Übungsblatt 10

10.1 Virtuelles Speichersystem

Nehmen Sie ein virtuelles Speichersystem mit folgenden Eigenschaften an:
• Wortgröße: 4 Bytes
• Seitengröße: 4 KiB
• Virtuelle Adresse: 32 Bits
• Physische Adresse: 32 Bits
• Zweistufige Seitenumsetzung (die Seitentabelle auf der 1. Stufe umfasst genau eine Seite).
• Jeder Seitentabelleneintrag (page table entry, PTE) muss an einer Wortgrenze beginnen (word-aligned sein).
• 4 Bits Zusatzinformationen (valid, dirty, reference, protection) pro Seitentabelleneintrag.
• Es gibt keinen TLB (translation look-aside buffer).

a) Geben Sie die Größe des virtuellen bzw. physischen Adressraums in GiB an.

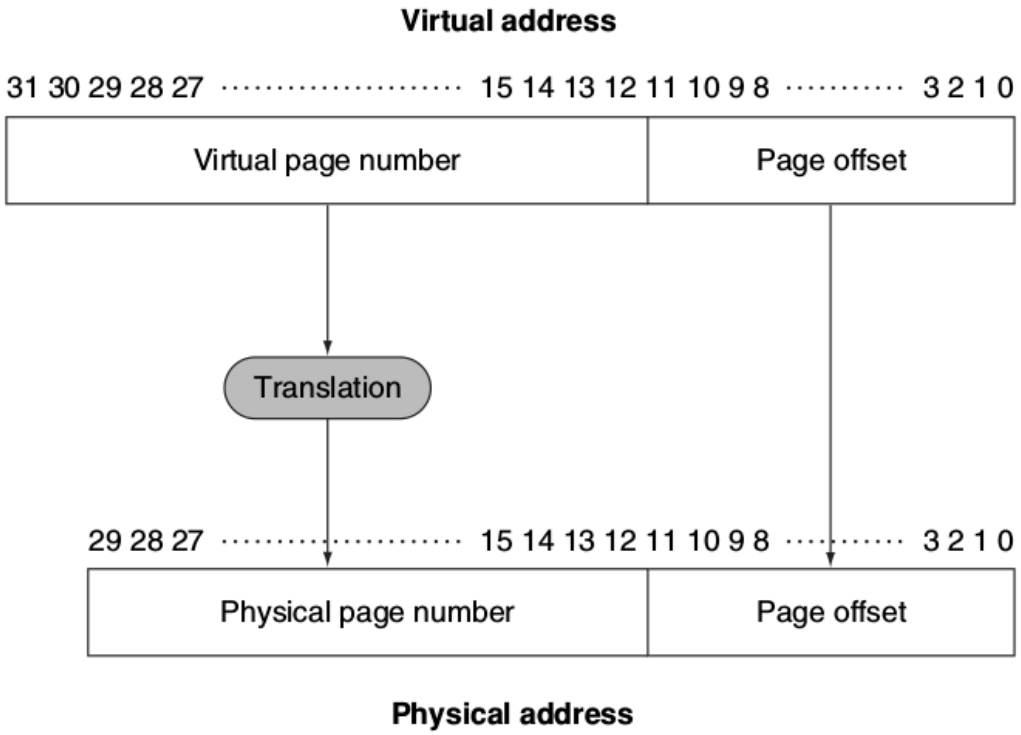


FIGURE 5.26 Mapping from a virtual to a physical address. The page size is $2^{12} = 4$ KiB. The number of physical pages allowed in memory is 2^{18}, since the physical page number has 18 bits in it. Thus, main memory can have at most 1 GiB, while the virtual address space is 4 GiB.

Der Adressraum ist abhängig von den jeweiligen Adresslängen.

Virtueller Adressraum	2^32 = 4.294.967.296 ~~ 4GiB
Physischer Adressraum	2^32 = 4.294.967.296 ~~ 4GiB

b) Bestimmen Sie die Größe eines Seitentabelleneintrags (page table entry, PTE) und die Anzahl der Einträge pro Seitentabelle. Leiten Sie daraus ab, wie die Bits der virtuellen Adresse 0x12345678 interpretiert werden (Indizes, Page Offset).
Virtual Addresses point towards physical addresses/page frames or disk-Address.

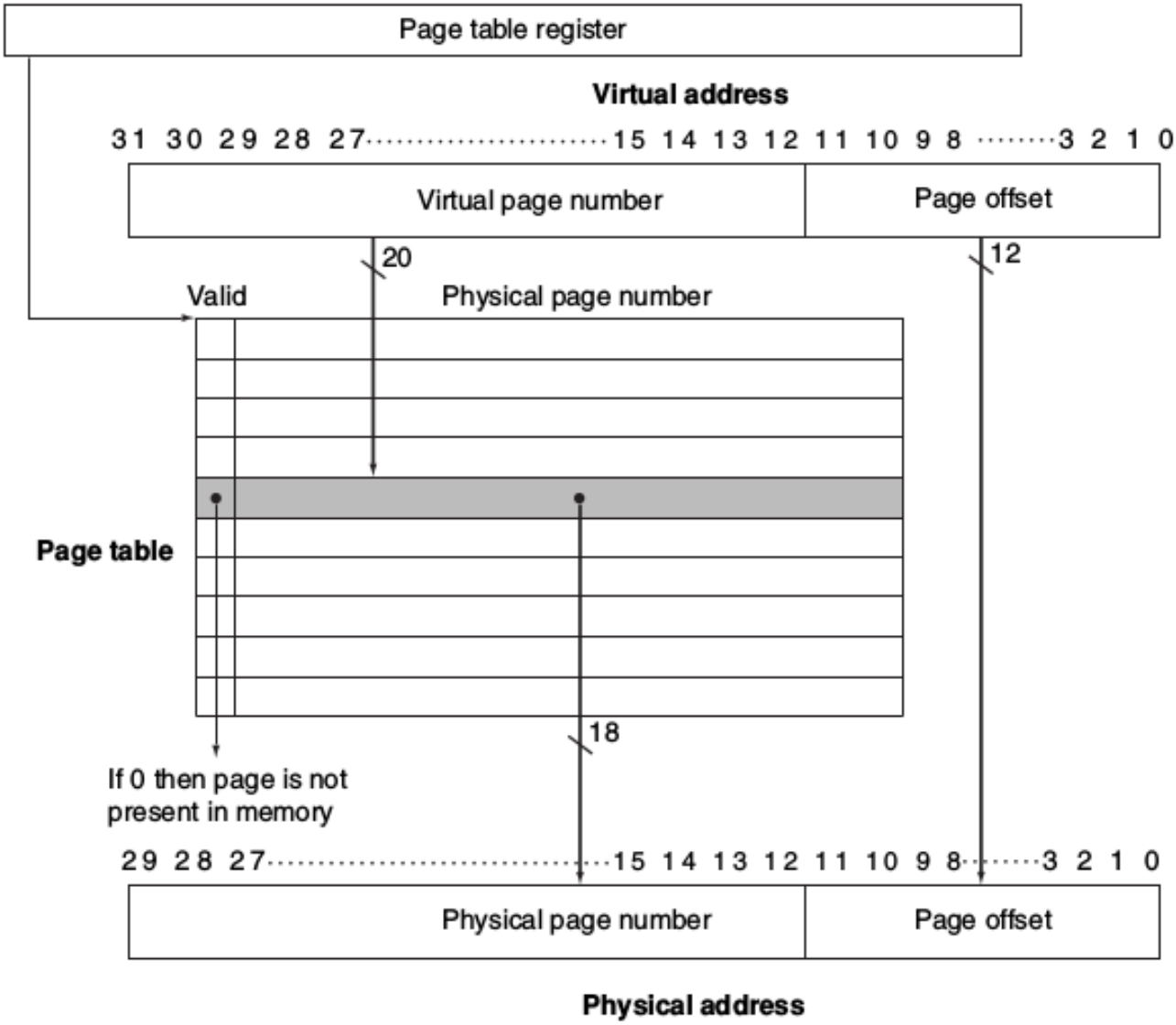


FIGURE 5.27 The page table is indexed with the virtual page number to obtain the corresponding portion of the physical address. We assume a 32-bit address. The page table pointer gives the starting address of the page table. In this figure, the page size is 2^{12} bytes, or 4 KiB. The virtual address space is 2^{32} bytes, or 4 GiB, and the physical address space is 2^{30} bytes, which allows main memory of up to 1 GiB. The number of entries in the page table is 2^{20}, or 1 million entries. The valid bit for each entry indicates whether the mapping is legal. If it is off, then the page is not present in memory. Although the page table entry shown here need only be 19 bits wide, it would typically be rounded up to 32 bits for ease of indexing. The extra bits would be used to store additional information that needs to be kept on a per-page basis, such as protection.

=> #PTE = 2^#Bits of Virtual Adress(32) / 2^#Pages (4KiB)
= 2^32 / 2^12
= 2^20

=> Seitengröße berrechnet sich aus mehreren Parametern. (Abb.: 5.27)

1. Visual Page Number = 32 – Page Offset = 20 Bit
2. Page Offset := ld(Seitengröße) → Seitengrö0e) 4KiB = 2^12 → Offset = 12 Bit.
3. Zusatzinformationen! 4 Bit.
   → VPN + 4 Bit = 24 Bit
4. “Jeder Seitentabelleneintrag (page table entry, PTE) muss an einer Wortgrenze beginnen (word-aligned sein).”

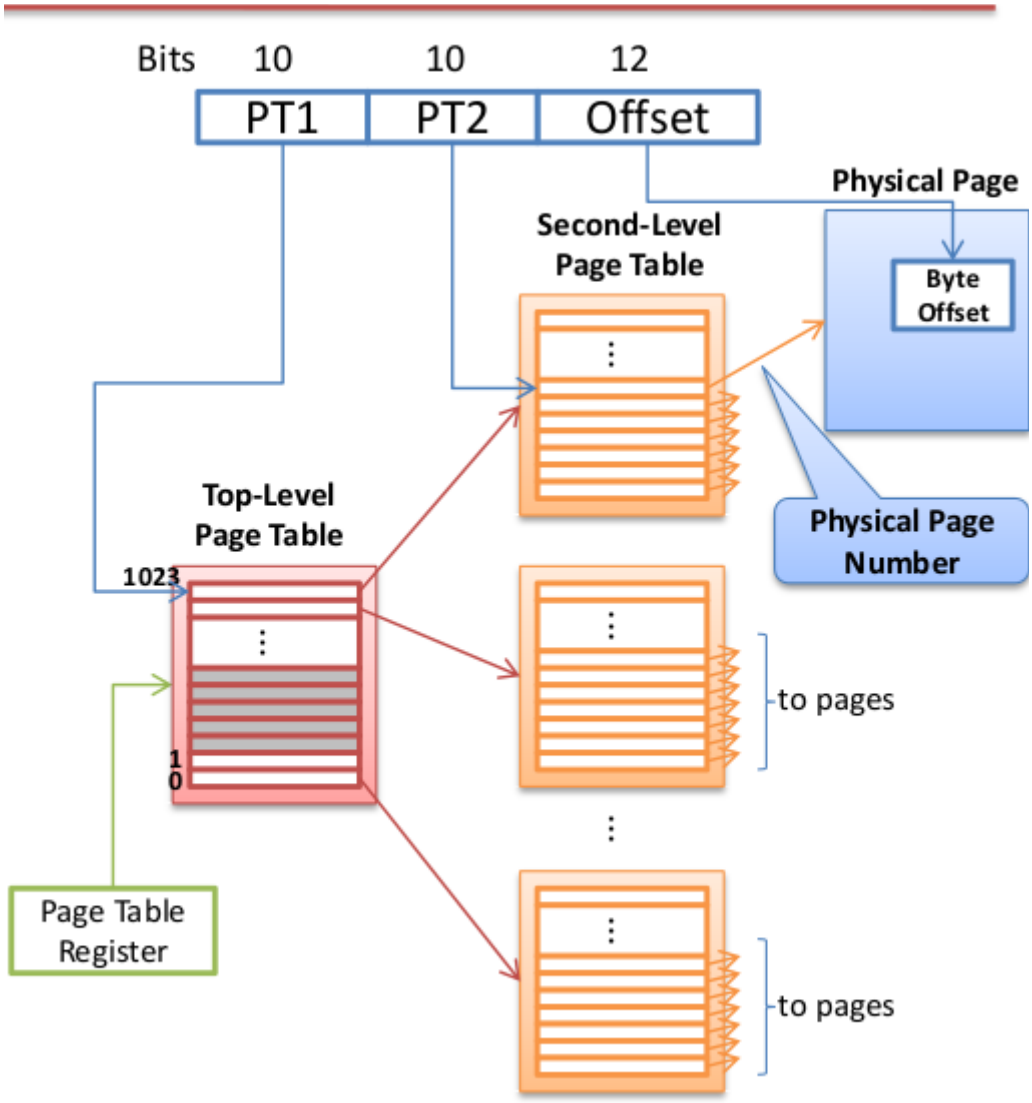
→ Es braucht ein Padding um auf 32 Bit zu kommen.
24 Bit + 8 Bit Padding = 32 Bit.

→ Ein Page Table Entry ist damit 32 Bit oder 4 Byte breit.

=> Anhand der Seitengröße von 4 KiB passen:
4KiB / 4B = 1024 Einträge
in das erste Level: PT1

=> Die Indexierung des PT1 errechnet sich aus ld(#Einträge) = ld(1024) ~ 2^10 = 10 Bit.

Wir haben eine 2-stufige Seitenumsetzung:



Der Second-Level Page Table, PT2 besitzt die gleichen Parameter wie PT1.
– "PTE stores the physical Page number" if page is present in memory.
– 32-Bit Virtual address partiitoned into
10-bit PT1 → Index to top-level page table ↔ Contains page number for second-level
10-bit PT2 → Index to second-level page table ↔ Physical page number
12-bit Offset → Byte offset within physical page

für die Adressberechnung von **der virtuellen Adresse 0x12345678** betrachtet man nun die Bitstellen.

1	2	3	4	5	6	7	8
0001	0010	0011	0100	0101	0110	0111	1000
	0001001000			1101000101		011001111000	
	PT1			PT2		OFFSET	

Die Addressierung in PT1 ist folglich:

PT1 = 00 0100 1000 = 0x048
PT2 = 11 0100 0101 = 0x345
Offset = 0110 0111 1000 = 0x678

c) Stellen Sie die Umsetzung einer virtuellen Adresse in eine physische Adresse grafisch dar.

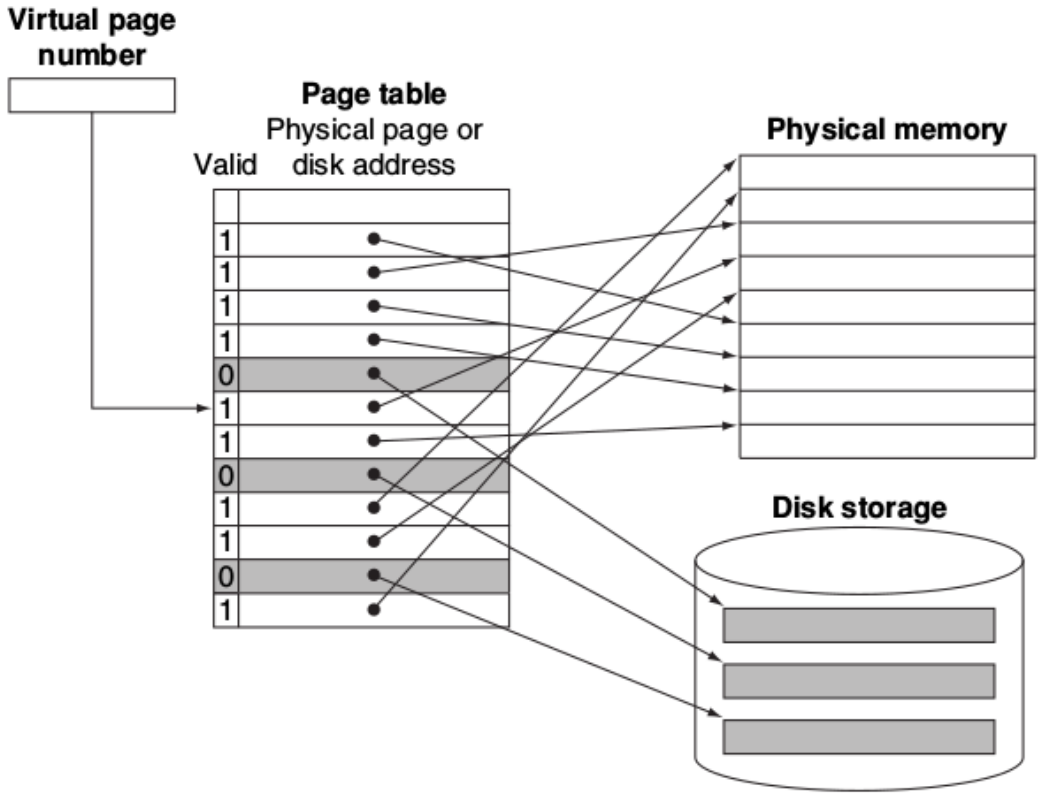
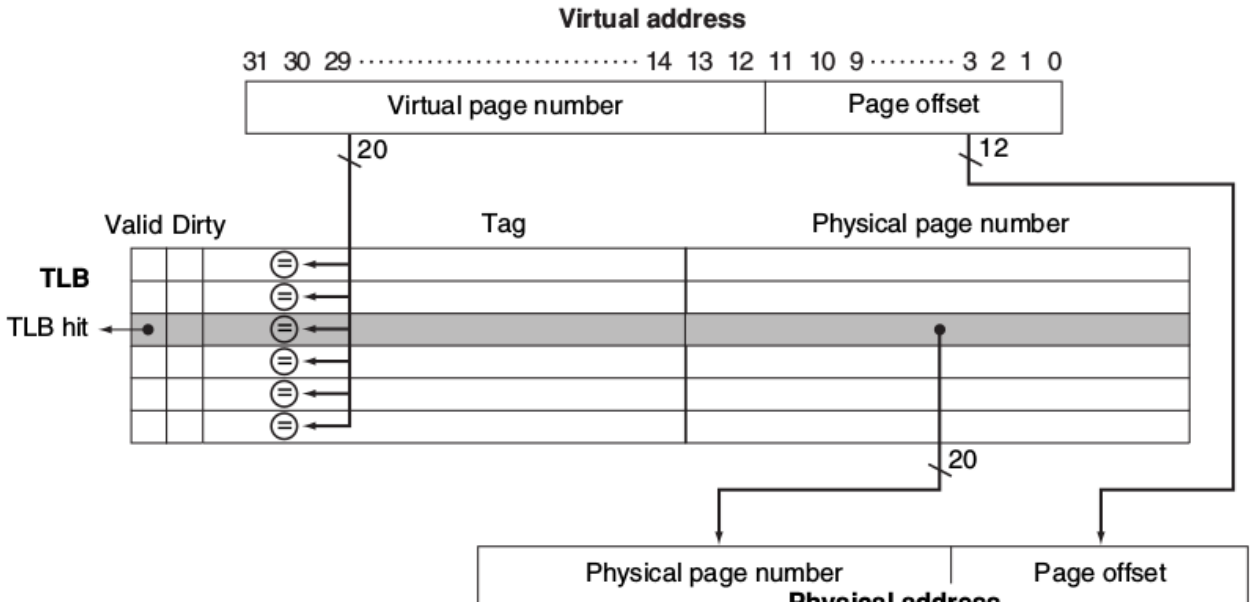


FIGURE 5.28 The page table maps each page in virtual memory to either a page in main memory or a page stored on disk, which is the next level in the hierarchy. The virtual page number is used to index the page table. If the valid bit is on, the page table supplies the physical page number (i.e., the starting address of the page in memory) corresponding to the virtual page. If the valid bit is off, the page currently resides only on disk, at a specified disk address. In many systems, the table of physical page addresses and disk page addresses, while logically one table, is stored in two separate data structures. Dual tables are justified in part because we must keep the disk addresses of all the pages, even if they are currently in main memory. Remember that the pages in main memory and the pages on disk are the same size.

10.2 Translation Look-aside Buffer (TLB)

Das virtuelle Speichersystem aus Ü 10.1 soll durch einen 4-fach satz-assoziativen (4-way set-associative) TLB mit insgesamt 128 Einträgen erweitert werden.

a) Stellen Sie die Umsetzung der virtuellen in physische Adressen mit Hilfe des TLB grafisch dar und geben Sie die Breite aller Felder und Signale an.
=> 128 Einträge zu je 4' Sätzen → 32 Einträge mit je 4 Sätze



→ Visual Page Number = 32 – Page Offset = <u>20 Bit</u>
Accounts for each virtual and physical page
→ Page Offset := ld(Seitengröße) → Seitengrö0e) 4KiB = 2^12 → Offset = <u>12 Bit</u>.
→ Zusatzinformation: 4 Bit

b) Welche Bits einer virtuellen Adresse werden vom TLB als Cache Tag bzw. als Cache Index verwendet?

Indexierung im TLB: 32 Einträge ↔ 0...31 Index.
→ ld(32) = 2^5 = <u>5</u>, 5 Bit des Tags werden für Indexierung benötigt.
→ der Cache Tag entspricht den 20 Bit der Page Number – Indexierungsbits
= 20 – 5 = <u>15 Bits</u>.

c) Aus welchen Feldern besteht ein TLB-Eintrag? Bestimmen Sie die Größe des TLB in Bytes.
TLB besteht aus

Zusatzinformation	Cache-Tag	Physische Seitennr	
4 Bit	15 Bit	20 Bit	… 39 Bit

Mit einem TLB von 32 Einträgen zu je 4 Sätzen, ergeben sich
39 Bit * 128 = 4992 Bit = 624 Byte

Wir greifen nicht mehr auf Speicher zu um Virtuelle/Physische zu übersetzen.

10.3 Virtuelles Speichersystem
Gegeben sei eine byte-adressierbare Maschine mit einem virtuellen Speichersystem mit 42-Bit virtuellem Adressraum, einer Seitengröße von 1 KiB und 34-Bit physischem Adressbereich. Jede Seitentabelle (Page Table) soll gerade eine Speicherseite belegen. Pro Eintrag in der Seitentabelle (Page Table Entry) werden außer der physischen Seitennummer (Page Frame Number) 7 Bit Zusatzinformationen benötigt. Ein Seitentabelleneintrag belege

eine ganze Anzahl von Worten (1 Wort = 4 Bytes).

– 42 Bit Virtueller Adressraum
– 34 Bit physischer Adressraum
– Seitengröße 1 KiB
– 7 Bit Zusatzinformation
– Jeder Seiteneintrage belege eine ganze Anzahl von Worten
– Jede Page Table per Speicherseite

a) Bestimmen Sie die Größe des physischen bzw. virtuellen Speicherbereichs sowie die Größe eines Seitentabelleneintrags.
Der Adressraum ist abhängig von den jeweiligen Adresslängen.

Virtueller Adressraum	2^42 = 549.755.813.888 = 550 Gb
Physischer Adressraum	2^34 = 17.179.869.184 = 17 Gb

###
2^42 Virtuelle
2^42B =2^32 KiB = 2^22 MiB = 2^12 GiB = 4 TiB
2^34 Physische
2^34B = 2^24 KiB = … = 4 GiB

b) Bestimmen Sie nachvollziehbar die minimale Anzahl von Stufen (Levels), die in einer mehrstufigen Seitentabellenhierarchie für die Adressumsetzung benötigt werden.
Generell gilt:

Seitengröße 2^10 – Offset = 10 Bit
1. → 34-10 = 24 Bits für VPN
2. → 7 Bit Zusatzinformation
3. => 1 Bit Padding

<u>Minimale Anzahl Levels</u>
Größe Seitentabelle / Größe PTE = 1 KiB / 32 Bits
= 2^10 B/2^2 B
= 2^8 B

… Bei 2^32 Seiten im Virtuellen Adressraum
=> Gesamtanzahl Seiten = Größe Virtueller Adressraum / Größe Seitentabelle
= 2^42 B / 2^10
= 2^32 Seiten.

Das Problem ist lösbar mit weiteren Stufen:
Gesamtanzahl Seiten <= (Einträge per Seite)^n

N = ceil(log(Gesamtanzahl Seiten) / log(Einträge pro Seite))
= ceil(log(2^32)) / log(2^8))
= ceil(32/8)
= 4 Pages

VA: 40 Bit
→ 30 Bit: Virtuelle Pages => 2^30 PTE
→ 10 Bit Offset, at

→ Berechnung Einträge
1 PTE = 4 Byte
Page Size = 1 KiB
#Page Table Entries / Page Table Size = 2048 / 4
= 512 Einträge

Beginne ganz Links auf der Adresse.
Die ersten 9 Bit zeigen auf Sublevel →
→ PT2 zeigt auf Subsublevel →

9 Bit	9 Bit	9 Bit	3 Bit

→ Direkt errechenbar aus #Einträge in Betracht zur Virtuellen Addresse.
=> 4 Levels, wobei PT4 mit 3 Bit besetzt ist. – Beachte OFFSET von VPN || OFFSET
=> Levels NUR IN VPN

30 #VPN / 9 = 3, … → Aufgerundet = 4

→ Dasselbe für die Physikalische Adresse á PPN || OFFSET

---

c) Auf welche Einträge (Indizes) wird bei Umsetzung der virtuellen Adresse 0x1122334455<<2 (40-Bit-Adresse wird um 2 Binärstellen nach links geschoben bzw. 2 Null-Bits werden angehängt, Ergebnis ist die 42-Bit-Adresse 0x4488CD1154) in eine physische Adresse in den einzelnen Seitentabellen zugegriffen?
Bei 4 Pages:

PT1	PT2	PT3	PT4	Offset
8 Bit	8 Bit	8 Bit	8 Bit	10 Bit
0001 0001	0010 0010	0001 0001	0010 0010	

d) Für einen Prozess muss virtueller Speicher im Ausmaß von 1000 Seiten in physische Adressen übersetzt werden. Wie viel Speicher wird im günstigsten Fall für die Seitentabellen benötigt?
… Eine Seitentabelle = 2^8 = 256 Einträge per Page.

→ Bei 1000/256 Einträgen → 4 Seitentabellen auf Stufe 4.
→ Auf allen weiteren Ebenen wird nur 1 Seite benötigt.
→ Für den Speicherbedarf ergeben sich dadurch 7 KiB.

Klausur: Wie gehabt

Mehrstufige Umsetzung VA → PA
Wir haben eine virtuelle Adresse mit 10 Bit Offset und 30 Bit Virtuelle Page Number.

30 Bit VPN	10 Bit Offset

Wir haben pro Page Table 512 Einträge (1Seite = 1 Page Table)
→ Nehme die 30 Bit VPN.
→ Mit ld(512) ergeben sich 9 Bit per Level

Aufteilung wie folgt:

1st Level – 9 Bit	2nd Level – 9 Bit	3rd Level – 9 Bit	3 Bit – Offset	(OFFSET, NOT ACCOUNTED)