מבנה מחשבים - תרגיל 7#

אריאל סטולרמן
קבוצה 02

(1)

נתון *cache* עם 64 בלוקים, גודל כל בלוק 16 בתים. נחשב לאן תמופה כתובת הבית 1200:

$\lfloor 1200 \div 16 \rfloor = 75$: נלך לבית ה"75", אך כיוון שגודל ה-*cache* הוא 64:

$75 \bmod 64 = 11$: מכאן שהכתובת תמופה לבלוק מספר 11.

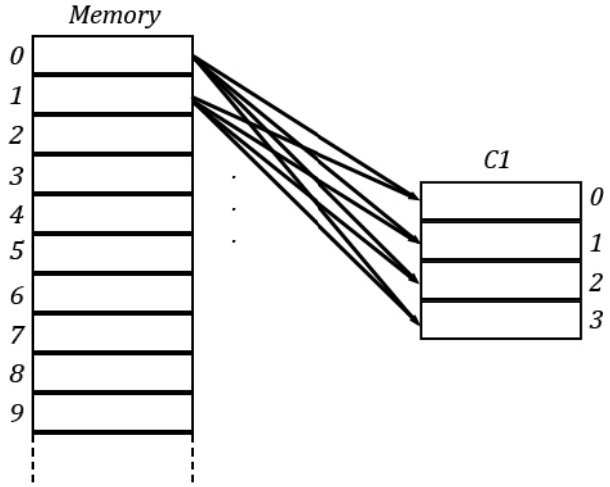
(2)

נסמן את ה-*cache* השונים:

- *C1*: *fully associative cache*
- *C2*: *two-way set associative cache*
- *C3*: *direct mapped cache*

לכל אחד מהקריאות נסמן *'m'* עבור *miss* ו-*'h'* עבור *hit*; בלוק מהזיכרון הראשי יסומן עם '', וב-*cache* ללא '':

C1:



call	*0*	*8*	*0*	*6*	*8*
m/h	*m*	*m*	*h*	*m*	*h*

קריאה ראשונה ל-'0': לא ב-*cache*, לכן *miss*. יושם ב-0.

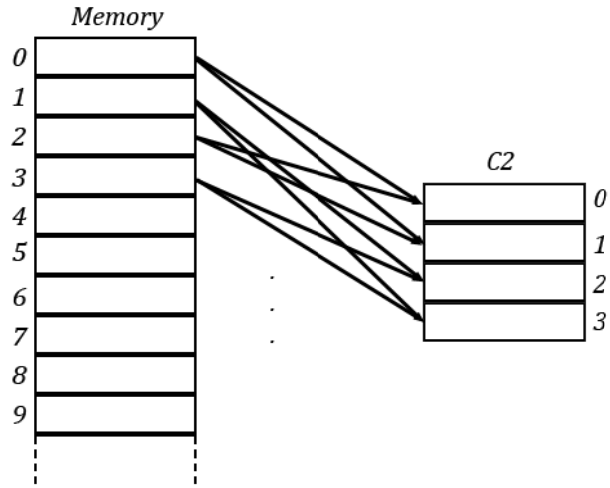
קריאה שניה ל-'8': לא ב-*cache*, לכן *miss*. יושם ב-1.

קריאה שלישית ל-'0': ב-*cache*, לכן *hit*.

קריאה רביעית ל-'6': לא ב-*cache*, לכן *miss*. יושם ב-2.

קריאה חמישית ל-'8': ב-*cache*, לכן *hit*.

C2:



Call	*0*	*8*	*0*	*6*	*8*
m/h	*m*	*m*	*h*	*m*	*m*

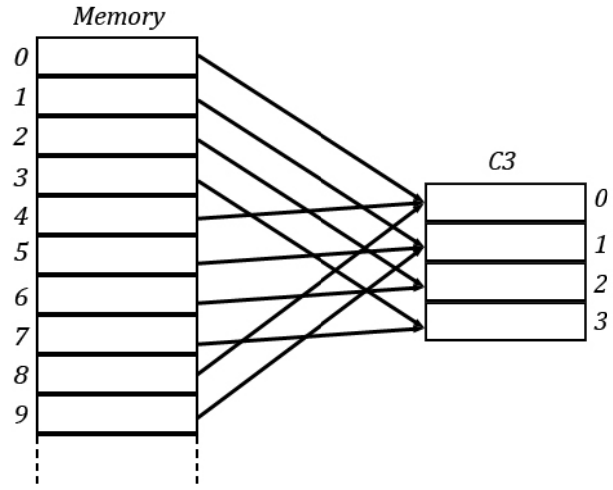
קריאה ראשונה ל-'0': לא ב-*cache*, לכן *miss*. יושם ב-0.

קריאה שניה ל-'8': לא ב-*cache*, לכן *miss*. יושם ב-1.

קריאה שלישית ל-'0': ב-*cache*, לכן *hit*.

קריאה רביעית ל-'6': לא ב-*cache*, לכן *miss*. לפי *LRU* יושם במקום '8', כלומר ב-0.

קריאה חמישית ל-'8': כעת לא ב-*cache*, לכן *miss*. לפי *LRU* יושם במקום '0', כלומר ב-1.



C3:

Call	*0*	*8*	*0*	*6*	*8*
m/h	*m*	*m*	*m*	*m*	*m*

קריאה ראשונה ל-'0': לא ב-*cache*, לכן *miss*. יושם ב-0.

קריאה שניה ל-'8': לא ב-*cache*, לכן *miss*. יושם ב-0 במקום '0'.

קריאה שלישית ל-'0': כעת לא ב-*cache*, לכן *miss*. יושם ב-0 במקום '8'.

קריאה רביעית ל-'6': לא ב-*cache*, לכן *miss*. יושם ב-2.

קריאה חמישית ל-'8': כעת לא ב-*cache*, לכן *miss*. יושם ב-0 במקום '0'.

(3)

נניח נתונים שני סוגי ה-*cache* (*two-way associative, direct mapped*) כמו אלו משאלה 2: 4 בלוקים, כל בלוק בגודל מילה אחת. להלן רצף קריאות מהזיכרון עבור שני ה-*cache* הנ"ל:

Call	*Two-way associative*	*Direct mapped*
0	*miss*: לא ב-*cache*, יושם ב-0.	*miss*: לא ב-*cache*, יושם ב-0.
2	*miss*: לא ב-*cache*, יושם ב-1.	*miss*: לא ב-*cache*, יושם ב-2.
0	*hit*: נמצא ב-*cache*.	*hit*: נמצא ב-*cache*.
4	*miss*: לא ב-*cache*, יושם לפי *LRU* במקום '2', כלומר במקום 1.	*miss*: לא ב-*cache*, יושם במקום '0', כלומר במקום 0.
2	*miss*: לא ב-*cache*, יושם לפי *LRU* במקום '0', כלומר במקום 0.	*hit*: נמצא ב-*cache*.
0	*miss*: לא ב-*cache*, יושם לפי *LRU* במקום '4', כלומר במקום 1.	*miss*: לא ב-*cache*, יושם במקום '4', כלומר במקום 0.
4	*miss*: לא ב-*cache*, יושם לפי *LRU* במקום '2', כלומר במקום 0.	*miss*:לא ב-*cache*, יושם במקום '0', כלומר במקום 0.
סה"כ	*6/7 misses*	*5/7 misses*

ב-*direct mapped cache* קיבלנו *miss* אחד פחות לדוגמא זו מאשר ב-*two-way associative cache*, כנדרש.

(4)

נתונים:

- *C1*: *direct mapped cache*, 16 בלוקים, כל בלוק בגודל מילה אחת. עלות *miss*: 8 מחזורי שעון.
- *C2*: *direct mapped cache*, 4 בלוקים, כל בלוק בגודל 4 מילים. עלות *miss*: 11 מחזורי שעון.

נסמן את ה-*miss rate* של *C1* ב-l, ואת של *C2* ב-m. צריך שיתקיים:

- $m < l$: כמות הפספוסים של *C2* נמוכה משל *C1*.
- $11m > 8l$: *C2* מבזבז יותר מחזורי שעון על *misses* מאשר *C1*.

נשים לב כי ההצבה: $m = 3, l = 4$ מקיימת את שני האי שוויונים הנ"ל: ל-*C2* יהיו רק 3 *misses* לעומת 4 ל-*C1*, אך בסה"כ יבזבז 33 מחזורי שעון על *misses*, לעומת *C1* שיבזבז 32 מחזורי שעון בלבד. נסתכל על הקריאות הבאות:

Call	*C1*	*C2*
0	*miss*: לא ב-*cache*, יושם ב-0.	*miss*: לא ב-*cache*, מילים 0-3 יושמו בבלוק המתחיל ב-0.
1	*miss*: לא ב-*cache*, יושם ב-1.	*hit*: הושם ב-*cache* בקריאה הקודמת.
4	*miss*: לא ב-*cache*, יושם ב-4.	*miss*: לא ב-*cache*, מילים 4-7 יושמו בבלוק המתחיל ב-4.
8	*miss*: לא ב-*cache*, יושם ב-8.	*miss*: לא ב-*cache*, מילים 8-11 יושמו בבלוק המתחיל ב-8.
סה"כ	*4 misses, 32 clk-cycles on misses*	*3 misses, 33 clk-cycles on misses*

(5)

להלן מידול *SR-FF* ב-*VHDL*:

```
entity SR-FF is
     port(S, R, CLK: in bit;
          Q: inout bit; QN: out bit:='1');
End SR-FF;

architecture SR-FF1 of SR-FF is
begin
     process(CLK)
     begin
          if CLK='1' then
               if (S='1' and R='0') then Q <= '1' after 10 ns;
               elseif (S='0' and R='1') then Q <= '0' after 10 ns;
               end if;
          end if;
     end process;
     QN <= not Q;
end SR-FF1;
```