אוניברסיטת חיפה
החוג למדעי המחשב
מערכות הפעלה – תרגול

תרגול מס' 4

Memory Management

1. הגדרות

1.1. Physical Memory

זהו הזיכרון של המחשב. במחשבים של היום ניתן למצוא 256-512MB.
כמות הזיכרון הפיזי שונה ממחשב למחשב ותלויה בלוח האם של המחשב. כיום, רוב הלוחות תומכים בזיכרון פיזי של עד 1.5GB.

1.2. Virtual Memory

זיכרון וירטואלי כשמו כן הוא – זיכרון אשר לא קיים פיזית במחשב ותלוי בנתונים של לוח האם. הזיכרון הווירטואלי גדול בהרבה מהזיכרון הפיזי (במחשבים של היום יכול להגיע ל-4GB).
תהליכים מסוימים צורכים כמויות זיכרון שונות. ישנן תוכנות הצורכות כמה מאות MB של זיכרון. לא בכל מחשב קיימת כמות הזיכרון הזאת ולכן הומצא הזיכרון הווירטואלי – אנחנו נותנים לתוכנית את ההרגשה שיש לה את כל הזיכרון שהיא צריכה, למרות שבפועל אין לנו את כל המשאבים לכך.
נשמע מסובך – נבין זאת טוב יותר בהמשך.
כיצד אנו יודעים כמה זיכרון וירטואלי יש לנו במערכת?
כמו שנאמר קודם, זה תלוי בחומרה במחשב. למשל, אם יש לנו לוח-אם עם פס של 32 סיביות, אזי כמות הזיכרון הווירטואלי = 2^{32} שזה 4GB.

1.3. Frame

הזיכרון הפיזי מחולק ליחידות שוות – גודל סטנדרטי של Frame הוא 4KB.

1.4. Page

הזיכרון הלוגי של כל תהליך מחולק ל-Pages. גודל דף הוא כגודל Frame במחשב.

1.5. Page Fault

נעשה ניסיון לפנות לכתובת וירטואלית אשר אינה מופיעה בזיכרון הפיזי של המחשב.

1.6. דוגמה

כדי להבין טוב יותר את ההגדרות שצוינו לעיל, הבה נסתכל על דוגמה פשוטה.

- נניח שהזיכרון הפיזי של המחשב הוא בגודל של 3 Frames (12KB) שנסמנם A,B,C.
- במחשב רץ תהליך אחד אשר גודל הזיכרון הווירטואלי שלו 32GB, אך הוא משתמש רק ב-4 Pages (שנמספר אותם 1,2,3,4).
- בתחילה פונה התהליך ל-Page 1 בזיכרון הווירטואלי. ה-Page נטען לזיכרון הפיזי ל-Frame A.
- לאחר מכן פונה התהליך ל-Page 4 שנטען לתוך B.
- בהמשך פונה ל-Page 2 שנטען לתוך C.
- כעת תתבצע פניה ל-Page 3. מאחר ואין לנו יותר מקום בזיכרון הפיזי, אחד מה-Pages (1, 2 או 4) יוצאו ובמקומם יוכנס Page 3.

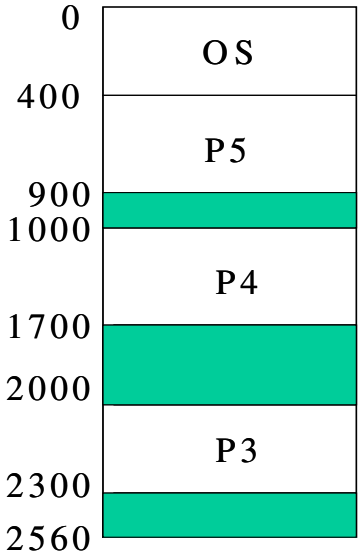
2. Fragmentation

ככל שיותר תהליכים נטענים ומורדים מהזיכרון, הזיכרון מתחלק לחלקים קטנים אשר אין בהם שימוש.

<u>דוגמא</u>:

Proc	Memory	Time
P1	600K	10
P2	1000K	5
P3	300K	20
P4	700K	8
P5	500K	15

--- **OS 400K** ---



נוצרו חלקים קטנים של זיכרון שלא מנוצלים !!!

2.1. External Fragmentation

מתקיים כאשר סה"כ הזיכרון מספיק גדול, אך לא ניתן לטעון תהליך כיוון שהזיכרון מפוצל.
אלטרנטיבה: הקצה חורים הגדולים מהשטח המבוקש, ע"מ למנוע היווצרות חורים קטנים. חסרון - יוצר בעיית **Internal Fragmentation** - זיכרון מבוזבז בתוך שטחים מוקצים.

אוניברסיטת חיפה

החוג למדעי המחשב

מערכות הפעלה – תרגול

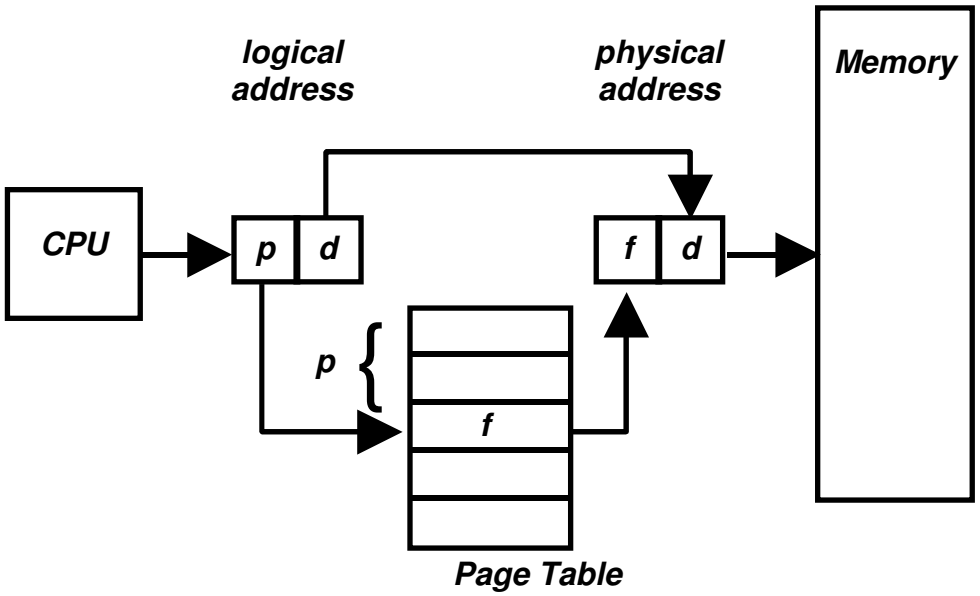
פתרון: compaction - אחוד חורים ע"י הזזת תהליכים. (פתרון יקר!)

פעם בכמה זמן או לפי דרישה ניתן "לדחוס" את כל הזיכרון כך שיישאר יותר זיכרון רציף פנוי.

למשל, בדוגמא הקודמת ישנם 660KB פנויים, אך מחולקים על פני 3 חלקים קטנים יותר.

מטרה: מניעת בעיית ה-Fragmentation ותמיכה בגודל זיכרון משתנה.

רעיון: נשתמש ב-Virtual Memory כפי שהוגדר קודם.



גודל הדף (בבתים) הוא חזקה של 2 - ע"מ להקל על המיפוי. בדרך כלל הוא 4KB.

כאשר משתמשים במנגנון ה-Paging, אין בעיית External Fragmentation, אך עדיין תיתכן בעיית Internal Fragmentation. נוכל להתגבר עליה ע"י הקטנת גודל ה-Page, אך נשלם בניהול מורכב יותר של הזיכרון.

דוגמא: האות F

כתובת לוגית 5 = 00101

דף # : 001

Offset : 01

תרגום :

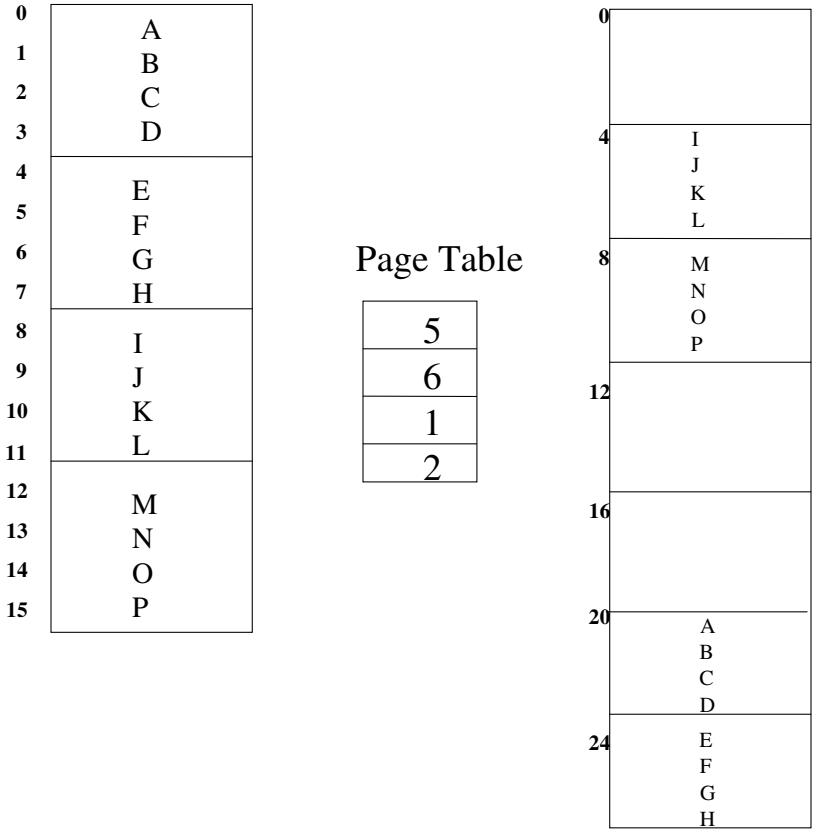
דף 1 מתרגם למסגרת 6

מסגרת # : 110

Offset : 01

תוצאת התרגום :

כתובת פיזית 25 = 11001



2.2. תכנון ה-Page Tables

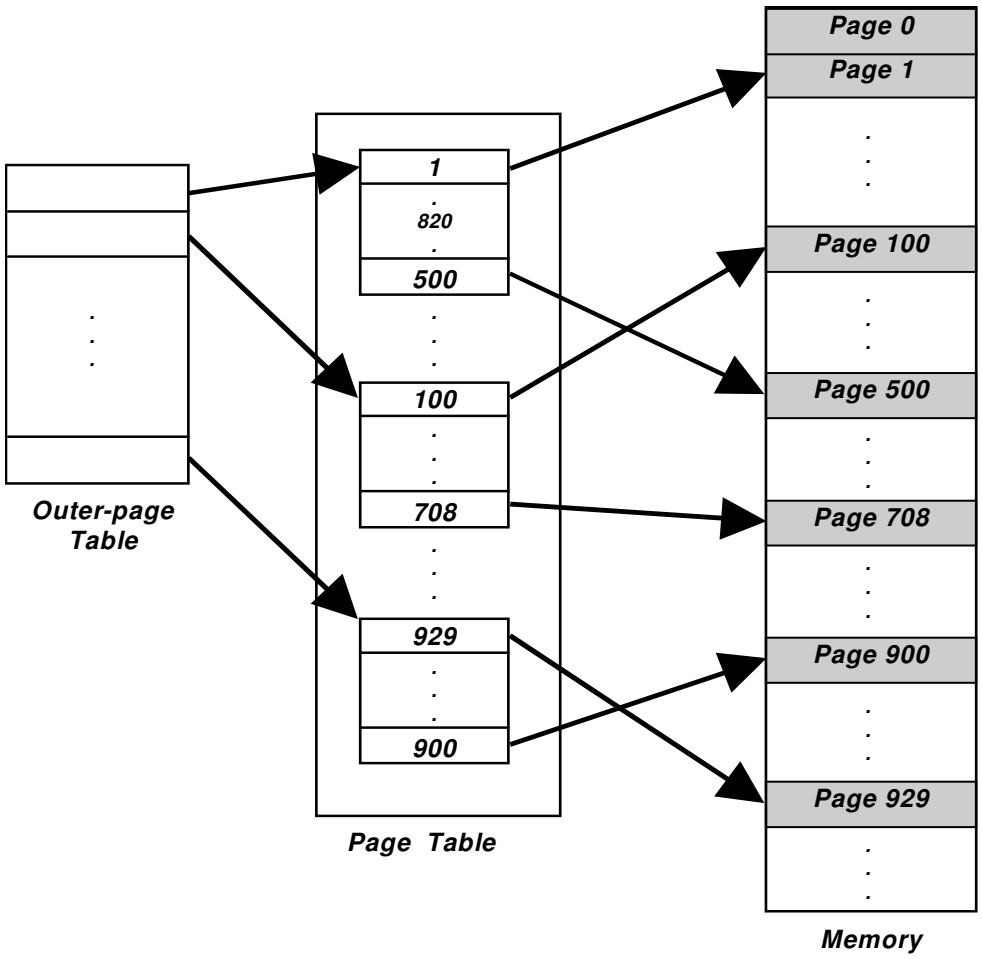
תפקיד טבלת הדפים למפות Logical Pages ל-Physical Frames.

בעיות:

- המיפוי חייב להיות מאד מהיר - כיון שנעשה עבור *כל* גישה לזיכרון
- טבלת הדפים יכולה להיות גדולה. מרחב זיכרון וירטואלי של 32 ביט עם דפים בגודל 4KB דורש כמיליון כניסות בטבלה !

2.3. Multi-Level Paging

לבנות טבלת paging "חיצונית" שתנהל גישות לטבלאות paging וליצור ע"י כך מספר רמות של indirection.



3. Translation Look-aside Buffer (TLB)

ה-TLB משמש כזיכרון מטמון של תרגומים. הוא מכיל חומרה שמחפשת במקביל עבור רשומות תרגומים קיימים. נשים לב שבהחלפת תהליכים, ה-TLB צריך להימחק!

ה-TLB מהיר יותר מזיכרון רגיל, והחיפוש בו נעשה במחיר של גישה אחת.

אחוז הפעמים שגישה לזיכרון נפתרת באמצעות ה-TLB נקרא **Hit Ratio**.

3.1. דוגמא לחישוב זמן הגישה הממוצע לזיכרון

TLB Search = 20 nanosecond

Memory Access = 100 nanosecond

מכאן עבור Hit Ratio = 80%:

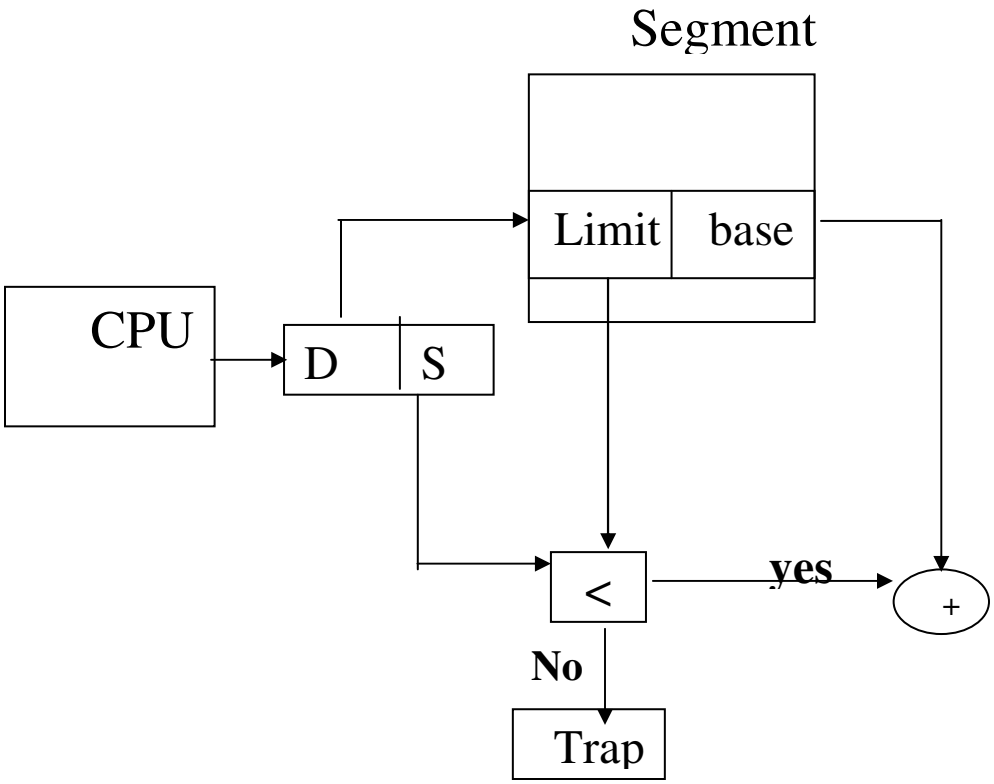
Effective Access Time = 0.8*120 + 0.2*220 = 140 nanosecond

אולם, עבור Hit Ratio = 98% :

Effective Access Time = 0.98*120 + 0.02*220 = 122 nanosecond

4. Segmentation

הרעיון הכללי הוא חלוקת הזיכרון לבלוקים המתאימים לוגית לתוכנית, כאשר גודל של כל בלוק זיכרון כזה משתנה. לכל סגמנט מוגדר מספרו הייחודי ואורך, והגישה בתוך הסגמנט נעשית ע"י הזוג (Segment Number : Offset inside the Segment).



בכל גישה לנתון בתוך הסגמנט נבדוק קיומם של 2 תנאים :

1. מספר הרגיסטר קטן ממספר הרגיסטרים הכולל שהוקצו לתהליך.
2. ה-Offset בתוך הסגמנט קטן מגודל הסגמנט.

הגישה תותר רק אם 2 התנאים מתקיימים.