ארגון המחשב ומערכות הפעלה

אביב תשפ"ד

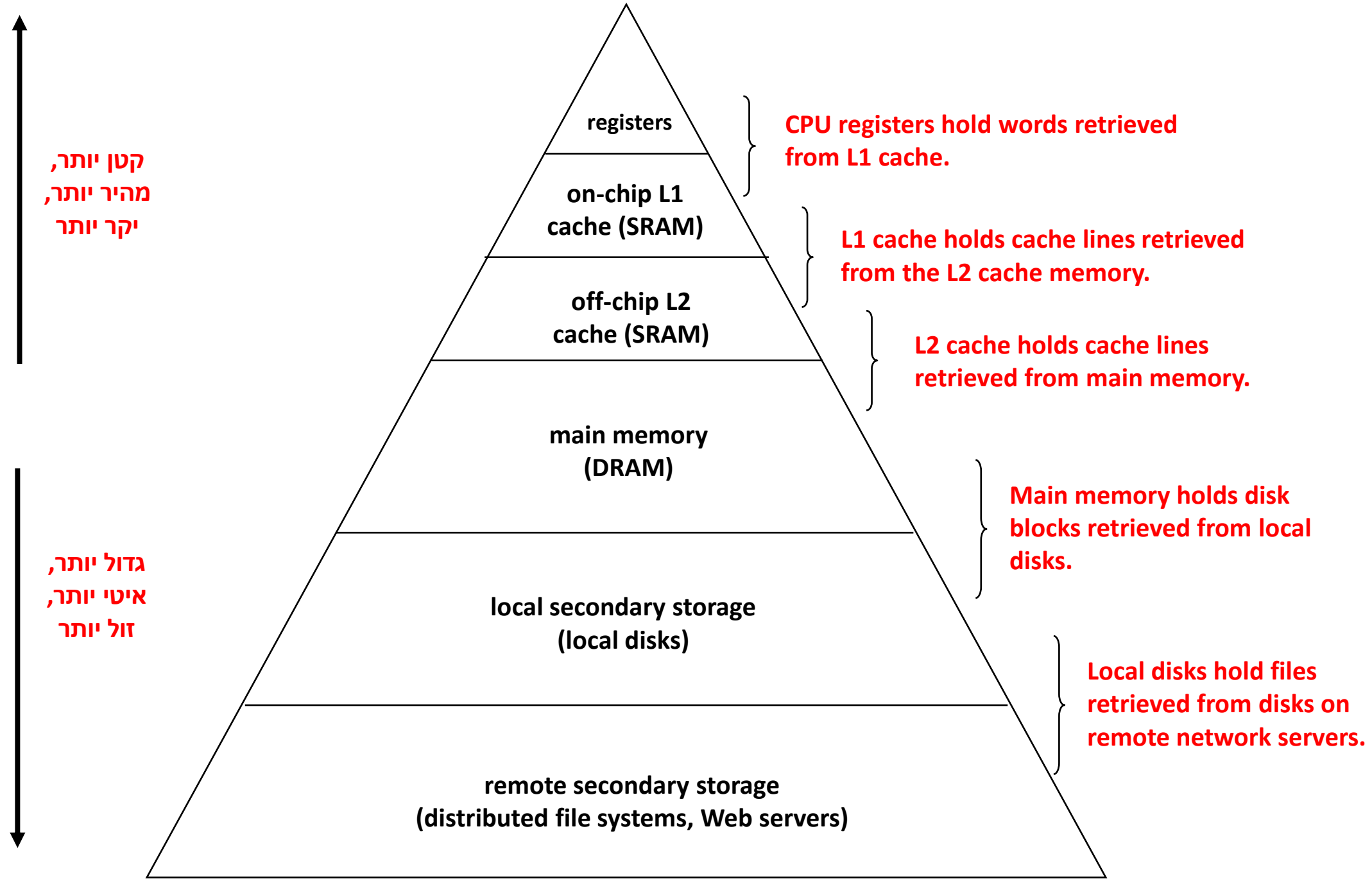
תרגול 9 – היררכיית הזיכרון

היררכיות זיכרון

אמרנו בהרצאה שישנן היררכיות לזיכרון- מה בעצם קורה?

- ככל שהזיכרון בעל נפח קטן יותר- הוא יעלה יותר מהר
- יש הפרשי קריאה / כתיבה בין המעבד לזיכרון. סקאלה לוגריתמית
- תכניות שכתובות טוב יותר- בעלות לוקאליות טובה יותר

מהסיבות האלה- ישנה היררכיה לזיכרון


קטן יותר,
מהיר יותר,
יקר יותר
גדול יותר,
איטי יותר,
זול יותר
registers
on-chip L1
cache (SRAM)
off-chip L2
cache (SRAM)
main memory
(DRAM)
local secondary storage
(local disks)
remote secondary storage
(distributed file systems, Web servers)
CPU registers hold words retrieved from L1 cache.
L1 cache holds cache lines retrieved from the L2 cache memory.
L2 cache holds cache lines retrieved from main memory.
Main memory holds disk blocks retrieved from local disks.
Local disks hold files retrieved from disks on remote network servers.

cache

<u>זיכרון מטמון (cache)</u>: זיכרון קטן ומהיר שמשמש "מבוא" לזיכרון גדול ואיטי יותר

הפעולה שמשתמשת ב-cache נקראת- caching.

מה הרעיון בעצם?

הרעיון המרכזי הוא שעבור כל k- התקן הזיכרון הקטן ביותר ברמה ה-k משמש כ-cache של התקן זיכרון ברמה ה-k+1 בהיררכיה.

← הזיכרון הגדול מאוחסן למטה ובזול- אבל ניגשים אליו בצורה מהירה יותר בזכות ה-cache

cache

נגיד ואנו רוצים לשלוף את האובייקט d מרמת הזיכרון k+1

נגדיר:

cache hit: התכנית מוצאת את d ברמה ה-k

cache miss: התכנית לא מוצאת את d ברמה ה-k ובתוצאה מכך צריך לרדת לרמה ה-k+1

מדד לביצוע של cache-

$$miss\ rate = \frac{\#cache\ misses}{total\ access\ to\ the\ memory}$$

Cache miss

יש 3 סוגים של cache miss:

1. Cold miss: ה cache ריק
2. Capacity miss: האובייקט אותו אנו רוצים להביא מהרמה k+1 גדול יותר מגודל ה cache ברמה ה k

Cache miss

יש 3 סוגים של cache miss:

3. Conflict miss: רק במקרים שבהם עובדים עם מיפוי של direct map.

   בתוצאה מכך, ייתכן שלמרות שיש מספיק מקום ב cache בשביל אובייקט d, עדיין נקבל miss.

   לדוגמא, נניח שכל בלוק i ברמה ה k+1 ממופה למקום ה i%4 ב cache.

   לכן בהנחה וה cache לא מכיל את בלוק 0 בהתחלה, הקריאות לבלוקים 0,8,0,8,0 יהיו תמיד miss.

cache locality

דיברנו על לוקאליות של תוכניות - תכניות נוטות לחזור ולהשתמש במידע שהשתמשו בו לאחרונה, או להשתמש במידע שפיזית נמצא ליד המידע הזה

<u>לוקאליות של זמן (temporal locality)</u>: מידע שהשתמשנו בו לאחרונה, סביר להניח שנשתמש בו שוב ← מעלים אותו ל-cache וזה חוסך זמן

<u>לוקאליות של מקום (spatial locality)</u>: קרוב לוודאי שעוד מעט נפנה למידע שנמצא קרוב למידע שהשתמשנו בו עכשיו ← שומרים ב-cache ואז זה יותר זמין

תרגיל 1

נתונה הפונקציה הבאה עם נתונים נוספים:

- גודל של float = 4 בתים
- Cache בגודל 32 בתים כאשר כל בלוק של 16 בתים.

  סך הכל 2 בלוקים.
- Sum ו- i נשמרים ברגיסטרים ולא דורשים זיכרון

```
float dotproduct(float x[8], float y[8])
 {
        float sum = 0.0;
        int i;
        for (i = 0; i < 8; i++)
                sum += x[i] * y[i];
        return sum;
}
```

1. כמה cache misses יהיו **בהרצה אופטימלית** של הפונקציה,

   בהנחה כי כל בלוק זיכרון יכול להיות ממופה לכל מקום ב-cache , ובהתחלה ה-cache ריק?
2. כמה cache misses יהיו בהרצה של הפונקציה, אם אנחנו יודעים ש-4 הערכים הראשונים של כל מערך

   ממופים לבלוק 0 ו-4 הערכים הבאים ממופים לבלוק 1, ובהתחלה ה-cache ריק?

```
float dotproduct(float x[8], float y[8])
 {
        float sum = 0.0;
        int i;
        for (i = 0; i < 8; i++)
                sum += x[i] * y[i];
        return sum;
}
```

תרגיל 1 – פתרון סעיף 1

<u>פתרון</u>: נקרא את הערך הראשון במערך x.

נספוג **cold miss** כי ה cache ריק בהתחלה. נזכור כי float תופס 4 בתים

וכל בלוק ב cache בגודל 16 בתים לכן יש מקום ל 4 איברי מערך בכל בלוק.

יחד עם האיבר הראשון ב x, נחלץ את 3 האיברים הבאים במערך ובכך נמלא את הבלוק הראשון.

נקרא את הערך הראשון במערך של y. נספוג **cold miss** כי הבלוק השני עדיין היה ריק.

לאחר חישוב 4 המכפלות של הזוגות נרצה להביא x[4]. נספוג **conflict miss** ונביא את יתר איברי x.

כנ"ל עבור איברי y – נספוג **conflict miss** ונביא את יתר איברי y.

בסך הכל יהיו 4 miss: 2 cold miss, 2 conflict miss

```
float dotproduct(float x[8], float y[8])
 {
        float sum = 0.0;
        int i;
        for (i = 0; i < 8; i++)
                sum += x[i] * y[i];
        return sum;
}
```

תרגיל 1 – פתרון סעיף 2

<u>פתרון</u>: נקרא את הערך הראשון במערך x.

נספוג **cold miss** כי ה cache ריק בהתחלה. יחד עם האיבר הראשון ב x, נחלץ עוד 3 איברים ובסך הכל 4 האיברים הראשונים יהיו בבלוק 0. כעת, נקרא את הערך הראשון במערך של y.

נספוג **conflict miss** כי הcache תפוס על ידי איברי x.

עד כה חישבנו מכפלה אחת. נרצה לקרוא את הערך x[1] ושוב נספוג **conflict miss** כי הcache תפוס על ידי איברי y.

נשים לב כי בכל איטרציה אנו סופגים 2 miss ומחשבים מכפלה אחת – **בסך הכל נקבל 16 miss**.

תרגיל 2

לקראת מסיבת הפקולטה, יו"ר ועד הסטודנטים הציע 4 אמנים שונים לפלייליסט,

והסטודנטים היו צריכים להצביע בשבילם. בכל פעם שסטודנט הצביע, בוצעה הדפסה של

נקודה לכל אומן.

נתון הקוד הבא:

```
struct point_artist
 {
        int Osher_Cohen;
        int Bad_Bunny;
        int Odeya;
        int Eyal_Golan;
}
```

```
struct point_artist  square [16] [16];
for ( int i = 0 ; i < 16 ; i++) {
        for ( int j = 0 ; j < 16 ; j++ ) {
                square[ i ][ j ].Osher_Cohen = 1;
                square[ i ][ j ].Bad_Bunny= 1;
                square[ i ][ j ]. Odeya= 1;
                square[ I ][ j ].Eyal_Golan = 1;
        }
}
```

```
struct point_artist  square [16] [16];
for ( int i = 0 ; i < 16 ; i++) {
        for ( int j = 0 ; j < 16 ; j++ ) {
                square[ i ][ j ].Osher_Cohen = 1;
                square[ i ][ j ].Bad_Bunny= 1;
                square[ i ][ j ].Odeya = 1;
                square[ I ][ j ].Eyal_Golan = 1;
        }
}
```

תרגיל 2

בנוסף נתון:

- הגודל של int הוא 4 בתים.
- square מתחיל בכתובת 0.
- ה-cache ריק בתחילת ריצת התכנית.
- הגישה היחידה לזיכרון היא למערך של square.
- ה-cache בגודל 2048 בתים, וכל בלוק בגודל 32 בתים – סך הכל 64 בלוקים.

1. כמה כתיבות לזיכרון התוכנית מבצעת?
2. כמה cache miss יהיו לנו במהלך הריצה?
3. חשבו את ה miss rate

תרגיל 2 - פתרון

```
struct point_artist  square [16] [16];
for ( int i = 0 ; i < 16 ; i++) {
        for ( int j = 0 ; j < 16 ; j++ ) {
                square[ i ][ j ].Osher_Cohen = 1;
                square[ i ][ j ].Bad_Bunny= 1;
                square[ i ][ j ].Odeya = 1;
                square[ I ][ j ].Eyal_Golan = 1;
        }
}
```

1. כמה כתיבות לזיכרון התוכנית מבצעת?
   תשובה: יש בסך הכל 16*16 איברים ב square.
   לכל איבר, שהוא struct, יש 4 איברים לכתוב בהם.
   בסך הכל 4*16*16 כתיבות לזיכרון.

2. כמה cache miss יהיו לנו במהלך הריצה?
   תשובה: אנו קוראים בלוקים בגודל 32 בתים. כל איבר פנימי ב struct הוא 4 בתים (int)
   לכן כל איבר ב square הוא 16 בתים. נקבל כי בכל בלוק יהיו 2 איברים של square.
   אם כך, כאשר נקרא את האיבר הפנימי הראשון באיבר הראשון של square נקבל
   cold miss ונביא לבלוק את יתר 7 האיברים הפנימיים (עוד 3 של האיבר הנוכחי ו4 של האיבר הבא).
   כלומר, נקבל miss כל 8 קריאות. בסך הכל $\frac{1}{8} \cdot (16^2 \cdot 4)$ miss.

3. ה miss rate הוא כמות ה miss חלקיי כמות הכתיבות לכן $\frac{1}{8}$.