**שאלה 4**

**לכל אחת משיטות ניהול הזיכרון הבאות, תאר האם תצפה למצוא:**

1. **Translation lookaside buffer (TLB)**
2. **Internal fragmentation**
3. **External fragmentation**

**שיטות ניהול הזיכרון הן:**

1. **Base and bound**
2. **Segmentation**
3. **Paging**
4. **Segmentation and paging**

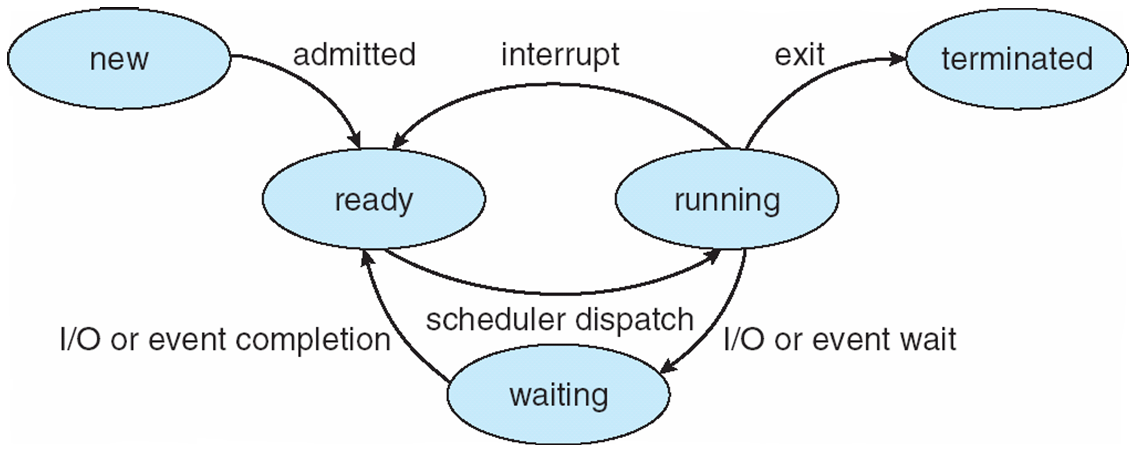
תשובה

|  |  |  |  |
| --- | --- | --- | --- |
| External fragmentation | Internal fragmentation | TLB |  |
| + | - | - | Base and bound |
| + | - | - (אם כי ניתן לחשוב על מקבילה) | Segmentation |
| - | + | + | Paging |
| - | + | + | Segmentation and paging |

שאלה 1

האם ייתכן ש- process שהיה במצב waiting יעבור למצב terminate? הסבר.

תשובה: בלתי אפשרי. אם ה- process חיכה למשהו אז ברגע שה- event קרה הרי שה- process היה צריך לחזור ולהריץ את הקוד שחיכה.



סמן נכון/לא נכון:

* Binary semaphore יכול לקבל ערכים נומריים 0 ו- 1 בלבד
* פעולה אטומית היא instruction או אוסף של instructions אשר חייב להתבצע בשלמותו ללא הפרעת interrupt.
* Deadlock הוא מצב בו שני תהליכים או יותר (או threads) ממתינים ל- event שצפוי לקרות מאוחר יותר
* Starvation הוא מצב בו נשללת מ- process הגישה למשאב לאור פעילות תחרותית של תהליך אחר
* כאשר process נמצא במצב blocked ב- semaphore queue, הוא עסוק ב- busy waiting. – לא נכון.
* Circular waiting הוא תנאי הכרחי אבל לא מספק ל- deadlock
* בניטרול שיקולי עלות, ניתן למנוע external fragmentation (כמעט לחלוטין) באמצעות שימוש תדיר ב- compaction
* Frame הוא חלק מה- main memory
* לא ניתן לבצע swap out לדיסק ל- Pages אותם חולקים שניים או יותר תהליכים
* Demand paging מחייב את התוכניתן לבצע פעילויות ייעודיות על-מנת לאכוף העלאת virtual page מבוקש
* Thrashing לעולם לא יהווה בעיה אם למערכת יש זיכרון פיזי של יותר מ- 1GB

סמן נכון/לא נכון:

* Binary semaphore יכול לקבל ערכים נומריים 0 ו- 1 בלבד - נכון
* פעולה אטומית היא instruction או אוסף של instructions אשר חייב להתבצע בשלמותו ללא הפרעת interrupt. – נכון
* Deadlock הוא מצב בו שני תהליכים או יותר (או threads) ממתינים ל- event שצפוי לקרות מאוחר יותר – לא נכון (deadlock זה כאשר ה- event לא יכול לקרות).
* Starvation הוא מצב בו נשללת מ- process הגישה למשאב לאור פעילות תחרותית של תהליך אחר - נכון
* כאשר process נמצא במצב blocked ב- semaphore queue, הוא עסוק ב- busy waiting. – לא נכון. כל הרעיון ב- semaphore queue הוא שהוא לא עושה busy waiting).
* Circular waiting הוא תנאי הכרחי אבל לא מספק ל- deadlock - נכון
* בניטרול שיקולי עלות, ניתן למנוע external fragmentation (כמעט לחלוטין) באמצעות שימוש תדיר ב- compaction - נכון
* Frame הוא חלק מה- main memory - נכון
* לא ניתן לבצע swap out לדיסק ל- Pages אותם חולקים שניים או יותר תהליכים – לא נכון (כי sharing לא בהכרח מחייב locking).
* Demand paging מחייב את התוכניתן לבצע פעילויות ייעודיות על-מנת לאכוף העלאת virtual page מבוקש – לא נכון
* Thrashing לעולם לא יהווה בעיה אם למערכת יש זיכרון פיזי של יותר מ- 1GB – לא נכון

הנח שהמערכת מצויה במצב הבא מבחינת הקצאת משאבים:



האם המערכת מצויה ב- deadlock state? במידה וכן, מהם התהליכים המצויים ב- deadlock? יש להראות את כל החישובים.

תשובה:

במצב הקיים ניתן להיענות לבקשתו של C בלבד. לאחר ש- C יסיים, כמות המשאבים available תהיה: [2,2,2,0]. במצב זה ניתן להיענות לבקשה של B, ולהאחר סיום תהליך B נקבל [4,2,2,1]. כעת ניתן להקצות את המשאבים הנדרשים ל- A. לסיכום, המערכת לא מצויה במצב של deadlock.

שאלה 6 (6 נקודות)

משה הוא ארכיטקט המערכת בחברת UltimateSystems המפתחת מערכת הפעלה חדשה למערכות מרובות מעבדים. על-מנת שלא להיתפס לקריטריון שיבוץ ספציפי, ובהשראת תורת המשחקים, מחליט משה שכל תהליך הממתין ב- ready queue יקבל עם כניסתו למערכת אסימון וירטואלי ובכל 10 מילישניות שהוא ממתין ב- ready queue אסימון נוסף (ללא מגבלה על כמות האסימונים). תהליכים ישובצו על המעבדים השונים לפרק זמן K (שייקבע על-ידי ה- system admin) ובעת שמתפנה אחד המעבדים מבקש ה- scheduler מכל אחד מהתהליכים להודיע (באמצעות מנגנון shared memory או message passing) כמה אסימונים הוא מוכן ל"שלם" תמורת שיבוץ כעת. קביעת כמות האסימונים אשר יציע תהליך תיעשה באמצעות קוד שיהיה חלק מהתוכנית שיצרה אותו, אשר התוכניתן יכול לעשות לו override . אסימונים שלא נוצלו עד השיבוץ יישמרו אצל התהליך וניתן יהיה לעשות בהם שימוש בפעם הבאה שייכנס ל- ready queue.

מהי הבעיה המרכזית של המנגנון המוצע בהקשר לעקרונות מערכות הפעלה שלמדת בקורס?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

תשובה

יש כאן המון overhead של תהליך ה- scheduling ולכן המנגנון לא יעיל.

שאלה 7 (8 נקודות)

כיצד פועל מנגנון Inverted Page Table וכיצד מתורגמת הכתובת הלוגית לפיסית באמצעותו (מומלץ לצרף להסבר סכימה מתאימה)? מהם היתרונות והחסרונות העיקריים של השיטה?

אופן פעולת השיטה:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

יתרונות:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

חסרונות:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

תרשים:

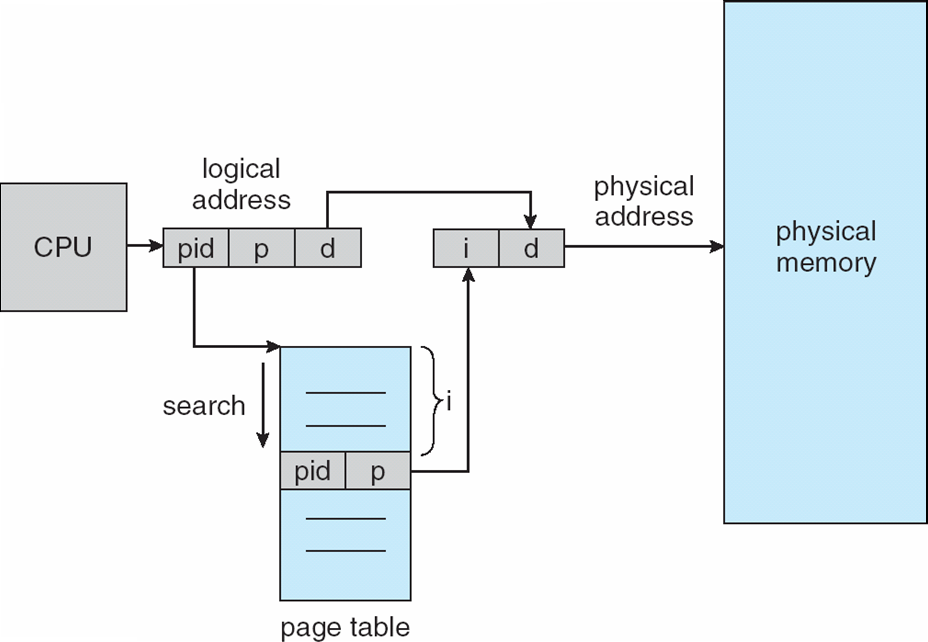
תשובה

Single page table, takes up less space.

Searching inverted table very expensive; hashing can help, but this often entails

specialized hashing hardware. Inverted table scheme makes sharing pages between

address spaces more difficult.



שאלה 8 (5 נקודות)

מדוע גודל ה- page הוא בחזקות של 2?

תשובה

**Answer:** Recall that paging is implemented by breaking up an address into a page and

offset number. It is most efficient to break the address into X page bits and Y offset bits,

rather than perform arithmetic on the address to calculate the page number and offset.

Because each bit position represents a power of 2, splitting an address between bits results

in a page size that is a power of 2.

שאלה 1 (18 נקודות)

1. עבור אלו פעולות כדאי להשתמש ב- DMA? נמק והסבר?
2. מהם שלושת התפקידים העיקריים של מערכת ההפעלה בכל הקשור ל- secondary-storage management?
3. מערכת MS-DOS אינה מספקת סביבה ל- concurrent processing. דון בשלושה סיבוכים עיקריים שמוסיף ה- concurrent processing למערכת הפעלה.

**Answer:** DMA is useful for transferring large quantities of data between memory and

devices. It eliminates the need for the CPU to be involved in the transfer, allowing the

transfer to complete more quickly and the CPU to perform other tasks concurrently.

**Answer:**

* \_ Free-space management.
* \_ Storage allocation.
* \_ Disk scheduling.

1. **Answer:**
2. \_ A method of time sharing must be implemented to allow each of several processes to have access to the system. This method involves the preemption of processes that do not voluntarily give up the CPU (by using a system call, for instance) and the kernel being reentrant (so more than one process may be executing kernel code concurrently).
3. Processes and system resources must have protections and must be protected from each other. Any given process must be limited in the amount of memory it can use and the operations it can perform on devices like disks.
4. Care must be taken in the kernel to prevent deadlocks between processes, so processesyaren’t waiting for each other’s allocated resources.

שאלה 6 (15 נקודות)

נתונה מערכת ניהול קבצים בדיסק בה גודל הבלוק הוא 512 bytes. הנח כי המידע לגבי כל קובץ (רשומת ה- fcb) מצוי כבר בזיכרון. ענה על השאלות הבאות ביחס לכל אחת משלושת שיטות ההקצאה שנלמדו בכיתה (contiguous, indexed, linked):

1. כיצד מתבצע המיפוי מכתובת לוגית (ה- byte ה- W של הקובץ) לכתובת פיזית (בלוק פיזי ומיקום בבלוק). עבור indexed יש להניח כי גודל הקובץ תמיד קטן מ- 64 בלוקים וכי הבלוק הפיזי הראשון של הקובץ הוא Z.
2. במידה וכרגע קראנו את הבלוק הלוגי העשירי ואנו מעוניינים בנתון מהבלוק הלוגי הרביעי של הקובץ, כמה בלוקים פיזיים נצטרך לקרוא מהדיסק?