6. *(____ / 11 Punkte)* Gegeben ist folgendes Programm in Pseudocode-Notation. Die Ausführung startet bei `Program Start()` und endet bei `exit`. Die Variable `sum` ist global und die Variable `n` sind lokal deklariert. Die Funktion `sum_rec` hat einen Übergabeparameter. Für die Verwaltung von Rücksprungadressen und Übergabeparameter wird ein Stack verwendet. Zuerst werden Rücksprungadressen und erst dann Übergabeparameter auf den Stack gelegt.

Tragen Sie in die Tabelle ein, an welchen Stellen der Stack mit welchen Operationen (push/pop) verändert wird. Tragen Sie außerdem die aktuellen Werte der globalen Variablen und den Inhalt des Stacks nach Durchführung der jeweiligen Instruktion ein. Die Adressen (`0:` bis `8:`) der Instruktionen sind jeweils links neben dem Pseudocode angegeben.

Hinweis: Übergabeparameter entsprechen lokalen Variablen in den aufgerufenen Funktionen und werden also erst am Ende der Funktion vom Stack gelöscht.

Hinweis: Möglicherweise werden nicht alle Zeilen der Tabelle benötigt.

```
     global sum;
     function sum_rec(n) {
0:     if n > 0 then {
1:       sum = sum+n;
2:       n = n-1;
3:       sum_rec(n);
       }
4:     return;
     }

     Program Start() {
5:     sum = 0;
6:     n = 2;
7:     sum_rec(n);
8:     exit;
     }
```

Adresse	sum	Stack-Operation	Stack-Inhalt
5	0		
6	0		
7	0	push(8) push(2)	8, 2
0	0		8, 2
1	2		8, 2
2	2		8, 2
3	2	push(4) push(1)	8, 2, 4, 1
0	2		— " —
1	3		— " —
2	3		— " —
3	3	push(4) push(0)	8, 2, 4, 1, 4, 0
0	3		— " —
4	3	pop(0) pop(4)	8, 2, 4, 1
4	3	pop(1) pop(4)	8, 2
4	3	pop(2) pop(8)	
8	3		

R2

R0 0,1,2,3

R1 4,5,6,7 R3 2 Byte im Speicher

R3 4

3. *(____ / 10 Punkte)* Schreiben Sie ein Micro16-Programm, das den Inhalt von zwei 8 Byte großen Bereichen im Speicher vergleicht. Es kann davon ausgegangen werden, dass die Startadresse des ersten Speicherbereichs in Register R0 und die Startadresse des zweiten Speicherbereichs in Register R1 liegt. Die zu vergleichenden 8 Byte liegen von diesen Startadressen ausgehend, aufsteigend im Speicher. Falls der Inhalt der beiden Speicherbereiche bitweise ident ist, so soll der Wert +1 im Register R7 abgelegt werden. Ansonsten soll der Wert 0 in R7 abgelegt werden.

Micro-Code:

```
R3 <- 64(dez)
:loop
(R3); if Z goto. gleich
MAR <- R0; rd
rd
R2 <- MBR
MAR <- R1; rd
rd
R3 <- MBR
R5 <- ~R3
R <- R3+1
R4 <- R2+R3
R0 <- R0+1
R1 <- R1+1
R3 <- R3+-1
(R4); if Z goto. loop
:ungleich
R7 <- 0
goto. end
:gleich
R7 <- 1
:end
```

(____ / *8 Punkte)* Die in einem Linux Rechner verbaute Netzwerkkarte eth0 wird durch die nachfolgenden Werte beschrieben:

Zeile	root@meinRechner:~$ ifconfig	
1	eth0	Link encap:Ethernet HWaddr 00:1e:c8:2d:30:45
2		inet addr:131.130.64.12 Bcast:131.130.64.127 Mask:255.255.255.128
3		inet6 addr: fe80::21e:c8ff:fe2d:3045/64 Scope:Link
4		UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
5		RX packets:15653695 errors:0 dropped:0 overruns:0 frame:0
6		TX packets:8781135 errors:0 dropped:0 overruns:0 carrier:0
7		collisions:0 txqueuelen:1000

(a) Teilen Sie das in Zeile 2 beschriebene Netzwerk in zwei gleich große Teile und geben Sie die daraus resultierenden Teilnetze in CIDR-Notation an.

126

131.130.64.0 - 131.130.64.63

131.130.64.64 - 131.130.64.127

(b) Sie erhalten vier Ethernetpakete mit je 130 Byte. Diese Pakete wurden via SSH über TCP, IPv4 und Ethernet übertragen. Wie viele Byte Nutzdaten wurden versendet, wenn jeder Header 10 Byte groß war und TCP die Paketgröße beschränkte?

1 Paket: 130 Byte - 10 - 10 - 10 = 90

100 Byte 1 Paket ohne SSH-Header

300 Byte 3 Pakete ohne SSH-Header

390 Byte Nutzdaten

(c) Kreuzen Sie an, ob es sich um wahre oder falsche Aussagen handelt.
(richtig: +1 Punkt, falsch: -1 Punkt, keine Antwort: 0 Punkte)

wahr	falsch	
⊗	○	Mittels Portforwarding kann man einen Computer in einem privaten Netzwerk erreichen.
○	⊗	Anfragen an die Netzwerkadresse werden an alle IPv4-Adressen im Netzwerk geschickt.
⊗	○	IPv4 ist dem Layer 3 im OSI-7-Schichten-Modell zu zu ordnen.

/ *11 Punkte)* Sie haben ein Programm mit bedingten Sprüngen in Pseudo-Code Notation gegeben. Der Prozessor verwendet Pipelining mit *Branch Prediction*. Die Pipeline besteht aus den vier Stufen *Instruction Fetch (F)*, *Decode (D)*, *Execute (E)* und *Memory & Write Back (M/W)*. Werte, die in M/W geschrieben werden, stehen im selben Takt in D bereits zur Verfügung.

Bedingte Sprünge werden in Stufe D erkannt. Ob gesprungen wird oder nicht, steht fest, wenn die Instruktion die Stufe E verlässt.
Wandert also eine Instruktion mit bedingtem Sprung in die Stufe D, so wird zunächst die nächste Instruktion in F geladen. Danach wird abhängig von der Sprungvorhersage, wieder die nächste Instruktion in die Pipeline geladen (kein Sprung), oder die Instruktion in F wird aus der Pipeline geworfen und es wird die Instruktion an der Sprungadresse in F geladen (Sprung).

Verlässt die Instruktion mit bedingtem Sprung die Stufe E und war die Sprungvorhersage falsch, muss die Pipeline geflushed werden. Das bedeutet, die Stufen F und D werden geleert und F wird im darauf folgenden Takt mit der nächsten korrekten Instruktion geladen.

Die Strategie zur Sprungvorhersage nimmt beim ersten Mal an, dass nicht gesprungen wird. Bei jedem weiteren bedingten Sprung wird angenommen, dass die Sprungentscheidung so wie beim letzten Mal ausfällt.

Zeichnen Sie den Ablauf der Pipelineverarbeitung für das gegebene Programm. Setzen Sie die Darstellung solange fort, bis alle Instruktionen vollständig abgearbeitet wurden.

Programm P:

i0: n ← *3*
i1: s ← *2*
i2: n ← *n-1*
i3: s ← *s+s*
i4: *if (n≠0) goto i2*
i5: *s ← s-1*
i6: nop

Zeit ↓	F	D	E	M/W
1	***i0***			
2	***i1***	***i0***		
3	***i2***	***i1***	***i0***	
4	***i3***	***i2***	***i1***	***i0***
5	***i4***	***i3***	***i2***	***i1***
6	i5	i4	i3	i2
7	i6	i5	i4	i3
8	i2			i4
9	i3	i2		
10	i4	i3	i2	
11	i5	i4	i3	i2
12	i2		i4	i3
13	i3	i2		i4
14	i4	i3	i2	
15	i5	i4	i3	i2
16	i2		i4	i3
17	i5			i4
18	i6	i5		
19		i6	i5	
20			i6	i5
21				i6

n=2 sprung

n=n-1

n=0

8. *(____ / 15 Punkte)* Ein 1 MiB großer n-Way Set-Associative L3-Cache besitzt 1024 Cache-Sets. Die Blockgröße beträgt 512 Bit und beinhaltet 16 adressierbare Datenwörter. In diesem System können Datenwörter im vollen Umfang zur Adressierung verwendet werden.

(a) Berechnen Sie die Adresslänge und Datenwortlänge in Bit sowie die Anzahl der Ways des Caches.

Adresslänge: 32 Bit

Datenwortlänge: $2^5 = 32$ Bit

Ways: $2^4 =$ 16 Ways

(b) Berechnen Sie für den gegebenen Cache die Längen von Tag, Offset und Index in Bit.

Tag-Länge: 18 Bit

Index-Länge: 10 Bit

Offset-Länge: 4 Bit

(c) Es wird die zufällige Erstetzungsstrategie (Random) verwendet, wobei nie der zuletzt verwendete Block ersetzt wird. Wie viele Bits sind für die Verwaltung der Ersetzungsstrategie pro Cache-Set minimal notwendig?

4 Bits

(d) Der L1-Cache hat eine *Hit-Rate* von 80% und der L2-Cache eine Hit-Rate von 95%. Wie viele Anfragen in Prozent werden durchschnittlich an den L3-Cache gestellt?

$1-(0{,}8+0{,}2\cdot 0{,}95)=0{,}01=1\%$

(e) Sie bauen den L3-Cache zu einem Fully Associative Cache um. Steigt oder sinkt die Hit-Rate statistisch? Welche technischen Gründe sprechen gegen einen solchen Umbau?

Aufwand zu hoch, Treffbestimmung bei Cachegröße zu langsam, Suche steigt

Statistik steigt

(d) Zeichnen Sie die Belegung der Pipeline für das gegebene Programm unter der Annahme, dass die Pipeline am Beginn und am Ende leer ist. Berücksichtigen Sie *RAW Data-Hazards* durch Stalling. Klammern Sie gestallte Instruktionen ein.

Zeit ↓	IF	ID	EXE	MEM	WB
1	01				
2	02	01			
3	03	(02)	01		
4	03	(02)		01	
5	03	02			01
6	04	03	02		
7	05	(09)	03	02	
8	05	(09)		03	02
9	05	09			03
10	06	(05)	09		
11	06	(05)		09	
12	06	05			09
13	07	06	05		
14		(07)	06	05	
15		(07)		06	05
16		07			06
17			07		
18				07	
19					07
20					
21					
22					

(e) Können die Instruktionen so umgeordnet werden, dass weniger Stalls auftreten als es bei der gegebenen Anordnung der Fall ist? Falls ja, wie?

zwischen 02 und 03 ausdrücklich

9. *(____ / 10 Punkte)* Sie wollen ein bereits existierendes Programm P beschleunigen, indem Sie dieses teilweise parallel ausführen. Das Programm P hat folgende Eigenschaften:

- 10% von P sind nicht parallelisierbar.
- 20% von P können auf maximal zwei Cores (Prozessorkerne) parallel ausgeführt werden.
- Der restliche Teil kann ohne Einschränkungen auf beliebig vielen Cores ausgeführt werden.

(a) Wie viele Cores (Prozessorkerne) sind mindestens notwendig, um einen Speed-Up (Geschwindigkeitszuwachs) von 3 zu erreichen? Die Synchronisierung und Kommunikation zwischen den Cores soll vernachlässigt werden.

$3 = \frac{1}{\frac{1}{10} + \frac{2}{10 \cdot 2} + \frac{7}{10n}} = \frac{1}{\frac{2}{10} + \frac{7}{10n}} \Rightarrow \frac{1}{3} = \frac{2}{10} + \frac{7}{10n}$

$\frac{4}{3} = \frac{7}{n}$

$\frac{10}{3} = 2 + \frac{7}{n}$

$4n = 21$

$n = \frac{21}{4} = 5,\ldots \Rightarrow 6$ Cores

(b) Wie hoch liegt der theoretisch mögliche Speed-Up?

$\frac{1}{\frac{2}{10} + \frac{7}{10n}} \quad n \to \infty \quad \frac{1}{\frac{2}{10}} = 5$

(c) Wie hoch ist die Ausnutzung in Prozent? Runden Sie auf eine ganze Zahl.

$\frac{\frac{1}{6}}{\underbrace{\frac{2}{10} + \frac{7}{60}}_{\frac{19}{60}}} = \frac{1}{6} \cdot \frac{60}{19} = \frac{60}{114} = 53\,\%$

(d) Wie können Sie eine Ausnutzung von 100% erreichen?

1 Kern verwende

Technische Grundlagen der Informatik			**Test 3** **19.01.2018** 100 Minuten **Gruppe A**
Matrikelnr.	Nachname	Vorname	Unterschrift

Deckblatt sofort ausfüllen und unterschreiben!

Bitte deutlich und nur mit **Kugelschreiber** schreiben. Verwenden Sie keine Korrekturhilfsmittel. Streichen Sie Passagen, die nicht gewertet werden sollen, deutlich durch.

Unleserliche Antworten werden nicht gewertet!

Geben Sie bei Rechenaufgaben immer den **Lösungsweg** an!

Es sind keine Hilfsmittel zugelassen. Dies inkludiert Bücher, Mitschriften, Ausdrucke von Folien, Smartphones, Taschenrechner etc.

Zusatzblätter werden nicht akzeptiert!

Bei **Ankreuzfragen** werden Minuspunkte auf Teilaufgaben übernommen. Das Minimum je Gesamtaufgabe beträgt 0 Punkte.

1	[8]	[ ]
2	[8]	[ ]
3	[10]	[ ]
4	[10]	[ ]
5	[17]	[ ]
6	[11]	[ ]
7	[11]	[ ]
8	[15]	[ ]
9	[10]	[ ]
Summe	[100]	[ ]

---

1. *(____ / 8 Punkte)* Kreuzen Sie an, ob es sich um wahre oder falsche Aussagen handelt. *(richtig: +2 Punkte, falsch: -2 Punkte, keine Antwort: 0 Punkte)*

wahr	falsch	
○	⊗	Der DMA-Controller verbindet die CPU mit der Southbridge.
⊗	○	Die CPU bearbeitet Interrupt-Anfragen.
⊗	○	Bei einem System mit PCH (Platform Controller Hub) ist PCIe direkt mit der CPU verbunden.
○	⊗	Bei einem System mit PCH (Platform Controller Hub) kommuniziert der PCH mittels den FSB (Front Side Bus) mit der CPU.

5. *(____ / 17 Punkte)* Sie arbeiten mit einem Prozessor, der eine fünfstufige Pipeline besitzt: *Instruction Fetch* (IF), *Instruction Decode and Read Registers* (ID), *Execute* (EXE), *Read or Write Memory* (MEM) und *Write Back Registers* (WB). Register werden in der ersten Takthälfte beschrieben und in der zweiten Takthälfte gelesen. Weiters besitzt der Prozessor Hardwareunterstützung für Post-/Pre- In-/Dekrement Operationen bei Speicherzugriffen. Das Ergebnis der Post-/Pre- In-/Dekrement Operation wird dabei in EXE berechnet und gegebenenfalls in WB in ein Register zurück geschrieben. Bedingt durch die Pipelinestruktur kann es zu *RAW Data-Hazards* kommen, welche durch Stalling vermieden werden. Auf dem Prozessor wird folgendes Programm ausgeführt:

```
01:  MOV  R0,  0x9F       # weise R0 die Konstante 0x9F zu
02:  LW   R1,  Mem[R0+]   # lade Wert aus Speicher in R1
03:  LW   R2,  Mem[0xA0]  # lade Wert aus Speicher in R2
04:  ADD  R3,  R1,   R2   # addiere R1 und R2, Ergebnis in R3
05:  SW   R3,  Mem[-R0]   # schreibe R3 in den Speicher
06:  DIV  R4,  R3,   2    # dividiere R3 durch 2, Ergebnis in R4
07:  SW   R4,  Mem[+R0]   # schreibe R4 in den Speicher
```

(a) Auf welche Speicheradressen in hexadezimaler Notation wird in den Zeilen 02, 05 und 07 zugegriffen und welcher Wert steht in R0 nach Ausführung des obigen Programms?

Zeile 02: 0x9F

Zeile 05: 0x9F

Zeile 07: 0xA0

R0= 0xA0

(b) Welche der nachfolgenden Addressierungs-Modi werden in Zeile 01 verwendet?
Register Mode, Immediate Mode, Direct-Addressing Mode, Register-Indirect Mode, Indirect-Addressing Mode

Immediate Mode

(c) Angenommen der Speicher ist wie folgt initialisiert. Wie sieht die Speicherbelegung nach Ausführung des obigen Programms aus? Befüllen Sie die leeren Stellen der Tabelle in hexadezimaler Notation, falls sich der Wert im Speicher geändert hat.

8+3 = E

Adresse	9C	9D	9E	9F	A0	A1	A2	...	100	101	102	103	...	10F
Wert vorher	69	32	FD	3	B	0	41	...	A	AF	23	B1	...	C7
Wert nachher				(E)$_{16}$	(7)$_{16}$			...					...	