

Lock-free?

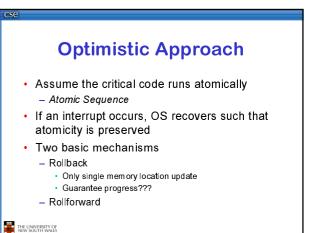
- · Avoid needing locking by using lock-free data structure
 - Still need short atomic sequences compare-and-swap
- · Lock-based data structure also need mutual exclusion to implement the lock primitive themselves.

THE UNIVERSITY OF NEW SOUTH WALES



- Syscalls?
- Processor Instructions?

THE UNIVERSITY OF NEW SOUTH WALES

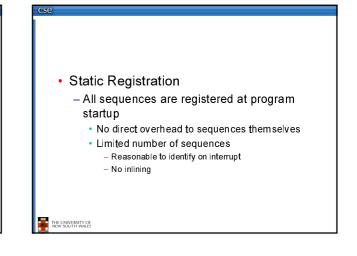


How does the OS know what is an atomic sequence?

Designated sequences

- Match well know sequences surrounding PC
 - Matching takes time
 - · sequence may occur outside an atomic sequences - Rollback might break code
 - Rollforward okay · Sequences can be inlined
 - · No overhead added to each sequence, overhead only on interruption

```
THE UNIVERSITY OF
NEW SOUTH WALES
```



Dynamic Registration

- Share a variable between kernel and userlevel, set it while in an atomic sequence
- Can inline, even synthesize sequences at runtime
- Adds direct overhead to each sequence

THE UNIVERSITY OF NEW SOUTH WALES

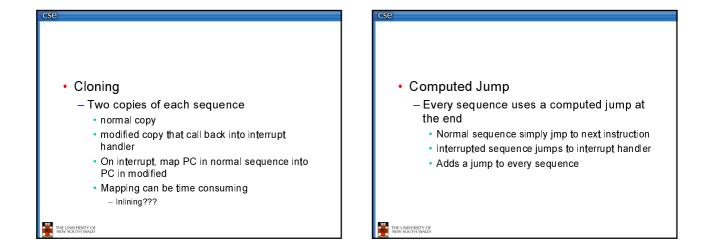
CSP

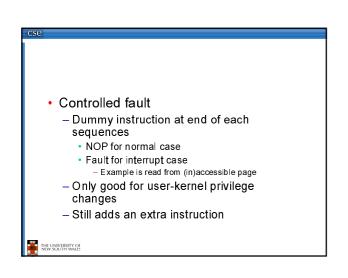
How to roll forward?

Code re-writing

- Re-write instruction after sequence to call back to interrupt handler
 - Cache issues need to flush the instruction cache??

THE UNIVERSITY OF NEW SOUTH WALES



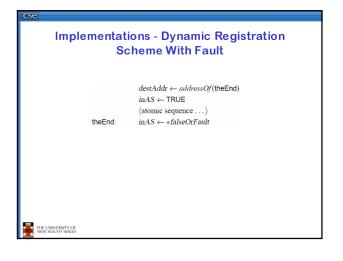


Limiting Duration of Roll forward

- Watchdog
- Restriction on code so termination can be inspected for

THE UNIVERSITY OF NEW SOUTH WALES

· · · · · · · · · · · · · · · · · · ·	Dynamic Registration With Jump
$destAddr \leftarrow addressOf(trinAS \leftarrow TRUE(atomic sequence)inAS \leftarrow FALSEjump destAddr$	neEnd)
theEnd:	
 stl zero, (r4) lda r3, sharedCounter ldl r2, (r3) addl r2, l, r2 stl r2, (r3) 	<pre># load address of inAS # load address of theEnd into r1 # inAs <- TRUE (0 = TRUE) # load address of sharedCounter # load value of sharedCounter # increment counter # store back new value # reset inAS to FALSE (not 0 = FALSE) # iump to address stored in r1</pre>
theEnd:	



Implementations - Dy Scheme W								
$destAddr \leftarrow addressOf(theEnd)$ inAS \leftarrow TRUE $\langle \operatorname{atomic sequence} \dots \rangle$ jump destAddr								
theEnd:								
lda r3, sharedCounter ld1 r2, (r3) add1 r2, 1, r2 st1 r2, (r3)	<pre># load address of theEnd into r1 # load address of sharedCounter # load value of sharedCounter # increment counter # store back new value # imput to address stored in r1</pre>							
<pre>* jmp (r1) theEnd:</pre>	# jump to address stored in r1							
THE UNIVERSITY OF New SOUTH Wales								

Results										
		DEC Alpha			HP PA-RISC 1.1					
	Technique	NULL	LIFO	FIFO	NULL	LIFO	FIFO			
	sigprocmask	1682	3045	3363	1787	3578	3590			
	Dyn/Fault	13	27	24	12	24	27			
	Dyn/Jump	9	16	13	11	21	27			
	Hyb/Jump	6	5	6	5	8	12			
	DI	4	3	4	4	5	12			
	CIPL	4	5	6	14	24	29			
	splx	44	89	88	30	63	73			
	PALcode	≥ 13	≥ 13	≥ 13	n/a	n/a	n/a			
	LL/STC	n/a	≥ 118	≥ 118	n/a	n/a	n/a			