Universität Bielefeld – Technische Fakultät
AG Rechnernetze und verteilte Systeme

Vorlesung 4: Memory

Wintersemester 2001/2002

Peter B. Ladkin
ladkin@rvs.uni-bielefeld.de

Address Translation

- Die Adressen, die das CPU benutzt, sind nicht identisch mit den Adressen, die der Speicher benutzt
- Die Adressen müssen übersetzt werden
- Diese Funktion u.a. heisst **Memory Management**

Bit, Byte, Word

- 1 Byte = 8 Bit (genügend, um ein Character in ASCII - die Amerikanische Standard - darzustellen)
- 1 Word =
  - 1 Byte ("8-Bit Maschine"), oder
  - 2 Byte ("16-Bit Maschine"), oder
  - 4 Byte ("32-Bit Maschine"), oder
  - 8 Byte ("64-Bit Maschine")

Address Space

- Prinzip: Die mögliche Adressen sind 1-Word gross
- 1 Word = 8 Bit, haben wir eine "8-bit" Maschine
- 1 Word = 2 Bytes = 16 Bit, haben wir eine "16-bit" Machine
- .....4 Bytes.....32 Bit ....."32-Bit"..
- .....8 Bytes.....64 Bit ....."64-Bit"..

Arithmetic

- Beispiel: exp(2,5) = 2.2.2.2.2 = 32
- Beispiel: exp (5,2) = 5.5
- Fakt: exp(2,10) = exp(exp(2,5),2) = exp(32,2) = 1028 = 1K (Kilo....)
- Aufpassen: 1KB = 1 Kilobyte
  aber 1Kb = I Kilobit

Arithmetic

- exp(1K,2) = exp(1024,2) = 1M
- 1M.1K = 1G

Address Space

- 8-Bit Maschine: exp(2,8) = 256 Wörter
- 16-Bit: exp(2,16) = exp(2,6).exp(2,10) = 64K
- 32-Bit exp(2,32) = exp(2,2).exp(2,30) = 4.exp(exp(2,10),3) = 4.exp(1K,3) = 4G
- 64-Bit exp(2,64) = exp(4G,2) = enorm gross!

Address Space

- 8-Bit Maschine: klein
- 16-Bit Maschine: kleiner als ein Chip (DRAM)
- 32-Bit Maschine: teuer und nimmt Platz
- 64-Bit Maschine: unbezahlbar

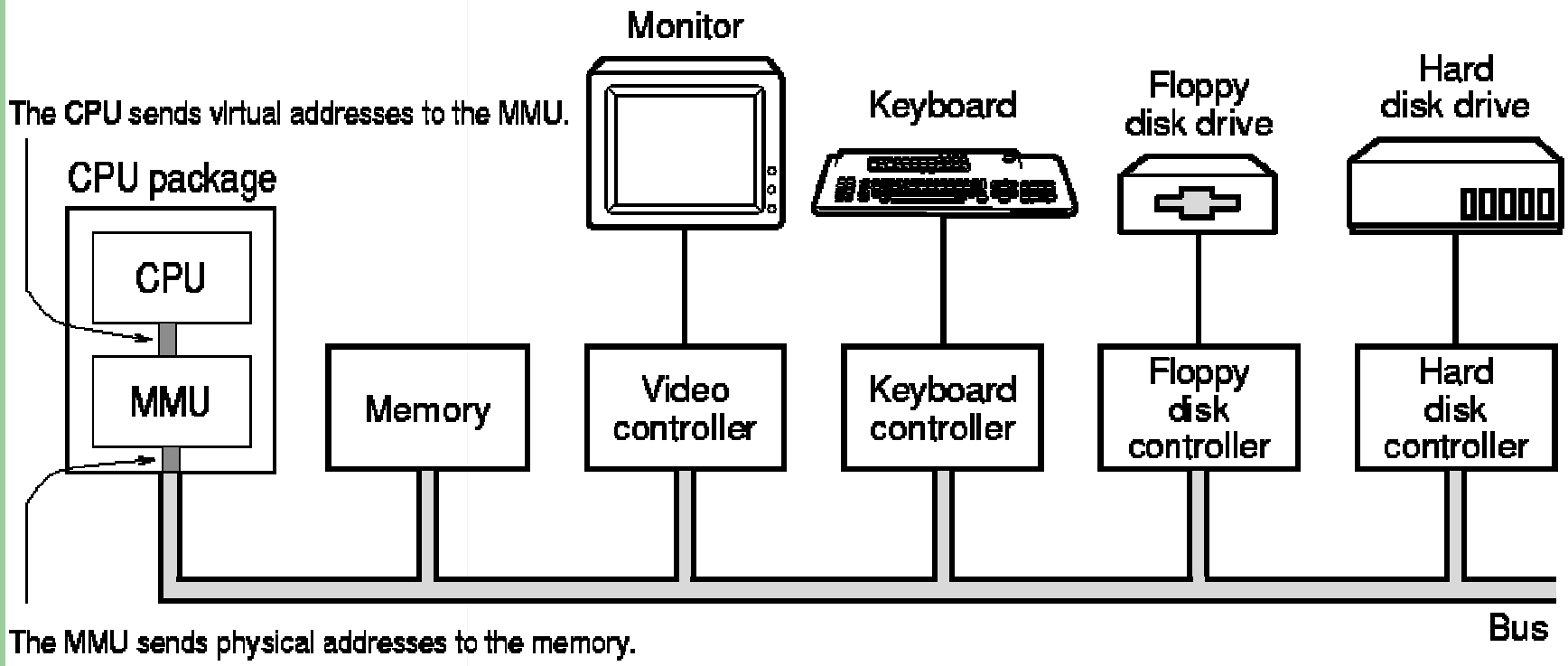
Address Space

- 32-Bit und 64-Bit Maschinen haben (oft) mehr Address Space als physikalischen Speicher
- Dieser Address Space heisst Virtual Memory (Virtueller Speicher)
- Der physikalische Speicher heisst Physical Memory
- Es gibt (oft) viel mehr VM als PM

Aufteilung des VM

- Ein Teil des VM ist in PM
- Der andere liegt auf der Festplatte
- Alle VM wird in Pages aufgeteilt
- Ein Page ist 512B. Oder 1KB. Oder 4KB. Oder..
- Nehmen wir z.B.: ein Page = 4KB

Memory Management Unit (MMU)

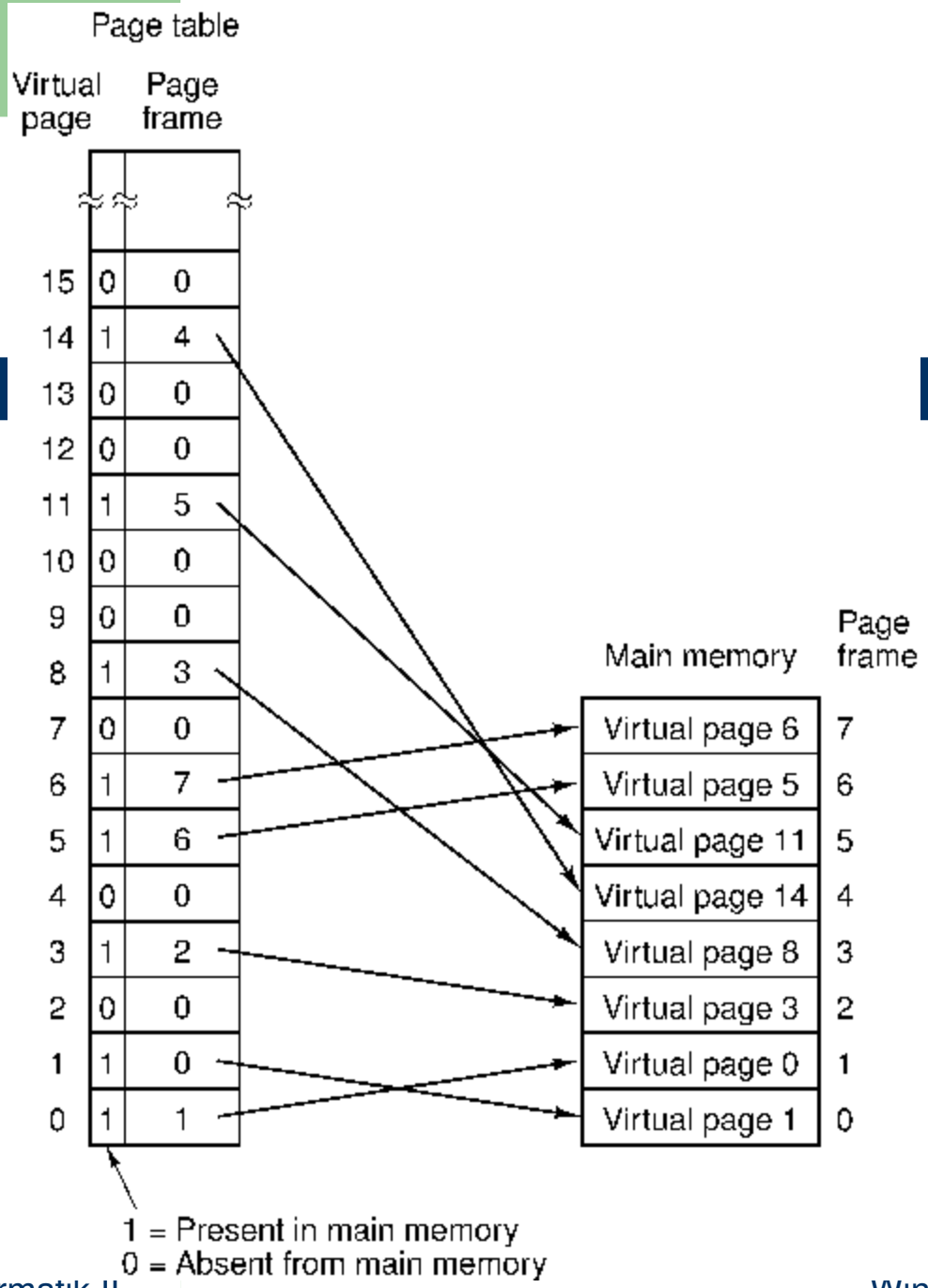


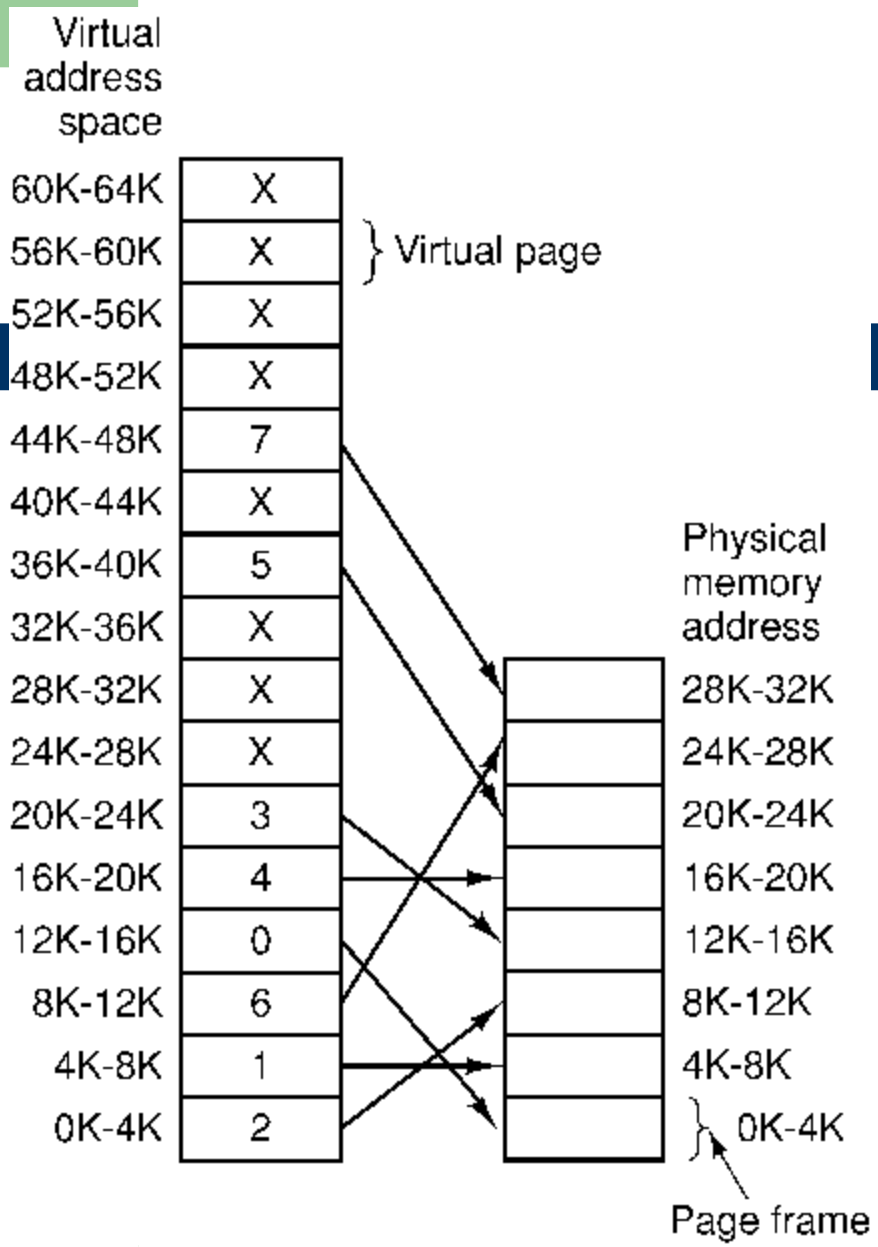
Memory Management

- Der MMU enthält eine Page Table
- Eine Page Table ist eine Datenstruktur
- Eine Page Table ist eine Array
- PT[20] = (MM, 13)
- PT[21] = (Festplatte, <FP-Adresse>)
- <FP-Adresse> ist nur die FP-Adresse in so weit sie in MMU dargestellt wird

VM zu PM

- Behaupten wir: 1 Page = 4K
- PM hat 8 Pages (heutzutage klein!)


Page table
Virtual page
Page frame
15 0 0
14 1 4
13 0 0
12 0 0
11 1 5
10 0 0
9 0 0
8 1 3
7 0 0
6 1 7
5 1 6
4 0 0
3 1 2
2 0 0
1 1 0
0 1 1
Main memory
Page frame
Virtual page 6 7
Virtual page 5 6
Virtual page 11 5
Virtual page 14 4
Virtual page 8 3
Virtual page 3 2
Virtual page 0 1
Virtual page 1 0
1 = Present in main memory
0 = Absent from main memory


Virtual address space
60K-64K
56K-60K
52K-56K
48K-52K
44K-48K
40K-44K
36K-40K
32K-36K
28K-32K
24K-28K
20K-24K
16K-20K
12K-16K
8K-12K
4K-8K
0K-4K
X
X
X
X
7
X
5
X
X
X
3
4
0
6
1
2
Virtual page
Physical memory address
28K-32K
24K-28K
20K-24K
16K-20K
12K-16K
8K-12K
4K-8K
0K-4K
Page frame

VM zu PM

- Pfeile sind "Pointer"
- Wie wird gepointet?
- VM-Address Space = 16 Pages = exp(2,4) Pg
- PM-Address Space = 8 Pages = exp(2,3) Pg
- VM braucht 4 Bit, um die Page zu identifizieren
- PM braucht 3 Bit, um die Page zu identifizieren

VM zu PM

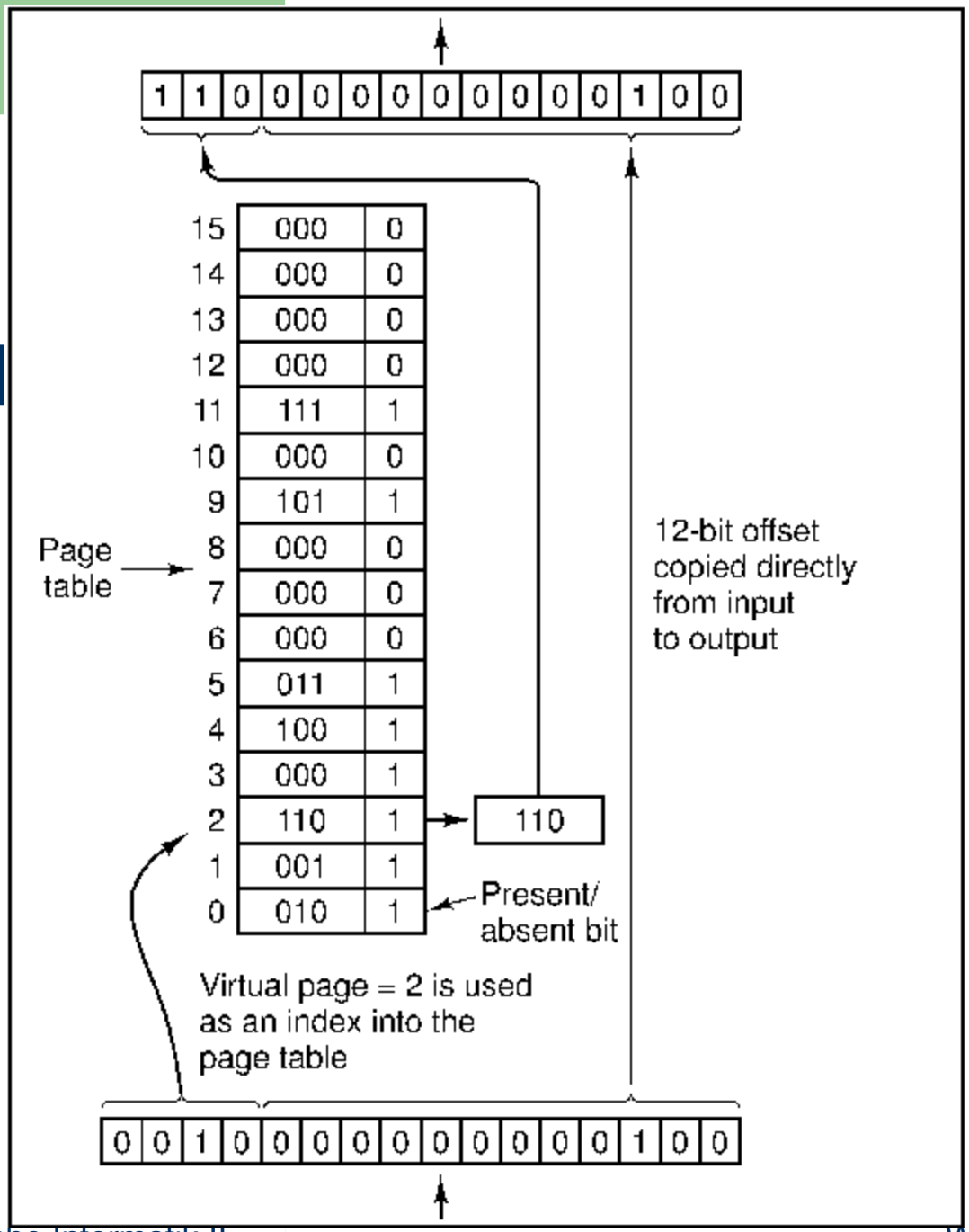
- VM braucht 4 Bit, um die Page zu identifizieren
- PM braucht 3 Bit, um die Page zu identifizieren
- Innerhalb einer Page hat man 4K = exp(2,12) Speicherelemente
- 12 Bit werden gebraucht, den Sp-Element innerhalb der Page zu identifizieren
- Diese 12-Bit Adresse heisst **Offset**

VM zu PM

- 64K Address Space = 16 Pages
- In Page + Offset aufgeteilt
- 4 Bit + 12 Bit
- Die Page wird in den High-Order Bits gegeben
- Der Offset wird in den Low-Order Bits gegeben
- Page in VM heisst Vpage
- Page in PM heisst Ppage

VM zu PM

- VPage + Offset: 4 + 12
- Die 4 werden zum Page-Table Index
- Der Wert in diesem Platz ist 3-Bit PPage
- PPage mit Offset zusammengestellt
- PPage + Offset an PM weitergegeben


1 1 0 0 0 0 0 0 0 0 0 0 1 0 0
Outgoing physical address (24580)
15 000 0
14 000 0
13 000 0
12 000 0
11 111 1
10 000 0
9 101 1
8 000 0
7 000 0
6 000 0
5 011 1
4 100 1
3 000 1
2 110 1
1 001 1
0 010 1
Page table
110
Present/ absent bit
12-bit offset copied directly from input to output
Virtual page = 2 is used as an index into the page table
0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0
Incoming virtual address (8196)

VM zu PM

- Es gibt ein Anwesend/Abwesend Bit
- "Present/Absent Bit"
- Dies bezeichnet, ob die Page in PM liegt (set) oder auf der Festplatte (not set)
- Dies unterscheidet Page 000 (Wert 000, Bit set) von Page Abwesend (Wert 000, Bit not set)

VM zu PM

- Falls Page Absent, wird 4-Bit VM Page-Nummer weitergegeben
- Ein Algorithmus wählt Platz (1 Page) in PM aus
- Die Page wird von der Festplatte in den PM gebracht
- Die Page Table wird modifiziert
- Die Adresse wird wie vorher übersetzt

VM zu PM: Paging Algorithmus

- Page-Austausch Auswahl
  - Hat die Page Sonderprivilegien (z.B. Key-Table?)
  - Ist die Page benutzt worden?
  - Falls ja, ist sie modifiziert worden?

Paging Algorithmus

- Sonderprivilegien
  - Eine Schlüsseltabelle für ein Mehr-Pages-DB
  - Eine Speicher-Verwaltungs-Page für den Benutzerraum
  - Benutzer-ausgewählt als immer anwesend
    - Sonderseite für Programm-Notfälle
    - Heftige Benutzung folgt lange Pause

Paging Algorithmus

- Page referenziert?
  - Daten schon benutzt worden
  - Zeichen, dass sie weiter benutzt wird - könnte zu "Working Set" des Programmes gehören
  - Wenn Page ausgeswappt und zu Working Set gehört, muss sie bald wieder in den Speicher kommen
  - Ineffizient, wenn es so passiert

Paging Algorithmus

- Page modifiziert?
  - Falls nein, die neue Page könnte sie überschreiben ohne dass sie vorher wieder in die Festplatte gespeichert werden muss
  - Falls ja, muss die in die Festplatte wieder gespeichert werden, bevor die neue Page in den Platz eingeholt werden kann

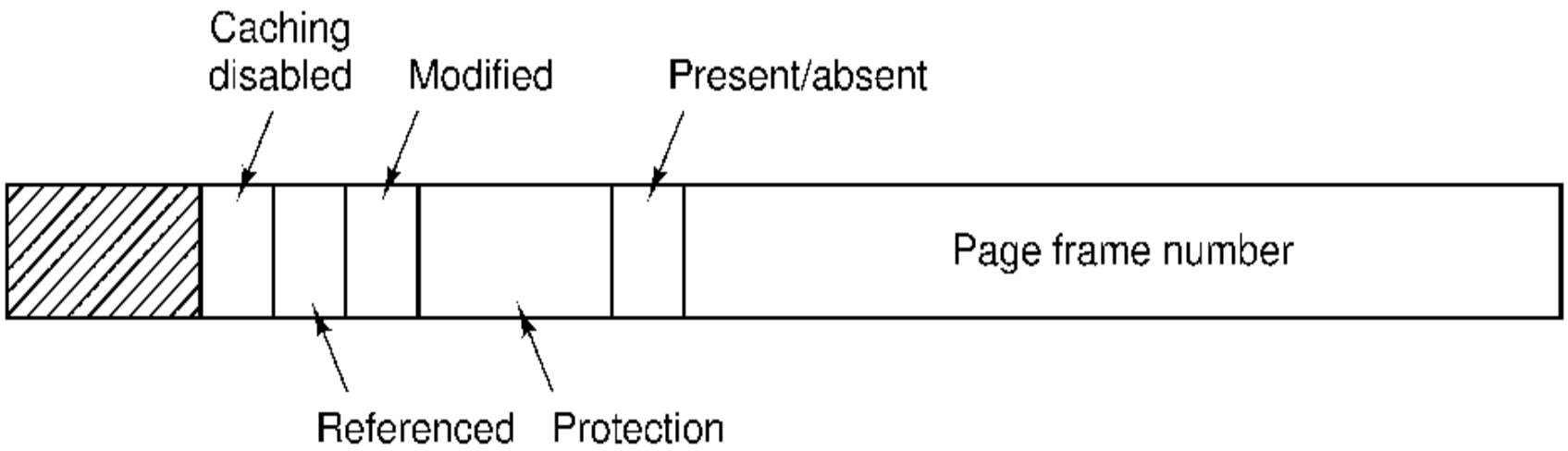
Paging Algorithmus

- "Working Set" (Peter J. Denning)
  - Jeder Zeit im Lauf eines Programmes gibt es eine Menge von PM-Adressen, die häufig in diesem Zeitraum benutzt werden
  - diese Menge beliebt jeder Zeit im PM-Adressenraum relativ lokalisiert
  - Die Menge "bewegt sich" im Lauf des Programmes innerhalb des Adressenraumes

Paging Algorithmus

- Basis: Festplatte-Speicher-Austausch ist zeitlich ineffizient (Zeit-Ressource-Verschwendung)
- Versuch: Ineffizienz so weit wie möglich zu vermeiden

Page-Table-Eingabe mit Extra Bits



Paging Algorithmus (einfachere)

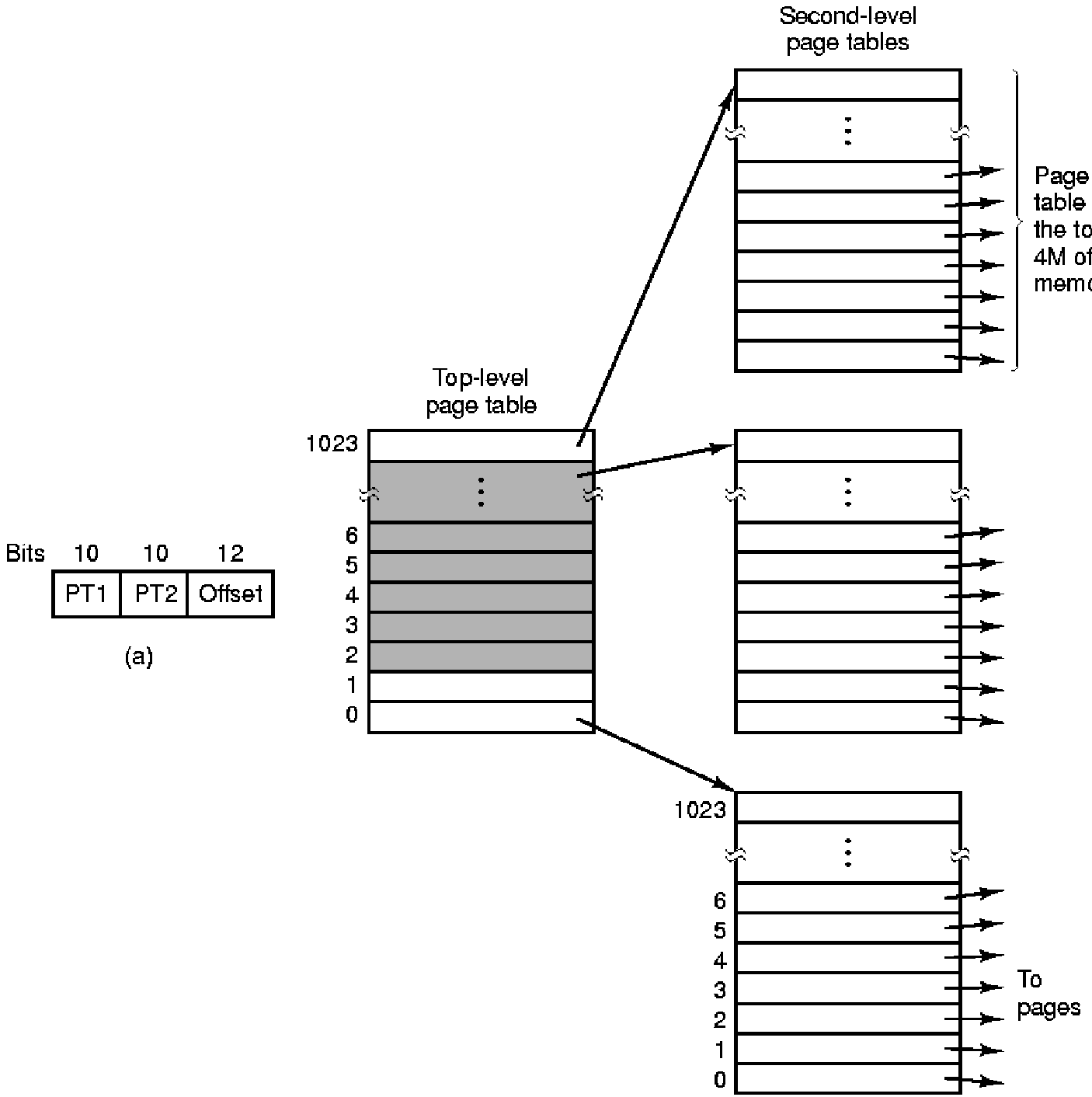
- Versuchen, die Working Set zu identifizieren
- Working Set Pages sind "sticky", falls weniger WS-Pages als PM-Pages
- Vom Rest, versuchen referenzierte aber nicht modifizierte Pages zu identifizieren
- Falls es keine mehr gibt, Page in Festplatte speichern, neue Page in die Stelle anholen

Paging Algorithmus

- Zusätzliche Datenstrukturen gebraucht
  - Ref not Mod Liste
  - Mod Liste
  - WS Menge (Schätzung)
- Effiziente Behandlung der Datenstrukturen
- Experimentale Informatik

32-Bit Adressenraum

- Page Tables hierarchisch arrangiert
- Zwei Ebene (Levels of Indirection)
- PT1-Werte sind Page-Nummer in PT2
- Adressen in 10 + 10 + 12 Bits aufgeteilt
  - Die High-Order 10 werden in PT1 indiziert
  - In dem von PT1 indizierten PT2 Tabelle stehen PM-Adressen


Second-level
page tables
Page
table for
the top
4M of
memory
Top-level
page table
1023
6
5
4
3
2
1
0
Bits 10 10 12
PT1
PT2
Offset
(a)
1023
6
5
4
3
2
1
0
To
pages

Übungen

- (Be)schreib ein Programm, das 4 Variablen hat, und davon die WS immer zwei beinhält, und jede Variabel irgendwann in der WS auftaucht
- (Be)schrieb ein Programm, das 4 Variablen hat, und davon jede Variabel ist ständig in der WS

Übungen

- In einem Quicksort-Verfahren mit 40 Elementen, wie sieht die WS aus, und wie ändert sich die WS im Lauf des Verfahrens?
- Dasgleiche mit Bubblesort