2 Speicherverwaltung (25)

a) Lokalität von Prozessen – Working Sets (11)

Das Working Set Modell beschreibt das Speicherzugriffsverhalten von Prozessen. In diesem Modell ist das Working Set $W(t, \Delta)$ definiert als die Menge der Speicherseiten, die in den letzten Δ Zeiteinheiten vor dem Zeitpunkt t referenziert wurden.

Die folgende Tabelle beschreibt eine Folge von Speicherseitenzugriffen eines Prozesses. In Spalte 2 jeder Zeile steht die Nummer der Seite, auf die in der Zeitscheibe $(t-1, t]$ zugegriffen wird. Geben Sie in den leeren Feldern die Working Sets $W(t, \Delta)$ für Δ gleich 2, bzw. 5 an.

t	*Seitennr.*	$W(t,2)$	$W(t,5)$
1	29		
2	12		
3	13		
4	22		
5	29		
6	13		
7	12		
8	45		
9	29		
10	12		
11	12		

b) Kombination aus Segmentierung und Paging (14)

Es werden folgende Begriffe (englische Notation) aus dem Buch zur Vorlesung verwendet:

Base	Basisadresse Seitentabelle des Segmentes
Length	Länge des Segmentes (Anzahl der Seiten des Segmentes)
Virt.Addr.	Virtuelle Adresse
Frame#	Seitenrahmennummer (im physischen Speicher)
Page#	Seitennummer (im virtuellen Speicher)
Seg#	Segmentnummer

Das in der Folge betrachtete Speicherverwaltungssystem verwendet zur Adressierung 20-bit Adressen. Für das Paging sind alle Seitenrahmen 256 Bytes (sedezimal (hexadezimal) `0x00100`) groß. Das verwendete Adressformat ist folgendes:

Seg# (8 bit)	Page# (4 bit)	Offset (8 bit)

Hierbei wird assoziativer Zugriff (associative mapping) auf die Segmenttabelle und direkter Zugriff (direct mapping) auf die Seitentabelle verwendet.

Verwenden Sie für die Adressumsetzung folgende Segmenttabelle und Seitentabelle (alle Werte sind als Sedezimalzahlen (Hexadezimalzahlen) angegeben):

Segmenttabelle		
Seg#	Base	Length
0x03	0x0AFFE	0x3
0xAC	0x7D00F	0x1
0x3D	0x1CE00	0x7
0x00	0x00000	0xD

Seitentabelle	
Address	Frame#
0x00000	0x123
0x00001	0x124
0x00002	0x376
...	...
0x0AFFE	0xEEE
0x0AFFF	0xABC
0x0B000	0x000
0x0B001	0xDDD
...	...
0x1CDFF	0x666
0x1CE00	0x7DA
...	...
0x1CE05	0x5A7
0x1CE06	0x5AC
...	...
0x7D00E	0x471
0x7D00F	0xAAA
0x7D010	0x815
...	...

Ermitteln Sie unter Benützung obiger Tabellen die physikalischen Adressen zu folgenden virtuellen Adressen (ergibt sich bei der Umwandlung eine ungültige Adresse, so schreiben Sie bitte **ungültig** in das entsprechende Feld):

Virtuelle Adresse	Physikalische Adresse (zu ermitteln)
0xAC2FF	
0x01111	
0x002FF	
0x3D6D7	
0xFF000	
0x0328A	
0x00D33	

4 Speicherverwaltung (25)

a) Segmentierung (6)

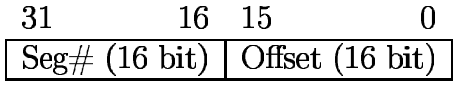
Das im folgenden beschriebene Speicherverwaltungssystem verwendet zur Adressierung 32-bit Adressen (virtuell & physikalisch). Für die angegebenen virtuellen Speicheradressen sind die entsprechenden physikalischen Adressen zu ermitteln. Von den angebenen Adressen sind die niederwertigen 16 Bit der Offset der Adresse. Die Verwendung der höherwertigen 16 Bit ist aus dem angegebenen Adressformat ersichtlich.

Alle Werte sind als Hexadezimalzahlen angegeben. Ergibt sich bei der Umwandlung aufgrund einer ungültigen Segmentnummer eine ungültige Adresse, so schreiben Sie bitte **"ungültig/Nummer"** in das entsprechende Feld. Sollte sich aufgrund eines unzulässigen Offsets eine ungültige Adresse ergeben, so schreiben Sie bitte **"ungültig/Offset"** in das entsprechende Feld.

Der Zugriff auf die Segmenttabellle erfolgt assoziativ (associative mapping). Es werden folgende Begriffe (englische Notation) aus dem Buch zur Vorlesung verwendet:

Base	Basisadresse des Segmentes	Virt.Addr.	Virtuelle Adresse
Length	Länge des Segmentes	Seg#	Segmentnummer

Das verwendete Adressformat ist folgendes:



Verwenden Sie für die Adressumsetzung folgende Segmenttabelle:

Segmenttabelle		
Seg#	Base	Length
0xFFFF	0x15FF B000	0xCCDD
0x1122	0x0808 B000	0x1001
0x1234	0x1266 0000	0x8000

Ermitteln Sie unter Benützung obiger Segmenttabelle die physikalischen Adressen zu folgenden virtuellen Adressen:

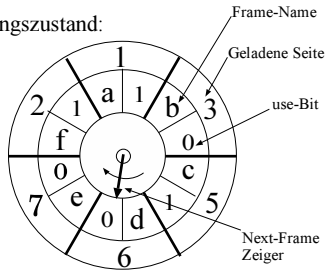
Virtuelle Adresse	Physikalische Adresse (zu ermitteln)
0xFFF1 3201	
0x1234 8000	
0x1234 8001	
0x1234 2001	
0x1121 1003	
0xFFFF 1503	

b) Replacement Policy (14)

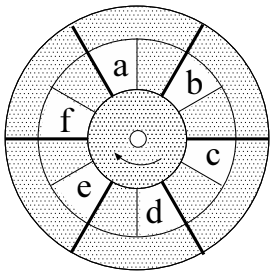
Simulieren Sie das Verhalten einer *clock policy* zum Auslagern von Speicher-Seiten, wenn neue Seiten geladen werden müssen.

Für jede Seite im Speicher existiert ein *use*-Bit. Die einzelnen Plätze im Hauptspeicher sind in diesem Beispiel mit Buchstaben von `a ... f` benannt. Der *next frame*-Zeiger bewegt sich in den verwendeten Grafiken im Uhrzeigersinn weiter.

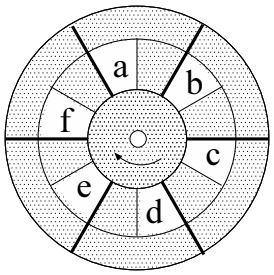
Anfangszustand:



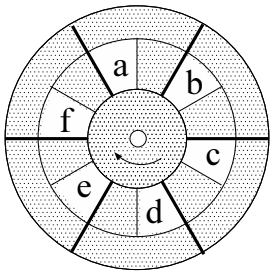
1.Zugriff: `access[1]`



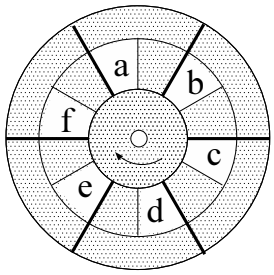
2.Zugriff: `access[8]`



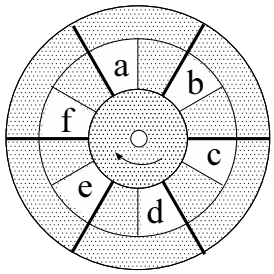
3.Zugriff: `access[3]`



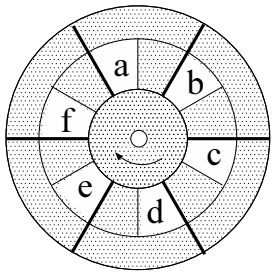
4.Zugriff: `access[6]`



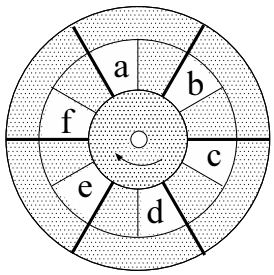
5.Zugriff: `access[7]`



6.Zugriff: `access[3]`



7.Zugriff: `access[9]`



Zeichnen Sie nun in den obigen Grafiken

- die Position des *next frame*-Zeigers
- die Inhalte aller Frames (geladene Seiten)

für die Ausführung von Befehlen mit Speicherzugriffen ein. Ein Befehl `access[x]` steht für einen Speicherzugriff auf die Seite `x`. Geben Sie den gefragten Speicherzustand **nach** der Ausführung des jeweiligen Befehls an, wobei Sie für die Befehle von einer sequentiellen Reihenfolge ausgehen können.

c) (3)

Nennen Sie drei Szenarien für den fehlerfreien Fall, unter denen Programme von nicht privilegierten Benutzern die gleiche physikalische Speicheradresse benutzen:

1.

2.

3.

d) (2)

Um wieviel Prozent **vergrößert** (nicht vervielfacht!) sich der virtuelle Adressraum, wenn Sie statt 32 bit langen 34 bit lange virtuelle Adressen verwenden?

Der virtuelle Adressraum vergrößert sich um %.

3 Speicherverwaltung (25)

a) Translation Lookaside Buffer (15)

Das gegebene Speicherverwaltungssystem basiert auf der Paging-Technik und verwendet einen *Translation Lookaside Buffer (TLB)* mit drei Einträgen. Der TLB wird mit der *Least Recently Used* Strategie verwaltet, d.h., wenn für eine neu einzutragende Seite kein Eintrag mehr frei ist, wird jener Eintrag ersetzt, der am längsten nicht benutzt wurde. Zu diesem Zweck enthält der TLB für jeden Eintrag ein Feld, das angibt seit wievielen TLB Zugriffen der entsprechende Eintrag nicht benutzt wurde. Bei jedem Zugriff werden die Zähler jener Einträge erhöht, die nicht gebraucht wurden, der Zähler des verwendeten oder neuen Eintrags wird auf 0 gesetzt.

Der TLB und die benötigte Page Table werden mittels **Associative Mapping** angesprochen. Die Pagegröße beträgt 4096 Bytes. Die Page-Table enthält für jeden Eintrag ein Feld *loaded*, das anzeigt, ob sich die entsprechende Seite im Hauptspeicher befindet.

Eine virtuelle Speicherdresse hat folgendes Format:

Page#(4 Bit)	Offset (12 Bit)

Eine physikalische Speicherdresse hat folgendes Format:

Frame#(8 Bit)	Offset (12 Bit)

Die Speicheradressen, Frame- und Pagenummern sind im Hexadezimalsystem (16er-System) angegeben.

- Auf dieser Seite ist ein Translation Lookaside Buffer und eine Page Table vorgegeben. Simulieren Sie ausgehend von diesen Daten den hintereinanderfolgenden Zugriff auf die virtuellen Speicheradressen auf der nächsten Seite.
- Befindet sich eine Seite nicht im Hauptspeicher, so können Sie die Nummer des Page-Frames für die Page aus den nicht belegten frei wählen. Tragen Sie diese in der Page Table auf dieser Seite ein und markieren Sie das entsprechende *Loaded*-Feld.
- Bestimmen Sie, ob es zu einem TLB Hit oder Miss kommt und ob es zu einem Main Memory Page Hit kommt oder nicht (Kreuzen Sie entsprechend **Ja** oder **Nein** an).
- Geben Sie weiters jeweils den Inhalt des TLB **nach** dem Zugriff auf die Page an.

Ausgangssituation

Translation Lookaside Buffer

Page#	Frame#	Last Use
2	0x47	0
4	0x00	1

Page Table

Page#	Loaded	Frame#
0	☒	0x2F
1	☐	
2	☒	0x47
3	☐	
4	☒	0x00

Virtuelle Adresse

0x0313

Physikalische Adresse

TLB Hit ◯ Ja ◯ Nein
Page Hit ◯ Ja ◯ Nein

Translation Lookaside Buffer

Page#	Frame#	Last Use

Virtuelle Adresse

0x042F

Physikalische Adresse

TLB Hit ◯ Ja ◯ Nein
Page Hit ◯ Ja ◯ Nein

Translation Lookaside Buffer

Page#	Frame#	Last Use

Virtuelle Adresse

0x1141

Physikalische Adresse

TLB Hit ◯ Ja ◯ Nein
Page Hit ◯ Ja ◯ Nein

Translation Lookaside Buffer

Page#	Frame#	Last Use

Virtuelle Adresse

0x4711

Physikalische Adresse

TLB Hit ◯ Ja ◯ Nein
Page Hit ◯ Ja ◯ Nein

Translation Lookaside Buffer

Page#	Frame#	Last Use

b) TLB-Zeitverhalten(4)

Man nehme an, ein Zugriff auf den TLB oder ein Zugriff auf den Cache-Speicher dauert 15ns, ein Zugriff auf den Hauptspeicher braucht 60ns und das Laden einer Seite vom Platte dauert 10ms. Gehen Sie davon aus, dass die Pagetable *zur Gänze im Hauptspeicher* gehalten wird und die durch den TLB referenzierten Seiten sowie der TLB vom Cache-Speicher unterstützt werden. Rechnen Sie den Quotienten minimale Zugriffszeit : maximale Zugriffszeit auf zwei signifikante Stellen genau aus.

Quotient $\frac{t_{min}}{t_{max}} =$

c) Verständnisfragen (6)

- Eine Austauschstrategie, die auf alle Seiten des Hauptspeichers angewandt wird, nennt man
  - ◯ lokale Ersetzungsstrategie
  - ◯ globale Ersetzungsstrategie
- Beim *Fixed Partitioning* sind die Partionen *immer* von gleicher Größe.
  - ◯ richtig
  - ◯ falsch
- Welche dieser Replacement-Policies ist am einfachsten zu implementieren?
  - ◯ Least recently used
  - ◯ Clock algorithmus
  - ◯ First in - first out
- Die Clock Replacement Strategie liefert weniger Page Faults als die Least Recently Used Replacement Strategy.
  - ◯ richtig
  - ◯ falsch
- Beim *Fixed Partitioning* können sich Partitionen im Hauptspeicher überlappen.
  - ◯ richtig
  - ◯ falsch
- Große Seitengrößen bei reinem Paging führen zu großen Seitentabellen (page tables).
  - ◯ richtig
  - ◯ falsch
- Um die interne Fragmentierung zu reduzieren, muss man die *Page Size*
  - ◯ verringern
  - ◯ vergrößern
- Wieviele Zugriffe auf die Pagetable benötigt ein System mit Translation Lookaside Buffer im Falle eines TLB Hits, um die virtuelle Adresse aufzulösen?

  Zugriff(e)

2 Speicherverwaltung (25)

a) Segmentierung (13)

In einem Computersystem mit virtuellem Speicher gibt es drei Tasks, `A`, `B`, und `C`, welche die folgenden Ressourcen benötigen:

Task	**Code** (bytes)	**Data** (bytes)
`A`	1250	798
`B`	3423	3745
`C`	1024	0

Geben Sie den Verschnitt an Speicher (= Speicher, der reserviert, aber nicht tatsächlich benötigt wird) in Bezug zum reservierten Speicher für jedes der Programme für die zwei Fälle in Punkt 1) und 2) in Prozent an.

Nehmen Sie an, dass Code und Daten getrennt gespeichert werden. Geben Sie unbedingt das Ergebnis in Form einer *Dezimalzahl* mit einer Stelle hinter dem Komma (z.B. 13.1 %) an.

1) Verwendung von Paging, Seitengröße 2 *KB* (=2048 *Bytes*) (6)

Task `A`: Anzahl der Seiten = ⇒ Verschnitt = . %

Task `B`: Anzahl der Seiten = ⇒ Verschnitt = . %

Task `C`: Anzahl der Seiten = ⇒ Verschnitt = . %

2) Segmentierung des Speichers (3)

Verschnitt von Task `A` = . %

Verschnitt von Task `B` = . %

Verschnitt von Task `C` = . %

3) Erklärung (4)

Erklären Sie einerseits für Paging, andererseits für Speichersegmentierung, durch welchen Effekt es dazu kommen kann, dass der vorhandene physikalische Speicher nicht optimal genützt wird. Bitte Namen der Effekte und Erklärung angeben.

Paging:

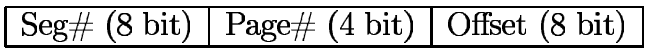
Segmentierung:

b) Kombination aus Segmentierung und Paging (12)

Es werden folgende Begriffe (englische Notation) aus dem Buch zur Vorlesung verwendet:

Base	Basisadresse Seitentabelle des Segmentes
Length	Länge des Segmentes (Anzahl der Seiten des Segmentes)
Virt.Addr.	Virtuelle Adresse
Frame#	Seitenrahmennummer (im physischen Speicher)
Page#	Seitennummer (im virtuellen Speicher)
Seg#	Segmentnummer

Das in der Folge betrachtete Speicherverwaltungssystem verwendet zur (physikalischen und virtuellen) Adressierung 20-bit Adressen. Für das Paging sind alle Seitenrahmen 256 Bytes (hexadezimal `0x00 100`) groß. Das verwendete Adressformat ist folgendes:



Hierbei wird assoziativer Zugriff (associative mapping) auf die Segmenttabelle und direkter Zugriff (direct mapping) auf die Seitentabelle verwendet.

Verwenden Sie für die Adressumsetzung folgende Segmenttabelle und Seitentabelle (alle Werte sind als Hexadezimalzahlen angegeben):

Segmenttabelle		
Seg#	Base	Length
0x02	0x04 21F	0x4
0xFC	0x7D 000	0x2
0x3F	0xC1 0E0	0x3
0x09	0x20 221	0xF

Seitentabelle	
Address	Frame#
...	...
0x04 21F	0x451
0x04 220	0x023
0x04 221	0x845
0x04 222	0x734
0x04 223	0x475
...	...
0x20 221	0x311
0x20 222	0xF00
...	...
0x20 22E	0x000
0x20 22F	0xDDD

Fortsetzung Seitentabelle	
Address	Frame#
0x20 230	0x666
0x20 231	0x765
...	...
0x7D 000	0x471
0x7D 001	0x871
0x7D 002	0x111
...	...
0xC1 0E0	0x394
0xC1 0E1	0x588
0xC1 0E2	0x127
0xC1 0E3	0x600
...	...

Ermitteln Sie unter Benützung obiger Tabellen die physikalischen Adressen zu folgenden virtuellen Adressen (ergibt sich bei der Umwandlung eine ungültige Adresse, so schreiben Sie bitte **ungültig** in das entsprechende Feld):

Virtuelle Adresse	Physikalische Adresse (zu ermitteln)
0x0123F	
0x3F311	
0x09EFF	
0x090D7	
0xFC000	
0x0238A	

2 Speicherverwaltung (25)

a) Segmentierung (13)

In einem Computersystem mit virtuellem Speicher gibt es drei Tasks, `A`, `B`, und `C`, welche die folgenden Ressourcen benötigen:

Task	**Code** (bytes)	**Data** (bytes)
`A`	1250	798
`B`	3423	3745
`C`	1024	0

Geben Sie den Verschnitt an Speicher (= Speicher, der reserviert, aber nicht tatsächlich benötigt wird) in Bezug zum reservierten Speicher für jedes der Programme für die zwei Fälle in Punkt 1) und 2) in Prozent an.

Nehmen Sie an, dass Code und Daten getrennt gespeichert werden. Geben Sie unbedingt das Ergebnis in Form einer *Dezimalzahl* mit einer Stelle hinter dem Komma (z.B. 13.1 %) an.

1) Verwendung von Paging, Seitengröße 2 *KB* (=2048 *Bytes*) (6)

Task `A`: Anzahl der Seiten = ⇒ Verschnitt = . %

Task `B`: Anzahl der Seiten = ⇒ Verschnitt = . %

Task `C`: Anzahl der Seiten = ⇒ Verschnitt = . %

2) Segmentierung des Speichers (3)

Verschnitt von Task `A` = . %

Verschnitt von Task `B` = . %

Verschnitt von Task `C` = . %

3) Erklärung (4)

Erklären Sie einerseits für Paging, andererseits für Speichersegmentierung, durch welchen Effekt es dazu kommen kann, dass der vorhandene physikalische Speicher nicht optimal genützt wird. Bitte Namen der Effekte und Erklärung angeben.

Paging:

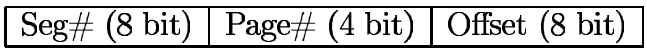
Segmentierung:

b) Kombination aus Segmentierung und Paging (12)

Es werden folgende Begriffe (englische Notation) aus dem Buch zur Vorlesung verwendet:

Base	Basisadresse Seitentabelle des Segmentes
Length	Länge des Segmentes (Anzahl der Seiten des Segmentes)
Virt.Addr.	Virtuelle Adresse
Frame#	Seitenrahmennummer (im physischen Speicher)
Page#	Seitennummer (im virtuellen Speicher)
Seg#	Segmentnummer

Das in der Folge betrachtete Speicherverwaltungssystem verwendet zur (physikalischen und virtuellen) Adressierung 20-bit Adressen. Für das Paging sind alle Seitenrahmen 256 Bytes (hexadezimal `0x00 100`) groß. Das verwendete Adressformat ist folgendes:



Hierbei wird assoziativer Zugriff (associative mapping) auf die Segmenttabelle und direkter Zugriff (direct mapping) auf die Seitentabelle verwendet.

Verwenden Sie für die Adressumsetzung folgende Segmenttabelle und Seitentabelle (alle Werte sind als Hexadezimalzahlen angegeben):

Segmenttabelle		
Seg#	Base	Length
0x02	0x04 21F	0x4
0xFC	0x7D 000	0x2
0x3F	0xC1 0E0	0x3
0x09	0x20 221	0xF

Seitentabelle	
Address	Frame#
...	...
0x04 21F	0x451
0x04 220	0x023
0x04 221	0x845
0x04 222	0x734
0x04 223	0x475
...	...
0x20 221	0x311
0x20 222	0xF00
...	...
0x20 22E	0x000
0x20 22F	0xDDD

Fortsetzung Seitentabelle	
Address	Frame#
0x20 230	0x666
0x20 231	0x765
...	...
0x7D 000	0x471
0x7D 001	0x871
0x7D 002	0x111
...	...
0xC1 0E0	0x394
0xC1 0E1	0x588
0xC1 0E2	0x127
0xC1 0E3	0x600
...	...

Ermitteln Sie unter Benützung obiger Tabellen die physikalischen Adressen zu folgenden virtuellen Adressen (ergibt sich bei der Umwandlung eine ungültige Adresse, so schreiben Sie bitte **ungültig** in das entsprechende Feld):

Virtuelle Adresse	Physikalische Adresse (zu ermitteln)
0x0123F	
0x3F311	
0x09EFF	
0x090D7	
0xFC000	
0x0238A	

2 Speicherverwaltung (20)

a) Kombination aus Segmentierung und Paging (13)

Es werden folgende Begriffe (englische Notation) aus dem Buch zur Vorlesung verwendet:

Base	Basisadresse Seitentabelle des Segmentes
Length	Länge des Segmentes (Anzahl der Seiten des Segmentes)
Virt.Addr.	Virtuelle Adresse
Frame#	Seitenrahmennummer (im physischen Speicher)
Page#	Seitennummer (im virtuellen Speicher)
Seg#	Segmentnummer

Das im Folgenden betrachtete Speicherverwaltungssystem verwendet zur Adressierung 32-bit Adressen(virtuell und physikalisch). Für das Paging sind alle Seitenrahmen 256 Bytes groß. Das verwendete Adressformat ist folgendes:

Seg# (12 bit)	Page# (12 bit)	Offset (8 bit)

Hierbei wird assoziativer Zugriff (associative mapping) auf die Segmenttabelle und direkter Zugriff (direct mapping) auf die Seitentabelle verwendet.

Verwenden Sie für die Adressumsetzung folgende Segmenttabelle und Seitentabelle (alle Werte sind als Hexadezimalzahlen angegeben):

Segmenttabelle		
Seg#	Base	Length
0x000	0xFFFFFFFF	0x001
0xABC	0xBC050010	0x002
0xB00	0xBC050012	0x0B5
0xEB0	0xA0010010	0x004
0xFFF	0x00000000	0x005

Seitentabelle	
Address	Frame#
0x00000000	0x123456
0x00000001	0x152451
0x00000002	0x152452
...	...
0xA0010010	0x761010
0xA0010011	0x761011
0xA0010012	0x761012
0xA0010013	0x761013
0xA0010014	0x761014
...	...
0xBC050010	0x666012
0xBC050011	0x666011
0xBC050012	0x666010
0xBC050013	0x66600F
...	...
0xBC0500C6	0x666000
0xBC0500C7	0x666001
0xBC0500C8	0x666002
...	...
0xFFFFFFFF	0x000001

Ermitteln Sie unter Benützung obiger Tabellen die physikalischen Adressen zu folgenden virtuellen Adressen (ergibt sich bei der Umwandlung eine ungültige Adresse, so schreiben Sie bitte **ungültig** in das entsprechende Feld):

Virtuelle Adresse	Physikalische Adresse (zu ermitteln)
0x000001FF	
0xABC00100	
0xFFF002AB	
0x000000EA	
0xEB000000	
0xB000B4FF	
0xABC001FF	
0xB000B500	
0xEB0002AA	

b) Verständnisfragen (7)

Kreuzen Sie bitte die richtigen Antworten an! Achtung! Falsche Antworten werden negativ gewertet!

- Eine Austauschstrategie, die auf alle Seiten des Hauptspeichers angewandt wird, nennt man
  - ◯ lokale Ersetzungsstrategie
  - ◯ globale Ersetzungsstrategie
- Beim *Fixed Partitioning* ist die Hauptspeichernutzung extrem effizient.
  - ◯ richtig
  - ◯ falsch
- Welche Replacement-Policy ist am einfachsten zu implementieren?
  - ◯ Least recently used
  - ◯ First in - first out
  - ◯ Optimale Strategie
- Beim *Fixed Partitioning* können sich keine Partitionen im Hauptspeicher überlappen.
  - ◯ richtig
  - ◯ falsch
- Zu welchen Effekten(mehrere möglich) kann es bei Segmentierung kommen?
  - ◯ Unterschiedliche Segmentlängen
  - ◯ Internal Fragmentation
  - ◯ External Fragmentation
- Bei einem Buddy System sind die anforderbaren Speicherblockgrößen immer größer als die größte bisher angeforderte Speicherblockgröße.
  - ◯ richtig
  - ◯ falsch

2 Memory Management - Replacement Policies (25)

Gegeben ist eine virtuelle Speicheradressierung (Paging mit Frame Allocation Size von 3) und eine Folge von Page-Zugriffen mit den folgenden Page-Nummern: 3,2,5,4,3,5,1,5,3,4,5. Es sind die Inhalte der Page-Frames nach jedem Page-Zugriff zu bestimmen. Zusätzlich sind eventuelle Page-Faults zu markieren.

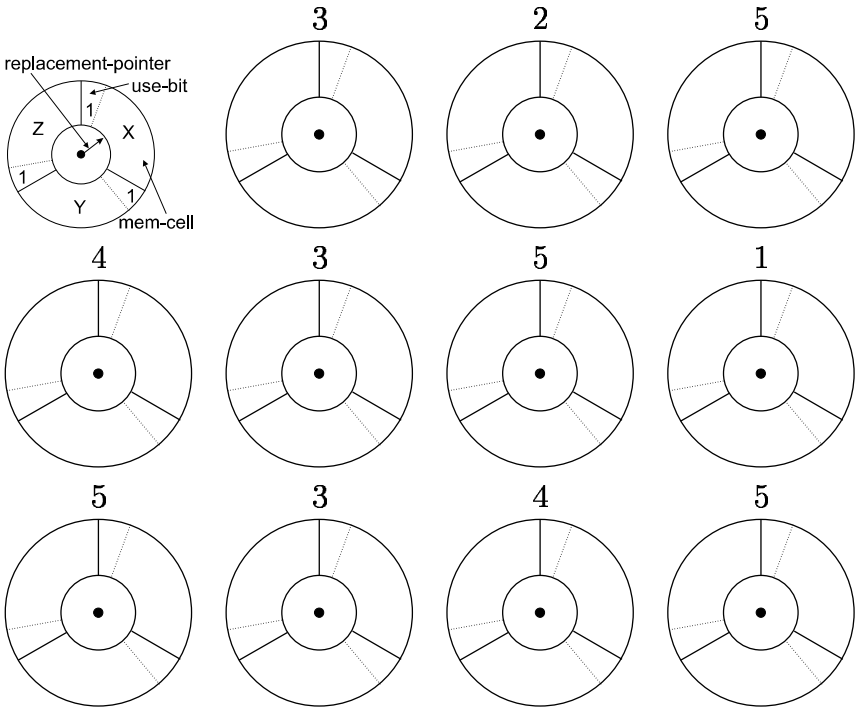
Tragen Sie in folgender Tabelle die Inhalte der Page-Frames ein, entsprechend der folgenden Replacement Policies:

- *Optimal* (OPT)
- *Least Recently Used* (LRU)
- *First-in,First-out* (FIFO)
- *Simple Clock Policy* (CLOCK)

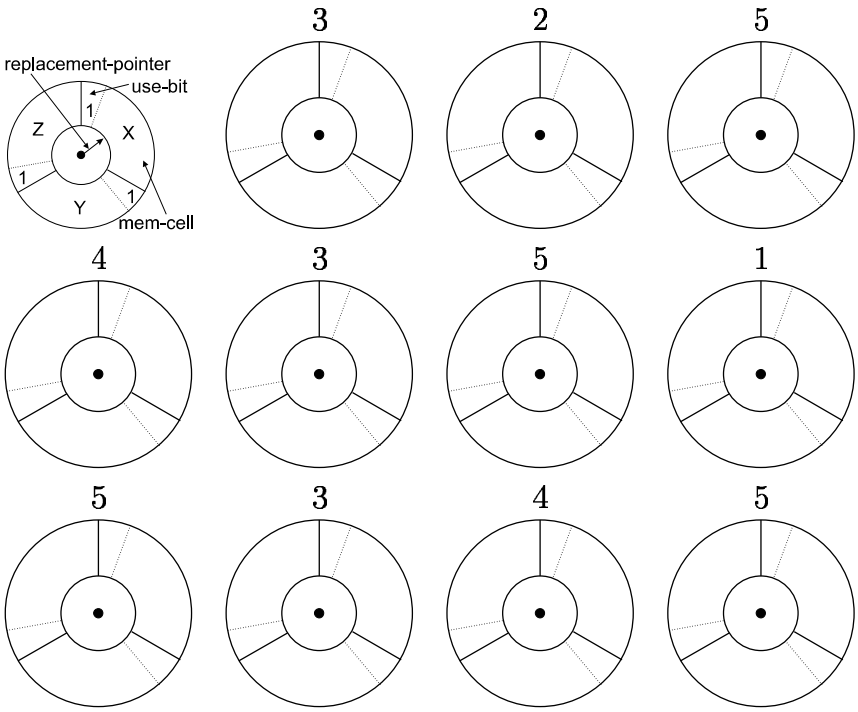
Weiters sind nachfolgende Page Faults mittels ⊗ zu markieren (für die CLOCK-Policy sind sämtliche Schritte in den Kreisdiagrammen auf der nächsten Seite einzuzeichnen - inklusive Replacement-Pointer und Usage-Bit). Sollte es gültige Alternativlösungen geben, so ist eine Variante davon auszuwählen.

	3	2	5	4	3	5	1	5	3	4	5
OPT											
	-										
	-	-									
miss	-	-	-	◯	◯	◯	◯	◯	◯	◯	◯
LRU											
	-										
	-	-									
miss	-	-	-	◯	◯	◯	◯	◯	◯	◯	◯
FIFO											
	-										
	-	-									
miss	-	-	-	◯	◯	◯	◯	◯	◯	◯	◯
CLOCK											
	-										
	-	-									
miss	-	-	-	◯	◯	◯	◯	◯	◯	◯	◯

Templates für die CLOCK-Policy (die Felder sind in der Reihenfolge *rechts - unten - links* aufzufüllen). Die Pages sind mit den Werten $\{X, Y, Z\}$ vorbelegt, welche zu den Werten $\{1, 2, 3, 4, 5\}$ paarweise disjunkt sind.



Reserve-Vorlage (die ungültige Version ist **durchzustreichen**):



Vergleich der Page-Faults

Übertragen Sie aus dem vorhergehendem Beispiel die Anzahl der Page-Faults.

Policy	Anzahl der Page-Faults
OPT	
LRU	
FIFO	
CLOCK	

Diese vier Replacement-Policies lassen sich für den allgemeinen Fall bezüglich der Effizienz mit den Bewertungen *besser/gleich/schlechter* vergleichen.

- Tragen Sie nun entsprechend den Page-Faults aus obiger Tabelle die Kriterien "<" (*besser*), "=" (*gleich*) oder ">" (*schlechter*) in die linken Teilspalten in der folgenden Tabelle ein. Die Bewertungen sind dabei jeweils vom Zeilennamen ausgehend zum Spaltennamen zu lesen (Beispiel: sollte OPT schlechter als LRU sein, tragen Sie dafür ">" im entsprechenden Feld ein).
- Markieren Sie mittels ⊗ in folgender Tabelle jene Paare von Policies, für welche die aus den **konkreten Zugriffen des obigen Beispieles** gewonnenen Aussage *besser/gleich/schlechter* **auch** für den **allgemeinen Fall** mit beliebigen Page-Zugriffen gilt.

	LRU		FIFO		CLOCK	
OPT		◯		◯		◯
LRU	-			◯		◯
FIFO	-		-			◯

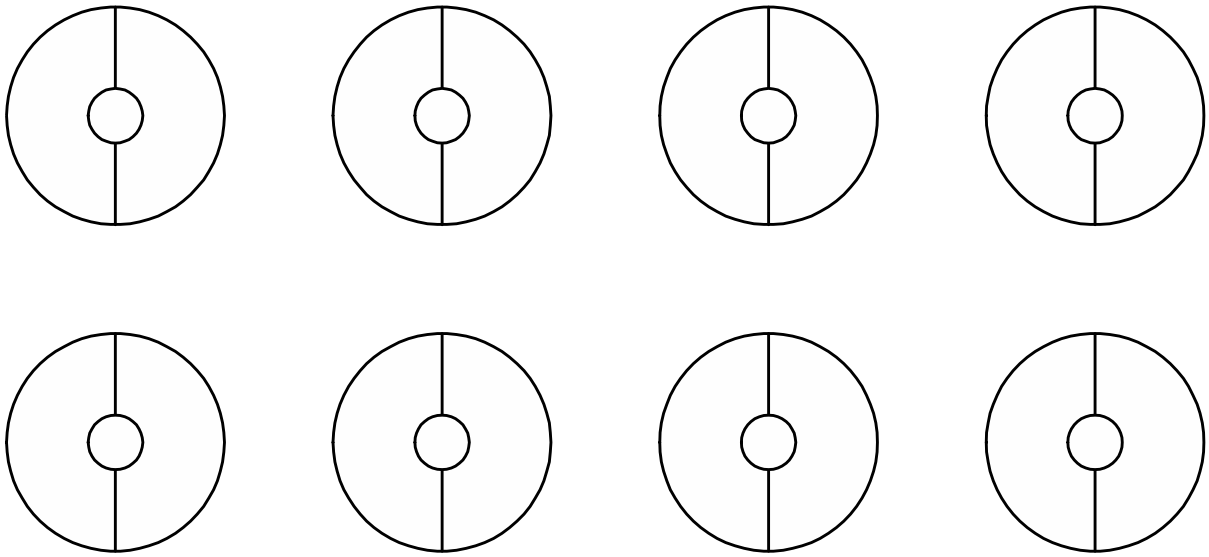
3 Speicherverwaltung - Replacement Policies(25)

a)(9)

Ein Prozess referenziert 5 pages A,B,C,D und E. Wie groß ist die Anzahl der Page Faults, die während dieser Zugriffe auftreten? Nehmen Sie jeweils an, dass der Speicher vor dem Zugriff leer ist.

Access Order	Replacement Policies	Allocation Size in Number of Pages	Number of Page Faults
A;B;C;D;A;B;E;A	First In First Out	3	
A;B;A;B;C;A;D;E	Simple Clock	2	
E;B;A;B;C;A;C;B	Least Recently Used	3	

Für etwaige Skizzen zur Simple Clock Policy verwenden Sie bitte folgende Vorlagen:



b)(16)

Ein Prozess hat 4 Page Frames alloziert. Die Allocation Size in Number of Pages ist 4. Folgende Abkürzungen werden verwendet:

TL..time loaded: Der Zeitpunkt als die Page das letzte Mal geladen wurde

TR..time reference: Der Zeitpunkt als auf die Page das letze Mal zugegriffen wurde

Die angegebenen Zeiten sind die Ticks vom Zeitpunkt des Prozessstarts (0) weggezählt.

Virtual Page Number	Page Frame	TL	TR	Clock Use Bit
2	0	60	161	1
1	1	130	160	1
0	2	26	162	1
3	3	20	163	1

Ein Page Fault der virtuellen Page 4 ist aufgetreten. Welcher Page Frame wird ausgetauscht werden, wenn die folgende Memory Managment Policy verwendet wird?

Policy	Page Frame	Warum wird der Page Frame ausgetauscht werden?
First In First Out		
Least Recently Used		
Simple Clock*		
Optimal**		

*Der Frame Allocation Pointer wird den Page Frame Nummern entsprechend weitergesetzt, die Sequenz ist also: 0,1,2,3,0,....

**Zukünftige Access Sequence nach der aktuellen Virtual Page 4: 3,0,1

3 Speicherverwaltung (25)

a) Translation Lookaside Buffer (12)

Das gegebene Speicherverwaltungssystem basiert auf der Paging-Technik und verwendet einen *Translation Lookaside Buffer (TLB)* mit drei Einträgen. Der TLB wird mit der **FIFO** Strategie verwaltet, d.h., wenn für eine neu einzutragende Seite kein Eintrag mehr frei ist, wird jener Eintrag ersetzt, der sich am längsten im TLB befindet. Zu diesem Zweck enthält der TLB für jeden Eintrag ein Feld (mit der Bezeichnung FIFO), das angibt, wie lange sich der entsprechende Eintrag in dem TLB befindet. Bitte beachten Sie, dass der niedrigste Wert dem am längsten im TLB befindlichen Eintrag zugeordnet ist. Der Zähler eines *neuen* Eintrags ist das um eins inkrementierte Maximum der bisherigen FIFO-Counters.

Der TLB und die benötigte Page Table werden mittels **Associative Mapping** angesprochen.

Eine virtuelle Speicheradresse hat folgendes Format:

Page#(8 Bit)	Offset (16 Bit)

Eine physikalische Speicheradresse hat folgendes Format:

Frame#(12 Bit)	Offset (16 Bit)

Die Speicheradressen, Frame- und Pagenummern sind im Hexadezimalsystem angegeben.

Auf der nächsten Seite ist ein Translation Lookaside Buffer und eine Page Table vorgegeben. Simulieren Sie ausgehend von diesen Daten den hintereinanderfolgenden Zugriff auf die virtuellen Speicheradressen auf den folgenden Seiten. Bitte beachten Sie dabei:

- Befindet sich eine Seite nicht im Hauptspeicher, so können Sie die Nummer des Page-Frames für die Page aus den nicht belegten frei wählen. Tragen Sie diese in der Page Table auf der nächsten Seite ein.
- Bestimmen Sie, ob es zu einem TLB Hit oder Miss kommt und ob es zu einem Main Memory Page Hit kommt oder nicht (Kreuzen Sie entsprechend **Ja** oder **Nein** an). (Hinweis: ein TLB Hit impliziert einen Page Table Hit)
- Geben Sie weiters jeweils den Inhalt des TLB **nach** dem Zugriff auf die Page an.

Ausgangssituation

Translation Lookaside Buffer (TLB):

Page#	Frame#	FIFO
22	0xFAC	2
3	0x723	1
12	0x333	3

Page Table:

Page#	Frame#
0	0x2FF
1	0x341
2	0x121
3	0x723
...	...
12	0x333
...	...
22	0xFAC
23	
24	
25	
...	...
37	
38	
39	
...	...
6A	0xA1C
6B	0xAFF

Virtuelle Adresse

0x223ABC

Physikalische Adresse

TLB Hit ◯ Ja ◯ Nein
Page Hit ◯ Ja ◯ Nein

Translation Lookaside Buffer

Page#	Frame#	FIFO

Virtuelle Adresse

0x3921C3

Physikalische Adresse

TLB Hit ◯ Ja ◯ Nein
Page Hit ◯ Ja ◯ Nein

Translation Lookaside Buffer

Page#	Frame#	FIFO

Virtuelle Adresse

0x031274

Physikalische Adresse

TLB Hit ◯ Ja ◯ Nein
Page Hit ◯ Ja ◯ Nein

Translation Lookaside Buffer

Page#	Frame#	FIFO

Virtuelle Adresse

0x121100

Physikalische Adresse

TLB Hit ◯ Ja ◯ Nein
Page Hit ◯ Ja ◯ Nein

Translation Lookaside Buffer

Page#	Frame#	FIFO

Virtuelle Adresse

0x6A001B

Physikalische Adresse

TLB Hit ◯ Ja ◯ Nein
Page Hit ◯ Ja ◯ Nein

Translation Lookaside Buffer

Page#	Frame#	FIFO

Virtuelle Adresse

0x245677

Physikalische Adresse

TLB Hit ◯ Ja ◯ Nein
Page Hit ◯ Ja ◯ Nein

Translation Lookaside Buffer

Page#	Frame#	FIFO

b) Page-Replacement Algorithms(9)

Ein Prozess referenziert die Pages A,B,C,D und E laut vorgegebenen Zugriffsschema. Simulieren Sie das Verhalten der entsprechenden Page-Replacement Algorithmen in der dafür vorgesehenen Tabelle. Am Anfang ist der TLB leer. Bei mehrdeutigen Lösungsmöglichkeiten geben Sie bitte eine gültige an. Markieren Sie einen Page Fault mit $\surd$ (letzte Zeile).

OPTIMAL

	A	B	A	D	E	B	A	C	A	D	A	C
page fault?												

Anzahl d. page faults:

LRU

	A	B	A	D	E	B	A	C	A	D	A	C
page fault?												

Anzahl d. page faults:

FIFO

	A	B	A	D	E	B	A	C	A	D	A	C
page fault?												

Anzahl d. page faults:

c) Verständnisfragen (4)

- Welche dieser Replacement-Policies ist am einfachsten zu implementieren?
  - ◯ Least recently used
  - ◯ Clock algorithmus
  - ◯ First in - first out
- Um die interne Fragmentierung zu reduzieren, muss man die *Page Size*
  - ◯ verringern
  - ◯ vergrößern
- Wieviele Zugriffe auf die Pagetable benötigt ein System mit Translation Lookaside Buffer im Falle eines TLB Hits, um die virtuelle Adresse aufzulösen?

  Zugriff(e)
- Im Vergleich zu den anderen Replacement-Policies produziert die *optimal policy* bei beschränkter Buffergröße
  - ◯ keine page faults
  - ◯ die wenigsten page faults
  - ◯ die meisten page faults

3 Speicherverwaltung (23)

a) Kombination aus Segmentierung und Paging

Es werden folgende Begriffe (englische Notation) aus dem Buch zur Vorlesung verwendet:

Base	Basisadresse Seitentabelle des Segmentes
Length	Länge des Segmentes (Anzahl der Seiten des Segmentes)
Virt.Addr.	Virtuelle Adresse
Frame#	Seitenrahmennummer (im physischen Speicher)
Page#	Seitennummer (im virtuellen Speicher)
Seg#	Segmentnummer

Das im Folgenden betrachtete Speicherverwaltungssystem verwendet zur Adressierung 32-bit Adressen (virtuell und physikalisch). Für das Paging sind alle Seitenrahmen 256 Bytes groß. Das verwendete Adressformat ist folgendes:

Seg# (12 bit)	Page# (12 bit)	Offset (8 bit)

Hierbei wird assoziativer Zugriff (associative mapping) auf die Segmenttabelle und direkter Zugriff (direct mapping) auf die Seitentabelle verwendet.

Verwenden Sie für die Adressumsetzung folgende Segmenttabelle und Seitentabelle (alle Werte sind als Hexadezimalzahlen angegeben):

Segmenttabelle		
Seg#	Base	Length
0x000	0x34500234	0x0FF
0x39A	0x2A2AA342	0x005
0x723	0x2EE23323	0x002
0xCD3	0xB0010010	0x001

Seitentabelle	
Address	Frame#
0x2A2AA342	0x456D43
0x2A2AA343	0x123499
0x2A2AA344	0x19D453
0x2A2AA345	0x4F5D1D
...	...
0x2EE23323	0x453AA1
0x2EE23324	0x434DAA
0x2EE23325	0x421111
...	...
0x345002E7	0x12432A
0x345002E8	0xABDDAD
0x345002E9	0xDF45F4
0x345002EA	0x45DAEE
...	...
0xB0010010	0x761010
0xB0010011	0x859631
0xB0010012	0x5A64D2
0xB0010013	0x5A42DF
0xB0010014	0x4AF5DA
...	...

Ermitteln Sie unter Benützung obiger Tabellen die physikalischen Adressen zu folgenden virtuellen Adressen (ergibt sich bei der Umwandlung eine ungültige Adresse, so schreiben Sie bitte **ungültig** in das entsprechende Feld) (13P, pro Fehler -2):

Virtuelle Adresse	Physikalische Adresse (zu ermitteln)
0x39A00123	
0xCD30016A	
0x39A002A2	
0x72300023	
0x39A00023	
0x72300100	
0xCD300101	
0x0000B502	
0x0000B4FF	

b) Verständnisfragen

Kreuzen Sie bitte die richtigen Antworten an. Bzw., geben Sie an, ob die Aussage richtig oder falsch ist oder beantworten Sie die Frage. (Pro Frage 1P(ausser 7.frage), 7. Frage: pro kreuz ein punkt - falsches Kreuz -1 Punkt

- Eine Austauschstrategie, die auf alle Seiten des Hauptspeichers angewandt wird, nennt man
  - ◯ lokale Ersetzungsstrategie
  - ◯ globale Ersetzungsstrategie
- Beim *Fixed Partitioning* sind die Partionen *immer* von gleicher Größe.
  - ◯ richtig
  - ◯ falsch
- Beim *Fixed Partitioning* können sich Partitionen im Hauptspeicher überlappen.
  - ◯ richtig
  - ◯ falsch
- Bei einem Buddy System sind die Blockgrößen das Produkt einer Zweierpotenz mit der kleinsten Blockgröße.
  - ◯ richtig
  - ◯ falsch
- Beim *Fixed Partitioning* ist die Hauptspeichernutzung extrem effizient.
  - ◯ richtig
  - ◯ falsch
- Welche Replacement-Policy ist am einfachsten zu implementieren?
  - ◯ Least recently used
  - ◯ First in - first out
  - ◯ Optimale Strategie
- Zu welchen Effekten (mehrere möglich) kann es bei Segmentierung kommen?
  - ◯ Unterschiedliche Segmentlängen
  - ◯ Internal Fragmentation
  - ◯ External Fragmentation
- Was ist ein *Working Set* im Speichermanagement?

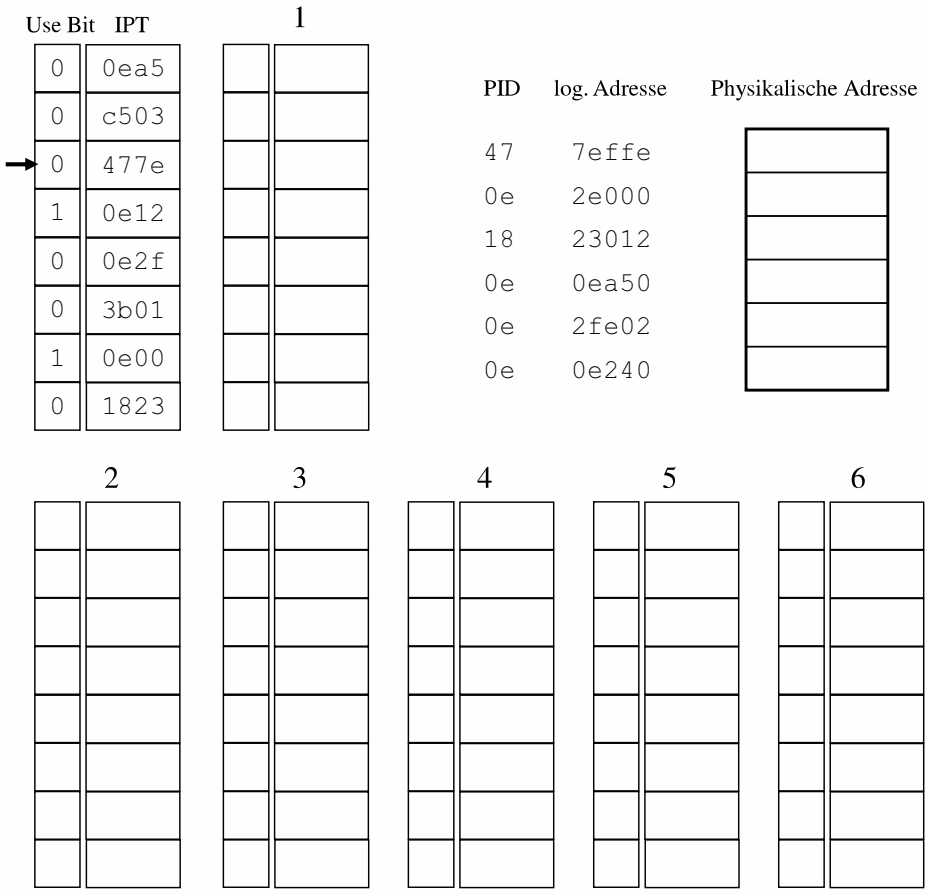
- Wenn alle Seiten eines Working Sets im Hauptspeicher sind, kann ein Prozess effizient abgearbeitet werden.
  - ◯ richtig
  - ◯ falsch

4 Memory Management (20)

Ein Betriebssystem verwendet Speicherseiten der Größe 4KB ($2^{12} Bytes$). Die Seitentabelle ist als *Inverted Page Table (IPT)* realisiert, d.h., die Seitentabelle hat soviele Einträge, wie es Speicherframes im System gibt. In jedem der Einträge ist für die im entsprechenden Frame geladene Seite die Prozessnummer des Prozesses, zu dem diese Seite gehört, sowie die Seitennummer innerhalb des Prozesses gespeichert. Dabei werden in jedem Eintrag 8 Bits für die Prozessnummer und 8 weitere Bits für die Seitennummer verwendet.

Die Abbildung zeigt eine Folge von sechs Adressreferenzen bestehend aus jeweils Prozessnummer (PID) und logischer Adresse im Prozess. Diese Speicherzugriffe sind von oben nach unten nacheinander durchzuführen, wobei bei Page Faults die Clock Policy zum Replacement von Seiten angewandt wird. Die aktuellen Werte der Use Bits sind links neben der IPT angegeben, der Pfeil kennzeichnet die aktuelle Position des Clock-Zeigers (Laufrichtung des Zeigers ist zyklisch von oben nach unten).

Geben Sie in den Tabellen den Inhalt der IPT und der Use Bits, sowie den Clock-Zeiger nach jeder Speicherreferenz an. Tragen Sie außerdem in die Tabelle rechts oben für jeden Speicherzugriff die physikalische Adresse ein.



VO 182.022 28. Jänner 2004

Prüfung Betriebssysteme

KNr. MNr. Zuname, Vorname

Ges.)(100) 1.)(30) 2.)(25) 3.)(20) 4.)(25) Zusatzblätter:

Bitte verwenden Sie nur dokumentenechtes Schreibmaterial!

1 Speicherverwaltung (30)

a) Lokalität von Prozessen – Working Sets (8)

Das Working Set Modell beschreibt das Speicherzugriffsverhalten von Prozessen. In diesem Modell ist das Working Set $W(t, \Delta)$ definiert als die Menge der Speicherseiten, die in den letzten Δ Zeiteinheiten vor dem Zeitpunkt t referenziert wurden.

Die folgende Tabelle beschreibt eine Folge von Speicherseitenzugriffen eines Prozesses. In Spalte 2 jeder Zeile steht die Nummer der Seite, auf die in der Zeitscheibe $(t-1, t]$ zugegriffen wird. Geben Sie in den leeren Feldern die Working Sets $W(t, \Delta)$ für Δ gleich 3, bzw. 4 an.

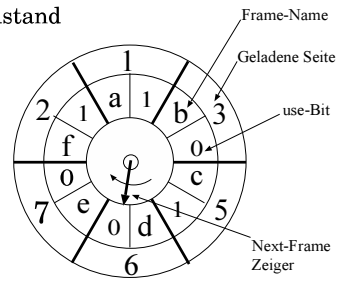
t	*Seitennr.*	$W(t,3)$	$W(t,4)$
1	0x23	0x23	0x23
2	0x18	0x23, 0x18	0x23, 0x18
3	0x19	0x23, 0x18, 0x19	0x23, 0x18, 0x19
4	0x18	0x18, 0x19	0x18, 0x19
5	0x23	0x19, 0x18, 0x23	0x19, 0x18, 0x23
6	0x19	0x18, 0x23, 0x19	0x18, 0x23, 0x19
7	0x18	0x23, 0x19, 0x18	0x23, 0x19, 0x18
8	0x3F	0x19, 0x18, 0x3f	0x23, 0x19, 0x18, 0x3f
9	0x23	0x18, 0x3f, 0x23	0x19, 0x18, 0x3f, 0x23
10	0x18	0x3f, 0x23, 0x18	0x18, 0x3f, 0x23

b) Replacement Policy (11)

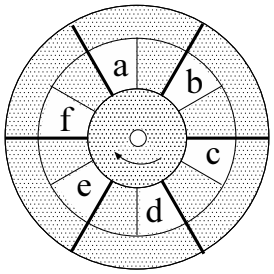
Simulieren Sie das Verhalten einer *clock policy* zum Auslagern von Speicher-Seiten, wenn neue Seiten geladen werden müssen.

Für jede Seite im Speicher existiert ein *use*-Bit. Die einzelnen Plätze im Hauptspeicher sind in diesem Beispiel mit Buchstaben von `a ... f` benannt. Der *next frame*-Zeiger bewegt sich in den verwendeten Grafiken im Uhrzeigersinn weiter.

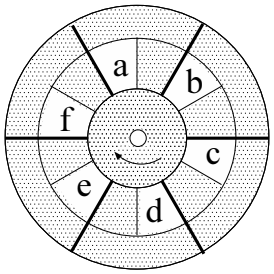
Anfangszustand



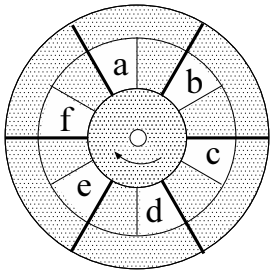
1.Zugriff: `access[1]`



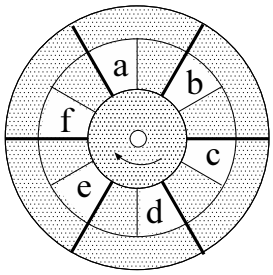
2.Zugriff: `access[8]`



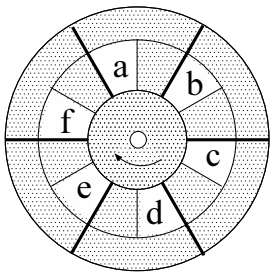
3.Zugriff: `access[4]`



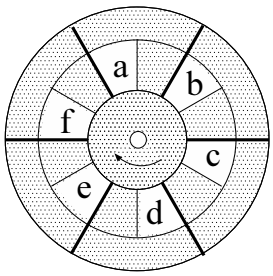
4.Zugriff: `access[2]`



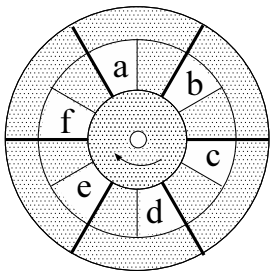
5.Zugriff: `access[7]`



6.Zugriff: `access[3]`



7.Zugriff: `access[1]`



Zeichnen Sie nun in den obigen Grafiken

- die Position des *next frame*-Zeigers
- die Inhalte aller Frames (geladene Seiten)

für die Ausführung von Befehlen mit Speicherzugriffen ein. Ein Befehl `access[x]` steht für einen Speicherzugriff auf die Seite `x`. Geben Sie den gefragten Speicherzustand **nach** der Ausführung des jeweiligen Befehls an, wobei Sie für die Befehle von einer sequentiellen Reihenfolge ausgehen können.

c) Verständnisfragen (11)

Kreuzen Sie bei den folgenden Aussagen an, ob diese richtig oder falsch sind.

- ◯ *richtig* ◯ *falsch* — Durch Vergrößerung des *Working Sets* von Prozessen kann man das Risiko von "thrashing" vermindern.
- ◯ *richtig* ◯ *falsch* — Die Clock Replacement Strategie liefert im Durchschnitt weniger Page Faults als die Least Recently Used Replacement Strategy.
- ◯ *richtig* ◯ *falsch* — Große Seitengrößen bei reinem Paging führen zu kleinen Seitentabellen (page tables).
- ◯ *richtig* ◯ *falsch* — Um die externe Fragmentierung zu reduzieren, muss man die *Page Size* vergrößern.
- ◯ *richtig* ◯ *falsch* — Unter "prepaging" versteht man einen Mechanismus, bei dem *nur* Pages, die in Zukunft referenziert werden, (und sonst keine) und damit benötigte "Pages" (Seiten) vor ihrer Verwendung in den Hauptspeicher gebracht werden.
- ◯ *richtig* ◯ *falsch* — Beim FIFO-Verfahren kann eine Vergrößerung der Anzahl der Arbeitsspeicherseiten zu einer Zunahme der Seitenfehler führen.
- ◯ *richtig* ◯ *falsch* — Eine virtuelle Adresse verweist immer auf eine Seite auf dem Sekundärspeicher.
- ◯ *richtig* ◯ *falsch* — Die Free Frame List verwaltet bei der Segmentierung die freien Frames im Speicher.
- ◯ *richtig* ◯ *falsch* — Bei der Clock Policy handelt es sich um eine Demand Paging Strategie.
- ◯ *richtig* ◯ *falsch* — Die LRU-Strategie wird in der Praxis wenig verwendet, da sie gegenüber den anderen Ansätzen eine hohe Seitenfehlerrate produziert.
- ◯ *richtig* ◯ *falsch* — Für einen gleich großen Hauptspeicher nutzt eine kleine Seitengröße das Lokalitätsprinzip besser aus als eine große Seitengröße.
- ◯ *richtig* ◯ *falsch* — Ein virtueller Speicher kann sowohl mit Paging als auch mit Segmentierung implementiert werden.
- ◯ *richtig* ◯ *falsch* — Beim Buddy System kann es zu externer Fragmentierung kommen.
- ◯ *richtig* ◯ *falsch* — Beim Buddy System kann es zu interner Fragmentierung kommen.

4 Memory Management (25)

Bei einem Einprogrammsystem (Single Task System) kann man den Hauptspeicher in zwei Bereiche einteilen, den Speicher für das Betriebssystem (Kernel) und für das gerade ausgeführte Programm. Bei einem Mehrprogrammsystem (Multi Task System) ist der Hauptspeicher in mehrere Bereiche unterteilt. Die Verwaltung dieser Bereiche übernimmt eine sogenannte Speicherverwaltung.

Welche typischen Anforderungen stellt man an eine solche Speicherverwaltung?(3 Punkte)

Erklären Sie den Begriff *virtueller Speicher*. (3 Punkte)

Die zwei Grundtechniken um virtuellen Speicher zu realisieren sind *Segmentierung* und *Paging*. Erklären Sie die Gemeinsamkeiten, Unterschiede, Probleme, ...diese beiden Techniken. (8 Punkte)

Erklären Sie assoziative Zuordnung (associative mapping) und direkte Zuordnung (direct mapping). Geben Sie dazu jeweils ein Beispiel an. (5 Punkte)

Erklären Sie für ein einstufiges Paging-System die genauen Vorgänge bei einem TLB-Hit und TLB-Miss. Auf welche Speicherseiten wird zugegriffen, wie wird entschieden welcher Eintrag ausgewählt wird? (6 Punkte)

4 Memory Management (25)

a) Virtueller Speicher mit Kombination aus Segmentierung und Paging 14

In folgendem Beispiel sollen Sie ausgehend von virtuellen Adressen die physikalischen Adressen bestimmen. Es werden folgende Begriffe (englische Notation) aus dem Buch zur Vorlesung verwendet:

Base	Basisadresse Seitentabelle des Segmentes
Length	Länge des Segmentes (Anzahl der Seiten des Segmentes)
Virt.Addr.	Virtuelle Adresse
Frame#	Seitenrahmennummer (im physischen Speicher)
Page#	Seitennummer (im virtuellen Speicher)
Seg#	Segmentnummer

Es handelt sich bei dem System um ein System mit 32-Bit-Adressen, wobei die niederwertigen 16 Bit immer den Offset einer Adresse bilden und die höherwertigen 16 Bit Segment- und Seitennummer bilden:

Seg# (8 bit)	Page# (8 bit)	Offset (16 bit)

Es wird dabei direkter Zugriff (direct mapping) sowohl auf die Segmenttabelle als auch auf die Seitentabelle verwendet.

Bei Paging sind alle Seiten 65536 Bytes (hexadezimal `0x0001 0000`) lang. Alle Werte sind als Hexadezimalzahlen angegeben. Ergibt sich bei der Umwandlung eine ungültige Adresse, so schreiben Sie bitte **ungültig** in das entsprechende Feld.

Verwenden Sie für die Adressumsetzung folgende Segmenttabelle:

Segmenttabelle		
Seg#	Base	Length
0x00	0x0827 7234	0x03
0x01	0x1043 6073	0x01
0x02	0x4074 8054	0x02
0x03	0x5322 23AA	0x10
0x04	0x3012 B578	0x0A

und folgende Seitentabelle:

Seitentabelle	
Address	Frame#
...	...
0x0827 7234	0x6752
0x0827 7235	0x7613
0x0827 7236	0x258A
0x0827 7237	0xB3CA
...	...
0x1043 6073	0x56EF
0x1043 6074	0x4567
...	...
0x3012 B57F	0x5014
0x3012 B580	0x5109
0x3012 B581	0xA145
0x3012 B582	0x2000
0x3012 B583	0x3234
...	...
0x4074 8054	0x7353
0x4074 8055	0xBABA
0x4074 8056	0x9789
0x4074 8057	0xAFFE
...	...
0x5322 23B7	0x5400
0x5322 23B8	0x5401
0x5322 23B9	0x0945
0x5322 23BA	0x1313
...	...

Ermitteln Sie unter Benützung obiger Tabellen die physikalischen Adressen zu den folgenden virtuellen Adressen. Markieren Sie die den Eintrag mit "invalid segment nr." bzw. "invalid page nr." im Fehlerfall.

Virtuelle Adresse	Physikalische Adresse (zu ermitteln)
0x030F 01CE	
0x0001 04B3	
0x0101 2019	
0x0409 1025	
0x0539 0715	
0x0407 197A	
0x0310 9788	

Bitte beachten Sie, dass bei diesem Beispiel falsche Antworten zu Punkteabzug führen.

b) Speicherfragmentierung 11

Geben Sie Beispiele von Speicherbelegungen für die jeweiligen Fragmentierungswerte an. Der Speicher besteht dabei aus durchnummerierten Speicherblöcken. Die allokierten Speicherbereiche sind immer auf die Grenzen von Speicherblöcken ausgerichtet. Beachten Sie dabei, dass die Beispiele so gewählt worden sind, dass die Aufteilung in Fragmentierungsbereiche ganzzahlig möglich ist. Nicht ganzzahlige Lösungen werden nicht gewertet!

Markieren Sie die Speicherbereiche dabei folgendermaßen:

 ...Nutzdaten (Granularität: 1 Block)

 ...Interne Fragmentierung (angegeben in % relativ zum allokierten Speicher)

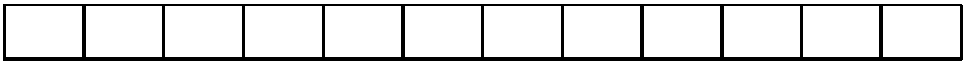
 ...Externe Fragmentierung (angegeben in % relativ zum Gesamtspeicher)

internal fragmentation: 0%, **external fragmentation:** 50%, **Größe der Allokationseinheit: 3 Blöcke**

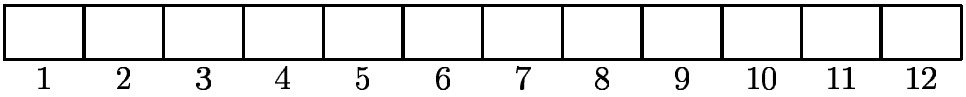
Ersatzvorlage:

internal fragmentation: 33.3%, **external fragmentation:** 0%, **Größe der Allokationseinheit:** **Blöcke (frei wählbar)**

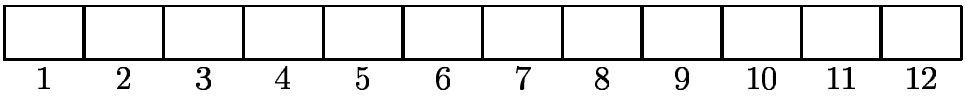
Ersatzvorlage:



internal fragmentation (if) = 50%, external fragmentation (ef) = 33.3%, Größe der Allokationseinheit: 4 Blöcke



Ersatzvorlage:



Nennen Sie eine Speicherverwaltungstechnik, bei der ausschließlich interne Fragmentierung auftreten kann:

Nennen Sie eine Speicherverwaltungstechnik, bei der ausschließlich externe Fragmentierung auftreten kann:

Nennen Sie eine Speicherverwaltungstechnik, bei der sowohl interne wie auch externe Fragmentierung auftreten kann:

4 Speicherverwaltung (22)

Das betrachtete Speicherverwaltungssystem verwendet zur Adressierung 16-Bit Adressen. Für die angegebenen virtuellen Speicheradressen sind, in Abhängigkeit von der Adressierungsart, die entsprechenden physikalischen Adressen zu ermitteln. Von den angebenen Adressen sind die niederwertigen 8 Bit der Offset der Adresse und die höherwertigen 8 Bit die Segmentnummer bzw. die Seitennummer. Bei Paging sind alle Seiten 256 Bytes (Hexadezimal 0x100) lang. Alle Werte mit führendem **0x** sind als Hexadezimalzahl angegeben. Ergibt sich bei der Umwandlung eine ungültige Adresse, so schreiben Sie bitte **ungültig** in das entsprechende Feld.

Es werden folgende Begriffe (englische Notation) verwendet:

Base	Basisadresse des Segmentes
Length	Länge des Segmentes
Frame#	Seitenrahmennummer (im physischen Speicher)
Page#	Seitennummer (im virtuellen Speicher)

a) Paging — Assoziativer Zugriff (associative mapping)

Adressübersetzungstabelle:

Page#	Frame#
0x11	0x33
0x01	0x17
0x17	0x20
0x42	0x08
0x05	0x05

Zu berechnende Adressen:

Virtuelle Adresse	Physikalische Adresse
0x083A	
0x01FF	
0x0505	
0x063A	

b) Paging — Direkter Zugriff (direct mapping)

Adressübersetzungstabelle:

Entry	Frame#
0	0x42
1	0x06
2	0x04
3	0x11
4	0x01
5	0xFA
6	0x02

Zu berechnende Adressen:

Virtuelle Adresse	Physikalische Adresse
0x0406	
0x0611	
0x11FA	
0xFA11	

c) Segmentierung — Direkter Zugriff (direct mapping)

Adressübersetzungstabelle:

Entry	Base	Length
0	0x1830	0x020
1	0x0810	0x0FF
2	0x42AC	0x100
3	0x0400	0x004
4	0x01FE	0x0F0
5	0x0010	0x030
6	0x3302	0x050
7	0x4220	0x010

Zu berechnende Adressen:

Virtuelle Adresse	Physikalische Adresse
0x1832	
0x0402	
0x4222	
0x0010	
0x0305	
0x01F2	
0x0650	

d) Verständnisfragen

Kreuzen Sie bitte die richtigen Antworten an! Achtung! Falsche Antworten werden negativ gewertet!

- Bei Paging kann das Problem der externen Fragmentierung auftreten.
  - ◯ richtig
  - ◯ falsch
- Beim *Fixed Partitioning* sind die Partionen *immer* von gleicher Größe.
  - ◯ richtig
  - ◯ falsch
- In einem fehlerfreien System können zwei oder mehr virtuelle Adressen auf eine physikalische Adresse abgebildet sein.
  - ◯ richtig
  - ◯ falsch
- Zu welchen Effekten kann es bei Segmentierung kommen?
  - ◯ Unterschiedliche Segmentlängen
  - ◯ Internal Fragmentation
  - ◯ External Fragmentation

  Je größer die "page size" (Seitengröße), desto mehr "pages" (Seiten) und "page frames" (Seitenrahmen) gibt es und desto größer müssen die "page tables" (Seitentabellen) sein.
  - ◯ richtig
  - ◯ falsch

4 Speicherverwaltung (22)

Das betrachtete Speicherverwaltungssystem verwendet zur Adressierung 16-Bit Adressen. Für die angegebenen virtuellen Speicheradressen sind, in Abhängigkeit von der Adressierungsart, die entsprechenden physikalischen Adressen zu ermitteln. Von den angebenen Adressen sind die niederwertigen 8 Bit der Offset der Adresse und die höherwertigen 8 Bit die Segmentnummer bzw. die Seitennummer. Bei Paging sind alle Seiten 256 Bytes (Hexadezimal 0x100) lang. Alle Werte mit führendem **0x** sind als Hexadezimalzahl angegeben, Werte mit abschließendem **b** sind als Binärzahl zu interpretieren. Ergibt sich bei der Umwandlung eine ungültige Adresse, so schreiben Sie bitte **ungültig** in das entsprechende Feld.

Es werden folgende Begriffe (englische Notation) verwendet:

Base	Basisadresse des Segmentes
Entry	Eintrag in der Adressübersetzungstabelle
Frame#	Seitenrahmennummer (im physischen Speicher)
Length	Länge des Segmentes
Page#	Seitennummer (im virtuellen Speicher)

a) Paging — Assoziativer Zugriff (associative mapping)

Adressübersetzungstabelle:

Page#	Frame#
00000101b	0x11
00100011b	0x01
00000001b	0x30
00110000b	0x41
00010001b	0x02

Zu berechnende Adressen:

Virtuelle Adresse	Physikalische Adresse
0x1541	
0x0110	
0x3105	
0x0502	

b) Paging — Direkter Zugriff (direct mapping)

Adressübersetzungstabelle:

Entry	Frame#
0	0x13
1	0x08
2	0x34
3	0x31
4	0x01
5	0x68
6	0xA0

Zu berechnende Adressen:

Virtuelle Adresse	Physikalische Adresse
0x1306	
0x0511	
0x028A	
0x0401	

c) Segmentierung — Direkter Zugriff (direct mapping)

Adressübersetzungstabelle:

Entry	Base	Length
0	0x2840	0x004
1	0x0563	0x0FF
2	0x72AB	0x100
3	0x0300	0x010
4	0x11FD	0x0F0
5	0x4444	0x030
6	0x3112	0x060
7	0x4220	0x010

Zu berechnende Adressen:

Virtuelle Adresse	Physikalische Adresse
0x04A0	
0x4222	
0x0010	
0x0305	
0x01F2	
0x0650	

d) Verständnisfragen

Kreuzen Sie bitte die richtigen Antworten an! Achtung! Falsche Antworten werden negativ gewertet!

- Welche Aussagen treffen beim *Fixed Partitioning* zu?
  - ◯ Die Partitionen sind immer von gleicher Größe
  - ◯ Der vorhandene Speicher wird uneffizient ausgenutzt
  - ◯ Das Problem der Internal Fragmentation tritt auf

- In einem fehlerfreien System können zwei oder mehr virtuelle Adressen auf eine physikalische Adresse abgebildet sein.
  - ◯ Richtig
  - ◯ Falsch

- Zu welchen Effekten kann es bei Paging kommen?
  - ◯ Unterschiedliche Längen der Pages
  - ◯ Internal Fragmentation
  - ◯ External Fragmentation

- Bei Speicherverwaltung mittels Segmentierung berechnet sich die physikalische Adresse aus der virtuellen Adresse durch:
  - ◯ Addition der physikalischen Segmentstartadresse mit der virtuellen Adresse
  - ◯ Multiplikation der physikalischen Segmentstartadresse mit dem Offset-Teil der virtuellen Adresse
  - ◯ Addition der physikalischen Segmentstartadresse mit dem Offset-Teil der virtuellen Adresse
  - ◯ Ersetzen der Seitennummer in der virtuellen Adresse durch die physikalische Segmentstartadresse

4 Memory Management (32)

a) Virtueller Speicher mit Kombination aus Segmentierung und Paging 18

In folgendem Beispiel sollen Sie ausgehend von virtuellen Adressen die physikalischen Adressen bestimmen. Es werden folgende Begriffe (englische Notation) aus dem Buch zur Vorlesung verwendet:

Base	Basisadresse Seitentabelle des Segmentes
Length	Länge des Segmentes (Anzahl der Seiten des Segmentes)
Virt.Addr.	Virtuelle Adresse
Frame#	Seitenrahmennummer (im physischen Speicher)
Page#	Seitennummer (im virtuellen Speicher)
Seg#	Segmentnummer

Es handelt sich bei dem System um ein System mit 32-Bit-Adressen, wobei die niederwertigen 16 Bit immer den Offset einer Adresse bilden und die höherwertigen 16 Bit Segment- und Seitennummer bilden:

Seg# (8 bit)	Page# (8 bit)	Offset (16 bit)

Es wird dabei direkter Zugriff (direct mapping) sowohl auf die Segmenttabelle als auch auf die Seitentabelle verwendet.

Bei Paging sind alle Seiten 65536 Bytes (hexadezimal `0x0001 0000`) lang. Alle Werte sind als Hexadezimalzahlen angegeben. Ergibt sich bei der Umwandlung eine ungültige Adresse, so schreiben Sie bitte "invalid segment nr." bzw. "invalid page nr." in das entsprechende Feld.

Verwenden Sie für die Adressumsetzung folgende Segmenttabelle:

Segmenttabelle		
Seg#	Base	Length
0x00	0x0102 7234	0x03
0x01	0x0300 6073	0x01
0x02	0xC9C0 8054	0x02
0x03	0x8A9F 23AA	0x10
0x04	0x2001 B578	0x0A

und folgende Seitentabelle:

Seitentabelle	
Address	Frame#
...	...
0x0102 7234	0x6752
0x0102 7235	0x7613
0x0102 7236	0x258A
0x0102 7237	0xB3CA
...	...
0x0300 6073	0x56EF
0x0300 6074	0x4567
...	...
0x2001 B57F	0x5014
0x2001 B580	0x5109
0x2001 B581	0xA145
0x2001 B582	0x2000
0x2001 B583	0x3234
...	...
0xC9C0 8054	0x7353
0xC9C0 8055	0xBABA
0xC9C0 8056	0x9789
0xC9C0 8057	0xAFFE
...	...
0x8A9F 23B7	0x5400
0x8A9F 23B8	0x5401
0x8A9F 23B9	0x0945
0x8A9F 23BA	0x1313
...	...

Ermitteln Sie unter Benützung obiger Tabellen die physikalischen Adressen zu den folgenden virtuellen Adressen. Markieren Sie die den Eintrag mit "invalid segment nr." bzw. "invalid page nr." im Fehlerfall.

Virtuelle Adresse	Physikalische Adresse (zu ermitteln)
0x030F 01CE	
0x0001 04B3	
0x8A9F 23B9	
0x0409 1025	
0x0409 0102	
0x04B0 1019	
0x0539 0715	
0x0407 294A	
0x0310 9728	

Bitte beachten Sie, dass bei diesem Beispiel falsche Antworten zu Punkteabzug führen.

b) Verständnisfragen (14)

Kreuzen Sie bitte die richtigen Antworten an! Achtung! Falsche Antworten werden negativ gewertet!

- Bei folgender Speicherverwaltungstechnik kann das Problem der internen Fragmentierung auftreten.
  - ◯ fixed partitioning
  - ◯ simple segmentation
  - ◯ virtual-memory paging
  - ◯ dynamic partitioning
  - ◯ simple paging
  - ◯ virtual memory with combination of paging and segmentation

- Beim *Fixed Partitioning* sind die Partionen *immer* von gleicher Größe.
  - ◯ richtig
  - ◯ falsch

- Bei folgender Speicherverwaltungstechnik tritt sowohl interne also auch externe Fragmentierung auf.
  - ◯ fixed partitioning
  - ◯ simple segmentation
  - ◯ virtual memory with combination of paging and segmentation
  - ◯ virtual-memory paging
  - ◯ dynamic partitioning
  - ◯ simple paging

- In einem fehlerfreien System können zwei oder mehr virtuelle Adressen auf eine physikalische Adresse abgebildet sein.
  - ◯ richtig
  - ◯ falsch

4 Memory Management (32)

a) Virtueller Speicher mit Kombination aus Segmentierung und Paging 18

In folgendem Beispiel sollen Sie ausgehend von virtuellen Adressen die physikalischen Adressen bestimmen. Es werden folgende Begriffe (englische Notation) aus dem Buch zur Vorlesung verwendet:

Base	Basisadresse Seitentabelle des Segmentes
Length	Länge des Segmentes (Anzahl der Seiten des Segmentes)
Virt.Addr.	Virtuelle Adresse
Frame#	Seitenrahmennummer (im physischen Speicher)
Page#	Seitennummer (im virtuellen Speicher)
Seg#	Segmentnummer

Es handelt sich bei dem System um ein System mit 32-Bit-Adressen, wobei die niederwertigen 16 Bit immer den Offset einer Adresse bilden und die höherwertigen 16 Bit Segment- und Seitennummer bilden:

Seg# (8 bit)	Page# (8 bit)	Offset (16 bit)

Es wird dabei direkter Zugriff (direct mapping) sowohl auf die Segmenttabelle als auch auf die Seitentabelle verwendet.

Bei Paging sind alle Seiten 65536 Bytes (hexadezimal `0x0001 0000`) lang. Alle Werte sind als Hexadezimalzahlen angegeben. Ergibt sich bei der Umwandlung eine ungültige Adresse, so schreiben Sie bitte "invalid segment nr." bzw. "invalid page nr." in das entsprechende Feld.

Verwenden Sie für die Adressumsetzung folgende Segmenttabelle:

Segmenttabelle		
Seg#	Base	Length
0x00	0x0102 7234	0x03
0x01	0x0300 6073	0x01
0x02	0xC9C0 8054	0x02
0x03	0x8A9F 23AA	0x10
0x04	0x2001 B578	0x0A

und folgende Seitentabelle:

4 Memory Management (25)

Das betrachtete Speicherverwaltungssystem verwendet zur Adressierung 16-Bit Adressen. Für die angegebenen virtuellen Speicheradressen sind, in Abhängigkeit von der Adressierungsart, die entsprechenden physikalischen Adressen zu ermitteln. Von den angebenen Adressen sind die niederwertigen 8 Bit der Offset der Adresse und die höherwertigen 8 Bit die Segmentnummer bzw. die Seitennummer. Bei Paging sind alle Seiten 256 Bytes (Hexadezimal 0x100) lang. Alle Werte mit führendem **0x** sind als Hexadezimalzahl angegeben, Werte mit abschließendem **b** sind als Binärzahl zu interpretieren. Ergibt sich bei der Umwandlung eine ungültige Adresse, so schreiben Sie bitte **ungültig** in das entsprechende Feld.

Es werden folgende Begriffe (englische Notation) verwendet:

Base	Basisadresse des Segmentes
Entry	Eintrag in der Adressübersetzungstabelle
Frame#	Seitenrahmennummer (im physischen Speicher)
Length	Länge des Segmentes
Page#	Seitennummer (im virtuellen Speicher)

a) Paging — Assoziativer Zugriff (associative mapping)

Adressübersetzungstabelle:

Page#	Frame#
00010110b	0x41
00001000b	0x5A
01011110b	0x13
00100100b	0xAF
00010001b	0x20

Zu berechnende Adressen:

Virtuelle Adresse	Physikalische Adresse
0x5E41	
0x24AB	
0x108A	
0x16B2	

b) Paging — Direkter Zugriff (direct mapping)

Adressübersetzungstabelle:

Entry	Frame#
0	0x24
1	0xB1
2	0x77
3	0x86
4	0xF4
5	0xCC
6	0x01

Zu berechnende Adressen:

Virtuelle Adresse	Physikalische Adresse
0x0311	
0x14AC	
0x0512	
0x08A8	

c) Segmentierung — Direkter Zugriff (direct mapping)

Adressübersetzungstabelle:

Entry	Base	Length
0	0x2840	0x004
1	0x0563	0x0FF
2	0x72AB	0x100
3	0x0300	0x010
4	0x11FD	0x0F0
5	0x4444	0x030
6	0x3112	0x060
7	0x4220	0x010

Zu berechnende Adressen:

Virtuelle Adresse	Physikalische Adresse
0x01F2	
0x0305	
0x0010	
0x04A0	
0x4222	

Bewertung: 1 Pluspunkt pro richtiger Lösung, 1 Punkt Abzug pro falscher Lösung.

d) Verständnisfragen (12)

Kreuzen Sie bitte die richtigen Antworten an. Achtung! Falsche Antworten werden negativ gewertet. (Bewertung: 2 Pluspunkte pro richtiger Antwort, 2 Punkte Abzug pro falscher Antwort.)

- Bei folgender Speichverwaltungstechnik tritt sowohl *interne* als auch *externe* Fragmentierung auf.
  - ◯ fixed partitioning
  - ◯ simple segmentation
  - ◯ virtual memory with combination of paging and segmentation
  - ◯ virtual memory paging
  - ◯ dynamic partitioning
  - ◯ simple paging
- Zu welchen Effekten kann es bei *Segmentierung* kommen?
  - ◯ Internal Fragmentation
  - ◯ External Fragmentation
- In einem fehlerfreien System können zwei oder mehrere virtuelle Adressen auf eine physikalische Adresse abgebildet sein.
  - ◯ richtig
  - ◯ falsch
- Eine virtuelle Adresse verweist immer auf eine Seite auf dem Sekundärspeicher.
  - ◯ richtig
  - ◯ falsch
- Um die externe Fragmentierung zu reduzieren, muss man die *Page Size* vergrößern.
  - ◯ richtig
  - ◯ falsch
- Ein virtueller Speicher kann sowohl mit Paging als auch mit Segmentierung implementiert werden.
  - ◯ richtig
  - ◯ falsch

2 Memory Management (20)

Was ist der Unterschied zwischen lokalem und globalem Page Replacement?

[redacted]

Die Page Table eines kleinen Computersystems ist als Inverted Page Table (IPT) mit acht Einträgen realisiert. Beim Auftreten von Page Faults wird in diesem Computersystem der Clock Algorithmus als globale Page Replacement Strategie eingesetzt.

Im zu lösenden Beispiel ist eine Ausgangsbelegung der Page Table (links), sowie eine Folge von Zugriffen auf den virtuellen Speicher, charakterisiert durch Prozessnummer (PID), adressierte Seite (page) und Offset, gegeben. Simulieren Sie die Ausführung der Zugriffsfolge von links nach rechts und tragen Sie für jeden Zugriff auf den virtuellen Speicher (a) die entsprechende physikalische Adresse, (b) den Zustand der IPT und (c) den Wert des Replacement Pointers (rep. pointer) für den Clock Algorithmus nach dem Speicherzugriff an (Sie brauchen bei jedem Schritt nur die Werte in der IPT anzugeben, die sich beim aktuellen Zugriff geändert haben).

#vielleicht beim lesen wird 0 gesetzt look book

PID	page	offset
1	0	00a0

phys. address: page fault

PID	page	use
0	1	5	1
1	3	1	0
2	2	3	0
3	2	1	0
4	1	2	1
5	1	3	1
6	1	4	1
7	3	3	1

rep. pointer: 5

PID	page	offset
1	0	00a2

phys. address: 0x100a2

PID	page	use
0			0
1	1	0	1
2			0
3			0
4			1
5			0
6			0
7			0

rep. pointer: 2

PID	page	offset
1	4	0002

phys. address: 0x60002

PID	page	use
0			0
1	1	0	1
2			0
3			0
4			1
5			0
6			0
7			0

rep. pointer: 2

PID	page	offset
2	3	fffe

phys. address: 0x2fffe

PID	page	use
0			0
1			1
2			0
3			0
4			1
5			0
6	1	4	1
7			0

rep. pointer: 2

PID	page	offset
2	4	0000

phys. address: page fault

PID	page	use
0	2	3	1
1			1
2			0
3			0
4			1
5			0
6			1
7			0

rep. pointer: 2

PID	page	use
0			
1			
2			0
3	2	4	1
4			
5			
6			
7			

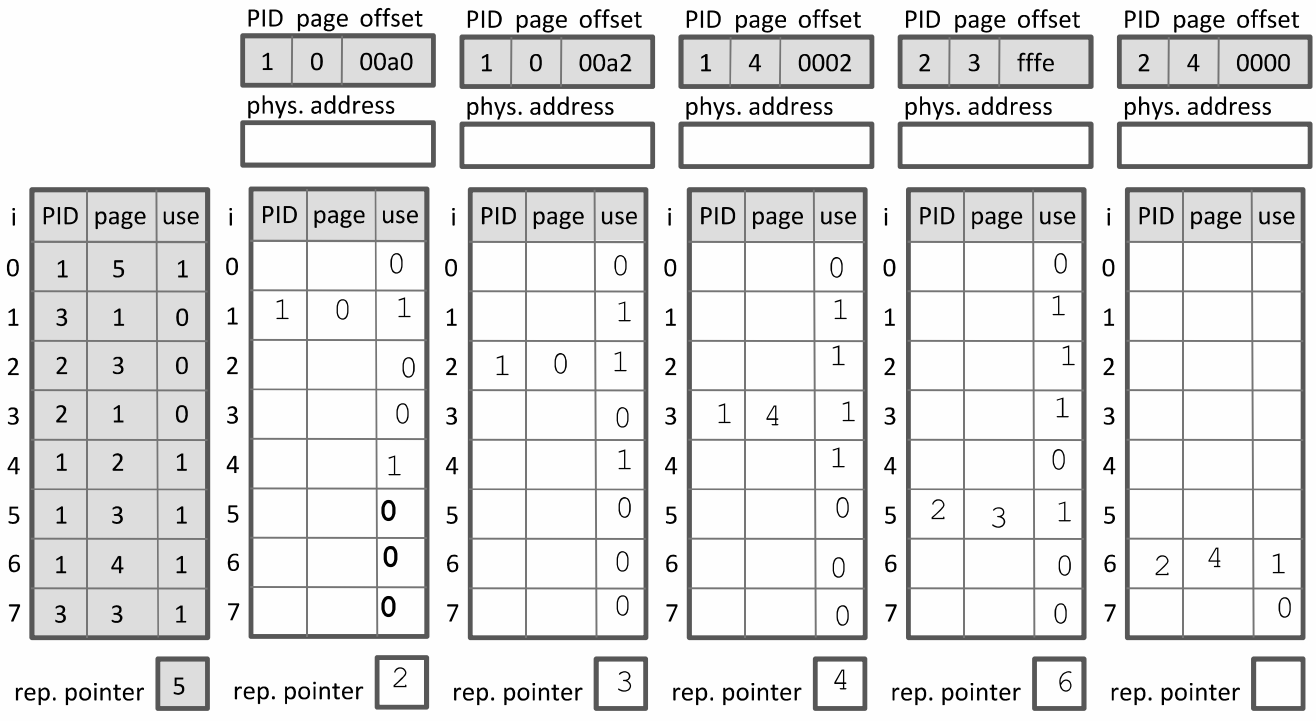
rep. pointer: 4

2 Memory Management (20)

Was ist eine Inverted Page Table?

Die Page Table eines kleinen Computersystems ist als Inverted Page Table (IPT) mit acht Einträgen realisiert. Beim Auftreten von Page Faults wird in diesem Computersystem der Clock Algorithmus als globale Page Replacement Strategie eingesetzt.

Im zu lösenden Beispiel ist eine Ausgangsbelegung der Page Table (links), sowie eine Folge von Zugriffen auf den virtuellen Speicher, charakterisiert durch Prozessnummer (PID), adressierte Seite (page) und Offset, gegeben. Simulieren Sie die Ausführung der Zugriffsfolge von links nach rechts und tragen Sie für jeden Zugriff auf den virtuellen Speicher (a) die entsprechende physikalische Adresse, (b) den Zustand der IPT und (c) den Wert des Replacement Pointers (rep. pointer) für den Clock Algorithmus nach dem Speicherzugriff an (Sie brauchen bei jedem Schritt nur die Werte in der IPT anzugeben, die sich beim aktuellen Zugriff geändert haben).



2 Page Replacement (25)

Gebeben ist ein Arbeitsspeicher mit vier Frames, dessen Seiten mit unterschiedlichen Ersetzungsstrategien ersetzt werden sollen: mit OPT, LRU, bzw. mit dem Clock Algorithmus. Geben Sie in den Tabellen für jeden Algorithmus die Speicherinhalte für jeden Frame nach jedem Zugriff der angegebenen Seitenzugriffsfolge an. Die Seitenzugriffsfolge ist für alle Algorithmen gleich. Sie ist jeweils in der Kopfzeile der Tabelle gegeben. Geben Sie in den Spalten die Speicherbelegung nach dem entsprechenden Seitenzugriff an und kennzeichnen Sie in der letzten, mit *PF* markierten Zeile das Auftreten von Page Faults. Der Arbeitsspeicher ist am Beginn leer.

OPT-Strategie:

	A	B	C	D	E	F	C	D	E	C	F	G	H	E	D
0															
1															
2															
3															
PF															

LRU-Strategie:

	A	B	C	D	E	F	C	D	E	C	F	G	H	E	D
0															
1															
2															
3															
PF															

Clock-Algorithmus:

	A	B	C	D	E	F	C	D	E	C	F	G	H	E	D
0															
1															
2															
3															
PF															

Vergleichen und diskutieren Sie die Anzahl der bei den Seitenersetzungsstrategien beobachteten Page Faults.