Technische Grundlagen der Informatik			**Test 2** **12.12.2014** 100 Minuten **Gruppe A**
Matrikelnr.	Nachname	Vorname	Unterschrift

Deckblatt sofort ausfüllen und unterschreiben!

Bitte deutlich und nur mit Kugelschreiber schreiben. Unleserliche Antworten werden nicht gewertet!

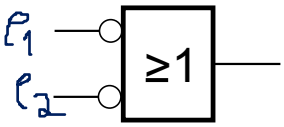
Buch, Mitschriften, Ausdrucke von Folien, Handys, Taschenrechner etc. sind nicht zugelassen!

Zusatzblätter werden nicht akzeptiert!

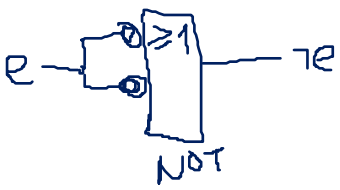
1	[8]	[ ]
2	[12]	[ ]
3	[10]	[ ]
4	[8]	[ ]
5	[8]	[ ]
6	[10]	[ ]
7	[8]	[ ]
8	[10]	[ ]
9	[6]	[ ]
10	[8]	[ ]
11	[12]	[ ]
Summe	[100]	[ ]

---

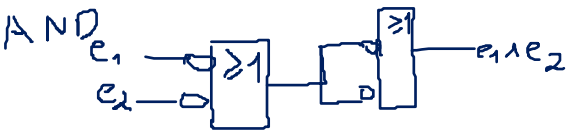
1. *(8 Punkte)* Zeigen Sie, dass das rechts dargestellte Gatter funktional vollständig ist, indem Sie damit die logischen Grundfunktionen NOT, AND und OR **grafisch** realisieren! Nullfunktion (log.0) und Einsfunktion (log.1) stehen Ihnen dabei nicht zur Verfügung!

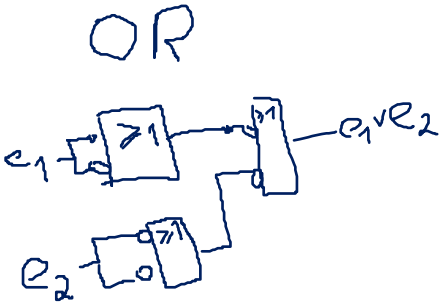


NOT:



e_1	e_2	a
0	0	1
0	1	1
1	0	1
1	1	0



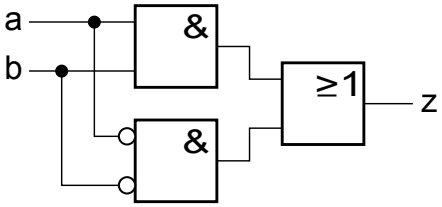


Immer mit dem NOT beginnen

2. *(12 Punkte)* Gegeben sind die Schaltungen (1) und (2) sowie jeweils 3 Schaltnetze. Geben Sie zu jedem Schaltnetz an, ob es sich dabei um eine *gültige* oder *nicht gültige* Realisierung der gegebenen Schaltung handelt. Eine Realisierung ist gültig, wenn das jeweilige Schaltnetz dieselbe Funktion wie die ursprünglich gegebene Schaltung ausführt.

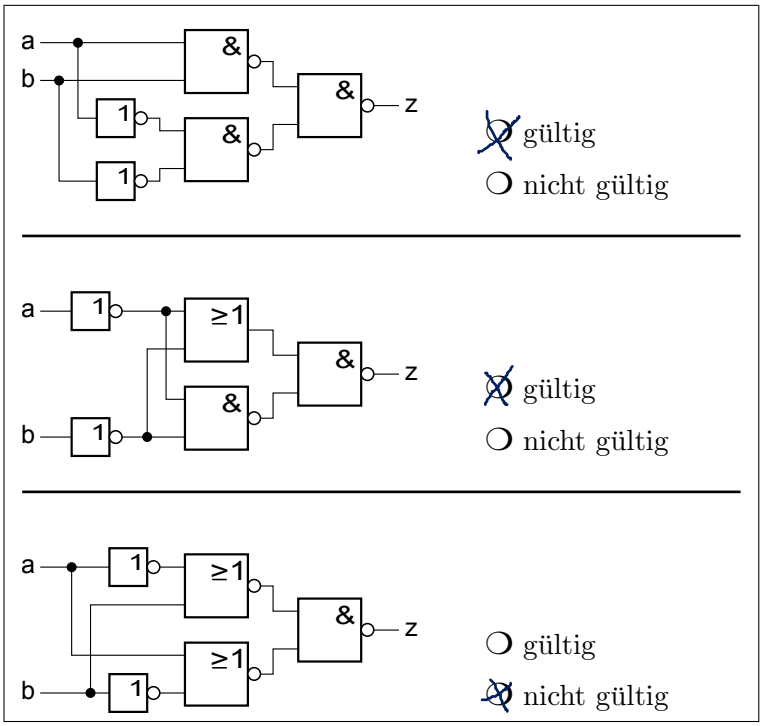
(korrekte Antwort: +2 Punkte; falsche Antwort: -2 Punkte; keine Antwort: 0 Punkte; Minimum: 0 Punkte)

Schaltung (1): Eingänge a und b sowie ein Ausgang z.

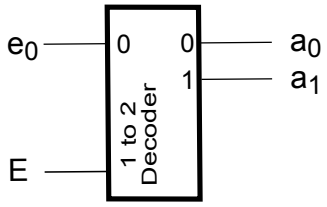


Möglichkeiten zur Lösung:
- Wahrheitstabelle
- In aussagenlogische formel umwandeln, graphische umformung

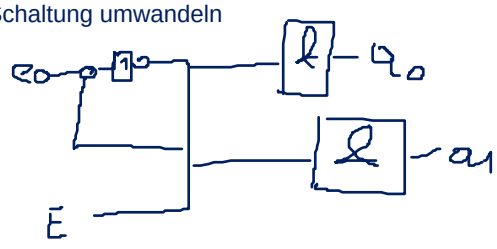
graphisches de morgan

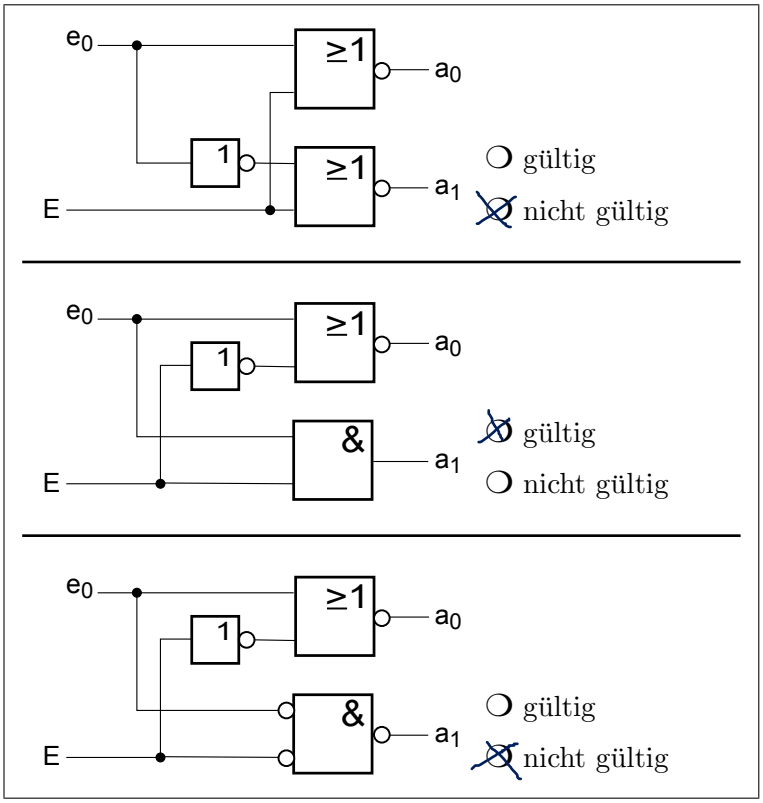


Schaltung (2): 1 zu 2 Decoder mit Eingang e_0, Enable-Eingang E und den Ausgängen a_0 und a_1.



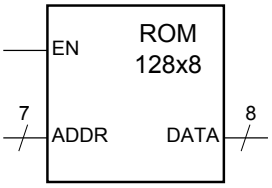
Decoder in Schaltung umwandeln

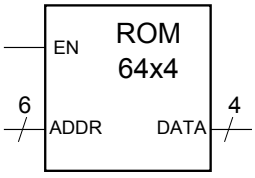


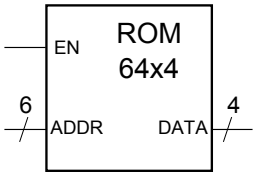


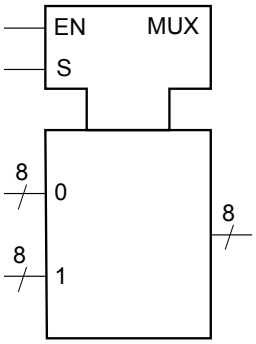
3. *(10 Punkte)* Sie benötigen einen 128×8 ROM, besitzen aber nur die unten vorgedruckten 64×4 ROM-Bausteine sowie den vorgedruckten (2 zu 1)-Multiplexer. Weitere Bauteile (Decoder, NOT-Gatter, etc.) stehen Ihnen nicht zur Verfügung!

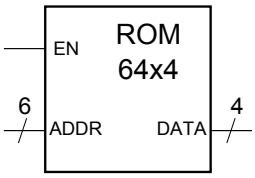
   (a) Vervollständigen Sie untenstehende Schaltung so, dass damit der rechts dargestellte 128×8 ROM-Baustein realisiert wird! Achten Sie auf eine korrekte Beschriftung der Ein- und Ausgänge der Schaltung!

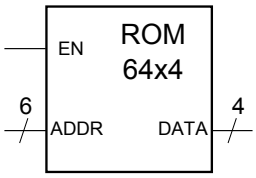












   (b) Wie groß ist die Speicherkapazität des 128×8 ROM-Bausteins in Bytes?

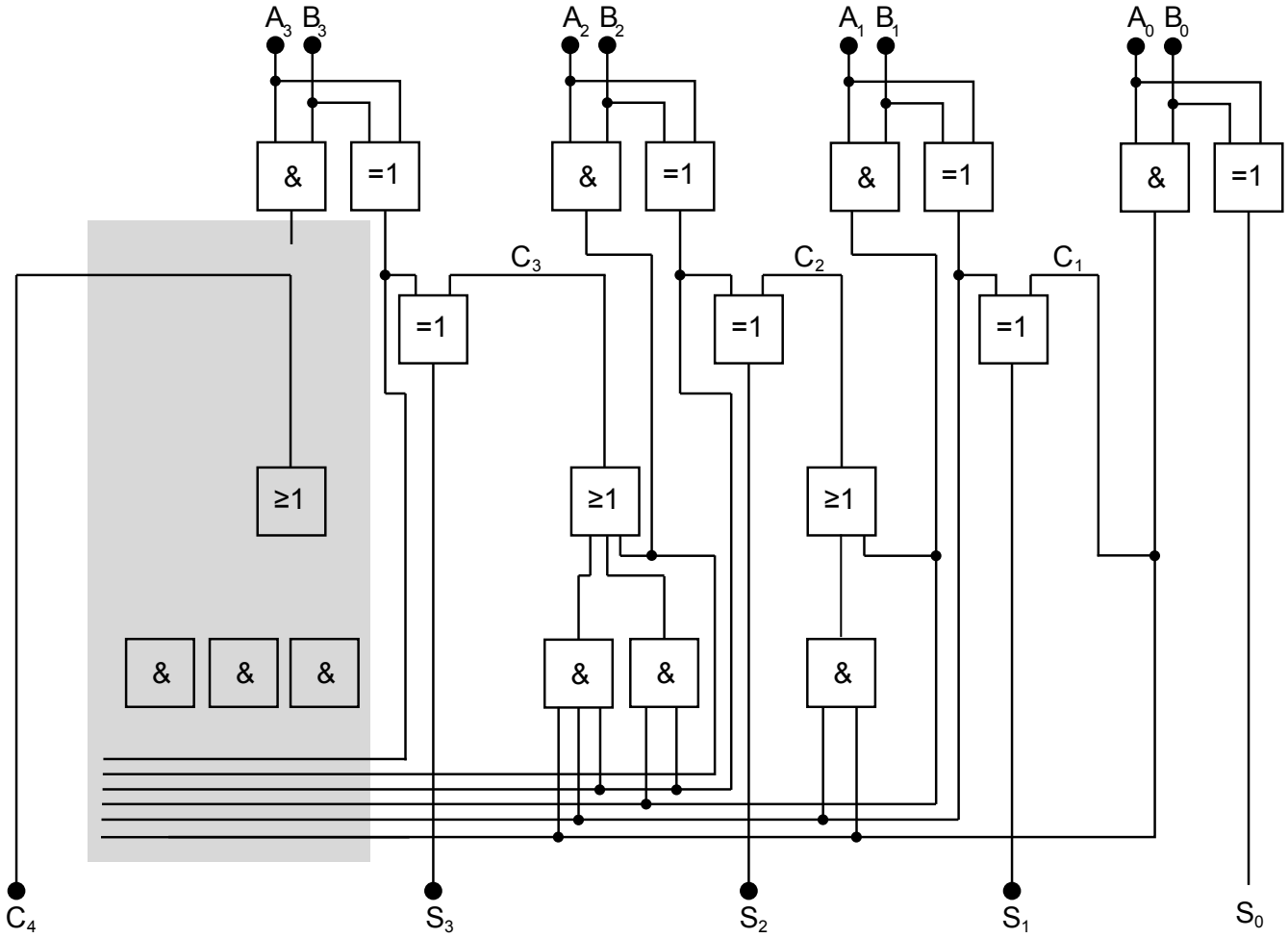
Platz für Notizen:

4. *(8 Punkte)* Überprüfen Sie folgende Aussagen auf Richtigkeit und kreuzen Sie entsprechend an!
(korrekte Antwort: +2 Punkte, falsche Antwort: -2 Punkte, keine Antwort: 0 Punkte; Minimum: 0 Punkte)

Ein Tri-State Ausgang hat 3 logische und 2 physikalische Zustände.	richtig ❍	❍ falsch
Bei der negativen Logik wird HIGH (höherer Spannungspegel) dem logischen Wert *false* zugewiesen.	richtig ❍	❍ falsch
Ein Halbaddierer besitzt einen Carry-Eingang.	richtig ❍	❍ falsch
Bei einem (n zu 1)-Multiplexer (MUX) werden grundsätzlich $ld(n)$ Steuersignale benötigt.	richtig ❍	❍ falsch

---

5. *(8 Punkte)* Vervollständigen Sie nachfolgendes Schaltbild für einen 4-Bit Carry Look Ahead Adder, sodass an Ausgang C_4 der Übertrag der letzten Stelle ausgegeben wird! Ergänzungen sind nur im grau hinterlegten Feld notwendig.



Platz für Notizen:

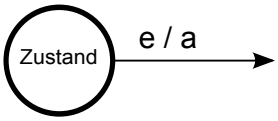
6. *(10 Punkte)* Gegeben ist die Tabelle der Zustandsübergänge inkl. Ausgabe eines Schaltwerks. Bei der Realisierung des Schaltwerks wurden JK-Flip-Flops mit den Beschaltungen J=0, K=1 (Reset) bzw. J=1, K=0 (Set) verwendet.

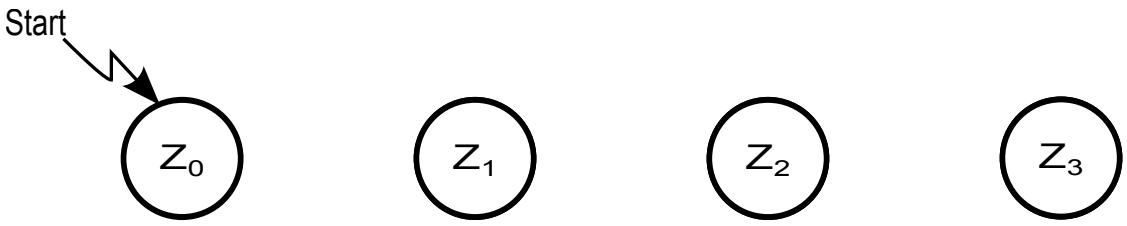
e	0	1	0	1	0	1	0	1
Q_2	0	0	1	1	1	1	0	0
Q_1	1	1	0	0	1	1	0	0
Zustand	Z_0		Z_1		Z_2		Z_3	
a	1	0	0	1	1	0	0	1
J_2	1	0	1	1	0	1	0	0
K_2	0	1	0	0	1	0	1	1
J_1	0	1	0	1	0	1	0	1
K_1	1	0	1	0	1	0	1	0

(a) Geben Sie die verwendete Zustandscodierung an!

Zustand	Q_1	Q_2
Z_0		
Z_1		
Z_2		
Z_3		

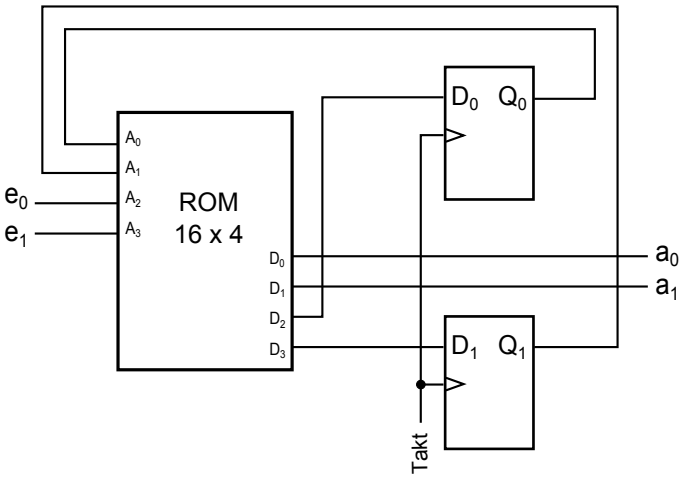
(b) Vervollständigen Sie die grafische Darstellung des Automaten! Verwenden Sie dabei die rechts angeführte Notation.



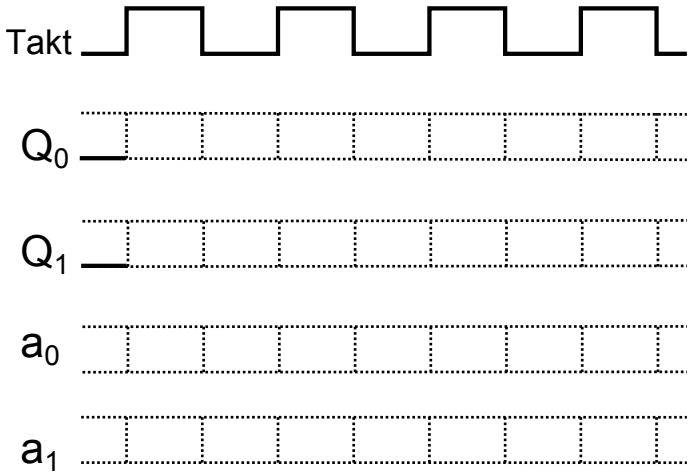


7. *(8 Punkte)* Gegeben ist das rechts dargestellte Schaltwerk mit zwei Eingabewerten e_0 und e_1 und zwei Ausgabewerten a_0 und a_1. Im 16×4 ROM-Baustein, der Teil der Schaltung ist, sind die in der Tabelle links angegebenen Werte gespeichert:

A_3	A_2	A_1	A_0	D_3	D_2	D_1	D_0
0	0	0	0	0	1	0	0
0	0	0	1	1	0	1	1
0	0	1	0	1	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	1	0	1	0
0	1	0	1	1	0	1	1
0	1	1	0	1	1	0	0
0	1	1	1	0	0	0	1
1	0	0	0	0	1	0	1
1	0	0	1	1	1	0	0
1	0	1	0	1	1	0	0
1	0	1	1	0	0	1	0
1	1	0	0	0	1	1	1
1	1	0	1	1	0	1	0
1	1	1	0	0	0	0	0
1	1	1	1	0	0	0	1

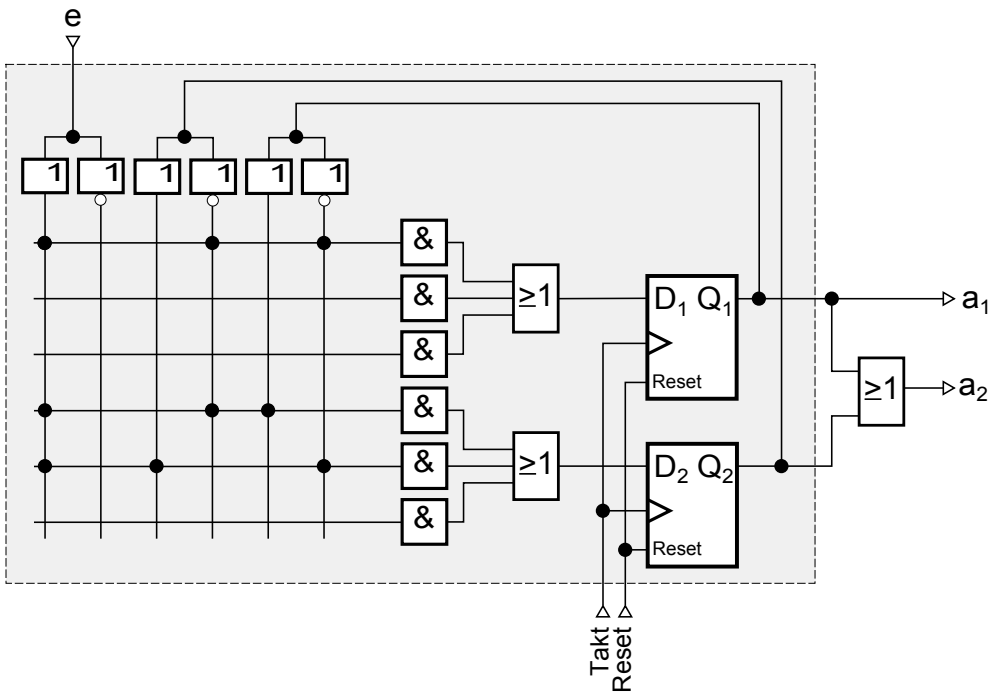


(a) Zeichnen Sie den Verlauf der Signale Q_0, Q_1, a_0 und a_1 für die Eingangswerte $e_0e_1 = 11$!



Platz für Notizen:

8. *(10 Punkte)* Gegeben ist ein Schaltwerk, das mit dem nachfolgenden PLD realisiert wurde:



(a) Lesen Sie die Übergangsfunktion (in Boolescher Form) aus!

(b) Lesen Sie die Ausgangsfunktion (in Boolescher Form) aus!

(c) Um welche Art von Schaltwerk (Moore vs. Mealy) handelt es sich? Begründen Sie!

(d) Welche Bauteile wurden als Zustandsspeicher verwendet?
(korrekte Antwort: +2 Punkte, falsche Antwort: -2 Punkt, keine Antwort: 0 Punkte; Minuspunkte werden ggf. auf die obigen Teilaufgaben angerechnet)

- ❍ JK-Flip-Flops
- ❍ JK-Latches
- ❍ D-Latches
- ❍ D-Flip-Flops

Platz für Notizen:

9. *(6 Punkte)* Analysieren Sie die nachfolgenden Micro16-Instruktionen und kreuzen Sie entsprechend an!
(korrekte Antwort: +1 Punkt, falsche Antwort: -1 Punkt, keine Antwort: 0 Punkte; Minimum: 0 Punkte)

gültig ❍	❍ ungültig	`R7 ← rsh(R5)+R6`
gültig ❍	❍ ungültig	`R5 ← ¬R7∧R8`
gültig ❍	❍ ungültig	`R9 ← (FFF1)`$_{16}$
gültig ❍	❍ ungültig	`MAR ← R3; MBR ← R0+R1; R2 ← R0+R1; wr`
gültig ❍	❍ ungültig	`R0 ← PC+R8+1`
gültig ❍	❍ ungültig	`(MBR); if Z goto 255`

---

10. *(8 Punkte)* Übersetzen Sie die nachfolgenden Instruktionen in binären Micro16-Code!
Bei beiden Instruktionen handelt es sich um gültige Micro16-Instruktionen.

Instruktion A: `R0 ← rsh(R4∧1)`

Instruktion B: `(MBR); if Z goto 36`

Codierung	Beschreibung
ALU = 01	Addition (linker Operand auf A-Bus)
ALU = 10	bitweises UND (linker Operand auf A-Bus)
ALU = 11	bitweise Negation (A-Bus)
SH = 01	shift left
SH = 10	shift right
COND = 01	Sprung auf N=1
COND = 10	Sprung auf Z=1
COND = 11	unbedingter Sprung

Register	Adresse
0	0000
+1	0001
-1	0010
R0	0100
R1	0101
...	...
R10	1110

	AMUX	COND	ALU	SH	MBR	MAR	RDWR	MS	ENS	S-BUS	B-BUS	A-BUS	ADR
A:													
B:													

Platz für Notizen:

11. *(12 Punkte)* Gegeben ist der folgende Micro16-Code:

```
     R5 <- lsh(1+1)
     R5 <- lsh(R5+R5)
     goto .START
:Z0
     R0 <- lsh(R0)
     :START
     (R5); if Z goto .ENDE
     R5 <- R5+(-1)
     R1 <- lsh(R1)
     R2 <- lsh(R2)
     (R0); if N goto .Z0
     R1 <- R1+1
:Z1
     R0 <- lsh(R0)
     (R5); if Z goto .ENDE
     R5 <- R5+(-1)
     R1 <- lsh(R1)
     R1 <- R1+1
     R2 <- lsh(R2)
     (¬R0); if N goto .Z1
     R2 <- R2+1
     goto .Z0
:ENDE
```

(a) Welchen Wert (als Dezimalzahl interpretiert) hat R5 nach der zweiten Programmzeile?

(b) Wofür wird R5 im Programm verwendet?

(c) Angenommen, obiges Programm realisiert einen Automaten mit zwei Zuständen Z_0 und Z_1. Die Eingabe wird von R0 beginnend beim *msb* eingelesen, die Ausgabe erfolgt in die Register R1 und R2 beginnend beim *lsb* (= analog zum Übungsbeispiel). Rekonstruieren Sie diesen Automaten!