Department Informatik
Lehrstuhl für Informatik 3 (Rechnerarchitektur)
Friedrich-Alexander-Universität Erlangen-Nürnberg
Martensstraße 3, 91058 Erlangen
27.09.2017
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 18 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!

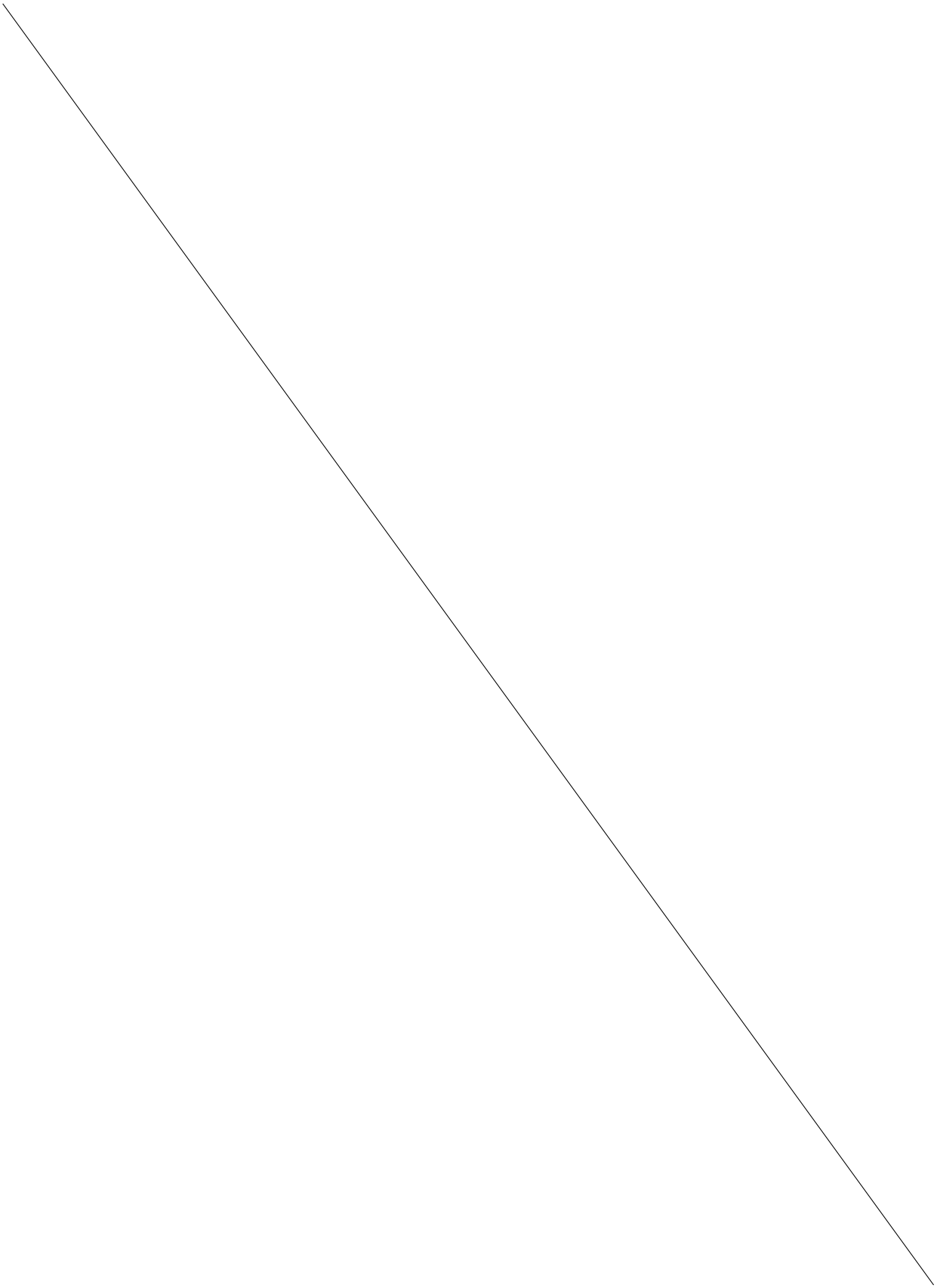
Durch meine Unterschrift bestätige ich

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

Erlangen, den 27.09.2017 .....
(Unterschrift)

Aufgabe	1	2	3	4	5	6	7	8
max. Punktzahl	6	10	12	16	11	13	16	6
erreichte Punktzahl								

Summe	/90		
Bonus	/15		
Gesamt		Note	



Aufgabe 1: Speicherlayout — 6 Punkte

Im Folgenden ist die Struktur *exam* und ein dazugehöriger Speicherauszug einer Instanz der Struktur auf einem MIPS32 Little Endian System gegeben, das sich an die SystemV ABI hält. Die Struktur fängt direkt am Anfang des angegebenen Speicherauszuges an.

```
1 struct exam {
2     unsigned int   id;
3     unsigned char  day;
4     unsigned char  month;
5     unsigned int   year;
6     unsigned char  questions;
7     unsigned int   participants;
8 };
```

Struktur

```
be ef ca fe 0d 02 1b 09
e1 07 00 00 07 06 05 04
64 00 00 00 bc ff 01 00
         ...
```

Speicherauszug

1. Welche Werte (in hexadezimaler Darstellung) haben die Elemente der Strukturinstanz, die durch den Speicherauszug dargestellt wird? Befüllen Sie dazu die folgende Tabelle. 3 Punkte

Feldname	**Wert**
id	
day	
month	
year	
questions	
participants	

2. Wie viele Byte Padding wurden für obige Struktur verwendet? 1 Punkt

3. Was muss allgemein gelten, damit ein Datenwort richtig ausgerichtet ist? 1 Punkt

4. Wie kann man den Speicherverbrauch der Struktur reduzieren, ohne compilerspezifische Mechanismen zu verwenden? 1 Punkt

Aufgabe 2: Arbeitsspeicher — 10 Punkte

Gegeben sei ein Arbeitsspeichermodul mit 1 GiB Größe, welches aus einem Chip mit zwei Bänken zu je 512 MiB besteht. Jede Bank besteht aus 2^{16} Zeilen, die je 2^{10} Spalten beinhalten. Eine Zelle umfasse 64 Bit. Die Busbreite betrage auch 64 Bit. Der maximale Speichertakt darf 400 MHz nicht überschreiten.

1. Wie nennt sich diese Art der Speicherorganisation unter Verwendung von Bänken? 1 Punkt

2. Es soll nun ein Burstzugriff ab Adresse `0x9000` stattfinden, wobei 16 Byte gelesen werden.

   Vervollständigen Sie die Kontrollleitungen *Befehl*, *Adresse*, *Bank* und *Daten* der folgenden Tabelle. Die bereits vorgegebenen Einträge betreffen die Bank 0, sodass nur noch die Werte für Bank 1 eingetragen werden müssen.

   Tabellenzellen mit irrelevantem Inhalt (*Don't Cares*) können leer gelassen werden. Wird in einem Takt allerdings kein Befehl gesendet, muss dies durch ein Minus (-) markiert werden.

   Jede Bank hat zwischen jedem Befehl 3 Takte Latenzzeit. Zwischen einem *Read*-Befehl und der Ausgabe der Daten vergehen 5 Takte.

   Es gibt keine Abhängigkeiten zwischen den Bänken. Gehen Sie davon aus, dass sich beide zu Beginn des Taktes 0 im Initialzustand befinden. 5 Punkte

Takt	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Befehl	Prech.			Active			Read								
Adresse				2			256								
Bank	0			0			0								
Daten												Byte 0–7			

3. Geben Sie die maximale Datenrate in $\frac{\text{MB}}{\text{s}}$ an, die zwischen Speichercontroller und Speichermodul unter Einhaltung der Maximalfrequenz bei Verwendung des DDR4-Standards (8-fach Prefetch) erreicht werden kann. (Mit Rechnung!) 2 Punkte

4. Wie unterscheidet sich die Datenübertragung von SDRAM gegenüber DDR-RAM? 1 Punkt

5. Warum ist DDR-RAM trotz gleichem Speichertaktes unter Umständen schneller als normaler RAM? 1 Punkt

Aufgabe 3: Cache — 12 Punkte

Für die folgende Aufgaben soll ein 2-fach-assoziativer Cache der Größe 256 Byte betrachtet werden. Die Adressen sind 32 Bit groß. In eine Cachezeile passen 32 Byte.

1. Teilen Sie die Adressbit in die für den Cache relevanten Teile (*Tag*, *Index* und *Offset*) auf.

   1 Punkt

2. Es soll nun nacheinander auf unterschiedliche Daten zugegriffen werden. Füllen Sie die zugehörige Tabelle mit den relevanten Daten. Innerhalb einer Menge wird als Ersetzungsstrategie LRU verwendet. Klammern Sie verdrängte Daten ein. Geben Sie zu jeder Cachezeile an, zu welcher Menge sie gehört.

   Es reicht jeweils das erste und letzte Byte der geladenen Daten anzugeben.

   Inhalt des Arbeitsspeichers:

```
  Offset                          Daten
0x00000000    d9 0d e7 5b 47 4e 72 19  35 a7 bd 20 47 79 47 95
0x00000010    e2 a0 27 f1 7f 0a 33 ed  32 18 28 79 6b 8f 02 23
0x00000020    55 e2 8d 79 f0 14 3b c3  b5 3d f7 75 ec fc 10 05
0x00000030    be 97 7d c1 2f e5 ff e6  0e 2e 5d d6 17 e9 56 88
0x00000040    40 78 6c f4 8b 8c 15 49  5d 38 bc aa 84 47 61 b7
...
0x00000080    b7 79 fd 56 e5 d3 dd a1  cf 44 06 ef 2e 9b 5a 01
0x00000090    33 00 5e 36 0a 01 f1 ce  89 85 26 79 b6 b6 bb b2
...
0x00000180    90 5f b5 9c 73 72 ad 6f  5c 18 76 83 74 12 45 73
0x00000190    65 0e 99 19 2b f8 5f ae  70 cc ab ea 53 10 90 da
```

   Zugriffe:

   1. `0x020`
   2. `0x008`
   3. `0x034`
   4. `0x01c`
   5. `0x08c`
   6. `0x192`

   6 Punkte

Menge	Tag	Daten

3. Wie hätte man die obige Verdrängung durch Änderung der Cacheorganisation und ohne Vergrößerung des Caches vermeiden können? 1 Punkt

4. Erklären Sie die Begriffe *zeitliche* und *räumliche Lokalität*. 2 Punkte

5. Erklären Sie die Begriffe *virtueller Cache* und *physikalischer Cache*. Welche Vorteile haben diese Arten jeweils? 2 Punkte

Aufgabe 4: Paging — 16 Punkte

Eine 32 Bit CPU habe eine zweistufige MMU. Jede Seite hat eine Größe von 4 KiB. Es werden zudem *Huge Pages* mit einer Größe von 4 MiB unterstützt. Die Seitentabellen umfassen 1024 Einträge zu je 4 Byte. Die Bit der Tabellenzeilen haben die folgende Bedeutung:

Erste Stufe:

31 – 12	11 – 5	4	3	2	1	0
Adresse der nächsten Stufe	unbenutzt	Huge Page Bit	Read-Enable	Write-Enable	Execute-Enable	Present

Die Bit 3, 2 und 1 der ersten Stufe werden nur verwendet, wenn das *Huge Page Bit* gesetzt ist und sind ansonsten unbenutzt. *Huge Pages* müssen nur auf 4 KiB ausgerichtet werden.

Zweite Stufe:

31 – 12	11 – 4	3	2	1	0
Adresse der nächsten Stufe	unbenutzt	Read-Enable	Write-Enable	Execute-Enable	Present

Unbenutzte Bit sind zu nullen.

1. Es soll nun eine Tabellenhierarchie für einen neuen Prozess angelegt werden. Der virtuelle Adressraum sieht wie folgt aus:

Textsegment	`0x00400000` – `0x00401fff`
Datensegment	`0x00644000` – `0x00644fff`
Heapsegment	`0x70000000` – `0x707fffff`
Stacksegment	`0x7fb00000` – `0x7fffffff`

   Erstellen Sie die Tabellenhierarchie, indem Sie die vorgegebenen Tabellen füllen. Nur Einträge, deren Present-Bit nicht 0 ist, müssen eingetragen werden. Die erste Spalte dient jeweils dazu, die Zeilennummer anzugeben. Die zweite Spalte soll den Eintrag in hexadezimaler Darstellung beinhalten.

   Sowohl für die Tabellen als auch für die Seiten des virtuellen Adressraums können Sie physikalischen Speicher ab Adresse `0x1000000` verwenden. Vergeben Sie ihn aufsteigend und lassen Sie dabei keine Lücken entstehen. Schreiben Sie außerdem zu jeder Tabelle dazu, an welcher Stelle im Speicher sie liegt.

   Verwenden Sie, falls möglich, *Huge Pages* und denken Sie daran, die Rechte angemessen zu setzen. Setzen Sie außerdem das *Page Directory Basis Register* (PDBR) auf die richtige Adresse für den neu erstellten Prozess.

   Hinweis: Die Anzahl der vorgegebenen Tabellen muss nicht die erforderliche sein! 10 Punkte

PDBR:

Adresse:	

Adresse:	

Adresse:	

2. Die gegebene MMU hat einen vollassoziativen TLB mit 2 Einträgen und der LRU-Ersetzungsstrategie. Es wird nacheinander auf die virtuellen Adressen `0x400000`, `0x7ffffffc`, `0x7ffffff8`, `0x400004`, `0x6443c0`, `0x400008`, `0x701e088c` und `0x40000c` zugegriffen.

   Spielen Sie die Zugriffe durch und tragen Sie jeweils den Tag und die Daten in die zugehörigen Felder der folgenden den TLB beschreibenden Tabelle ein. Klammern Sie verdrängte Metainformationen und Daten ein. Die Flags können weggelassen werden.

   Der TLB verhält sich für *Huge Pages*, als wären diese viele Seiten mit 4 KiB Größe. 6 Punkte

Tag	**Daten**

Aufgabe 5: Assembler — 11 Punkte

Es sei folgende MIPS32-Assemblerfunktion `p` gegeben, die ein `int` als Parameter erwartet und ein `int` zurückgibt.

```
1  p:
2      addiu $t1, $0, 2
3      sltu  $t9, $a0, $t1
4      bnez  $t9, fail
5      srl   $t0, $a0, 1    # Shift Right Logical ($t0 = $a0 >> 1)
6      addiu $t0, $t0, 1
7      j outer_cond
8
9  outer:
10     move $t2, $a0
11     j inner_cond
12
13 inner:
14     subu $t2, $t2, $t1
15
16 inner_cond:
17     sltu $t9, $t1, $t2
18     bne  $t9, $0, inner
19     beq  $t1, $t2, inner
20
21     bne  $t2, $0, skip
22     move $v0, $0
23     jr   $ra
24 skip:
25     addiu $t1, $t1, 1
26 outer_cond:
27     sltu $t9, $t1, $t0
28     bne  $t9, $0, outer
29
30     addiu $v0, $0, 1
31     jr $ra
32 fail:
33     move $v0, $0
34     jr $ra
```

1. Wie könnte die Funktion `p` in C-Code ausgesehen haben? Formen Sie sie um und schreiben Sie eine äquivalente C-Funktion, die keine `goto`s enthält.

   Alle Variablen sind vom Typ `int` — 8 Punkte

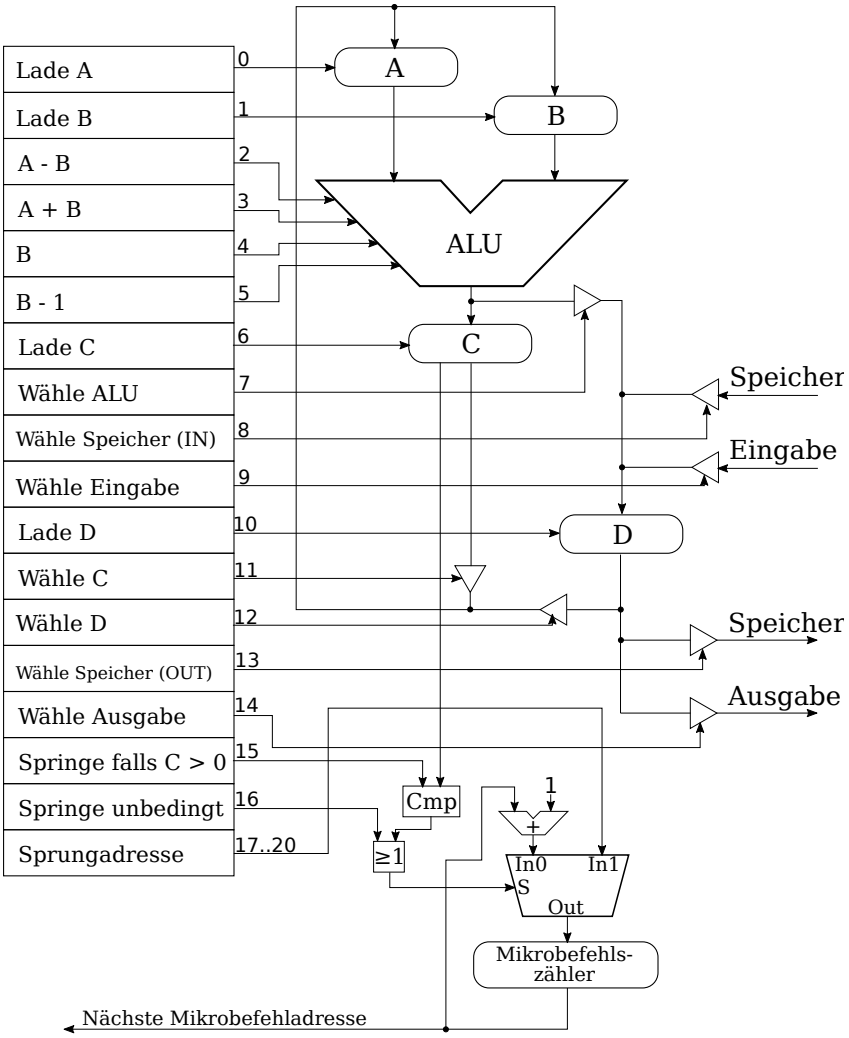
(Platz für C-Code)

2. Nennen Sie zwei Möglichkeiten, Funktionen Parameter zu übergeben. 1 Punkt

3. Was sind `caller-save`- und `callee-save`-Register? 2 Punkte

Aufgabe 6: Mikroprogrammierung — 13 Punkte

Es sei folgendes Mikroprogrammwerk gegeben.



1. Implementieren Sie ein Mikroprogramm, das die Operation „Ausgabe = Speicher · Eingabe“ implementiert, wobei Sie davon ausgehen können, dass vom Speicher und von der Eingabe gelesene Werte echt größer null sind. Zu Beginn ist der Wert der Register `B` und `C` null.

   Nutzen Sie die folgende Tabelle, um das Programm zu implementieren. Leer gelassene Felder entsprechen dem Wert null. Bei Sprüngen muss die Adresse explizit angegeben werden. Die 20. Steuerleitung ist das niederwertigste Bit. Vermeiden Sie unnötige Befehle und Sprünge.

   Tragen Sie zu jedem Befehl auch eine Erklärung dazu ein (wie bekannt aus der Übung, z.B. A-B → C).

   (Die Tabellenlänge muss nicht der Programmlänge entsprechen!) — 10 Punkte

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

2. Was sind die Vor- und Nachteile der vertikalen Mikroprogrammierung gegenüber der horizontalen Mikroprogrammierung? 1 Punkt

3. An welchen Stellen im obigen Mikroprogrammwerk kann man Bit in der Mikroprogramminstruktion mithilfe vertikaler Mikroprogrammierung einsparen, sodass weiterhin die gleichen Instruktionen wie bei der horizontalen Mikroprogrammierung eingesetzt werden können.

   Wie viele Bit hat eine solche vertikale Mikroinstruktion? 2 Punkte

Aufgabe 7: Pipelining — 16 Punkte

Gegeben sei folgende Pipeline mit fünf Stufen. Sie verfüge über keine Mechanismen wie Sprungvorhersage, spekulative Ausführung, Forwarding oder Ähnlichem. Um eine korrekte Ausführung zu garantieren wird stattdessen die jeweilige Pipelinestufe angehalten bis der Konflikt behoben wurde

Befehl Holen	**Operanden Holen**	**Befehl Ausführen**	**Speicherzugriff**	**Ergebnis Schreiben**
Wird immer jeden Takt durchgeführt	Befehlszähler setzen; Warten bei Hazards; Operanden lesen			Das Ergebnis steht erst nach dieser Stufe zur Verfügung

1. Es soll nun das folgende Programm auf der Pipeline ausgeführt werden. Schreiben Sie, wie aus der Übung bekannt, in welcher Stufe sich welcher Befehl zu jedem Takt befindet. Nutzen Sie dafür die Tabelle auf der nächsten Seite. Um eine eindeutige Zuordnung der Instruktionen zu ermöglichen, sollen zudem die Zeilennummern der Instruktionen an die Mnemonics in der Tabelle angehängt werden. Außerdem sollen die Werte der Register `$a0`, `$t0` und `$v0` zu jedem Takt angegeben werden.

   Zellen in denen sich momentan kein Befehl, bzw. ein `nop` befindet, können leer gelassen werden. Alle Instruktionen am Ende des Programmes entsprechen auch `nop`s. Jeder Befehl muss alle Pipelinestufen durchlaufen.

   Der Einstiegspunkt in das Programm liegt beim Label `_start`.

   Die Anzahl der benötigten Takte entspricht nicht der Tabellenlänge! — 10 Punkte

```
1  fak:
2      addiu $t0, $0, 1
3      addiu $v0, $0, 1
4  loop:
5      beq $t0, $a0, end
6      addiu $t0, $t0, 1
7      mul $v0, $t0, $v0
8      j loop
9  end:
10     jr $ra
11 _start:
12     addiu $a0, $0, 2
13     jal fak
```

Takt	BH	OH	BA	MEM	ES	`$a0`	`$t0`	`$v0`
0	addiu12					?	?	?
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								

2. Welche Arten von Hazards sind bei der Ausführung aufgetreten? 1 Punkt

3. Berechnen Sie den Speedup für das auf der Pipeline ausgeführte Beispielprogramm gegenüber einer CPU ohne Pipelining. 1 Punkt

4. Warum erreicht man in der Realität, auch wenn keine Hazards auftreten, nicht den asymptotischen Speedup? 1 Punkt

5. Welche Möglichkeit haben Sie in der Vorlesung und Übung kennen gelernt, Hazards zu vermeiden bzw. deren Auswirkungen zu mildern?

   Beschreiben Sie kurz die Funktionsweise. 2 Punkte

6. Superskalare Architekturen sind eine andere in der Vorlesung vorgestellte Möglichkeit, Parallelität auf Instruktionsebene in Prozessoren einzubauen. Wie funktioniert das Konzept grundlegend? 1 Punkt

Aufgabe 8: Verschiedenes — 6 Punkte

1. Welche Werke existieren im von Neumann'schen Universalrechenautomaten (URA)?
2 Punkte

2. Nennen Sie zwei Alternativen zum URA-Prinzip. 1 Punkt

3. Welche der in der Vorlesung vorgestellten Arten zur Ein- und Ausgabe von Daten zu Peripheriegeräten weist im Ein-Prozess-Betrieb die geringste Latenz auf? 1 Punkt

4. Erklären Sie die Arten der Fragmentierung, die in der Vorlesung besprochen wurden.
2 Punkte

Zusätzlicher Platz