DidadoY

Previo Práctica 5

@	byte	bloque M	línea MC	TAG	HIT/MISS	TAG OUT
10f92150	10	87C90A	0A	10f92	MISS	-
10f92151	11	87C90A	0A	10f92	HIT	-
10f8a192	12	87C50C	0C	10f8a	MISS	-
10f92153	13	87C90A	0A	10f92	HIT	-
10f8b195	15	87C58C	0C	10f8b	MISS	10f8a
10f8b195	15	87C58C	0C	10f8b	HIT	-
10f93156	16	87C98A	0A	10f93	MISS	10f92
10f92157	17	87C90A	0A	10f92	MISS	10f93
10f8a198	18	87C50C	0C	10f8a	MISS	10f8b
10f93159	19	87C98A	0A	10f93	MISS	10f92
12f92250	10	97C912	12	12f92	MISS	-
10f92151	11	87C90A	0A	10f92	MISS	10f93
10f8a192	12	87C50C	0C	10f8a	HIT	-
12f92253	13	97C912	12	12f92	HIT	-
10f8b195	15	87C58C	0C	10f8b	MISS	10f8a
10f8b195	15	87C58C	0C	10f8b	HIT	-
10f93156	16	87C98A	0A	10f93	MISS	10f92
12f92257	17	97C912	12	12f92	HIT	-
10f8a298	18	87C514	14	10f8a	MISS	-
10f93159	19	87C98A	0A	10f93	HIT	-

@	byte	bloque M	conj MC	VIA	TAG	HIT/MISS	TAG OUT
10f92150	10	87C90A	0A	0	21F24	MISS	-
10f92151	11	87C90A	0A	0	21F24	HIT	-
10f8a192	12	87C50C	0C	0	21F14	MISS	-
10f92153	13	87C90A	0A	0	21F24	HIT	-
10f8b195	15	87C58C	0C	1	21F16	MISS	-
10f8b195	15	87C58C	0C	1	21F16	HIT	-
10f93156	16	87C98A	0A	1	21F26	MISS	-
10f92157	17	87C90A	0A	0	21F24	HIT	-
10f8a198	18	87C50C	0C	0	21F14	HIT	-
10f93159	19	87C98A	0A	1	21F26	HIT	-
12f92250	10	97C912	12	0	25F24	MISS	-
10f92151	11	87C90A	0A	0	21F24	HIT	-
10f8a192	12	87C50C	0C	0	21F14	HIT	-
12f92253	13	97C912	12	0	25F24	HIT	-
10f8b195	15	87C58C	0C	1	21F16	HIT	-
10f8b195	15	87C58C	0C	1	21F16	HIT	-
10f93156	16	87C98A	0A	1	21F26	HIT	-
12f92257	17	97C912	12	0	25F24	HIT	-
10f8a298	18	87C514	14	0	21F14	MISS	-
10f93159	19	87C98A	0A	1	21F26	HIT	-

3. El vector accederá a 4*10240 (integer) posiciones = 40960 posiciones; Como la línea de Cache ocupa 32 bytes, cada 32 bytes habrá un MISS, de manera que el número de MISS es:

MISS = 40960 / 32 = 1280

HIT = 40960 - 1280 = 39680

4. El vector accederá, nuevamente, a 40960 posiciones. Como el vector se accede secuencialmente (primero V[0], luego v[1] …) y como cada línea de Cache ocupa 32 bytes, cada 32 bytes habrá un MISS ya que no se puede aprovechar la 2-asociatividad por el acceso secuencial, de manera que el número de MISS es:
 # MISS = 40960 / 32 = 1280
 # HIT = 40960 - 1280 = 39680

5.
 # MISS = 80000/4 = 20000
 # HIT = 80000-20000 = 60000

6. Cada vector ocupa 40000 (10000 * 4) posiciones de memoria, y se acceden consecutivamente. Además, en cada iteración se accede a ambos. Esto hace que, por ejemplo, si se accede a vec1[0] éste irá al conjunto 0, vía 0, mientras que cuando se acceda a vec2[0], que también va al conjunto 0, vaya a la vía 1 aprovechando la 2-asociatividad y evitando solapamiento y, por lo tanto, muchos misses. De manera que, cada 32 bytes, habrá un MISS para cada vector. Por lo tanto, en número de MISS es:
#MISS = 40000/32 + 40000/32 = 2500
#HIT = 80000-2500 = 77500