Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχ. Και Μηχ. Υπολογιστών

Προηγμένα Θέματα Αρχιτεκτονικής Υπολογιστών

8ο εξάμηνο, Ροή Υ

Ακαδημαϊκή Περίοδος : 2011-2012

4η Άσκηση

Γερακάρης Βασίλης
<vgerak@gmail.com>
A.M. :03108092

Β' Μέρος

Β.1 - Πρωτόκολλο MESI

Γνωρίζουμε ότι το μέγεθος της κάθε cache είναι 4KB και το block size είναι 16 bytes. Επομένως :

#blocks = 4KB / 16B = 2^8 blocks

Επίσης, η cache μας είναι 2-way set-associative, άρα έχουμε ότι :

#sets = #blocks/2 = 2^7 γραμμές/set

Τέλος ισχύει :

BlockOffset = $\log_2$ (BlockSize) = 4 bits

IndexSize = $\log_2$ (#sets) = 7 bits

Tag size = 16 – (7 + 4) = 5 bits

Με βάση τα παραπάνω, οι διευθύνσεις που χρησιμοποιούνται από το πρόγραμμα αναλύονται ως εξής :

Address	Tag	Index	Block Offset
073C	00000	1110011	1100
0734	00000	1110011	0100
0738	00000	1110011	1000
0730	00000	1110011	0000
1F34	00011	1110011	0100
1F3C	00011	1110011	1100
2730	00100	1110011	0000
273C	00100	1110011	1100

Παρατηρούμε δηλαδή ότι όλες οι διευθυνσείς που αναφέρονται, απεικονίζονται στο ίδιο set (115) και επομένως αρκεί να αναφερόμαστε μόνο σε αυτό. Το tag μιας διεύθυνσης δείχνει σε ποιο block τουset αναφερόμαστε.

Επομένως, όταν παρακάτω γράφουμε “Bl.0” , αναφερόμαστε στο block 0 που απεικονίζεται στο 115ο set κ.ο.κ. Το κάθε block έχει το δικό του MESI state, όπως φαίνεται στους παρακάτω δύο πίνακες :

Action	P0 (MESI state)			P1 (MESI state)			Memory state (Block)
	Bl. 0	Bl. 3	Bl. 4	Bl. 0	Bl. 3	Bl. 4	
Start	I	I	I	I	I	I	V (Block 0)
P0: Read 073C	E	I	I	I	I	I	V (Block 0)
P1: Read 0734	S	I	I	S	I	I	V (Block 0)
P1: Write ‘1111’ to 0734	I	I	I	M	I	I	I (Block 0)
P0: Read 0738	S	I	I	S	I	I	V (Block 0)
P0:Write ‘2222’ to 0730	M	I	I	I	I	I	I (Block 0)
P1: Write ‘3333’ to 1F34	M	I	I	I	M	I	I (Block 3)
P0: Write ‘4444’to 1F3C	M	M	I	I	I	I	I (Block 3)
P0: Read 073C	M	M	I	I	I	I	I (Block 0)
P1: Write ‘5555’ to 2730	M	M	I	I	I	M	V (Block 4)
P0: Read 273C	M	I	S	I	I	S	V (Block 3)
P0: Write ‘66666’to273C	M	I	M	I	I	I	I (Block 4)

Actions	Cache P0 Set 115		Cache P1 Set 115		Memory
	1st Cache Line	2nd Cache Line	1st Cache Line	2nd Cache Line	
Start	Empty	Empty	Empty	Empty	Not Modified
P0:Read 073C	Bl.0	Empty	Empty	Empty	Not Modified
P1:Read 0734	Bl.0	Empty	Bl.0	Empty	Not Modified
P1:write ‘1111’ to 0734	Empty	Empty	[0734]<-1111	Empty	Not Modified
P0:read 0738	[0734]<-1111	Empty	[0734]<-1111	Empty	[0734]<-1111
P0:write ‘2222’ to 0730	[0730]<-2222	Empty	Empty	Empty	Not Modified
P1:write ‘3333’ to 1F34	Bl.0	Empty	[1F34]<-3333	Empty	[0730]<-2222
P0:write ‘4444’to 1F3C	Bl.0	Bl.3,[1F3C]<-4444	Empty	Empty	[1F34]<-3333
P0: read 073C	Bl.0	Bl.3	Empty	Empty	[1F3C]<-4444
P1:write ‘5555’ to 2730	Bl.0	Bl.3	Bl.4 [2730]<-5555	Empty	[2730]<-5555
P0:read 273C	Bl.0	Bl.4	Bl.4	Empty	Not Modified
P0:write ‘6666’to273C	Bl.0	[273C]<-6666	Empty	Empty	Not Modified

B.2 Memory Consistency

Αριθμούμε τις εντολές του παράλληλου προγράμματος. Έχουμε :

#	Processor 1	#	Processor 2
1	X = 2;	7	While (flag = 0);
2	Y = 1;	8	r3 = X;
3	flag = 1;	9	r4 = Y;
4	while (flag == 1);	10	Z = 4;
5	r1 = X;	11	flag = 0;
6	r2= Z;		

1) Σύμφωνα με το relaxed model που δίνεται μπορούμε να αλλάξουμε τη σειρά των εντολών αν και μόνο αν αυτές δεν αναφέρονται στον ίδιο καταχωρητή ή στην ίδια θέση μνήμης.

   Συνεπώς οι εντολές:

   1, 2, 3 του p1 και **8, 9, 11** του p2

   μπορούν να εκτελεστούν με οποιαδήποτε σειρά.

   Αντιθέτως οι εντολές **X=2;** και **r1=X**; πρέπει να εκτελεστούν με την σειρά αυτή στον p1, καθώς και οι προσπελάσεις στην flag.

   Έτσι, ο r1 μπορεί να πάρει μόνο την τιμή 2 στο τέλος. Για τους υπόλοιπους καταχωρητές υπάρχει επιτρεπτή ακολουθία εντολών ώστε να πάρουν οποιαδήποτε συνδυασμό τιμών. Επομένως, τελικά, όλοι οι επιτρεπτοί συνδυασμοί είναι :

r1	**r2**	**r3**	**r4**
2	0	0	0
2	0	0	1
2	0	2	0
2	0	2	1
2	4	0	0
2	4	0	1
2	4	2	0
2	4	2	1

2)

Η sequential consistency επιβάλλει τη σειριακή εκτέλεση των εντολών των 2 επεξεργαστών. Δηλαδή, όσο το flag=0 το δεύτερο stream “περιμένει” στο while, καθώς το πρώτο stream εκτελείται. Αντίστοιχα, το δεύτερο stream μετά την εκκίνησή του (αφού δημοσιευτεί και η εγγραφή flag=1), θα δημοσιεύσει σειριακά τις εγγραφές του μέχρι που θα δώσει και το flag=0, οπότε το πρώτο stream θα συνεχίσει και θα ολοκληρώσει.

Προκύπτει λοιπόν ο παρακάτω μοναδικός συνδυασμός τιμών:

r1	**r2**	**r3**	**r4**
2	4	2	1

3) Για να κάνουμε το χαλαρό μοντέλο να μας δίνει το ίδιο αποτέλεσμα με το sequentially consistent σύστημα θα πρέπει να χρησιμοποιήσουμε την flag ως μεταβλητή συγχρονισμού, προσθέτοντας τις 2 εντολές SYNCH όπως φαίνεται παρακάτω:

#	Processor 1	#	Processor 2
1	X = 2;	7	While (flag = 0);
2	Y = 1;	8	r3 = X;
3	**SYNCH**	9	r4 = Y;
4	flag = 1;	10	Z = 4;
5	while (flag == 1);	11	**SYNCH**
6	r1 = X;	12	flag = 0;
7	r2= Z;		