זיכרון מטמון 2

מבנה מחשבים ספרתיים
234267

כיצד נבחר את גודל המטמון?

מצד אחד, נרצה מטמון גדול כמה שיותר כדי לקבל שיפור ב- hit rate.

מצד שני , כאשר המטמון גדול מדי , הוא צורך אנרגיה רבה וזמן הגישה עלול להיות ארוך באופן משמעותי.

כיצד נוכל לשלב בין הצורך למטמון גדול למטמון זריז?

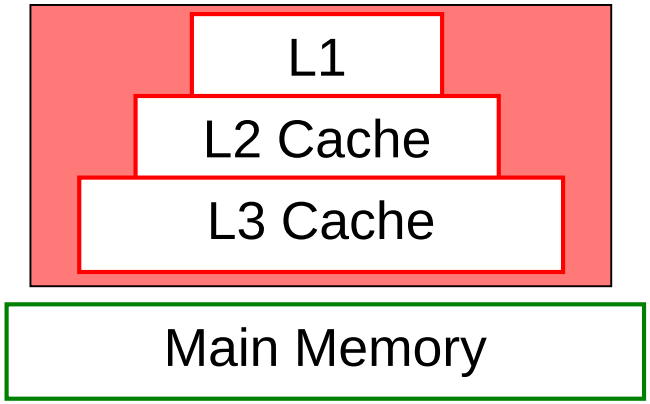
הפתרון: היראכיית זיכרון

המעבד מכיל מספר רמות של מטמון.

המטמון ברמה הראשונה (L1) הוא מטמון קטן ומהיר.

זמן הגישה ל-L1 הוא מחזורי שעון בודדים.

הרמות הגבוהות מכילות מטמונים גדולים יותר ובעלי זמן גישה איטי יותר.



Multi level Cache

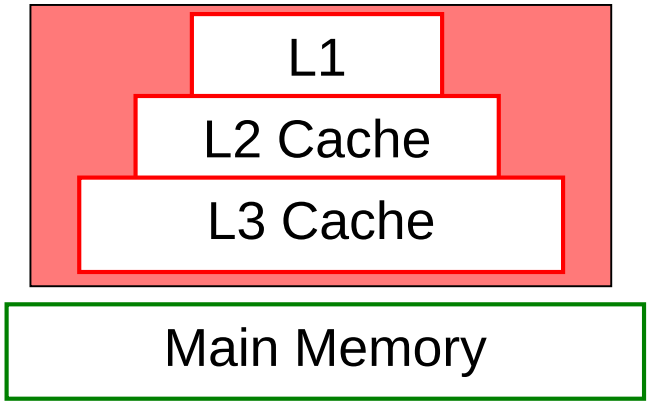
הירארכיית זיכרון

כאשר ניגש למידע בזיכרון , ניגש תחילה ל-L1.

במקרה של פגיעה, משתמשים בנתון שב-L1.

בהחטאה, נעביר את בקשת הגישה לרמה מעל (L2).

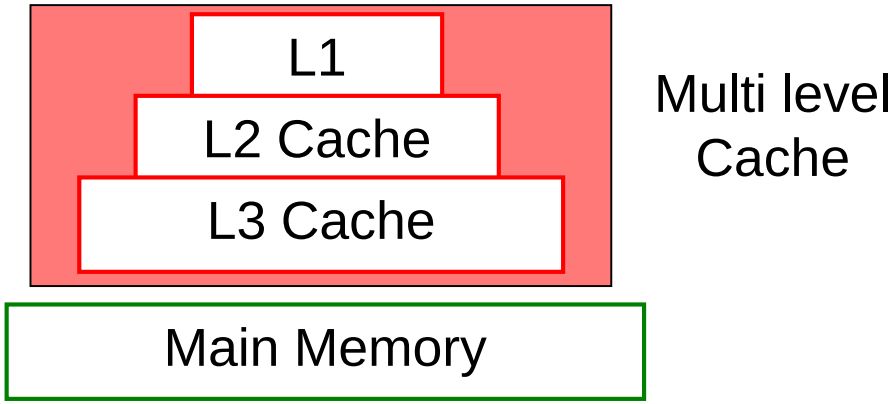
עבור כל החטאה ניגש לרמה גבוהה יותר עד שבסופו של דבר ניגש לזיכרון עצמו.



Multi level Cache

שאלה 1

במחשב C123 הותקנה היררכית זיכרון המכילה 3 רמות של cache:



ה- CPU יזם קריאה מכתובת m.

בקשה זאת גררה שרשרת ארועים בשלושת המטמונים.

אילו מהאירועים הבאים אפשריים, אילו לא ולמה?

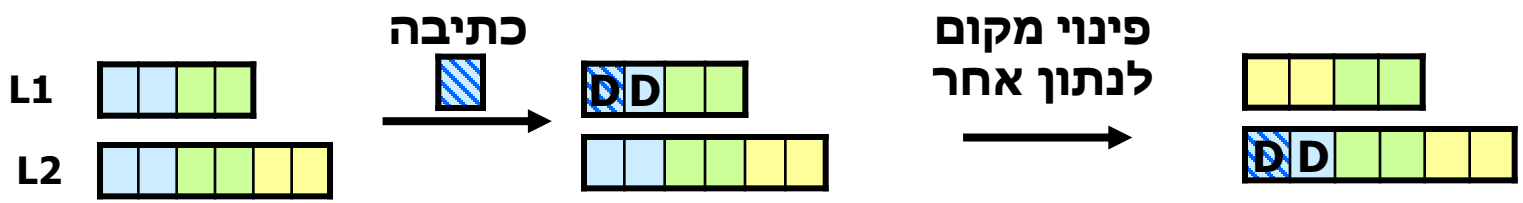
כל האירועים מתוארים לפי סדר: קודם ב- L1 אח"כ L2 ובסוף ב- L3.
M=Miss H=Hit ומשבצת ריקה מסמלת שלא קרה דבר (כלומר לא בוצעה פניה למטמון הזה).

L1	L2	L3	אפשרי?	נימוק
H	H	H		
H	M			
M	H			
	M	H		
H				
M	M	M		
M	H	M		
M	H	H		
M	M	H		
M	M			
M		H		

L1	L2	L3	אפשרי?	נימוק
H	H	H	לא	אחרי hit ברמה מסויימת לא פונים לרמה שמתחתיה.
H	M		לא	אחרי hit ברמה מסויימת לא פונים לרמה שמתחתיה.
M	H		כן	המידע נמצא ברמה השנייה
	M	H	לא	הפנייה לרמות עפ"י סדר. חייבים לעבור דרך L1.
H			כן	המידע נמצא ברמה הראשונה
M	M	M	כן	המידע נמצא רק בזכרון הראשי
M	H	M	לא	אחרי hit ברמה מסויימת לא פונים לרמה שמתחתיה.
M	H	H	לא	אחרי hit לא פונים לרמות שמתחת.
M	M	H	כן	המידע נמצא ברמה השלישית
M	M		לא	חסרה פניה לרמה L3 אחרי ההחטאות ברמות L1 ו- L2
M		H	לא	הפנייה לרמות עפ"י סדר. חייבים לעבור דרך L2.

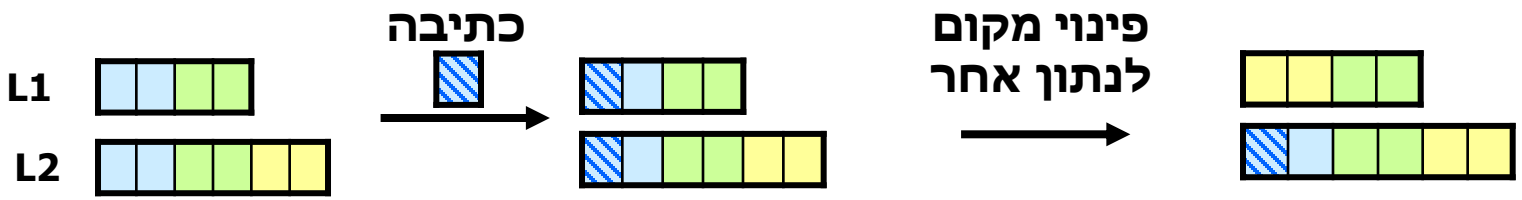
בחינה מחדש: מדיניות עדכון למטמון

Write back – בזמן כתיבה נרשום רק לרמה הדרושה. העדכון לשאר הרמות יבוצע כאשר הנתון ייזרק מהמטמון.



הציור – עבור מטמון (fully associative, 2 מילים בבלוק) בעל שתי רמות שפועלות במדיניות כתיבה write back.

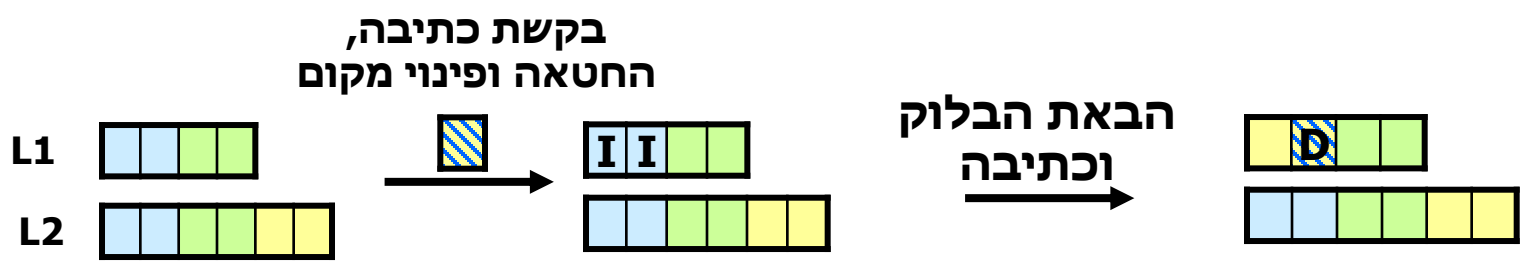
Write through – בזמן כתיבה נרשום את הערך החדש של הנתון גם לרמה שמתחתיו. כאשר הנתון ייזרק מהמטמון, אין צורך לעדכן את הרמה שמתחתיו.



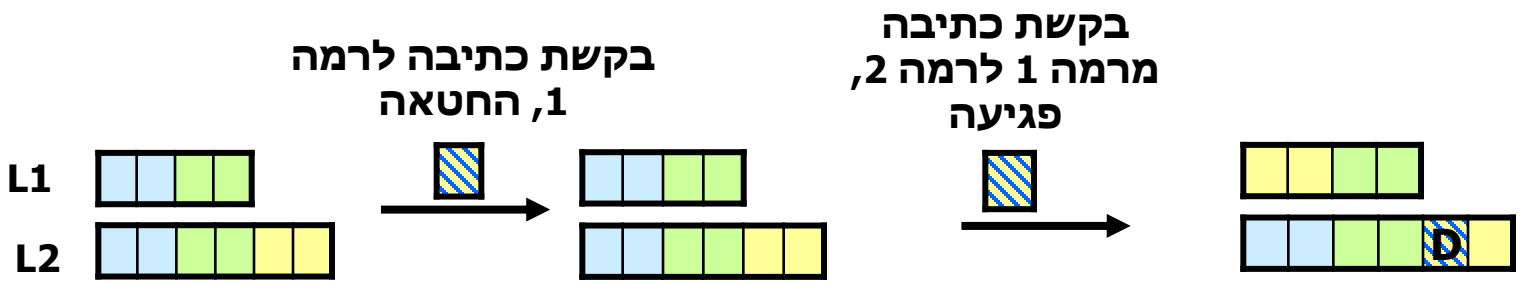
הציור – עבור מטמון (fully associative, 2 מילים בבלוק) בעל שתי רמות שפועלות במדיניות כתיבה write through.

בחינה מחדש -מדיניות כתיבה : Write Miss

Write allocate – במקרה של החטאה מורידים לרמה שמתחת בקשה להבאת הבלוק. טרם הבאת הבלוק מהרמה התחתונה מפנים את המקום המתאים (מבחינת set ומדיניות החלפה).



No Write allocate – במקרה של החטאה מורידים לרמה שמתחת את בקשת הכתיבה עצמה. אין הבאה של הבלוק לרמה הנדרשת.



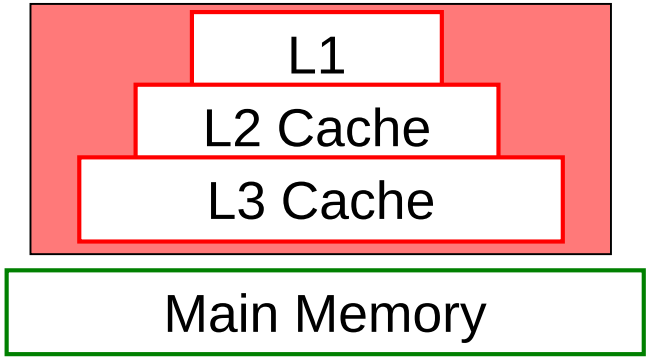
הציור – עבור מטמון (fully associative, 2 מילים בבלוק) בעל שתי רמות שפועלות במדיניות כתיבה write back

עקרון ההכלה

במטמון ההיררכי שעובד לפי עקרון ההכלה כל רמה של הזיכרון ההיררכי מכילה את כל הנתונים שנמצאים ברמה שמעליה.

הדבר נועד ע"מ לפשט את הפרוטוקולים של התקשורת בין רמה לרמה.

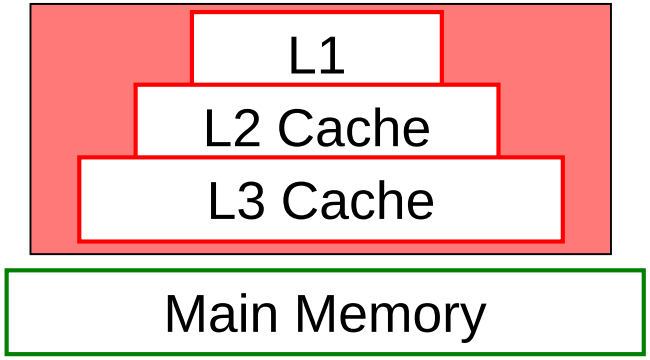
העיקרון לא מתקיים בכל המעבדים.



Multi level Cache

עקרון ההכלה

כדי לוודא שעיקרון ההכלה מתקיים,כל פינוי של שורת מטמון מרמה נמוכה (L2 או L3) יגרור פינוי של אותה השורה גם מהמרמות הגבוהות יותר.



Multi level Cache

שאלה 2

נתונה מערכת פנטיום עם L2 cache כך שעיקרון ההכלה מתקיים.

בכל אחד מהמקרים הבאים סמן האם L2 מועיל או לא, ונמק.

1. שתי רמות המטמון באותו גודל.
2. זמן הגישה לשתי רמות המטמון זהה, L2 גדול מ- L1.
3. זמן הגישה ל- L2 ולזיכרון הראשי זהה .

1. לא מועיל. אין תועלת ב- L2, כי לפי עקרון ההכלה התוכן של L1 מוכל ב- L2. ולכן כל מה שנמצא ב- L2 נמצא גם ב- L1 (ולכן לא נצטרך לקרוא או לכתוב כלום מ- L2) וכל מה שלא נמצא ב- L1 גם לא נמצא ב- L2, (ולכן לא נצטרך לחפש כלום ב- L2)
2. מועיל. יש תועלת ב- L2 כי הוא גדול יותר. היות וזמן הגישה זהה, נעדיף לקרוא ישר מ- L2 ולכן דווקא L1 אינו מועיל לביצועי המערכת.
3. לא מועיל. L2 מיותר כי ניתן לגשת לזיכרון הראשי באותה מהירות וללא חשש להחטאה.

עיקרון ההכלה

נתון מטמון בעל 3 רמות מטמון השומר על עיקרון ההכלה

עם העברת בקשת קריאת תוכן כתובת m נוצרה במטמונים שרשרת אירועים במהלכה נדרש מטמון L3 לפנות אחת מהשורות שלו אל הזכרון הראשי

הנח שלכל המטמונים אותו גודל שורה וכולם מאורגנים ב- K-way. כמו כן הנח שכל המטמונים היו מלאים בעת בקשת הקריאה.

מה היה המצב בהיררכיית המטמונים שהוביל לצורך בהחלפת שורה ב- L3?

תשובה: אם יש צורך להחליף שורה ב- L3 מכאן שהיו 3 החטאות בכל שלוש הרמות של המטמון.

עיקרון ההכלה

Snoop

- snoop the caches above when an entry is victimized in order to keep inclusion

במקרה שרמת מטמון מסויימת רוצה למחוק נתון מסוים היא "מסתכלת" על הדרגה שמעל ובודקת האם הנתון נמצא גם שם, ואם כן יש למחוק אותו קודם כדי לשמור על עקרון ההכלה.

נתאר (באופן איכותי) את שהתרחש מרגע גילוי הצורך בהחלפת השורה ב- L3 ועד התייצבות תוכן היררכיית הזיכרון ומסירת תוכן כתובת m למעבד .

כל המטמונים עובדים לפי מדיניות write-back, מדיניות LRU ושומרים על עקרון ההכלה.

עיקרון ההכלה

1. L1 מזהה read miss ומבקש את הנתון ממטמון L2
2. L2 מזהה read miss ומבקש את הנתון ממטמון L3
3. L3 מזהה read miss ובוחר שורה להחלפה
4. Snoops:
   - מבצע snoop אל L2 כדי לבדוק אם עותק של שורה זאת נמצא אצלו
   - אם יש עותק של השורה המוחלפת ב- L2, L2 מבצע snoop אל L1
   - אם יש עותק של השורה המוחלפת ב- L1:
     - אם העותק במצב M (modified), L1 כותב אותו אל L2.
     - בכל מקרה, L1 מסמן את השורה הנ"ל כפנוייה (מצב I – Invalid).
   - אם יש עותק של השורה המוחלפת ב- L2:
     - אם העותק במצב M (modified), L2 כותב אותו אל L3.
     - בכל מקרה, L2 מסמן את השורה הנ"ל כפנוייה (מצב I – Invalid).
7. אם השורה המתפנה מ- L3 היא במצב M, L3 כותב אותה לזכרון הראשי ומסמן את השורה הזאת כפנויה (מצב I).

עיקרון ההכלה

8. L3 קורא את השורה הנדרשת (החדשה) מהזכרון הראשי ומעדכן LRU.

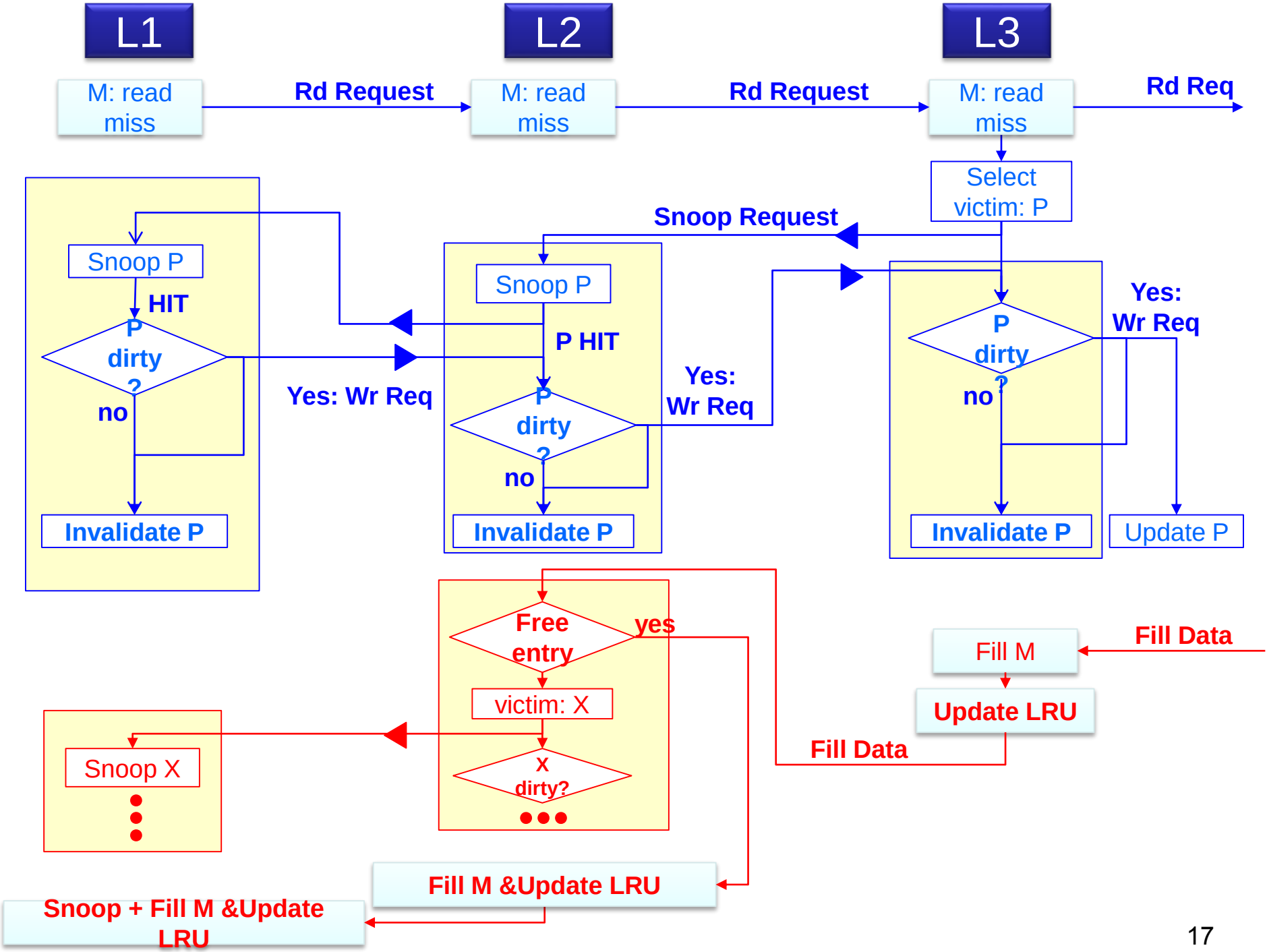
9. במידה ויש מקום פנוי ב-L2 ב- set הרלוונטי, L2 קורא את השורה הנדרשת מ- L3 אל אחת השורות הפנויות ב- set המתאים ומעדכן LRU.

   במידה ואין מקום פנוי, L2 בוחר שורה לפנות, מבצע snoop & Invalidate ל- L1 (כולל כתיבה של המידע מרמה L1 לרמה L2 במקרה של dirty), ורושם את המידע לרמה L3 במידה והשורה הנ"ל מסומנת אצלו כ- dirty.

   לאחר שפונה מקום מתאים, הנתון נכתב ל- L2, ומעודכן LRU.

10. במידה ויש מקום מתאים ב- L1, L1 קורא את השורה הנדרשת מ- L2 אל אחת השורות הפנויות ב- set המתאים ומעדכן LRU.

    במידה ואין מקום מתאים, מפונה אחת השורות (נכתבת לרמה L2 במידה ובמצב dirty), מעתיק את הנתון מרמה L2 ומעדכן LRU.


L1
L2
L3
M: read miss
Rd Request
M: read miss
Rd Request
M: read miss
Rd Req
Select victim: P
Snoop Request
Snoop P
HIT
P dirty ?
no
Yes: Wr Req
Invalidate P
Snoop P
P HIT
P dirty ?
no
Yes: Wr Req
Invalidate P
P dirty ?
no
Yes: Wr Req
Invalidate P
Update P
Free entry
yes
victim: X
X dirty?
Snoop X
Fill M
Fill Data
Update LRU
Fill Data
Fill M &Update LRU
Snoop + Fill M &Update LRU

שאלה 3

נתון מטמון בעל 3 רמות השומר על עיקרון ההכלה.
מטמונים L2 ו- L3 פועלים בעת write miss במדיניות no write allocate (כלומר, במקרה של החטאה בקשת הכתיבה מועברת לרמה הבאה).
בשלב מסויים בצע המעבד כתיבה לכתובת n. בקשה זו גררה שרשרת ארועים בהיררכיית הזכרון אשר נפתחה ב- write miss במטמון L1 ובמהלכה (של שרשרת הארועים) פונתה שורה ממטמון L3.

<u>שאלה:</u> מהי מדיניות הwrite miss - של מטמון L1?

<u>תשובה</u>: מדיניות write miss של L1 היא write allocate, אחרת לא היה צורך לקרוא שורות למטמון ולפנות שורה מ- L3. (הוכחה על דרך השלילה)

<u>שאלה</u>: האם יתכן שיתרחש במעבד אירוע מסוג L2 write miss? אם כן – באילו סיטואציות?

תשובה: לא.

עקב עקרון ההכלה לא יתכן write miss באחת הרמות התחתונות בלי שיהיה write miss ב- L1. כל write miss ב- L1 שהוא write allocate גורר בקשת קריאה (כלומר line fill) מ- L1 אל L2, מכיוון שאנחנו שומרים על עקרון ההכלה, לא יתכן write miss ב- L2.

בגלל ש L1 במדיניות write-allocate וע"ס עקרון ההכלה אין משמעות לעובדה ש- L2 ו- L3 במדיניות no-write-allocate מכיוון שלא יתכן בהם write miss.

שאלה 4

נתון מעבד בעל שתי רמות מטמון .

גודל הכתובת 32 ביט.

גודל שורת מטמון עבור 2 המטמונים הוא 32 bytes .

מדיניות הכתיבה עבור שני המטמונים היא Write back

מדיניות הפינוי עבור שני המטמונים היא LRU .

עקרון ההכלה מתקיים.

מטמון L1 הוא 2-way set associative וגודלו 1KB.

מטמון L2 הוא 2-way set associative וגודלו 4KB .

שאלה 4

מריצים את התוכנית הבאה:

```
Int  arr[512];
Int S=0;
For (int i=0; i<2; i++) {
        for (int j=0; j<512; j++)
                s+=arr[j];
        }
```

<u>הנחות</u>:

*המשתנים S,i,j והמצביע arr שמורים ברגיסטרים.

*גודל משתנה מסוג Integer הוא 4 בתים.

*המטמון ריק בתחילת התוכנית.

*המערך מיושר לזיכרון .

שאלה 4

א.מהו אחוז הפגיעה עבור L1 ?

```
Int  arr[512];
Int S=0;
For (int i=0; i<2; i++) {
        for (int j=0; j<512; j++)
                s+=arr[j];
        }
```

L1 : block size 32 bytes,1KB data size, LRU replacement policy

אנו עוברים על המערך פעמיים:

באיטרציה הראשונה של הלולאה החיצונית, נפספס במטמון רק כאשר נקרא בלוק חדש.

מכיוון שגודל שורת מטמון הוא 32 Byte וגודל משתנה integer הוא 4 בתים, נכנסים 8 איברים בבלוק.

אנו מפספסים רק בגישה 1 על כל 8 גישות ולכן ה- HR הוא 7/8 .

שאלה 4

א.מהו אחוז הפגיעה עבור L1 ?

```
Int  arr[512];
Int S=0;
For (int i=0; i<2; i++) {
        for (int j=0; j<512; j++)
                s+=arr[j];
        }
```

L1 : block size 32 bytes,1KB data size, LRU replacement policy

בתחילת האיטרציה השנייה, המטמון יכיל רק את החצי השני של המערך (איברים 256-511) בגלל שמדיניות הפינוי היא LRU .

כך שיהיו לנו החטאות עבור כל גישה ראשונה הבלוקים שמכילים את התאים 0-255 .

בנוסף, כאשר ניגש לתא 256 בפעם השנייה יכיל המטמון רק את איברים 0-255 ולכן נחטיא כל פעם שניגש לבלוק חדש גם עבור חלק המעבר השני.

לכן ה- HR באיטרציה השנייה ועבור התוכנית כולה ה- HR הוא 7/8 .

שאלה 4

א.מהו אחוז הפגיעה עבור L2 ?

```
Int  arr[512];
Int S=0;
For (int i=0; i<2; i++) {
        for (int j=0; j<512; j++)
                s+=arr[j];
        }
```

L1 : block size 32 bytes,1KB data size, LRU replacement policy

אחוז הפגיעה עבור L2 מוגדר כאחוז הבקשות הגישה לזיכרון ב- L2 בהן מתרחשת פגיעה . כלומר, נתייחס רק לבקשות בהן ישנה החטאה ב-L1 .

באיטרציה הראשונה, ניגש למטמון רק במעבר בין בלוקים. מכיוון ש-L2 ריק בתחילת הריצה, תמיד נחטיא. לכן עבור הגישה הראשונה ה-HR הוא 0 .

L2 גדול מספיק כדי להכיל את המערך כולו, לכן באיטרציה השנייה תהיה פגיעה בכל גישה ל- L2 . לכן סה"כ ה-HR של L2 הוא ½ .

שאלה 4

ג. מוסיפים רמת מטמון נוספת, L3 בגודל של 8MB זיכרון.
זמן הגישה ל-L3 הוא כמחצית מזמן הגישה לזיכרון.
כיצד הוספת L3 תשפיע על ביצועי התוכנית
(בהנחה שהוא ריק בתחילת ריצתה) ?

נשים לב שהחטאות היחידות שיש בקוד עבור L2 הן החטאות מסוג compulsory, כך שב- L3 יהיו לנו רק החטאות compulsory.
לכן, לא קיבלנו שיפור בביצועים .

לעומת זאת, החטאות ב- L2 הן יקרות יותר, מכיוון שעכשיו נבצע גישה נוספת ל- L3 (שבהכרח תהיה החטאה בתוכנית הנ"ל) לפני שניגש לזיכרון עצמו.
לכן, הדבר יפגע בביצועי התוכנית .

שאלה 5

להלן תוכנית לחישוב ממוצע ציונים:

```
sum = 0;
for (i = 0; i < NUM_STUDENTS; i++) {
    sum += grades[i];
}
average = sum / NUM_STUDENTS;
```

כל איבר במערך grades הוא בגודל 4 בתים.

גודל שורה ב-L1 cache היא 16 בתים.

הזמן הדרוש להבאת שורה מהזיכרון הוא 32 cycle.

נדרש לתכנן מערכת Prefetch בחומרה שתשפר את ה-hit rate ב-L1

יש להתייחס הן לאלגוריתם המערכת המוצעת והן לתזמון המערכת.

שאלה 5

תשובה:

מערכת Prefetch יעילה צריכה להוציא בקשה להבאת שורת Cache בתזמון כזה שכאשר תגיע הבקשה (Demand) לאותה שורה היא כבר תהיה ב Cache.

במקרה שלנו דרושים 32 מחזורי שעון כדי להביא שורה ל Cache ולכן צריך לתזמן את המערכת ה-Prefetch כך שנייצר בקשה לכתובת המתאימה 32 מחזורי שעון לפני שהתכנית מגיעה לבקש את הכתובת.

נשתמש באלגוריתם Stride. כיוון שבכל שורה ישנם 4 איברים של המערך, בכל גישה לשורה n יש ליצר בקשת Preftech לשורה n+8.

כך נבטיח הגעת הנתונים ל Cache בתזמון מתאים לפני שהם ידרשו על ידי התוכנית.