Grundlagen der Systemsoftware

Übungsblatt 6 \* Gruppe G02-A \* Back, Behrendt, Stäger \* SoSe 2015

# Aufgabe 1: Speicherverwaltung (14 Punkte)

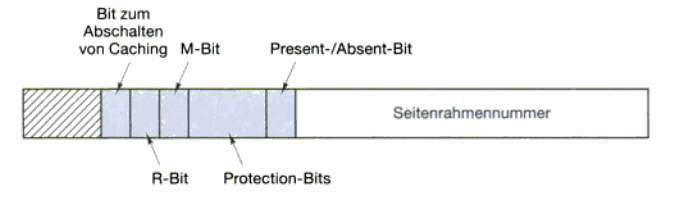
Es wird das Innenleben einer Memory Management Unit (MMU) mit Seitenadressierung betrachtet, wobei eine Seitentabelle mit 16 Seiten verwendet wird, virtuelle Adressen sind 16 Bit, die physikalischen Adressen sind 15 Bit lang und die Wortlänge sei 1 Byte.

**a) Wie groß ist eine Seite?**

Wir wissen aus der Aufgabenstellung, dass es 16 Seiten gibt. Außerdem wissen wir, dass eine virtuelle Adresse 16 Bit lang ist, wobei diese 16 Bit aufgeteilt werden um die Seitenzahl und den Offset auf dieser Seite anzugeben.

Um 16 Seiten adressieren zu können, brauchen wir die 4 höherwertigen Bits von den 16 Bit der virtuellen Adresse, da . Somit bilden die restlichen 12 Bit der virtuellen Adresse den Offset, was bedeutet, dass es Werte für den Offset gibt. Hieraus ergibt sich automatisch die Seitengröße. Wir können mit Hilfe des 12-Bit-langen Offsets 4096 Bits innerhalb der Seite Adressieren. Die Seitengröße beträgt also 4096 Bit.

**b) (optional) Woraus besteht ein typischer Seitentabelleneintrag und welche Bedeutung hat insbesondere das Present/Absent-Bit? Bei höchstens wie vielen Einträgen der Seitentabelle kann im gegebenen Fall das Present/Absent-Bit auf 1 („Present“) gesetzt sein?**



* Seitenrahmennummer: physikalische Adresse des Rahmens, der den Inhalt der entsprechenden Seite speichert
* Present/Absent-Bit: gibt an, ob die Seite momentan im Speicher verfügbar ist
* Protection-Bits: gibt an, welche Form des Zugriffs auf diese Seite erlaubt ist (Lesen/Schreiben)
* Modified-Bit (M-Bit) und Referenced-Bit (R-Bit): gibt an, ob der Rahmeninhalt verändert wurde, nachdem er von der Platte gelesen wurde
* R-Bit: wird beim Zugriff auf eine Seite gesetzt. Hilft dem Betriebssystem zu entscheiden, welche Seite ausgelagert werden soll

**c) Gegeben sei ein Ausschnitt der Seitentabelle zu einem Zeitpunkt (vgl. Abbildung rechts). Welche physikalische Adresse ergibt sich für folgende virtuelle Adressen?**

Da wir aus Teil a) wissen, dass die 4 höherwertigen Bits der Seitennummer entsprechen, müssen wir hier immer nur den ersten Hexadezimal-Wert betrachten, um den entsprechenden Eintrag in der Tabelle zu finden.

|  |  |  |  |
| --- | --- | --- | --- |
| Virtuelle Adresse | Eintrag in der Tabelle | physikalische Adresse | Offset |
| 0x5fe8 | 0x5 = 5 | 001 | 0xfe8 |
| 0xfeee | 0xf =15 | Seitenfehler (Absent) | 0xeee |
| 0xa470 | 0xa = 10 | 000 | 0x470 |
| 0x0101 | 0x0 = 0 | 101 | 0x101 |

**d) Um eine möglichst optimale Seitengröße zu wählen, muss man zwischen mehreren zum Teil gegenläufigen Faktoren abwägen. Bitte nennen Sie mindestens 2 Faktoren, die für eher kleine Seiten sprechen, und mindestens einen Grund, welcher für eher große Seiten spricht.**

Für kleine Seiten

* Ein gewähltes Daten-Segment wird selten eine ganze Zahl von Seiten füllen. Im Durchschnitt ist die letzte Seite immer nur halb gefüllt, es wird also Speicher verschwendet. Dieser Verschnitt wird auch als interne Fragmentierung bezeichnet.
* Auch Programme, die den Speicher verwenden, blockieren mindestens immer eine komplette Seite. Ist diese Seite z.B. 32 Byte groß, das Programm benötigt jedoch nur 4 Byte, belegt es trotzdem 32 Byte des Speichers, da eine Seite nicht von mehreren Programmen verwendet werden kann.

Für große Seiten

* Je kleiner die Seite, desto mehr Seiten werden benötigt, desto größer wird die Seitentabelle.
* Ladezeiten für eine Seite werden nur geringfügig durch die Größe der Seite definiert, eher durch Suchzeit und Rotationsverzögerung. Viele Seitensprünge (weil die Seiten zu klein sind) dauern also deutlich länger als eine große Seite zu laden.

**e) Sei die durchschnittliche Prozessgröße im Speicher und die Länge eines Seitentabelleneintrags. Erläutern Sie kurz den Zusammenhang zwischen der Seitentabellengröße im Speicher und der internen Fragmentierung bei verschiedenen Seitengrößen und ermitteln Sie die optimale Seitengröße in KiB unter der Annahme, dass im Durchschnitt die Hälfte der letzten Seite eines Prozesses leer bleibt. Geben Sie dabei den vollständigen Rechenweg an und erklären Sie Ihre Annahmen.**

Die Seitentabellengröße ist abhängig von der Anzahl der Seiten, die die Tabelle referenzieren muss. Sind die Seiten an sich sehr klein, müssen mehrere Seiten von der Seitentabelle verwaltet werden, somit steigt die Größe der Tabelle.

Tanenbaum behauptet, es gäbe keine optimale Seitengröße, er gibt jedoch für bekannte Prozessgrößen eine Formel an:

Diese Formel ergibt sich aus der Ableitung der Speicherverbrauchsformel für einen Prozess:

Wobei die Anzahl der Bytes in der Seitentabelle und den Verlust durch interne Fragmentierung angibt. Daraus ergibt sich:

# Aufgabe 2: Seitenersetzungsalgorithmen (14 Punkte)

**a) Gegeben sei ein Seitenspeicher der Größe 3. Zum Zeitpunkt *t* = 0 sei der Speicher leer. Illustrieren Sie (z. B. anhand von Tabelle 1) den Zustand des Seitenspeichers bei Verwendung des (a) optimalen Seitenersetzungsalgorithmus und (b) LRU-Algorithmus (Least Recently Used) zu den Zeitpunkten 1 bis 15 für die Referenzkette 1, 2, 3, 4, 5, 6, 1, 3, 1, 6, 3, 5, 4, 2, 1. Markieren Sie die Spalten, in denen ein Seitenalarm (engl. page fault ) auftritt.**

optimalen Seitenersetzungsalgorithmus:

Der optimaleSeitenersetzungsalgorithmus wählt unter allen derzeit in einem Seitenrahmen eingelagerten virtuellen Seiten diejenige Seite zur Ersetzung aus, welche in Zukunft am längsten nicht mehr benötigt wird.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| t | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| Seite | 1 | 2 | 3 | 4 | 5 | 6 | 1 | 3 | 1 | 6 | 3 | 5 | 4 | 2 | 1 |
| Fault | x | x | x | x | x | x |  |  |  |  |  | x | x | x |  |
| Seiten im Speicher | 1 | 1,2 | 1,3,2 | 1,3,4 | 1,3,5 | 1,3,6 | 3,1,6 | 1,6,3 | 6,3,1 | 3,1,6 | 1,6,3 | 1,5,3 | 1,4,5 | 1,2,4 | 1,2,4 |

LRU-Algorithmus (Least Recently Used):

Der LRU-Algorithmus wählt unter allen derzeit in einem Seitenrahmen eingelagerten virtuellen Seiten diejenige Seite zur Ersetzung aus, welche am längsten nicht verwendet wurde.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| t | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| Seite | 1 | 2 | 3 | 4 | 5 | 6 | 1 | 3 | 1 | 6 | 3 | 5 | 4 | 2 | 1 |
| Fault | x | x | x | x | x | x | x | x | x | x |  | x | x | x | x |
| Seiten im Speicher | 1 | 2,1 | 3,2,1 | 4,3,2 | 5,4,3 | 6,5,4 | 7,6,5 | 3,7,6 | 1,3,7 | 6,1,3 | 3,6,1 | 5,3,6 | 4,5,3 | 2,4,5 | 1,2,4 |

**b) (optional) Der optimale Seitenersetzungsalgorithmus wird gerne für theoretische Betrachtungen herangezogen. Warum kommt er nicht in realen Betriebssystemen zum Einsatz (max. 3 Sätze)?**

Der optimale Seitenersetungsalgorithmus geht davon aus, dass bereits zum Berechnungszeitpunkt t die zukünftigen Seitenzugriffe bekannt sind. Dies kann jedoch unmöglich der Fall sein, auch die MMU nicht in die Zukunft sehen kann. Somit ist der optimale Seitenersetungsalgorithmus ein theoretischer Algorithmus.

c) (optional) Ergibt LRU als Seitenersetzungstrategie Sinn, sofern das L in LRU für Last und nicht, wie korrekterweise, für Least stehen würde?  
Der Name des Algorithmus besagt, welche Seite der Algorithmus als nächstes entfernt. Least Recently Used bedeutet in diesem Kontext die zuletzt am wenigsten genutzte Seite. Würde der Algorithmus Last Recently Used heißen, dann würde der Algorithmus immer die zuletzt genutzte Seite (die neuste) entfernen, was eher nicht gewünscht ist.

# Aufgabe 3: Synchronisation (12 Punkte)

**Der Gebrauch von Semaphoren kann anhand der Lösung des klassischen Reader-Writer-Problems anschaulich demonstriert werden. Bei dem Reader-Writer-Problem gibt es zwei verschiedene Klassen, die beide auf einen gemeinsamen Datenbereich zugreifen. Instanzen der Klasse Writer schreiben in ihrer Hauptmethode (processWriter()) in den gemeinsamen Datenbereich und benötigen daher den exklusiven Zugriff. Instanzen der Klasse Reader hingegen greifen in ihrer Hauptmethode (processReader()) nur lesend auf den Datenbereich zu und können daher auch parallel auf dem gemeinsamen Datenbereich agieren.**

**a) Entwerfen Sie eine Lösung für das Reader-Writer-Problem, indem Sie die Hauptmethoden (processWriter() und processReader()) der beiden Klassen in Pseudocode angeben. Der Zugriff auf den gemeinsamen Datenbereich erfolgt durch die Methoden readData() (im Falle eines Readers) und durch die Methode writeData() (im Falle eines Writers). Verwenden Sie in Ihrer Lösung folgende Hilfsvariablen und geben Sie deren Initialisierungswerte mit an:**

* **ein binäres Semaphor W, um den kritischen Anschnitt zu schützen, in dem die eigentlichen Datenzugriffe (durch writeData() bzw. readData()) erfolgen**
* **eine Zähler-Variable NumberOfActiveReaders, die die Anzahl der aktiven Reader angibt, die gerade auf den gemeinsamen Datenbereich zugreifen**
* **ein binäres Semaphor Mutex, welches dafür sorgt, dass die Zähler-Variable NumberOfActiveReaders nur durch einen Reader zur Zeit modifiziert werden kann**

|  |
| --- |
| global boolean LOCKED = FALSE;  global integer NumberOfActiveReaders = 0;  global boolean NumberOfActiveReaders\_MUTEX = FALSE; |
| class Reader {  public function processReader() {  //warten auf Freigabe  while(LOCKED == TRUE || NumberOfActiveReaders\_MUTEX == TRUE) {}  //NumberOfActiveReaders updaten  NumberOfActiveReaders\_MUTEX = TRUE;  NumberOfActiveReaders++;  NumberOfActiveReaders\_MUTEX = FALSE;  //Daten lesen  readData();    while(NumberOfActiveReaders\_MUTEX == TRUE) {}  //NumberOfActiveReaders updaten  NumberOfActiveReaders\_MUTEX = TRUE;  NumberOfActiveReaders--;  NumberOfActiveReaders\_MUTEX = FALSE;  }  } |
| class Writer {  public function processWriter() {  //warten auf Freigabe  while(1) {  if(LOCKED == FALSE && NumberOfActiveReaders\_MUTEX == FALSE) {  if(NumberOfActiveReaders == 0) {  //Freigabe erhalten und kein Reader ließt mehr    //Daten schreiben  LOCKED = TRUE;  writeData();  LOCKED = FALSE;    return;  }  }  }  }  } |

**b) Werden bei Ihrer Lösung bestimmte Prozesse benachteiligt? Wenn ja, deuten Sie bitte kurz (max. 5 Sätze) eine nöglichen Lösungsweg an.**  
Diese Lösung benachteiligt den Writer. Sobald der Writer schreiben möchte, muss er warten bis alle Reader fertig sind mit dem Lesen und auch auf alle neu dazukommenden Reader. Zur Lösung müsste es den Writern möglich sein, sich anzukündigen, sodass die Reader am Anfang einmal warten müssen, bis der Writer fertig ist. Dies bevorzugt wiederum die Writer. Eine andere Möglichkeit besteht darin, dass der Writer eine Kopie von den Werten zu Beginn anfertigt, die die Reader dann immer lesen können. Dann müssen die Reader nicht warten und bekommen immer Daten.