Aufgabe 1: Schaltwerksanalyse (33 Punkte)

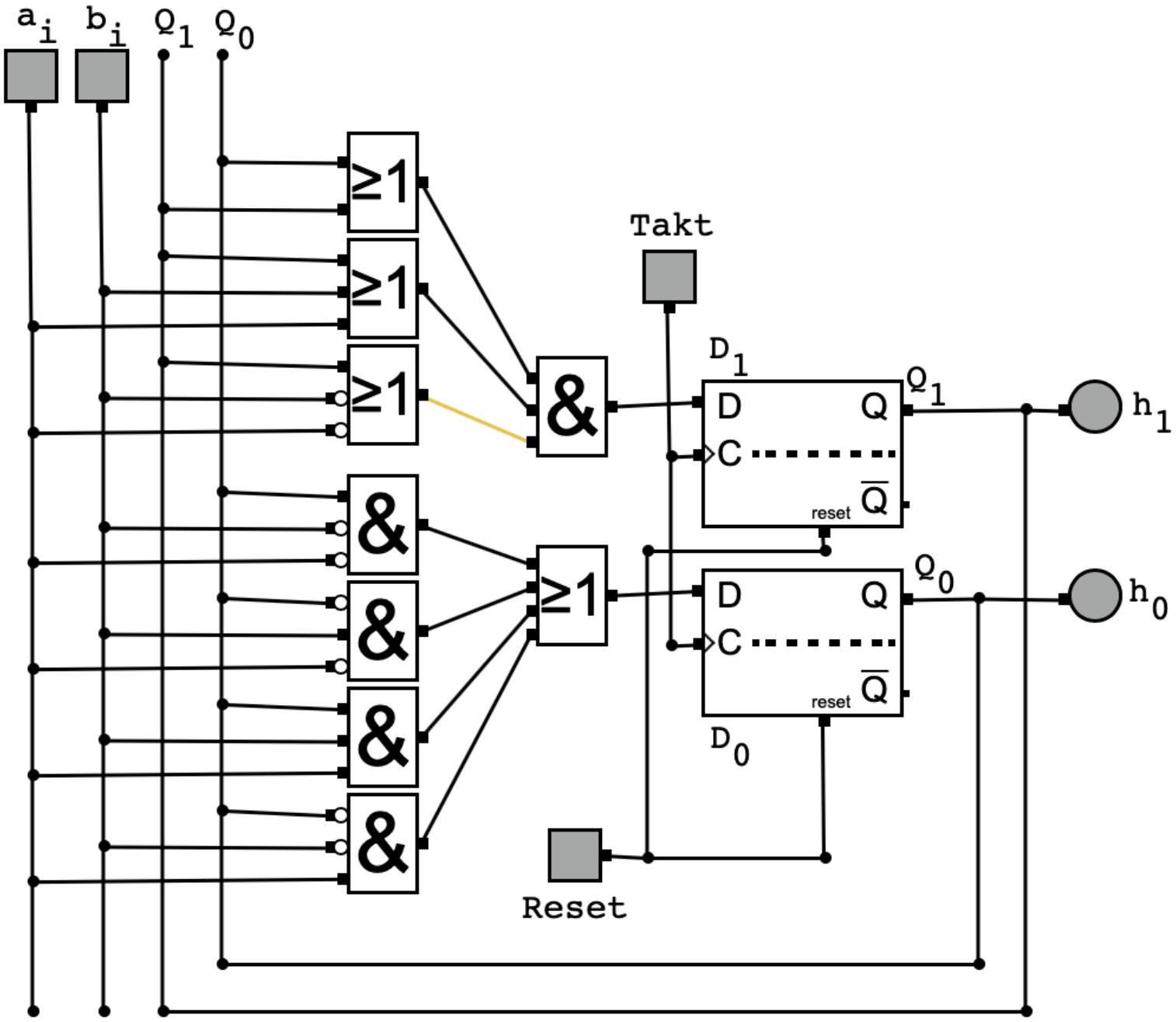


Abbildung 1: Schaltwerk

a) Betrachten Sie für diese Teilaufgabe das in Abbildung 1 dargestellte Schaltwerk, dessen Startzustand der Zustand 0 (Q_1Q_0=00) ist, und beantworten Sie die nachfolgenden Fragen.

Um welchen Automatentyp handelt es sich? Bitte begründen Sie Ihre Antwort kurz.

Wie viele Zustände hat dieser Automat? Bitte begründen Sie Ihre Antwort kurz.

Geben Sie die Ausgabefunktionen h_1 und h_0 des Automaten an.

Geben Sie die Ansteuergleichungen der für die Realisierung des Automaten verwendeten D-Flipflops D_1 und D_0 an.

Bitte geben Sie die zu den Ansteuergleichungen korrespondierenden Zustandsübergangs-funktionen des Automaten an.

Für die Realisierung des Schaltwerkes stehen leider nur D-Flipflops zur Verfügung, die über keinen separaten RESET-Eingang verfügen, sodass die Möglichkeit des Zurücksetzens durch eine zusätzliche Eingabevariable für den Automaten realisiert werden muss.
Welche der nachfolgenden Aussagen ist wahr bzw. falsch? Bitte kreuzen Sie die zutreffende Eigenschaft der jeweiligen Aussage an.

wahr	falsch	Aussage
☐	☐	Die Anzahl der Zustände ändert sich.
☐	☐	Die Anzahl der Eingangsvariablen erhöht sich um eins.
☐	☐	Die Ansteuergleichungen aller Flipflops bleiben gleich.
☐	☐	Die Anzahl der Ausgabebits erhöht sich auf drei.
☐	☐	Bei der Minimierung der Zustandsübergangsfunktionen des Automaten müssen fünf Variablen berücksichtigt werden.
☐	☐	Für die Realisierung werden mehr Flipflops benötigt, als bei der gegebe-nen Variante.
☐	☐	Das Zurücksetzen des Schaltwerks kann weiterhin unabhängig vom Takt (asynchron) erfolgen.

b) Gegeben sei die nachfolgende Zustandsübergangstabelle

Z_1	Z_0	$(Z_1Z_0)^{n+1}$ bei Eingabe ab				y_1	y_0
		00	01	10	11		
0	0	00	01	01	00	0	0
0	1	01	10	10	01	0	1
1	0	10	11	11	10	1	0
1	1	xx	xx	xx	xx	1	1

Geben Sie den Zustandsübergangsgraphen des Automaten an.

Welche Ausgabe-Sequenz von y_1y_0 erzeugt der Automat nach der bitweisen Eingabe der beiden Binärstrings a = 010 und b = 001. Als Startzustand sei $Z_1Z_0 = 00$ zu wählen. Es ist egal, ob Sie mit dem MSB oder LSB beginnen.

Welcher Metrik zwischen zwei in diesem Falle 3-stelligen Binärzahlen entspricht der durch den Automaten bestimmte Wert (letzte Ausgabe)?

Geben Sie die Zustandsübergangsfunktion für Z_1^n **in disjunktiver** und die für Z_0^n **in konjunktiver** Minimalform an! Ihnen stehen für die Bestimmung der Übergangsfunktionen folgende KV-Diagramme zur Verfügung. Bitte füllen Sie diese komplett aus und kennzeichnen Sie Ihre Minimierungen!

Z_1^{n+1} $(a,b)^n$

$(Z_1Z_0)^n$	00	01	11	10
00				
01				
11				
10				

Z_0^{n+1} $(a,b)^n$

$(Z_1Z_0)^n$	00	01	11	10
00				
01				
11				
10				

Z_1^{n+1} $(a,b)^n$

$(Z_1Z_0)^n$	00	01	11	10
00				
01				
11				
10				

Ersatz

Z_0^{n+1} $(a,b)^n$

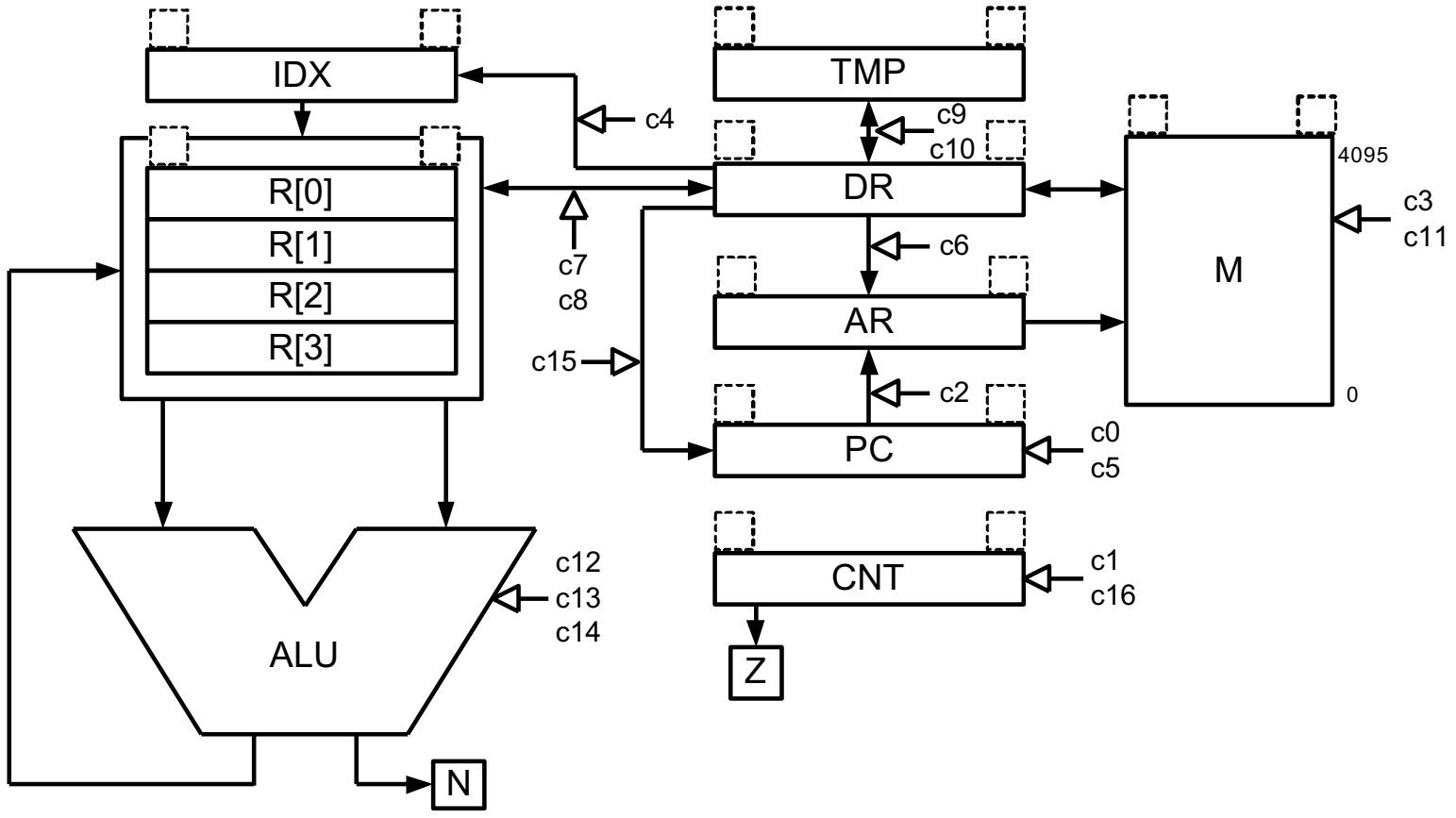
$(Z_1Z_0)^n$	00	01	11	10
00				
01				
11				
10				

Ersatz

Zustandsübergangsfunktionen:

$\Sigma_{A1} =$ / 33 Punkte

Aufgabe 2: CPU (60 Punkte)



Die Breiten der nachfolgenden Steuersignale müssen hier nicht eingetragen werden!

```
c0  PC <- 0              c6   AR <- DR              c12  R[] <- R[] + R[]
c1  CNT <- 0             c7   R[] <- DR(  )         c13  N.R[] <- R[] - R[]
c2  AR <- PC             c8   DR(    ) <- R[]       c14  R[] <- R[] + 1
c3  read M               c9   TMP <- DR             c15  PC <- DR
c4  IDX <- DR(  )        c10  DR <- TMP             c16  Z.CNT <- CNT + 1
c5  PC <- PC + 1         c11  write M
```

Abbildung 2: Operationswerk. Die Breiten der einzelnen Register und Registerteilbereiche sind bei der Angabe der Steuersignale nicht vorgegeben, da diese Teil der Aufgaben sind und von Ihnen in Aufgabe c) im RT-Code spezifiziert werden müssen.

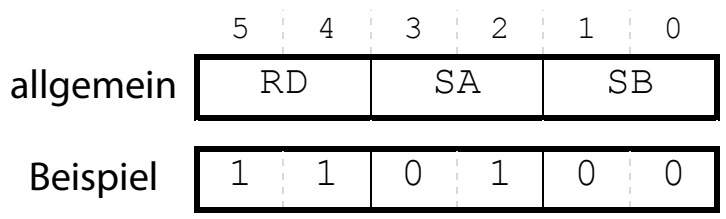


Abbildung 3: Format des IDX Registers. Das Beispiel zeigt die Auswahl von `R[3]` als Zielregister RD, `R[1]` als Quellregister SA und `R[0]` als Quellregister SB.

Gegeben sei das in Abbildung 2 dargestellte Operationswerk (OW) mit dem zwölf Bit breiten Speicher M der sowohl als Befehls- als auch als Datenspeicher genutzt wird. Das OW besitzt eine ALU, die addieren, subtrahieren sowie den am Quelleingang SA anliegenden Wert inkrementieren kann und das Negative-Flag N setzt (N = 1), sollte bei der auszuführenden Operation ein negatives Ergebnis erzeugt werden. Als Quell- und Zielregister der ALU dienen die vier acht Bit breiten Register des Register-Arrays R und deren Auswahl erfolgt über das Index-Register IDX gemäß Abbildung 3. Für den Zugriff auf die Register gilt somit

```
R[IDX(5:4)] <- R[IDX(3:2)] x R[IDX(1:0)],
```

wobei `x` eine Rechenoperation repräsentiert. Weiter verfügt das OW über das zwei Bit breite Zählregister CNT, welches das Z-Flag setzt (Z = 1), falls der aktuelle Zählwert null beträgt.

a) Ergänzen Sie im OW aus Abbildung 2 die fehlenden Breitenangaben der einzelnen Register in den gestrichelten Kästchen.

b) Schreiben Sie einen RT-Code für das gegebene OW, der die ganzzahligen Elemente v_i mit $0 \leq v_i \leq 15, i = 0, \ldots, 7$ eines im Speicher abgelegten Zeilenvektors v aufsummiert. Ihnen stehen dafür lediglich die gegebenen Operationen des OW zur Verfügung.
Der Einfachheit halber darf in dieser Teilaufgabe die Indizierung der Register des Register-Arrays direkt durch Angabe des Indexes statt über das Register IDX erfolgen (Bsp.: `R[1] <- R[0] + R[1]`). Sie können davon ausgehen, dass alle Register mit null initialisiert sind.

Im Speicher (Abbildung 4) seien beginnend ab Adresse 0 der Hexadezimalwert $X10 gefolgt von den Elementen des Zeilenvektors $\overline{v} = (10, 11, 1, 2, 4, 5, 3, 8)$ als Hexadezimalwerte $XAB, $X12, $X45 und $X38 abgelegt. Das X steht hierbei für einen beliebigen einstelligen Hexadezimalwert.

$$\overline{v} = (10, 11, 1, 2, 4, 5, 3, 8)$$

$XAB $X12 $X45 $X38

Adr	11	10	9	8	7	6	5	4	3	2	1	0	
$000	$X				$1				$0				Teiler b
$001	$X				$A				$B				v_0 und v_1: 10 und 11 als $XAB
$002	$X				$1				$2				v_2 und v_3: 1 und 2 als $X12
$003	$X				$4				$5				v_4 und v_5: 4 und 5 als $X45
$004	$X				$3				$8				v_6 und v_7: 2 und 8 als $X38

Abbildung 4: Speicherbelegung $x \in \{0,1\}$

Da jeweils zwei Elemente des Vektors als ein Wert an einer Speicheradresse abgelegt sind, ist es erforderlich, den Wert in diese beiden Elemente auf Basis der nachfolgenden Berechnungsvorschrift der Modulo-Funktion zu zerlegen, bevor diese aufsummiert werden können.

```
# Berechne value mod b = a bei gegebenem b
# Nach der Berechnung gilt zudem value = c * b + a
c = 0
a = value
loop: a = a - b;
      if a < 0 then
        a = a + b
      else
        c = c + 1, goto loop
      end
```

Beispiel:
Für value = $AB und Teiler b = $10 gilt $AB mod $10 = $B = a und für den durch den Algorithmus gelieferten Wert c = $A.

Ergänzen Sie die fehlenden Registerbreiten sowie die Deklaration des Speichers M.

Verwenden Sie folgende Registerbelegung bzgl. des gegebenen Algorithmus:

R[0] = value bzw. a , R[1] = b, R[2] = c, R[3] = Summe

Gehen Sie folgendermaßen vor:

1) Laden Sie den Wert für b, der in diesem Fall als Teiler für die Trennung der beiden Werte fungiert, von der Speicheradresse 0
2) Laden Sie den ersten Wert aus dem Speicher
3) Zerlegen Sie ihn mithilfe der Modulo-Funktion gemäß des gegebenen Pseudocodes
4) Addieren Sie die beiden Vektorelemente auf die Summe
5) Fahren Sie mit dem nächsten Element fort (in einer Schleife ab 2) weiter machen)

```
declare register  DR(    ), AR(    ), IDX(    ), PC(    ), TMP(    )
                  Z, N, CNT(    )
declare memory    M(       )
declare register array R(7:0)[4]
INIT: PC <- 0, CNT <- 0;
```

c) Das gegebenen Operationswerk soll um ein mikroprogrammiertes Steuerwerk zu einer mikroprogrammierten CPU ergänzt werden, die über den nachfolgenden Befehlssatz verfügt.

Opcode	Befehl	Beschreibung
0	LOAD RD, X	Lädt den sich unter Adresse X im Speicher befindenden Wert in das Register R[RD] des Register-Arrays
1	ADD RD, SA, SB	Addiert den Wert R[SB] auf den Wert R[SA] auf und legt das Resultat in R[RD] ab
2	SUB RD, SA, SB	Subtrahiert den Wert R[SB] von dem Wert R[SA] und legt das Resultat in R[RD] ab
3	INC RD, SA	Inkrementiert den Wert des Registers R[SA] und legt das Ergebnis in R[RD] ab
4	JUMP X	Setzt den Programmablauf an Adresse X fort
5	JUMPN X	Setzt den Programmabaluf an Adresse X fort, wenn N = 1 gilt

Das Befehls- und Datenformat seien gemäß Abbildung 5 definiert. Sollte ein Befehl eine Adresse als Parameter (LOAD, JMUP, JMUPN) enthalten, so ist diese stetst direkt nach dem Befehl im Speicher (Adr+1) abzulegen. Alle restlichen Befehle belegen jeweils nur einen Speicherplatz.

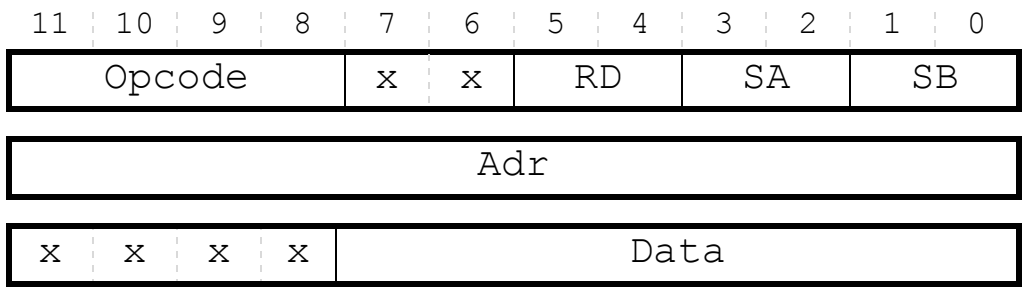


Abbildung 5: Befehls- und Datenformat

Ergänzen Sie im nachfolgenden RT-Programm, die fehlenden Registerbreiten und Indizes sowie die zu den Registertransferoperationen korrespondierenden Kontrollsignale (rechts neben dem Code).

```
declare register  DR(), AR(), IDX(), PC(), TMP()
                  N, Z, CNT()

declare memory    M()
```

Es gilt die Deklaration der Register und des Speichers M aus Aufgabe b). Hier nicht nochmal angeben

---

```
declare register array R(7:0)[4]

INIT: PC <- 0, CNT <- 0;                                              __________

FETCH: AR <- PC;                                                      __________
       read M;                                                        __________
       IDX <- DR(      ), PC <- PC + 1 | switch DR(       ) {        __________
         case 0: goto LOAD
         case 1: goto ADD
         case 2: goto SUB
         case 3: goto INC
         case 4: goto JUMP
         case 5: goto JMPN
         default: goto FETCH};

LOAD:  AR <- PC, PC <- PC + 1;                                        __________
       read M;                                                        __________
       AR <- DR;                                                      __________
       read M;                                                        __________
       R[IDX(     )] <- DR(     ) | goto FETCH;                       __________

ADD:   R[IDX(     )] <- R[IDX(     )] + R[IDX(     )] | goto FETCH;  _______

SUB:   N.R[IDX(     )] <- R[IDX(     )] - R[IDX(     )] | goto FETCH; _____

INC:   R[IDX(     )] <- R[IDX(     )] + 1 | goto FETCH;                __________

JUMP:  PC <- DR | goto FETCH;                                         __________

JMPN:  if N = 1 then
          PC <- DR, goto FETCH                                        __________
         else goto FETCH fi;
```

d) Implementieren Sie den Befehl `ADDS RD, SA, SB, X`, der auf den Wert von R[SA] den sich unter der Adresse X im Speicher befindenden Wert aufaddiert und das Resultat unter R[RD] ablegt. Hierbei sei zu beachten, dass RD und SA stets verschiedene Register sein sollen und nur RD von diesem Befehl verändert werden darf. Resultiert hieraus eine Bedingung für den Aufruf des Befehls? Falls ja, geben Sie dies an. Bitte geben Sie die Kontrollsignale zu den einzelnen RT-Operationen an.

e) Erstellen Sie ein horizontales Mikroprogramm, welches das Verhalten des in der Tabelle gegebenen RT-Codes sowie des JUMPN Befehls realisiert. Füllen Sie hierfür nachfolgende Tabelle aus. Leere Felder werden hierbei gegebenenfalls als 0 interpretiert. Sie müssen also nur die 1en eintragen. Ergänzen Sie zudem den für das Mikroprogramm eventuell abgeänderten RT-Code des JUMPN Befehls. Es stehen Ihnen lediglich die nachfolgenden Condition Select Signale zur Verfügung:

Condition Select	Funktion
000	Nicht springen
001	Springe zu der dekodierten Opcode-Adresse (Mapping-ROM)
010	Springe, falls Z = 1
011	Springe falls N = 1
111	Springe unbedingt

f) Erstellen Sie nun ein vertikales Mikroprogramm, indem Sie den entsprechenden Bereich in der Tabelle ergänzen. Eine gesonderte Angabe der Kodierung in einer separaten Tabelle ist nicht erforderlich.

Adresse					cs			Sprungadresse					Horizontale Kodierung Kontrollsignale c[16:0]																Vertikale Kodierung Kontrollsignale c[3:0]				Entsprechender RT-Befehl
4	3	2	1	0	2	1	0	4	3	2	1	0	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	3	2	1	0	
0	0	0	0	0																														INIT: PC <- 0, CNT <- 0;
0	0	0	0	1																														FETCH: AR <- PC;
0	0	0	1	0																														read M;
0	0	0	1	1																														IDX <- DR( ), PC <- PC + 1 | switch DR( )
0	0	1	0	0																														LOAD: AR <- PC, PC <- PC + 1;
0	0	1	0	1																														read M;
0	0	1	1	0																														AR <- DR;
0	0	1	1	1																														read M;
0	1	0	0	0																														R[IDX( )] <- DR( )| goto FETCH;
0	1	0	0	1																														ADD: R[IDX( )] <- R[IDX( )] + R[IDX( )] | goto FETCH;
0	1	0	1	0																														SUB: N.R[IDX( )] <- R[IDX( )] - R[IDX( )] | goto FETCH;
0	1	0	1	1																														INC: R[IDX( )] <- R[IDX( )] + 1 | goto FETCH;
0	1	1	0	0																														JUMP: PC <- DR | goto FETCH;
0	1	1	0	1																														JMPN:
0	1	1	1	0																														
0	1	1	1	1																														
1	0	0	0	0																														
1	0	0	0	1																														
1	0	0	1	0																														
1	0	0	1	1																														
1	0	1	0	0	Die nachfolgenden Zeilen sind für Aufgabe i) reserviert																													
1	0	1	0	1																														
1	0	1	1	0																														
1	0	1	1	1																														
1	1	0	0	0																														
1	1	0	0	1																														

g) Geben Sie das Mapping-ROM an.

h) Geben Sie den Speicherbedarf des horizontalen und des reinen vertikalen Mikroprogramms (ohne evtl. Dekodierer usw.) an.

i) Der Automat mit dem Startzustand S0 aus Abbildung 6 beschreibt den Ablauf eines Befehls auf dem Operationswerk inklusive Befehlshol- und Befehlsausführungsphase. Erweitern Sie das horizontale Mikroprogramm ab Adresse 10101 um den durch den Automaten definierten Befehl, indem Sie den korrespondierenden RT-Code und den Teil des Mikroprogramms ergänzen. Welche Funktionalität hat der Befehl?

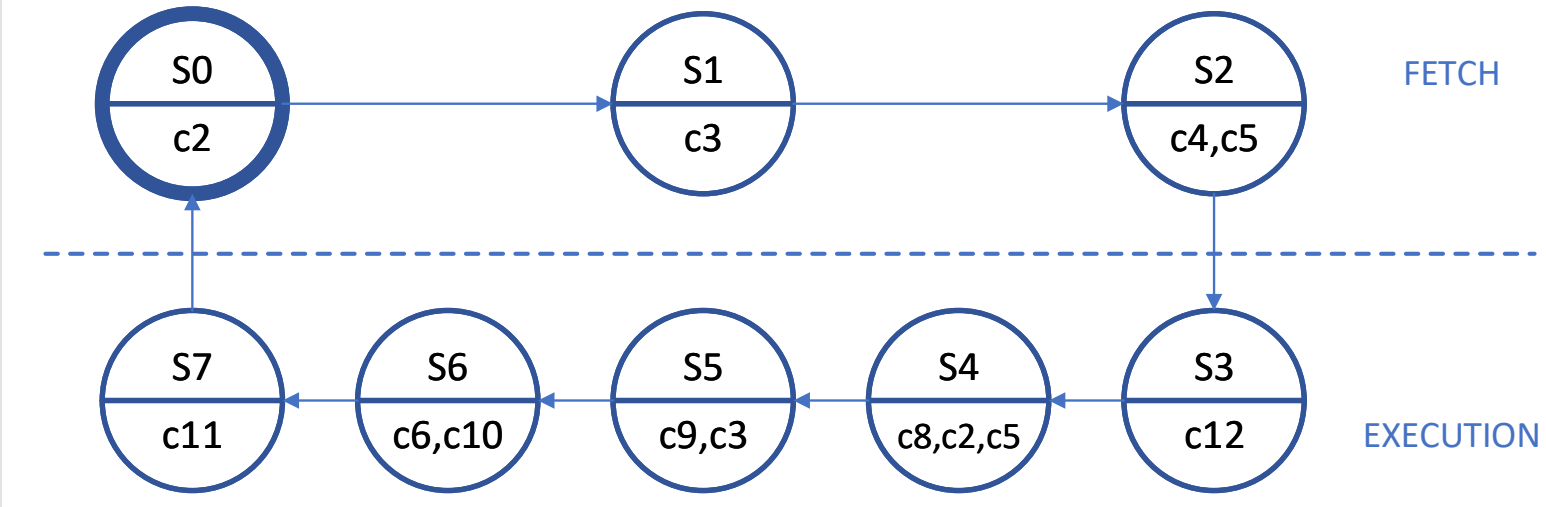


Abbildung 6: Automat für Befehl

$\Sigma_{A2} =$ /60 Punkte

Aufgabe 3: Allgemeine Fragen (7 Punkte)

a) Erläutern Sie den Unterschied zwischen einem Schaltnetz und einem Schaltwerk.

b) Beschreiben Sie je einen Vorteil festverdrahteter Steuerwerke gegenüber mikroprogrammierten Steuerwerken und umgekehrt.

c) Welche Arten von Signalen dienen zur Kommunikation zwischen Operationswerk und Steuerwerk?

d) Worin unterscheiden sich quasi-horizontales und vertikal Befehlsformat im Bereich der Mikroprogrammierung und welche Form hat bei gleicher Funktionalität die kürzere Ausführungszeit? Begründen Sie Ihre Antwort kurz.

$\Sigma_{A3} =$ /7 Punkte