Klausur zur Vorlesung

Rechnerarchitektur

Prof. Christian Plessl
Fachgebiet Hochleistungs-IT-Systeme
Universität Paderborn

14.02.2020

- Die Bearbeitungsdauer beträgt für alle Studenten **90 Minuten.** Es sind **alle 5 Aufgaben** zu bearbeiten.
- Es sind keine Hilfsmittel zugelassen.
- Schreiben Sie nicht mit Bleistift oder Rotstift.
- Verwenden Sie kein eigenes Papier. Bei Bedarf bekommen Sie Papier bei der Klausuraufsicht.
- Schreiben Sie auf jedes Blatt (auch auf das Konzeptpapier) in Blockschrift Ihren Namen und Ihre Matrikelnummer.
- Bei mehreren präsentierten Lösungen wird die Aufgabe nicht gewertet! Streichen Sie daher bei Angabe mehrerer Lösungsansätze die nicht zu bewertenden Lösungen durch! Verwenden Sie kein Tipp-Ex.
- Abschreiben und abschreiben lassen oder Hilfe Dritter führt zum Nichtbestehen der Klausur.

Nachname: _____

Vorname: _____

Matrikelnummer: _____

Studiengang: _____

Aufkleber

Aufgabe	**1**	**2**	**3**	**4**	**5**	$\sum$
Punkte	**12**	**18.5**	**21**	**23.5**	**15**	90
Erreicht						

Aufgabe 1 (Multiple Choice) **[12 Punkte]**

Bei den folgenden Fragen können eine oder mehrere Antworten richtig sein. Kreuzen Sie die richtigen Antworten deutlich an. Für jede vollständig korrekt beantwortete Frage gibt es 2 Punkte, ansonsten 0 Punkte.

(a) Welche Aussagen zur MIPS Instruktionssatzarchitektur sind korrekt:

(2 Punkte)

- ☐ Mit dem Sprungbefehl j kann man nicht auf beliebige Stellen im Programm springen
- ☐ Die maximal unterstützte Speichergrösse ist 4TB
- ☐ Alle Argumente müssen beim Funktionsaufruf auf dem Stack abgelegt werden
- ☐ Arithmetische Operationen mit Immediate Operanden sind gleich schnell wie Operationen mit Register Operanden

(b) Wie viele Tag Bits braucht ein 4-fach assoziativer Cache mit 128 Cacheindizes und einer Blockgrösse von 8 32-Bit Worten? Nehmen Sie die aus der Vorlesung bekannte MIPS Architektur an.

(2 Punkte)

- ☐ 20 bits
- ☐ 21 bits
- ☐ 22 bits
- ☐ 23 bits

(c) Welche Eigenschaften zur Daisy-Chain-Arbitrierung sind korrekt:

(2 Punkte)

- ☐ Alle Geräte haben auf lange Sicht einen garantierten und fairen Zugang zum Bus
- ☐ Die Position eines Gerätes am Bus bestimmt seine Priorität
- ☐ Die Datenübertragung muss zwingend synchron erfolgen
- ☐ Die maximale Anzahl der Geräte am Bus muss beim Entwurf des Arbiters nicht bekannt sein

(d) Mit welchen Methoden können Data Hazards aufgelöst werden?

(2 Punkte)

- ☐ Umordnen von Instruktionen
- ☐ Forwarding
- ☐ Dynamische Sprungvorhersage
- ☐ Register mit Halbtaktzugriff

(e) Welche Vorteile hat das statische Pipeline Scheduling gegenüber dem dynamischen Scheduling:

(2 Punkte)

- ☐ Der Prozessorentwurf wird vereinfacht
- ☐ Der Compilerentwurf wird vereinfacht
- ☐ Abhängigkeiten lassen sich für den Compiler einfacher erkennen und vermeiden

(f) Die Anzahl der ausgeführten Instruktionen für ein Programm (I_c) wird beeinflusst durch:

(2 Punkte)

- ☐ den Algorithmus
- ☐ den Instruktionssatz
- ☐ die mikroelektronische Technologie
- ☐ den Compiler

Aufgabe 2 (Assembler) **[18.5 Punkte]**

Gegeben sind zwei Null-terminierte Zahlenfolgen {1, 3, 2, 0} und {2, 1, 1, 6, 0}, referenziert durch $a0 und $a1. Die Zahlenfolgen sollen durch die gegebene Assembler-Prozedur `listOP` verarbeitet werden. Die gestrichelten Linien sind für den Assembler-Code ohne Bedeutung und dienen nur als Markierung für Aufgabenteil (b).

```
01  listOP:  addi  $sp,    $sp,    -8
02           sw    $ra,    4($sp)
03           sw    $s0,    0($sp)
04           #--------------------------
05           add   $v0,    $zero,  $zero
06           lw    $t0,    0($a0)
07           beq   $t0,    $zero,  return
08           lw    $t1,    0($a1)
09           beq   $t1     $zero   return
10
11           sub   $s0,    $t0,    $t1
12           addi  $a0,    $a0,    4
13           addi  $a1,    $a1,    4
14           jal   listOP
15           add   $v0,    $v0,    $s0
16
17  return:  lw    $s0,    0($sp)
18           lw    $ra,    4($sp)
19           addi  $sp,    $sp,    8
20           #--------------------------
21           jr    $ra
```

(a) Wofür wird bei Prozeduraufrufen der Stack verwendet? Wann ist dies notwendig? (2 Punkte)

(b) Betrachten Sie den Aufruf der Assembler-Prozedur `listOP`. Geben Sie für jedes Erreichen einer der gestrichelten Linien den Wert des Stackpointers `$sp`, den Inhalt des Stacks, die Werte von `$ra`, `$s0`, `$v0` und den Speicherinhalt an den Stellen `$a0` und `$a1` an. Als Rücksprungadressen verwenden Sie die Zeilenangaben des Assembler-Quellcodes. Es ist ausreichend, die ersten 7 Stackveränderungen anzugeben. Zur Orientierung sind die ersten beiden Veränderungen bereits gegeben. (10.5 Punkte)

	$sp=156	$sp=148	$sp=____	$sp=____	$sp=____	$sp=____	$sp=____
160	$ra=?	$ra=?					
156	$s0=?	$s0=?					
152		$ra=15					
148		$s0=-1					
144							
140							
136							
132							
mem[$a0]	1	3					
mem[$a1]	2	1					
$ra	?	15					
$s0	?	-1					
$v0	?	0					

Ersatzlösung. Ungültige Lösung streichen.

	$sp=156	$sp=148	$sp=____	$sp=____	$sp=____	$sp=____	$sp=____
160	$ra=?	$ra=?					
156	$s0=?	$s0=?					
152		$ra=15					
148		$s0=-1					
144							
140							
136							
132							
mem[$a0]	1	3					
mem[$a1]	2	1					
$ra	?	15					
$s0	?	-1					
$v0	?	0					

(c) Skizzieren Sie die Funktion von `listOP`, indem Sie den untenstehenden Code vervollständigen. (5 Punkte)

```
int listOP($a0, $a1){
    if((memory[$a0] == 0) or (memory[$a1] == 0)){
       return 0;
    }
    else{

       temp = _____________________________________________

       return _____________________________________________
    }

}
```

Ersatzlösung. Ungültige Lösung streichen.

```
int listOP($a0, $a1){
    if((memory[$a0] == 0) or (memory[$a1] == 0)){
       return 0;
    }
    else{

       temp = _____________________________________________

       return _____________________________________________
    }

}
```

(d) Welchen Rückgabewert liefert `listOP`? (1 Punkt)

Aufgabe 3 (Erweiterung Mehrzyklenimplementierung) **[21 Punkte]**

Die aus der Vorlesung bekannte MIPS-Mehrzyklenimplementierung soll um die Instruktion `jri` erweitert werden, welche einen unbedingten Sprung an eine Adresse, die aus dem Hauptspeicher geladen wird, ausführt. Die Speicheradresse, an der das Sprungziel hinterlegt ist, ergibt sich aus dem Inhalt von Register `$t0` sowie einem Wort-Offset `imm`:

```
jri imm($t0)
```

(a) Geben Sie zunächst eine mögliche Implementierung von `jri` aus bekannten MIPS-Befehlen an: (5 Punkte)

	ERSATZ:

Nun wird die Instruktion `jri` direkt in der Architektur implementiert:

(b) Geben Sie einen geeigneten Instruktionstyp für `jri` an: (1 Punkt)

Instruktionstyp: _____

(c) Geben Sie die Instruktions-Kodierung für `jri` mit dem von Ihnen gewählten Instruktionstyp an. Tragen Sie die Bestandteile des Befehls ein. Nicht benötigte Bits können als X (don't care) gekennzeichnet werden. (3 Punkte)

31 26	25 0
Opcode	

(d) Ergänzen Sie den Datenpfad und die Steuersignale in Abb. 1, um die Ausführung von `jri` zu ermöglichen. (6 Punkte)

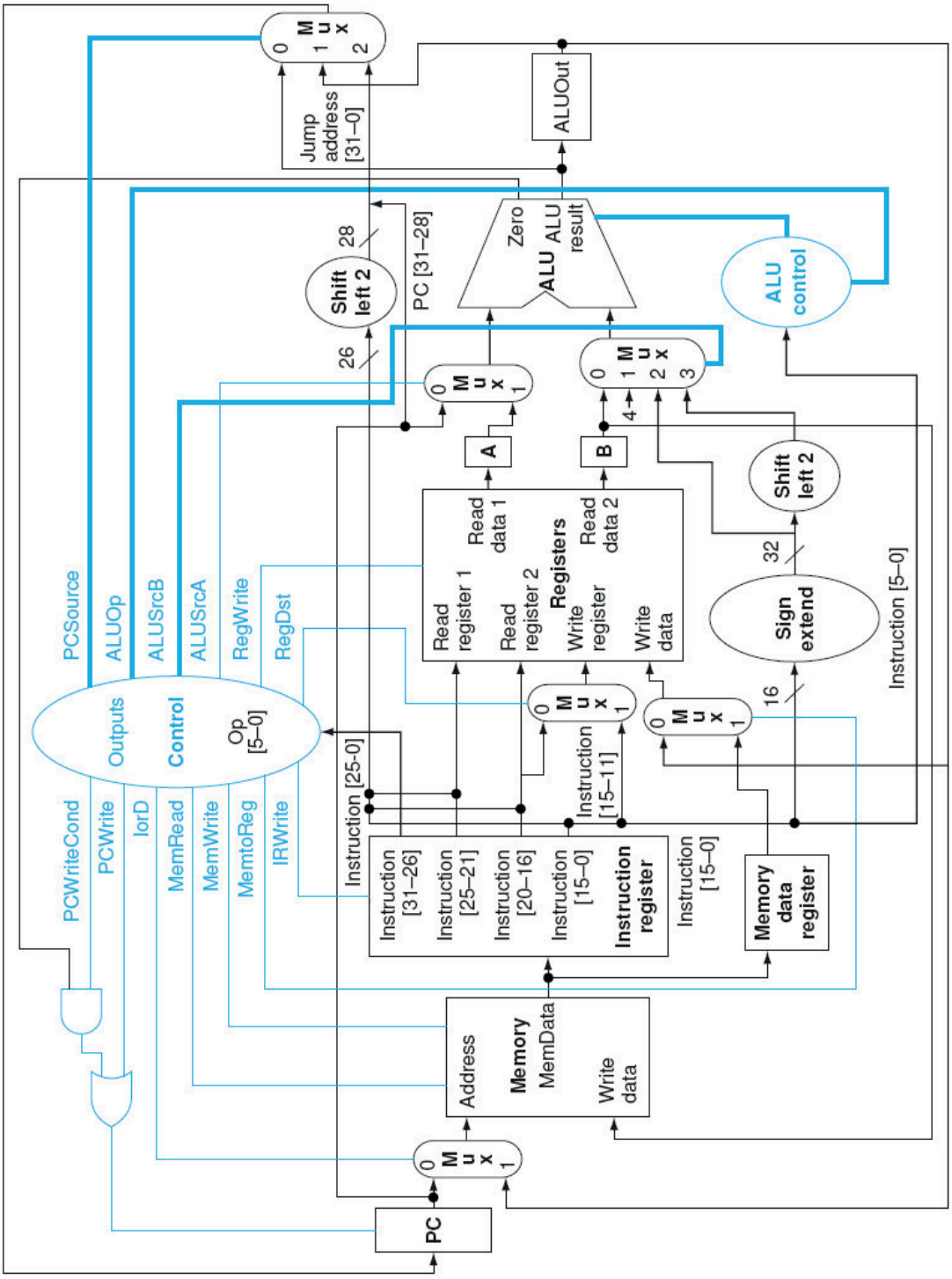
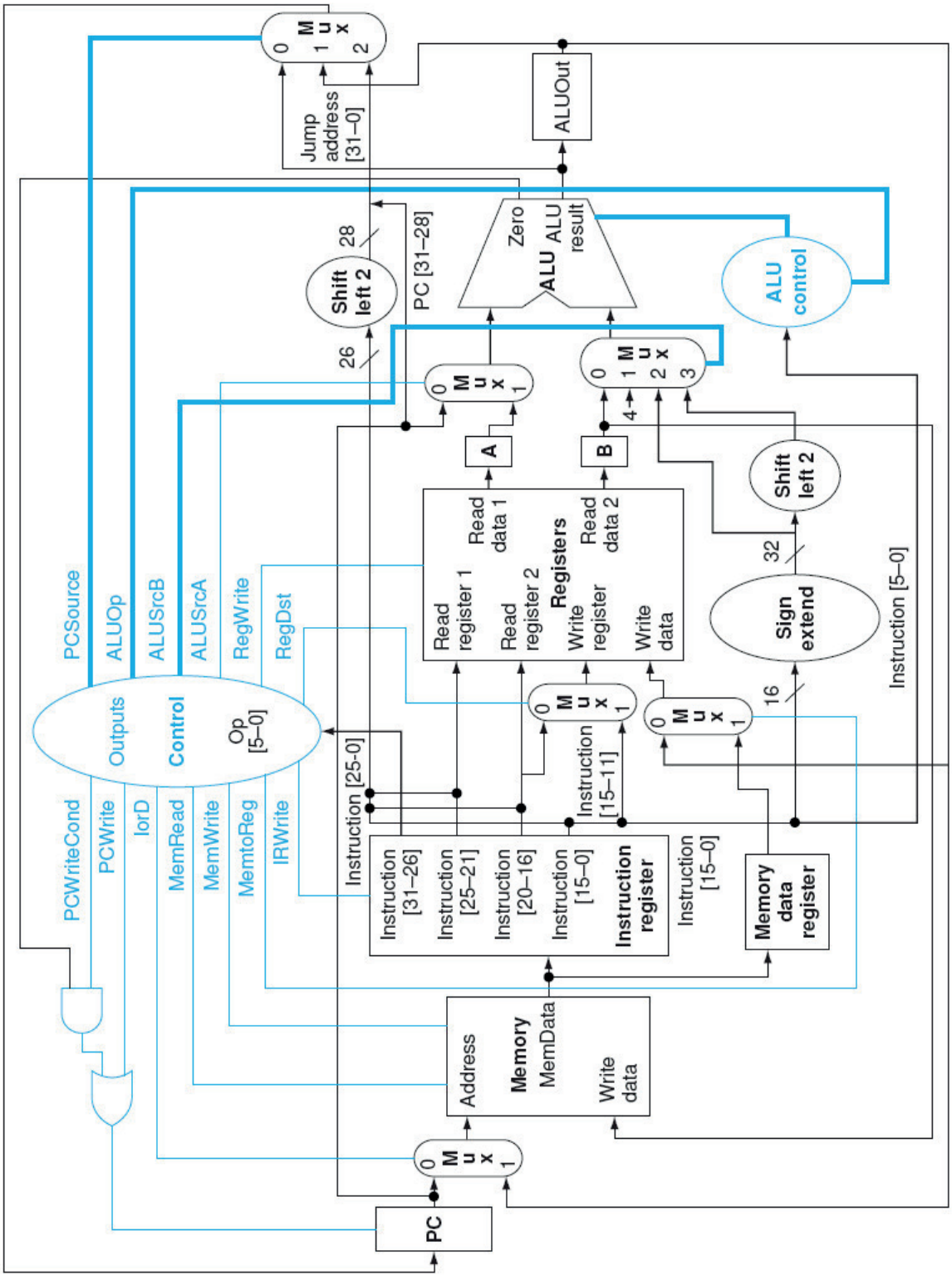


Abbildung 1: Datenpfad

ERSATZ - ungültige Lösung streichen!



(e) Geben Sie die Steuersignale des Controllers bei Ausführung von `jri` auf dem von Ihnen im vorherigen Aufgabenteil erstellten Datenpfad in Form des aus der Vorlesung bekannten Automaten an. Nennen Sie zudem den resultierenden CPI-Wert der `jri`-Instruktion. (6 Punkt)

CPI-Wert: _____

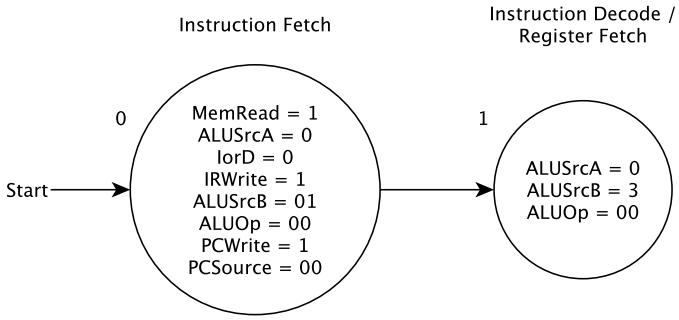
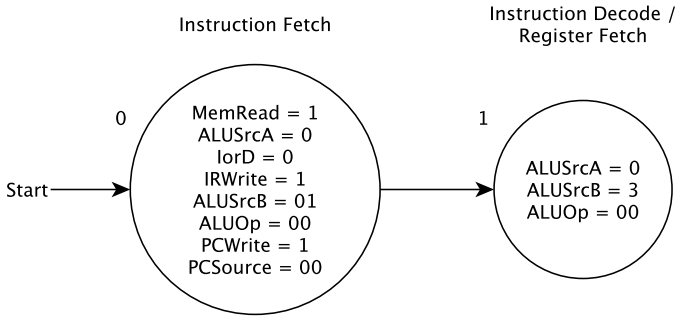


Abbildung 2: Controller-Automatengraph



Controller-Automatengraph
ERSATZ - ungültige Lösung streichen!

Aufgabe 4 (Cache / Prozessorperformance) **[23.5 Punkte]**

Tabelle 1 vergleicht drei Prozessoren. Diese unterscheiden sich lediglich in Taktfrequenz und Cacheorganisation. Die Cache miss-penalty beträgt für alle drei Systeme 4+(Blockgröße [Worte]) Zyklen (Blockgröße = 2^m).

Prozessor	T	Cacheorganisation	instruction miss-rate	data miss-rate
P1	400 ps	direkte Abbildung, $m = 0$	4%	6%
P2	300 ps	direkte Abbildung, $m = 4$	3%	5%
P3	350 ps	3-fach satzassoziativ, $m = 3$	1%	5%

Tabelle 1: Vergleich dreier Prozessoren

Für folgenden Workload wurde am Prozessor P1 ein CPI von 2,56 gemessen.

```
1  pseudo_bench: addi $a0, $zero, 2
2                addi $a1, $zero, 2
3                li $v0,10
4                subi $sp, $sp, 4
5                sw $ra, 0($sp)
6                addi $v0, $a1, 1
7                li $a1, 1
8                move $s0, $a0
9                lw $ra, 0($sp)
10               addi $sp, $sp, 4
```

(a) Bei welchen Instruktionen handelt es sich bei diesem Workload um Datenzugriffe? Geben Sie alle Zeilennummern an. (2 Punkt)

(b) Wie lautet die miss-penalty in Zyklen für die drei Systeme? (1.5 Punkt)

(c) Wie lange warten die Prozessoren durchschnittlich bei dem gegebenen Workload auf den Speicher? Errechnen Sie die Memory Stall Cycles. (12 Punkt)

(d) Wie lautet die durchschnittlich benötigte Ausführungszeit für den gegebenen Workload? (8 Punkt)

Aufgabe 5 (Virtueller Speicher) **[15 Punkte]**

Gegeben sei ein virtueller Speicher der Größe 1 GB und ein physikalischer Adressraum der Größe 256 MB. Sowohl die virtuellen Seiten als auch die physikalischen Seitenrahmen seien 32 MB groß. Der Speicher sei byteadressiert.

(a) Wie viele Bit werden für die phyiskalische und wie viel Bit werden für die virtuelle Adresse benötigt? Wie viele Bit werden für den Seitenoffset benötigt? (3 Punkte)

(b) Welche Größe hat die benötigte page table bei einer virtuellen Adresse mit 32 Bit und einer Seitengrößen von 16 KB mit 4 Byte pro Eintrag. (3 Punkte)

(c) Erläutern Sie die Ersetzungsstrategie LRU. (3 Punkte)

(d) Gegeben sei nun eine 32 Bit lange virtuelle Adresse. Die virtuelle Seitennumer sei 3 Bit lang und die physikalische Seitennummer sei 2 Bit lang. Gegeben sei außerdem die in Tabelle 2 gezeigte page table. Die Spalten geben die virtuelle Seitennummer (virt.), die physikalische Seitennummer (phy.), das valid Bit (v), das dirty Bit (d), sowie das reference Bit (r) an.

virt.	phy.	v	d	r
000	00	0	0	0
001	00	0	0	0
010	11	1	0	0
011	10	0	0	0
100	01	0	0	0
101	01	0	0	0
110	01	0	0	0
111	10	0	0	0

Tabelle 2: Page table

Führen Sie die Zugriffe auf die folgenden virtuellen Adressen in der angegebenen Reihenfolge durch. Geben Sie die page table nach jedem Zugriff an. Nutzen Sie hierfür das Lösungsschema in Tabelle 3. (6 Punkte)

Hinweis: Das reference Bit wird während der 4 Zugriffe **nicht** zurückgesetzt.

1.) Lesend: 0b 0000 1101 1000 1100 0010 1011 0000 0001
2.) Schreibend: 0b 0110 0010 1010 0001 0000 1111 1011 1000
3.) Lesend: 0b 1011 0100 0001 0011 0001 1000 1111 1010
4.) Schreibend: 0b 0101 1111 1110 1011 0000 0000 0000 1011

Zugriff: 1

virt.	phy.	v	d	r
000				
001				
010				
011				
100				
101				
110				
111				

Zugriff: 2

virt.	phy.	v	d	r
000				
001				
010				
011				
100				
101				
110				
111				

Zugriff: 3

virt.	phy.	v	d	r
000				
001				
010				
011				
100				
101				
110				
111				

Zugriff: 4

virt.	phy.	v	d	r
000				
001				
010				
011				
100				
101				
110				
111				

Tabelle 3: Page table. (Unten Ersatz: ungültige Lösung streichen)

Zugriff: ____

virt.	phy.	v	d	r
000				
001				
010				
011				
100				
101				
110				
111				

Zugriff: ____

virt.	phy.	v	d	r
000				
001				
010				
011				
100				
101				
110				
111				

Zugriff: ____

virt.	phy.	v	d	r
000				
001				
010				
011				
100				
101				
110				
111				

Zugriff: ____

virt.	phy.	v	d	r
000				
001				
010				
011				
100				
101				
110				
111				

Zugriff: ____

virt.	phy.	v	d	r
000				
001				
010				
011				
100				
101				
110				
111				

Zugriff: ____

virt.	phy.	v	d	r
000				
001				
010				
011				
100				
101				
110				
111				

Konzeptpapier: Falls der Platz unter den einzelnen Aufgaben nicht ausreicht, können Sie diese Seiten für Zwischenrechnungen nutzen. Bitte Lösung und Lösungsweg eindeutig mit der Aufgabennummer markieren!