=0

01-Nov-2021

1/4

Set background

Clear frame

SC +

W → R

SC ✓

W
↓
R

W → R

SC ✓

W
↓
W

W → R

↓
rmw

EX-1

P1

Flag1 = 1

if (Flag2 == 0) ✓

// CS

rmw (flag1, 1)

rmw (flag2, read-value)

P2

Flag2 = 1

if (Flag1 == 0)

// CS

W → R

P3

R Read Flag1 ✓

↓

R Read Flag2 ✓

W → R

↓

rmw

NS

TANVISH

NALABOLU SAN...

YK

KOUSIK RAJESH

VOGESH KUMAR

AS

ASWATHY N S

K CHITRA

AS

ADITYA KUMAR S...

SARASWATULA P...

KS

KUSHAL SANGW...

KARTIKEYA SAXE...

AM

IMLIJUNGLA LON...

ANSHUL MITTAL

VA

NARESH BHARAS...

VARHADE AMEY...

VC

VADIGE PRANEET...

SWATI UPADHYAY

SU

Hemangee Kalpe...

77% (2:21) Mon, November 1, 15:46

Safety-nets in relax W -> R

- Rmw instruction cannot be reordered with previous writes, therefore SC holds
- Disadvantage of having rmw in TSO, PC:
 - (i) the systems rmw instr implementation may not be suitable to use for all read and write
 - (ii) rmw which replaces 'read' involves a wasteful write and thus inv of all other copies
- SUN Sparc-V9 provides memory barrier (MEMBAR) or fence instruction to ensure the desired ordering

Wr X
MEMBAR
Rd X
- MEMBAR makes all write preceding it to complete before issuing subsequent read instruction
- W->R relax is good from hardware perspective, but from compiler perspective not much beneficial because R->W are finely interleaved in a program
- Therefore what we want is to relax R->R and W->W as well



TANVISH



NALABOLU SAN...



KOUSIK RAJESH



YOGESH KUMAR



ASWATHY N S



K CHITRA



ADITYA KUMAR S...



SARASWATULA P...



KUSHAL SANGW...



KARTIKEYA SAXE...



IMLIJUNGLA LON...



ANSHUL MITTAL



NARESH BHARAS...



VARHADE AMEY...



VADIGE PRANEET...



SWATI UPADHYAY



Hemangee Kalpe...

Relax W->R and W->W

- Relaxing write to read and write to write program orders
- Allows write to **bypass** earlier writes (to different location)
- Allows write buffer **merge**, retire write before previous write are complete
- Write misses are **overlapped** and seen as out-of-program-order
- Efficient as newer data values are seen by processor **earlier**
- **Sun Sparc PSO is the only model in this category**
- PSO guarantees write atomicity for write by other processors
- **PSO violates SC in ex-3**
 - **New value of flag can be seen by P2 before wr-A is complete**
- Solution: use STBAR instruction (same as MEMBAR) to preserve W->W order; for W->R order use TSO method (i.e. use of rmw)
- P1 { **A=1 ; STBAR ; Flag=1; } P2 {same as earlier}**



TANVISH



NALABOLU SAN...



KOUSIK RAJESH



YOGESH KUMAR



ASWATHY N S



K CHITRA



ADITYA KUMAR S...



SARASWATULA P...



KUSHAL SANGW...



KARTIKEYA SAXE...



IMLIJUNGLA LON...



ANSHUL MITTAL



NARESH BHARAS...



VARHADE AMEY...



VADIGE PRANEET...



SWATI UPADHYAY



Hemangee Kalpe...

01-Nov-2021 - Google Jamboard

28_Oct - Mentimeter

jamboard.google.com/d/1Bdp_J-09t0rgXuhYJEtZxtLTJhsVBFsb3mwrB1E1R4/viewer?f=1

01-Nov-2021

2/4

Set background Clear frame

Share

ex-2

SC → W → R ✓

SC = W → W ✗

P1

Data = 2000

Head = 1

P2

while(Head == 0);

Read Data

ex-3

A = W → R ✓

SC = W → W ✗

P1

A = 1

Flag = 1

P2

print B

print A

74% (3:00) Mon, November 1, 15:54

TANVISH

NS

NALABOLU SAN...

KOUSIK RAJESH

YK

VOGESH KUMAR

AS

ASWATHY N S

K CHITRA

AS

ADITYA KUMAR S...

SP

SARASWATULA P...

KS

KUSHAL SANGW...

KARTIKEYA SAXE...

AM

IMLIJUNGLA LON...

ANSHUL MITTAL

VA

NARESH BHARAS...

VARHADE AMEY...

VC

VADIGE PRANEET...

SU

SWATI UPADHYAY

Hemangee Kalpe...

01-Nov-2021 - Google Jamboard

28_Oct - Mentimeter

jamboard.google.com/d/1Bdp_J-09t0rgXuhYJEtZxtLTJhsVBFsb3mwrB1E1R4/viewer?f=1

01-Nov-2021

2/4

Set background Clear frame

Share

ex-2

SC → W → R ✓

SC = W → W X

P1

W Data = 2000

W Head = 1

P2

while(Head == 0);

Read Data

ex-3

A = W → R ✓

SC = W → W X

P1

A = 1

Flag = 1

P2

print B

print A

74% (3:00) Mon, November 1, 15:54

TANVISH

NS

NALABOLU SAN...

KOUSIK RAJESH

YK

VOGESH KUMAR

AS

ASWATHY N S

K CHITRA

AS

ADITYA KUMAR S...

SP

SARASWATULA P...

KS

KUSHAL SANGW...

KARTIKEYA SAXE...

AM

IMLIJUNGLA LON...

ANSHUL MITTAL

VA

NARESH BHARAS...

VARHADE AMEY...

VC

VADIGE PRANEET...

SU

SWATI UPADHYAY

Hemangee Kalpe...

Relax W- \rightarrow R and W- \rightarrow W

- To implement this with write buffers, include STBAR in FIFO of write buffer
- STBAR makes sure that writes before it are complete and retired
- Implementation of STBAR
 - Using counter
 - Count++ when write sent to memory
 - Count-- when memory sends ACK
 - All old write complete when count=0
- Again this optimisation is not sufficiently flexible to be useful to a compiler

=> we want to relax all orders !



TANVISH



NALABOLU SAN...



KOUSIK RAJESH



YOGESH KUMAR



ASWATHY N S



K CHITRA



ADITYA KUMAR S...



SARASWATULA P...



KUSHAL SANGW...



KARTIKEYA SAXE...



IMLIJUNGLA LON...



ANSHUL MITTAL



NARESH BHARAS...



VARHADE AMEY...



VADIGE PRANEET...



SWATI UPADHYAY



Hemangee Kalpe...

Relaxing all Program Orders

- No program orders are guaranteed by default, except data and control dependences within a process
- Benefit:
 - (1) multiple reads outstanding -> can be bypassed by later write
 - (2) multiple outstanding reads complete OoO -> hide read latency
 - (3) good for dynamically scheduled processors
 - (4) only these models allow key reordering and elimination of accesses -> done by compiler optimisations
- => these compiler optimisations are important + transparent to programmer.
 - Therefore these are the only reasonably high-performance memory models
- 5 models: WO, RC, RMO, PowerPC, Digital-Alpha
- All relax R->R, W ; W->W ; W->R
- All read own write early
- RC and PowerPC: read others' write early
- All models violate SC in (ex-1..6) all examples !



TANVISH



NALABOLU SAN...



KOUSIK RAJESH



YOGESH KUMAR



ASWATHY N S



K CHITRA



ADITYA KUMAR S...



SARASWATULA P...



KUSHAL SANGW...



KARTIKEYA SAXE...



IMLIJUNGLA LON...



ANSHUL MITTAL



NARESH BHARAS...



VARHAIDE AMEY...



VADIGE PRANEET...



SWATI UPADHYAY



Hemangee Kalpe...

WO : weak ordering

- Memory operations into two categories: Data and Synchronisation
- To enforce program order between two operations, the Programmer has to identify at least one of the memory operation in program as synchronisation operation
- Model's intuition = Reorder memory operation of data region between synch-op does not affect correctness of program
- Operation declared as synch-op provide safety net for enforcing program order. Implementation can be done using counters
- Complete all Rd and Wr before synch-op. Do not issue next-op until synch done
- The Read-Write block has several instructions which can be reordered within that block
- When synch-op are infrequent, as in many parallel programs WO typically provides considerable reordering freedom to hardware and compiler



TANYISH



NALABOLU SAN...



YOGESH KUMAR



ASWATHY N S



K CHITRA



ADITYA KUMAR S...



SARASWATULA P...



KUSHAL SANGW...



KARTIKEYA SAXE...



IMIJUNGLA LON...



ANSHUL MITTAL



NARESH BHARAS...



VARHADE AMEY...



VADIGE PRANEET...



SWATI UPADHYAY



Hemangee Kalpe...