6.175 Final Project
Part 0: Understanding Non-Blocking Caches and Cache Coherency

Notation

- Addresses are ordered triples:
  - (tag, index, offset)
- Cache lines are addressed with ordered pairs:
  - (tag, index)
- Cache slots are addressed by index
- Reading a cache line from memory:
  - M[(tag, index)]

Non-Blocking Cache

- Given: Processor requests and memory responses
- Assignment: Complete the following tables (not all cells should be filled)
  - We will focus on Loads first and Stores second
  - In later tables we integrate Loads and stores together

Multiple Requests in Flight – Part 1

Processor		Memory		Slot 0		Slot 1		Slot 2		Slot 3	
Req	Resp	Req	Resp	V	W	V	W	V	W	V	W
				0	0	0	0	0	0	0	0
1: Ld (0,0,0)											
2: Ld (0,1,0)											
3: Ld (0,2,0)											
4: Ld (0,3,0)											

Multiple Requests in Flight – Part 2

Processor		Memory		Slot 0		Slot 1		Slot 2		Slot 3	
Req	Resp	Req	Resp	V	W	V	W	V	W	V	W
			M[(0,0)]								
			M[(0,1)]								
			M[(0,3)]								
			M[(0,2)]								

Multiple Requests in Flight – Part 1 Example Solution

Processor		Memory		Slot 0		Slot 1		Slot 2		Slot 3	
Req	Resp	Req	Resp	V	W	V	W	V	W	V	W
				0	0	0	0	0	0	0	0
1: Ld (0,0,0)		Ld (0,0)		0	1	0	0	0	0	0	0
				0	1	0	0	0	0	0	0
2: Ld (0,1,0)		Ld (0,1)		0	1	0	1	0	0	0	0
				0	1	0	1	0	0	0	0
3: Ld (0,2,0)		Ld (0,2)		0	1	0	1	0	1	0	0
				0	1	0	1	0	1	0	0
4: Ld (0,3,0)		Ld (0,3)		0	1	0	1	0	1	0	1
				0	1	0	1	0	1	0	1
				0	1	0	1	0	1	0	1

Multiple Requests in Flight – Part 2 Example Solution

Processor		Memory		Slot 0		Slot 1		Slot 2		Slot 3	
Req	Resp	Req	Resp	V	W	V	W	V	W	V	W
				0	1	0	1	0	1	0	1
			M[(0,0)]	1	0	0	1	0	1	0	1
	Resp 1			1	0	0	1	0	1	0	1
			M[(0,1)]	1	0	1	0	0	1	0	1
	Resp 2			1	0	1	0	0	1	0	1
			M[(0,3)]	1	0	1	0	0	1	1	0
	Resp 4			1	0	1	0	0	1	1	0
			M[(0,2)]	1	0	1	0	1	0	1	0
	Resp 3			1	0	1	0	1	0	1	0
				1	0	1	0	1	0	1	0

Same Cache Line, Different Offset

Processor		Memory		Slot 0		other
Req	Resp	Req	Resp	V	W	# elem in LdQ
				0	0	0
1: Ld (0,0,0)						
2: Ld (0,0,1)						
			M[(0,0)]			
3: Ld (0,0,2)						
4: Ld (0,0,3)						

Same Index, Different Tag

Processor		Memory		Slot 0		
Req	Resp	Req	Resp	V	W	Tag
				0	0	?
1: Ld (0,0,0)						
2: Ld (1,0,0)						
			M[(0,0)]			
			M[(1,0)]			

Stores

Processor		Memory		Slot 0			# elements in
Req	Resp	Req	Resp	V	W	D	StQ
				0	0	0	0
St x (0,0,0)							
St y (0,0,1)							
			M[(0,0)]				
St z (0,0,2)							

Store Bypassing

Processor		Memory		Slot 0				# elements in	
Req	Resp	Req	Resp	V	W	D	Data(0)	StQ	LdQ
				0	0	0	x	0	0
1: Ld (0,0,0)									
St y (0,0,0)									
St z (0,0,0)									
2: Ld (0,0,0)									
			M[(0,0)]						

Resending Requests

Processor		Memory		Slot 0				# elements in	
Req	Resp	Req	Resp	V	W	D	Tag	StQ	LdQ
				0	0	0	?	0	0
St y (0,1,0)									
St z (0,0,0)									
			M[(0,0)]						
1: Ld (1,0,0)									
			M[(0,1)]						
			M[(1,0)]						
			M[(0,0)]						

Cache Coherency

- Given: Initial cache states for a single address and a cache request for that address
- Assignment: Write the rules each module needs to execute to perform the cache request
  - You may have to keep track of what messages are still in the message network. Unfortunately there is not enough space to include it in the table.

No Contest: Cache 0 - Ld

Cache 0			Cache 1			Parent				
						Cache 0		Cache 1		rule
state	waitp	rule	state	waitp	rule	state	waitc	state	waitc	
I	no		I	no		I	no	I	no	

No Contest: Cache 0 – Ld Example Answer

Cache 0			Cache 1			Parent				
						Cache 0		Cache 1		rule
state	waitp	rule	state	waitp	rule	state	waitc	state	waitc	
I	no		I	no		I	no	I	no	
I	S	1 (to S)	I	no		I	no	I	no	
I	S		I	no		S	no	I	no	2 (to S)
S	no	3 (to S)	I	no		S	no	I	no	

Other Cache is Writing: Cache 0 – Ld

Cache 0			Cache 1			Parent				
						Cache 0		Cache 1		rule
state	waitp	rule	state	waitp	rule	state	waitc	state	waitc	
I	no		M	no		I	no	M	no	

Lots of Downgrading: Cache 0 – St

M state for different tag. Need to first evict this line, and then upgrade to M for the desired tag

Cache 0			Cache 1			Parent				
						Cache 0		Cache 1		rule
state	waitp	rule	state	waitp	rule	state	waitc	state	waitc	
M	no		M	no		M	no	M	no	