Department Informatik
Lehrstuhl für Informatik 3 (Rechnerarchitektur)
Friedrich-Alexander-Universität Erlangen-Nürnberg
Martensstraße 3, 91058 Erlangen
28.09.2016
Klausur zu

Grundlagen der Rechnerarchitektur und -organisation

.....	.....
Matrikelnummer	Geburtsdatum
.....	.....
Name	Vorname

- Es sind *keine* Hilfsmittel erlaubt!
- Legen Sie den Ausweis (mit Lichtbild!) griffbereit auf den Platz!
- Dieses Aufgabenheft umfasst 20 Seiten. Überprüfen Sie die Vollständigkeit!
- Gesondert beigelegte Blätter werden nicht bewertet.
- Schreiben Sie deutlich! Unleserliches wird nicht bewertet!
- Es darf nicht mit der Farbe rot geschrieben werden!
- Offensichtlich falsche oder überflüssige Antworten können zu Punktabzug führen!
- Begründen Sie Ihre Antworten!

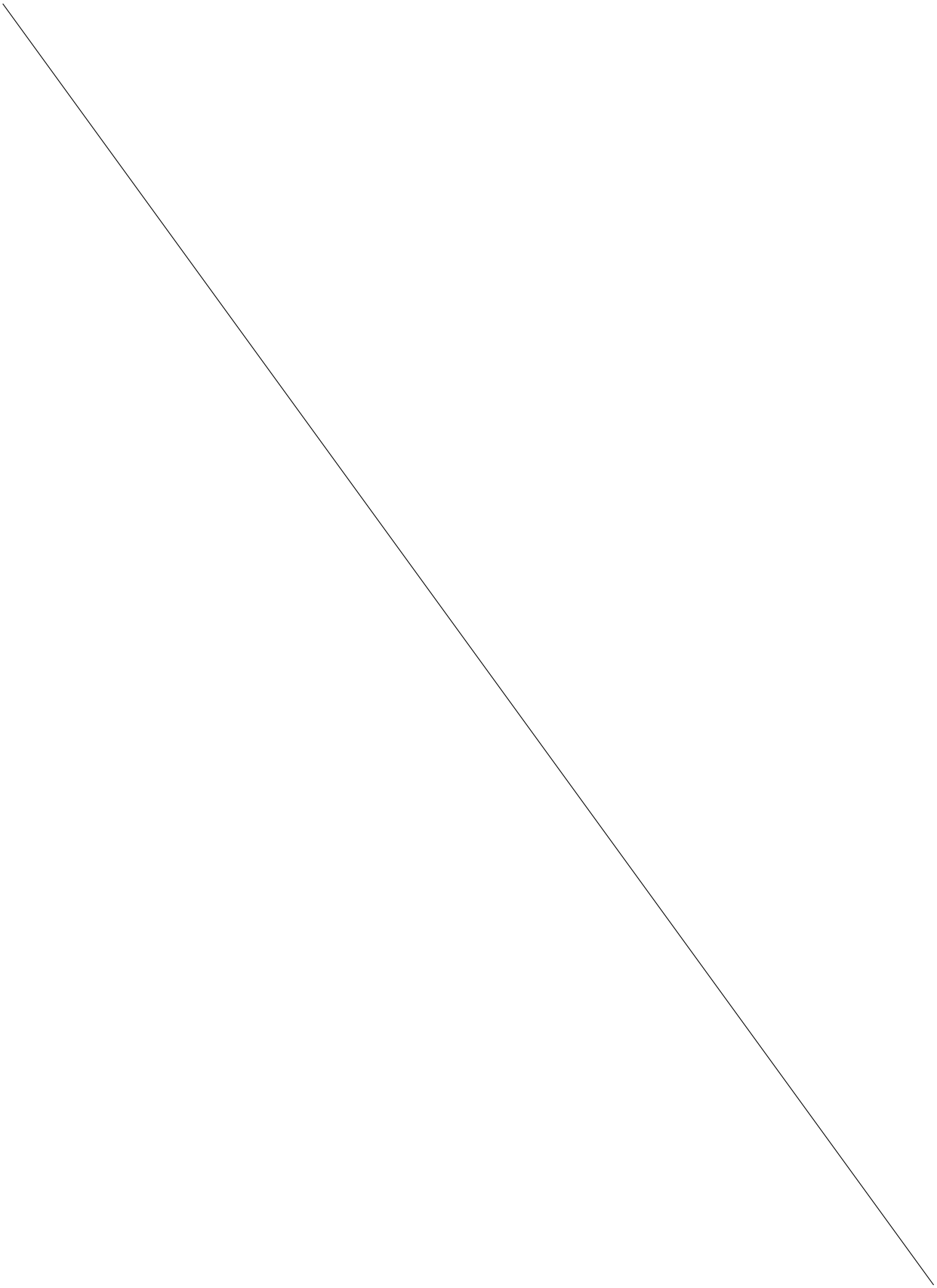
Durch meine Unterschrift bestätige ich

- den Empfang der vollständigen Klausurunterlagen
- die Kenntnisnahme der obigen Informationen.

Erlangen, den 28.09.2016 .....
(Unterschrift)

Aufgabe	1	2	3	4	5	6	7	8	9
max. Punktzahl	10	5	5	9	12	12	6	13	18
erreichte Punktzahl									

Summe	/90		
Bonus	/15		
Gesamt		Note	



Aufgabe 1: Allgemein — 10 Punkte

Welche der folgenden Aussagen sind wahr, welche falsch? Kreuzen Sie an.

Punktabzug bei falschen Antworten!

1. Multi-/Vielkernarchitekturen — 5 Punkte

wahr	falsch	
☐	☐	Superskalarität muss vom Compiler unterstützt werden.
☐	☐	Pipelining führt zu Daten- und Steuerungshazards.
☐	☐	Pipelining und Multithreading schließen sich gegenseitig aus.
☐	☐	Mehr Pipelinestufen führen automatisch zu einer höheren Performance des Prozessors.
☐	☐	Simultanes Multithreading (SMT) erfordert einen komplexeren Prozessoraufbau als Zeitscheiben-Multithreading.
☐	☐	VLIW erfordert Parallelisierung durch Betriebssystem-Threads.
☐	☐	SMT erfordert Parallelisierung durch Betriebssystem-Threads.
☐	☐	Bei SMT können Threads unterschiedlicher Prozesse gleichzeitig auf einem Kern ausgeführt werden.
☐	☐	Eine Dualcore-CPU erfordert mehr Hardware als ein Kern mit 2-fachem SMT.
☐	☐	Superskalarität erfolgt transparent für den Programmierer.

2. Speicher — 5 Punkte

wahr	falsch	
☐	☐	64-Bit Prozessoren können mehr Speicher adressieren als 32-Bit Prozessoren.
☐	☐	Bei DRAM wird die Information als Ladung eines Kondensators gespeichert.
☐	☐	Paging soll Speicherzugriffe beschleunigen.
☐	☐	Durch Segmentierung kann Prozessen mehr Arbeitsspeicher zur Verfügung gestellt werden, als physikalisch vorhanden ist.
☐	☐	Segmentierung erlaubt es, Speicher von Prozessen vor ungewolltem Zugriff zu schützen.
☐	☐	Speicherschutz kann nicht effizient alleine durch das Betriebssystem realisiert werden.
☐	☐	Im TLB werden alle Flags eines Seitentabellen-Eintrags gespeichert.
☐	☐	Bei Segmentierung entscheidet die MMU bei jedem Speicherzugriff, ob der Zugriff gewährt wird oder nicht.
☐	☐	Paging vermeidet externe Fragmentierung.
☐	☐	Segmentierung und Paging erlauben es, mehreren Prozessen gemeinsamen Speicher zur Verfügung zu stellen.

Aufgabe 2: Speicherverwaltung — 5 Punkte

1. Wann ist ein Datum korrekt ausgerichtet (Alignment)? — 1 Punkt

2. Wie viel Platz braucht folgende Datenstruktur auf einer 32-Bit-Architektur, wenn korrektes Alignment verwendet wird? Arrays werden gemäß ihres skalaren Typs ausgerichtet. Ändern Sie dabei nicht die Reihenfolge der Variablen! Verwenden Sie die folgende Tabelle. Tragen Sie die Füllbytes explizit ein.

```
1 struct student {
2   char[11] name;          //char: 1 Byte
3   char[12] vorname;
4   short    versuch;       //short: 2 Byte
5   char     geschlecht;
6   int      matnr;         //int: 4 Byte
7   char     bestanden;
8   float    note;          //float: 4 Byte
9 };
```

4 Punkte

Variable	erstes Byte	letztes Byte
name	0	

Aufgabe 3: IO — 5 Punkte

1. Welchen Vorteil hat *DMA* gegenüber *Programmierter Ein-/Ausgabe*? 1 Punkt

2. Welchen Vorteil hat *Programmierte Ein-/Ausgabe* gegenüber *Unterbrechungsgesteuerter Ein-/Ausgabe* auf einem System mit einem laufenden Prozess? 1 Punkt

3. Welchen Vorteil hat *Unterbrechungsgesteuerte Ein-/Ausgabe* gegenüber *Programmierter Ein-/Ausgabe* auf einem System mit mehreren laufenden Prozessen? 1 Punkt

4. Welchen Vorteil hat *Eingeblendeter Speicher* gegenüber *Isolierter Ein-/Ausgabe* bei der Programmierung in Hochsprache (z.B. C)? 1 Punkt

5. Wie kann der *Burst-Modus* den Transfer sequentieller Daten beschleunigen? 1 Punkt

Aufgabe 4: Assembler — 9 Punkte

Gegeben sei das folgende Intel-x86-Assemblerprogramm:

```
1  .intel_syntax noprefix
2  .data
3  values:
4          .long 9,4,2,1,6,8,11,3,12,10,5,7
5  length:
6          .long 12
7  .text
8  .globl  main
9  main:
10         call  sort
11         mov  eax, 0
12         ret
13 sort:
14         push ebx
15         mov eax,0
16         jmp .LcondOuter
17 .Louter:
18         mov ecx, eax
19         jmp .LcondInner
20 .Linner:
21         mov edx, values[0+eax*4]
22         mov ebx, values[0+ecx*4]
23         cmp edx, ebx
24         jg .Lswap
25         jmp .Lskip
26 .Lswap:
27         mov values[0+eax*4], ebx
28         mov values[0+ecx*4], edx
29 .Lskip:
30         add ecx, 1
31 .LcondInner:
32         cmp ecx, length
33         jl .Linner
34         add eax, 1
35 .LcondOuter:
36         cmp eax, length
37         jl .Louter
38         pop ebx
39         ret
```

Alle Adressen und Daten sind 32 Bit breit. Ein Befehl ist wie folgt aufgebaut (Intel-Syntax):

Befehl	Bedeutung
`mov edx, values[0+eax*4]`	$reg[edx] = mem[values + reg[eax] \cdot 4]$

1. Wie könnte der ursprüngliche C-Code ohne `goto` ausgesehen haben? Achten Sie auf Parameter und Rückgabewerte von Funktionen. Alle Variablen sind vom Typ `int`.

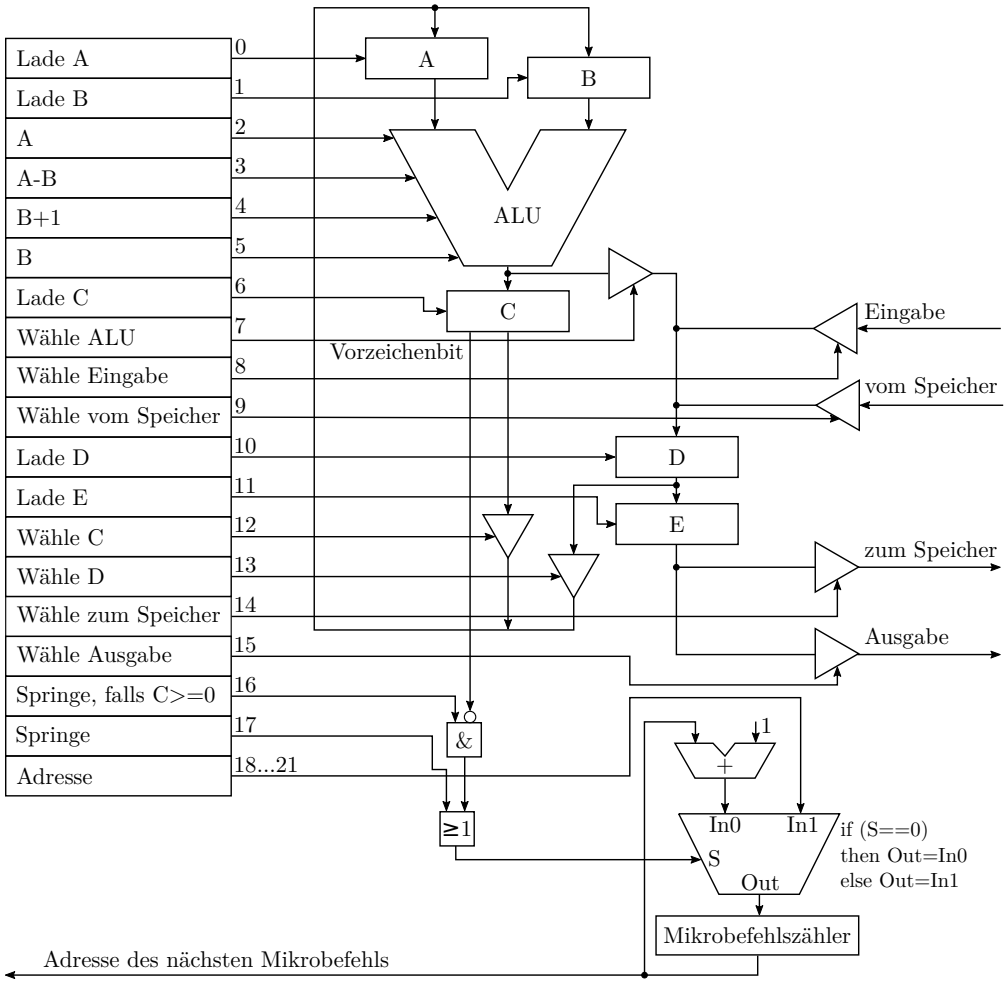
   Hinweis:
   Identifizieren Sie zunächst einzelne Kontrollstrukturen wie Schleifen und Bedingungen und analysieren Sie anschließend deren Körper. 8 Punkte

2. Wieviel Byte auf dem Stack benötigt das Programm zur Laufzeit maximal?

   Die Rücksprungadresse der Funktion `main` befindet sich zu Beginn auf dem Stack und soll mitgezählt werden. (Begündung!) 1 Punkt

Aufgabe 5: Mikroprogrammierung — 12 Punkte

Gegeben sei folgender Teil eines mikroprogrammierten Prozessors:



Die obige CPU soll nun um den Makrobefehl `Reg[A] = Reg[A] / Eingabe` erweitert werden. In `Reg[A]` und `Eingabe` befinden sich natürliche Zahlen echt größer Null. `Reg[A]` soll also am Ende das ganzzahlige Divisionsergebnis beinhalten (Ganzzahldivision, immer Abrunden).

Das Register B hat zu Beginn den Wert `0`. Alle Register dürfen überschrieben werden.

1. Überlegen Sie sich den Ablauf in Pseudocode, der sich an den möglichen Operationen orientiert. 2 Punkte

2. Schreiben Sie ein Mikroprogramm zur Realisierung des beschriebenen Befehls.

   Verwenden Sie die folgende Tabelle und tragen Sie jeweils auch die Bedeutung (z. B. D $\rightarrow$ E) in die Spalte „Erklärung“ ein. Leer gelassene Steuerleitungen entsprechen dem Wert `0`. Bei Sprüngen muss die Zieladresse explizit angegeben werden! Das niederwertigste Bit des Sprungziels entspricht Steuerleitung 21. Vermeiden Sie unnötige Befehle und Sprünge.

   Eine Fehlerbehandlung ist nicht nötig.

   (Die Tabellenlänge entspricht nicht der erwarteten Mikroprogrammlänge!) 10 Punkte

Adr.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	Erklärung
0																							
1																							
2																							
3																							
4																							
5																							
6																							
7																							
8																							
9																							
10																							
11																							
12																							
13																							
14																							
15																							

Aufgabe 6: Parallelverarbeitung — 12 Punkte

Gegeben sei eine Architektur mit einer fünfstufigen Pipeline, bestehend aus *Befehl holen* (BH), *Befehl decodieren* (BD), *Operanden holen* (OH), *Befehl ausführen* (BA) und *Ergebnis sichern* (ES).

1. Wie hoch ist der theoretisch maximale Speedup allgemein und wie wird er für diese Pipeline bestimmt? 2 Punkte

2. Führen Sie folgendes Programm aus.

```
1   mov eax, 0
2   mov ecx, 1
3   jmp Lcond
4 Lbdy:
5   inc eax
6 Lcond:
7   cmp eax, ecx
8   jl Lbdy
9   xor eax, 0xFF
```

   Die Pipeline verfüge über keine erweiterten Mechanismen wie Sprungvorhersage, spekulative Ausführung, Forwarding, etc. Um dennoch ein korrektes Ergebnis zu garantieren, soll statt dessen die Ausführung weiterer Instruktionen so lange verzögert werden, bis kein Konflikt mehr vorliegt, bzw. das Sprungziel definitiv bekannt ist, allerdings auch nicht länger.

   Tragen Sie in der Tabelle auf der nächsten Seite in jeder Stufe *die Instruktion und die Zeile* des dort gerade ausgeführten Befehls ein (z.B. `mov1`). Geben Sie außerdem den aktuellen Registerinhalt von `eax` und `ecx` an, soweit bekannt. Felder mit `nop` können leer gelassen werden.

   Bei Sprungbefehlen soll der Befehlszähler so früh wie möglich aktualisiert werden. *Unbedingte Spünge* werden in `BD` ausgewertet. Der Befehl `cmp` aktualisiert das Statusregister zur Sprungauswertung erst in der Phase *ES*, *bedingte Sprünge* werden in der *OH*-Phase ausgewertet.

   Jeder Befehl muss die gesamte Pipeline durchlaufen. Lassen Sie am Ende die Pipeline leer laufen.

   (Die Tabellenlänge entspricht nicht der erwarteten Ausführungszeit!) 10 Punkte

Takt	BH	BD	OH	BA	ES	`eax`	`ecx`
0	mov1					?	?
1							
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							
18							
19							
20							
21							
22							
23							
24							
25							
26							
27							
28							
29							
30							
31							
32							
33							
34							

Aufgabe 7: Befehlssatzarchitekturklassen — 6 Punkte

Folgendes soll von unterschiedlichen Architekturen berechnet werden:

$$z = x + 5 \cdot y$$

Es stehen vier verschiedene Architekturen mit unterschiedlichem Befehlssatz zur Verfügung.

Operanden können sich in Registern `reg` oder dem Arbeitsspeicher `mem` befinden, oder als unmittelbarer Operand `imm` im Befehl codiert werden. Es darf pro Befehl jedoch nur maximal ein Operand im Arbeitsspeicher adressiert werden.

Die Variablen `x,y,z` können direkt als Speicheroperand verwendet werden. Register stehen als `R0,R1,...` zur Verfügung. `x` und `y` dürfen nicht überschrieben werden.

a)

```
mov       reg|mem, reg|mem|imm
add|mul   reg|mem, reg|mem|imm
push      reg|imm
pop       reg|mem
```

b)

```
ld        mem|imm
st        mem
add|mul   mem|imm
```

c)

```
ld|st     reg, mem
ldi       reg, imm
add|mul   reg, reg|imm, reg|imm
```

d)

```
push      mem|imm
pop       mem
add|mul
```

1. Ordnen Sie den vier Architekturen die in Frage kommende Befehlssatzarchitekturklasse zu. (Keine Begründung) — 2 Punkte

a)

b)

c)

d)

2. Implementieren Sie die obige Berechnung unter Verwendung der jeweils gegebenen Instruktionen. 4 Punkte

a)

b)

c)

d)

Aufgabe 8: Cache — 13 Punkte

1. Was versteht man unter räumlicher und zeitlicher Lokalität typischer Programme? 2 Punkte

2. Welchen Vorteil bieten getrennte Daten- und Instruktionscaches? 1 Punkt

Eine 32-Bit CPU verwende einen 2-fach assoziativen Cache mit insgesamt 8 Blöcken. Jeder Block umfasse 16 Byte. Der Cache sei zu Beginn leer.

Ein Programm liest nacheinander *jeweils 4 Byte* von den folgenden Adressen:
`0xc, 0x20, 0x64, 0x1a8, 0x24`

Der relevante Speicherbereich hat folgenden Inhalt:

```
0x00 00 00 00:  34 2f 6d a4 56 96 9b ec  a5 f0 73 51 a2 4b 39 5a
0x00 00 00 10:  95 14 0f e0 66 48 de 8e  4a b7 cb 38 77 f2 44 f8
0x00 00 00 20:  00 a4 d5 31 38 32 81 f4  b1 c3 de 33 e0 9f fb ff
0x00 00 00 30:  2e 54 60 9d aa 85 ba ad  28 ad 92 d7 b7 10 1f 21
...
0x00 00 00 60:  48 e6 87 60 a5 bf 20 f6  e6 24 cd 4b 40 79 c7 c8
0x00 00 00 70:  ee dd d7 23 2d ab ab e4  6c 8c bd 8f cc 10 1f 7b
...
0x00 00 01 a0:  82 e1 30 15 b7 0c ac 90  e3 44 7b 6b ef e4 00 63
0x00 00 01 b0:  61 9f 2a c4 b6 35 77 a6  aa 92 0c c7 e1 55 a5 eb
0x00 00 01 c0:  22 18 8b 5b d9 6a 0c ad  de f4 db bb b4 43 ee 5d
0x00 00 01 d0:  e2 cb 16 8e 32 ec f3 15  72 9a e7 bd be 0b 9d 42
```

3. Geben Sie die Aufteilung der Adressen in die Cache-relevanten Bestandteile an. 1 Punkt

4. Tragen Sie den Inhalt des Caches nach obigen Zugriffen in die Tabelle ein. Es wird jeweils der Eintrag verdrängt, der am längsten nicht verwendet wurde (LRU).

   Nummerieren Sie die Mengen entsprechend deren Index! Es genügt, jeweils das erste und letzte Byte der Daten im Cache-Block einzutragen. Verdrängte Einträge sollen erkennbar bleiben. 8 Punkte

Menge	Block	Valid	Tag	Daten
	0			
	1			
	2			
	3			
	4			
	5			
	6			
	7			

5. Wie viele Conflict Misses treten bei obigen Zugriffen auf? 1 Punkt

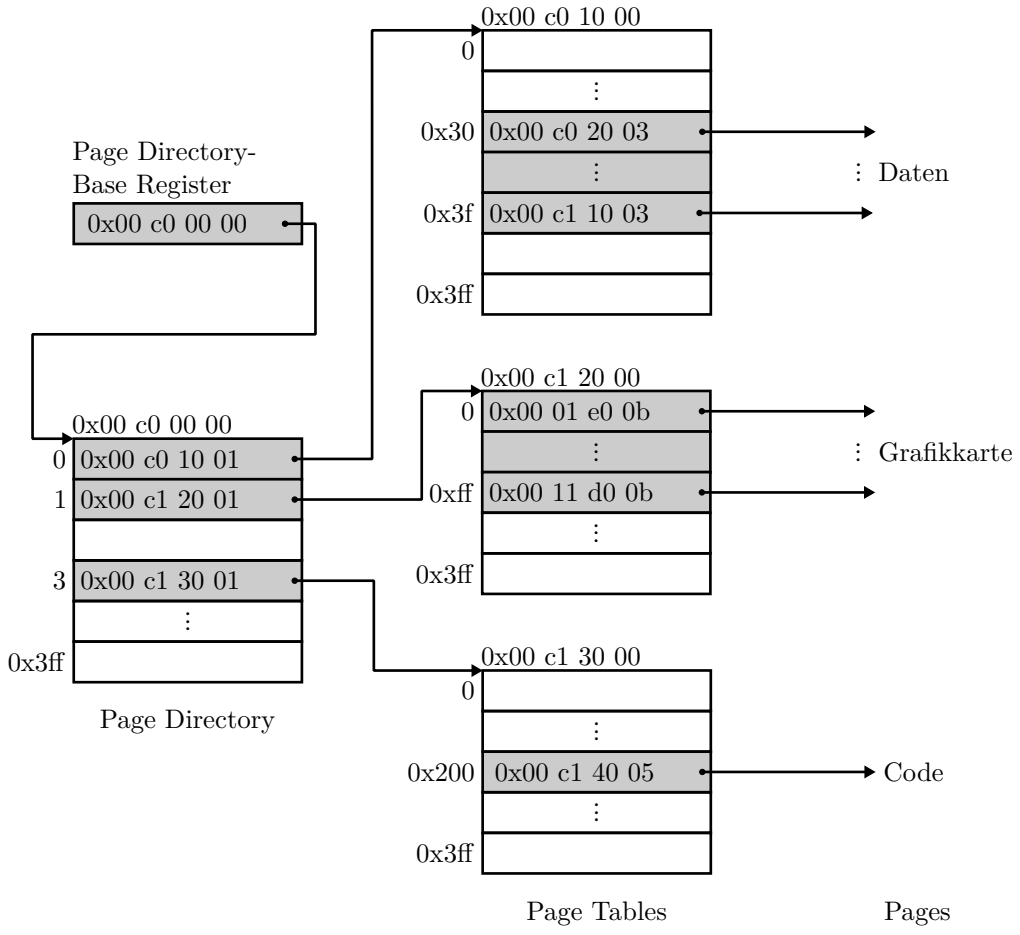
Aufgabe 9: Paging — 18 Punkte

Eine CPU biete eine Memory-Management-Unit mit folgenden Eigenschaften:

- zweistufige Adresstabellen
- je 1024 Einträge zu je 4 Bytes in den Tabellen
  - Bit 31-12: höherwertige Bits der physikalischen Adresse
  - Bit 11-4: unbenutzt
  - Bit 3: Cache-disable-Bit
  - Bit 2: Execute-enable-Bit
  - Bit 1: Write-enable-Bit
  - Bit 0: Present-Bit
- Pages zu je 4 KiB Größe

Arbeitsspeicher ist ab der physikalischen Adresse `0x00 c0 00 00` verfügbar, die Grafikkarte ist ab Adresse `0x00 01 e0 00` eingeblendet.

Der virtuelle Adressraum ist wie folgt aufgebaut. Gültige Einträge sind grau hinterlegt. Bei Bereichen ist jeweils der erste und letzte Eintrag gegeben, dazwischenliegende Einträge haben die gleichen Flags und die zugeordneten Seiten liegen fortlaufend im Speicher.



1. Überlegen Sie sich für die folgenden Codeabschnitte, ob bei, oder im Anschluß an ihre Ausführung eine Exception auftritt. Geben Sie jeweils an, auf welche Tabelleneinträge zugegriffen wird, und nennen Sie ggf. die Ursache der Exception. 8 Punkte

a) `inc 0x01 00 00 00`

b) `call 0x00 03 4c 1c`

c)
```
mov eax, 0x00 e0 0f fc
mov [eax+4], 4
```

d)
```
mov eax, 0x00 e0 0f fc
movl ecx, [eax]
```

2. Welche Gründe gibt es, dass für eine Seite das Present-Bit nicht gesetzt ist? 2 Punkte

3. Gegeben sei eine 64-Bit-Architektur mit 48-Bit breiten virtuellen und physikalischen Adressen. Die Verwaltungsseiten sollen wie üblich die Größe einer Page besitzen und verwenden immer 8 Byte pro Eintrag.

   Wie viele Einträge besitzen die Verwaltungsseiten und wie viele Stufen besitzt die Tabellenhierarchie bei folgenden Seitengrößen? 6 Punkte

a) 4 KiB

b) 256 B

c) 256 KiB

4. Was speichert der *Translation Lookaside Buffer* und welche Aufgabe hat er? 2 Punkte

Zusätzlicher Platz

Zusätzlicher Platz